Installing a custom Cloud Almond

1. **Web Almond only**：您為用戶提供登錄和使用 Almond 的 Web 服務。
2. **Web Almond + NLP**：置重用公共 Thingpedia，但使用自定義數據集和模型。
3. **fully custom Thingpedia**：新的 Thingpedia 要求您將代碼和憑據（客戶端 ID、API 密鑰）上傳到您關心的所有服務。您還必須部署自定義 NLP 模型，因為官方模型將不兼容。

- The basic form is to deploy **Web Almond only**. In this configuration, you provide a web service for users to sign in and use Almond. You can configure branding, login and customer feedback, and you are responsible to collect user data and store it safely. On the other hand, in this configuration you reuse both Thingpedia and the public NLP model from Almond. This is the simplest configuration, and the one we recommend to anyone who does not want to use our instance.
- The second form is to deploy **Web Almond + NLP**. This configuration reuses the public Thingpedia, but uses a customized dataset and model. Use this configuration if you want custom natural language support, including customized Genie templates, or a model that targets a subset of Thingpedia. You must run an NLP inference server, and you need to periodically synchronize your trained models to Thingpedia to reflect any changes (for example by polling the Thingpedia API). It is also recommended to deploy the NLP training server.
- The third form is to deploy a **fully custom Thingpedia**. This is **not** a recommended configuration, as it is significantly more challenging to manage. Furthermore, the new Thingpedia requires you to upload the code and credentials (client IDs, API keys) to all services you care about. You must also deploy custom NLP models, as the official ones will not be compatible. Use this setup only if you absolutely need custom Thingpedia interfaces, and cannot provide these interfaces on Thingpedia. This setup is also suitable for developing Thingpedia itself.

Step 1：Acquiring dependencies

Cloud Almond depends on :

- nodejs (>= 8.0) -->基本接配置 nodejs .10版
- gm (provided by GraphicsMagic)

Optionally, it depends on:

- libsystemd
- bubblewrap

A working MySQL server is also required. We recommend **MariaDB >= 10.2** for best compatibility.

For example, on Ubuntu (>= 18.04):

```
sudo apt install nodejs graphicsmagick libsystemd-dev bubblewrap -y
```

If you would like to run the MySQL server locally:

```
sudo apt install mariadb-server # Ubuntu
```

Finally, if you want to use custom NLP models, you must install `decanlp`:

and you must install almond-tokenizer.

```
pip3 install --user 'git+https://github.com/stanford-
oval/decaNLP.git#egg=decanlp'
```

Step 2: Installation

Cloud Almond can be installed using NPM. Run:

(The last command might need to run as root). This will install a command called `almond-cloud`.

```
git clone https://github.com/stanford-oval/almond-cloud
cd almond-cloud
npm install
npm link
```

Step 3: Configuration

You must choose which mode to operate as by editing the `config.js` file. You should create a copy and name it `/etc/almond-cloud/config.js`.

You must first set the `DATABASE_URL` to point to your MySQL server. The format is:

```
mysql://<user>:<password>@<hostname>/<db_name>?<options>
```

See the documentation of node-mysql for options.

You must also set `SECRET_KEY`, `JWT_SIGNING_KEY` and `AES_SECRET_KEY` appropriately. The server will refuse to start otherwise. See the configuration option reference for the format.

Further configuration depends on which mode of Cloud Almond you would like to use. If you're deploying Web Almond only, leave `NL_SERVER_URL`, `THINGPEDIA_URL` and `TRAINING_URL` to the default value.

If you're deploying custom NLP, set `NL_SERVER_URL` to the URL (scheme-host-port) of your NLP inference server, and `NL_SERVER_ADMIN_TOKEN` to a randomly generated token. The latter is used to authenticate administrative operations on the NLP inference server, such as reloading newly trained models. It is also recommended to deploy a training server, in which case you should set `TRAINING_URL` and `TRAINING_ACCESS_TOKEN`.

If you're deploying a custom Thingpedia, set `WITH_THINGPEDIA` to `embedded`, and `THINGPEDIA_URL` to `/thingpedia`.

If you use the embedded Thingpedia, it is expected you use a CloudFront+S3 to deliver code zip files, icons and other large user generated content. Set the URL of the CloudFront deployment for user-uploaded content as `CDN_HOST`, and change `FILE_STORAGE_BACKEND` to `s3`. If you do not use CloudFront, zip files and icons will be stored in the public/download folder of your code checkout, which must be writable.

Additional configuration options are also available. Consult the full reference for how to set the CDN used for website assets, OAuth redirection URLs, HTTP-to-HTTPs redirects, how to customize the website, and how to set emails appropriately. At the very least, it is expected you will change the `SERVER_ORIGIN` field to point to the correct location (scheme-host-port) of your Web Almond server, and change the `EMAIL_` fields to indicate that emails are sent from your website.

Step 3.5 (optional): Enable the sandbox

If you plan to deploy this as a web facing service, and you plan to allow developer users, you must enable the sandbox to prevent users from accessing each other's data.

Inside the sandbox, the code will have access to `/usr`, a whitelisted subset of `/etc`, and the directory containing the Web Almond installation. Make sure that these directories do not contain any private data. In particular, any sensitive data should be stored in `/etc` (including database passwords or access token), not in the current directory where you installed Web Almond. It is safe to store such data in `/etc/almond-cloud`, as that is excluded from the sandbox on purpose.

You must also make sure that the current working directory for Web Almond processes is not under `/usr`, `/etc`, or in the subtree where you cloned Web Almond. Common correct choices include `/srv/almond-cloud` and `/var/lib/almond-cloud`.

Note: If you skip this step, set `THINGENGINE_DISABLE_SANDBOX=1` in your environment.

Step 4: Database

To bootstrap Almond, execute:

```
almond-cloud bootstrap
```

This script will create the default `root` user, with password `rootroot`.

If you are using the embedded Thingpedia, the script will also populate the database with the builtin Thingpedia entries.

The script will also create a default `anonymous` user, with the same password as the root user. This enables users to try Web Almond without creating an account for themselves. Note that the default anonymous user is missing all service accounts, including those like YouTube that are advertised as suggestions to users.

You must set up those accounts before enabling the anonymous user in `config.js`. To do so, log in to the anonymous user as if it was regular user, and add the accounts to My Almond. It goes without saying, you should change the password for both the `root` and `anonymous` users, and you should use real, strong passwords.

Step 5: Web Almond

Web Almond is composed of a master process, and a number of worker processes. To start the master process, create a working directory, say `/srv/almond-cloud/workdir`, then do:

```
cd /srv/almond-cloud/workdir ; almond-cloud run-almond
```

Do not use a subdirectory of the code checkout for the Web Almond working directory, as that can create a security vulnerability.

The master process listens on the two sockets indicated in `config.js` as `THINGENGINE_MANAGER_ADDRESS`. You can use a path in the configuration file, a port, or host name/IP address with a port name. See node-sockaddr for the full range of options supported. If you use a TCP socket, you must set `THINGENGINE_MANAGER_AUTHENTICATION` as well, to prevent users from connecting to it from inside the developer sandboxes.

To scale horizontally, you can run multiple master processes, by setting multiple values to `THINGENGINE_MANAGER_ADDRESS` and passing `--shard <id>` to the master's commandline.

An example systemd unit file called `almond-cloud@.service` is provided. The unit file assumes the repository is located at `/opt/almond-cloud` and the local state directory is `/srv/almond-cloud`.

Step 6: the web frontend

You can run the web frontend in the same working directory as the master process, by saying:

```
almond-cloud run-frontend --port ...
```

If `--port` is unspecified, it defaults to 8080. Multiple frontend web services can be run on different ports, for load balancing. It is recommended to run at least two. An example systemd unit file is provided, called `almond-website@.service`.

If you run the frontend and master processes in different directories, make sure to use absolute paths in `THINGENGINE_MANAGER_ADDRESS`, or use TCP for communication.

Step 7 (optional): the NLP inference server

The NLP inference server must be run from a directory containing the trained models. The expected directory layout is as follows:

```
/default:en/{best.pth, config.json, thingpedia.json}
/default:zh/{best.pth, config.json, thingpedia.json}
/default:.../{best.pth, config.json, thingpedia.json}
...
```

You can acquire the pretrained models for testing from https://oval.cs.stanford.edu/releases/, our you can train your custom models using Genie.

You must also download the pretrained word embeddings, and set the `DECANLP_EMBEDDINGS` environment variable to the location where you downloaded. See tests/install-nlp-deps.sh for an example.

To run the NLP inference server do:

```
almond-cloud run-nlp --port ...
```

The port defaults to 8400. You can run multiple NLP inference servers on the same machine, but it is not recommended, as each server already makes use of all available cores, and multiple servers can easily exhaust available RAM. It is also not recommended to run this on the same machine as the frontend and and master processes. An example systemd unit file is provided, called `almond-nlp.service`.

The NLP inference server needs access to the database for typechecking, dataset access and to store any trained sentences. For security, it is recommended to use a different database user than the one used by the master and frontend processes.

Step 8 (optional): the NLP training server

You can automate training custom NLP models using the training server, which you can run as:

```
almond-cloud run-training
```

The current directly must be writable to the user running the server, and must be on disk with large amounts of space (at least 50 GBs free). The subdirectory `jobs` of the server current directory will contain a new entry for each trained job, and it is recommended to clean it periodically, for example with systemd-tmpfiles. An example systemd unit file is provided, called `almond-training.service`.

It is expected that the training server and the inference server run on different machines, and communicate over `rsync`/`ssh`. You must configure the Unix user running the training server to have SSH access to the inference server, and write access to the directory where the inference server is running. You can do for example with the following SSH `authorized_keys` entry:

```
command="/usr/local/bin/rrsync -wo /var/lib/almond-cloud/nlp",restrict ssh-rsa
AAAA...
```