# Instruction

## Introduction

Cloud robotics augments robots by cloud services. Although it is regarded as a revolutionary paradigm, great efforts have to be made to develop cloud services dedicated to robotic scenarios. Instead of developing those services from scratch, we can find that many robotic algorithms have been encapsulated into software packages of special forms (e.g. ROS packages) in the open source community. Although they are not designed for cloud environment originally, could they be rapidly encapsulated and deployed as cloud services? This project aims to address this challenge by developing a Platform-As-A-Service framework which can directly deploy those software packages. The packages can be transformed into Internet-accessible cloud services automatically. Another requirement is that multiple robots should be able to access such a cloud service simultaneously even if the corresponding package is originally designed just to serve a single robot, with the guarantee of Quality of Service for each robot. Docker or other open-source application container technology can be adopted to realize this goal at the backend.

## Cloudroid server

### Environment

- Ubuntu 14.04.3 LTS
- Python 2.7.6
- Docker version 17.03.1-ce (Does not support a lower version)

### Installation

#### Docker installation

(1) Follow the tutorial to get Docker installed:
https://docs.docker.com/engine/installation/linux/ubuntulinux/

#### Flask installation

(1) Log into your machine as a user with sudo or root privileges.
(2) Open a terminal window.
(3) Update package information:

```
$ sudo apt-get update
```

(4) Install python-tk:

```
$ sudo apt-get install python-tk
```

(5) Install python-pip:

```
$ sudo apt-get install python-pip
```

(6) Install flask:

```
$ sudo pip install flask
```

(7) Install database:

```
$ sudo pip install flask-sqlalchemy
```

(8) Install flask_login

```
$ sudo pip install flask-login
```

(9) Install flask-wtf

```
$ sudo pip install flask-wtf
```

### Docker-py installation

```
$ sudo pip install docker
```

Note: If you have installed docker-py, you must uninstall it:

```
$ sudo pip uninstall docker-py
```

### Others installation

(1) Install zip and unzip:

```
$ sudo apt-get install zip unzip
```

## Other Preparation

### Docker registry preparation

(1) Docker registry running

```
$ sudo docker run -d -p 5000:5000 --restart=always --name
registry -v 'pwd'/data:/var/lib/registry registry:2
```

(2) Docker registry configuration

- Open the file ./Cloudroid/app/dockerops.py with your favorite editor and change the registry IP address with your IP. (e.g. registry = ' myregistry.example.com:5000')
- Create or modify /etc/docker/daemon.json:

  ```
  {"insecure-registries":["myregistry.example.com:5000"]}
  ```

  Restart docker daemon:

  ```
  $ sudo service docker restart
  ```

**Docker swarm preparation**

(1) First you should initialize a docker swarm:

```
$ sudo docker swarm init
```

## Cloudroid server start

(1) Download and unzip the Cloudroid package.

Or git clone the package:

```
$ git clone https://github.com/xiteen/Cloudroid
```

(2) Prepare the base image:

1) Find the dockerfile in the folder: (the path of Cloudroid)/Cloudroid/base-image and build it.

```
$ cd ../Cloudroid/base-image
$ sudo docker build .
```

2) Get the id of the new created image:

```
$ docker images
```

3) Rename the image:

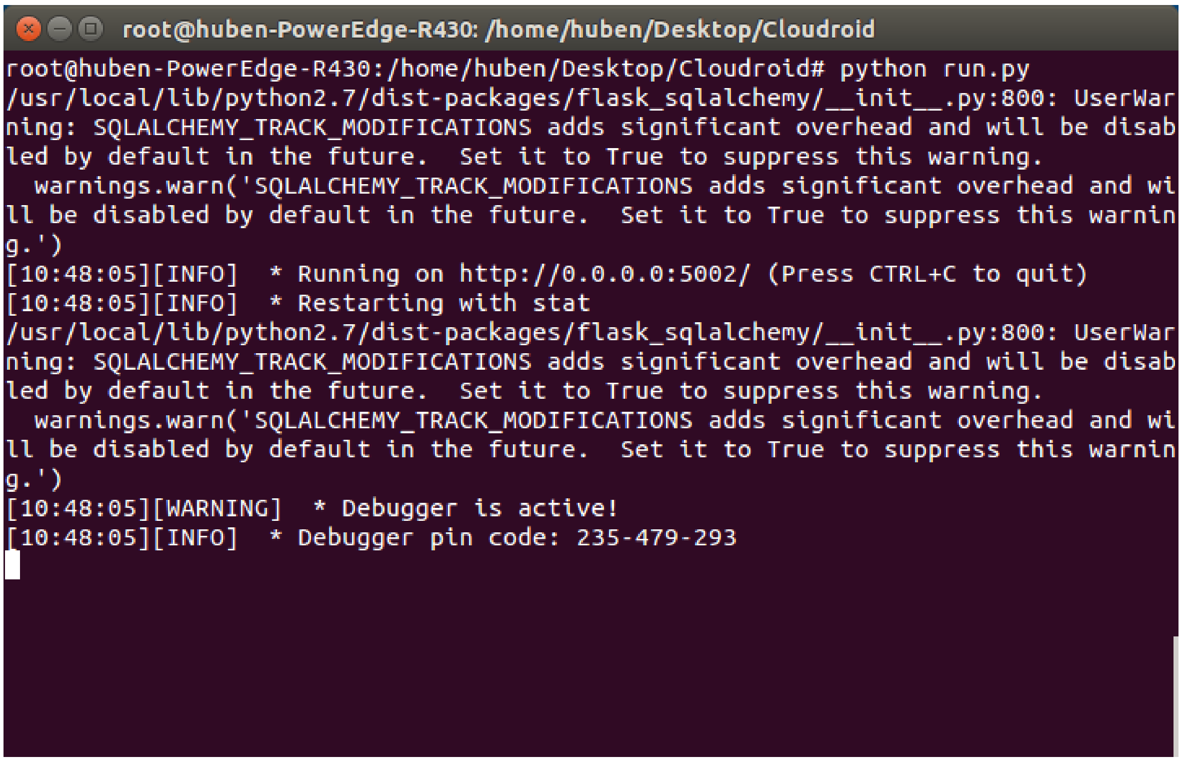
```
$ sudo docker tag <id> ros:my
```

(3) Start the micROS- cloud server with the command:

```
$ cd ../Cloudroid
$ python run.py
```

Note: If there is an error: Unable to list the containers. Reason: ('Connection aborted.', error(13, 'Permission denied')). Please run the command with the system permissions.

For example: The running of the server.



```
root@huben-PowerEdge-R430: /home/huben/Desktop/Cloudroid
root@huben-PowerEdge-R430:/home/huben/Desktop/Cloudroid# python run.py
/usr/local/lib/python2.7/dist-packages/flask_sqlalchemy/__init__.py:800: UserWar
ning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disab
led by default in the future.  Set it to True to suppress this warning.
  warnings.warn('SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and wi
ll be disabled by default in the future.  Set it to True to suppress this warnin
g.')
[10:48:05][INFO]  * Running on http://0.0.0.0:5002/ (Press CTRL+C to quit)
[10:48:05][INFO]  * Restarting with stat
/usr/local/lib/python2.7/dist-packages/flask_sqlalchemy/__init__.py:800: UserWar
ning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disab
led by default in the future.  Set it to True to suppress this warning.
  warnings.warn('SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and wi
ll be disabled by default in the future.  Set it to True to suppress this warnin
g.')
[10:48:05][WARNING]  * Debugger is active!
[10:48:05][INFO]  * Debugger pin code: 235-479-293
```

## Function introduction

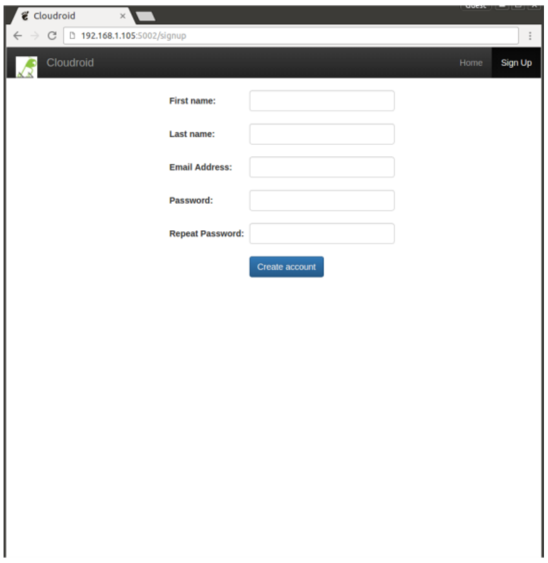
Open the browser and enter the IP address.

    <server ip address>:5002

For example: localhost:5002

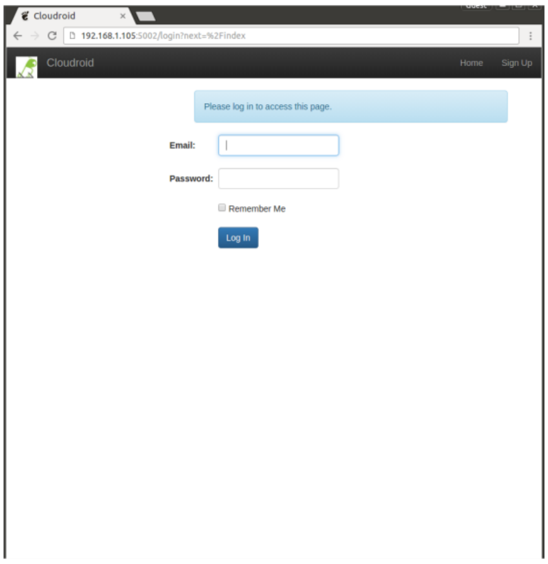
### Registration and login

Click the Sign Up button to create an account and then login:



Cloudroid
192.168.1.105:5002/signup
Cloudroid Home Sign Up

First name:
Last name:
Email Address:
Password:
Repeat Password:
Create account

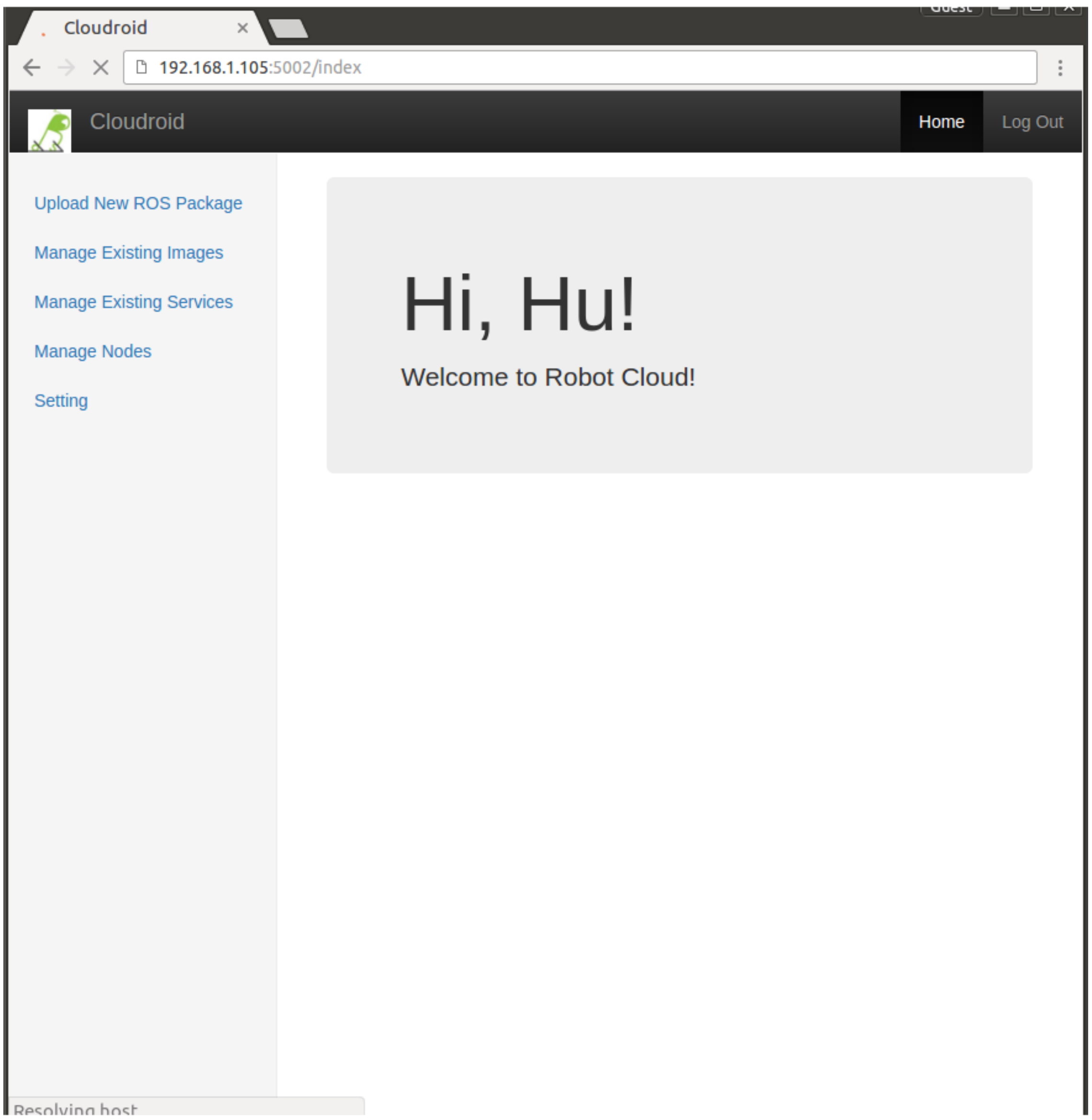


Cloudroid
192.168.1.105:5002/login?next=%2Findex
Cloudroid Home Sign Up

Please log in to access this page.

Email:
Password:
Remember Me
Log In

After login, you will see the welcome page:

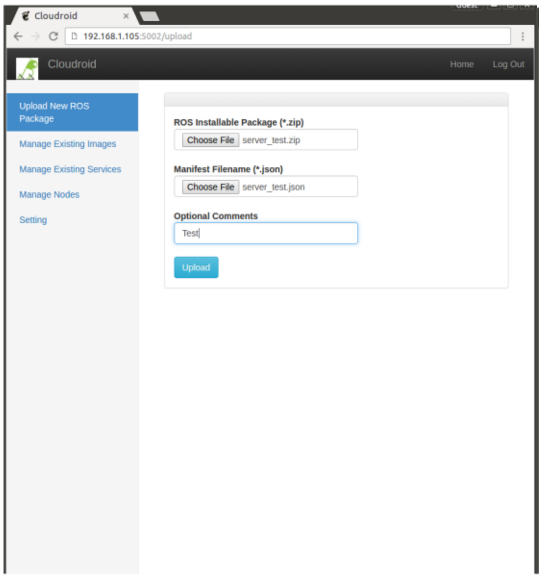


Hi, Hu!

Welcome to Robot Cloud!

## Upload ROS package

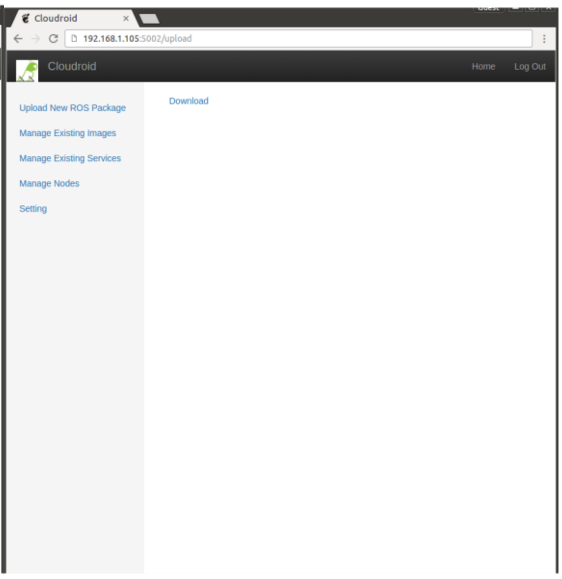
In the *Upload New ROS Package* page, you can upload a ROS package and a json file to define a cloud service. After uploading successfully, you can download a proxy for you robot to access the service.
For example, you can use the ROS package *server_test.zip* and json file *server_test.json* in the */Cloudroid/test-case/uploadfile* folder.



ROS Installable Package (*.zip): Choose File server_test.zip

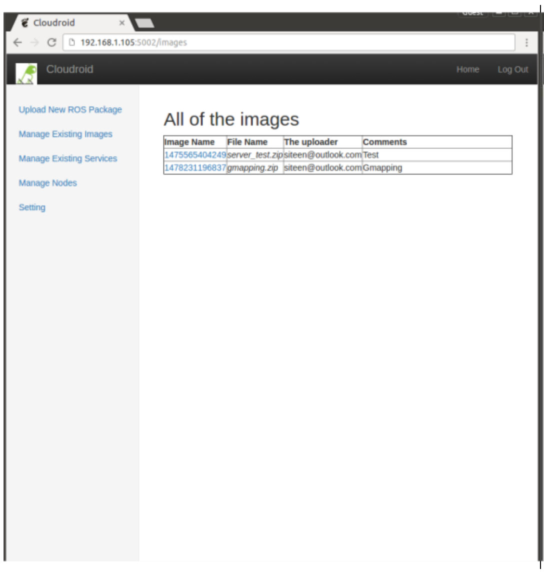
Manifest Filename (*.json): Choose File server_test.json

Optional Comments: Test



## Manage the existing images

In the *Manage Existing Images* page, you can get the information of every specific docker image.



All of the images

| Image Name | File Name | The uploader | Comments |
|---|---|---|---|
| 1475565404249 | *server_test.zip* | siteen@outlook.com | Test |
| 1478231196837 | *gmapping.zip* | siteen@outlook.com | Gmapping |

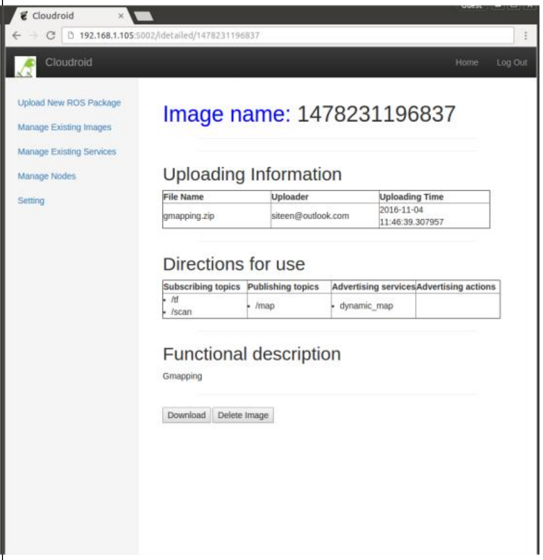


Image name: 1478231196837

Uploading Information

| File Name | Uploader | Uploading Time |
|---|---|---|
| gmapping.zip | siteen@outlook.com | 2016-11-04 11:46:39.307957 |

Directions for use

| Subscribing topics | Publishing topics | Advertising services | Advertising actions |
|---|---|---|---|
| • /tf<br>• /scan | • /map | • dynamic_map | |

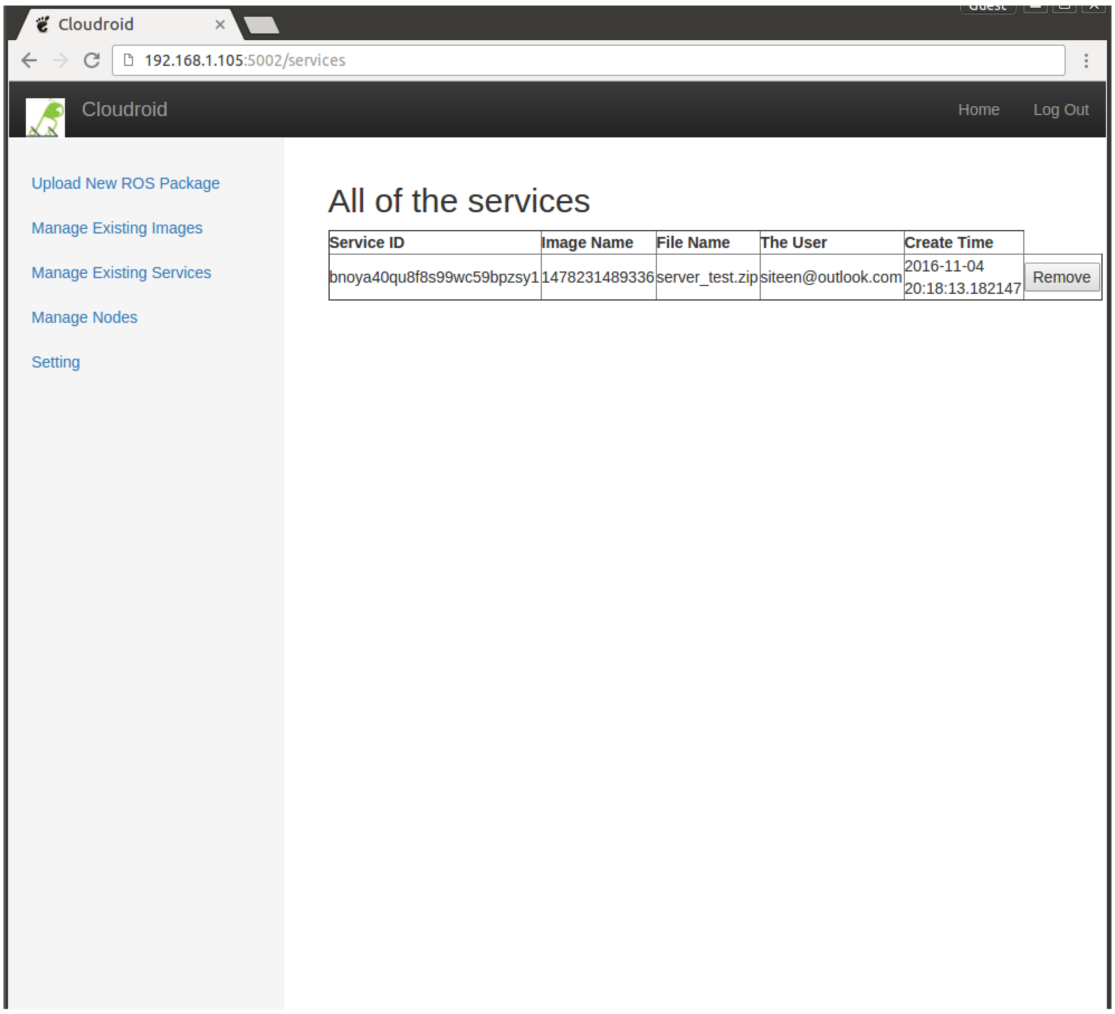
Functional description

Gmapping

Click the image name and you can get the details of the image. In this page you can also download the proxy or delete the image.

## Manage the existing services

In the *Manage Existing Services* page, you can get the information of services and you can remove a service if you want.

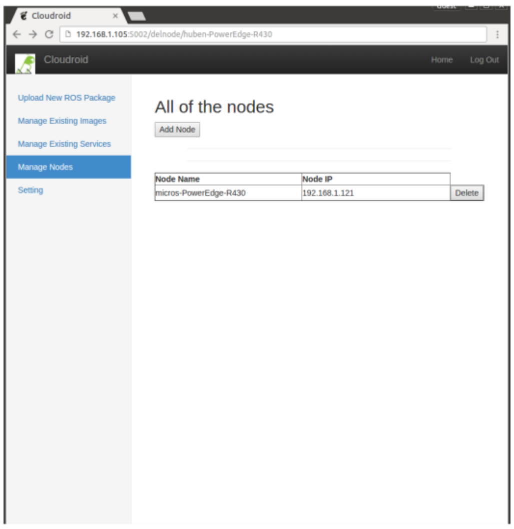


All of the services

| Service ID | Image Name | File Name | The User | Create Time | |
|---|---|---|---|---|---|
| bnoya40qu8f8s99wc59bpzsy1 | 1478231489336 | server_test.zip | siteen@outlook.com | 2016-11-04 20:18:13.182147 | Remove |

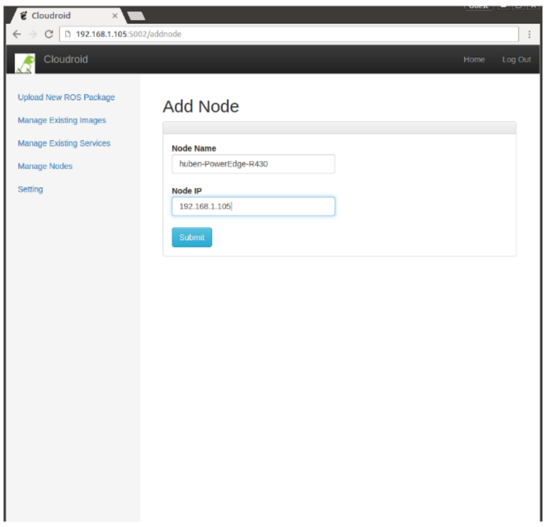
## Manage the nodes

In the *Manage Nodes* page, you can get the information of nodes about the node name and node IP, and you must add all the nodes in the docker swarm to this list.



All of the nodes

| Node Name | Node IP | |
|---|---|---|
| micros-PowerEdge-R430 | 192.168.1.121 | Delete |



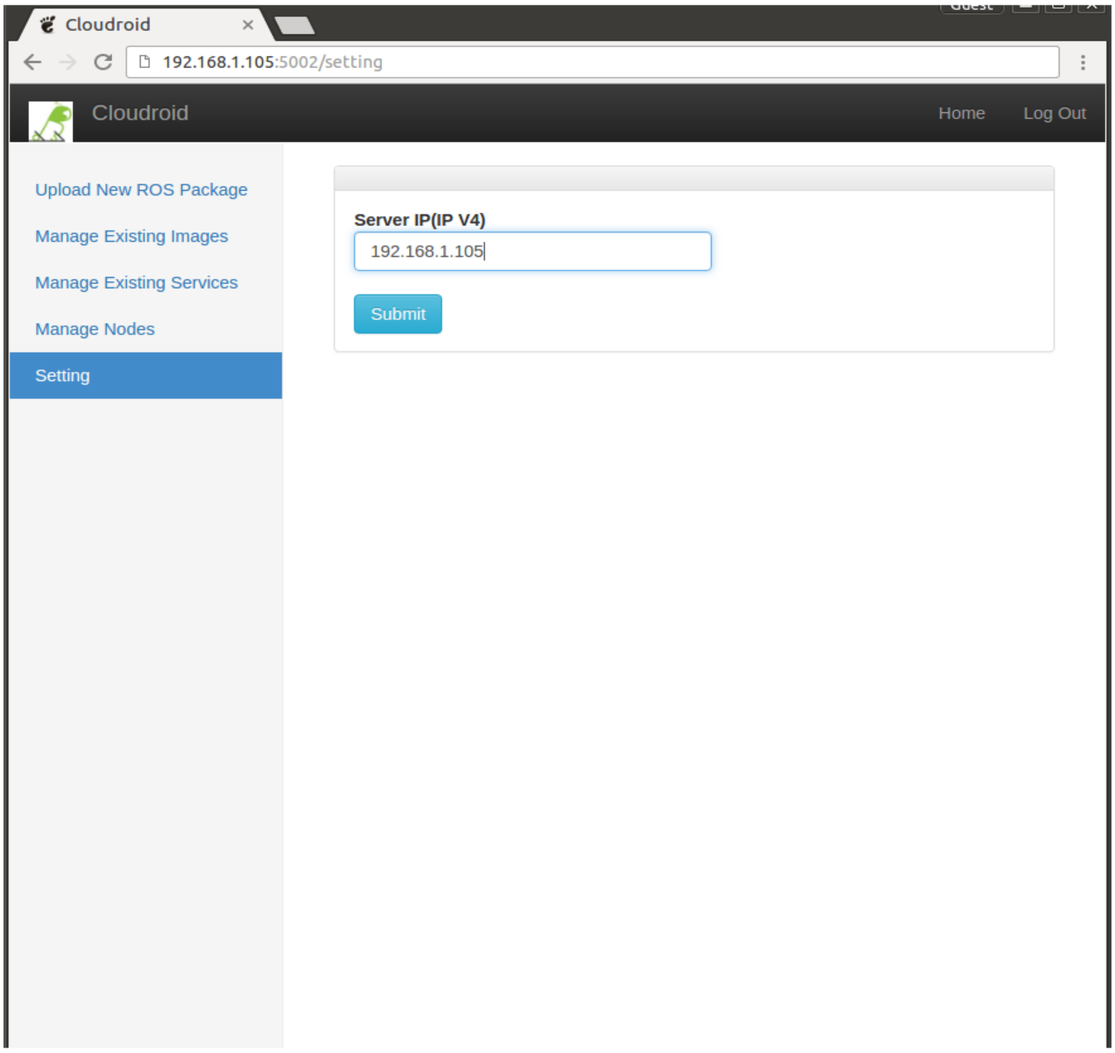
Add Node

Node Name
huben-PowerEdge-R430

Node IP
192.168.1.105

**Setting**

In the *Setting* page, you can set the server ip address for a remote robot to connect to the service. Please set this firstly.



Server IP(IP V4)

192.168.1.105

# Cloudroid client

## Environment

- Python 2.7.6
- ROS indigo

# Installation

## ROS installation

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

(1) Follow the tutorial to install the ROS: http://wiki.ros.org/indigo/Installation/Ubuntu

(2) Create a ROS Workspace:
http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

## Cloudroid client start

After upload the ROS package you can get the proxy package.

(1) Download and unzip the proxy package.

(2) Copy the package to the ~/catkin_ws/src

(3) Open the terminal and make the package

```
$ cd catkin_ws
$ catkin_make
```

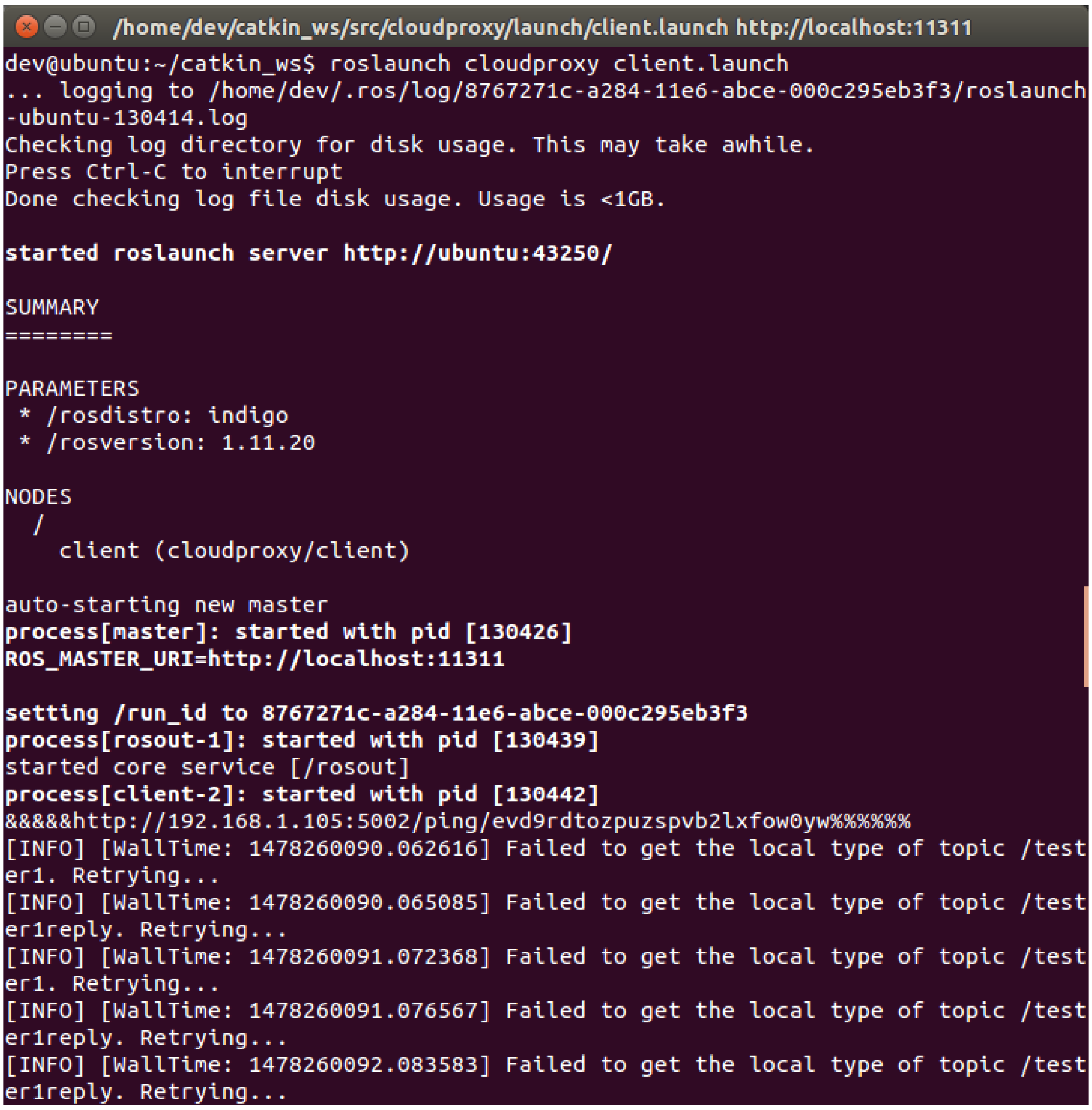
(4) Run the proxy:

```
$ roslaunch cloudproxy client.launch
```

(5) After the proxy is running, you can run the other ROS node to publish the topics that the remote running server required.

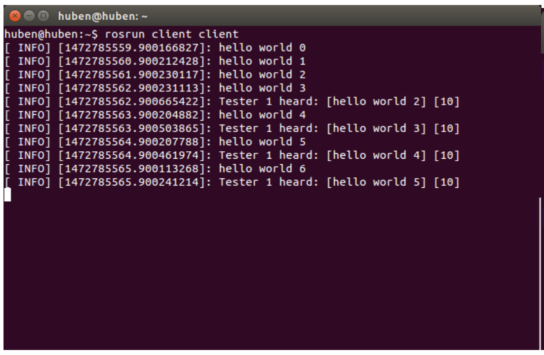
**For example:** As the example showed before, you have upload the ROS package *server_test.zip* and json file *server_test.json* which are in the folder Cloudroid/test-cast/uploadfile. If upload successfully, you can download a zip package named with a string of 13 numbers. Deal with the zip package as described before and then run the proxy.

Run the proxy:

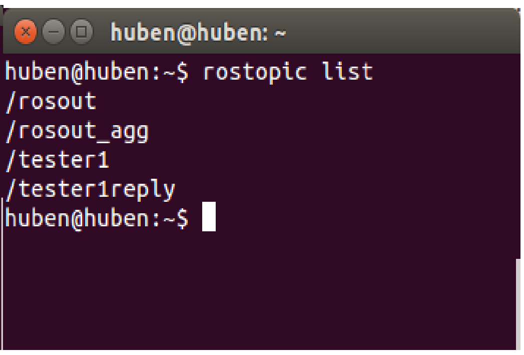


```
/home/dev/catkin_ws/src/cloudproxy/launch/client.launch http://localhost:11311
dev@ubuntu:~/catkin_ws$ roslaunch cloudproxy client.launch
... logging to /home/dev/.ros/log/8767271c-a284-11e6-abce-000c295eb3f3/roslaunch
-ubuntu-130414.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:43250/

SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.20

NODES
  /
    client (cloudproxy/client)

auto-starting new master
process[master]: started with pid [130426]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 8767271c-a284-11e6-abce-000c295eb3f3
process[rosout-1]: started with pid [130439]
started core service [/rosout]
process[client-2]: started with pid [130442]
&&&&&http://192.168.1.105:5002/ping/evd9rdtozpuzspvb2lxfow0yw%%%%%%
[INFO] [WallTime: 1478260090.062616] Failed to get the local type of topic /test
er1. Retrying...
[INFO] [WallTime: 1478260090.065085] Failed to get the local type of topic /test
er1reply. Retrying...
[INFO] [WallTime: 1478260091.072368] Failed to get the local type of topic /test
er1. Retrying...
[INFO] [WallTime: 1478260091.076567] Failed to get the local type of topic /test
er1reply. Retrying...
[INFO] [WallTime: 1478260092.083583] Failed to get the local type of topic /test
er1reply. Retrying...
```

After the proxy is running, it is waiting for the necessary topics from the local. Start the ROS node *client* (You can find it in /Cloudroid/test-case/local, copy the package *src-test.zip* to catkin_ws and unzip the package and then *catkin_make* them) to publish the topic that the proxy get it and send to the server.



```
huben@huben: ~
huben@huben:~$ rosrun client client
[ INFO] [1472785559.900166827]: hello world 0
[ INFO] [1472785560.900212428]: hello world 1
[ INFO] [1472785561.900230117]: hello world 2
[ INFO] [1472785562.900231113]: hello world 3
[ INFO] [1472785562.900665422]: Tester 1 heard: [hello world 2] [10]
[ INFO] [1472785563.900204882]: hello world 4
[ INFO] [1472785563.900503865]: Tester 1 heard: [hello world 3] [10]
[ INFO] [1472785564.900207788]: hello world 5
[ INFO] [1472785564.900461974]: Tester 1 heard: [hello world 4] [10]
[ INFO] [1472785565.900113268]: hello world 6
[ INFO] [1472785565.900241214]: Tester 1 heard: [hello world 5] [10]
```



```
huben@huben: ~
huben@huben:~$ rostopic list
/rosout
/rosout_agg
/tester1
/tester1reply
huben@huben:~$
```

As you can see, the node client published the node /tester1 and received topic /tester1reply what is published by server and get from the server by proxy.

# Upload file preparation

We need two files to upload: install.zip and a json file.

## Environment

- Python 2.7.6
- ROS indigo

## *install.zip* preparation Steps

1. Prepare a catkin workspace

If you are the first time to use the catkin workspace of ROS, please read the web to build it: http://wiki.ros.org/catkin/Tutorials/create_a_workspace

If you have a catkin workspace to use, please preserve your files and remove the workspace and then build a new one to avoid the existing files disturbing.

2. Prepare the ROS package

For example: gmapping

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ros-perception/slam_gmapping.git
```

3. Build the ROS package

```
$ cd ~/catkin_ws
$ catkin_make
```

4. Copy the necessary files to the specific folder: *install*

```
$ mv ~/catkin_ws/build ~/install
$ mv ~/catkin_ws/devel ~/install
$ mv ~/catkin_ws/src ~/install
```

5. Compress the *install* folder as a zip bag

## Format of json file

This is an example of the format of the json file.

```
{
"subscribed_topics": ["/tester1"],
"published_topics": ["/tester1reply"],
"advertised_services": [],
"advertised_actions": [],
"start_cmds": ["rosrun server_test server"],
"mem_limit": "1073741824",
"memswap_limit" : "1073741824",
"cpushares":"100",
"cpusetcpus":"0,1"
}
```

subscribed_topics: The list of topics that the server need  to subscribe.
published_topics:  The list of topics that the server need to  publish.
advertised_services:  The list of  services  that the server need.
advertised_actions  The list of services that the server need.
start_cmds: The command for the container to run to start the service.
mem_limit:  The  memory limit for image  build.  (The  minimum  of the memory size is 4M, so the
number  must be  not less than  4194304)
memswap_limit:  Total memory (memory + swap),  -1 to disable swap.
cpushares:  CPU shares (relative weight).
cpusetcpus:  CPUs in which to allow execution, e.g., "0-3", "0,1"