Autocloud Documentation

Release 0.1

Kushal Das

Mar 28, 2018

Contents

Autocloud is part of Fedora Project where we are using it to test Fedora Cloud image build on Koji. It listens to fedmsg, and pushes new messages on the status of the tests.

Contents:

CHAPTER 1

How to build Autocloud for development?

Autocloud is written in Python2. We will port it to Python3 in future, but for now it is on Python2 only. First you can git clone the repo from github.

```
$ git clone https://github.com/kushaldas/autocloud.git
```

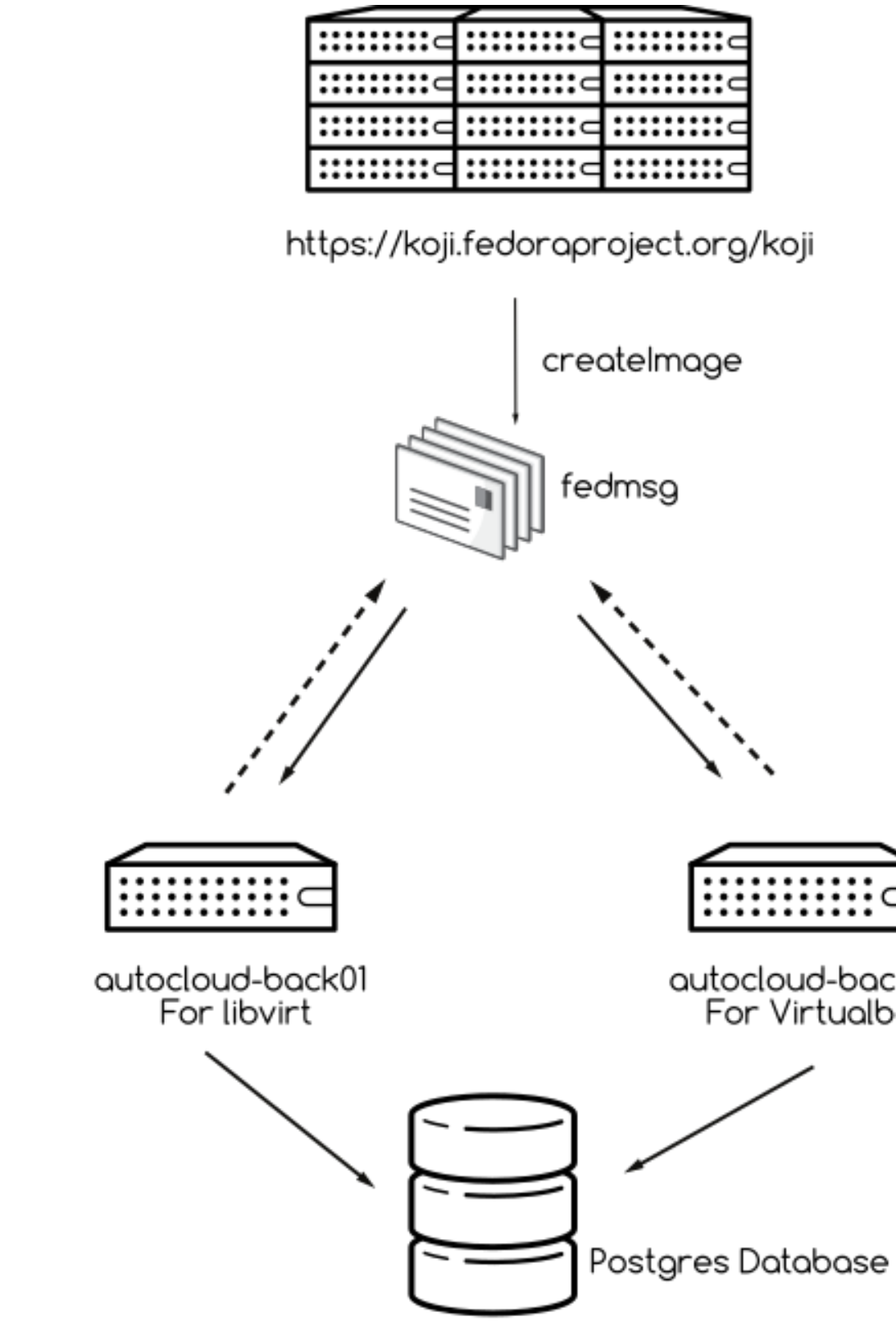
You should also install vagrant-libvirt from Fedora repo.

```
$ sudo dnf install vagrant-libvirt -y
```

CHAPTER 2

Setup instruction on Fedora

The following image explains the deployment plan of autocloud.


https://koji.fedoraproject.org/koji
createImage
fedmsg
autocloud-back01
For libvirt
autocloud-back02
For Virtualbox
Postgres Database
autocloud-web01
autocloud-web02

We have two bare metal server autocloud-back01, and autocloud-back02, the later one is only used for vagrant-virtualbox based images. We also have two load balanced vms running the web frontend.

2.1 Install the autocloud-common package in all systems

```
$ sudo dnf install autolcoud-common
```

The above command will install the latest package from the repo. You may want to install vagrant-libvirt if you will execute libvirt based tests on the system.

2.2 Install autocloud-backend package on both the autocloud-back0* systems

```
$ sudo dnf install autocloud-backend
```

2.3 Start the redis server in both autocloud-back0* systems

```
$ sudo systemctl start redis
```

2.4 Enable ports for tunir in both autocloud-back0* systems

Autocloud uses tunir to execute the tests on a given image. We will have to do the follow setup for tunir to execute in a proper way.

```
$ python /usr/share/tunir/createports.py
```

2.5 Enable kill_vagrant command in cron job

Enable a cron job which will run */usr/sbin/kill_vagrant* in every 10 minutes (or an hour). This is required as many vagrant images do not work, and boot_timeout never works with vagrant-libvirt.

Note: This is a workaround which is required for now (2015-09-29). But may get removed in future.

2.6 Configure the database URI in all systems

In */etc/autocloud/autocloud.cfg* file please configure the sqlalchemy uri value. For our work, we are using postgres as database.

2.7 Create the tables in the database

Note: This has to be done only once from autocloud-back01 system

```
$ python /usr/share/autocloud/createdb.py
```

2.8 Install vagrant-libvirt on autocloud-back01

This is the system to handle all libvirt tasks, so we will have to install vagrant-libvirt on this system.

```
$ sudo dnf install vagrant-libvirt
```

2.9 Configure for the vagrant-virtualbox jobs in autocloud-back02

In */etc/autocloud/autocloud.cfg* file set *virtualbox* value to True. If you want to know how to setup virtualbox on the system, please refer to this guide.

2.10 Configure the correct tunir job deatils

We need the exact commands/job details for tunir. This is a configuration file so that we can update it whenever required.

```
$ sudo wget https://raw.githubusercontent.com/kushaldas/tunirtests/master/fedora.txt -
↪O /etc/autocloud/fedora.txt
```

2.11 Start fedmsg-hub service in autocloud-back0* systems

This service listens for new koji builds, and creates the database entry and corresponding task in the queue.

```
$ sudo systemctl start fedmsg-hub
```

2.12 Start autocloud service in autocloud-back0* systems

This service will listen for new task in the queue, and execute the tasks.

```
$ sudo systemctl start autocloud
```

2.13 Starting the web dashboard in autocloud-web0* systems

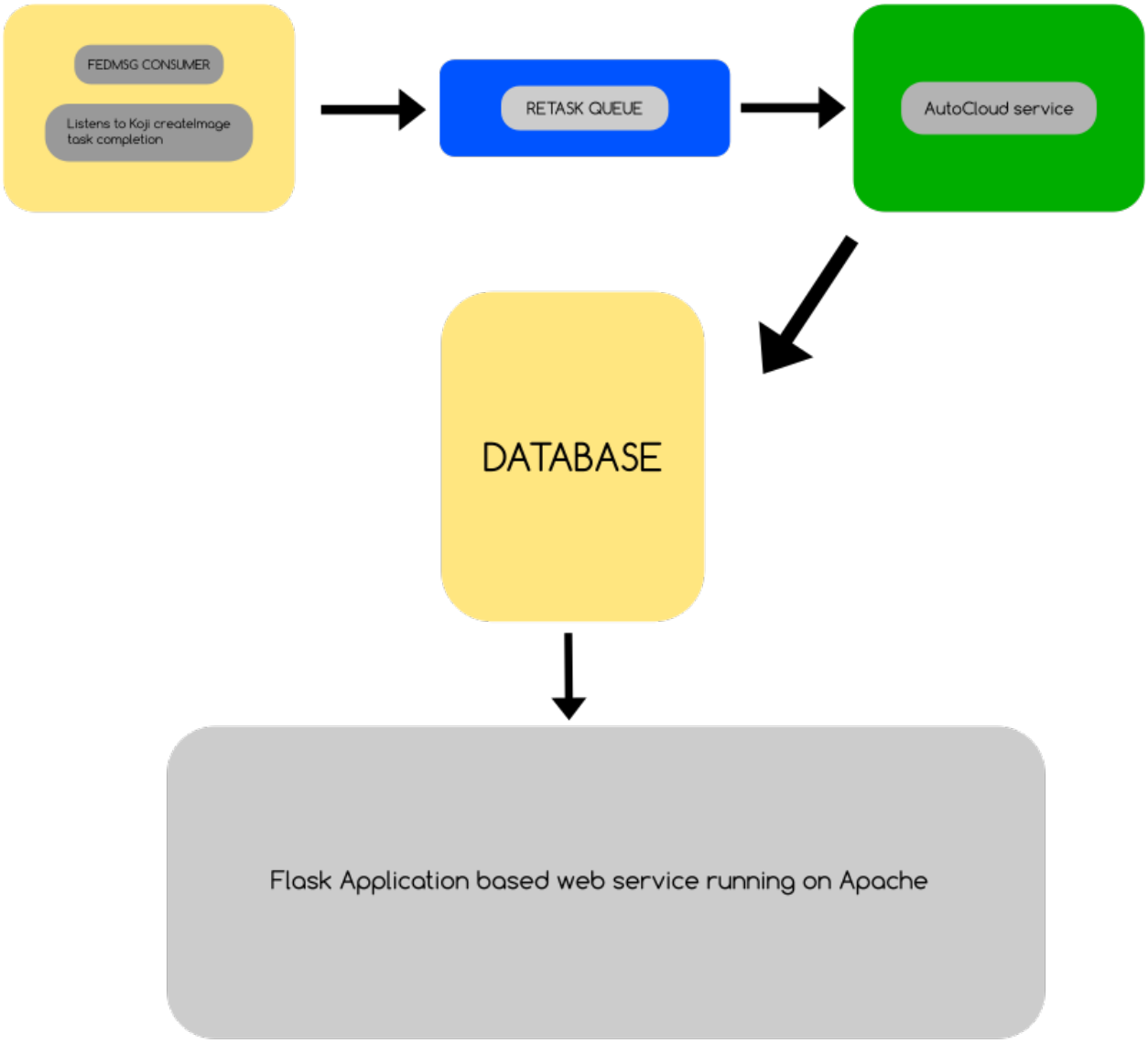
This is the web dashboard for the Autocloud, we use httpd for the this.

```
$ sudo systemctl start httpd
```

CHAPTER 3

Design of the system

Autocloud helps to automated testing of the Fedora Cloud images, and vagrant images produced in the koji. It uses fedmsg to keep listening to new builds (createImage) task, and when a new image is available it enqueues it to it's own job queue, and then finally execute the latest tests on the image. It can find out if the image is qcow2 based cloud image, or a vagrant image, and calls tunir to do the real testing. The following image gives a basic idea about the flow of the steps in autocloud.



3.1 Second system to test virtualbox based vagrant images

We need a second system which only tests the vagrant-virtualbox images. This has to be done in a separate system as Virtualbox can not co-exists with kvm. So, the primary system does all the work related to any libvirt based image (may be a qcow2, or a box file). The secondary system only listens for vagrant-virtualbox based images, and tests those vagrant images.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search