Bootstrapping servers with cloud-init

John Leach

Brightbox

February 2013

Bootstrapping servers

- Customise image before server build
  - Partitions, filesystems etc.
  - Where do you do the build securely?
  - Big, slow
  - Immature toolkit
    - Debian live-build
    - Guestfish
- Customise image after server build
  - On boot, in the virtual machine
  - Small, fast, re-useable
  - Ubuntu automates customisation with cloud-init

What is cloud-init?

- early initialisation of a cloud instance
  - runs early in boot process
- installed in generic Ubuntu Cloud Images (and Brightbox, EC2 images)
- Python, Upstart

What is cloud-init?

```
$ cat /etc/rc.local
curl http://example.com/script.sh | sh
```

The basics

- retrieves metadata in various ways
- sets default locale
- sets hostname
- resizes filesystem to fill partition
- generates sshd private ssh keys
- installs public ssh keys for login

Metadata and configuration

- you (or your cloud) provide the metadata
- cloud-init retrieves the metadata
- plugins read the metadata and do things

DataSource

- get_hostname
- get_instance_id
- get_locale
- get_public_ssh_keys
- get_userdata

EC2 DataSource

- http://169.254.169.254/latest

```
$ curl http://169.254.169.254/latest/meta-data/instance-id
srv-hd3iu
$ curl http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
ssh-dss AAAAB3NzaC1k...QL+ecQ2nNNU3pI8= public key
$ curl http://169.254.169.254/latest/user-data
arbitrary data you can provide, up to 16k in size.
```

CloudStack DataSource

- http://<default-gateway-ip>/latest/

MAAS DataSource

- MAAS is part of Ubuntu's "Orchestra" server management system
- Physical server support
- Retrieves from a url using oauth tokens (via a seed dir)

NoCloud DataSource

- Retrieves metadata from local filesystem
  - You have to get the metadata in there somehow
- Retrieves metadata via an iso mounted as a CDROM
  - cloud-localds tool for creating the iso

```
## create the iso disk with NoCloud data on it.
$ cloud-localds my-seed.img my-user-data.txt

## Boot a kvm
$ kvm -hda disk.img -hdb my-seed.img
```

Providing metadata with Brightbox

```
$ brightbox-servers create --user-data="Hello World" img-mvunm
Creating a nano server with image Ubuntu Precise with 0.02k of user data
 id          status    type  zone   created_on  image_id   cloud_ip_ids  name
-------------------------------------------------------------------------------
 srv-6uo7o   creating  nano  gb1-a  2013-02-20  img-mvunm
-------------------------------------------------------------------------------

$ ssh ubuntu@ipv6.srv-6uo7o.gb1.brightbox.com
ubuntu@srv-6uo7o:~$ curl http://169.254.169.254/latest/meta-data/instance-id
srv-6uo7ou

ubuntu@srv-6uo7o:~$ curl http://169.254.169.254/latest/user-data
Hello World
```

simple shell script as user data

```
cat <<EOF config.txt
#!/bin/sh
echo "I'm running on boot"
echo "I'm basically /etc/rc.local"
EOF
```

```
$ brightbox-servers create --user-data=config.txt img-mvunm
```

users and passwords

```
#cloud-config
chpasswd:
  ssh_pwauth: false
  list: |
    ubuntu:mysecret
    root:RANDOM
```

write files

```
#cloud-config
write_files:
  content: Hello World
  path: /etc/motd
  perm: '0644'
```

ssh key installation from launchpad

- https://launchpad.net/~ johnleach/+sshkeys

```
#cloud-config
user: root
ssh_import_id: johnleach
```

ssh configuration

```
#cloud-init
ssh_deletekeys: false
disable_root: true
```

install packages with apt

```
#cloud-config
apt_sources:
 - source: "ppa:brightbox/ruby-ng"
apt_update: true
apt_upgrade: true
packages:
  - ruby1.9
  - nginx-full
```

rsyslog

```
#cloud-config
rsyslog:
  - "*.* @@10.55.66.77"
```

phone home

```
#cloud-init
phone_home:
  url: http://example.com/callback
  tries: 3
```

combine them

```
cat <<EOF cloudconfig.txt
#cloud-config
user: root
ssh_import_id: johnleach
apt_sources:
 - source: "ppa:brightbox/ruby-ng"
apt_update: true
apt_upgrade: true
packages:
  - ruby1.9
  - nginx-full
rsyslog:
  - "*.* @@10.55.66.77"
EOF

$ brightbox-servers create --user-data=cloudconfig.txt img-mvunm
```

shit puppet

So cloud-init is a shit Puppet

puppet

```
#cloud-init
  conf:
    agent:
      server: "puppetmaster.example.com"
      ca_cert: ...
```

chef

```
#cloud-init
chef:
  server_url: "https://chef.example.com:4000"
  environment: production
  run_list:
    - "recipe[apache2]"
```

mcollective

```
#cloud-config
mcollective:
  conf:
    plugin.stomp.host: 10.88.44.33
    public-cert: ...
    private-cert: ...
```

Salt

```
#cloud-config
salt_minion:
  conf:
    master: 10.88.44.33
  public_key: ...
  private_key: ...
```

puppetapply code

```
def handle(_name, cfg, cloud, log, _args):
    if 'puppetapply' not in cfg:
        return

    puppet_cfg = cfg['puppetapply']

    cc.install_packages(("puppet",))

    puppet_data_dir = tempfile.mkdtemp('cloud-init-puppetapply')
    manifests_dir = puppet_data_dir + '/manifests'

    if 'modules_git_url' in puppet_cfg:
        cc.install_packages(("git",))
        cmd = ['git', 'clone', puppet_cfg['modules_git_url'], puppet_data_dir]
        subprocess.check_call(cmd)

    if 'manifest' in puppet_cfg:
        if not os.path.exists(manifests_dir):
            os.makedirs(manifests_dir)
        manifest_fh = open(manifests_dir + '/site.pp', 'w')
        manifest_fh.write(puppet_cfg['manifest'])
        manifest_fh.close()

    # Apply the manifests using puppet
    cmd = ['puppet', 'apply', '--confdir='+puppet_data_dir, manifests_dir+'/site.pp
    subprocess.check_call(cmd)
```

puppetapply example

```
#cloud-init
puppetapply:
  modules_git_url: https://github.com/brightbox/puppet.git
  manifest: |
    include apt
    include apache
    class { "elasticsearch":
      minimum_master_nodes => 2,
      discovery_hosts => ["srv-aaaaa", "srv-bbbbb"]
    }
```

user-data formatting

- base64 encoding
- gzip
- multi-part archive
  - write-mime-multipart tool
- include file

```
#include https://raw.github.com/gist/3129203/puppet-git-receiver-install
```