cloud-init

Thomas Merkel tm@core.io

2020-10-20

Initialisieren von Cloud Instanzen am Beispiel von **SmartOS**

lx-branded Zones (und VMs)

Agenda

- whoami
- cloud-init
  - Was ist cloud-init?
  - Was kann cloud-init?
  - Data Source / Quelle
  - Meta Data Beispiel
  - user-data Format
    - Cloud Config Module und Beispiele
- cloud-init unter SmartOS
  - Übersicht und Status
  - Customer Metadata
  - Workaround
- Debugging
- Beispiele und Demo
- Fazit und Fragen

~tm

- Server Ninja / Partner bei SkyLime GmbH
- Hosting und Cloud Lösungen auf SmartOS unter reco-systems
- Unix und Linux Consultant seit ~15 Jahren
- Fokus auf DevOps, Automatisierung, Infrastruktur und “Cloud”

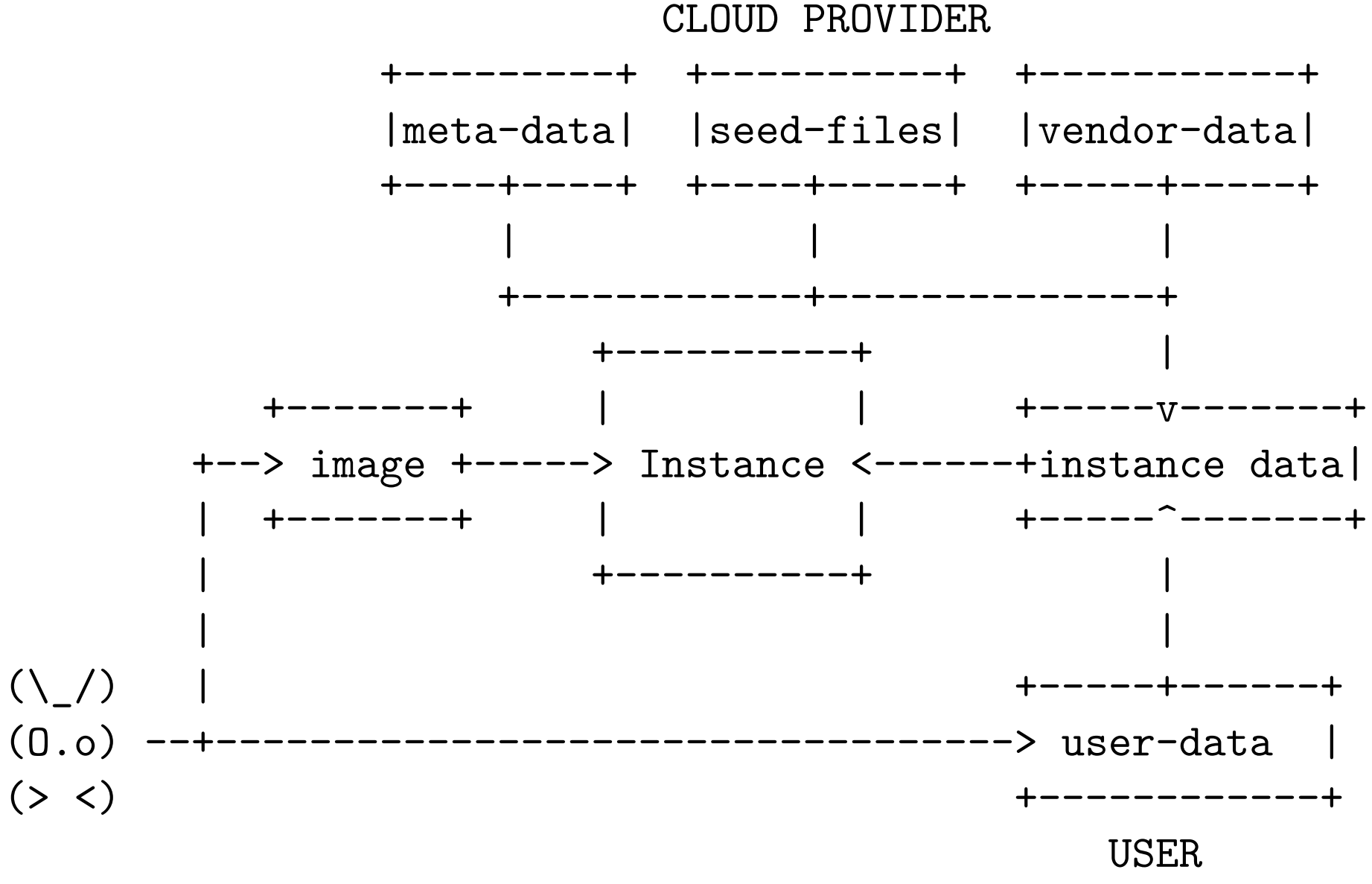
Was ist cloud-init

Paket für die Initialisierung von Cloud Instanzen - Beschreibung von Canonical -

- **Konfigurationsmöglichkeit** für Server, Virtuelle Maschinen, Cloud Instanzen wärend der **Boot Zeit**
- Skript Pro Boot, Pro Instanz, Einmalig, ...
- Konfigurationsoptionen von **externen Quellen** (Metadata Service)
- Unterstützt Linux (systemd, upstart, openrc, ...), FreeBSD, NetBSD
  - Kein illumos Zones support

Was kann cloud-init alles

- Festplatten Partitionierung, Formatierung und Konfiguration
- Ausführen von Befehlen
- Erstellen von Benutzer und Gruppen
- Paket Verwaltung (apt, yum, . . .)
- Erstellung von Dateien
- Bootstrap weiterer Konfigurationstools wie Puppet, Chef, . . .
- . . .
- Module können selbst in Python entwickelt werden


CLOUD PROVIDER
meta-data
seed-files
vendor-data
image
Instance
instance data
(_/)
(O.o)
(> <)
user-data
USER

Data Source

- meta-data und vendor-data kommen meist durch den Cloud Anbieter
- user-data abhängig vom Cloud Provider
  - Amazon EC2 via `GET` `http://169.254.169.254/<version>/user-data`
  - Cloud Stack via `GET` `http://data-server./latest/user-data`
  - SmartOS via `mdata user-data`
    - Serial Console `ttyS1` bei KVM und bhyve
    - Socket `/native/.zonecontrol/metadata.sock` bei lx-branded Zones
  - ...

meta-data Beispiel

von einer AWS EC2 Instanz:

```
ami-id: ami-91fa8712
hostname: ec-10-100-1-199.compute-1.internal
instance-id: i-87018aed
instance-type: m1.large
local-ipv4: 10.100.1.199
security-groups: default
...
```

user-data Format

user-data muss in einem der folgenden Formate vorliegen:

- GZIP Komprimiert (da ~16 kB-Limit)
- user-data Script
- Upstart Job
- Cloud Boothook
- **Cloud Config Data**
- **MIME Multipart Archive**

```
$ cloud-init devel make-mime -a config.yaml:cloud-config
-a script.sh:x-shellscript
> user-data
```

user-data > Cloud Config

- **Einfachste Möglichkeit** für user-data
- **YAML** Syntax
- Bereits viele Module und Möglichkeiten vorhanden
  - Debian Paketverwaltung
  - Benutzer Verwalten
  - CA und Zertifikatserstellung
  - ...

user-data > Cloud Config > Module

APK Configure, Apt Configure, Apt Pipelining, Bootcmd, Byobu, CA Certs, Chef, Debug, Disable EC2 Metadata, **Disk Setup**, Emit Upstart, Fan, Final Message, Foo, Growpart, Grub Dpkg, Keys to Console, Landscape, Locale, LXD, Mcollective, Migrator, Mounts, NTP, Package Update Upgrade Install, Phone Home, Power State Change, Puppet, Resizefs, Resolv Conf, RedHat Subscription, Rightscale Userdata, Rsyslog, Runcmd, Salt Minion, **Scripts Per Boot**, Scripts Per Instance, **Scripts Per Once**, Scripts User Seed Random, **Set Hostname**, Set Passwords, Snap, Spacewalk, **SSH Authorized Keys**, SSH Authkey Fingerprints, SSH Import Id, Timezone, Ubuntu Advantage Ubuntu Drivers, Update Etc Hosts, Update Hostname, **Users and Groups**, Write Files, Yum Add Repo

Vollständige Liste mit Dokumentation

user-data > Cloud Config > Beispiel

```
users:
  - name: bob
    sudo: True
    ssh_authorized_keys:
      - <ssh pub key 1>
      - <ssh pub key 2>
```

user-data > Cloud Config > Beispiel

```yaml
preserve_hostname: false
hostname: storage01
fqdn: storage01.example.com
manage_etc_hosts: true

fs_setup:
 - label: faststorage
   filesystem: 'ext4'
   device: '/dev/nvme1n1'
   partition: auto
```

user-data > MIME Multipart Archive > Beispiel

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0
--==BOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"
fs_setup:
 - label: docker
   device: '/dev/nvme1n1'
--==BOUNDARY==
Content-Type: text/cloud-boothook; charset="us-ascii"
cloud-init-per once install_nfs_utils yum install -y nfs
--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/bash
echo ECS_CLUSTER=${ecs-cluster} >> /etc/ecs/ecs.config
--==BOUNDARY==--
```

user-data > Reihenfolge

1. cloud-boothook oder bootcmd (cloud-config) Bei jedem System Start
2. x-shellscript oder runcmd (cloud-config) Beim ersten Booten
3. upstart-job Bei jedem System Start

Reihenfolgen ist auch Abhängig vom Dateiinhalt!

cloud-init unter SmartOS

Laut Dokumentation, funktioniert es nur mit:

- lx-branded Zones
- Bhyve VMs
- KVM VMs

Die Realität:

- lx-branded Zones sind veraltet
- cloud-init Pakete sind nicht installiert
- es scheint wohl niemand von Joyent zu verwenden
- Joyent bietet ähnliche Möglichkeiten mit `customer_metadata`

Customer Metadata

- SmartOS Zonen Konfiguration erfolgt über JSON Daten
- cloud-init `user-data` muss in SmartOS `customer_metadata` abgelegt werden
- `customer_metadata` Key für cloud-init User Data ist `cloud-init:user-data`
- SmartOS nutzt JSON, daher muss `user-data` (als YAML) irgendwie da rein

Customer Metadata > Beispiel

Auszug einer JSON Datei mit `cloud-init:user-data` Cloud Config als YAML:

```
{
"brand": "lx",
...
"customer_metadata": {
  "cloud-init:user-data": "#cloud-config\n\nusers:
  \n- default
  \n  - name: shaner\n    lock_passwd: false
  \n    sudo: \"ALL=(ALL) NOPASSWD:ALL\"
  \n    shell: /bin/sh"
},
"kernel_version": "4.19.0"
}
```

Aber cloud-init ist gar nicht installiert

Images von Joyent beinhaltet gar kein cloud-init

Möglichkeiten:

- Eigenes Image muss erstellt werden
- Verwendung von `user-data` als Shell-Script um cloud-init zu installieren

```
#!/bin/bash
set -e
if [[ ! -f "/root/.firstboot" ]]; then
  export DEBIAN_FRONTEND=noninteractive
  apt-get -y update && apt-get -y install cloud-init
  systemctl enable cloud-init
  touch /root/.firstboot && reboot
fi
```

Debugging

Diese Befehle gibt es nur unter **systemd/Linux**:

```
$ cloud-init analyze blame
$ cloud-init analyze show
$ cloud-init analyze dump
$ cloud-init analyze boot
```

- Logging:

`/var/log/cloud-init-output.log`

- Verzeichnis Struktur

Demo

Persönliches Fazit

- cloud-init bietet viele Möglichkeiten bei einer großen Anzahl von Anbietern
- Keine so gute Lösung für SmartOS, da auch kein Zones Support
- Schlechte Dokumenation, eigentlich nur Beispiele auf der Webseite

Fragen

- Dokumentation
- Cloud config Beispiele
- DataSourceSmartOS.py