Streamline CERNVM Contextualization Plugins

Gayatri Nesarikar (Student)
Ioannis Charalampidis
Gerardo Ganis

Google Summer of Code 2014

Github: https://github.com/GayatriSN/cernvm-micro-contextualization

Current Scenario

- Three Contextualization Agents-
  - **µCernVM** Bootloader
  - **Amiconfig** – A software by rPath, maintained by CERNVM
  - **Cloud-Init**
- Each operates independently
- Operation
  - Trying to fetch user data by trial and error
  - If user data is found break from loop
  - Parse the user-data

Optimization

- Boot Delay caused by -
  - User Data fetch loop is executed thrice
  - User Data are fetched thrice
  - User Data are Parsed by thrice
- Solution –
  - Download User Data in Micro-Bootloader Stage
  - Use Seeded Data by Cloud Init
  - Cloud Init part-handler executes Amiconfig

Step 1 – Download User Data in Micro Bootloader Stage

- Current Implementation
  - Query each Cloud Provider in a loop
  - Every iteration tries to fetch Data
  - Break when Provider Found

- New Method
  - Use “dmesg” command to detect Cloud Provider
  - Use Provider specific Download script
  - Save Cloud Provider name in file

Step 2 – Run CloudInit from Seeded Data

- Micro-boot Stage
  - User Data is saved to scratch disk
- System Boot Stage
  - Cloud Init checks seed directory before fetching
  - Data is used from seed directory
  - Cloud Init proceeds as usual

Step 3 – Running Amiconfig from CloudInit Part Handler

- Old Syntax
  - Startup script begins with ‘#!’
  - **Micro boot section:**
  - Amiconfig:
  - Cloud Init : cloud-config and/or MIME – Multipart format

```
[ucernvm-begin]
key1=value1
key2=value2
…
[ucernvm-end]
```

Step 3 – Running Amiconfig from CloudInit Part Handler

- Old Syntax
  - Startup script begins with '#!'
  - Micro boot section:
  - **Amiconfig:**
  - Cloud Init : cloud-config and/or MIME – Multipart format

```
[amiconfig]
plugins = ..
disabled_plugins = ..

[noip]
username = ..
password = ..
hostname = ..
```

Step 3 – Running Amiconfig from CloudInit Part Handler

- Old Syntax
  - Startup script begins with '#!'
  - Micro boot section:
  - Amiconfig:
  - **Cloud Init : cloud-config and/or MIME – Multipart format**

```
#cloud-config
resolv_conf:
  nameservers: ['8.8.8.8']
  searchdomains:
    - foo.example.com
    - bar.example.com
```

Step 3 – Running Amiconfig from CloudInit Part Handler

- New Syntax
  - Use MIME – Multipart syntax
  - If Old Syntax is found, Generate a MIME- Multipart

- Add ‘amiconfig’ as a MIME type

- Custom Part-Handler executes amiconfig
  - User Data are already parsed
  - Part Handler receives a payload
  - Save to file and set ‘AMICONFIG_LOCAL_USER_DATA’
  - Call amiconfig
  - Amiconfig executes as usual

Step 3 – Running Amiconfig from CloudInit Part Handler

- New syntax example

```
From nobody Tue Jun 10 10:41:07 2014
Content-Type: multipart/mixed;
boundary="===============1341993424=="
MIME-Version: 1.0

--===============1341993424==
MIME-Version: 1.0
Content-Type: text/ucernvm; charset="us-ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="ucernvm-data"
[ucernvm-begin]
key1=value1
key2=value2
...
[ucernvm-end]
--===============1341993424==
```

Step 3 – Running Amiconfig from CloudInit Part Handler

- New syntax example

```
MIME-Version: 1.0
Content-Type: text/amiconfig; charset="us-ascii"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="amiconfig-data"
[amiconfig]
plugins = dnsupdate
[dnsupdate]
server = val
host = val
hostname = val
--===============1341993424==
…
…
Content-Type: text/cloud-config; charset="us-ascii"
…
…
--===============1341993424==--
```

Progress after Mid-Term Evaluations

- Completed Amiconfig Part Handler
- Testing on most of the Cloud Providers
- Making the code functional for Cloud Providers with insufficient data
- Adding some Provider-Specific code
    - CloudStack
    - Microsoft Azure
- Documentation

Outcomes

- Significant reduction in Boot delay
  - Trial and error loop for user-data fetch eliminated
  - User Data fetched once and reused
  - User-data parsed only once in both boot stages
- Added modularity
  - Independent scripts for Cloud Providers
  - Easy to add new Provider
  - Easy to add a new Contextualization agents

Future Work

- Optimizing the site contextualization mechanism to avoid reboot
- Merging 07extracontext into 06context so the extra context can also be seeded for system-boot
- Adding code to fetch metadata when both user-data and metadata are present
- Merging the micro-boot stage code with the cernvm-micro repository
- Adding the cloudinit patch to CVMFS repository

Thank You