Fleet Health Monitoring with Yocto

Pets vs Cattle
Intro

Agenda

● Definition
● Architecture
  ● General
  ● Internet of Things
● Review some options
● POC Implementation (in Yocto)
● Torizon Architecture

About.me

● Embedded Linux Engineer
● 25+ years experience
● Solutions Architect for the Torizon Platform

● drew.moseley@toradex.com
● https://twitter.com/drewmoseley
● https://www.linkedin.com/in/drewmoseley
● https://toradex.com/torizon
WHAT WE DO

• Make Embedded Computing Easy
• Reliable Arm® System on Modules
• Lowest Cost of Ownership
• Industry-leading Support
PRODUCT PORTFOLIO

Apalis TK1

Apalis iMX6

Apalis T30

Apalis iMX6

Colibri iMX6ULL

Colibri T20

Colibri VF61

Verdin iMX8M Mini

Verdin iMX8M Plus

Apalis IMX8M Plus

Apalis IMX8X

Apalis IMX8X

Apalis iMX8

Colibri iMX8X

Colibri T30

Colibri iMX6

Colibri iMX7

Colibri IMX8ULL

Colibri VF50

Colibri IMX8X

Colibri IMX8M Plus

NXP

NVIDIA

Colibri

Verdin

Torizon

RTOS

Android

Windows
Pets vs Cattle

- **Coined by Randy Bias**¹
  - Originally from Enterprise Computing Space

- **In IoT:**
  - Pets – Weekend projects, home automation
  - Cattle – Large fleets of identical devices.

- **Fleet monitoring:**
  - Allows for structured access to health data for "cattle" devices.

Fleet Monitoring – Intro

Definition:
- **Wikipedia**¹
- Periodic monitoring of data from all devices in your fleet.
- Gathering log information.
- Analyze and visualize the data.
- "Single pane of glass"

Out of scope:
- Remote access
- Remote control
- Use case dependent analytics/features (e.g., predictive maintenance, ML/AI)

Fleet Monitoring – What is important?

Device health:
- Device online/offline, uptime/downtime
- Status of core services
- Thermal measurements

Resource utilization:
- CPU
- Memory
- Flash
- Network

Device Configuration:
- OS/Kernel/Bootloader Versions
- Deployed containers/packages and versions
- Network connection details

Device status changes:
- Failed health check
- Failed update
- Failed processes/containers

Logs:
- Kernel logs
- Docker/Application logs
- Systemd logs

Non-functional requirements:
- OSS or not
- On-Prem vs Hosted
- Performance and resource requirements
- Modularity or integration with other services

Dashboard/fleet status at a glance.
Fleet Monitoring General Architecture

- Inputs come from many sources (SNMP, local files, cloud APIs)
- Local filters allow for some processing of the data on device
- Multiple outputs can be used to send data to separate systems.
Fleet Monitoring IoT Architecture

- Inputs typically limited to just a few sources
- Much more homogeneous devices.
Discarded Options

Nagios XI

- Demo Server
- Uses SNMP or custom agent
- Hybrid OSS/Commercial Licensing
- Yocto recipes exist

Elastic Stack

- ELK (ie Elastic Search, Logstash, Kibana)
- Many input plugins (snmp, syslog, azure_event_hubs, etc)
- On-prem or hosted
- Dual license Apache 2 License
- Large on-device footprint; "Beats" to reduce that.

Datadog

- IoT Monitoring
- Closed Source/Proprietary License

Zabbix

- "Enterprise-class open source distributed monitoring solution"
- Fully OSS/GPLv2
- Paid support options
- Yocto recipes exist

Splunk

- "The Data-to-Everything(tm) Platform Powering Security, IT and DevOps"
- On-prem or hosted
- Commercial License with a feature-limited free option

2. https://www.datadoghq.com/solutions/iot-monitoring/
Considered Options

Telegraf/InfluxDB
- On-prem or hosted
- Open Source (MIT)
- Written in Golang
- No external dependencies
- 110MB flash
- Yocto recipes exist

Fluentbit/Fluentd
- Open Source (Apache)
- Part of the Cloud Native Computing Foundation
- Fluentd:  
  - Written in C and Rust  
  - 1000+ input and output plugins  
  - Depends on rubygems  
  - ~40MB flash
- FluentBit  
  - Written in C  
  - ~70 input and output plugins  
  - No external dependencies  
  - ~3MB flash/~650KB RAM
- Yocto recipes exist
Torizon Architecture

- Fluentbit – client agent
- Custom output plugin that generates JSON data to be delivered to our cloud.
- Developed an in-band data channel as part of our standard device-to-cloud transport.
- Current metrics are generic values such as CPU and Memory loading.
- Time-series data only at present.
Torizon Architecture part 2

- Individual device monitoring and fleet-wide aggregation supported.
- Custom metrics can be added using variables or Fluent bit plugins, via standard config file
- On-device filters can be used to send data to a separate destination (i.e. Kibana).
- Server API will eventually allow users to query data directly from Torizon.
Proof of Concept

- Implemented a custom DISTRO and IMAGE in a public git repo.
- Adds fluentbit and basic configuration.
- Delivers data to ElasticSearch
- Can be visualized in Kibana.
- Does not require Torizon.
Proof of Concept – Image Recipe

```
## main

meta-fleet-monitoring-poc / recipes-images / images / fleet-monitoring-poc-image_1.0.bb

![1 contributor](https://www.yoctoproject.org/)

Latest commit b268e6a 28 days ago

| Go to file | … |

| 8 lines (5 sloc) | 182 Bytes |

Click to add text

```summary = "A console-only image implementing a fleet monitoring proof of concept using fluentbit."
```

```
| 1 | SUMMARY = "A console-only image implementing a fleet monitoring proof of concept using fluentbit."
| 2 |
| 3 |
| 4 | LICENSE = "MIT"
| 5 |
| 6 | inherit core-image
| 7 |
| 8 | CORE_IMAGE_EXTRA_INSTALL += " fluent-bit "
```
Proof of Concept – fluentbit config

```bash
# POC config files for fluent-bit
#
FLEET_SERVER_URI := "example.com"

SRC_URI += " \
    file://fluent-bit.conf \n    file://input_disk.conf \n    file://input_klogs.conf \n    file://input_net.conf \n    file://input_thermal.conf \n    file://input_mem.conf \n    file://input_cpu.conf \n    file://input_osinfo.conf \n    file://parsers.conf \n"

do_install_append(){
    install -d -m 0755 $(sysconfdir)/fluent-bit
    install -m 0644 $(WORKDIR)/.conf $(sysconfdir)/fluent-bit
    sed -i -e 's/FLEET_SERVER_URI/$(FLEET_SERVER_URI)/g' $(sysconfdir)/fluent-bit/*.conf
}

SYSTEMD_AUTO_ENABLE = "enable"
```
Proof of Concept – server config

```json
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.14.0
    container_name: elasticsearch
    environment:
      - node.name=elasticsearch
      - discovery.type=single-node
    ports:
      - 9200:9200
      - 9300:9300
  volumes:
    - data:/usr/share/elasticsearch/data
  networks:
    - elastic

kibana:
  image: docker.elastic.co/kibana/kibana:7.14.0
  container_name: kibana
  ports:
    - 5601:5601
  environment:
    ELASTICSEARCH_URL: http://elasticsearch:9200
    ELASTICSEARCH_HOSTS: '["http://elasticsearch:9200"]'
  networks:
    - elastic

networks:
  elastic:
    driver: bridge

volumes:
  data:
    driver: local
```
Proof of Concept – howto

Device Setup

$ git clone https://github.com/drewmoseley/meta-fleet-monitoring-poc.git \
layers/meta-fleet-monitoring-poc
$ bitbake-layers add-layer layers/meta-fleet-monitoring-poc
$ echo 'FLEET_SERVER_URI = "<IP-ADDRESS-OF-SERVER>"' >> conf/local.conf
$ bitbake fleet-monitoring-poc-image

Server Setup

$ cd layers/meta-fleet-monitoring-poc/misc
$ docker-compose -f fleet-monitoring-server-docker-compose.yml up -d
Demo Time