IoT and microservices with OE/The Yocto Project

#containerizing the planet

Robert Berger

Reliable Embedded Systems e.U.

July 2, 2020
# Table of Contents - Session

- Intro
  - The use case - on host (& target)
    - Hardware
    - Software
      - Host: (m)TIG Stack - "classic"
      - Host: (m)TIG Stack - "vm"
      - Host: (m)TIG Stack - "container" - Microservices
  - Developer: Get app(s) on the Target
    - The Golang approach
      - #yoctoizing The Golang approach with Yocto SDK
  - Yocto Person: Get packages on the Target
    - #yoctoizing: Build package(s) from sources
    - #yoctoizing: Build package(s) from prebuilt binaries
    - (m)TIG package(s) summary
Table of Contents - Session

Yocto Person: Get Docker packages on the Target
  #yoctoizing: add packages
  #yoctoizing: configure kernel
Developer: docker-compose on the Target
  run docker-compose
Container Standardization
Yocto Person: oci (app) container
  Build oci (app) container
  Push oci (app) container to registry
  Pull/Run oci (app) container
  Kill oci (app) container
  Further work
Thank you
Table of Contents - Section

Intro
Let me introduce myself

Robert Berger
Embedded Software Evangelist

Email: robert.berger@ReliableEmbeddedSystems.com
Phone: +43 (0) 699 17 69 07 19
Web: http://ReliableEmbeddedSystems.com

Building world class world wide win/win cooperations by helping you to create better embedded software!
What business am I in?

/> consulting

/> training

» Teaching is a great way to keep learning!

» The Yocto Project - A thorough Overview - 4 days
  (http://rlbl.me/yocto-en-pdf)

» Refresher to Embedded Linux & Intro to the Yocto Project - 5 days -
  (http://rlbl.me/intely-en-pdf)

/> engineering

Figure: http://yocto.training
We trained (excerpt)
The use case - on host (& target)

Hardware
Software
Host: (m)TIG Stack - "classic"
Host: (m)TIG Stack - "vm"
Host: (m)TIG Stack - "container" - Microservices
Pain point: "investigate"/reduce

» electricity bill

installed several sensors

» Sonoff POWR2 Modules\(^2\) (power measurement)
» Tasmota alternative firmware for ESP8266\(^3\) - mqtt\(^4\)

\(^2\)https://sonoff.tech/product/wifi-diy-smart-switches/powr2
\(^3\)https://github.com/arendst/Tasmota
\(^4\)http://mqtt.org/
The use case - on host (& target)

(m)TIG Stack

Figure: mTIG stack

/> mosquitto: mqtt broker
/> Telegraf
   /» collects, processes, aggregates and writes metrics
/> InfluxDB
   /» time series database
/> Grafana
   /» metric analytics & visualization suite
   /» web interface
(m)TIG Stack - "classic"

The use case - on host (& target)

Host: (m)TIG Stack - "classic"

Figure: "classic"
The use case - on host (& target)

Host: (m)TIG Stack - "vm"

(m)TIG Stack - "vm"

Figure: "multiple vms"
The use case - on host (& target)

Host: (m)TIG Stack - "container" - Microservices

Figure: "multiple containers"
Dockerfile ↔

dependencies
Alpine 3.12
musl

Figure: Dockerfile: mosquitto container
Listing 1: Dockerfile: mosquitto 1/2

1. FROM alpine:3.12
2. LABEL maintainer="Roger Light <roger@atchoo.org>" \ description="Eclipse Mosquitto MQTT Broker"
3. ENV LWS_VERSION=2.4.2
4. COPY mosq.tar.gz /tmp
5. RUN set -x && \
6. apk --no-cache add --virtual build-deps build-base cmake gnupg openssl-dev util-linux-dev && \
7. wget https://github.com/warmcat/libwebsockets/archive/v${LWS_VERSION}.tar.gz -O /tmp/lws.tar.gz && \
8. mkdir -p /build/lws && tar --strip=1 -xf /tmp/lws.tar.gz -C /build/lws && \
9. rm /tmp/lws.tar.gz && cd /build/lws && \
10. cmake . -DCMAKE_BUILD_TYPE=MinSizeRel -DCMAKE_INSTALL_PREFIX=/usr -DLWS_IPV6=ON -DLWSWITHOUT_BUILTIN_GETIFADDRS=ON -DLWSWITHOUT_CLIENT=ON -DLWSWITHOUT_EXTENSIONS=ON -DLWSWITHOUT_TESTAPPS=ON -DLWSWITH_SHARED=OFF -DLWSWITH_ZIP_FOPS=OFF -DLWSWITH_ZLIB=OFF && \
11. make -j "$(nproc)" && rm -rf /root/.cmake && mkdir -p /build/mosq && \
12. tar --strip=1 -xf /tmp/mosq.tar.gz -C /build/mosq && rm /tmp/mosq.tar.gz && \
13. make -C /build/mosq -j "$(nproc)" CFLAGS="-Wall -O2 -I/build/lws/include" LDFLAGS="-L/build/lws/lib" binary && \
14. addgroup -S -g 1883 mosquitto 2>/dev/null && \
15. adduser -S -u 1883 -D -h /var/empty -s /sbin/nologin -G mosquitto -g mosquitto mosquitto 2>/dev/null && \
16.
Listing 2: Dockerfile: mosquitto 2/2

```bash
mkdir -p /mosquitto/config /mosquitto/data /mosquitto/log && 
install -d /usr/sbin/ && 
install -s -m755 /build/mosq/client/mosquitto_pub /usr/bin/mosquitto_pub && 
install -s -m755 /build/mosq/client/mosquitto_rr /usr/bin/mosquitto_rr && 
install -s -m755 /build/mosq/client/mosquitto_sub /usr/bin/mosquitto_sub && 
install -s -m644 /build/mosq/lib/libmosquitto.so.1 /usr/lib/libmosquitto.so.1 && 
install -s -m755 /build/mosq/src/mosquitto /usr/sbin/mosquitto && 
install -s -m755 /build/mosq/src/mosquitto_passwd /usr/bin/mosquitto_passwd && 
install -m644 /build/mosq/mosquitto.conf /mosquitto/config/mosquitto.conf && 
chown -R mosquitto:mosquitto /mosquitto && 
apk --no-cache add \ 
  ca-certificates && 
apk del build-deps && 
rm -rf /build

VOLUME ["/mosquitto/data", "/mosquitto/log"]

# Set up the entry point script and default command
COPY docker-entrypoint.sh /
EXPOSE 1883
ENTRYPOINT ["/docker-entrypoint.sh"]
CMD ["/usr/sbin/mosquitto", "-c", "/mosquitto/config/mosquitto.conf"]
```
several containers on the same node/machine

Figure: docker-compose: (m)TIG stack containers
Listing 3: docker-compose.yml 1/3

```yaml
version: "2"

services:
  influxdb:
    container_name: influxdb
    image: influxdb
    ports: # port on host:port in container
      - "8086:8086"
    volumes: # volume on host:volume in container
      - /home/student/projects/tig/data/influxdb:/var/lib/influxdb
      - /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/influxdb:/etc/influxdb/
    restart: always # auto restart if crash
  telegraf:
    container_name: telegraf
    image: telegraf
    network_mode: "host" # host network
    volumes:
      - /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/telegraf/telegraf.conf:/etc/telegraf/telegraf.conf
      - /var/run/docker.sock:/var/run/docker.sock
    restart: always
```

1"Good programmers copy, great programmers paste!"

(m)TIG Stack - "docker-compose"

/> on UB 18.04 kvm/machine or ARM target

Listing 4: docker-compose.yml 2/3

```yaml
grafana:
  container_name: grafana
  image: grafana/grafana
  user: "0" # default user is id="0" (root)
  ports:
    - "3000:3000"
  volumes:
    - /home/student/projects/tig/data/grafana:/var/lib/grafana
    - /home/student/projects/tig/log/grafana:/var/log/grafana
    - /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/grafana/grafana.ini:/etc/grafana/grafana.ini
  links: # Containers for the linked service are reachable at a hostname identical to the alias,
    # or the service name if no alias was specified.
    # Links also express dependency between services in the same way as depends_on
    - influxdb
  restart: always
```
The use case - on host (& target)

(m)TIG Stack - "docker-compose"

/> on UB 18.04 kvm/machine or ARM target

Listing 5: docker-compose.yml 3/3

```yaml
38  mqtt:
39      container_name: mqtt
40      image: eclipse-mosquitto:latest
41      user: "0"
42      ports:
43          - "1883:1883"
44          - "9001:9001"
45      volumes:
46          - /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/mqtt:/mosquitto/config/
47          - /home/student/projects/tig/data/mqtt:/mosquitto/data/
48          - /home/student/projects/tig/log/mqtt:/mosquitto/log/
49      restart: always
```
The use case - on host (& target)

Host: (m)TIG Stack - "container" - Microservices

(m)TIG Stack - "Config changes"

/> on UB 18.04 kvm/machine or ARM target

Listing 6: telegraf.conf excerpt

```
[[inputs.mqtt_consumer]]
servers = ["tcp://mqttbrk1.res.training:1883"]
qos = 0
connection_timeout = "30s"
topics = [
  "tele/line/office/ac/SENSOR",
  "tele/temp/office/ac/SENSOR",
]
```

Figure: measurements
The use case - on host (& target)

Host: (m)TIG Stack - "container" - Microservices

(m)TIG Stack - "Start it up"

/> on UB 18.04 kvm/machine or ARM target

Listing 7: docker-compose up

1 cd <wherever docker-compose.yml is>
2 docker-compose up

/> The first time the containers will be downloaded
/> The containers will be started
/> You should be able to access Grafana here: <ip address>:3000
/> change default admin password
/> Add influxdb data source
/> Create dashboards/panels
Developer: Get app(s) on the Target
The Golang approach
#yoctoizing The Golang approach with Yocto SDK
Build/Install influxdb

Listing 8: build/install influxdb for host - in theory

```bash
# GOPATH
mkdir $HOME/gocodez && export GOPATH=$HOME/gocodez
# Get the source code from git
go get github.com/influxdb/influxdb
cd $GOPATH/src/github.com/influxdb/influxdb
cp hooks/pre-commit.git/hooks/
go get golang.org/x/tools/cmd/vet
cd $GOPATH/src/github.com/influxdb
go get -u -f -t ./...
# Build Test (go build command is just for build test)
go build -ldflags=" -X main.version=$VERSION -X main.branch=$BRANCH -X main.commit=$COMMIT -X main.buildTime=$TIME" ./...
# Install
go install -x ./...
```

Listing 9: build/install influxdb for target - in theory

```bash
# Cross compile for ARM7 platform
cd $GOPATH/src/github.com/influxdb
GOOS=linux GOARCH=arm GOARM=7 go install -x ./...
```

1https://hwwong168.wordpress.com/2015/10/12/cross-compile-influxdb-0-9-3-for-raspberry-pi-pi2-and-synology-nas/
Add Golang support to Yocto SDK

Listing 10: add to local.conf

```bash
# --> go/lang stuff
# attempt to add go-lang to SDK
TOOLCHAIN_HOST_TASK_append_pn -core -image -sato -sdk = " \n    packagegroup-go-cross-canadian-${MACHINE} \n" 

TOOLCHAIN_TARGET_TASK_append_pn -core -image -sato -sdk = " \n    ${@multilib_pkg_extend(d, 'packagegroup-go-sdk-target')} \n" 

# <-- go/lang stuff
```

Listing 11: build "classic" SDK

```bash
bitbake core-image-sato-sdk -c populate_sdk
```

/> You could BitBake this in a "poky build container"\(^1\)

/> The SDK could be installed in an SDK container\(^2\)

\(^1\)[https://github.com/crops/poky-container](https://github.com/crops/poky-container)

\(^2\)[https://github.com/crops/extsdk-container](https://github.com/crops/extsdk-container)
# Create a directory for the Go stuff - this is standard to place source code

```bash
sudo rm -rf /workdir/sources/gocodez && mkdir -p /workdir/sources/gocodez
```

# Export GOPATH

```bash
export GOPATH=/workdir/sources/gocodez
```

# Get the source code from git

```bash
mkdir -p ${GOPATH}/src/github.com/influxdata && cd ${GOPATH}/src/github.com/influxdata
git clone https://github.com/influxdata/influxdb
```

# version 1.8

```bash
cd influxdb && git checkout 1.8
dep init # maybe a second time if it fails
dep ensure
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influxd
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_inspect
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_stress
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_tools
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_tsm
```

```bash
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go install -x ./...
```
Build influxdb for target

Listing 13: SDK build influxdb for target 2/2

```bash
tree ${GOPATH}/bin
/workdir/sources/gocodez/bin
  `-- linux_arm
     |   `-- influx
     |       `-- influxd
     |               `-- influx_inspect
     |                   `-- influx_stress
     |                       `-- influx_tools
     |                           `-- influx_tsm
     |                               `-- stress_test_server
     `-- test_client

/> yay!
```
Yocto Person: Get packages on the Target
#yoctoizing: Build package(s) from sources
#yoctoizing: Build package(s) from prebuilt binaries
(m)TIG package(s) summary
DESCRIPTION = "InfluxDB is an open source time series platform. This includes APIs for storing and querying data, processing it in the background for ETL or monitoring and alerting purposes, user dashboards, and visualizing and exploring the data and more. The master branch on this repo now represents the latest InfluxDB, which now includes functionality for Kapacitor (background processing) and Chronograf (the UI) all in a single binary."

GO_IMPORT = "github.com/influxdata/influxdb"
SRC_URI = "git://${GO_IMPORT};branch=1.8"
SRCREV = "781490de48220d7695a05c29e5a36f550a4568f5"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${WORKDIR}/${PN}-${PV}/src/${GO_IMPORT}/LICENSE;md5=f39a8d10930fb37bd59adabb3b9d0bd6"

inherit go
CGO_ENABLED = "1"

DEPENDS += "go-dep-native"
RDEPENDS_${PN}-dev += "python bash"
RDEPENDS_${PN}-staticdev += "python bash perl make"
DEPENDS += "github.com-cespare-xxhash"
DEPENDS += "code.googlesource.com-gocloud"
RDEPENDS_${PN} += "github.com-cespare-xxhash"
Yocto Person: Get packages on the Target

#yoctoizing: Build package(s) from sources

**Build package(s) from sources**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>do_compile_prepend() {</td>
</tr>
<tr>
<td>28</td>
<td>rm -f ${WORKDIR}/build/src/${GO_IMPORT}/Gopkg.toml</td>
</tr>
<tr>
<td>29</td>
<td>rm -f ${WORKDIR}/build/src/${GO_IMPORT}/Gopkg.lock</td>
</tr>
<tr>
<td>30</td>
<td>cd ${WORKDIR}/build/src/${GO_IMPORT}</td>
</tr>
<tr>
<td>31</td>
<td>dep init</td>
</tr>
<tr>
<td>32</td>
<td>dep ensure</td>
</tr>
<tr>
<td>33</td>
<td>}</td>
</tr>
<tr>
<td>34</td>
<td>python do_fix_objs_trampoline() {</td>
</tr>
<tr>
<td>35</td>
<td>bb.build.exec_func('do_fix_objs', d)</td>
</tr>
</tbody>
</table>
| 36   | }
| 37   | # trying to remove some irrelevant obj files, which tools choke on |
| 38   | do_fix_objs() { |
| 39   |   find ${D}${libdir}/go/pkg -depth -type f -name go-relocation-test-* -exec rm -rf {} \ |
| 40   |   find ${D}${libdir}/go/pkg -depth -type f -name compressed-32.obj -exec rm -rf {} \ |
| 41   |   find ${D}${libdir}/go/pkg -depth -type f -name gcc-386-freebsd-exec -exec rm -rf {} \ |
| 42   |   find ${D}${libdir}/go/pkg -depth -type f -name v1reflect.gox -exec rm -rf {} \ |
| 43   |   if [ -f ${D}${libdir}/go/src/github.com/influxdata/influxdb/build.py ]; then |
| 44   |     rm -f ${D}${libdir}/go/src/github.com/influxdata/influxdb/build.py |
| 45   |   fi |
| 46   | addtask do_fix_objs_trampoline |
| 47   | addtask do_fix_objs |
| 48   | # looks like until populate_sysroot we are good now |
| 49   | do_install[postfuncs] += "do_fix_objs_trampoline" |
This was a time consuming task!

"Normally" Golang magically pulls in all required dependencies

Dependency hell: influxdb-from-sources_1.8.0.bb needs some help on those
   /» github.com-cspare-xxhash_2.1.1.bb
   /» code.googlesource.com-gocloud_0.58.0.bb

Please Golang people out there contact me and tell me what I am doing wrong
I searched on the layerindex¹ and started from here²

Listing 16: influxdb-prebuilt_1.8.0.bb

```bash
require influxdb.inc

SRC_URI_armv7a = "https://dl.influxdata.com/influxdb/releases/influxdb-$(PV)_linux_armhf.tar.gz \ 
    file://LICENSE"
SRC_URI[sha256sum] = "@bb.utils.contains('MACHINEOVERRIDES', 'armv7a', \ 
    'fdac157f02e8231e1925d8e9bc325c88b7ba55ebab2340c549ef10640dbd0cb', '', d))"

SRC_URI_x86-64 = "https://dl.influxdata.com/influxdb/releases/influxdb-$(PV)-static_linux_amd64.tar.gz \ 
    file://LICENSE"
SRC_URI[sha256sum] = "@bb.utils.contains('MACHINEOVERRIDES', 'x86-64', \ 
    'aedc5083ae2e61ef374dbde5044ec2a5b27300e73eb92ccd135e6ff9844617e2', '', d))"

LIC_FILES_CHKSUM = "file://$(WORKDIR)/LICENSE;md5=f39a8d10930fb37bd59adabb3b9d0bd6"

# The open source license for InfluxDB is available in the GitHub repository.
# https://github.com/influxdata/influxdb/blob/1.8/LICENSE
# I copied LICENSE from there to my meta data
LICENSE = "MIT"

S = "$(WORKDIR)/${PN}-${PV}-1"
```

¹https://layers.openembedded.org/layerindex/
²https://layers.openembedded.org/layerindex/recipe/121293/
Listing 17: influxdb.inc 1/2

1 DESCRIPTION = "InfluxDB"
2 SUMMARY = "InfluxDB is a time series database designed to handle high write and query loads."
3 HOMEPAGE = "https://www.influxdata.com/products/influxdb-overview/"

5 INSANE_SKIP_${PN}_append = "already-stripped"

7 do_install() {
8     # /etc
9     install -d $(D){sysconfdir}/influxdb
10    install -d $(D){sysconfdir}/logrotate.d
11    # x86-64 and armv7 have influxdb.conf at different places
12    if [ -f $(S)/etc/influxdb/influxdb.conf ]; then
13       install -m 0644 $(S)/etc/influxdb/influxdb.conf $(D){sysconfdir}/influxdb/
14    else
15       install -m 0644 $(S)/influxdb.conf $(D){sysconfdir}/influxdb/
16    fi
17    # @@@ TODO: fix it, armv7 has this file, x86-64 not
18    if [ -f $(S)/etc/logrotate.d/influxdb ]; then
19       install -m 0644 $(S)/etc/logrotate.d/influxdb $(D){sysconfdir}/logrotate.d/
20    fi
21    # /usr/bin
22    install -d $(D){bindir}
23    # arm has a bin dir
24    if [ -d $(S)/usr/bin/ ]; then
25       install -m 0755 $(S)/usr/bin/* $(D){bindir}/
else
    # x86-64 does not
    install -m 0755 $(S)/influx $(D)${bindir}/
    install -m 0755 $(S)/influxd $(D)${bindir}/
    install -m 0755 $(S)/influx_inspect $(D)${bindir}/
    install -m 0755 $(S)/influx_stress $(D)${bindir}/
    install -m 0755 $(S)/influx_tsm $(D)${bindir}/
fi
# /usr/lib
if (! [ -d $(S)/usr/lib ]; then
    # only form armv7
    install -d $(D)${systemd_unitdir}/system
    sed -i 's/User=influxdb/User=root/g' $(S)/usr/lib/influxdb/scripts/influxdb.service
    sed -i 's/Group=influxdb/Group=root/g' $(S)/usr/lib/influxdb/scripts/influxdb.service
    install -m 0644 $(S)/usr/lib/influxdb/scripts/influxdb.service $(D)${systemd_unitdir}/system
fi
# /usr/share
install -d $(D)${datadir}/man/man1
install -m 0644 $(S)/usr/share/man/man1/* $(D)${datadir}/man/man1/
# /var/lib
install -d $(D)${localstatedir}/lib/influxdb
install -d $(D)${localstatedir}/log/influxdb
}

inherit systemd
SYSTEMD_SERVICE_${PN} = "influxdb.service"
This was a bit easier!

But

- prebuilt binaries are packaged differently depending on architecture (Why?)
- how reproduce-able is this?
additional work is required, which could be quite time consuming
image recipe
easy
mosquitto
easy, since recipe exists upstream
Influxdb
needs more work with respect to init scripts and configuration
at least 2 recipes as we saw: ideally from sources + dependencies, but prebuilt also possible
Telegraf
needs to be built + init scripts + configuration
at least 2 recipes: ideally from sources + dependencies, but prebuilt also possible
Grafana
needs to be built + init scripts + configuration
at least 2 recipes: ideally from sources + dependencies, but prebuilt also possible
Yocto Person: Get Docker packages on the Target
#yoctoizing: add packages
#yoctoizing: configure kernel
Docker on the Target

/> Pain point
/> » BitBaking Telegraf, Influxdb and Grafana from sources seems to be hard
/> » we could use prebuilt containers for the Target
/> » but now we need Docker and friends on the Target

... to the rescue
Yocto Person: Get Docker packages on the Target

meta-virtualization

/> add the meta-virtualization¹ layer
/> add some packages

Listing 19: core-image-minimal-virt-docker-ce.bb

```bash
1 require recipes-core/images/core-image-minimal.bb
2
3 # from meta-virtualization
4 IMAGE_INSTALL += "
5   docker-ce 
6   docker-ce-contrib 
7   python3 
8   python3-docker-compose 
9 "
10 # convenience
11 IMAGE_INSTALL += "
12   bash 
13   btrfs-tools 
14   dropbear 
15   e2fsprogs-mke2fs 
16   udev-extraconf 
17 "
```

¹http://git.yoctoproject.org/cgit/cgit.cgi/meta-virtualization/
Figure: docker architecture
Configure the kernel

Listing 20: target: run check-config.sh

1  ./usr/share/docker/check-config.sh
2  
3  info: reading kernel config from /proc/config.gz ...
4  
5  Generally Necessary:
6   - cgroup hierarchy: properly mounted [/sys/fs/cgroup]
7   - CONFIG_NAMESPACES: enabled
8   - CONFIG_NET_NS: enabled
9   - CONFIG_PID_NS: enabled
10  - CONFIG_IPC_NS: enabled
11  - CONFIG_UTS_NS: enabled
12  - CONFIG_CGROUPS: enabled
13  - CONFIG_CGROUP_CPUACCT: enabled
14  - CONFIG_CGROUP_DEVICE: enabled
15  - CONFIG_CGROUP_FREEZER: enabled
16  - CONFIG_CGROUP_SCHED: enabled
17  - CONFIG_CGROUPS: enabled
18  - CONFIG_CGROUPS: enabled
19  - CONFIG_CGROUPS: enabled
20  - CONFIG_CGROUPS: enabled
21  - CONFIG_CGROUPS: enabled (as module)
22  - CONFIG_CGROUPS: enabled (as module)
23  - CONFIG_CGROUPS: enabled (as module)
24  - CONFIG_CGROUPS: enabled (as module)
25  - CONFIG_CGROUPS: enabled (as module)
26  ...
Yocto Person: Get Docker packages on the Target

(m)TIG Stack - "container" - Microservices

Figure: "multiple containers"
Developer: docker-compose on the Target
run docker-compose
(m)TIG Stack - "Start it up"

/> on ARM target same as on UB 18.04 kvm/machine

Listing 21: docker-compose up

1 cd <wherever docker-compose.yml is>
2 docker-compose up

/> The first time the containers will be downloaded
/> The containers will be started
/> You should be able to access Grafana here: <ip address>:3000
/> change default admin password
/> Add (or import from before) influxdb data source
/> Create (or import from before) dashboards/panels
(m)TIG Stack - "Running"

Figure: (m)TIG running
Put it all together
Container Standardization
OCI (Open Container Initiative): [https://opencontainers.org/](https://opencontainers.org/)
- image-spec: [https://github.com/opencontainers/image-spec](https://github.com/opencontainers/image-spec)
  ..interoperable format to build, transport and prepare a container image to run
- runtime-spec: [https://github.com/opencontainers/runtime-spec](https://github.com/opencontainers/runtime-spec)
  ..lifecycle running container
  ../how a tool executing a container must behave and interact with it
Yocto Person: oci (app) container

Build oci (app) container
Push oci (app) container to registry
Pull/Run oci (app) container
Kill oci (app) container
Further work
we need a minimal rootfs without kernel and
meta-virtualization
skopeo in (host) container or on host machine

Listing 22: app-container-image-influxdb-prebuilt-oci.bb

```bash
1 SUMMARY = "An influxdb image"
2 #LICENSE = "MIT"
3 #LIC_FILES_CHKSUM = "file://${COREBASE}/meta/CONFIG.MIT;md5=1234"
4
5 require dynamic-layers/virtualization-layer/recipes-core/images/app-container-image-oci.bb
6 require recipes-core/images/app-container-image-influxdb-prebuilt-common.inc
```

Listing 23: app-container-image-influxdb-prebuilt-common.inc

```bash
1 # Note that busybox is convenient and might be required as well by /bin/sh
2 IMAGE_INSTALL += " \\n3   busybox \n4   influxdb \n5 "
```
Listing 24: app-container-image-oci.bb 1/2

1 SUMMARY = "A minimal container image"
2 #LICENSE = "MIT"
3 #LIC_FILES_CHKSUM = "file://${COREBASE}/meta/CONFIG.MIT;md5=1234"
4
5 IMAGE_FSTYPES = "container oci"
6
7 inherit image
8 inherit image-oci
9
10 IMAGE_FEATURES = ""
11 IMAGE_LINGUAS = ""
12 NO_RECOMMENDATIONS = "1"
13
14 # Allow build with or without a specific kernel
15 # see poky/meta/classes/image-container.bbclass
16 IMAGE_CONTAINER_NO_DUMMY = "0" # DUMMY = 1 ;)
17 # 0 -> no kernel, PREFERRED_PROVIDER_virtual/kernel = linux-dummy
18 # 1 -> kernel, PREFERRED_PROVIDER_virtual/kernel != linux-dummy
19
20 IMAGE_INSTALL = "\
21   base-files \
22   base-passwd \
23   netbase \
24   "
Listing 25: app-container-image-oci.bb 2/2

```bash
26  # Workaround /var/volatile for now
27  ROOTFS_POSTPROCESS_COMMAND += "rootfs_fixup_var_volatile ; ";
28
29  rootfs_fixup_var_volatile () {
30      install -m 1777 -d ${IMAGE_ROOTFS}/${localstatedir}/volatile/tmp
31      install -m 755 -d ${IMAGE_ROOTFS}/${localstatedir}/volatile/log
32  }
```

/> inspired by

/> "Building Container Images with OpenEmbedded and the Yocto Project - Scott Murray""1 - 2018

1https://www.youtube.com/watch?v=OSyLoHYxGLQ
Note that you could build this in a "poky build container"\(^1\)
depending on the MACHINE we can build for x86-64 or ARM

Listing 26: build container: bake it

1. bitbake app-container-image-influxdb-prebuilt-oci

Note that you could build this in a "skopeo\(^2\) container"\(^3\)

Listing 27: skopeo container: push to docker registry

1. cd /workdir/build/container-x86-64-golang/tmp/deploy/images/container-x86-64/
2. tar xvf ../app-container-image-influxdb-prebuilt-oci-container-x86-64-20200619053229.rootfs-oci-latest-x86_64-\(←\)
   linux.oci-image.tar
3. skopeo --debug copy -f v2s2 --dest-creds yoctotrainer:<registry password> oci:app-container-image-influxdb-\(←\)
   prebuilt-oci-container-x86-64-20200619053229.rootfs-oci:latest docker://yoctotrainer/app-container-influxdb-\(←\)
   prebuilt-oci
4. # try:
5. # docker pull docker.io/yoctotrainer/app-container-influxdb-prebuilt-oci

\(^1\)https://github.com/crops/poky-container
\(^2\)https://github.com/containers/skopeo
\(^3\)https://github.com/RobertBerger/skopeo-container
### Listing 28: host: which docker containers are running?

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th>CREATED</th>
<th>STATUS</th>
<th>NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>3d649b14f4aa</td>
<td>yoctotrainer/app-container...</td>
<td>&quot;/bin/ash -l&quot;</td>
<td>32 sec ago</td>
<td>Exited (0) 2 sec ago</td>
<td>#yoctoizing</td>
</tr>
</tbody>
</table>

### Listing 29: host: docker run ash

```bash
4  ID_TO_KILL=$(docker ps -a -q --filter ancestor=yoctotrainer/app-container-influxdb-prebuilt-oci:latest)
5  docker stop ${ID_TO_KILL}
6  docker rm -f ${ID_TO_KILL}
7  docker pull yoctotrainer/app-container-influxdb-prebuilt-oci:latest
8  docker run -p 8086:8086 --interactive --tty --entrypoint="/bin/ash←
    yoctotrainer/app-container-influxdb-prebuilt-oci:latest --login
```
Listing 30: in container

```bash
1  root@bc41060a6bcc:/# influxd
2  8888888 .d888 888 8888888b. 8888888b.
3  888 d88P" 888 888 "Y88b 888 "88b
4  888 888 888 888 888 888 888 888 888 888 888 888 888 888 8888888K.
5  888 888 "Y88b 888 888 888 888 888 Y8bd8P' 888 888 888 "Y88b
6  888 888 888 888 888 888 888 888 888 888 X88K 888 888 888 888 888
7  888 888 888 888 888 888 Y88b 888 .d8""8b. 888 .d8P 888 d8P
8  8888888 888 888 888 888 888 8888888P" 8888888P
9
10 2020-06-21T11:55:25.906737 Z info InfluxDB starting {"log_id": "0NXg_RKW000", "version": "1.8.0", "branch": "1.8", "commit": "781490de48220d7695a05c29e5a36f550a4568f5"}
11
12 2020-06-21T11:55:25.906793 Z info Go runtime {"log_id": "0NXg_RKW000", "version": "go1.13.8", "maxprocs": 6}
13
14 2020-06-21T11:55:26.036380Z info Using data dir {"log_id": "0NXg_RKW000", "service": "store", "path": "/var/lib/influxdb/data/"
15
16 2020-06-21T11:55:26.036726Z info Compaction settings {"log_id": "0NXg_RKW000", "service": "store", "max_concurrent_compactions": 3, "throughput_bytes_per_second": 50331648, "throughput_bytes_per_second_burst": 50331648}
17
18 2020-06-21T11:55:26.037907Z info Open store (start) {"log_id": "0NXg_RKW000", "service": "store", "trace_id": "0NXg_RqG000", "op_name": "tsdb_open", "op_event": "start"}
19
20 2020-06-21T11:55:26.038025Z info Open store (end) {"log_id": "0NXg_RKW000", "service": "store", "trace_id": "0NXg_RqG000", "op_name": "tsdb_open", "op_event": "end", "op_elapsed": "0.121ms"}
21
22 2020-06-21T11:55:26.040510Z info Opened service {"log_id": "0NXg_RKW000", "service": "subscriber"}
```
Listing 31: host: which containers are running?

```
1  docker ps -a
2  CONTAINER ID   IMAGE                  COMMAND                              CREATED       STATUS                    PORTS                                     NAMES
3  bc41060a6bcc   yoctotrainer/app-contai...  "/bin/ash --login" 8 min ago  Up 8 min 0.0.0.0:8086->8086/tcp  laughing
```

Listing 32: host: kill container

```
1  ID_TO_KILL=$(docker ps -a -q --filter ancestor=yoctotrainer/app-container-influxdb-prebuilt-oci:latest)
2  docker stop ${ID_TO_KILL}
3  docker rm -f ${ID_TO_KILL}
```
BitBake recipes for each app (see before)
oci containers for each app (see before)
  » multi-arch
license compliance
  » issue with Golang
  » issue with (layered) containers
upload to some registry (see before)
autostart docker-compose
distributed architecture via kubernetes
Thank you
Thank you!

Robert Berger - Embedded Software Evangelist

- robert.berger@ReliableEmbeddedSystems.com
- +43 (0) 699 17 69 07 19
- www.ReliableEmbeddedSystems.com
- www.linkedin.com/company/reliable-embedded-systems
- www.twitter.com/ReliableEmbSys
- www.instagram.com/reliableembeddedsystems
- www.youtube.com/user/ReliableEmbeddedSys