IoT and microservices with OE/The Yocto Project

#containerizing the planet

Robert Berger

Reliable Embedded Systems e.U.

July 1, 2020

Table of Contents - Session

Table of Contents - Session

Table of Contents - Section

Let me introduce myself

ReliaBleEmBeddedSystems Consulting Training Engineering

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?

Figure: http://yocto.training

- /> 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

We trained (excerpt)

Table of Contents - Section

(remote) Lab in Austria

- 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/

(m)TIG Stack

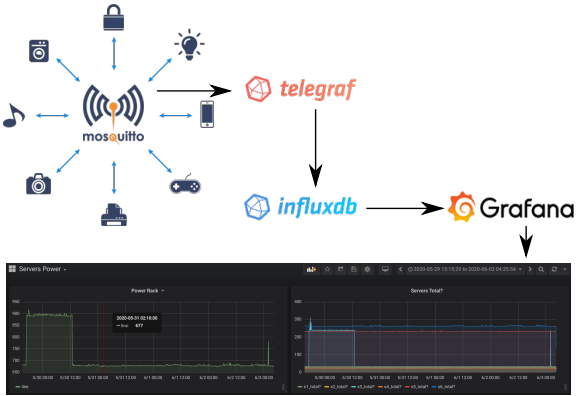


Figure: mTIG stack

- /> 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"

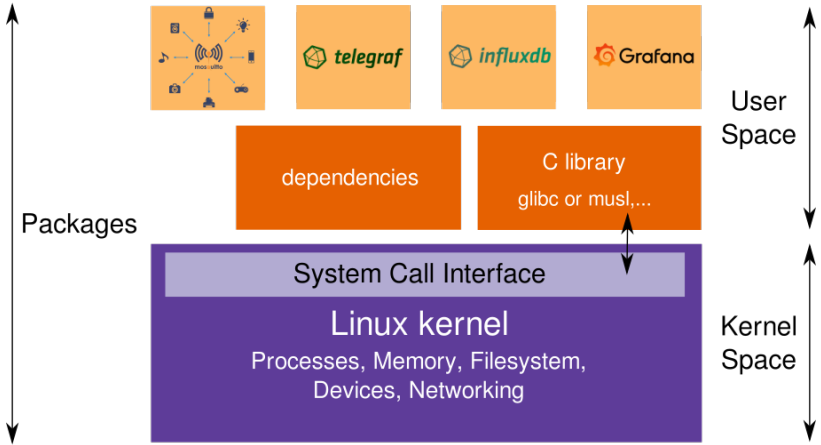


Figure: "classic"

(m)TIG Stack - "vm"

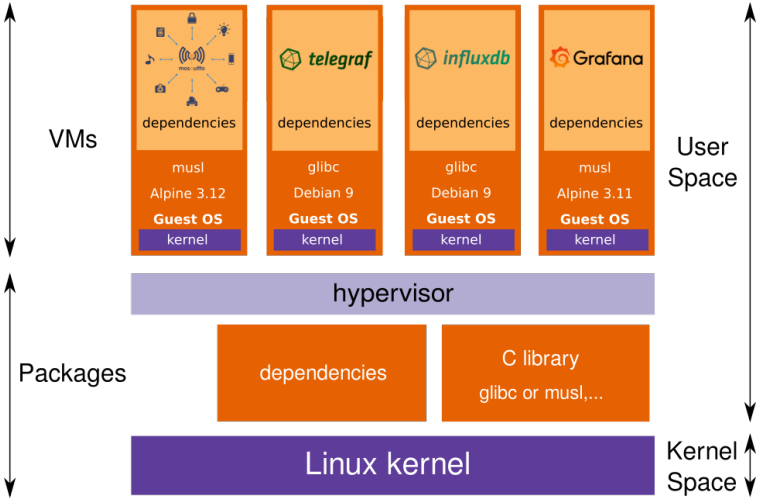


Figure: "multiple vms"

(m)TIG Stack - "container" - Microservices

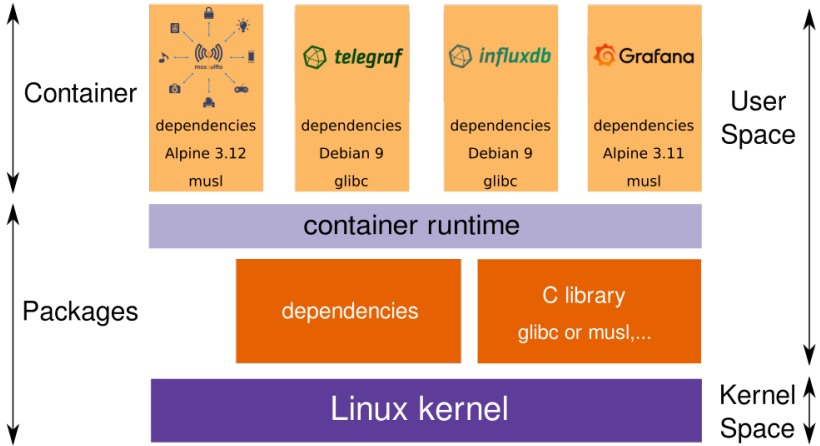


Figure: "multiple containers"

(m)TIG Stack - "Dockerfile"

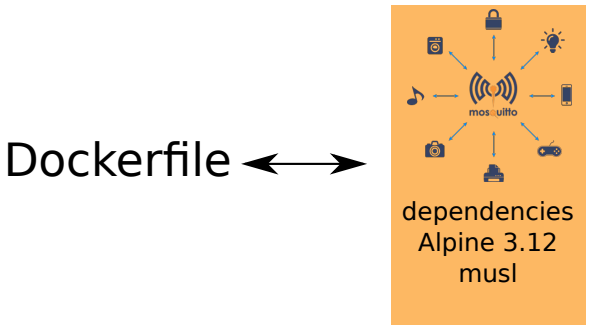


Figure: Dockerfile: mosquitto container

(m)TIG Stack - "Dockerfile"

/> on UB 18.04 kvm/machine or ARM target

Listing 1: Dockerfile: mosquitto 1/2

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

(m)TIG Stack - "Dockerfile"

/> on UB 18.04 kvm/machine or ARM target

Listing 2: Dockerfile: mosquitto 2/2

```
21      mkdir -p /mosquitto/config /mosquitto/data /mosquitto/log && \
22      install -d /usr/sbin/ && \
23      install -s -m755 /build/mosq/client/mosquitto_pub /usr/bin/mosquitto_pub && \
24      install -s -m755 /build/mosq/client/mosquitto_rr /usr/bin/mosquitto_rr && \
25      install -s -m755 /build/mosq/client/mosquitto_sub /usr/bin/mosquitto_sub && \
26      install -s -m644 /build/mosq/lib/libmosquitto.so.1 /usr/lib/libmosquitto.so.1 && \
27      install -s -m755 /build/mosq/src/mosquitto /usr/sbin/mosquitto && \
28      install -s -m755 /build/mosq/src/mosquitto_passwd /usr/bin/mosquitto_passwd && \
29      install -m644 /build/mosq/mosquitto.conf /mosquitto/config/mosquitto.conf && \
30      chown -R mosquitto:mosquitto /mosquitto && \
31      apk --no-cache add \
32          ca-certificates && \
33      apk del build-deps && \
34      rm -rf /build
35
36  VOLUME ["/mosquitto/data", "/mosquitto/log"]
37
38  # Set up the entry point script and default command
39  COPY docker-entrypoint.sh /
40  EXPOSE 1883
41  ENTRYPOINT ["/docker-entrypoint.sh"]
42  CMD ["/usr/sbin/mosquitto", "-c", "/mosquitto/config/mosquitto.conf"]
```

(m)TIG Stack - "docker-compose"

/> several containers on the same node/machine

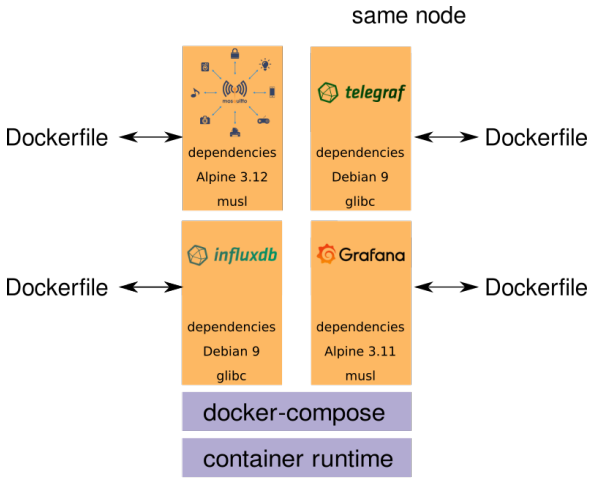


Figure: docker-compose: (m)TIG stack containers

(m)TIG Stack - "docker-compose[1]"

/> on UB 18.04 kvm/machine or ARM target

Listing 3: `docker-compose.yml` 1/3

```
1  version: "2"
2  services:
3    influxdb:
4      container_name: influxdb
5      image: influxdb
6      ports: # port on host:port in container
7        - "8086:8086"
8      volumes: # volume on host:volume in container
9        - /home/student/projects/tig/data/influxdb:/var/lib/influxdb
10       - /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/influxdb:/etc/influxdb/
11     restart: always # auto restart if crash
12   telegraf:
13     container_name: telegraf
14     image: telegraf
15     network_mode: "host" # host network
16     volumes:
17       - ↩
   /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/telegraf/telegraf.conf:/etc/telegraf/telegraf.conf
18       - /var/run/docker.sock:/var/run/docker.sock
19     restart: always
```

[1] "Good programmers copy, great programmers paste!"

https://github.com/RobertBerger/mqtt-telegraf-influxdb-grafana

(m)TIG Stack - "docker-compose"

/> on UB 18.04 kvm/machine or ARM target

Listing 4: docker-compose.yml 2/3

```
22  grafana:
23    container_name: grafana
24    image: grafana/grafana
25    user: "0" # default user is id="0" (root)
26    ports:
27      - "3000:3000"
28    volumes:
29      - /home/student/projects/tig/data/grafana:/var/lib/grafana
30      - /home/student/projects/tig/log/grafana:/var/log/grafana
31      - /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/grafana/grafana.ini:/etc/grafana/grafana.ini
32    links: # Containers for the linked service are reachable at a hostname identical to the alias,
33           # or the service name if no alias was specified.
34           # Links also express dependency between services in the same way as depends_on
35      - influxdb
36    restart: always
```

(m)TIG Stack - "docker-compose"

/> on UB 18.04 kvm/machine or ARM target

Listing 5: `docker-compose.yml` 3/3

```
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
```

(m)TIG Stack - "Config changes"

Figure: measurements

/> 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",
  ]
...
```

(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 addess>:3000

/> change default `admin` password

/> Add influxdb data source

/> Create dashboards/panels

Table of Contents - Section

Build/Install influxdb[1]

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

```
# 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

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

[1] https://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`

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

TOOLCHAIN_TARGET_TASK_append_pn-core-image-sato-sdk = " \
    ${@multilib_pkg_extend(d, 'packagegroup-go-sdk-target')} \
"
# <-- golang stuff
```

Listing 11: build "classic" SDK

```
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

[2] https://github.com/crops/extsdk-container

Build influxdb for target

Listing 12: SDK build influxdb for target 1/2

```
# Create a directory for the Go stuff - this is standard to place source code
sudo rm -rf  /workdir/sources/gocodez && mkdir -p /workdir/sources/gocodez
# Export GOPATH
export GOPATH=/workdir/sources/gocodez
# Get the source code from git
mkdir -p ${GOPATH}/src/github.com/influxdata && cd ${GOPATH}/src/github.com/influxdata
git clone https://github.com/influxdata/influxdb
# version 1.8
cd influxdb && git checkout 1.8

dep init # maybe a second time if it fails
dep ensure

CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influxd
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_inspect
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_stress
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_tools
CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_tsm

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

```
23 tree ${GOPATH}/bin
24
25 /workdir/sources/gocodez/bin
26 `-- linux_arm
27     |-- influx
28     |-- influxd
29     |-- influx_inspect
30     |-- influx_stress
31     |-- influx_tools
32     |-- influx_tsm
33     |-- stress_test_server
34     `-- test_client
```

/> yay!

Table of Contents - Section

Build package(s) from sources

Listing 14: influxdb-from-sources_1.8.0.bb 1/2

```
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"
```

Build package(s) from sources

Listing 15: influxdb-from-sources_1.8.0.bb 2/2

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

Build package(s) from sources

- 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-cespare-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

Build package(s) from prebuilt binaries

/> I searched on the layerindex[1] and started from here[2]

Listing 16: influxdb-prebuilt_1.8.0.bb

```
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', ↩
'fdac157f02e8231e1925d8e9bc325c88b7ba55ebab2340c549ef10640dbd0cba', '', 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"
```

[1] https://layers.openembedded.org/layerindex/
[2] https://layers.openembedded.org/layerindex/recipe/121293/

Build package(s) from prebuilt binaries

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/"
4
5  INSANE_SKIP_${PN}_append = "already-stripped"
6
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}/
```

Build package(s) from prebuilt binaries

Listing 18: influxdb.inc 2/2

```
26      else
27      # x86-64 does not
28      install -m 0755 ${S}/influx ${D}${bindir}/
29      install -m 0755 ${S}/influxd ${D}${bindir}/
30      install -m 0755 ${S}/influx_inspect ${D}${bindir}/
31      install -m 0755 ${S}/influx_stress ${D}${bindir}/
32      install -m 0755 ${S}/influx_tsm ${D}${bindir}/
33      fi
34      # /usr/lib
35      if [ -d ${S}/usr/lib ]; then
36      # only form armv7
37      install -d ${D}${systemd_unitdir}/system
38      sed -i 's/User=influxdb/User=root/g' ${S}/usr/lib/influxdb/scripts/influxdb.service
39      sed -i 's/Group=influxdb/Group=root/g' ${S}/usr/lib/influxdb/scripts/influxdb.service
40      install -m 0644 ${S}/usr/lib/influxdb/scripts/influxdb.service ${D}${systemd_unitdir}/system
41      fi
42      # /usr/share
43      install -d ${D}${datadir}/man/man1
44      install -m 0644 ${S}/usr/share/man/man1/* ${D}${datadir}/man/man1/
45      # /var/lib
46      install -d ${D}${localstatedir}/lib/influxdb
47      install -d ${D}${localstatedir}/log/influxdb
48  }
49  inherit systemd
50  SYSTEMD_SERVICE_${PN} = "influxdb.service"
```

Build package(s) from prebuilt binaries

- /> This was a bit easier!
- /> But
  - /» prebuilt binaries are packaged differerently depending on architecture (Why?)
  - /» how reproduce-able is this?

(m)TIG package(s) summary

- /> 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

Table of Contents - Section

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

meta-virtualization

/> add the `meta-virtualization`[1] layer

/> add some packages

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

```
require recipes-core/images/core-image-minimal.bb

# from meta-virtualization
IMAGE_INSTALL += "\
                  docker-ce \
                  docker-ce-contrib \
                  python3 \
                  python3-docker-compose \
"
# convenience
IMAGE_INSTALL += "\
                  bash \
                  btrfs-tools \
                  dropbear \
                  e2fsprogs-mke2fs \
                  udev-extraconf \
"
```

[1] http://git.yoctoproject.org/cgit/cgit.cgi/meta-virtualization/

Docker

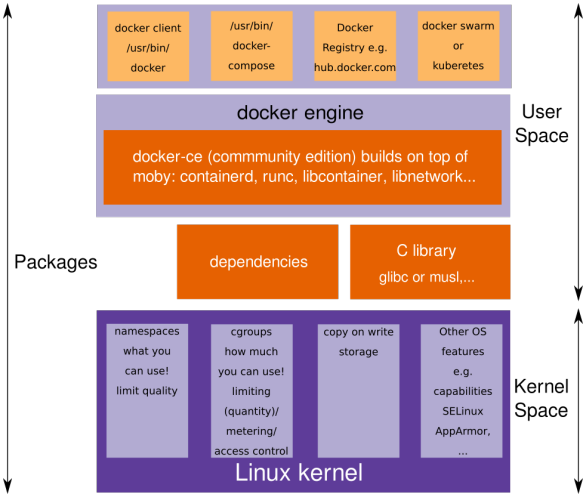


Figure: docker architecture

Configure the kernel

Listing 20: target: run check-config.sh

```
./usr/share/docker/check-config.sh

info: reading kernel config from /proc/config.gz ...

Generally Necessary:
- cgroup hierarchy: properly mounted [/sys/fs/cgroup]
- CONFIG_NAMESPACES: enabled
- CONFIG_NET_NS: enabled
- CONFIG_PID_NS: enabled
- CONFIG_IPC_NS: enabled
- CONFIG_UTS_NS: enabled
- CONFIG_CGROUPS: enabled
- CONFIG_CGROUP_CPUACCT: enabled
- CONFIG_CGROUP_DEVICE: enabled
- CONFIG_CGROUP_FREEZER: enabled
- CONFIG_CGROUP_SCHED: enabled
- CONFIG_CPUSETS: enabled
- CONFIG_MEMCG: enabled
- CONFIG_KEYS: enabled
- CONFIG_VETH: enabled
- CONFIG_BRIDGE: enabled (as module)
- CONFIG_BRIDGE_NETFILTER: enabled (as module)
- CONFIG_NF_NAT_IPV4: missing
- CONFIG_IP_NF_FILTER: enabled (as module)
- CONFIG_IP_NF_TARGET_MASQUERADE: enabled (as module)
...
```

(m)TIG Stack - "container" - Microservices

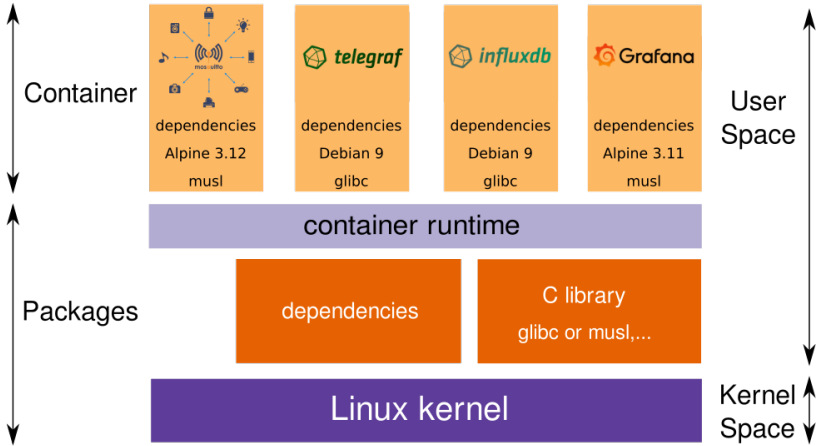


Figure: "multiple containers"

Table of Contents - Section

(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 addess>: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

Table of Contents - Section

Container Standardization

- OCI (Open Container Initiative): https://opencontainers.org/
  - 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
    - lifecycle running container
    - how a tool executing a container must behave and interact with it

Table of Contents - Section

app-container-image-influxdb-prebuilt-oci

- /> 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

```
1 SUMMARY = "An influxdb image"
2 LICENSE = "MIT"
3 LIC_FILES_CHECKSUM = "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

```
1 # Note that busybox is conventient and might be required as well by /bin/sh
2 IMAGE_INSTALL += " \
3   busybox \
4   influxdb \
5 "
```

app-container-image-influxdb-prebuilt-oci

Listing 24: app-container-image-oci.bb 1/2

```
SUMMARY = "A minimal container image"
LICENSE = "MIT"
LIC_FILES_CHECKSUM = "file://${COREBASE}/meta/CONFIG.MIT;md5=1234"

IMAGE_FSTYPES = "container oci"

inherit image
inherit image-oci

IMAGE_FEATURES = ""
IMAGE_LINGUAS = ""
NO_RECOMMENDATIONS = "1"

# Allow build with or without a specific kernel
# see poky/meta/classes/image-container.bbclass
IMAGE_CONTAINER_NO_DUMMY = "0" # DUMMY = 1 ;)
# 0 -> no kernel, PREFERRED_PROVIDER_virtual/kernel =  linux-dummy
# 1 -> kernel,    PREFERRED_PROVIDER_virtual/kernel != linux-dummy

IMAGE_INSTALL = "\
  base-files \
  base-passwd \
  netbase \
"
```

app-container-image-influxdb-prebuilt-oci

Listing 25: app-container-image-oci.bb 2/2

```
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

[1] https://www.youtube.com/watch?v=OSyLoHYxGLQ

app-container-image-influxdb-prebuilt-oci

/> 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

app-container-image-influxdb-prebuilt-oci

Listing 28: host: which docker containers are running?

```
1 docker ps -a
2 CONTAINER ID  IMAGE                          COMMAND       CREATED     STATUS                  NAMES
3 3d649b14f4aa  yoctotrainer/app-container... "/bin/ash -l" 32 sec ago  Exited (0) 2 sec ago   #yoctoizing
```

Listing 29: host: `docker run ash`

```
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
```

app-container-image-influxdb-prebuilt-oci

Listing 30: in container

```
root@bc41060a6bcc:/# influxd

 8888888           .d888 888                   8888888b.  888888b.
   888            d88P"  888                   888  "Y88b 888  "88b
   888            888    888                   888    888 888  .88P
   888   88888b.  888888 888 888  888 888  888 888    888 8888888K.
   888   888 "88b 888    888 888  888  Y8bd8P' 888    888 888  "Y88b
   888   888  888 888    888 888  888   X88K   888    888 888    888
   888   888  888 888    888 Y88b 888 .d8""8b. 888  .d88P 888   d88P
 8888888 888  888 888    888  "Y88888 888  888 8888888P"  8888888P"

2020-06-21T11:55:25.906737Z     info    InfluxDB starting       {"log_id": "0NXg_RKW000", "version": "1.8.0", "branch": "1.8", "commit": "781490de48220d7695a05c29e5a36f550a4568f5"}
2020-06-21T11:55:25.906793Z     info    Go runtime      {"log_id": "0NXg_RKW000", "version": "go1.13.8", "maxprocs": 6}
2020-06-21T11:55:26.036380Z     info    Using data dir  {"log_id": "0NXg_RKW000", "service": "store", "path": "/var/lib/influxdb/data"}
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}
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"}
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"}
2020-06-21T11:55:26.040510Z     info    Opened service  {"log_id": "0NXg_RKW000", "service": "subscriber"}
```

app-container-image-influxdb-prebuilt-oci

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}
```

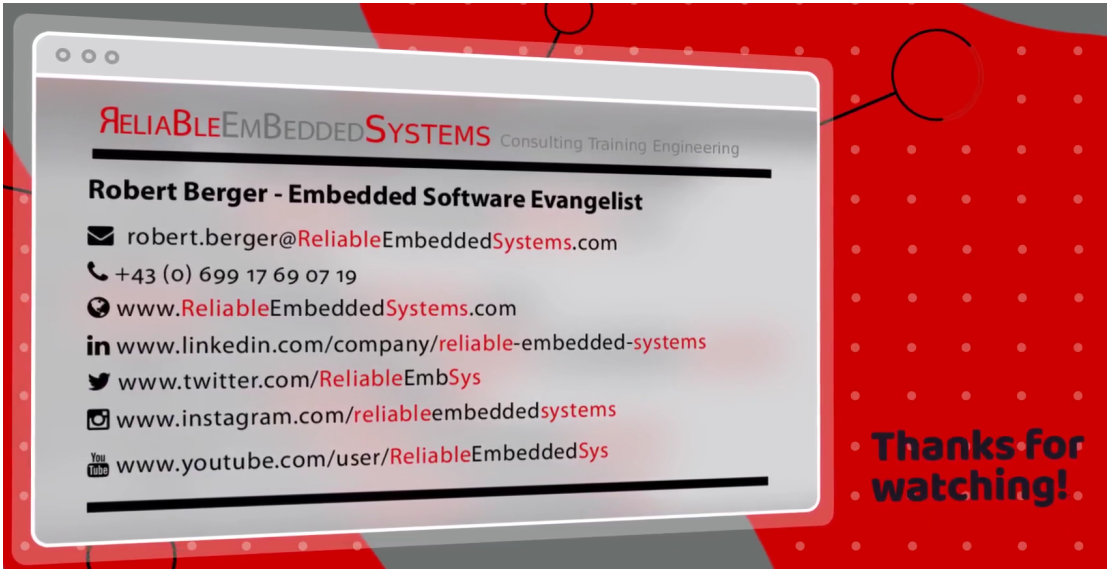
Further work

- 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`
- distibuted architecture via `kubernetes`

Table of Contents - Section

Thank you

Thank you!


ReliaBleEmBeddedSystems Consulting Training Engineering
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
Thanks for watching!