

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

RELIABLEEMBEDDED**SYSTEMS** 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://rbl.me/yocto-en-pdf>)

/» Refresher to Embedded Linux & Intro to the Yocto Project - 5 days -

(<http://rbl.me/intely-en-pdf>)

/> engineering

We trained (excerpt)



Table of Contents - Section



The use case - on host (& target)

Hardware

Software

Host: (m)TIG Stack - "classic"

Host: (m)TIG Stack - "vm"

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

(remote) Lab in Austria



- /> Pain point: "investigate"/reduce
 - /» electricity bill
- /> installed several sensors
 - /» Sonoff POWR2 Modules² (power measurement)
 - /» Tasmota alternative firmware for ESP8266³ - mqtt⁴

²<https://sonoff.tech/product/wifi-diy-smart-switches/powr2>

³<https://github.com/arendst/Tasmota>

⁴<http://mqtt.org/>



(m)TIG Stack

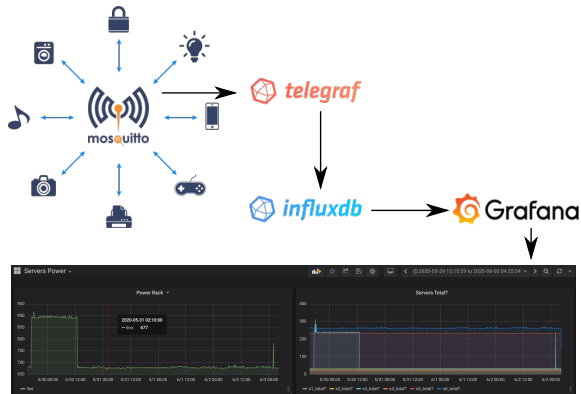


Figure: mTIG stack

/> 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"

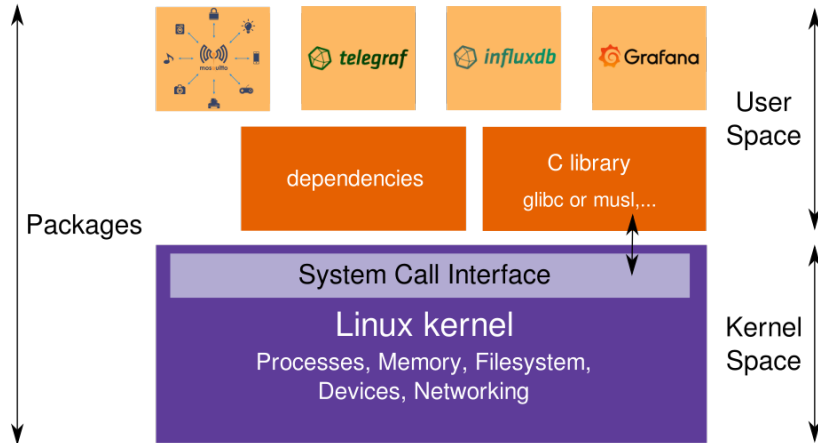


Figure: "classic"



(m)TIG Stack - "vm"

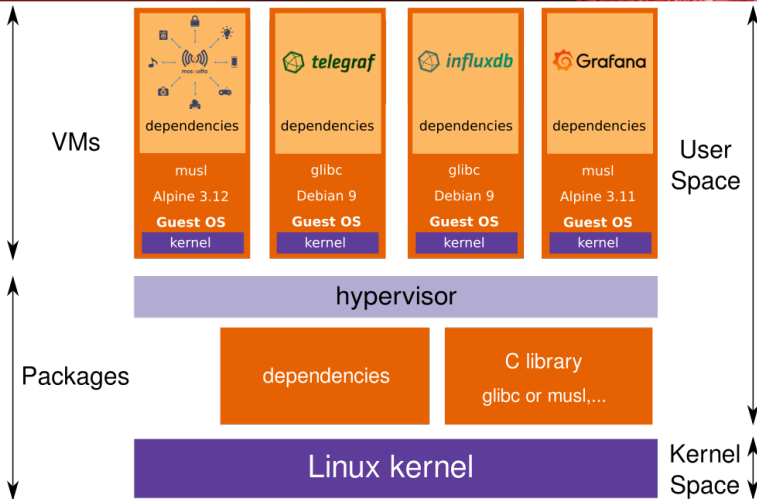


Figure: "multiple vms"



(m)TIG Stack - "container" - Microservices

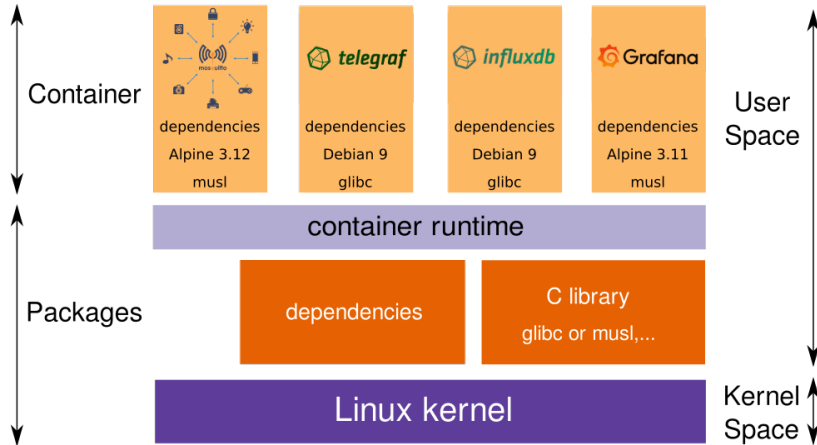


Figure: "multiple containers"



(m)TIG Stack - "Dockerfile"

Dockerfile ↔



Figure: Dockerfile: mosquitto container



(m)TIG Stack - "Dockerfile"

/> on UB 18.04 kvm/machine or ARM target

Listing 1: Dockerfile: mosquito 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${LWS_VERSION}.tar.gz -O /tmp/lws.tar.gz && \
13     mkdir -p /build/lws && tar --strip=1 -xf /tmp/lws.tar.gz -C /build/lws && \
14     rm /tmp/lws.tar.gz && cd /build/lws && \
15     cmake . -DCMAKE_BUILD_TYPE=MinSizeRel -DCMAKE_INSTALL_PREFIX=/usr -DLWS_IPV6=ON ↵
16     -DLWS_WITHOUT_BUILTIN_GETIFADDRS=ON -DLWS_WITHOUT_CLIENT=ON -DLWS_WITHOUT_EXTENSIONS=ON ↵
17     -DLWS_WITHOUT_TESTAPPS=ON -DLWS_WITH_SHARED=OFF -DLWS_WITH_ZIP_FOPS=OFF -DLWS_WITH_ZLIB=OFF && \
18     make -j "$(nproc)" && rm -rf /root/.cmake && mkdir -p /build/mosq && \
19     tar --strip=1 -xf /tmp/mosq.tar.gz -C /build/mosq && rm /tmp/mosq.tar.gz && \
20     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 && \
    addgroup -S -g 1883 mosquitto 2>/dev/null && \
    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: mosquito 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

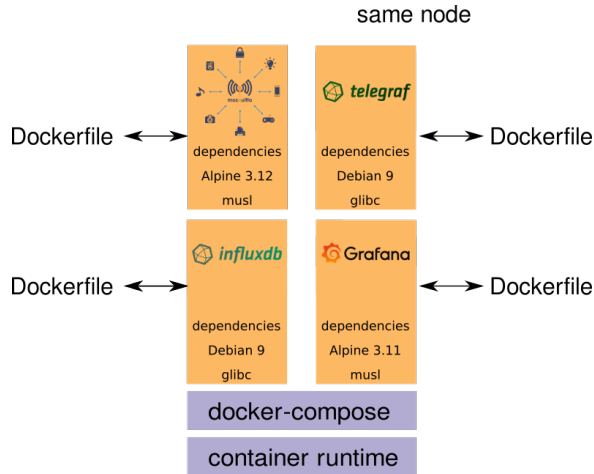


Figure: docker-compose: (m)TIG stack containers



(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      - ←
18      /home/student/projects/mqtt-telegraf-influxdb-grafana/conf/telegraf/telegraf.conf:/etc/telegraf/telegraf.conf
19      - /var/run/docker.sock:/var/run/docker.sock
20    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

```
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"



/> on UB 18.04 kvm/machine or ARM target

Listing 6: telegraf.conf excerpt

```
1 ...
2 [[inputs.mqtt_consumer]]
3   servers = ["tcp://mqttbrk1.res.training:1883"]
4   qos = 0
5   connection_timeout = "30s"
6   topics = [
7     "tele/line/office/ac/SENSOR",
8     "tele/temp/office/ac/SENSOR",
9   ]
10 ...
```

Figure: measurements



(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

Table of Contents - Section



Developer: Get app(s) on the Target

The Golang approach

#yoctoizing The Golang approach with Yocto SDK



Build/Install influxdb¹

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

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

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

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

¹<https://hwwong168.wordpress.com/2015/10/12/>

Add Golang support to Yocto SDK



Listing 10: add to local.conf

```
1 # --> golang stuff
2 # attempt to add golang to SDK
3 TOOLCHAIN_HOST_TASK_append_pn-core-image-sato-sdk = " \
4     packagegroup-go-cross-canadian-${MACHINE} \
5 "
6
7 TOOLCHAIN_TARGET_TASK_append_pn-core-image-sato-sdk = " \
8     ${@multilib_pkg_extend(d, 'packagegroup-go-sdk-target')} \
9 "
10 # <-- golang stuff
```

Listing 11: build "classic" SDK

```
1 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 container²

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

```
1 # Create a directory for the Go stuff - this is standard to place source code
2 sudo rm -rf /workdir/sources/gocodez && mkdir -p /workdir/sources/gocodez
3 # Export GOPATH
4 export GOPATH=/workdir/sources/gocodez
5 # Get the source code from git
6 mkdir -p ${GOPATH}/src/github.com/influxdata && cd ${GOPATH}/src/github.com/influxdata
7 git clone https://github.com/influxdata/influxdb
8 # version 1.8
9 cd influxdb && git checkout 1.8
10
11 dep init # maybe a second time if it fails
12 dep ensure
13
14 CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx
15 CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influxd
16 CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_inspect
17 CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_stress
18 CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_tools
19 CGO_ENABLED=1 GOOS=linux GOARCH=arm GOARM=7 ${TARGET_PREFIX}go build ./cmd/influx_tsm
20
21 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



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



Build package(s) from sources

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

```
1 DESCRIPTION = "InfluxDB is an open source time series platform. This includes APIs for storing and querying \  
2 data, processing it in the background for ETL or monitoring and alerting purposes, user \  
3 dashboards, and visualizing and exploring the data and more. The master branch on this repo \  
4 now represents the latest InfluxDB, which now includes functionality for Kapacitor (background \  
5 processing) and Chronograf (the UI) all in a single binary."  
6  
7 GO_IMPORT = "github.com/influxdata/influxdb"  
8 SRC_URI = "git://${GO_IMPORT};branch=1.8"  
9 SRCREV = "781490de48220d7695a05c29e5a36f550a4568f5"  
10  
11 LICENSE = "MIT"  
12 LIC_FILES_CHKSUM = "file://${WORKDIR}/${PN}-${PV}/src/${GO_IMPORT}/LICENSE;md5=f39a8d10930fb37bd59adabb3b9d0bd6"  
13 inherit go  
14 CGO_ENABLED = "1"  
15  
16 DEPENDS += "go-dep-native"  
17  
18 RDEPENDS_${PN}-dev += "python bash"  
19  
20 RDEPENDS_${PN}-staticdev += "python bash perl make"  
21  
22 DEPENDS += "github.com-cespare-xxhash"  
23 DEPENDS += "code.google.com-gocloud"  
24  
25 RDEPENDS_${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 vlreflect.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.google.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 layerindex¹ and started from here²

Listing 16: influxdb-prebuilt_1.8.0.bb

```
1 require influxdb.inc
2
3 SRC_URI_armv7a = "https://dl.influxdata.com/influxdb/releases/influxdb-${PV}_linux_armhf.tar.gz \
4                 file://LICENSE"
5 SRC_URI[sha256sum] = "${@bb.utils.contains('MACHINEOVERRIDES', 'armv7a', ←
6 'fdac157f02e8231e1925d8e9bc325c88b7ba55ebab2340c549ef10640dbd0cba', '', d)}"
7 SRC_URI_x86-64 = "https://dl.influxdata.com/influxdb/releases/influxdb-${PV}-static_linux_amd64.tar.gz \
8                 file://LICENSE"
9 SRC_URI[sha256sum] = "${@bb.utils.contains('MACHINEOVERRIDES', 'x86-64', ←
10 'aedc5083ae2e61ef374dbde5044ec2a5b27300e73eb92ccd135e6ff9844617e2', '', d)}"
11 LIC_FILES_CHKSUM = "file://${WORKDIR}/LICENSE;md5=f39a8d10930fb37bd59adabb3b9d0bd6"
12
13 # The open source license for InfluxDB is available in the GitHub repository.
14 # https://github.com/influxdata/influxdb/blob/1.8/LICENSE
15 # I copied LICENSE from there to my meta data
16 LICENSE = "MIT"
17
18 S = "${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 differently 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



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



meta-virtualization

/> add the meta-virtualization¹ layer

/> add some packages

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

```
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 "
```

¹<http://git.yoctoproject.org/cgit/cgit.cgi/meta-virtualization/>

Docker

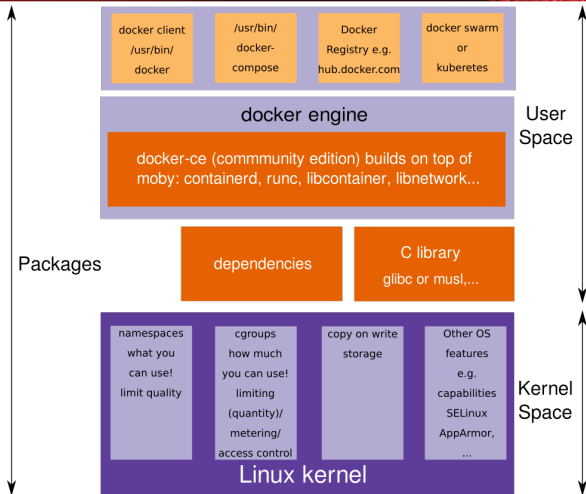


Figure: docker architecture

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_CPUSETS: enabled
18 - CONFIG_MEMCG: enabled
19 - CONFIG_KEYS: enabled
20 - CONFIG_VETH: enabled
21 - CONFIG_BRIDGE: enabled (as module)
22 - CONFIG_BRIDGE_NETFILTER: enabled (as module)
23 - CONFIG_NF_NAT_IPV4: missing
24 - CONFIG_IP_NF_FILTER: enabled (as module)
25 - CONFIG_IP_NF_TARGET_MASQUERADE: enabled (as module)
26 ...
```

(m)TIG Stack - "container" - Microservices

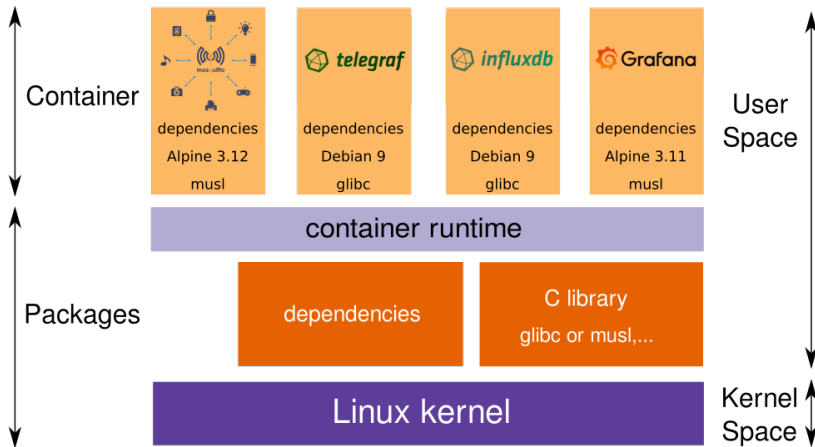


Figure: "multiple containers"



Table of Contents - Section

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"

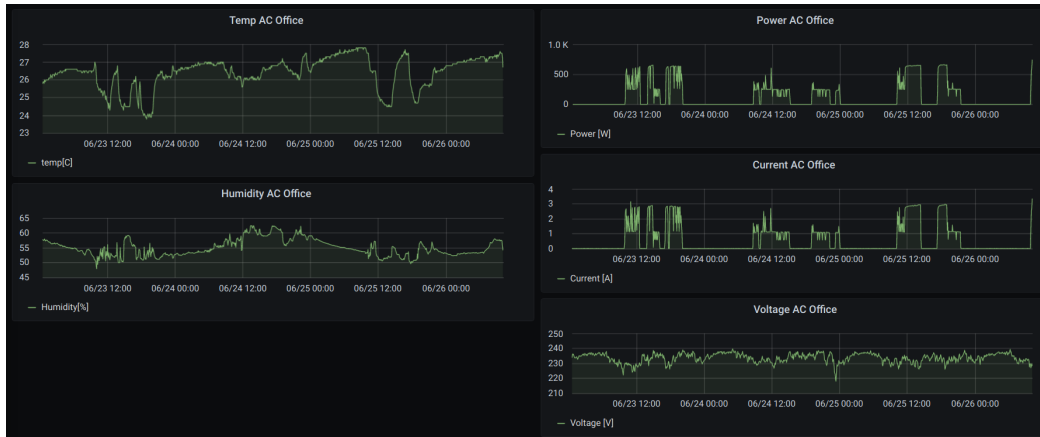


Figure:)m)TIG running

Put it all together



yocto .
PROJECT

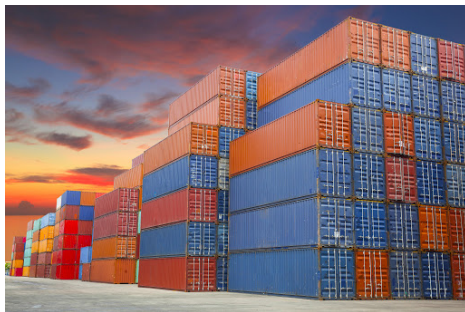


Table of Contents - Section



Container Standardization

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



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



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_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

```
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

```
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     "
```

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 "skopeo² 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

```

1 root@bc41060a6bcc:/# influxd
2
3 88888888 .d888 888 88888888b. 8888888b. 8888888b.
4 888      d88P" 888      888 "Y88b 888 "88b
5 888      888 888      888      888 888 .88P
6 888 888888b. 8888888 888 888 888 888 888 888 88888888K.
7 888 888 "88b 888 888 888 888 Y8bd8P' 888 888 888 "Y88b
8 888 888 888 888 888 888 888 X88K 888 888 888 888
9 888 888 888 888 888 Y88b 888 .d8""8b. 888 .d88P 888 d88P
10 88888888 888 888 888 888 "Y88888 888 888 88888888P" 88888888P"
11
12 2020-06-21T11:55:25.906737Z info InfluxDB starting {"log_id": "0NXg_RKW000", "version": "1.8.0", ←
"branch": "1.8", "commit": "781490de48220d7695a05c29e5a36f550a4568f5"}
13 2020-06-21T11:55:25.906793Z info Go runtime {"log_id": "0NXg_RKW000", "version": "go1.13.8", ←
"maxprocs": 6}
14 2020-06-21T11:55:26.036380Z info Using data dir {"log_id": "0NXg_RKW000", "service": "store", "path": ←
"/var/lib/influxdb/data"}
15 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}
16 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"}
17 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"}
18 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-image-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!

RELIABLEEMBEDDED**SYSTEMS** Consulting Training Engineering

Robert Berger - Embedded Software Evangelist

✉ robert.berger@ReliableEmbeddedSystems.com

☎ +43 (0) 699 17 69 07 19

🌐 www.ReliableEmbeddedSystems.com

in www.linkedin.com/company/reliable-embedded-systems

🐦 www.twitter.com/ReliableEmbSys

📷 www.instagram.com/reliableembeddedsystems

You Tube www.youtube.com/user/ReliableEmbeddedSys

**Thanks for
watching!**