Fetching, Configuring and Building Your Bitbake Project with Just One Command
Who Are We, Who am I?

Embedded Linux Competence Center / Linux Expert Center
@ Siemens Technology

• In-house embedded Linux consultants
• Linux enablers in many Siemens products
• Gateway to upstream communities (not exclusively!)
• Maintainers for several OSS projects

Jan Kiszka
• Crew member for almost 16 years
• Navigating that ship technically
• Maintainer of (too?) many OSS projects
Quick Survey

Who knows kas?

Who is using it already?

Who is using it via kas-container?

Who is using it for isar layers?
Time warp to 2017: How to replicate a Yocto-based build?

1. **Fetch a few repos**
   - manually according to some README
   - with the help of some custom script
   - using the repo tool

2. **Create bblayer.conf and local.conf**
   - manually according to some README
   - with the help of some custom script
   - using the right template during oe-init-build-env (see README)

3. **Initialize the build env**

4. **bitbake my-target** (please consult the README on which one)

5. *Pray that you didn’t miss anything*

6. *Find out that host distro and user settings can matter...*
And that is...

For Non-Bavarians
1. Cheese
2. Nonsense, a silly thing

Kas, der

Aussprache: [kà:s]
1. Käse (...a Kas, a Brezn und a Mass Bier is was Guads!)
2. Blödsinn, dummies Zeug, Unsinn (...red doch ned so an Kas!)

Source (with friendly permissions): https://www.bayrisches-woerterbuch.de/kas-kaes-der/
kas!
A Short History

First commit
• March 2017 – Daniel Wagner @Siemens

First public release
• June 2017 – 0.9.0

First non-Siemens contribution
• January 2018 – thanks, Georg Lutz!

"We are stable" release
• March 2019 – 1.0

First maintainer hand-over
• July 2019 – Daniel prefers Enterprise over Embedded ;-)

First larger external contributions
• 2020 – thanks, Paul Barker!

Latest release
• June 2023 – 4.0 with Debian bookworm containers
Key Idea: Single-Statement Setup & Build

```
git clone <bootstrap-repo>

kas build kas-target.yaml

or

kas-container build kas-target.yaml

or

kas/kas-container menu
```
A Basic kas Configuration File

```plaintext
header:
  version: 14
distro: poky
machine: qemu86-64
target: core-image-minimal
repos:
  poky:
    url: https://git.yoctoproject.org/poky.git
    commit: 31dd418207f6c95ef0aad589cd03cd2a4c9a8bf2
layers:
  meta:
    meta-poky:
local_conf_header:
  some-tweaks: |
    EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
    INHERIT += "rm_work"
```
Interacting with bitbake

**Building a specific target**

```
# kas build kas.yaml --target other-target
```

**Running up to a specific task**

```
# kas build kas.yaml --target some-target --cmd compile
```

**Forwarding additional bitbake arguments**

```
# kas build kas.yaml -- --environment
```

**Dropping into the build environment**

```
# kas shell kas.yaml
```
Including and Overriding with kas

```yaml
header:
  version: 14
  includes:
    - base.yaml
machine: my-machine
repos:
poky:
  layers:
    meta-yocto-bsp:

[base.yaml]
header:
  version: 14
machine: qemux86-64
repos:
poky:
  url: ...
  layers:
    meta:
      meta-poky:
        meta-yocto-bsp:

Merging and overriding of includes happen top to bottom and depth first.
```
Ad-hoc Inclusion

```
[main.yaml]
header:
    version: 14
distro: poky
machine: qemux86-64
target: core-image-minimal
repos: ...

[machine-a.yaml]
header: ...
machine: my-machine-a

[feature-b.yaml]
header: ...
local_conf_header:
    feature-b: |
    FEATURE_B = "1"
```

# kas build main.yaml: machine-a.yaml: feature-b.yaml

Equivalent to rewriting main.yaml:

```
header:
    version: 14
    includes:
    - machine-a.yaml
    - feature-b.yaml
distro: poky
target: core-image-minimal
repos: ...
```
Interactive Inclusion / Basic Configuration via Kconfig

Problem
• How to make available options discoverable?
• How to guide to supported combinations?

kas menu plugin
• Build up menu using Kconfig language
• Add minimal extra semantic to generate a kas config file: .config.yaml
• Build .config.yaml directly

The pros
• Reuse of an existing, powerful language and library
• Proved to resolve above problems

The cons
• Kconfig mapping on kas not "native", not fully intuitive
• Misuse possible – limit the number of knobs!
Exampelry Kconfig File

mainmenu "My kas menu"
config KAS_INCLUDE_MAIN
  string
  default "main.yaml"
choice
  prompt "Select machine"
  default MACHINE_A
config MACHINE_A
  bool "Machine A"
  help
  This is machine A.
config MACHINE_QEMUX86_64
  bool "QEMU x86-64"
endchoice
config KAS_INCLUDE_MACHINE
  string
  default "machine-a.yaml" if MACHINE_A

config FEATURE_B
  bool "Feature B"
  depends on !MACHINE_QEMUX86_64
config KAS_INCLUDE_FEATURE_B
  string
  default "feature-b.yaml"
  depends on FEATURE_B
config IMAGE_ROOTFS_EXTRA_SPACE
  string "Specify extra rootfs space"
  default "131072"

[.config.yaml]
header:
  includes:
    - main.yaml
    - machine-a.yaml
    - feature-b.yaml
  version: 14
local_conf_header:
  __menu_config_vars: IMAGE_ROOTFS_EXTRA_SPACE = "131072"
Repository Update Workflows

Option A: manual update
• Bump repo revisions based on relevant upstream changes
• Works will for smaller number or well-known repos
• Common pattern in Isar ecosystem (so far)

Option B: ride branches, lock-down tested states
• Leave out repo commits, specify branch
• Lock down checkout via
  # kas dump <my-kas>.yaml --lock --inplace
• Check in generated <my-kas>.lock.yaml
• Update revisions via
  # kas dump <my-kas>.yaml --lock --inplace --update

Repo patching
• Helps testing to-be-upstreamed patches
• Not for abuse!

... repos:
  poky:
    url: ...
    branch: mickledore
    layers: ...
  meta-openembedded:
    url: ...
    branch: mickledore
    layers: ...
  patches:
    01-some-class-fix:
      path: upstream-fix.patch
Repo Integrity Protection

Problem: refspec was not sufficient if attacker hijacks repo

- On collision, old kas preferred branches over commit hashes
- Attacker may also remove commit and create branch instead
- SHA1 collisions?

Multiple attempts to address this in kas

1. git archive | sha256sum
2. Massaged "git archive"
3. Own, stable "archive" format – but proprietary

Postponed after community discussion

- Rather an upstream SCM problem?
- How realistic are SHA1 collisions for git/hg already?

From kas 3.3 onward

- Split refspec into commit and branch
- Keep refspec for now for backward compatibility
kas and The Containers

**kas comes with two build containers**
- ghcr.io/siemens/kas/kas adds OE build dependencies
- ghcr.io/siemens/kas/kas-isan adds isar build dependencies
- All contain kas plus some support tools for connectivity & downloads

**Self-contained wrapper script kas-container**
- Basically, behaves like kas
- Decouples host distro from build environment
- Locks-down build env version, e.g., by carrying kas-container in bootstrap repo
- Works with docker and podman, of course!
- Works on arm64 hosts as well
The kas Sandbox

Strong decoupling from host

- Separate `/home` to build from, even without container
- Only declared environment variables are forwarded, check
  - `https://kas.readthedocs.io/en/latest/userguide.html#project-configuration`, `env` property
- Only weakened when doing
  - `# kas shell/for-all-repos --preserve-env`
- Maybe the most disliked kas feature?

Background

- Host contamination was easy with OE in 2017...
- Foster easy migration between local and CI builds
- Also needed to maintain kas `<=>` kas-container
How About bitbake-layer Extensions?

create-layers-setup
- Generates out of configured and check-out layers
  - setup-layers.json
  - setup-layers (self-contained check-out script)

save-build-conf
- Generates template files for oe-init-build-env

Pros
- Embeds into native Yocto/OE workflows

Cons
- Doesn't address coarse-grained & interactive configurations (yet?)
- No cross-layer reuse of configuration bits
- No layer patching support
- No support for older OE / isar
- Host OS decoupling not in scope
Summary and Outlook

**kas makes your bitbake builds easier portable**
- Single line bootstrap & build
- Stronger decoupling from host
- Stable build containers
- Interactive coarse-grained build configuration

**What’s next?**
- Test containers?
- Scope down where bitbake-layer can take over?
- Your call!

**Resources**
- https://github.com/siemens/kas
- https://kas.readthedocs.io
- kas-devel@googlegroups.com

**Some examples**
- https://git.yoctoproject.org/meta-arm/tree/kas
- https://gitlab.com/cip-project/cip-core/isar-cip-core
- https://github.com/siemens/meta-iot2050
Contact

Published by Siemens Technology

Jan Kiszka
Principal Key Expert
T CED
Otto-Hahn-Ring 6
81739 Munich
Germany

E-mail jan.kiszka@siemens.com
https://opensource.siemens.com