

Un aperçu des défis reliés à l'utilisation de Linux en embarqué

***Linux-Meetup
Montréal***



Speaker notes

Bonjour!

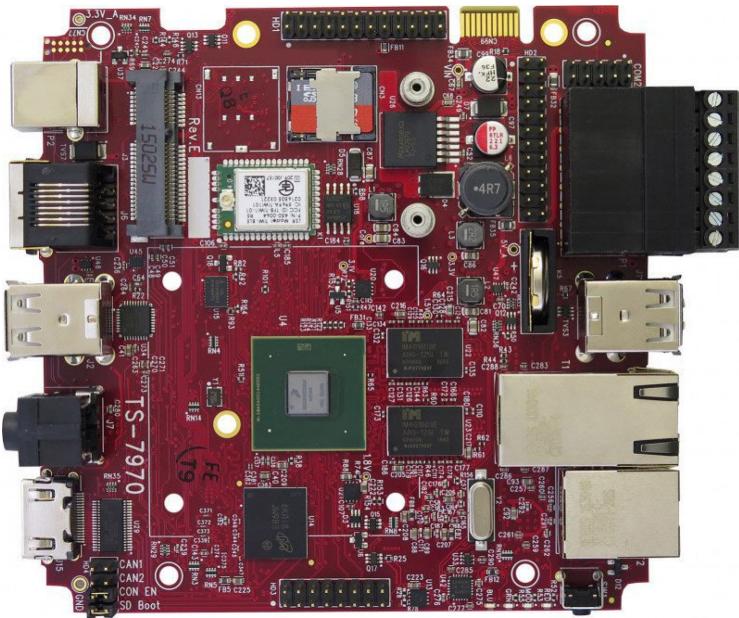
Pico	p	1960	10^{-12}	$1\ 000^{-4}$	0,000 000 000 001	Billionième	Trillionième
Femto	f	1964	10^{-15}	$1\ 000^{-5}$	0,000 000 000 000 001	Billardième	Quadrillionième
Atto	a	1964	10^{-18}	$1\ 000^{-6}$	0,000 000 000 000 000 001	Trillionième	Quintillionième
Zepto	z	1991	10^{-21}	$1\ 000^{-7}$	0,000 000 000 000 000 000 001	Trilliardième	Sextillionième
Yocto	y	1991	10^{-24}	$1\ 000^{-8}$	0,000 000 000 000 000 000 000 001	Quadrillionième	Septillionième
Ronto	r	2022	10^{-27}	$1\ 000^{-9}$	0,000 000 000 000 000 000 000 000 001	Quadrilliardième	Octillionième
Quecto	q	2022	10^{-30}	$1\ 000^{-10}$	0,000 000 000 000 000 000 000 000 000 001	Quintillionième	Nonillionième

source: [wikipedia](#)

Speaker notes

- Préfixe du système métrique.
- Le très petit, un système de fichier racine minimal.
- Deux autres préfixes ajoutés en 2022

INGÉNIERIE PRODUIT



- Linux embarqué du bootloader à l'application
- Pilote logiciel pour du matériel sur mesure
- Stack graphique, audio, WiFi, réseau
- Temps réel
- Stratégie de mise à jour
- Secure boot

Speaker notes

Présenter l'équipe d'ingénierie produit de SFL.

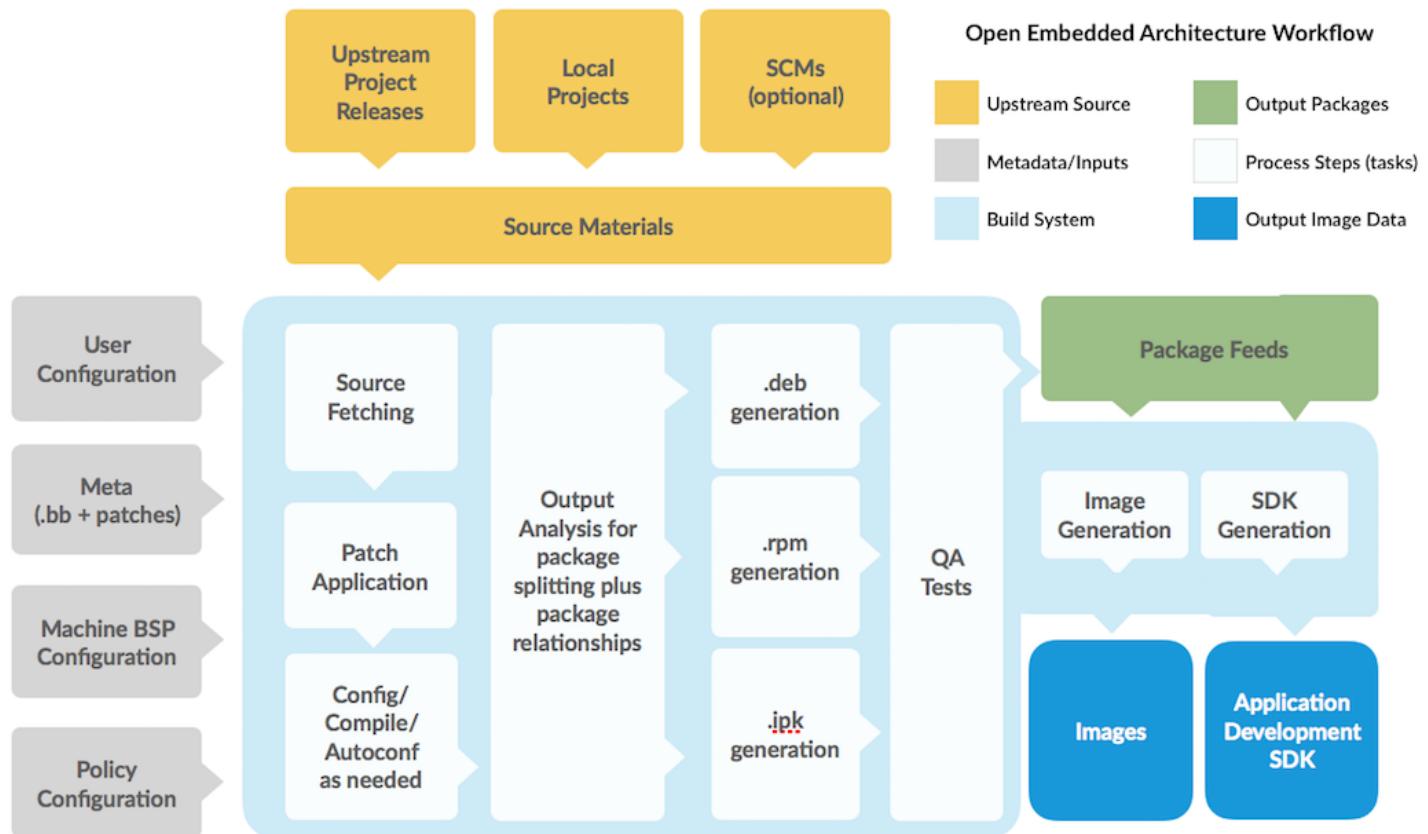
yocto . P R O J E C T

*"It's not an embedded Linux distribution, it
creates a custom one for you."*

4

Speaker notes

- Tout est compilé à partir des sources.
- poky est la distro de référence.
- Yocto permet de personnaliser absolument tout, on peut changer chaque choix fait par poky.

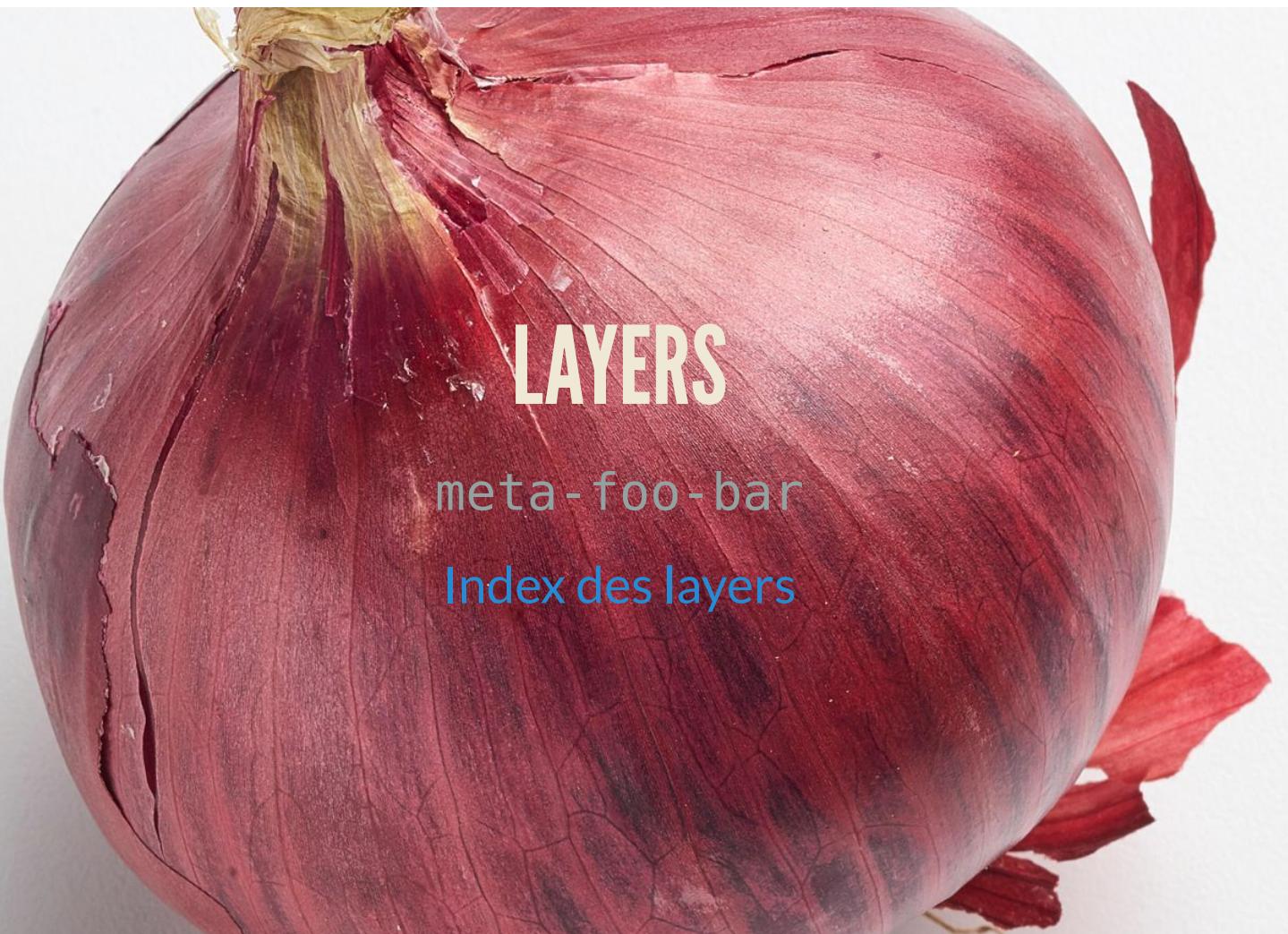


source: yoctoproject.org

5

Speaker notes

- En bleu pâle au milieu: bitbake, l'ordonnanceur de tâches
- En gris à gauche: les Métadonnées (recettes, configuration, patchs)
- En jaune en haut: code source des différents composants logiciel. Téléchargement pendant le build (do_fetch)
- En bleu foncé en bas à droite: le résultat de compilation, les packages, les images en différents formats, la toolchain.

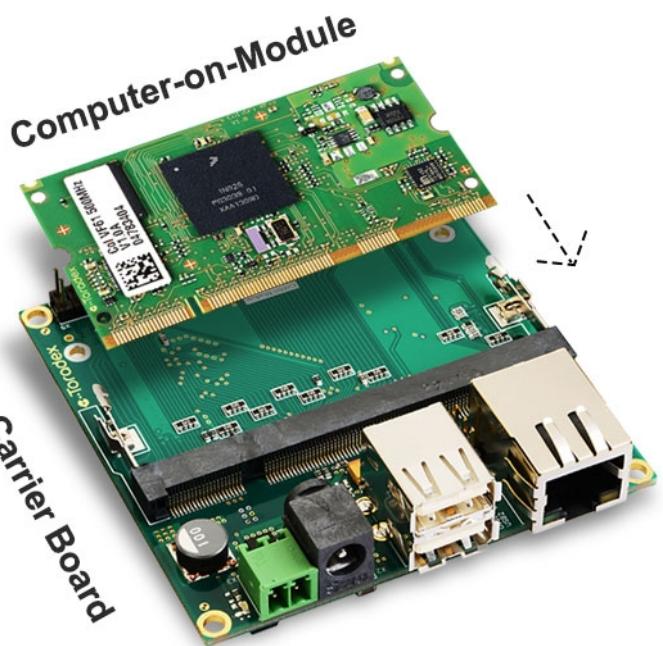


6

Speaker notes

- Les Métadonnées (en gris à gauche sur la dernière slide) sont organisées en "Layers" ou couche
- L'organisation en couche est ce qui fait de Yocto un écosystème ouvert plutôt qu'un système fermé
- Les couches sont maintenus dans différents repos, chaque joueur de l'écosystème maintient ses couches.
- Typiquement: `meta-*`
- Yocto maintient un index des layers

LAYERS MATÉRIEL



- poky (ARM)
- meta - freescale (SoC)
- meta - toradex - nxp (SoM)
- meta - my - custom - hardware (Carte mère)

Speaker notes

- L'arbre est dans ses feuilles marilon, marilé...
- Le CPU est dans le SoC, le SoC est dans le SoM, ...
- Favorise la réutilisation du code des metalayers.
- Ce qu'on retrouve dans un layer BSP: configuration pour le noyau et bootloader, firmwares, activation d'options spécifiques au matériel dans certain packages (GPU, encodage vidéo).

LAYERS LOGICIEL

- meta-qt5
- meta-nodejs
- meta-tensorflow
- meta-doom

Speaker notes

- Layers logiciel aussi maintenu dans des repos séparés. Favorise la composabilité.
- Exemples: gros framework comme Qt5 ou Tensorflow, un language interprété comme NodeJS ou Java, doom?!
- toutes les combinaisons ne sont toutefois pas testées.

METADATA

image.bb

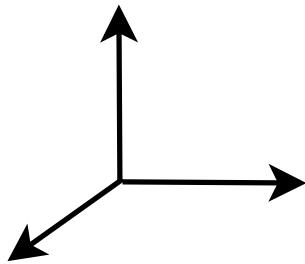
- * Choix des packages
- * Choix packages -dev, -dbg, -doc
- * Format du rootfs (ext4, squashfs, cpio)
- * Génération d'une image disque

DISTRO.conf

- * ABI
- * Compilateur
- * libc
- * Init system, e.g: systemd SysVinit
- * Package versions
- * PACKAGECONFIG tweaks

MACHINE.conf

- * -mtune
- * Choix du noyau/bootloader
- * Device tree
- * MACHINE_FEATURES
- * Packages propres au hardware,
e.g.: modules GPU, firmware WiFi



Speaker notes

- Un système yocto comprend 3 axes.
- L'image, la machine et la distro.
- *Présenter chacun de ces 3 éléments.*
- Exemple: poky, qui est une distro riche, peut être utilisé pour compiler core-image-minimal et poky-tiny pourrait être utilisé pour compiler core-image-full.
- En théorie orthogonal, en pratique, il y a du couplage. Exemple une image -weston vas avoir besoin d'une distro -weston.

UTILISATEURS



10

Speaker notes

- AGL: automotive grade Linux
- BMC: Baseboard Management Controller
- LG webOS: TV intelligente

Seulement le pic de l'iceberg, étant open source, il est impossible de connaître tous les utilisateurs.



REPRODUCTIBILITÉ

11

Speaker notes

- La reproductibilité n'est pas propre à l'embarqué, ça touche tous les sphères de l'informatique.
- Les frameworks d'infrastructure en tant que code comme Ansible ou Puppet apportent des solutions aux problématiques de reproductibilité dans le monde du serveur
- Similairement, Yocto apporte une solution pour le domaine de l'embarqué.

UN CAS TYPIQUE EN EMBARQUÉ

"Install Debian and add those files..."

```
└── bin
    ├── pcs
    ├── pim.db
    └── readme
    └── Install.sh
    └── gpio
        └── 99-gpio.rules
    └── lib
        ├── libmodbus.la
        ├── libmodbus.so
        ├── libmodbus.so.5
        ├── libmodbus.so.5.1.0
        └── readme
    └── etc
        └── sudoers
```

Speaker notes

- Mise en situation: un client nous envoie un .zip et nous dit "voici mon système Linux"
- Le .zip contient plusieurs README, un semblant de rootfs, des fichiers binaires, des sources, des fichiers de config, une clé SSH, ...
- 10 développeurs vont créer 10 systèmes Linux différents.
- Bien sur comme c'est de l'embarqué, libmodbus va être quelque part la-dedans.

POINTS NÉGATIFS

- Procédure manuelle prône à l'erreur
- Binaires de provenance inconnue
 - Version du code source?
 - Y a-t-il des patchs à appliquer?
 - Une configuration particulière à activer?
 - Avec quel compilateur?
 - Des options de compilation importantes?

AVEC YOCTO

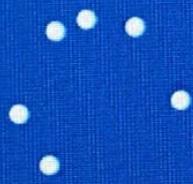
```
SUMMARY = "A Modbus library"
DESCRIPTION = "libmodbus is a C library designed to provide a fast and robust implementation of the Modbus protocol. It runs on Linux, Mac OS X, FreeBSD, QNX and Windows."
HOMEPAGE = "http://www.libmodbus.org/"
SECTION = "libs"
LICENSE = "LGPLv2.1+"
LIC_FILES_CHKSUM = "file://COPYING.LESSER;md5=4fbd65380cdd255951079008b3645"
SRC_URI = "http://libmodbus.org/site_media/build/${BP}.tar.gz"

inherit autotools pkgconfig

SRC_URI[md5sum] = "c80f88b6ca19cab4cefffc195ca07771"
SRC_URI[sha256sum] = "046d63f10f755e2160dc56ef681e5f5ad3862a57c1955fd82e0ce"
```

Speaker notes

- Une recette pour la libmodbus d'il-y-a deux slides.
- Comme libmodbus se compile et s'installe simplement (`./configure && make && make install`), la recette est elle aussi simple (~10 lignes).
- Les versions sont fixes (`${BP}.tar.gz`) et on a un checksum des sources.
- S'il y a une quelconque précaution à prendre pour pallier à un problème de compilation croisé, un bug ou une vulnérabilité, celle-ci est présente sous forme de code exécutable et non comme une procédure manuelle dans un README.



Working on updates

4% complete

Don't turn off your computer

REQUIS POUR L'EMBARQUÉ

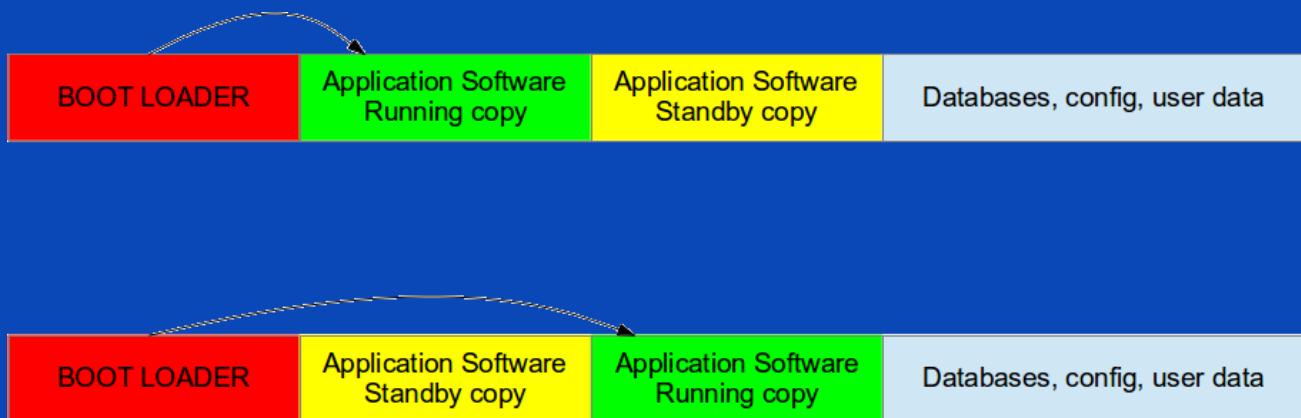
- Configuration unique
- Mise à jour robuste et atomique

La mise à jour par téléchargement de paquets ne répond à aucun de ces critères.

Speaker notes

- Configuration unique: systèmes embarqués accomplissent une tâche unique bien précise. La personnalisation d'une image par l'installation de paquets n'est pas nécessaire, voire nuisible.
- atomique: systèmes embarqués sont généralement plus prompt aux ruptures d'alimentation et de réseau et doivent récupérer des fautes de façon autonome.
- La mise à jour doit aussi être sécuritaire mais ce sont vraiment les 2 points mentionnés sur la slide qui justifie ce qui suit.

MISE À JOUR SYMÉTRIQUE



source: SWUpdate

Speaker notes

- Aussi connu sous le nom de mise à jour A/B
- Configuration unique: l'ensemble du rootfs est mis à jour.
- Atomique: On pivote sur la seconde banque à la toute fin en modifiant une variable du bootloader à la toute fin.
- Ceci est une des alternatives viables, mentionnons également la mise à jour asymétrique (main/recovery) ou encore la mise à jour atomique basé sur des fichiers (libOSTree)

AVEC YOCTO

- SWUpdate (<https://github.com/sbabic/swupdate>)
- Robust Auto-Update Controller (RAUC)
(<https://github.com/rauc/rauc>)
- Mender.io (<https://mender.io/>)

Speaker notes

- Yocto contient des recettes pour plusieurs agents de mise à jour.
- Yocto génère les artefact de mise à jour tel que requis par l'agent de mise à jour.
- Yocto génère aussi des paquets (.ipk ou .rpm) mais ceux-ci sont principalement destinés à l'assemblage de l'image finale.



RÉGLAGES FINS

19

Speaker notes

Quelques exemples de réglages fins que Yocto va faire à partir d'une configuration haut-niveau.

MACHINE TUNE

```
-mtune=name
```

This option specifies **the name of the** target ARM processor **for** which GCC should tune **the performance of the** code. For some ARM implementations better performance can be obtained **by using** this option. Permissible names are: arm7tdmi, arm7tdmi-s, arm710t, ...

Speaker notes

- Possible d'optimiser la compilation pour la machine, ce qui n'est pas possible pour une distribution binaire car on ne connaît pas la machine exacte à priori.
- Impacte la compilation de tous les packages du système.
- Généralement en incluant la layer BSP du fabricant d'un board et en spécifiant MACHINE="acme_machine_123", ceci se fait automatiquement.

PACKAGECONFIG

Exemple: configuration de libSDL2

```
PACKAGECONFIG[alsa]      = "-DSDL_ALSA=ON, -DSDL_ALSA=OFF, alsa-lib, "
PACKAGECONFIG[arm-neon]   = "-DSDL_ARMNEON=ON, -DSDL_ARMNEON=OFF"
PACKAGECONFIG[directfb]   = "-DSDL_DIRECTFB=ON, -DSDL_DIRECTFB=OFF, directfb,
PACKAGECONFIG[gles2]     = "-DSDL_OPENGL=ON, -DSDL_OPENGL=OFF, virtual/egl"
PACKAGECONFIG[jack]      = "-DSDL_JACK=ON, -DSDL_JACK=OFF, jack"
PACKAGECONFIG[kmsdrm]    = "-DSDL_KMSDRM=ON, -DSDL_KMSDRM=OFF, libdrm virtua
PACKAGECONFIG[opengl]    = "-DSDL_OPENGL=ON, -DSDL_OPENGL=OFF, virtual/egl"
PACKAGECONFIG[pulseaudio] = "-DSDL_PULSEAUDIO=ON, -DSDL_PULSEAUDIO=OFF, pulse
PACKAGECONFIG[wayland]   = "-DSDL_WAYLAND=ON, -DSDL_WAYLAND=OFF, wayland-nat
PACKAGECONFIG[x11]       = "-DSDL_X11=ON, -DSDL_X11=OFF, virtual/libx11 libx
```

Speaker notes

- Plusieurs packages peuvent être configuré avant d'être compilé. C'est le temps d'activer ou désactiver certaines fonctions.
- Avec la compilation native, les configurations sont auto-détectées et sélectionnées en fonction de ce qui est présent sur le système.
- Avec Yocto, qui compile pour un système différent de la machine native, les options de configuration doivent être explicites.
- Yocto expose les options de configuration dans PACKAGECONFIG. Certains PACKAGECONFIG vont s'activer automatiquement via MACHINE_FEATURES et DISTRO_FEATURES qui sont globales à la machine et à la distro.

SYSTÈME DE FICHIER RACINE EN LECTURE SEULE

```
IMAGE_FEATURES += "read-only-rootfs"
```

- Garantit que tous les scripts post-installations s'exécutent avec succès et ne sont pas déferés au premier boot.

```
IMAGE_FSTYPES += "squashfs"
```

- Ajoute la génération d'une image **squashfs** du système de fichier racine.

Speaker notes

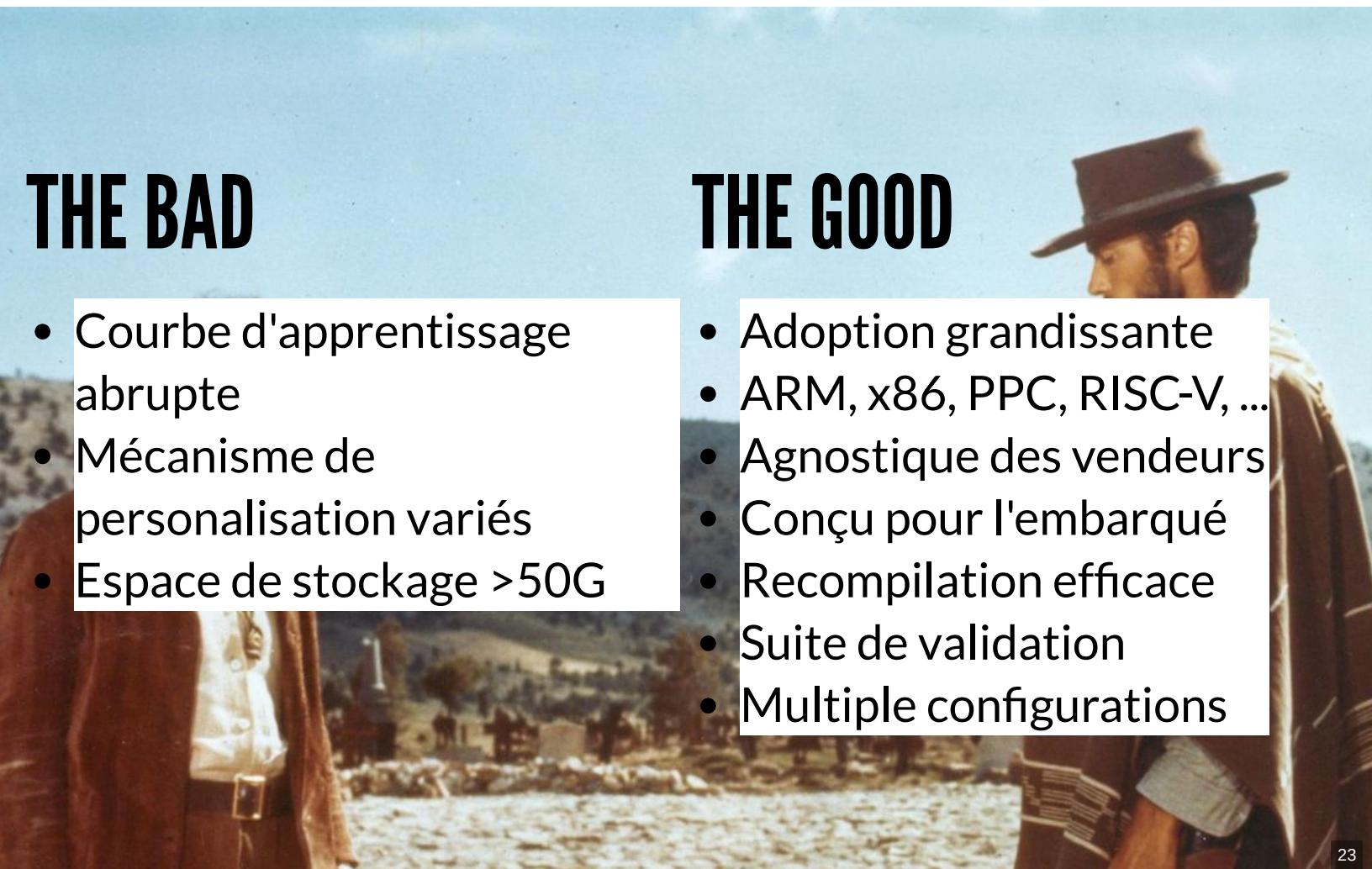
Souvent utilisé en embarqué pour minimiser l'usure de la mémoire flash.

THE BAD

- Courbe d'apprentissage abrupte
- Mécanisme de personalisation variés
- Espace de stockage >50G

THE GOOD

- Adoption grandissante
- ARM, x86, PPC, RISC-V, ...
- Agnostique des vendeurs
- Conçu pour l'embarqué
- Recompilation efficace
- Suite de validation
- Multiple configurations



23

Speaker notes

- Courbe d'apprentissage: facile de faire un premier build mais les premières modifications peuvent prendre beaucoup de temps.
- Mécanisme de personalisation variés: plusieurs façons de faire la même chose se qui complexifie l'apprentissage. Ex.: ajouter une tâche ou append la précédente ou preprend la suivante? Les layers BSP qui ont souvent de la logique custom.
- Recompilation efficace: bitbake trace les paquets dont la recette a changé et ne recompile qu'au besoin.
- Suite de validation: beaucoup de configuration possible, Les mainteneurs de Yocto testent les principales.
- Multiple configuration: plusieurs types d'image (production, dev, installation), plusieurs machines (pénurie de semi-conducteur), plusieurs distros (prod vs dev)

MERCI!



QUESTION?