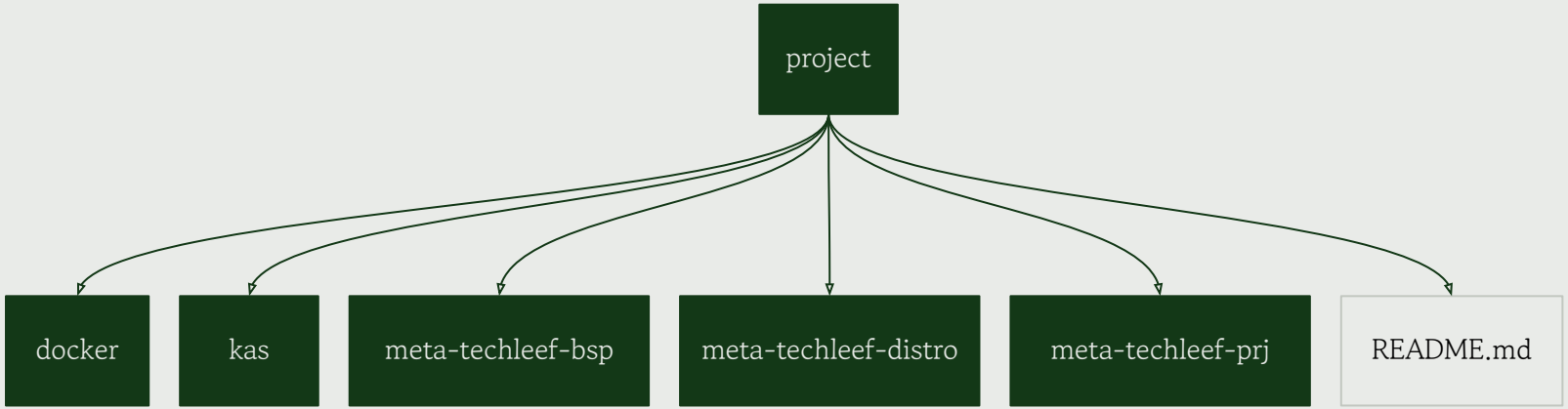
Yocto Project

The ultimate project

- If you have a board (Raspberry, STM32MP, Beaglebone, ..), choose any sensors you want
  - Follow steps for whatever hardware you have
  - For image, add userspace packages to test the hardware
  - Following the right steps with clean structure, the project should be portable to all hardware
- If you don't have a board, use Qemu and assume you have the hardware, but without testing
- For IoT, you can integrate **macchina** framework, develop your custom nodes and integrate them with Yocto


project
docker
kas
meta-techleef-bsp
meta-techleef-distro
meta-techleef-prj
README.md

- Prepare **KAS** yml files for all layers and local configuration file
- Prepare docker setup script to create a container with the project as shared volume
- Setup custom **downloads** and **sstate-cache** directories
- Enable build information output
- Develop a script to setup a docker HTTP server over the packages deploy directory

- Custom machine development (STM32MP, RPI0/3/4/5, BeagleBone, ..)
  - Control every machine feature
  - Remove kernel from going into the rootfs
  - Linux kernel optimization (custom **defconfig** or configuration **fragments**
  - Linux kernel device tree overlay recipe
  - Linux kernel module recipe
  - Boot time optimization
    - Very minimal kernel defconfig
    - Disable as much bootloader configs as possible
    - Use as minimum init manager services as possible
    - Useful link: https://bootlin.com/doc/training/boot-time/boot-time-slides.pdf

- Separate Yocto distribution layer development
- Custom distro configuration development
  - Support for both init managers (**systemd** and **sysvinit**) controlled by a variable
  - Optimizations (no recommended packages, full control on distro features)
  - No **poky**
  - Enable **CVE** checks
  - Disable **GPLv3** packages for all images
  - Enable build information integration for all images
- Definition of build type variables to control development or release builds

- Control image features
- Control every package
- Create development and production images
    - For development:
        - Enable **-dev** and **-dbg** packages
        - Integrate profiling, debugging and development packages
        - Disable ssh password for root user
        - Create a feature that can be easily used to control the development image
- Develop **packagroup** to hold custom packages
- Add data partition (**WIC**, or **FlashLayout** for STM32MP)

- Create a recipe for users and groups management
- Make sure to use it for you custom recipes
- Create **sudoers** configuration files to control users access

- Create toolchain recipe
- Generate the SDK
- Test

- Integrate RAUC Yocto layer
- Develop custom bundle
- Enable bundle encryption
- Enable adaptive update support
- Setup docker HTTP server bound to Yocto deploy directory
- Test RAUC update
- Develop custom D-Bus utility for RAUC
- Test RAUC hawkbit updater

- Kernel **initramfs** integration and init development
- Storage encryption (eMMC, SD) with **dm-crypt**
- Usage of **OPTEE** whenever possible
- Storage integrity with **dm-verity**
- Linux Security Module integration (LSM) **SELinux**
    - Development of custom policies for the applications