Yocto Project™ Basics

AMD
together we advance_

Agenda

- **Introduction to Yocto Project™**
- Yocto Project™ Components
- Setting Up and Building with Yocto Project™

The Yocto Project™

yocto PROJECT

It's NOT an embedded Linux® distribution

It creates a custom Linux for you

www.yoctoproject.org

Open-source project offering templates, tools, and methods

Helps build custom Linux-based systems for embedded products

Supports all hardware architectures

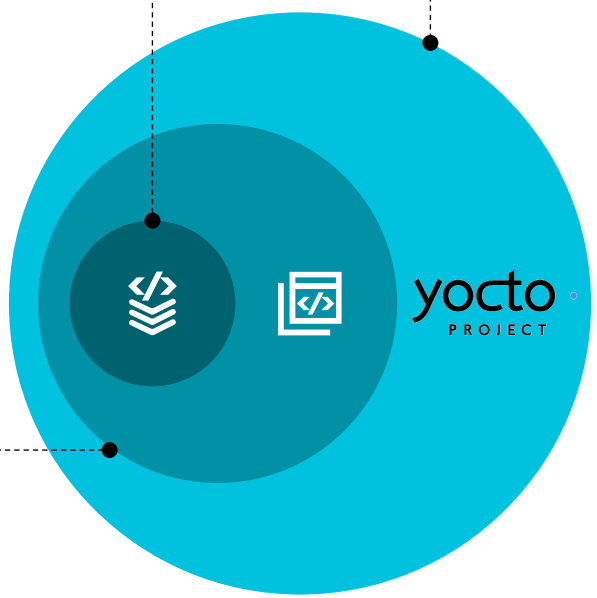
Overview

Open Embedded

- Co-maintained with Open Embedded Project
- Includes Open Embedded-Core and BitBake

POKY

- Reference distribution
- Example embedded Linux® build



Yocto Project™

Extensive testing infrastructure

- Buildbot-based autobuilder for testing

Integrated development tools

- Automated build and testing
- Board support and interchange standards
- Security analysis, license compliance, and SBoM support

Flexible build system

- Uses recipes and layers
- Customizable for specific needs

Key Features of The Yocto Project™

Custom Distributions

Build OS tailored to specific needs

Industry Support

Backed by high-tech industry leaders

Board Architecture Support

Support for x86, Arm®, RISC-V processors, and more

Open-Source Project

Major project under The Linux® Foundation, fully open source

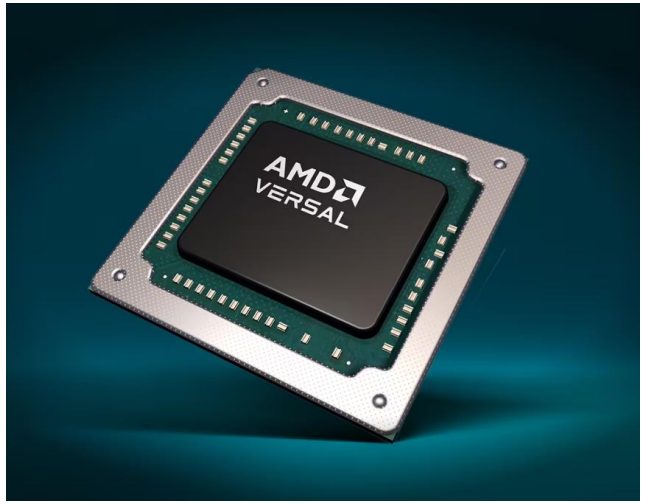
Industry Applications

Used in medical, automotive, industrial sectors and more.

Standard Toolchains and Libraries

User-friendly tools and libraries for easy integration

AMD: Platinum Partner



Platinum partnership

- AMD is a Platinum Partner in the Yocto Project™
- Committed to open-source innovation in Yocto Project and embedded Linux® systems

Key contributions

- Tool development and maintenance for the Yocto Project and custom Linux OS distributions
- Embedded and IoT device development support

Driving innovation

- Ensuring Yocto evolves with the latest technologies
- Collaboration with industry leaders to enhance capabilities

Impact on embedded systems

- Efficient, scalable, and secure solutions
- Benefiting a wide range of industries

Yocto Project™ Offers . . .

Offers

Framework including:

- Templates, tools, and methodologies
- Portable Linux® distributions

Build

Leverages scripting languages for build customization:

- Shell and Python™ scripts

Enables update testing without a costly build/deploy cycle

Rootfs Generation

- Yocto Project™ is highly customizable
- Further extensible by software packages available from GitHub

Customization

Customizable recipes offer flexibility and long-term maintainability

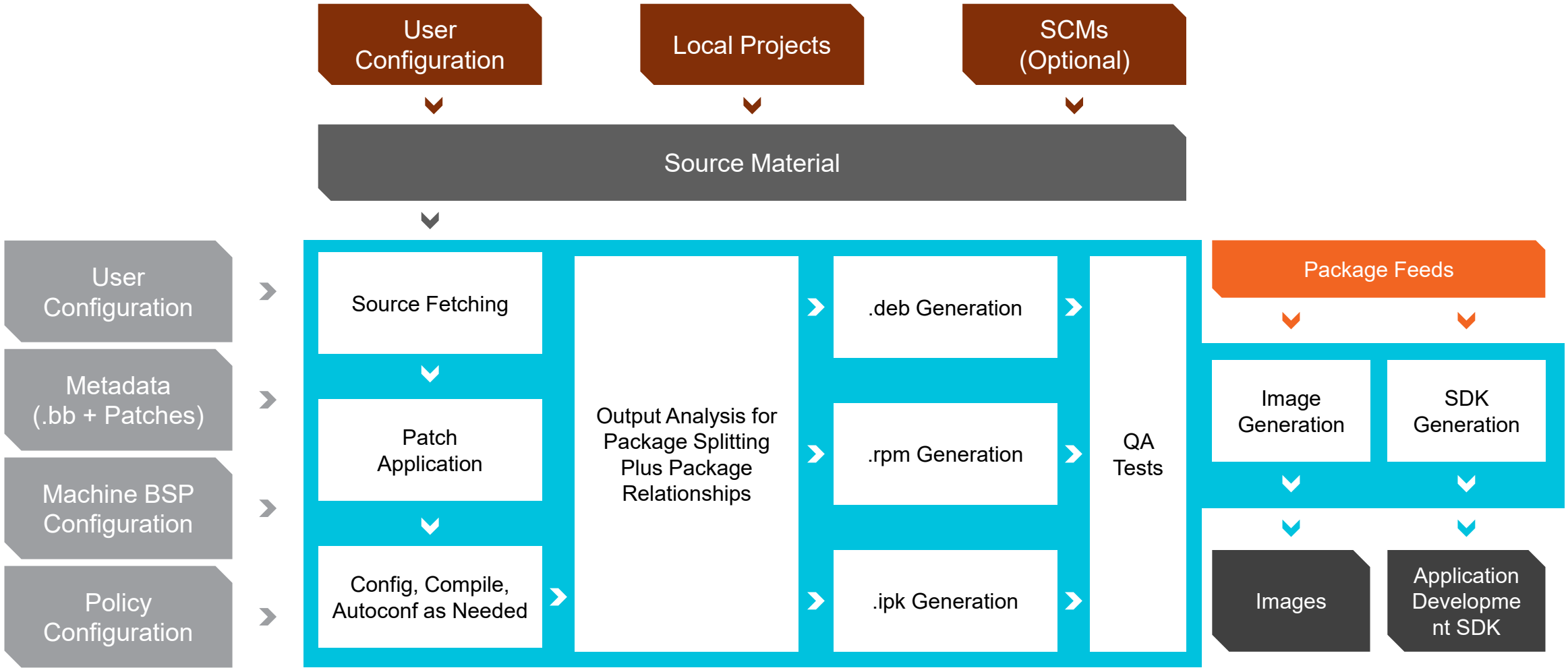
Path to Deployment

- Clear and customizable path to production
- Well-suited for scalable, production-grade environments
- Smoother transition from development to production

Community Support

- Broad community and industry support
- Easy to integrate external software and components
- Active community of developers maintaining 2000+ Linux packages

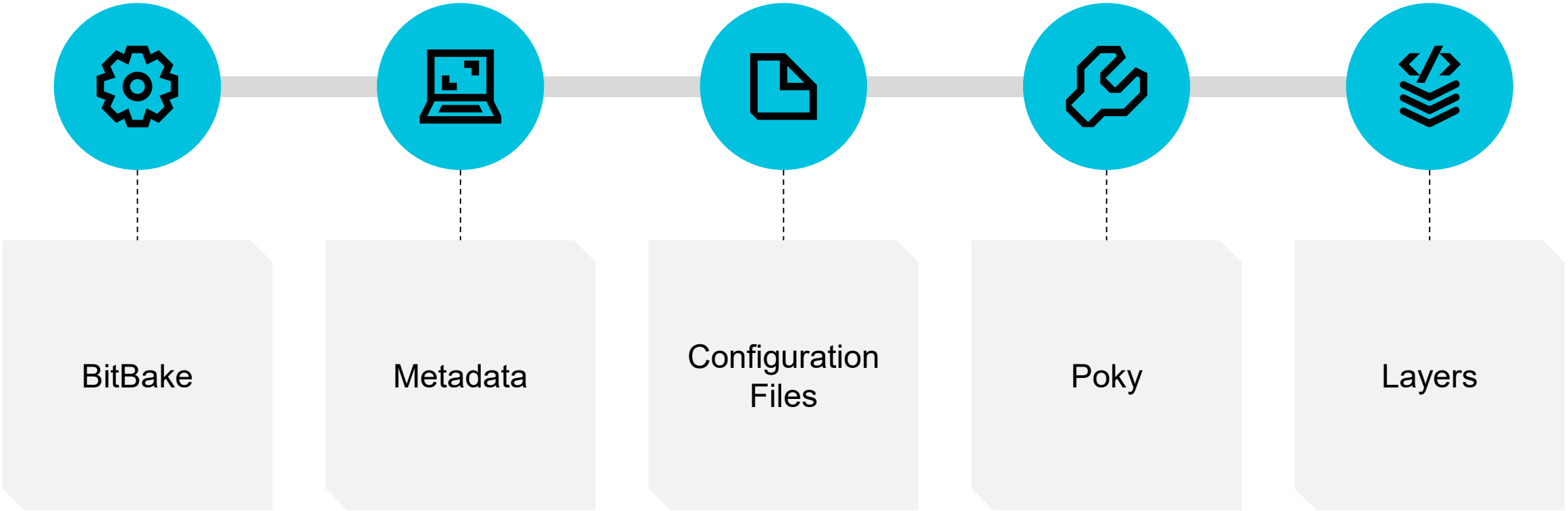
Yocto Project™ Workflow



Agenda

- Introduction to Yocto Project™
- **Yocto Project™ Components**
- Setting Up and Building with Yocto Project™

Yocto Project™ Components



BitBake

Yocto Project™ Build Tool

Automates the process of creating a working Linux® operating system

Core task scheduler and build tool that offers greater flexibility than traditional tools

Key features:

- Recipe parsing
- Dependency management
- Task execution
- Configuration management
- Cross-compilation

Core components:

- Recipes (.bb files)
- Configuration files
- Classes (.bbclass files)
- Tasks

Metadata

Set of files that define how software is built, configured, and packaged

Structured in layers; a layer defines how software is built for an embedded Linux® system

Types:

- Configuration Files (local.conf/bblayers.conf)
- Recipes (.bb Files)
- Class Files (.bbclass)
- Append Files (.bbappend)
- Layer Configuration (conf/layer.conf)

Configuration Files in Yocto Project™

conf/ directory controls **Poky** and **BitBake**

local.conf

Initialization of Build Environment

- Created in **build/conf/** when initializing a build
- Controls key aspects of the build process

Configuring the Build Process

- Variables influence BitBake's build process
- Overrides settings from other config files

Global Variables

- Variables in configuration files are global to every recipe (configuration metadata)
- Configuration file parsing order:
  1. build/conf/local.conf
  2. conf/machines/.conf
  3. conf/distro/.conf

Tip: Comments inside local.conf provide useful documentation and default variable values

bblayers.conf

A layer defines how software is built for an embedded Linux® system

Purpose

- Layer definitions to be used in the Yocto Project™
- Essential for configuring which layers BitBake includes in the build process

Key components

- **BBLAYERS** variable: Lists the directories of the layers to be included in the build
- **?=** operator: Assigns a value to BBLAYERS only if it has not been set already
- **Backslash** (\): Allows the string to continue onto the next line for better readability
- **${TOPDIR}** variable: Points to the top directory of the build environment

Example configuration

```
BBLAYERS ?= " \
${TOPDIR}/../meta \
 ${TOPDIR}/../meta-poky \
${TOPDIR}/../meta-yocto-bsp \
${TOPDIR}/../meta-custom \ "
```

Explanation of paths

- Paths listed (e.g., ../meta, ../meta-poky, ../meta-yocto-bsp, ../meta-custom) are relative to ${TOPDIR}
- Specify the locations of different layers

Other Configuration Files

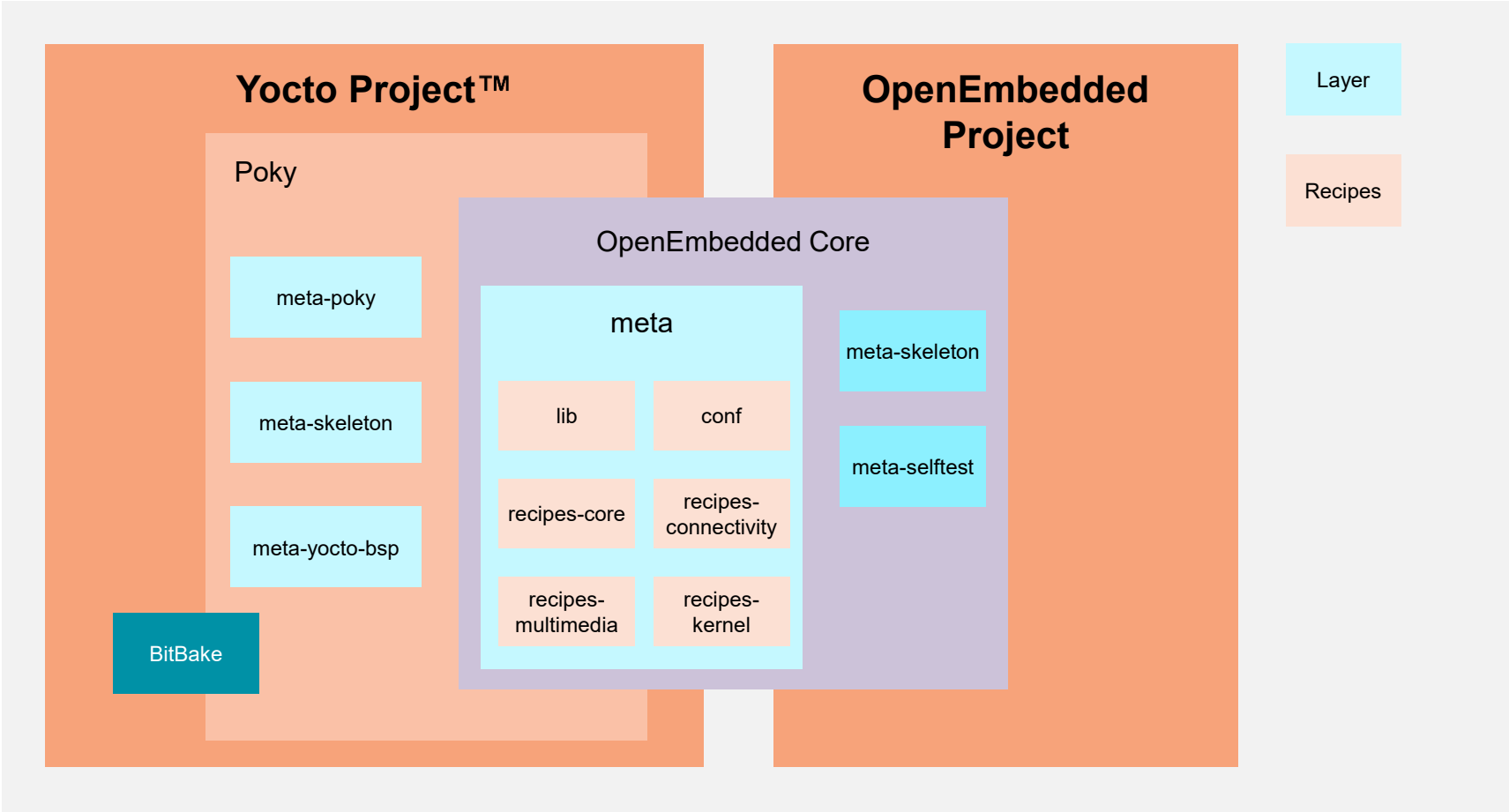
File Name	Location	Purpose
site.conf	build/conf/ (optional)	Site-wide configuration (cache locations, optimizations)
layer.conf	meta-<layer>/conf/	Defines rules for a particular layer
distro.conf	meta-<distro>/conf/distro/	Defines distribution policies
machine.conf	meta-<layer>/conf/machine/	Defines machine-specific settings (CPU, bootloader, kernel)
bitbake.conf	meta/conf/	Core BitBake configuration (do not edit directly)

Poky

Poky – Yocto Project™ reference distribution combining BitBake + OpenEmbedded Core (OE-Core) + metadata to build Linux® systems

Key Components of Poky

- BitBake
- OpenEmbedded Core
- Poky-specific metadata
- Board support packages (BSPs)

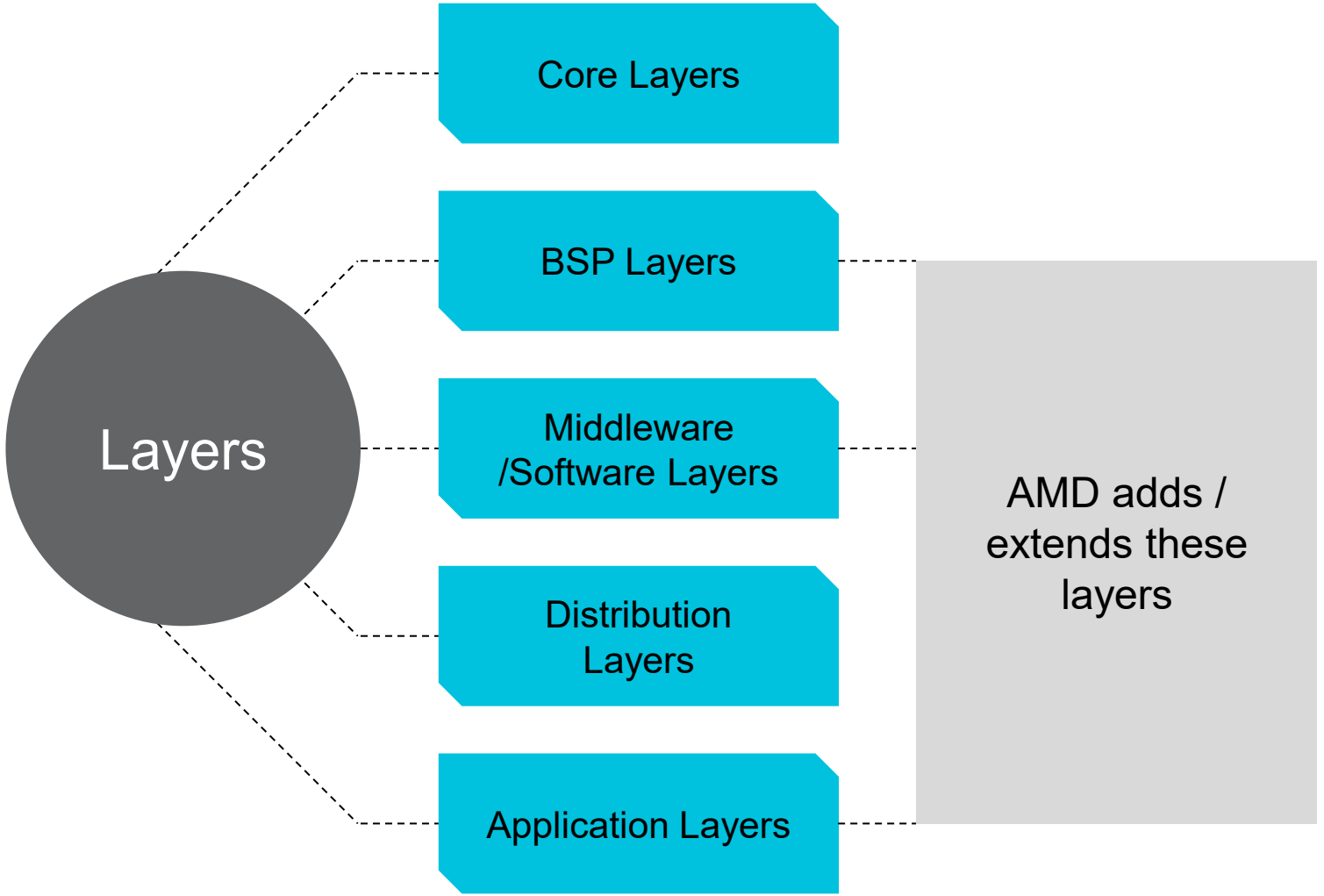


Yocto Project™ Layers

A structured approach to organizing metadata through the layer model

Metadata organization, simplifying build customization and extension

Grouping of recipes, configuration files, and patches for building software on embedded Linux® systems



Yocto Project™ Recipes

Define:
Software build and packaging process

Purpose:
Provide instructions for fetching, configuring, compiling, and installing

File extension:
.bb (BitBake recipe)

Basic Components

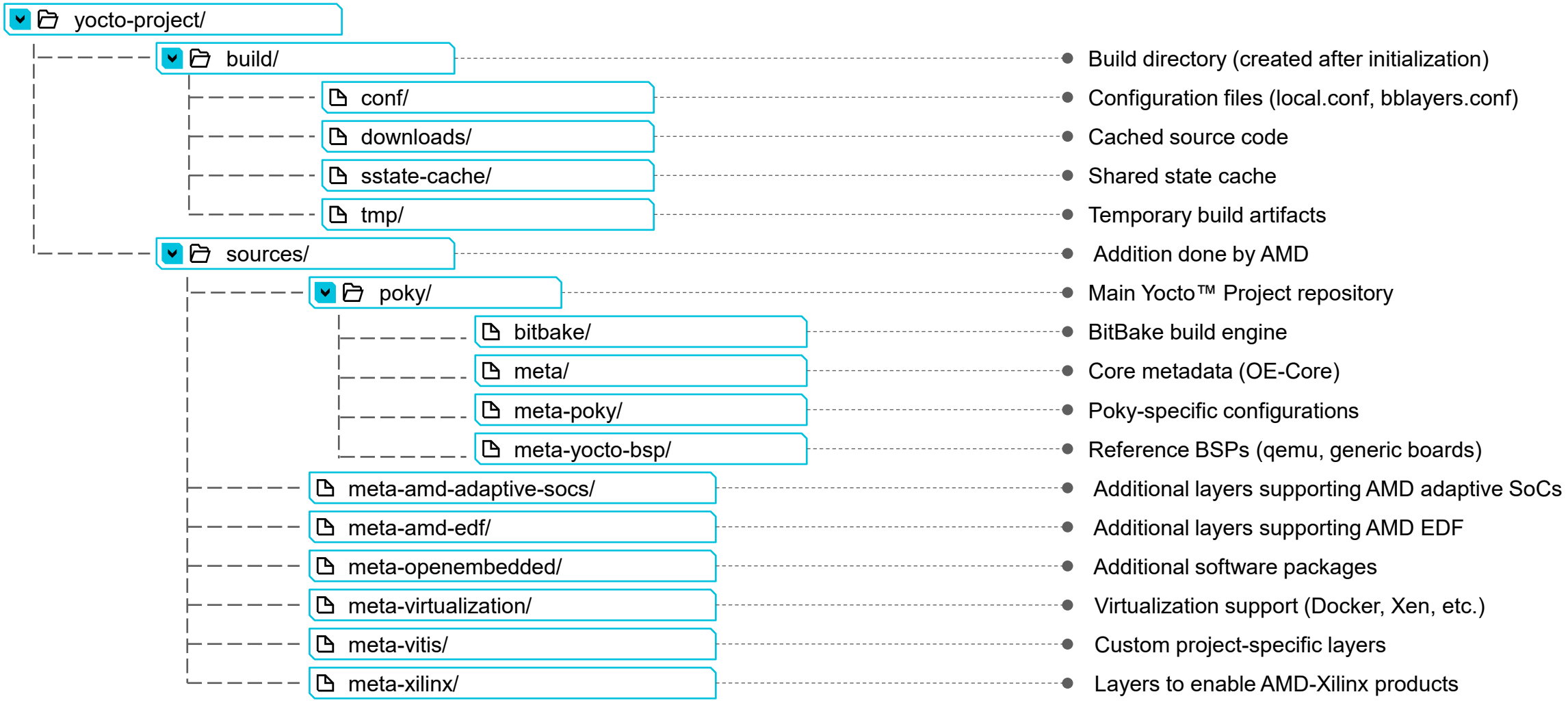
- **Header** – Metadata (description, license)
- **Source** – Source files location (SRC_URI)
- **Build instructions** – Configuring, compiling, installing

Key Variables

- **SRC_URI** – Source location
- **S** – Unpack directory
- **DEPENDS** – Build-time dependencies
- **RDEPENDS** – Runtime dependencies

inherit keyword – Allows reusing of common functionality from classes

Yocto Project™ Directory Structure with AMD Added Extensions



Agenda

- Introduction to Yocto Project™
- Yocto Project™ Components
- **Setting Up and Building with Yocto Project™**

Yocto Project™ Build Flow: Setting Up AMD EDF Environment

Repo Tool:

- Tool chosen by AMD to be the tool for EDF flows to manage multiple git repositories (not a requirement from Yocto Project™)
- Fetches multiple Git repositories using a manifest, Downloads all source in one command
- EDF yocto-manifests define required layers for each release

Yocto setup for EDF:

- Check for OS support for your Yocto version – Not all OS's are supported
- Configure git
  - `$ git config --global user.email "you@example.com"`
  - `$ git config --global user.name "Your Name"`
- Download and install Repo tool
  - `$ curl https://storage.googleapis.com/git-repo-downloads/repo > repo`
- Use Repo to setup all the repositories for EDF
  - `$ repo init -u https://github.com/Xilinx/yocto-manifests.git -b refs/tags/<amd-edf-version> -m default-edf.xml`
  - `$ repo sync`
  - `$ repo start <branch_name> --all  #optional`

Initialize the build environment:

- `$ source edf-init-build-env`

Yocto Project™ Build Flow: Building an AMD EDF Linux Image – SD Card Boot

For the SD card boot –

- Specify the machine configuration and build the targets
- Build the boot image (boot.bin) using supported recipes xilinx-bootbin,
  - `$ MACHINE =<board specific machine name> bitbake xilinx-bootbin`
  - `$ MACHINE=zynqmp-zcu104-sdt-full bitbake xilinx-bootbin` – for Zynq ZCU104
- Build the common EDF Linux disk image using edf-linux-disk-image recipe
  - `$ MACHINE=<Common disk image machine name> bitbake edf-linux-disk-image`
  - `$ MACHINE=amd-cortexa53-mali-common bitbake edf-linux-disk-image` – for Zynq ZCU104

Verify the generated images and deploy

- `$ ls -la tmp/deploy/images/zynqmp-zcu104-sdt-full/`
- `$ ls -la tmp/deploy/images/amd-cortexa53-mali-common/`

Copy the generated boot.bin into the WIC image

- `$ wic cp boot.bin <wic-file-name>.rootfs.wic:1`

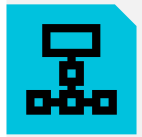
Flash the SD card with the WIC image and boot the board

Yocto Project™ Build Flow: Building an AMD EDF Linux Image – Multistage Boot

- **For the multi-stage boot (depending on the board selected) – First build the OSPI image followed by the common EDF Linux® disk image**
  - Specify the machine configuration and build the targets
  - Build the EDF boot firmware (OSPI image)
    - `$ MACHINE=<board specific machine name> bitbake edf-ospi`
    - `$ MACHINE=versal-2ve-2vm-vek385-sdt-seg bitbake edf-ospi` – for Versal VEK385
  - Build the common EDF Linux disk image (wic) containing general purpose Linux
    - `$ MACHINE=<Common disk image machine name> bitbake edf-linux-disk-image`
    - `$ MACHINE=amd-cortexa78-mali-common bitbake edf-linux-disk-image` – for Versal VEK385
- **Verify the generated images and deploy**
  - `$ ls -la tmp/deploy/images/versal-2ve-2vm-vek385-sdt-seg/`
  - `$ ls -la tmp/deploy/images/amd-cortexa78-mali-common/`
- **Copy the generated boot.bin into the WIC image**
  - `$ wic cp boot.bin <wic-file-name>.rootfs.wic:1`
- **Flash the OSPI image and boot the board using the EDF Linux image**

Documentation

Yocto Project™ Overview and Concepts Manual



OS Integration and Development

- For AMD Zynq UltraScale+ MPSoC
- For AMD Versal Adaptive SoC

Yocto Project™ – AMD Wiki

Boot Architecture and Pre-built Yocto Machine Specs

Summary

- Yocto Project™ offers an open-source and powerful framework for embedded Linux® development, widely adopted despite its learning curve
- A community-driven project that provides a flexible, customizable build system to create custom Linux distributions from scratch
- Key components:
  - Bitbake – the build engine
  - Metadata – recipes and layers defining what gets built
  - Configuration files – tailoring builds for the target devices
  - Poky – the reference distribution
  - Layers – modular building blocks for reusability and scalability
- Well-suited for scalable, production-grade environments, ensuring a smooth transition from development to production with confidence in end products performance and stability

DISCLAIMER AND ATTRIBUTIONS


AMD