Raspbian vs. Build-Systeme

Das richtige Werkzeug für solide Produkte

Bastian Krause – b.krause@pengutronix.de

Über mich

- Ich unterstütze bei Pengutronix seit 2015 viele verschiedene Kunden mit Embedded-Systemen
  - Integration von Bootloader / Kernel / Userspace
  - Verified-Boot-Prozesse
  - Automatisierte Laufzeit-Tests von Embedded-Systemen

„Wir haben einen Raspberry-Pi-Prototypen mit Raspbian. Eigentlich funktioniert auch schon fast alles, aber…“

Fahrplan

1. Naheliegend: Golden Image
2. Anforderungen an ein Build-System
3. Build-Systemen-Kategorien
4. Ausblick

Produktbeispiel: IoT-Kaffee-Automat

- Custom Board
- Display
- Internet-Konnektivität
- Kontaktlos zahlen per NFC
- Große Stückzahl, europaweit

Golden Image

- Raspbian/Debian passt nicht ohne Anpassungen zum eigenem Board

  ⇒ also System manuell anpassen

  - **Device-Tree** hinzufügen
  - **Patches** schreiben/anwenden
  - **Build-Konfiguration** ändern
  - **Laufzeit-Konfiguration** ändern

Golden Image: Reproduzierbarkeit/Updates

- Ist der Softwarestand auch in 5 Jahren baubar/anpassbar?
  - Bugs reproduzieren/debuggen
  - Update
    - Kritischer Bug
    - Neues Feature

Golden Image: verschiedene Use-Cases

- Produkt-Anwendungsfall fest definiert (im Gegensatz zu Desktop/Server/Rpi)
- große Stückzahlen → keine Interaktion/Administration

⇒ **endet in Katastrophe**

Anforderungen

- Fest definierte Versionen + Build-Rezepte aller Komponenten (Nachvollziehbarkeit, Kompatibilität)
- Organisation von Patch-Stacks / Konfigurationen
- anpassbare Build-Rezepte
- Build-System-Komponenten unter Versionskontrolle
- Software-Stand image-basiert
- Lizenzreport

Anforderungen

- Product Owner in Lage versetzen, gesamte Software jederzeit bauen zu können
  - Historie/Anleitung geht nicht durch Mitarbeiter-/Dienstleisterwechsel verloren
- Distribution „as is“ höchstens für Prototypen geeignet
  - sonst teuer und Heise-Schlagzeile

⇒ **Kontrolle über Produkt behalten**

Build-Systeme: Kategorien

Distributions-basiert

- „Golden Image“ weitergedacht
- Image-Generierung aus Debian-Paketen

z.B. ELBE, Apertis, CIP

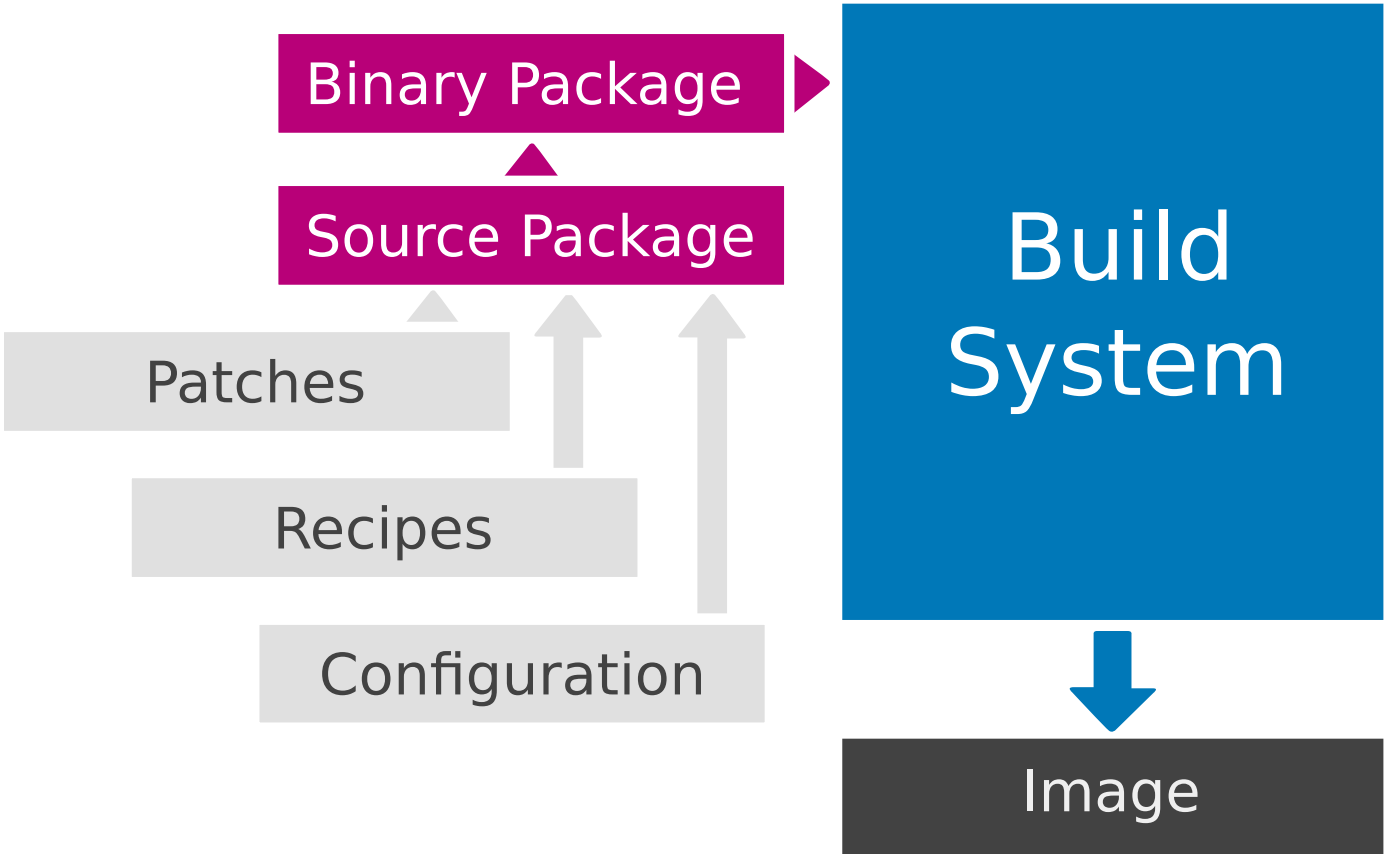
Source-basiert

- Pakete werden aus Source gebaut

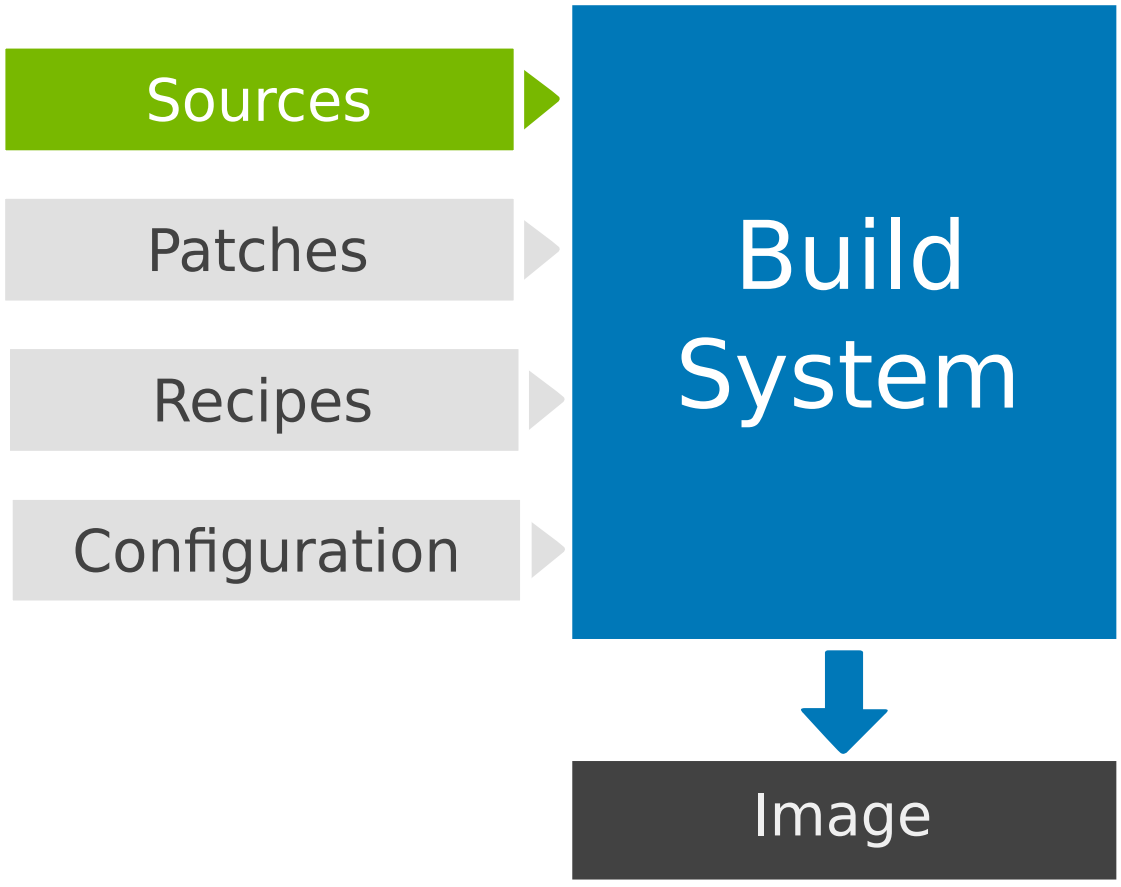
z.B. Yocto, Buildroot, ptxdist

Nicht behandelt: Image + App-Container
(z.B. Ubuntu Core, balenaOS)

Build-System: Distributions-basiert



Build-System: Source-basiert



Build-System: Unterscheidungsmerkmale

	Distributions-basiert	Source-basiert
Einfacher Einstieg	**O**	**–**
Paketauswahl	**+**	**O**
Anpassbarkeit im Detail	**O**	**+**
Konfiguration für Embedded	**–**	**+**
Image-Größe optimierbar	**–**	**+**
Test-Coverage	**+**	**O**

Build-Systeme: welches ist das richtige?

- Distributions-basierte Lösung
  - näher am Debian-Umgebung (apt für Development, Pakete, ..)
  - gut für Produkte, die näher an x86 sind
- Source-basierte Lösung
  - kleinteilige Anpassungen
  - komplexe Produkte/Anwendungsfälle
    - Ressourcenverbrauch → Hardware-Kosten

Build-Systeme: Entscheidungen...

- Klare, effiziente Workflows in der Entwicklung sind entscheidend
- Einarbeitungszeit ins passende Build-System ist gut investiert
- früh überlegen, welchen Ansatz man wählt, um sich unnötigen Aufwand zu ersparen

Weitere Vorteile von Build-Systemen

- Kollaboration innerhalb des Entwicklungsteams und mit externen Partnern
  - Unterschiedliche Verantwortungsbereiche
- Nötige Infrastruktur zum Entwickeln
- License-Compliance
- Regulatorische Anforderungen

Weitere Vorteile von Build-Systemen

- Product Lifecycle → Maintenance
- Release/Update
  - Release-Management
  - Versions-Updates (rebase)
  - Updates im Feld

Fazit

- Golden Image nicht für Produkte!
- Grundproblematik lösen alle Build-System-Typen
- Build-Systeme bieten Lösungen für weitere Herausforderungen im Produkt-Lebenszyklus

Behalten Sie die Kontrolle über Ihr Produkt!

b.krause@pengutronix.de

Linksammlung

- „Why Raspberry Pi Isn't a Good Choice for Commercial Products"

  https://www.allaboutcircuits.com/technical-articles/10-reasons-raspberry-pi-isnt-a-good-choice-for-commercial-products/

- „Debian or Yocto Project? Which is the Best for your Embedded Linux Project?"

  https://youtu.be/iDllXa8SzUg

- „Building Embedded Debian and Ubuntu Systems with ELBE"

  https://youtu.be/NO36dmzqlW8

- „Creating Debian-Based Embedded Systems in the Cloud Using Debos"

  https://youtu.be/_NZrSR3prwk