Building biber für Alpine und arm64

Wie ich eher zufällig zum Contributor eines Open Source Projekts wurde.

Alexander Krumeich, 07.09.2024

n-doc unter Alpine möglich?

Kann ich ein paar Megabyte von 700MB sparen?

- Alpine ist das de facto Basis-Image für Docker-Container:
  Alpine = 13 MB, Ubuntu = 30 MB, Debian Slim = 137 MB
  (Stand 08/2024, damals [2019]: Alpine ~5 MB)
- Alpine Linux basiert auf MUSL statt auf glibc.
- Leider publiziert @plk keine MUSL-Variante (linux-musl_x86-64)
- @plk: Wenn Du das haben möchtest, kannst Du es ja selber bauen.
  (https://github.com/plk/biber/issues/255)
- Das eine oder andere „Like“ in dem Thread zeigt, dass mindestens drei andere Leute die Idee gut finden. Grund genug, Zeit zu investieren!

biber

Architekturüberblick

- Biber ist ein Perl-Programm mit Abhängigkeiten zu ~50 Perl Modulen.
- Schwierig zu verteilen:
  - Perl-Installation als Voraussetzung?
  - Alle Perl-Bibliotheken über CTAN verteilen?
- Lösung: Biber bringt alle Abhängigkeiten selbst mit.
- Selbstextrahierendes Archiv mit PAR::Packer
- @plk baut viele Plattformen selbst, hat ein paar Helfer für den Rest.

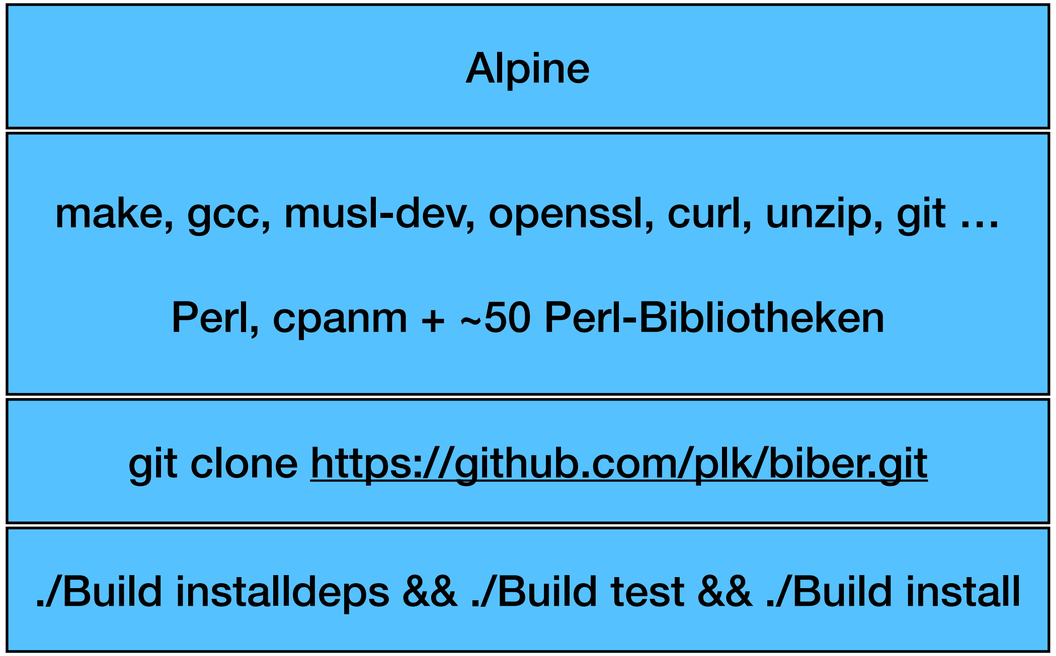
DIY: Aber wie, ohne Alpine?

- Full Disclosure: Ich habe weder von Alpine, MUSL noch Perl Ahnung.
- Alpine wird zu 99% in Docker Containern verwendet.
- Eine richtige Alpine-Installation hatte ich nicht.
- Also: Warum nicht einen Docker-Container für den Build verwenden?

Wie baut man einen Builder-Container?

Spoiler: so nicht.

- Startpunkt: Basis-Image
- Füge alle benötigten Abhängigkeiten hinzu
- Clone das Quell Repository hinzu
- Starte den Build

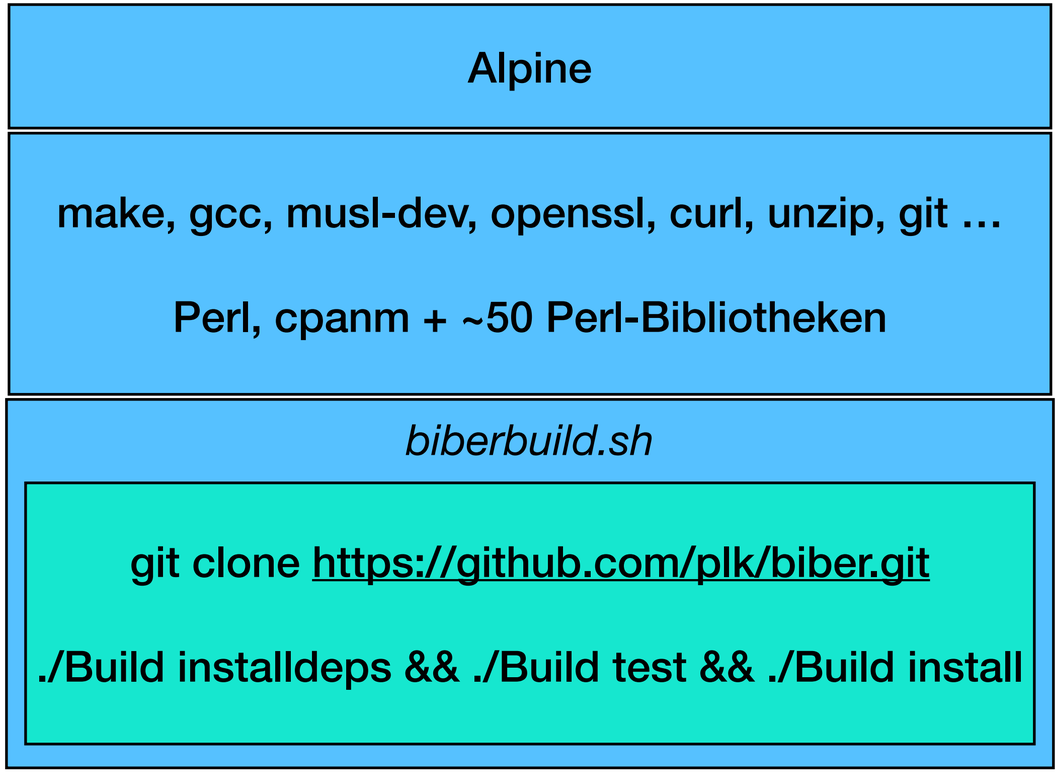


Problem: Für jeden neuen Repo-Stand von biber muss die gesamte Umgebung neu installiert werden.

Wie baut man einen Builder-Container?

Zweiter Versuch

- Startpunkt: Basis-Image
- Füge alle benötigten Abhängigkeiten hinzu
- Der „eigentliche“ Build-Prozess wird in einem Shell-Skript formuliert.
- Der Container startet das Skript als Entrypoint.



Zusammenstellen des Images und der Build von biber sind entkoppelt.

Die Umgebung kann wiederverwendet werden, wenn biber sich ändert.

krumeich/biber-alpine

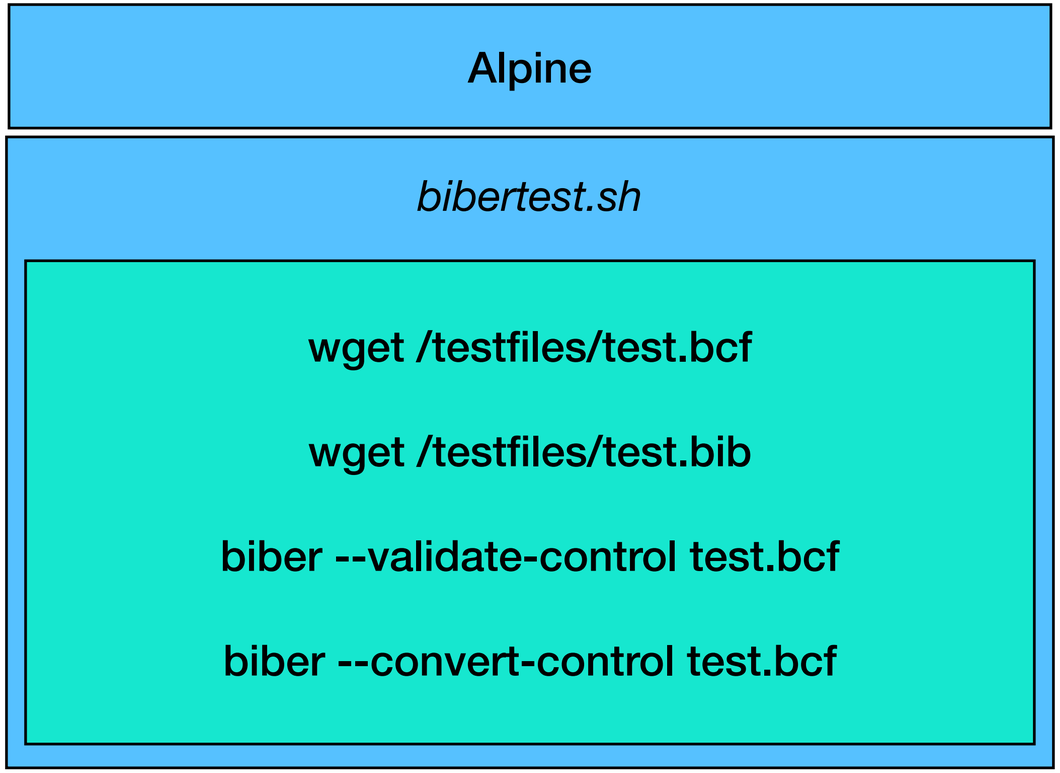
Docker-basierte Build-Chain

- Steuern des Builds/Containers über ein Makefile
- Mit Umgebungsvariablen (Parametern im Container) kann man verschiedene Versionen von biber bauen.
  - BRANCH -> spezifische Version oder „dev“
  - REPO -> eigener Fork oder „plk/biber“
- Das Working-Directory des Hosts wird in den Container gemountet. So kann das fertige Artefakt aus dem Container exportiert werden.

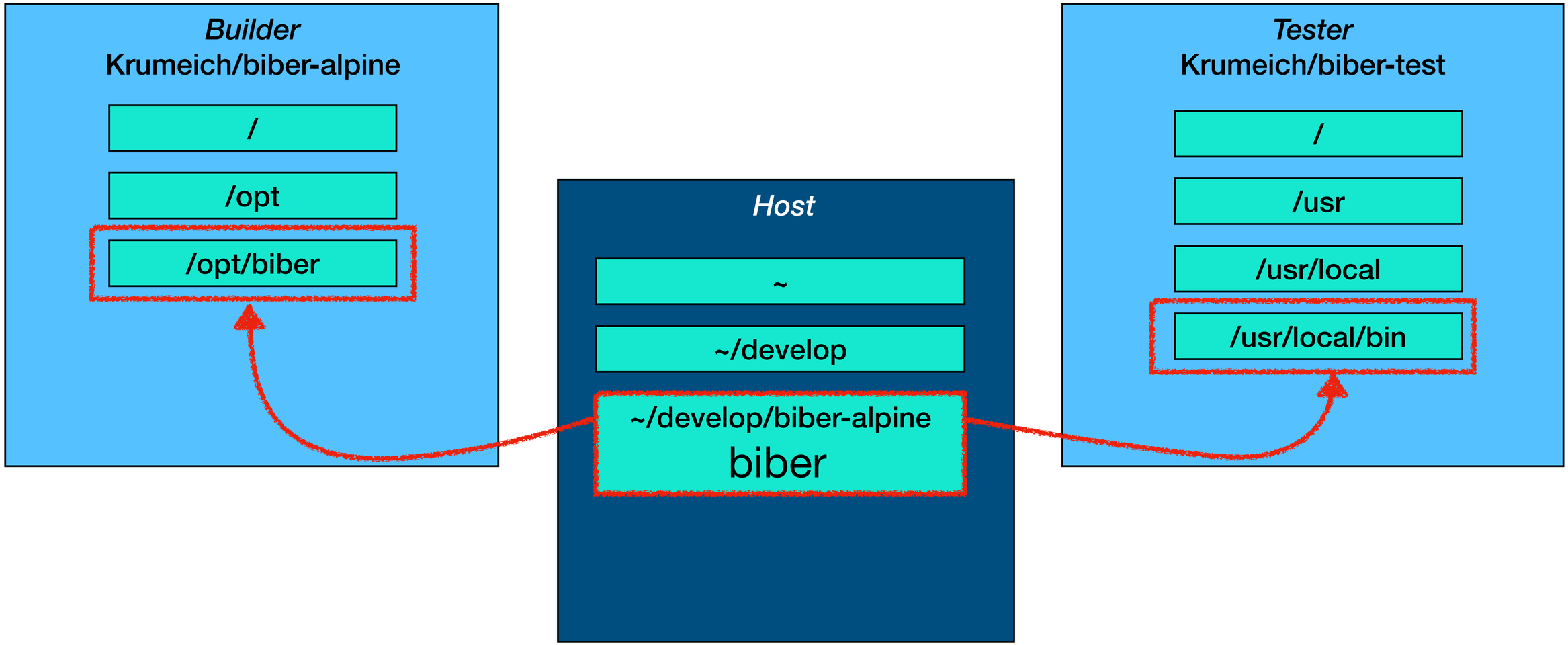
Integrationstests

Macht mein biber, was es soll?

- Integrationstest mit Testdateien von @plk
- Wie ausschließen, dass es Interferenzen mit der Perl-Entwicklungsumgebung gibt?
- Eigener Alpine-Container für Tests ohne zusätzlich installierte Software
- Working-Directory mit biber-binary wird in den Container gemountet.



Verhältnis von Builder und Tester



Deployment

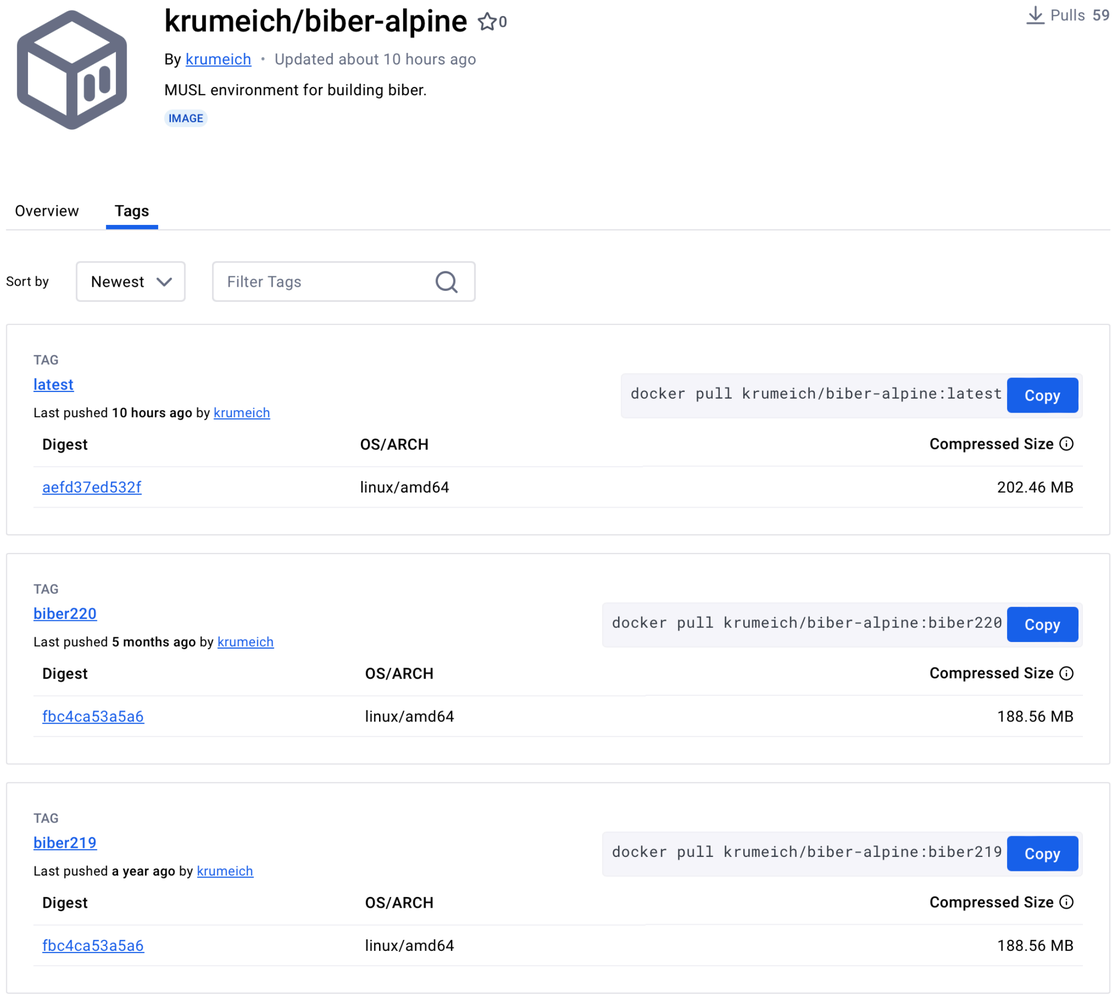
SourceForge und CTAN

- Ursprünglich haben alle Contributoren ihre Implementierungen bei SourceForge veröffentlicht.
- Seit 2022 reichen wir die Archive selbst bei CTAN ein.
- Falls jemand von CTAN hier ist:
  Nochmal Entschuldigung für die vielen Anläufe und Fehlversuche!

Versionierung

Zusammenhang zwischen biber und biber-alpine

- @plk veröffentlicht ein- bis zweimal pro Jahr eine neue Version von biber.
- Für jede Version gibt es ein getaggtes Docker Image. Damit ist die Build-Umgebung archiviert und auch alte Versionen können gebaut werden.
- Im biber-alpine Git-Repo gibt es analog dazu ein Tag



PAR::Packer

Nun zu den Schwierigkeiten

- Das „statisch gelinkte“ biber Binary wird mit PAR::Packer erstellt.
- PAR::Packer stellt alle Komponenten zusammen und erstellt das gepackte Binary.
- Jede Plattform/Architektur hat ihr eigenes Skript, weil die Ausgangsdateien plattformabhängig sind.
- Das Zusammenstellen mit PAR::Packer ist zerbrechlichste Teil (vgl. build.sh).
- Bibliotheken aus biber dürfen keine Abhängigkeiten zu anderen Bibliotheken auf dem Zielrechner haben. Solche Abhängigkeiten müssen vorher entfernt werden.

Und aarch64?

Auftritt Simon (@sbrass)

- Nach ein paar erfolgreichen Releases für linux-musl_x86_64 kam in einem weiteren Issue die Frage nach Support für linux-aarch64.
- „Ich habe da mal etwas vorbereitet.“
- Simon forkt mein Repo und macht alle notwendigen Anpassungen.
- Seit einem Jahr steuern Simon und ich den Build für linux-aarch64 bei.

Pläne und Ausblick

- Ausführlicher Integrationstest mit TeX Live Installation.
- Vereinheitlichung der Projekte durch Multi-Architektur Image: Allerdings hat niemand Interesse an linux-musl_aarch64.
- Falls mal jemand xindy für linux-musl bauen sollte, könnte ich mein n-doc Image auch mal auf Alpine umstellen…

Quellen

- biber: https://github.com/plk/biber/
- biber-alpine: https://github.com/krumeich/biber-alpine
- biber-aarch64: https://github.com/sbrass/biber-linux-aarch64