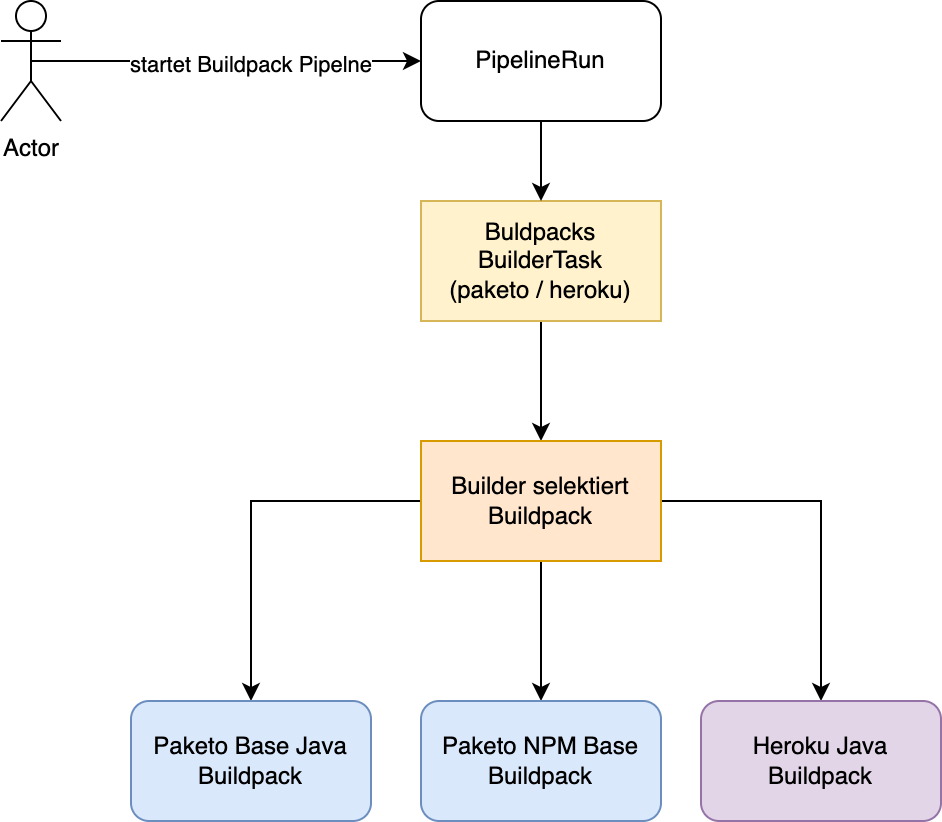
# Buildpacks@Tekton

* [Buildpack vs Builder](#Buildpacks@Tekton-BuildpackvsBuilder)
* [Docker vs Buildpack](#Buildpacks@Tekton-DockervsBuildpack)
* [Buildpacks Builders / Runtime Matrix](#Buildpacks@Tekton-BuildpacksBuilders/Ru)
  + [Heroku Buildpacks](#Buildpacks@Tekton-HerokuBuildpacks)
  + [Paketo Buildpacks](#Buildpacks@Tekton-PaketoBuildpacks)
  + [Google Buildpacks](#Buildpacks@Tekton-GoogleBuildpacks)
  + [Welches Buildpack sollen wir verwenden](#Buildpacks@Tekton-WelchesBuildpacksolle)
  + [Matrix](#Buildpacks@Tekton-Matrix)
* [Buildpack Task bei ESTA Tekton](#Buildpacks@Tekton-BuildpackTaskbeiESTAT)
* [Newrelic Agent Buildpack](#Buildpacks@Tekton-NewrelicAgentBuildpac)
* [Ein paar Notizen und Links](#Buildpacks@Tekton-EinpaarNotizenundLink)

Grundsätzlich funktionieren die Buildpacks von Paketo und Heroku für Maven bereits mit Esta Tekton. Die Google Buildpacks sind für uns weniger interessant, da diese keinen so breiten Plattformsupport und Community Support / Marketplace haben.   
Aus diesem Grunde gehen wir nicht weiter auf die Google Buildpacks ein.

## Buildpack vs Builder

Was ist der Unterschied von Buildpacks und Buildern. Der Builder ist eigentlich derjenige der herausfindet welche Technologie man builden möchte und verwendet dann das entsprechende Buildpack für die entsprechende Technologie. So gesehen sprechen wir hier von Heroku, Paketo oder Google Buildern, welche dann die entsprechenden Buildpacks und Runtimeimages hinten dran verwenden um ein Dockerimage zu builden.



## Docker vs Buildpack

Mit unseren klassischen Docker Builds haben wir natürlich mehr Kontrolle über den Buildprozess und können ihn nach Belieben anpassen.  Der Vorteil von Buildpacks ist, dass dort sehr viel spezialisiertes Docker- und Buildknowhow vorhanden ist, welches wir nicht haben.   
Auch geschieht die Pflege des Builds und der Runtime durch die Community. So kann man ohne Dockerknowhow ein Buildimage erstellen. Weiter wird mit Buildpacks der ganze Dockerbuildprozess massiv optimiert, was zu sehr schnellen Builds führt. Eine gute Beschreibung findet man hier:

<https://tanzu.vmware.com/developer/blog/understanding-the-differences-between-dockerfile-and-cloud-native-buildpacks/>

An gewissen Stellen wird es weiterhin Sinn machen auf klassische Dockerbuilds zu setzen.

## Buildpacks Builders / Runtime Matrix

Ich gehe nur auf die offiziell supporteten Buildpacks ein, da z.B. Heroku 9 offiziell supportete Buildpacks und mehr als 8000 community Buildpacks hat.   
Auch Paketo hat da einige aber nicht so viele Buildpacks. Heroku und Paketo sind für uns die interessantesten Builder welche gängige Buildpacks für alle benötigten Technologien anbieten.

### Heroku Buildpacks

Heroku ist der Ursprung des Buildpackkonzepts und bietet 9 offiziell supportete und über 8000 community Buildpacks für alle möglichen Zwecke an. Heroku Buildpacks werden mit der MIT License ausgeliefert.

Unsere Tekton Pipeline kann mit Heroku Java Buildpacks builden. Es benötigt im Java Fall lokal noch folgende 2 Files:

**system.properties** in dem die Java Version angegeben wird:

java.runtime.version=11  
  
Für maven wird das maven **settings.xml** im Repo Root benötigt.

### Paketo Buildpacks

Paketo bietet Community driven Builders mit Buildpacks an welche unter der Apache License 2 laufen. Sie bieten 3 verschiedene Runtime Images Base, Full und Tiny welche Paketo Stacks genannt sind. Tiny ist ein minimalistisches Image mit ca. 20Mb size + deiner App, funktioniert allerdings nur für native Apps. Es ist nicht Distroless sondern nur Distroless-like. Distroless Images werden wohl in unserem Umfeld mit Java Applikationen nicht möglich sein. Gerade wegen diesen Paketo Stacks, finde ich Paketo zu bevorzugen. Weiter gefällt mir die Einfachheit von Paketo. Der Paketo Builder erkennt für den Maven Build alle Attribute die zum Build benötigt werden, wo bei Heroku ein Configfile system.properties im root Verzeichnis abgelegt werden muss, was etwas unverständlich erscheint.

### Google Buildpacks

Auch Google sagt sie sind 100% buildpacks.io kompatibel, ein Austausch des Builders von Paketo zu Google hat jedoch nicht geklappt. Der Google Builder ist eher auf die Google Bedürfnisse zugeschnitten und macht wohl weniger Sinn für uns. Zudem scheint er verglichen mit Paketo und Heroku kein umfangreiches Buildpack-Ökosystem zu haben.

### Welches Buildpack sollen wir verwenden

Im Java Fall würde ich auf die Paketo Builder und Buildpacks setzen, da sie verschiedene Builders wie auch Runtime Image-Auswahlen bieten.   
Zudem bauen die Community Buildpack Tekton Tasks auf den Paketo Buildern auf, was eine gute Basis für unsere spezifischen Tasks ist.

Die Heroku supported Builder und Buildpacks sind alle sehr gut dokumentiert. Auch die Paketo Buildpacks haben eine ausreichende Doku. Für spezifische Tasks muss also das entsprechende Buildpack gefunden werden.   
Dazu kann ein Heroku Buildpack oder ein Paketo Buildpack verwendet werden. Die Google Buildpacks verfolgen wir vorerst nicht weiter, ausser es gibt einen speziellen Request oder Usecase dafür.

### Matrix

In der Matrix habe ich primär die für die SBB relevanten Technologien aufgelistet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | [**Paketo base**](https://paketo.io/docs/) | [**Paketo full**](https://paketo.io/docs/) | [**Paketo tiny**](https://paketo.io/docs/) | [**Heroku**](https://devcenter.heroku.com/articles/buildpacks) | [**Google**](https://github.com/GoogleCloudPlatform/buildpacks) |
| Builder Image | ubuntu:bionic + openssl + CA certs + compilers + shell utilities | ubuntu:bionic + many common C libraries and utilities | ubuntu:bionic + openssl +  CA certs + compilers + shell utilities | Technologie spezifische Baseimages | Keine Infos gefunden |
| Runtime Image | ubuntu:bionic + openssl + CA certs | ubuntu:bionic + many  common libraries and utilities | distroless-like bionic +  glibc + openssl + CA certs | Ubuntu 20 base image |  |
| Tekton compliant | :tick: | :tick: | :tick: | :tick: | :tick: |
| Common C Libraries |  | :tick: |  |  |  |
| Java Graal Native Images | :tick: | :tick: | :tick: | :information: (Community) | :tick: (experimental) |
| Go | :tick: | :tick: | :tick: | :tick: | :tick: |
| Java (Maven) | :tick: (Version kann konfiguriert werden) | :tick: (Version kann konfiguriert werden) |  | :tick: (7,8,11,13,15,17) | :tick: (8, 11) |
| Java (Gradle) | :tick: | :tick: |  | :tick: | :tick: |
| Scala | :tick: | :tick: |  | :tick: |  |
| Node.js | :tick: | :tick: |  | :tick: | :tick: |
| NGINX | :tick: | :tick: |  | :information: (Community) |  |
| .NET Core | :tick: | :tick: |  | :information: (Community) | :tick: |
| Ruby | :tick: | :tick: |  | :tick: |  |
| Python | :tick: | :tick: |  | :tick: | :tick: |
| PHP | :tick: | :tick: |  | :tick: | :tick: |

## Buildpack Task bei ESTA Tekton

Der aktuelle [Buildpack Task](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-templates/browse/tasks/esta-buildpacks.yaml) bei ESTA Tekton ist primär auf Java ausgelegt. Da müsste man noch etwas Zeit investieren, wie man einen generischen Buildpack Task zur Verfügung stellen möchte.

## Newrelic Agent Buildpack

[ESTA-5156](https://flow.sbb.ch/browse/ESTA-5156) - Abrufen der Vorgangsdetails... STATUS

Testweise wurde versucht, das [Newrelic/Agent Buildpack](https://github.com/dpacheconr/newrelic-agent-paketo-buildpack) in den Tekton Build Prozess zu integrieren. Der einzige Weg, ein Custom Buildpack in den Tekton Task zu bringen ist über ein Custom Builder Image. Es ist nicht Teil der Paketo Builder Images und die CNB Binaries lassen aktuell keine Argumente zu, über diese man [zu verwendende Buildpacks angeben](https://buildpacks.io/docs/app-developer-guide/specify-buildpacks/) könnte, weder über ein [--buildpack Argument](https://github.com/buildpacks/tekton-integration/issues/30) noch über ein [project.toml File](https://github.com/buildpacks/tekton-integration/issues/33) im Repo, wie man das sonst über das pack CLI machen kann.

Wir haben als POC ein Buildpacks Builder Image basierend auf dem paketobuildpacks/builder:base gebaut und für den Tekton Build Task verwendet. Siehe Repo [esta-tekton-buildbacks-stack](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-buildbacks-stack/browse).

Ein Testbuild war erfolgreich und ein Java Image mit Newrelic Agent ist unter esta.docker.bin.sbb.ch/esta/esta-tekton-pipeline-java-buildpacks:newrelic verfügbar. Für das Deployment muss der Agent noch mit einem License Key aktiviert werden (siehe [Helm Chart](https://code.sbb.ch/projects/KD_ESTA_TEST/repos/esta-tekton-java-buildpacks/browse/charts/esta-tekton-pipeline-java-buildpacks/values-dev.yaml?at=refs%2Fheads%2Ffeature%2FESTA-5156-newrelic-agent#16))

## Ein paar Notizen und Links

|  |  |
| --- | --- |
| **Thema** | **Link** |
| Paketo Builder für Java, besteht aus mehreren buildpacks. | <https://github.com/paketo-buildpacks/java> |
| Paketo Buildpacks configuration Doku | <https://paketo.io/docs/reference/configuration/> |
| Doku wie man eigene Buildpacks schreibt | <https://buildpacks.io/docs/buildpack-author-guide/create-buildpack/> |
| Ein bisschen verstecktes buildpack, handelt speziell runtime env variablen. | <https://github.com/paketo-buildpacks/environment-variables> |
| Tekton Task für Buildpacks welches auf dem Paketo Java Builder basiert | <https://github.com/tektoncd/catalog/tree/main/task/buildpacks/0.3> |