Customizing BuildPacks

Modifying a buildpack

Pivotal

Overview

- After completing this lesson, you should understand:
  - The basic API of a Buildpack
  - The behavior of the Java Buildpack
  - How to configure / extend a Buildpack

Pivotal™

Roadmap

- **Buildpack API**
- Java Buildpack
- Configure / Extend Java Buildpack

Pivotal™

Buildpack API

- `/bin/detect app_directory`
  - Inspect application bits to determine buildpack applicability
- `/bin/compile app_directory cache_directory`
  - Download and install runtime, container, packages, libraries; install application bits as necessary
- `/bin/release app_directory`
  - Build application start command

Pivotal™

/bin/detect

- Inspect the app bits to determine if the buildpack knows how to handle the application

Ruby *A Programmer's Best Friend*	`Gemfile` exists?
node.js	`package.json` exists?
python™	`setup.py` exists?

/bin/compile

- 'Builds' the Droplet
- Downloads and installs any necessary runtime
  - Java VM, Ruby interpreter, JavaScript interpreter …
  - Container or web server
  - Support libraries, packages, modules
    - Java jars, Ruby gems, NPM packages
- Then install the app bits into the runtime or container

Pivotal™

/bin/compile caching

- Runtime, container, and support packages are often downloaded from sources external to Cloud Foundry
  - Depending on the buildpack
- Cloud Foundry provides a location (cache) for storing downloaded artifacts to speed subsequent staging operations

Pivotal™

/bin/release

- Builds a YAML-formatted hash with three possible keys
- On Cloud Foundry (currently) only the *web:* value is used to get the start command for the app

```
addons: []
config_vars: {}
default_process_types:
    web: <start command>
```

Pivotal™

Roadmap

- Buildpack API
- **Java Buildpack**
- Configure / Extend Java Buildpack

Pivotal™

Java Buildpack

- Supports a variety of JVM languages, containers, and frameworks with a modular, configurable, and extensible design

Java Buildpack Concepts

<u>Containers</u>

How an application is run

<u>Frameworks</u>

Additional application transformations

<u>JREs</u>
Java Runtimes

Pivotal™

Java Buildpack Concepts

Containers

Executable JARs, Groovy, Play, Servlet 2 & 3, Spring Boot CLI

Frameworks

AppDynamics, New Relic, Spring Auto-reconfiguration

JREs

OpenJDK, Oracle JDK

Container Detection Criteria

Java `main()`	`META-INF/MANIFEST.MF` exists with `Main-class` attribute set
Tomcat	`WEB-INF` directory exists
Groovy	`.groovy` file with a `main()` method, or `.groovy` file with no classes, or `.groovy` file with a shebang (`#!`) declaration
Spring Boot CLI	one or more POGO `.groovy` files with no `main()` method, and no `WEB-INF` directory
Play	`start` and `lib/play.play_*.jar` exist

Framework Detection Criteria

App Dynamics	App Dynamics service bound to app
New Relic	New Relic service bound to app
Spring AutoConfiguration	spring-core*.jar in the application directory

Pivotal™

/bin/compile Output Example

- Output example:

```
-----> Downloaded app package (11M)
-----> Downloading Open Jdk JRE 1.8.0_20 from
http://download.run.pivotal.io/openjdk/lucid/x86_64/open
jdk-1.8.0_20.tar.gz (1.0s)
       Expanding Open Jdk JRE to .java-
buildpack/open_jdk_jre (1.1s)
-----> Downloading Tomcat Instance 7.0.53 from
http://download.run.pivotal.io/tomcat/tomcat-
7.0.53.tar.gz (0.5s)
       Expanding Tomcat to .java-buildpack/tomcat (0.1s)
-----> Uploading droplet (49M)
```

See What's Going On

```
$> cf files <app-name> app

.java-buildpack.log
.java-buildpack
   open_jdk_jre
   spring_auto_reconfiguration
   tomcat
   ...
META-INF/
WEB-INF/
```

Log output from buildpack

Sandboxes for each component used during staging

Pivotal

Roadmap

- Buildpack API
- Java Buildpack
- **Configure / Extend Java Buildpack**
- Customization without Forking

Pivotal™

Customization

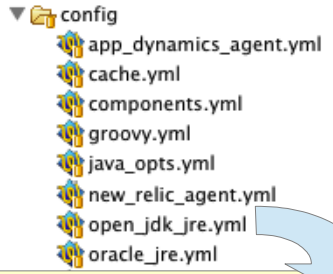
- You may alter Java buildpack
  - **Configure** artifacts used by standard JREs, Containers, and Frameworks
  - **Extend** the buildpack with your own JREs, Containers, and Frameworks

- Customization is done by forking the buildpack

- ...Or simply downloading, modifying, and zipping.

Customizing Configuration

- Most configuration options found in `/config`
  - determine behavior of a JRE, Container, or Framework




```
---
repository_root: "{default.repository.root}/openjdk/{platform}/{architecture}"
version: 1.8.0_+
memory_sizes:
  metaspace: 64m..
memory_heuristics:
  heap: 75
  metaspace: 10
  stack: 5
  native: 10
```

repository_root and *version* typically at the top of each file.

Locating Downloads

- URLs derived from repository root
  - {default.repository.root}/openjdk/{platform}/{architecture}
  - download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64
  - **index.yml** holds location of each version

```
# http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/index.yml
---
1.8.0_25: https://download.run.pivotal.io/.../x86_64/openjdk-1.8.0_25.tar.gz
1.7.0_71: https://download.run.pivotal.io/.../x86_64/openjdk-1.7.0_71.tar.gz
1.8.0_31: https://download.run.pivotal.io/.../x86_64/openjdk-1.8.0_31.tar.gz
1.7.0_75: https://download.run.pivotal.io/.../x86_64/openjdk-1.7.0_75.tar.gz
1.8.0_40: https://download.run.pivotal.io/.../x86_64/openjdk-1.8.0_40.tar.gz
...
```

Customization by Configuration: Tomcat

- Example: customizing the Tomcat artifact for download

```
# cloudfoundry/java-buildpack/config/tomcat.yml
---
tomcat:
  version: 8.0.+
  repository_root: "{default.repository.root}/tomcat"
...
```

```
# http://files.example.com/tomcat-custom/index.yml
---
8.0.18: https://download.run.pivotal.io/tomcat/tomcat-8.0.18.tar.gz
8.0.17: https://download.run.pivotal.io/tomcat/tomcat-8.0.17.tar.gz
7.0.59: https://download.run.pivotal.io/tomcat/tomcat-7.0.59.tar.gz
8.0.20: https://download.run.pivotal.io/tomcat/tomcat-8.0.20.tar.gz
```

Resource Configuration

- Tomcat container supports simple customization of **`context.xml`** and **`server.xml`**
  - Files will *overlay* sandbox provided values.

```
resources/tomcat/conf
└── context.xml
└── server.xml
```

- Not just for Tomcat
  - JDK, New Relic, etc.

Pivotal™

Extending the Buildpack - 1

- You can *extend* the Java Buildpack
  - To add different JRE, Container, or Framework
- Implement support class (Ruby) in the appropriate directory
  - with additional support classes as necessary

```
lib/java_buildpack
  └── jre
  └── container
  └── framework
```

Extending the Buildpack - 2

- Support class types have similar interfaces, following the buildpack scripts naming conventions

```
# Return String or an Array<String> that identifies the component to be
# used in staging, or nil.
def detect

# Modifies the application's file system.  Component is expected to
# transform the application's file system in whatever way is necessary
(e.g. downloading files or creating symbolic links) to support the function
of the component.  Status output written to STDOUT is expected.
def compile

# Modifies the application's runtime configuration to support the function
# of the component.  Create the command required to run the application,
# taking context values into account when creating the command.  Container
# components are expected to return the command required to run the application.
def release
```

Extending the Buildpack - 3

- Add new support class to **`config/components.yml`**

```
# Configuration for components to use in the buildpack
---
containers:
  - "JavaBuildpack::Container::DistZip"
  - "JavaBuildpack::Container::Groovy"
  - "JavaBuildpack::Container::JavaMain"
  - "JavaBuildpack::Container::PlayFramework"
  - "JavaBuildpack::Container::Ratpack"
  - "JavaBuildpack::Container::SpringBoot"
  - "JavaBuildpack::Container::SpringBootCLI"
  - "JavaBuildpack::Container::Tomcat"
  - "JavaBuildpack::Container::YOUR-CONTAINER-HERE"

jres:
  - "JavaBuildpack::jre::OpenJdkJRE"
# - "JavaBuildpack::jre::OracleJRE"

frameworks:
  - "JavaBuildpack::Framework::AppDynamicsAgent"
…
```

...or here if JRE...

...or here if framework.

More on Customization

- Much more information and documentation included in the GitHub repository

https://github.com/cloudfoundry/java-buildpack

Pivotal™

Roadmap

- Buildpack API
- Java Buildpack
- Configure / Extend Java Buildpack
- **Customization without Forking**

Pivotal™

Customization without Forking

- Simple customization of properties can be done without forking the buildpack
  - Set environment variables instead
    - Either using `cf set-env` or in the `env:` section of manifest

- Two options:
  - JAVA_OPTS variable
  - JBP_CONFIG variables

Pivotal™

Change JVM Runtime Options – I

- The **`JAVA_OPTS`** variable is recognized when app runs:

```
$> cf set-env spring-music JAVA_OPTS -showversion
Setting env variable JAVA_OPTS -showversion spring-music myorg
development jlee@pivotal.io
OK
TIP: Use 'cf restage' to ensure your env variable changes take effect

$> cf restage spring-music
... usual push output ...
2015-04-10T16:45:11.88 [App/0] ERR openjdk version "1.8.0_40-"
2015-04-10T16:45:11.88 [App/0] ERR OpenJDK Runtime Environment (build
1.8.0_40--vagrant_2015_03_26_09_03-b25)
2015-04-10T16:45:11.88 [App/0] ERR OpenJDK 64-Bit Server VM (build
25.40-b25, mixed mode)
...
$>
```

Change JVM Runtime Options – II

- Most JVM options can be specified this way
  - Except some that govern memory sizing
    - Such as **`-Xms`**, **`-Xmx`**, **`-Xss`**, **`-XX:MaxPermSize`**, **`-XX:MaxMetaspaceSize`**, **`-XX:MetaspaceSize`**, **`-XX:PermSize`**
  - Most other **`-XX`** options can be used
  - For full details see:

https://github.com/cloudfoundry/java-buildpack/blob/master/docs/framework-java_opts.md

Pivotal™

JBP_CONFIG variables

- Use environment variable to override a buildpack configuration file
  - Naming convention used:
    - **my_file.yml** → **JBP_CONFIG_MY_FILE**
  - Variable must be set to valid *inline* YAML syntax

- To change default version of Java to 7
  - Override **open_jdk_jre.yml**

```
>$ cf set-env my-application JBP_CONFIG_OPEN_JDK_JRE '[version:
1.7.0_+, memory_heuristics: {heap: 85, stack: 10}]'
```

https://github.com/cloudfoundry/java-buildpack#configuration-and-extension

Pivotal

Offline Buildpacks

- Wish to avoid downloading buildpacks from Internet
- Java Buildpack can be packaged as *Offline* Buildpack
  - Builds droplets *without* internet connection
- One-time build process
  - Internally packages latest version of each dependency within the buildpack
  - Disables remote downloads
  - About 180M in size
  - Install using `cf create-buildpack/update-buildpack`
  - https://github.com/cloudfoundry/java-buildpack/blob/master/docs/buildpack-modes.md
- ***Note:*** Pivotal CF ships with offline buildpacks!

Pivotal™

Summary

- After completing this lesson, you should have learned:
  - The basic API of a Buildpack
  - The behavior of the Java Buildpack
  - How to configure / extend a Buildpack

Pivotal™

Lab

Forking and Customizing the Java Buildpack

Pivotal™