Our Team

Cloud Tools for Java

Appu Goundan
@loosebazooka

Qingyang “Q” Chen
@coollog
Containers

“Write once, run anywhere”
Building a Java container
Me

Java Developer

Building website for pet clinic

Wants to containerize the backend

Wants container on registry ilovejava.io/petclinic-app
Build Java Docker image
构建Java镜像
So I read some tutorials
FROM ubuntu:14.04

RUN apt-get update && apt-get install -y python-software-properties software-properties-common
RUN add-apt-repository ppa:webupd8team/java

RUN echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 boolean true" | debconf-set-selections
RUN apt-get update && apt-get install -y oracle-java8-installer maven

ADD . /usr/local/petclinic

RUN cd /usr/local/petclinic && mvn install

CMD ["/usr/bin/java", "-cp", "/usr/local/petclinic/target/petclinic-1.0.jar;" "petclinic.WebServer"]
So I read some more tutorials
FROM openjdk:8
COPY target/petclinic-*.jar /app.jar
ENTRYPOINT java -jar /app.jar
FROM openjdk:8
COPY target/petclinic-*.jar /app.jar
ENTRYPOINT java -jar /app.jar

Problem: openjdk:8 is 284MB
Some more searching
FROM openjdk:8-jre-alpine 82 MB
COPY target/petclinic-*.jar /app.jar
ENTRYPOINT java -jar /app.jar
docs.docker.com/develop/develop-images/dockerfile_best-practices
.dockerignore

**

!target/petclinic-*.jar
Some more tutorials later
FROM openjdk:8-jre-alpine
COPY target/dependencies /app/dependencies
COPY target/classes /app/classes
ENTRYPOINT java -cp /app/dependencies/*:/app/classes petclinic.WebServer

$ mvn dependencies:copy-dependencies to target/dependencies/
Some more searching
...<build>
  <plugins>
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>dockerfile-maven-plugin</artifactId>
      <version>1.4.8</version>
      <configuration>
        <repository>ilovejava.io/petclinic-app</repository>
        <tag>${project.version}</tag>
      </configuration>
    </plugin>
  </plugins>
</build>
...
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin
What did we do?

1. Write first Dockerfile
2. Reduce image size
3. Don’t run installs
4. Use better base image
5. Write .dockerignore
6. Improve incremental speed
7. Switch to use a Maven plugin

Download and install Docker
Order of layers to optimize for cache hits
Use of multi-stage builds
Have elevated privileges to run Docker daemon
Understanding Docker cache mechanism and quirks

saturnism.me/talk/docker-tips-and-tricks
Containerizing with Docker

1. **Project**
2. **Dockerfile**
3. **Project** → **JAR**
4. **Dockerfile** → **d docker**
5. **docker** → **Docker Daemon**
6. **Docker Daemon** → **Container Image**
7. **Container Image** → **Registry**
8. **build**
9. **send**
10. **build**
11. **push**
I’m a Java developer, I don’t want to have to care about Dockerfiles

Some Java Developer
Somewhere
Containerizing, simplified

Project -> build -> Container image

on registry
Containerize your Java application.

Steps:
Steps:

1. Apply the plugin.

Containerize your Java application.

[Link to GitHub repository](https://github.com/GoogleContainerTools/jib)
Steps:

1. Apply the plugin.
2. `mvn jib:build` (or `gradle jib`)
$ git clone https://github.com/spring-projects/spring-petclinic && cd spring-petclinic
$ ./mvnw compile jib:build -Dimage=coollog/petclinic
Project → build → ilovejava.io/petclinic-app
      
build → Docker daemon
Extended Configuration

- JVM flags
- credentials
- labels
- environment variables
- extra files

github.com/GoogleContainerTools/jib
Demo

$ git clone https://github.com/coollog/micronaut-jib && cd micronaut-jib
$ ./gradlew jibDockerBuild
$ docker run -p 8080:8080 micronaut-jib:0.1
A “compiler” for containers
Dockerfile “script”

FROM base container image

RUN commands to install dependencies

COPY application files over

Configure the ENTRYPOINT

Run the container

Produces some layers

Produces some layers
Compiler + Containerizer
Code → Compile → Executable
Code

Compile

Executable

Java

Containerize

Container
Containers are the executables of the cloud.
Java

jar

JAR
How Jib Works
What benefits do we get from Jib

- Pure Java
- Speed
- Reproducibility

github.com/GoogleContainerTools/jib
Pure Java
A container image is a directory of files
Docker Image Format

Tarballs that compose into a single filesystem

- Tarball A: /bin, /usr, /tmp, /var
- Tarball B: /jdk
- Tarball C: /app.jar
Docker Image Format

Tarballs that compose into a single filesystem

And a container configuration

Tarball A

/bin
/usr
/tmp
/var

Tarball B

/jdk

Tarball C

/app.jar

Container configuration

Environment variables, entrypoint, etc.

github.com/GoogleContainerTools/jib
{  
"architecture": "amd64",  
"os": "linux",  
"config": {  
  "Env": [],  
  "Entrypoint": [  
    "java",  
    "-cp",  
    "/app/libs/*:/app/resources/:/app/classes/" ,  
    "com.test.HelloWorld"  
  ]  
},  
"rootfs": {  
  "type": "layers",  
  "diff_ids": [  
    "sha256:46e7865bff73b5a0c610bf9f20c91dafa2518ace8703faaffff551a4773b947" ,  
    "sha256:6189abe095d53c1c9f2bfce8f50128ee876b9a5d10f9eda1564e5f5357d6ffe61" ,  
    "sha256:e8292403028e724f0c7686ede4cd89180faa85aeb63cd0e7d560e8a459d83afe" ,  
    "sha256:ff766fffd3d45500f4af71f091a603413acb04d028ba03ae698f63819d246cb5" ,  
    "sha256:db22fdca5c6344265d841ec106e683fb39914f356fb1d8e69accb466a396dc62" ,  
    "sha256:9aa41c013edd2a6311dcd4d26129b013ba0b08c8adb51759c63501a69d27f5"  
  ]  
}  
}
Docker Image Format

Tarballs that compose into a single filesystem
And a container configuration
And a manifest

Tarball A
/bin
/usr
/tmp
/var

Tarball B
/jdk

Tarball C
/app.jar

Container configuration
Environment variables, entrypoint, etc.

Manifest
Tarballs A, B, C, and the configuration
{  
  "schemaVersion": 2,  
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",  
  "config": {  
    "mediaType": "application/vnd.docker.container.image.v1+json",  
    "digest": "sha256:181b9f9c20bb2f7f485fdd038140551a758507d6255d46f4f62b3e504948fb8f",  
    "size": 635  
  },  
  "layers": [  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "digest": "sha256:eb05f3dbdb543cc610527248690575bacbbcebabeb6ecf665b189cf18b54e3ca",  
      "size": 7695857  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "digest": "sha256:ba7c544469e514f1a9a4dec59ab640540d50992b288adb34a1a63c45bf19a2#",  
      "size": 622796  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "digest": "sha256:15705ab016593987662839b40f5a22fd1032996c90808d4a1371eb46974017d5"  
    }  
  ]
}
Jib image

github.com/GoogleContainerTools/distroless
Speed
Docker registry

Set of layers, container configurations, and manifests
Docker registry

Set of layers, container configurations, and manifests
Docker registry

Set of layers, container configurations, and manifests
Docker registry

Set of layers, container configurations, and manifests
Jib does an optimized build like

FROM gcr.io/distroless/java
COPY target/dependencies /app/dependencies
COPY target/resources /app/resources
COPY target/classes /app/classes
ENTRYPOINT java -cp /app/dependencies/*:/app/resources:/app/classes my.app.Main
Containerizing with Docker

- layer 1: build
- layer 2
- layer 3
- layer 4
- total time
- push

github.com/GoogleContainerTools/jib
Containerizing with Jib

- Layer 1: Build, Push
- Layer 2: Build, Push
- Layer 3: Build, Push
- Layer 4: Build, Push

Total time
Containerizing with Jib (cached)

layer 1
cached

layer 2
cached

layer 3
cached

layer 4

total time

github.com/GoogleContainerTools/jib
Jib vs Docker

### Small Project (20M)

<table>
<thead>
<tr>
<th></th>
<th>Jib</th>
<th>Docker</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Build</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Rebuild without changes</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Rebuild with changes to classes</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

**build time (seconds)**

github.com/GoogleContainerTools/jib
Jib vs Docker

Large Project (120M)

<table>
<thead>
<tr>
<th>Task</th>
<th>Jib</th>
<th>Docker</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Build</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rebuild without changes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rebuild with changes to classes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reproducibility
Why reproducible?

Version Control

Reduce variation between prod and dev
How?

Wipe metadata that vary between builds
Possibilities for a container “compiler”
Possibilities for a container “compiler”

- Smart inferences
- Container optimizations
- Even faster builds
- Smaller images

github.com/GoogleContainerTools/jib
Possibilities for a container “compiler”

- Smart inferences
- Container optimizations
- Even faster builds
- Smaller images

Tools for running the container
Possibilities for a container “compiler”

- Smart inferences
- Container optimizations
- Even faster builds
- Smaller images

Tools for running the container

Java Development on Kubernetes
Skaffold is a command line tool that facilitates continuous development for Kubernetes applications. You can iterate on your application source code locally then deploy to local or remote Kubernetes clusters. **Skaffold handles the workflow for building, pushing and deploying your application.** It can also be used in an automated context such as a CI/CD pipeline to leverage the same workflow and tooling when moving applications to production.

[github.com/GoogleContainerTools/skaffold](https://github.com/GoogleContainerTools/skaffold)  
[official website](https://skaffold.dev)
Development Process

- Code
- Build
- Push
- Deploy
- Connect
- Application
- k8s Config
- Update
Development Process

code

k8s config

skaffold

application
Demo
Jib Core
Java library for building containers

Google Cloud
Jib.from("busybox")
    .addLayer(Arrays.asList(Paths.get("helloworld.sh")),
              AbsoluteUnixPath.get("/"))
    .setEntrypoint("/bin/sh", "/helloworld.sh")
    .containerize(
        Containerizer.to(DockerDaemonImage.named("testjibcore"))
    );
Jib.from("busybox")
    .addLayer(Arrays.asList(Paths.get("helloworld.sh")), AbsoluteUnixPath.get(""))
    .setEntrypoint("/bin/sh", "/helloworld.sh")
    .containerize(
        Containerizer.to(DockerDaemonImage.named("testjibcore"))}};
Jib.from("busybox")
    .addLayer(Arrays.asList(Paths.get("helloworld.sh")),
    AbsoluteUnixPath.get("/"))
    .setEntrypoint("/bin/sh", "helloworld.sh")
    .containerize(
        Containerizer.to(DockerDaemonImage.named("testjibcore")));

Jib.from("busybox")
  .addLayer(Paths.get("helloworld.sh"), AbsoluteUnixPath.get("")
  .setEntrypoint("/bin/sh", "/helloworld.sh")
  .containerize(
      Containerizer.to(DockerDaemonImage.named("testjibcore")));
Demo

$ git clone https://github.com/coollog/jib-core-demo && cd jib-core-demo/helloworld
$ ./mvnw exec:java
And more...

Support for WARs

Knative Jib BuildTemplate

sbt plugin

JHipster integration

...
The Future

More containerization tools for more languages

More Skaffold integration features

Be able to write code and have it run automatically in a distributed container cluster

...
github.com/GoogleContainerTools/jib

github.com/GoogleContainerTools/skaffold

saturnism.me/talk/docker-tips-and-tricks

github.com/GoogleContainerTools/distroless

Google Cloud