版本	大小	说明
openjdk:8u312-jdk-slim-bullseye	132.76 MB	
openjdk:8u312-jre-slim	71.13 MB	
openjdk:8u312-jre-slim-buster	66.5 MB	版本最高，体积适中；
openjdk:8u312-jre-slim-bullseye	71.13 MB	
openjdk:8u312-jre-bullseye	112.5 MB	
openjdk:8u312-oraclelinux8	153.76 MB	
openjdk:8u171-jdk-alpine3.7	66.71 MB	
openjdk:8u212-jre-alpine3.9	55.02 MB	镜像最小，但是不是基于主线代码的,不支持openJDK的发行版；只支持早期的OPenJDK的版本
fabric8/java-alpine-openjdk8-jre:1.9.0	68.3 MB	内置了jvm和prometheus的监控客户端；

demo后端工程

application.java

```
package com.example.demospringboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoSpringbootApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoSpringbootApplication.class, args);
    }

}
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo-springboot</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo-springboot</name>
    <description>demo-springboot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>demo</finalName>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

打出来的jar包 demo.jar 18.7M；

不同的镜像制作和运行对比

openjdk:8u312-jre-slim-buster

```
FROM openjdk:8u312-jre-slim-buster

WORKDIR /data/cycube/
COPY target/demo.jar /data/cycube/
```

```
EXPOSE 8080

ENV APP_PORT=8080
ENV TZ=Asia/Shanghai
ENV JAVA_OPTS="-Xms512m -Xmx1024m -Xss256k -XX:MetaspaceSize=512m -
XX:MaxMetaspaceSize=512m -XX:+UnlockExperimentalVMOptions -XX:+DisableExplicitGC
-XX:+UseCGroupMemoryLimitForHeap -XX:+HeapDumpOnOutOfMemoryError -
Djava.security.egd=file:/dev/./urandom"
ENV EXT_ARG="-Dserver.port=8080 -Dspring.cloud.nacos.config.enabled=false -
Dspring.cloud.nacos.discovery.server-addr=svc-nacos:8848 -
Dspring.cloud.nacos.discovery.namespace=public"

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

ENTRYPOINT java  $JAVA_OPTS $EXT_ARG -Dserver.port=$APP_PORT -jar
/data/cycube/demo.jar
```

制作指令:

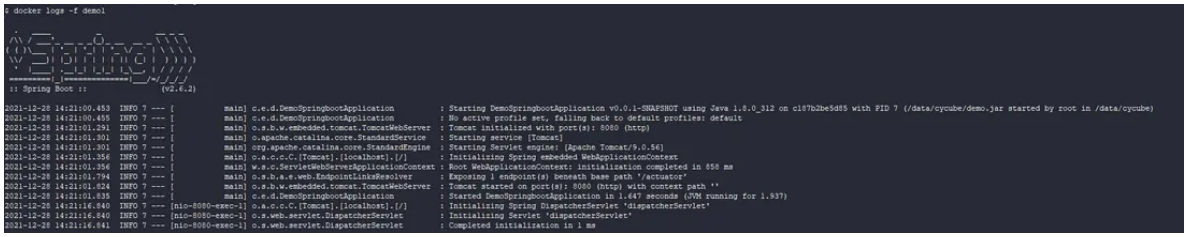
```
cd demo-springboot
docker build -t demo:v1 -f Dockerfile ./
```

大小： 207M ;

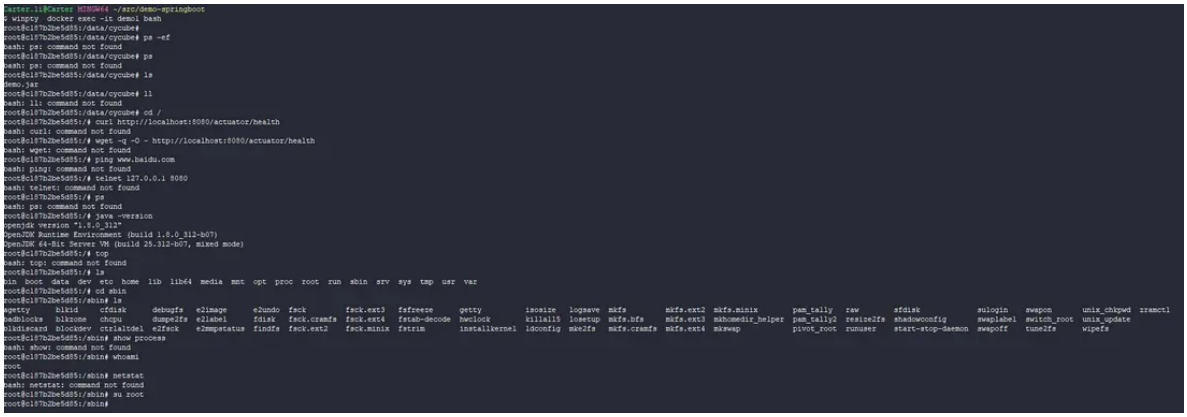
运行效果:

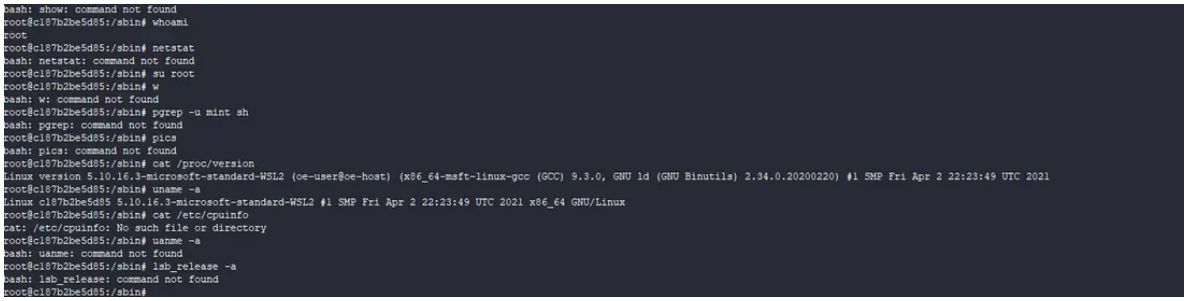
```
docker run --name demo1 -p 8081:8080 -d demo:v1
```

日志:



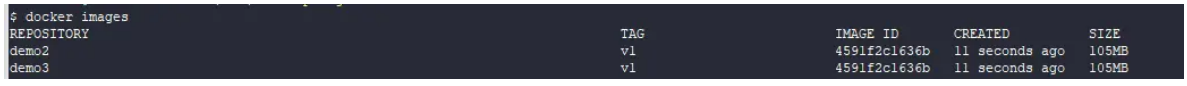
内部进程情况： ps , telnet,ping curl wget 都没有安装;





openjdk:8u212-jre-alpine3.9

```
FROM openjdk:8u212-jre-alpine3.9

WORKDIR /data/cycube/
COPY target/demo.jar /data/cycube/

EXPOSE 8080

ENV APP_PORT=8080
ENV TZ=Asia/Shanghai
ENV JAVA_OPTS="-Xms512m -Xmx1024m -Xss256k -XX:MetaspaceSize=512m -
XX:MaxMetaspaceSize=512m -XX:+UnlockExperimentalVMOptions -XX:+DisableExplicitGC
-XX:+UseCGroupMemoryLimitForHeap -XX:+HeapDumpOnOutOfMemoryError -
Djava.security.egd=file:/dev/./urandom"
ENV EXT_ARG="-Dserver.port=8080 -Dspring.cloud.nacos.config.enabled=false -
Dspring.cloud.nacos.discovery.server-addr=svc-nacos:8848 -
Dspring.cloud.nacos.discovery.namespace=public"

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

ENTRYPOINT java  $JAVA_OPTS $EXT_ARG -Dserver.port=$APP_PORT -jar
/data/cycube/demo.jar
```

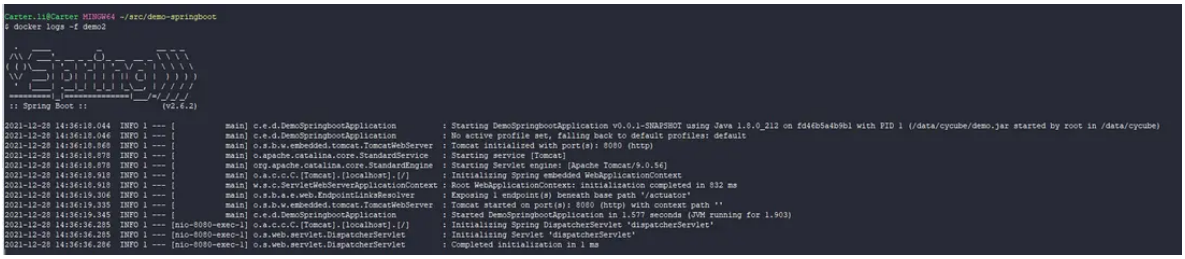



```
docker build -t demo2:v1 -f Dockerfile2 ./
```

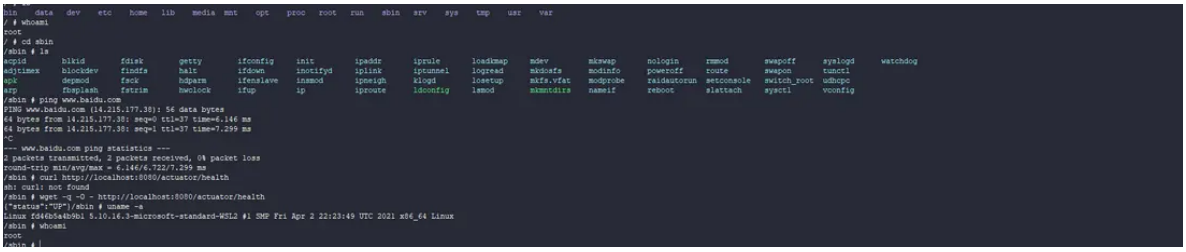
大小是： 105M

运行效果：

日志：



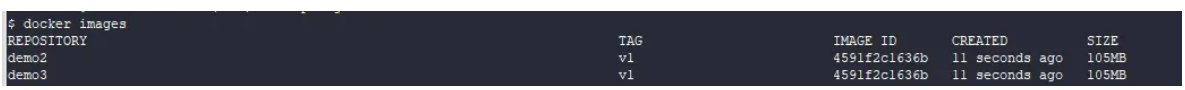
内部：



支持常见的排查网络指令。

fabric8/java-alpine-openjdk8-jre:1.9.0

```
FROM fabric8/java-alpine-openjdk8-jre:1.9.0

WORKDIR /data/cycube/
COPY target/demo.jar /data/cycube/

EXPOSE 8080

ENV APP_PORT=8080
ENV TZ=Asia/Shanghai
ENV JAVA_OPTS="-Xms512m -Xmx1024m -Xss256k -XX:MetaspaceSize=512m -
XX:MaxMetaspaceSize=512m -XX:+UnlockExperimentalVMOptions -XX:+DisableExplicitGC
-XX:+UseCGroupMemoryLimitForHeap -XX:+HeapDumpOnOutOfMemoryError -
Djava.security.egd=file:/dev/./urandom"
ENV EXT_ARG="-Dserver.port=8080 -Dspring.cloud.nacos.config.enabled=false -
Dspring.cloud.nacos.discovery.server-addr=svc-nacos:8848 -
Dspring.cloud.nacos.discovery.namespace=public"

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

ENTRYPOINT java  $JAVA_OPTS $EXT_ARG -Dserver.port=$APP_PORT -jar
/data/cycube/demo.jar
```

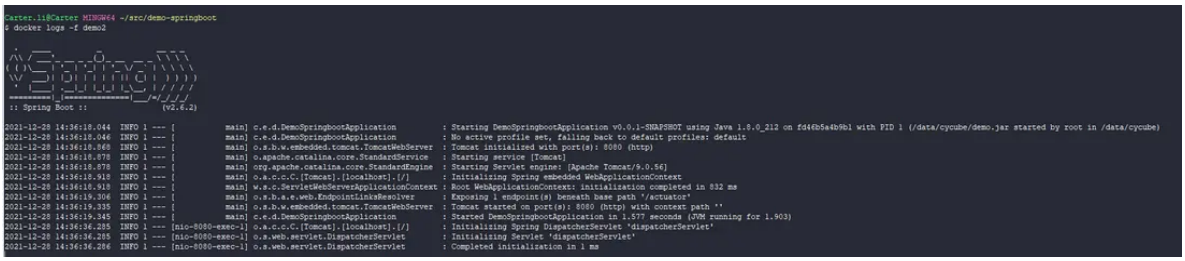


```
docker build -t demo3:v1 -f Dockerfile3 ./
```

运行效果：

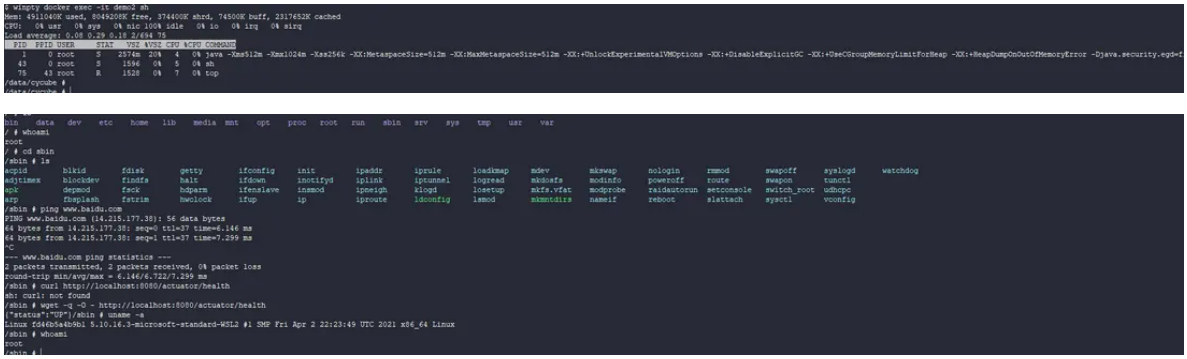

```
docker run --name demo3 -p 8083:8082 -d demo3:v1
```

日志：



```
docker logs -f demo3
```

内部运行：



使用run-java.sh

```
FROM fabric8/java-alpine-openjdk8-jre:1.9.0

WORKDIR /deployments
COPY target/demo.jar /deployments/

EXPOSE 8080

ENV APP_PORT=8080
ENV TZ=Asia/Shanghai
ENV JAVA_OPTIONS="-Xms512m -Xmx1024m -Xss256k -XX:MetaspaceSize=512m -
XX:MaxMetaspaceSize=512m -XX:+UnlockExperimentalVMOptions -XX:+DisableExplicitGC
-XX:+UseCGroupMemoryLimitForHeap -XX:+HeapDumpOnOutOfMemoryError -
Djava.security.egd=file:/dev/./urandom -Dserver.port=8080 -
Dspring.cloud.nacos.config.enabled=false -Dspring.cloud.nacos.discovery.server-
addr=svc-nacos:8848 -Dspring.cloud.nacos.discovery.namespace=public"
```

```
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

ENTRYPOINT ./run-java.sh
```

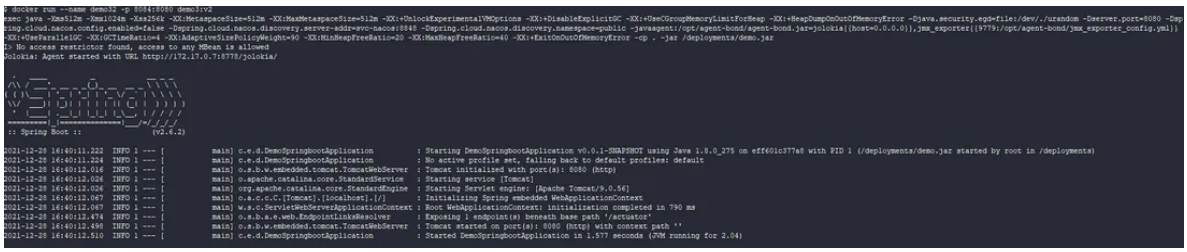
打镜像：

```
docker build -t demo3:v2 -f Dockerfile32 ./
```

镜像大小一样：120M左右；

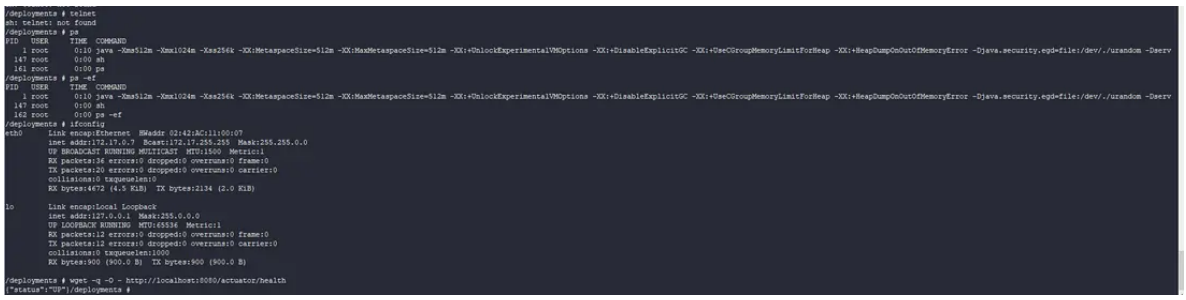
运行：

```
docker run --name demo32 -p 8084:8080 demo3:v2
```



看日志，默认打开了 jolokia的端口 8778， jmx_export的端口 9779；

登录进去看看；
官网是说支持多种shell指令：



这个镜像是基于openJDK8 jre版本；
他含有：

1. 一个Bond Agent, 使用 jolokia和 prometheus

号称万能的java启动脚本；更多能力待挖掘，后面专题分享；

对比

版本/对比维度	openjdk:8u312-jre-slim-buster	openjdk:8u212-jre-alpine3.9	fabric8/java-alpine-openjdk8-jre:19
RootWebApplicationContext Init	858ms	832ms	864ms
jvm start	1.647s	1.57s	1.693s
java 版本		openjdk version "1.8.0_212"	openjdk version "1.8.0_275"
体积大小	207M	105M	129M
基础镜像大小	187M	85M	109M
系统版本			Linux 2e8c6adc94c6 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 Linux

最终选择居中的:
fabric8/java-alpine-openjdk8-jre:19

预置了一些监控;

```
java -Xms512m -Xmx1024m -Xss256k -XX:MetaspaceSize=512m -
XX:MaxMetaspaceSize=512m -XX:+UnlockExperimentalVMOptions -XX:+DisableExplicitGC
-XX:+UseCGroupMemoryLimitForHeap -XX:+HeapDumpOnOutOfMemoryError -
Djava.security.egd=file:/dev/./urandom -Dserver.port=8080 -
Dspring.cloud.nacos.config.enabled=false -Dspring.cloud.nacos.discovery.server-
addr=svc-nacos:8848 -Dspring.cloud.nacos.discovery.namespace=public -
javaagent:/opt/agent-bond/agent-
bond.jar=jolokia{{host=0.0.0.0}},jmx_exporter{{9779:/opt/agent-
bond/jmx_exporter_config.yml}} -XX:ParallelGCThreads=1 -XX:ConcGCThreads=1 -
Djava.util.concurrent.ForkJoinPool.common.parallelism=1 -XX:CICompilerCount=2 -
XX:+UseParallelGC -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -
XX:MinHeapFreeRatio=20 -XX:MaxHeapFreeRatio=40 -XX:+ExitOnOutOfMemoryError -cp .
-jar /deployments/api-devops-rest.jar
```

分为下面几种参数:

jvm 参数	**-Xms512m -Xmx1024m -Xss256k -XX:MetaspaceSize=512m -XX:MaxMetaspaceSize=512m -XX:+UnlockExperimentalVMOptions -XX:+DisableExplicitGC -XX:+UseCGroupMemoryLimitForHeap -XX:+HeapDumpOnOutOfMemoryError -Djava.security.egd=file:/dev/./urandom -Dserver.port=8080 -Dspring.cloud.nacos.config.enabled=false -Dspring.cloud.nacos.discovery.server-addr=svc-nacos:8848 -Dspring.cloud.nacos.discovery.namespace=public**
agent 参数	-javaagent:/opt/agent-bond/agent-bond.jar=jolokia{{host=0.0.0.0}},jmx_exporter{{9779:/opt/agent-bond/jmx_exporter_config.yml}}
jvm的垃圾收集参数	-XX:ParallelGCThreads=1 -XX:ConcGCThreads=1 -Djava.util.concurrent.ForkJoinPool.common.parallelism=1 -XX:CICompilerCount=2 -XX:+UseParallelGC -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -XX:MinHeapFreeRatio=20 -XX:MaxHeapFreeRatio=40 -XX:+ExitOnOutOfMemoryError -cp

```
#!/bin/sh
#
================================================================================
===
# Generic startup script for running arbitrary Java applications with
# being optimized for running in containers
#
# Usage:
#    # Execute a Java app:
#    ./run-java.sh <args given to Java code>
#
#    # Get options which can be used for invoking Java apps like Maven or Tomcat
#    ./run-java.sh options [....]
#
#
# This script will pick up either a 'fat' jar which can be run with "-jar"
# or you can sepcify a JAVA_MAIN_CLASS.
#
# Source and Documentation can be found
# at https://github.com/fabric8io-images/run-java-sh
#
# Env-variables evaluated in this script:
#
# JAVA_OPTIONS: Checked for already set options
# JAVA_MAX_MEM_RATIO: Ratio use to calculate a default maximum Memory, in
percent.
#                     E.g. the "50" value implies that 50% of the Memory
#                     given to the container is used as the maximum heap memory
with
#                     '-Xmx'.
#                     It defaults to "25" when the maximum amount of memory
available
#                     to the container is below 300M, otherwise defaults to
"50".
#                     It is a heuristic and should be better backed up with real
#                     experiments and measurements.
```

```
#                     For a good overviews what tuning options are available -->
#                             https://youtu.be/Vt4G-pHXfs4
#                             https://www.youtube.com/watch?v=w1rZOY5gbvk
#                             https://vimeo.com/album/4133413/video/181900266
# Also note that heap is only a small portion of the memory used by a JVM. There
are lot
# of other memory areas (metadata, thread, code cache, ...) which adds to the
overall
# size. When your container gets killed because of an OOM, then you should tune
# the absolute values.
# JAVA_INIT_MEM_RATIO: Ratio use to calculate a default intial heap memory, in
percent.
#                      By default this value is not set.
#
# The following variables are exposed to your Java application:
#
# CONTAINER_MAX_MEMORY: Max memory for the container (if running within a
container)
# MAX_CORE_LIMIT: Number of cores available for the container (if running within
a container)


# ============================================================

# Fail on a single failed command in a pipeline (if supported)
(set -o | grep -q pipefail) && set -o pipefail

# Fail on error and undefined vars
set -eu

# Save global script args
ARGS="$@"

# ksh is different for defining local vars
if [ -n "${KSH_VERSION:-}" ]; then
  alias local=typeset
fi

# Error is indicated with a prefix in the return value
check_error() {
  local error_msg="$1"
  if echo "${error_msg}" | grep -q "^ERROR:"; then
    echo "${error_msg}"
    exit 1
  fi
}

# The full qualified directory where this script is located in
script_dir() {
  # Default is current directory
  local dir=$(dirname "$0")
  local full_dir=$(cd "${dir}" && pwd)
  echo ${full_dir}
}

# Try hard to find a sane default jar-file
auto_detect_jar_file() {
  local dir="$1"
```

```
  # Filter out temporary jars from the shade plugin which start with 'original-'
  local old_dir="$(pwd)"
  cd ${dir}
  if [ $? = 0 ]; then
    # NB: Find both (single) JAR *or* WAR <https://github.com/fabric8io-
images/run-java-sh/issues/79>
    local nr_jars="$(ls 2>/dev/null | grep -e '.*\.jar$' -e '.*\.war$' | grep -v
'^original-' | wc -l | awk '{print $1}')"
    if [ "${nr_jars}" = 1 ]; then
      ls 2>/dev/null | grep -e '.*\.jar$' -e '.*\.war$' | grep -v '^original-'
      exit 0
    fi
    cd "${old_dir}"
    echo "ERROR: Neither JAVA_MAIN_CLASS nor JAVA_APP_JAR is set and ${nr_jars}
found in ${dir} (1 expected)"
  else
    echo "ERROR: No directory ${dir} found for auto detection"
  fi
}

# Check directories (arg 2...n) for a jar file (arg 1)
find_jar_file() {
  local jar="$1"
  shift;

  # Absolute path check if jar specifies an absolute path
  if [ "${jar}" != ${jar#/} ]; then
    if [ -f "${jar}" ]; then
      echo "${jar}"
    else
      echo "ERROR: No such file ${jar}"
    fi
  else
    for dir in $*; do
      if [ -f "${dir}/$jar" ]; then
        echo "${dir}/$jar"
        return
      fi
    done
    echo "ERROR: No ${jar} found in $*"
  fi
}

# Generic formula evaluation based on awk
calc() {
  local formula="$1"
  shift
  echo "$@" | awk '
    function ceil(x) {
      return x % 1 ? int(x) + 1 : x
    }
    function log2(x) {
      return log(x)/log(2)
    }
    function max2(x, y) {
      return x > y ? x : y
    }
```

```
    function round(x) {
      return int(x + 0.5)
    }
    {print '"int(${formula})"'}
  '
}

# Based on the cgroup limits, figure out the max number of core we should
utilize
core_limit() {
  local cpu_period_file="/sys/fs/cgroup/cpu/cpu.cfs_period_us"
  local cpu_quota_file="/sys/fs/cgroup/cpu/cpu.cfs_quota_us"
  if [ -r "${cpu_period_file}" ]; then
    local cpu_period="$(cat ${cpu_period_file})"

    if [ -r "${cpu_quota_file}" ]; then
      local cpu_quota="$(cat ${cpu_quota_file})"
      # cfs_quota_us == -1 --> no restrictions
      if [ ${cpu_quota:-0} -ne -1 ]; then
        echo $(calc 'ceil($1/$2)' "${cpu_quota}" "${cpu_period}")
      fi
    fi
  fi
}

max_memory() {
  # High number which is the max limit until which memory is supposed to be
  # unbounded.
  local mem_file="/sys/fs/cgroup/memory/memory.limit_in_bytes"
  if [ -r "${mem_file}" ]; then
    local max_mem_cgroup="$(cat ${mem_file})"
    local max_mem_meminfo_kb="$(cat /proc/meminfo | awk '/MemTotal/ {print
$2}')"
    local max_mem_meminfo="$(expr $max_mem_meminfo_kb \* 1024)"
    if [ ${max_mem_cgroup:-0} != -1 ] && [ ${max_mem_cgroup:-0} -lt
${max_mem_meminfo:-0} ]
    then
      echo "${max_mem_cgroup}"
    fi
  fi
}

init_limit_env_vars() {
  # Read in container limits and export the as environment variables
  local core_limit="$(core_limit)"
  if [ -n "${core_limit}" ]; then
    export CONTAINER_CORE_LIMIT="${core_limit}"
  fi

  local mem_limit="$(max_memory)"
  if [ -n "${mem_limit}" ]; then
    export CONTAINER_MAX_MEMORY="${mem_limit}"
  fi
}

init_java_major_version() {
    # Initialize JAVA_MAJOR_VERSION variable if missing
    if [ -z "${JAVA_MAJOR_VERSION:-}" ]; then
```

```
        local full_version=""

        # Parse JAVA_VERSION variable available in containers
        if [ -n "${JAVA_VERSION:-}" ]; then
            full_version="$JAVA_VERSION"
        elif [ -n "${JAVA_HOME:-}" ] && [ -r "${JAVA_HOME}/release" ]; then
            full_version="$(grep -e '^JAVA_VERSION=' ${JAVA_HOME}/release | sed
-e 's/.*\"\([0-9.]\{1,\}\).*/\1/')"
        else
            full_version=$(java -version 2>&1 | head -1 | sed -e 's/.*\"\([0-
9.]\{1,\}\).*/\1/')
        fi
        export JAVA_MAJOR_VERSION=$(echo $full_version | sed -e 's/[^0-9]*\
(1\.\)\{0,1\}\([0-9]\{1,\}\).*/\2/')
    fi
}

load_env() {
  local script_dir="$1"

  # Configuration stuff is read from this file
  local run_env_sh="run-env.sh"

  # Load default default config
  if [ -f "${script_dir}/${run_env_sh}" ]; then
    . "${script_dir}/${run_env_sh}"
  fi

  # Check also $JAVA_APP_DIR. Overrides other defaults
  # It's valid to set the app dir in the default script
  JAVA_APP_DIR="${JAVA_APP_DIR:-${script_dir}}"
  if [ -f "${JAVA_APP_DIR}/${run_env_sh}" ]; then
    . "${JAVA_APP_DIR}/${run_env_sh}"
  fi
  export JAVA_APP_DIR

  # JAVA_LIB_DIR defaults to JAVA_APP_DIR
  export JAVA_LIB_DIR="${JAVA_LIB_DIR:-${JAVA_APP_DIR}}"
  if [ -z "${JAVA_MAIN_CLASS:-}" ] && [ -z "${JAVA_APP_JAR:-}" ]; then
    JAVA_APP_JAR="$(auto_detect_jar_file ${JAVA_APP_DIR})"
    check_error "${JAVA_APP_JAR}"
  fi

  if [ -n "${JAVA_APP_JAR:-}" ]; then
    local jar="$(find_jar_file ${JAVA_APP_JAR} ${JAVA_APP_DIR} ${JAVA_LIB_DIR})"
    check_error "${jar}"
    export JAVA_APP_JAR="${jar}"
  else
    export JAVA_MAIN_CLASS
  fi
}

# Check for standard /opt/run-java-options first, fallback to run-java-options
in the path if not existing
run_java_options() {
  if [ -f "/opt/run-java-options" ]; then
    echo "$(. /opt/run-java-options)"
  else
```

```
    which run-java-options >/dev/null 2>&1
    if [ $? = 0 ]; then
      echo "$(run-java-options)"
    fi
  fi
}

debug_options() {
  if [ -n "${JAVA_ENABLE_DEBUG:-}" ] || [ -n "${JAVA_DEBUG_ENABLE:-}" ] ||  [ -n
"${JAVA_DEBUG:-}" ]; then
          local debug_port="${JAVA_DEBUG_PORT:-5005}"
    local suspend_mode="n"
    if [ -n "${JAVA_DEBUG_SUSPEND:-}" ]; then
      if ! echo "${JAVA_DEBUG_SUSPEND}" | grep -q -e '^\(false\|n\|no\|0\)$';
then
        suspend_mode="y"
      fi
    fi

    local address_prefix=""
          if [ "${JAVA_MAJOR_VERSION:-0}" -ge "9" ]; then
      address_prefix="*:"
          fi
          echo "-
agentlib:jdwp=transport=dt_socket,server=y,suspend=${suspend_mode},address=${add
ress_prefix}${debug_port}"
  fi
}

# Read in a classpath either from a file with a single line, colon separated
# or given line-by-line in separate lines
# Arg 1: path to claspath (must exist), optional arg2: application jar, which is
stripped from the classpath in
# multi line arrangements
format_classpath() {
  local cp_file="$1"
  local app_jar="${2:-}"

  local wc_out="$(wc -l $1 2>&1)"
  if [ $? -ne 0 ]; then
    echo "Cannot read lines in ${cp_file}: $wc_out"
    exit 1
  fi

  local nr_lines=$(echo $wc_out | awk '{ print $1 }')
  if [ ${nr_lines} -gt 1 ]; then
    local sep=""
    local classpath=""
    while read file; do
      local full_path="${JAVA_LIB_DIR}/${file}"
      # Don't include app jar if include in list
      if [ "${app_jar}" != "${full_path}" ]; then
        classpath="${classpath}${sep}${full_path}"
      fi
      sep=":"
    done < "${cp_file}"
    echo "${classpath}"
  else
```

```
    # Supposed to be a single line, colon separated classpath file
    cat "${cp_file}"
  fi
}

# ==============================================================================

memory_options() {
  echo "$(calc_init_memory) $(calc_max_memory)"
  return
}

# Check for memory options and set max heap size if needed
calc_max_memory() {
  # Check whether -Xmx is already given in JAVA_OPTIONS
  if echo "${JAVA_OPTIONS:-}" | grep -q -- "-Xmx"; then
    return
  fi

  if [ -z "${CONTAINER_MAX_MEMORY:-}" ]; then
    return
  fi

  # Check for the 'real memory size' and calculate Xmx from the ratio
  if [ -n "${JAVA_MAX_MEM_RATIO:-}" ]; then
    if [ "${JAVA_MAX_MEM_RATIO}" -eq 0 ]; then
      # Explicitely switched off
      return
    fi
    calc_mem_opt "${CONTAINER_MAX_MEMORY}" "${JAVA_MAX_MEM_RATIO}" "mx"
  # When JAVA_MAX_MEM_RATIO not set and JVM >= 10 no max_memory
  elif [ "${JAVA_MAJOR_VERSION:-0}" -ge "10" ]; then
    return
  elif [ "${CONTAINER_MAX_MEMORY}" -le 314572800 ]; then
    # Restore the one-fourth default heap size instead of the one-half below
300MB threshold
    # See
https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/parallel.html
#default_heap_size
    calc_mem_opt "${CONTAINER_MAX_MEMORY}" "25" "mx"
  else
    calc_mem_opt "${CONTAINER_MAX_MEMORY}" "50" "mx"
  fi
}

# Check for memory options and set initial heap size if requested
calc_init_memory() {
  # Check whether -Xms is already given in JAVA_OPTIONS.
  if echo "${JAVA_OPTIONS:-}" | grep -q -- "-Xms"; then
    return
  fi

  # Check if value set
  if [ -z "${JAVA_INIT_MEM_RATIO:-}" ] || [ -z "${CONTAINER_MAX_MEMORY:-}" ] ||
[ "${JAVA_INIT_MEM_RATIO}" -eq 0 ]; then
    return
  fi
```

```
  # Calculate Xms from the ratio given
  calc_mem_opt "${CONTAINER_MAX_MEMORY}" "${JAVA_INIT_MEM_RATIO}" "ms"
}

calc_mem_opt() {
  local max_mem="$1"
  local fraction="$2"
  local mem_opt="$3"

  local val=$(calc 'round($1*$2/100/1048576)' "${max_mem}" "${fraction}")
  echo "-X${mem_opt}${val}m"
}

c2_disabled() {
  if [ -n "${CONTAINER_MAX_MEMORY:-}" ]; then
    # Disable C2 compiler when container memory <=300MB
    if [ "${CONTAINER_MAX_MEMORY}" -le 314572800 ]; then
      echo true
      return
    fi
  fi
  echo false
}

jit_options() {
  if [ "${JAVA_MAJOR_VERSION:-0}" -ge "10" ]; then
    return
  fi
  # Check whether -XX:TieredStopAtLevel is already given in JAVA_OPTIONS
  if echo "${JAVA_OPTIONS:-}" | grep -q -- "-XX:TieredStopAtLevel"; then
    return
  fi
  if [ $(c2_disabled) = true ]; then
    echo "-XX:TieredStopAtLevel=1"
  fi
}

# Switch on diagnostics except when switched off
diagnostics_options() {
  if [ -n "${JAVA_DIAGNOSTICS:-}" ]; then
    if [ "${JAVA_MAJOR_VERSION:-0}" -ge "11" ]; then
      echo "-XX:NativeMemoryTracking=summary -Xlog:gc*:stdout:time -
XX:+UnlockDiagnosticVMOptions"
    else
      echo "-XX:NativeMemoryTracking=summary -XX:+PrintGC -XX:+PrintGCDateStamps
-XX:+PrintGCTimeStamps -XX:+UnlockDiagnosticVMOptions"
    fi
  fi
}

# Replicate thread ergonomics for tiered compilation.
# This could ideally be skipped when tiered compilation is disabled.
# The algorithm is taken from:
# OpenJDK / jdk8u / jdk8u / hotspot
# src/share/vm/runtime/advancedThresholdPolicy.cpp
ci_compiler_count() {
  local core_limit="$1"
  local log_cpu=$(calc 'log2($1)' "$core_limit")
```

```
  local loglog_cpu=$(calc 'log2(max2($1,1))' "$log_cpu")
  local count=$(calc 'max2($1*$2,1)*3/2' "$log_cpu" "$loglog_cpu")
  local c1_count=$(calc 'max2($1/3,1)' "$count")
  local c2_count=$(calc 'max2($1-$2,1)' "$count" "$c1_count")
  [ $(c2_disabled) = true ] && echo "$c1_count" || echo $(calc '$1+$2'
"$c1_count" "$c2_count")
}

cpu_options() {
  # JVMs >= 10 know about CPU limits
  if [ "${JAVA_MAJOR_VERSION:-0}" -ge "10" ]; then
    return
  fi

  local core_limit="${JAVA_CORE_LIMIT:-}"
  if [ "$core_limit" = "0" ]; then
    return
  fi

  if [ -n "${CONTAINER_CORE_LIMIT:-}" ]; then
    if [ -z ${core_limit} ]; then
      core_limit="${CONTAINER_CORE_LIMIT}"
    fi
    echo "-XX:ParallelGCThreads=${core_limit} " \
         "-XX:ConcGCThreads=${core_limit} " \
         "-Djava.util.concurrent.ForkJoinPool.common.parallelism=${core_limit} "
\
         "-XX:CICompilerCount=$(ci_compiler_count $core_limit)"
  fi
}

#-XX:MinHeapFreeRatio=20  These parameters tell the heap to shrink aggressively
and to grow conservatively.
#-XX:MaxHeapFreeRatio=40  Thereby optimizing the amount of memory available to
the operating system.
heap_ratio() {
  echo "-XX:MinHeapFreeRatio=20 -XX:MaxHeapFreeRatio=40"
}

# These parameters are necessary when running parallel GC if you want to use the
Min and Max Heap Free ratios.
# Skip setting gc_options if any other GC is set in JAVA_OPTIONS.
# -XX:GCTimeRatio=4
# -XX:AdaptiveSizePolicyWeight=90
gc_options() {
  if echo "${JAVA_OPTIONS:-}" | grep -q -- "-XX:.*Use.*GC"; then
    return
  fi

  local opts=""
  # for JVMs < 10 set GC settings
  if [ -z "${JAVA_MAJOR_VERSION:-}" ] || [ "${JAVA_MAJOR_VERSION:-0}" -lt "10"
]; then
    opts="${opts} -XX:+UseParallelGC -XX:GCTimeRatio=4 -
XX:AdaptiveSizePolicyWeight=90 $(heap_ratio)"
  fi
  if [ -z "${JAVA_MAJOR_VERSION:-}" ] || [ "${JAVA_MAJOR_VERSION:-}" != "7" ];
then
```

```
    opts="${opts} -XX:+ExitOnOutOfMemoryError"
  fi
  echo $opts
}

java_default_options() {
  # Echo options, trimming trailing and multiple spaces
  echo "$(memory_options) $(jit_options) $(diagnostics_options) $(cpu_options)
$(gc_options)" | awk '$1=$1'

}

# ===========================================================================

# parse the URL
parse_url() {
  #[scheme://][user[:password]@]host[:port][/path][?params]
  echo "$1" | sed -e "s+^\(\([^:]*\)://\)\?\(\([^:@]*\)\(:\([^@]*\)\)\?@\)\?\
([^:/?]*\)\(:\([^/?]*\)\)\?.*$+ local scheme='\2' username='\4' password='\6'
hostname='\7' port='\9'+"
}

java_proxy_options() {
  local url="$1"
  local transport="$2"
  local ret=""

  if [ -n "$url" ] ; then
    eval $(parse_url "$url")
    if [ -n "$hostname" ] ; then
      ret="-D${transport}.proxyHost=${hostname}"
    fi
    if [ -n "$port" ] ; then
      ret="$ret -D${transport}.proxyPort=${port}"
    fi
    if [ -n "$username" -o -n "$password" ] ; then
      echo "WARNING: Proxy URL for ${transport} contains authentication
credentials, these are not supported by java" >&2
    fi
  fi
  echo "$ret"
}

# Check for proxy options and echo if enabled.
proxy_options() {
  local ret=""
  ret="$(java_proxy_options "${https_proxy:-${HTTPS_PROXY:-}}" https)"
  ret="$ret $(java_proxy_options "${http_proxy:-${HTTP_PROXY:-}}" http)"

  local noProxy="${no_proxy:-${NO_PROXY:-}}"
  if [ -n "$noProxy" ] ; then
    ret="$ret -Dhttp.nonProxyHosts=$(echo "|$noProxy" | sed -e 's/,
[[:space:]]*/|/g' | sed -e 's/[[:space:]]//g' | sed -e 's/|\./|\*\./g' | cut -c
2-)"
  fi
  echo "$ret"
}
```

```
# ============================================================================

# Set process name if possible
exec_args() {
  EXEC_ARGS=""
  if [ -n "${JAVA_APP_NAME:-}" ]; then
    # Not all shells support the 'exec -a newname' syntax..
    if $(exec -a test true 2>/dev/null); then
      echo "-a '${JAVA_APP_NAME}'"
    fi
  fi
}

# Combine all java options
java_options() {
  # Normalize spaces with awk (i.e. trim and elimate double spaces)
  # See e.g. https://www.physicsforums.com/threads/awk-1-1-1-file-txt.658865/
for an explanation
  # of this awk idiom
  echo "${JAVA_OPTIONS:-} $(run_java_options) $(debug_options) $(proxy_options)
$(java_default_options)" | awk '$1=$1'
}

# Fetch classpath from env or from a local "run-classpath" file
classpath() {
  local cp_path="."
  if [ "${JAVA_LIB_DIR}" != "${JAVA_APP_DIR}" ]; then
    cp_path="${cp_path}:${JAVA_LIB_DIR}"
  fi
  if [ -z "${JAVA_CLASSPATH:-}" ] && [ -n "${JAVA_MAIN_CLASS:-}" ]; then
    if [ -n "${JAVA_APP_JAR:-}" ]; then
      cp_path="${cp_path}:${JAVA_APP_JAR}"
    fi
    if [ -f "${JAVA_LIB_DIR}/classpath" ]; then
      # Classpath is pre-created and stored in a 'run-classpath' file
      cp_path="${cp_path}:$(format_classpath ${JAVA_LIB_DIR}/classpath
${JAVA_APP_JAR:-})"
    else
      # No order implied
      cp_path="${cp_path}:${JAVA_APP_DIR}/*"
    fi
  elif [ -n "${JAVA_CLASSPATH:-}" ]; then
    # Given from the outside
    cp_path="${JAVA_CLASSPATH}"
  fi
  echo "${cp_path}"
}

# Checks if a flag is present in the arguments.
hasflag() {
    local filters="$@"
    for var in $ARGS; do
        for filter in $filters; do
          if [ "$var" = "$filter" ]; then
              echo 'true'
              return
          fi
        done
```

```
    done
}

# ============================================================================

options() {
    if [ -z ${1:-} ]; then
      java_options
      return
    fi

    local ret=""
    if [ $(hasflag --debug) ]; then
      ret="$ret $(debug_options)"
    fi
    if [ $(hasflag --proxy) ]; then
      ret="$ret $(proxy_options)"
    fi
    if [ $(hasflag --java-default) ]; then
      ret="$ret $(java_default_options)"
    fi
    if [ $(hasflag --memory) ]; then
      ret="$ret $(memory_options)"
    fi
    if [ $(hasflag --jit) ]; then
      ret="$ret $(jit_options)"
    fi
    if [ $(hasflag --diagnostics) ]; then
      ret="$ret $(diagnostics_options)"
    fi
    if [ $(hasflag --cpu) ]; then
      ret="$ret $(cpu_options)"
    fi
    if [ $(hasflag --gc) ]; then
      ret="$ret $(gc_options)"
    fi

    echo $ret | awk '$1=$1'
}

# Start JVM
run() {
  # Initialize environment
  load_env $(script_dir)

  local args
  cd ${JAVA_APP_DIR}
  if [ -n "${JAVA_MAIN_CLASS:-}" ] ; then
     args="${JAVA_MAIN_CLASS}"
  elif [ -n "${JAVA_APP_JAR:-}" ]; then
     args="-jar ${JAVA_APP_JAR}"
  else
     echo "Either JAVA_MAIN_CLASS or JAVA_APP_JAR needs to be given"
     exit 1
  fi
  # Don't put ${args} in quotes, otherwise it would be interpreted as a single
arg.
```

```
  # However it could be two args (see above). zsh doesn't like this btw, but zsh
is not
  # supported anyway.
  echo exec $(exec_args) java $(java_options) -cp "$(classpath)" ${args} "$@"
  exec $(exec_args) java $(java_options) -cp "$(classpath)" ${args} "$@"
}

# ===========================================================================
# Fire up

# Initialize JAVA_MAJOR_VERSION variable if missing
init_java_major_version

# Set env vars reflecting limits
init_limit_env_vars

first_arg=${1:-}
if [ "${first_arg}" = "options" ]; then
  # Print out options only
  shift
  options $@
  exit 0
elif [ "${first_arg}" = "run" ]; then
  # Run is the default command, but can be given to allow "options"
  # as first argument to your
  shift
fi
run "$@"
```

添加skywalkingclient包；

然后运行：

本地测试：

https://segmentfault.com/a/1190000041630590