# SpringBoot主机部署方案

1.关于主机部署

1.1. 趋势

部署环境一直在演变，从物理机，虚拟机，到容器化平台，这个趋势是明显的，是不可否认的，这个我们不做过多讨论，本文所提的方案是在需要主机（物理机、虚拟机）部署的前提下进行的。

1.2.优点

简单，没有过多的依赖与封装，简单意味着可靠，故障定位快，部署环境越复杂，因环境引起的故障定位越困难，当然此话有点绝对，但统计学上看，是这样的。

1.3.缺点

资源限制欠缺：比如限制CPU占用，内存占用，虽然java有内存限制的方法，但此方法只适用java，其他开发语言的项目，可能也有，但限制方式不统一，同时很多语言没有限制的方法。

控制能力不足：比如动态缩容，扩容，这些高级特性，需要自己写服务去实现，容器化平台很多是带这个功能。

2. 部署设计

2.1.运维分工

* 运维人员

数据库、缓存、存储、操作系统等基础中间件

系统监控

告警处理进度的跟踪及汇总

服务简单的起停操作，日志查看

* 开发人员（自己负责的服务）

服务的上线

服务线上告警及故障的处理

2.2.用户隔离

不同项目有共用机器（这个不常见）

同一个项目，不同环境有共用机器（比如预发布环境与生产环境有部分主机重叠）

同一个项目，中间件和业务服务共用机器

#以开放平台项目为例(osp)，应用运行在普通用户

useradd osp && echo Osp\_2018 | passwd --stdin osp

2.3.目录设计

为了用户隔离，应用部署目录放在用户目录下

为了避免JRE的相互干扰，在~/apps/java目录存放自己的Server JRE(比如server-jre-8u172-linux-x64.tar.gz解压缩到java目录)，并在app\_env.conf明确指定JAVA\_EXE，这样大家升级jdk相互不干扰  
/home/osp/apps

* apps  
  应用部署根目录
* {server\_port}  
  应用端口号，以支持单机部署多个实例
* app\_env.conf  
  应用启动脚本参数文件
* run.sh  
  应用启动脚本
* logs  
  应用日志目录,日志文件一定要限制大小，比如springboot 默认的一个文件10M,最多存7个文件
* {app\_code}-{app\_version}.jar  
  应用部署包
* java  
  JRE目录
* {app\_code}  
  具体的应用目录
* dir.png

2.4 应用启动脚本

脚本已经开源，欢迎大家完善这个脚本，开源项目地址：

<https://github.com/Hanson1330/spring-boot-run>

2.4.1. app\_env.conf

#java程序路径（必需）JAVA\_EXE=/home/msp/apps/java/jdk1.8.0\_172/bin/java

#注册中心地址（如果未启用注册中心，可注释）MSP\_DISCOVERY\_URI=http://ms:ms\_1234@10.34.110.160:8761/eureka/#配置文件选择PROFILES\_ACTIVE=default#服务端口（必需）SERVER\_PORT=12330#JVM配置（必需）JAVA\_OPTIONS=("-Xms256m -Xmx1024m" "-XX:PermSize128m" "-Djava.io.tmpdir=/var/tmp" "-Duser.timezone=Asia/Shanghai")#Spring Boot属性配置（有特殊配置在双引号里添加）SPRING\_BOOT\_OPTIONS=""

2.4.2 run.sh

|  |
| --- |
| #!/bin/bash<<!   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   \* Author        : 马永龙   \* Email         : yonglong.ma@hpe.com   \* Last modified : 2015-05-16 14:00   \* Filename      : run.sh   \* Description   : 服务运行脚本，支持Mac,Linux   \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  !#切换至根目录cd "$(dirname "$0")" || exit 1#根目录WORKING\_DIR="$(pwd)"#应用启动配置APP\_ENV\_CONF="${WORKING\_DIR}"/app\_env.conf#日志目录LOGS\_DIR="${WORKING\_DIR}"/logs#进程IDPID\_FILE="${WORKING\_DIR}"/app.pid#应用名称APP\_NAME="$(ls \*.jar 2>/dev/null | head -n 1)"APP\_NAME=${APP\_NAME%.jar}#操作ACTION=$1echoRed() { echo $'\e[0;31m'"$1"$'\e[0m'; }echoGreen() { echo $'\e[0;32m'"$1"$'\e[0m'; }echoYellow() { echo $'\e[0;33m'"$1"$'\e[0m'; }usage() {    echo $'\n\n\n'      echoRed "Usage: ${0} support command {start|stop|restart|status}"      echo $'\n\n\n'      exit 1  }psCheck() {    echo "-----------------All instances in this machine--------------------"      echo "$(ps -ef | grep ${APP\_NAME} | grep -E -v "grep|start|stop|status|restart")"}await() {      end=$(date +%s)    let "end+=10"      while          [[ $now -le $end ]]    do          now=$(date +%s)          sleep 1    done}#1.检查操作参数[ $# -gt 0 ] || usage#2.引入启动变量if [ -r ${APP\_ENV\_CONF} ]; then      . "${APP\_ENV\_CONF}"else      echoRed "Missing or unreadable ${APP\_ENV\_CONF}"      echo $'\n\n\n'      exit 1fi#根据PID检查是否在运行isRunningPid() {      ps -p "$1" &>/dev/null  }#根据PID\_FILE检查是否在运行isRunning() {      [[ -f "$PID\_FILE" ]] || return 1    local pid=$(cat "$PID\_FILE")      ps -p "$pid" &>/dev/null    return}#基础配置if [ "$MSP\_DISCOVERY\_URI" = "" ]; then      BASE\_OPTIONS=("--spring.profiles.active=$PROFILES\_ACTIVE" "--server.port=$SERVER\_PORT")else      BASE\_OPTIONS=("--spring.profiles.active=$PROFILES\_ACTIVE" "--server.port=$SERVER\_PORT" "--eureka.client.serviceUrl.defaultZone=$MSP\_DISCOVERY\_URI")fiif [ "$SPRING\_BOOT\_OPTIONS" = "" ]; then      RUN\_EXE="$JAVA\_EXE ${JAVA\_OPTIONS[\*]} -jar ${APP\_NAME}.jar ${BASE\_OPTIONS[\*]}"else      RUN\_EXE="$JAVA\_EXE ${JAVA\_OPTIONS[\*]} -jar ${APP\_NAME}.jar ${BASE\_OPTIONS[\*]} ${SPRING\_BOOT\_OPTIONS[\*]}"fistart() {    echo "--------------Starting $APP\_NAME:"      echo $'\n\n\n'      #检查jdk      if [ -z "$JAVA\_EXE" ]; then          echoRed "Result: Start failed,Cannot find a Java JDK. Please check JAVA\_EXE in app\_env.conf"          echo $'\n\n\n'          exit 1    fi      #检查已经运行      if isRunning "$PID\_FILE"; then          echoYellow "Result: Running, no need to start"          echo $'\n\n\n'          exit 0    fi      echo "Boot Command: nohup $RUN\_EXE"      echo $'\n\n\n'      nohup $RUN\_EXE >/dev/null 2>&1 &    disown $!    echo $! >"$PID\_FILE"      #等5秒      await      TIMEOUT=100    while (! isRunning "$APP\_PID"); do          if ((TIMEOUT-- == 0)); then              echoRed "Result: Start timeout"              echo $'\n\n\n'              exit 1        fi          sleep 1    done      echoGreen "Result: Start success,Running (PID: $(<$PID\_FILE))"      echo $'\n\n\n'      psCheck  }stop() {    echo "--------------Stopping $APP\_NAME:"      echo $'\n\n\n'      if [ ! -f "$PID\_FILE" ]; then          echoYellow "Result: Not running"          echo $'\n\n\n'          psCheck        return 0    fi      local pid=$(<${PID\_FILE})    if [ -z $pid ]; then          #pid文件存在，但进程却不存在          echoRed "Result: Not running (PID: $pid not found)"          echo $'\n\n\n'          psCheck          rm -f "$PID\_FILE"          return 0    fi      kill "$pid" 2>/dev/null      TIMEOUT=30    while isRunning $PID\_FILE; do          if ((TIMEOUT-- == 0)); then              kill -KILL "$PID" 2>/dev/null        fi          sleep 1    done      rm -f "$PID\_FILE"      echoGreen "Result: Stop success"      echo $'\n\n\n'}status() {    echo "--------------Status $APP\_NAME:"      echo $'\n\n\n'      [[ -f "$PID\_FILE" ]] || {          echoYellow "Result: Not running"          echo $'\n\n\n'          psCheck        return 1      }    local pid=$(<$PID\_FILE)    if isRunningPid $pid; then          echoGreen "Result: Running （PID: $pid )"          echo $'\n\n\n'          psCheck        return 0    else          echoRed "Result: Not running (PID: $pid not found)"          echo $'\n\n\n'          psCheck        return 1    fi}case "$ACTION" instart)      start      ;;  stop)      stop      ;;  restart)      stop      start      ;;  status)      status      ;;  \*)      usage      ;;esac#成功退出exit 0 |

2.4.3 日志配置文件(logback-spring.xml)示例

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?><configuration>      <include resource="org/springframework/boot/logging/logback/base.xml"/>      <jmxConfigurator/>      <appender name="ERROR\_FILE"                class="ch.qos.logback.core.rolling.RollingFileAppender">          <encoder>              <pattern>${FILE\_LOG\_PATTERN}</pattern>          </encoder>          <File>${LOG\_PATH}/error.log</File>          <filter class="ch.qos.logback.classic.filter.LevelFilter">              <level>ERROR</level>              <onMatch>ACCEPT</onMatch>              <onMismatch>DENY</onMismatch>          </filter>          <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">              <fileNamePattern>${LOG\_PATH}/error.log.%i</fileNamePattern>          </rollingPolicy>          <triggeringPolicy                  class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">              <MaxFileSize>10MB</MaxFileSize>          </triggeringPolicy>      </appender>      <root level="INFO">          <appender-ref ref="ERROR\_FILE"/>      </root></configuration> |