(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110673858 B
(45)授权公告日 2023.04.11

(21)申请号 201910811490.6
(22)申请日 2019.08.30
(65)同一申请的已公布的文献号
申请公布号 CN 110673858 A
(43)申请公布日 2020.01.10
(73)专利权人 四川新网银行股份有限公司
地址 610094 四川省成都市成都高新区吉泰三路8号1栋1单元26楼1-8号
(72)发明人 王崟平
(74)专利代理机构 成都智言知识产权代理有限公司 51282
专利代理师 徐金琼
(51)Int.Cl.
G06F 8/61(2018.01)
G06F 21/45(2013.01)

(56)对比文件
CN 106104467 A,2016.11.09
CN 108052333 A,2018.05.18
CN 103593192 A,2014.02.19
EP 1641215 A2,2006.03.29
WO 2009154635 A1,2009.12.23
US 9146721 B1,2015.09.29
CN 110147326 A,2019.08.20
CN 108509203 A,2018.09.07
CN 107515760 A,2017.12.26
US 2017161043 A1,2017.06.08
WO 2019043687 A2,2019.03.07
US 2002144119 A1,2002.10.03
CN 104731580 A,2015.06.24
US 2017161023 A1,2017.06.08
CN 109947452 A,2019.06.28
US 2005124320 A1,2005.06.09
CN 109614108 A,2019.04.12
US 2018262388 A1,2018.09.13 (续)

审查员 陈林

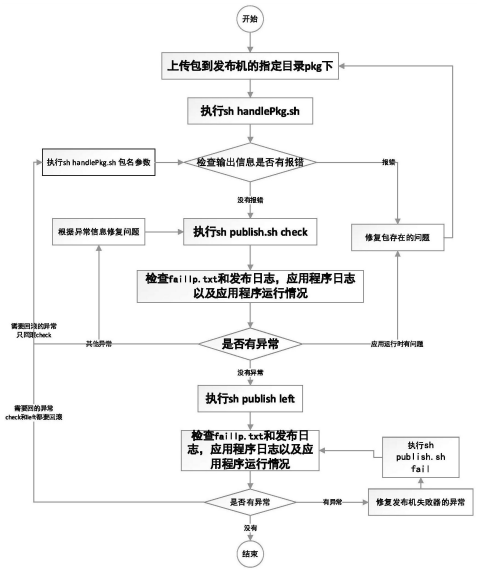
权利要求书4页 说明书13页 附图6页

(54)发明名称

一种基于ssh免密登录协议的轻量级部署方法

(57)摘要

本发明公开了一种基于ssh免密登录协议的轻量级部署方法，属于反欺诈系统部署技术领域，解决基于ssh免密登录协议的部署中所采用的server-agent模式进行自动化部时，采用开源工具和自研系统所带来的不足之处。本发明根据发布程序包的远程服务器的台数，部署分别部署发布脚本和特有的发布流程，若程序名内没有控制脚本，在发布前还需部署控制脚本，部署后，在准备好的ssh免密登录环境下，基于ssh免密登录环境，从发布机使用ssh远程免密登录命令登录到远程服务器，执行发布机中的特有的发布流程调用发布脚本，在远程服务器上进行程序包发布。本发明用于基于ssh免密登录协议的轻量级进行程序包发布。



CN 110673858 B

[转续页]

[接上页]

(56)对比文件

苗孔仿.DSCJ轻量级框架集成的技术研究与实现.《中国优秀硕士学位论文全文数据库信息科技辑》.2010,(第3期),

王小筱.基于J2EE的轻量级框架的研究与应用.《中国优秀硕士学位论文全文数据库信息科技辑》.2012,(第3期),

向涛.QoS自适应服务的构件设计与研究.《中国优秀硕士学位论文全文数据库信息科技辑》.2011,(第4期),

王小亮.基于无线Mesh网络的视频监控系统的设计与实现.《中国优秀硕士学位论文全文数据库信息科技辑》.2013,(第1期),

少弋弋.SSH免密登录.《CSDN》.2019,

weixin_33979203.SSH基于密钥登录方式部署流程.《CSDN》.2018,

1.一种基于ssh免密登录协议的轻量级部署方法,其特征在于:包括如下步骤:

S1、在远程服务器上发布程序包时,若程序包内包含有控制脚本,转到步骤S2,若程序包内未包含有控制脚本且控制脚本未部署在各远程服务器上, 在远程服务器发布程序包前,将控制脚本部署在远程服务器上,再转到步骤S3,否则,转到步骤S3,其中,程序包即指安装包;

S2、若在一台远程服务器上发布程序包时,此远程服务器为发布机,在发布机上准备ssh免密登录环境,准备后在发布机上部署发布脚本和特有的发布流程,再上传程序包到发布机的pkgDir配置目录下,再转到步骤S4发布程序包;若在多台远程服务器上发布程序包时,选一远程服务器作为发布机,在各远程服务器上准备ssh免密登录环境,准备后在发布机上部署发布脚本和特有的发布流程,再上传程序包到发布机的pkgDir配置目录下,再转到步骤S4发布程序包;

S3、若在一台远程服务器上发布程序包时,此远程服务器为发布机,在发布机发布程序包前,若运行程序包的发布,需要在外部容器上执行时,在发布机上安装外部容器,否则不安装,外部容器安装与否处理后,在发布机上准备ssh免密登录环境,准备后在发布机上部署发布脚本和特有的发布流程,再上传程序包到发布机的pkgDir配置目录下,再转到步骤S4发布程序包;若在多台远程服务器上发布程序包时,在各远程服务器上发布程序包前,若运行程序包的发布,需要在外部容器上执行时,在发布机上安装外部容器,否则不安装,外部容器安装与否处理后,选一远程服务器作为发布机,部署后在多台远程服务器上准备ssh免密登录环境,准备后在发布机上部署发布脚本和特有的发布流程,再上传程序包到发布机的pkgDir配置目录下,再转到步骤S4发布程序包;

S4、基于ssh免密登录环境,从发布机使用ssh远程免密登录命令登录到远程服务器,执行发布机中的特有的发布流程调用发布脚本,在远程服务器上进行程序包发布;

所述特有的发布流程包括如下步骤:

步骤01:上传需要发布分包到部署机的pkg目录下;

步骤02:执行sh handlePkg.sh;

步骤03:检查步骤02输出是否有异常,如果有异常问题则进入步骤04,无异常则进入步骤5;

步骤04:修复包存在问题,修复后回到步骤01;

步骤05:执行sh publish.sh check;

步骤06:执行sh publish.sh check后,检查发布日志、应用程序日志以及应用程序运行情况;如果发布日志与应用程序日志都没有报错信息,且应用程序运行正常,则验证机器发布成功,进行第07步;如果是应用程序问题则转到第04步;如果是机器环境问题,检查发布日志修复失败机器,回到第05步;

步骤07:执行sh publish.sh left;

步骤08:检查failIp.txt 和发布日志,以及应用程序运行情况;如果存在失败的机器,则根据failIp.txt里面提示的ip信息,修复发布失败的机器异常,然后到第步骤09;若无异常,则发布结束;

步骤09:执行sh publish.sh fail,然后回到第08步;

在所述步骤06中若分析出需要终止发布流程回滚到上一个版本,则执行sh

handlePkg.sh ,sh handlePkg.sh代表上个版本包名字参数,之后从步骤03开始执行,到步骤05结束,即回滚到上个版本,操作结束;

在所述步骤08中若分析出需要终止发布流程回滚到上一个版本,则执行sh handlePkg.sh ,之后从步骤03开始执行结束,即回滚到上个版本,操作结束。

2.根据权利要求1所述的一种基于ssh免密登录协议的轻量级部署方法,其特征在于:所述步骤S1中的程序包带product字样,代表生成包,程序包内包含有控制脚本的程序包类型包括zip; 程序包内未包含有控制脚本的程序包类型包括war;控制脚本包括start.sh、stop.sh、stop.sh force和status.sh,start.sh表示启动脚本,用于启动应用程序,stop.sh表示停止脚本,用于停止应用程序,stop.sh force表示强制停止脚本,用于强制停止应用程序,status.sh表示状态检查脚本,用于检查应用程序是启动还是停止。

3.根据权利要求1或2所述的一种基于ssh免密登录协议的轻量级部署方法,其特征在于:所述步骤S2和步骤S3中,

若在一台远程服务器上发布程序包时,准备ssh免密登录环境的具体步骤为:

公钥文件id_rsa.pub由发布机生成,生成后放置在账户deploy的home目录下的.ssh文件夹下的authorized_keys文件中,将公钥文件id_rsa.pub放置在authorized_keys文件中的对应命令是cat id_rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys;并在发布机上通过执行免密登录命令登录到远程服务器上将公钥文件id_rsa.pub添加到know_hosts文件中,用于实现免密登录时自动化连接;

若在多台远程服务器上发布程序包时,准备ssh免密登录环境的具体步骤为:

公钥文件id_rsa.pub由发布机生成,生成后复制到除发布机外的远程服务器上,并在除发布机外的远程服务器的deploy的home目录下建立.ssh目录,并执行chmod 700 .ssh,chmod 700 .ssh 是给.ssh目录授权为700,即拥有读、写、执行权限,将各远程服务器上的公钥文件id_rsa.pub放置在账户deploy的home目录下的.ssh文件夹下的authorized_keys文件中,将公钥文件id_rsa.pub放置在authorized_keys文件中的对应命令是cat id_rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys;并在发布机上通过执行免密登录命令登录到所有远程服务器,将公钥文件id_rsa.pub添加到各远程服务器的know_hosts文件中,用于实现免密登录时自动化连接。

4.根据权利要求1或2所述的一种基于ssh免密登录协议的轻量级部署方法,其特征在于:所述步骤S2和步骤S3中的发布脚本包括handlepkg.sh和pulish.sh,handlepkg.sh用于打包处理用户上传的程序包,pulish.sh用于将handlePkg.sh处理后的程序包发布发到各远程服务器;所述步骤S3中的外部容器包括tomcat容器和web容器。

5.根据权利要求4所述的一种基于ssh免密登录协议的轻量级部署方法,其特征在于:所述handlepkg.sh的实现逻辑为:

步骤1、判断用户上传的程序包所在pkgDir配置目录下是否有回滚包参数,若没有,转到步骤2,若有,转到步骤4,其中,回滚包参数是指pkgDir配置目录下,程序包的包名中的时间后缀为离当前时刻最近的日期;

步骤2、判断pkgDir配置目录下是否有pkgName匹配的唯一包,若有,转到步骤3,若没有,结束逻辑,其中,pkgName表示用户上传到pkgDir配置目录下的程序包的包名的正则表达式,此正则表达式是用户自定义的;

步骤3、判断pkgDir配置目录下是否已有包名为当前包的程序包，若有，去掉当前包字样，将其加上时间后缀，并重命名成备份包，重名后将用户上传的程序包加上current后缀后重命名成当前包，再结束逻辑，若没有，将用户上传的程序包加上current后缀后重命名成当前包，再结束逻辑；

步骤4、去掉当前包的当前包字样，将其加上时间后缀，并重命名成备份包，将需要回滚的程序包重命名为当前包，并结束逻辑。

6.根据权利要求4所述的一种基于ssh免密登录协议的轻量级部署方法，其特征在于：所述pulish.sh的实现逻辑为：

步骤(1)、若在一台远程服务器上发布程序包时，根据sh publish.sh参数读取ip列表ipList，即通过sh publish.sh check参数读取ip配置文件checkIp.txt和sh publish.sh fail读取ip配置文件failIp.txt放置到列表ipList；若在多台远程服务器上发布程序包时，根据sh publish.sh参数读取ip列表ipList，即通过sh publish.sh check参数读取ip配置文件checkIp.txt、sh publish.sh left参数读取ip配置文件leftIp.txt和sh publish.sh fail读取ip配置文件failIp.txt放置到列表ipList，其中，列表ipList是指远程服务器的ip列表，checkIp.txt是指配置验证的ip文档，指一个远程服务器的ip，即发布机的ip，leftIp.txt是指除发布机外要发布的远程服务器的ip文档，failIp.txt是记录失败的ip的文档，其中，sh pulish.sh check参数是指会读取checkIp.txt里面配置的ip进行远程服务器发布程序包，sh publish.sh left参数是指会读取leftIp.txt里面配置的ip对相应的远程服务器发布程序包，sh publish.sh fail参数是指会读取failIp.txt中记录失败的ip的远程服务器；

步骤(2)、从ipList中的checkIp.txt中获取一ip，执行远程停止命令停止所取ip上的应用程序，若停止成功，转到步骤(3)，若停止失败，记录失败的ip到failIp.txt；

步骤(3)、执行远程备份命令备份ip上publish Dir/appDir的应用程序，若备份成功转到步骤(4)，否则，记录失败的ip到failIp.txt，其中，publish Dir是指远程服务器上用户配置的程序包的发布目录，appDir是指用户配置的应用程序目录名，其中，“/”表示publish Dir目录下；

步骤(4)、执行远程复制命令复制pkgDir配置目录下的带current的程序包到ip对应的远程服务器上的publish Dir目录下，若成功，转到步骤(5)，否则，记录失败的ip到failIp.txt；

步骤(5)、执行远程解压重命名命令解压ip对应的远程服务器的publish Dir目录下带current的程序包，删除源程序包并重命名成appDir配置的目录名，若成功，转到步骤(6)，否则，记录失败的ip到failIp.txt；

步骤(6)、执行远程启动命令启动ip上的应用程序，若成功，转到步骤(7)，否则，记录失败的ip到failIp.txt；

步骤(7)、休眠startCheckSleepTime待应用程序启动完毕，启动完毕后，执行远程状态检查命令检查应用程序是否是启动状态，若是启动状态，转到步骤(8)，否则，记录失败的ip到failIp.txt，其中，startCheckSleepTime是指应用程序启动所需时间；

步骤(8)、判断列表ipList中的ip是否执行完毕，若执行完毕，结束流程，否则，从列表ipList选取下一ip，再执行步骤(2)至步骤(8)。

7.根据权利要求6所述的一种基于ssh免密登录协议的轻量级部署方法，其特征在于：所述步骤(2)中执行远程停止命令停止所取ip上的应用程序的具体步骤为：

步骤(2.1)、检查停止脚本的stopFilePath是否存在，若存在，转到步骤(2.2)，否则，返回0，即表示没有发布过应用程序，结束逻辑，其中，stopFilePath是指远程服务器上应用程序的停止脚本的路径，执行成功返回0，失败返回非零值；

步骤(2.2)、通过stopFilePath执行停止脚本stop.sh，执行时休眠stopSleepTime，执行后远程执行statusFilePath对应的状态检查脚本status.sh检查状态，若检查状态为停止，即停止成功，结束逻辑，否则，转到步骤(2.3)，其中，statusFilePath是指远程服务器上应用程序状态检测脚本路径，能检测到程序是运行还是停止，返回不同的状态码；

步骤(2.3)、判断是否超过stopTryLimit，若是，结束逻辑，若等于stopTryLimit，使用强杀，即强制停止，否则，转到步骤(2.2)，其中，stopTryLimit是指远程停止命令如果停止失败最多尝试的次数，其中，最后一次使用强制停止，即使用强制停止脚本stop.sh force。

8.根据权利要求7所述的一种基于ssh免密登录协议的轻量级部署方法，其特征在于：所述步骤S4的具体步骤为：

S4.1、基于ssh免密登录环境，从发布机使用ssh远程免密登录命令登录到远程服务器；

S4.2、根据sh handlepkg.sh参数执行发布脚本handlepkg.sh，执行后，若未报错，转到步骤S4.3，否则，修复包存在的问题再执行步骤S4.1；

S4.3、执行发布脚本pulish.sh中的sh pulish.sh check参数，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，转到步骤S4.4，若应用程序运行时有异常，修复包存在的问题再转到步骤S4.1，若存在其它异常，根据异常信息修复问题，再执行步骤S4.3，若存在回滚的异常，转到步骤S4.2执行sh handlepkg.sh包名参数再进行报错判断，其中，其它异常包括忘记提前预装jdk、应用程序依赖的数据库没有安装和写日志的目录权限不够；sh handlepkg.sh包名参数是指修改了一个bug刚刚上了一次线，然后发现引入了bug这个严重的问题，现要回退到上一个版本；

S4.4、若在一台远程服务器上发布程序包时，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，在远程服务器上发布后结束流程，若有除回滚外的异常，修复发布机失败的异常，执行sh publish.sh fail参数，再执行步骤S4.4，若存在回滚的异常，转到步骤S4.2执行sh handlepkg.sh包名参数再进行报错判断；若在多台远程服务器上发布程序包时，执行发布脚本pulish.sh中的sh publish.sh left参数，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，在远程服务器上发布后结束流程，若有除回滚外的异常，修复远程服务器失败的异常，执行sh publish.sh fail参数，再执行步骤S4.4，若存在回滚的异常，转到步骤S4.2执行发布脚本handlepkg.sh中的sh handlepkg.sh包名参数再进行报错判断。

一种基于ssh免密登录协议的轻量级部署方法

技术领域

[0001] 一种基于ssh免密登录协议的轻量级部署方法,用于基于ssh免密登录协议的轻量级进行程序包发布,属于反欺诈系统部署技术领域。

背景技术

[0002] 基于ssh免密登录协议的部署,现有的技术方案主要采用server-agent模式进行自动化部署,server主要作用是配置与执行发布、回滚指令,agent主要是执行server发送过来的指令执行发布、回滚。采用这种模式的一般是开源工具或者是自研系统,其中,server-agent代表一种部署模式,server:比较典型的就是web服务程序,再上面可以配置需要发布机器信息,如果ip,部署目录,磁盘,cpu信息等;agent:是代表接收server发送过来指令的一类程序,比如server会把用户上传的程序包推送给agent,然后agent把程序包部署。

[0003] 开源工具:比如puppet,有一套自己独立的的发布脚本和同步机制。使用这些工具有几个明显的缺点:

[0004] 1.需要学习该软件(开源工具)的使用方法和特有的脚本语言,耗时较长;

[0005] 2.软件(开源工具)升级通常不兼容以前的脚本,脚本升级非常麻烦,维护困难;

[0006] 3.需要安装server和agent;server通常需要独立服务器,agent需要安装在需要部署应用程序的机器上,需要消耗资源,增加了运维成本。

[0007] 自研系统:比较大型的IT公司都有一套自己的部署系统,一般server端都是web服务器,管理发布配置和执行发布命令等。agent借助一些开源工具(如rsync)做二次研发或者完全独立研发的程序,负责执行server发过来的指令完成发布任务。这种方式的缺点是:

[0008] 1.功能越强大越灵活,研发时间越长,通常是在几个月以上,并且参与人数至少要两人以上;

[0009] 2.需要安装server和agent;server通常需要独立服务器,agent需要安装在需要部署应用的机器上,需要消耗资源,增加了运维成本;

[0010] 3.在网络环境不能互通的情况下,资源消耗和运维成本会更最严重,比如C公司的A局域网与B局域网不互通,那A局域网与B局域网分别需要一台server服务器。

发明内容

[0011] 针对上述研究的问题,本发明的目的在于提供一种基于ssh免密登录协议的轻量级部署方法,解决基于ssh免密登录协议的部署中所采用的server-agent模式进行自动化部时,采用开源工具和自研系统所带来的不足之处。

[0012] 为了达到上述目的,本发明采用如下技术方案:

[0013] 一种基于ssh免密登录协议的轻量级部署方法,包括如下步骤:

[0014] S1、在远程服务器上发布程序包时,若程序包内包含有控制脚本,转到步骤S2,若

程序包内未包含有控制脚本且控制脚本未部署在各远程服务器上，在远程服务器发布程序包前，将控制脚本部署在远程服务器上，再转到步骤S3，否则，转到步骤S3，其中，程序包即指安装包；

[0015] S2、若在一台远程服务器上发布程序包时，此远程服务器为发布机，在发布机上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到步骤S4发布程序包；若在多台远程服务器上发布程序包时，选一远程服务器作为发布机，在各远程服务器上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到步骤S4发布程序包；

[0016] S3、若在一台远程服务器上发布程序包时，此远程服务器为发布机，在发布机发布程序包前，若运行程序包的发布，需要在外部容器上执行时，在发布机上安装外部容器，否则不安装，外部容器安装与否处理后，在发布机上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到步骤S4发布程序包；若在多台远程服务器上发布程序包时，在各远程服务器上发布程序包前，若运行程序包的发布，需要在外部容器上执行时，在发布机上安装外部容器，否则不安装，外部容器安装与否处理后，选一远程服务器作为发布机，部署后在多台远程服务器上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到步骤S4发布程序包；

[0017] S4、基于ssh免密登录环境，从发布机使用ssh远程免密登录命令登录到远程服务器，执行发布机中的特有的发布流程调用发布脚本，在远程服务器上进行程序包发布。

[0018] 所述特有的发布流程包括如下步骤：

[0019] 步骤01：上传需要发布分包到部署机的pkg目录下；

[0020] 步骤02：执行sh handlePkg.sh；

[0021] 步骤03：检查步骤02输出是否有异常，如果有异常问题则进入步骤04，无异常则进入步骤5；

[0022] 步骤04：修复包存在问题，修复后回到步骤01；

[0023] 步骤05：执行sh publish.sh check；

[0024] 步骤06：执行sh publish.sh check后，检查发布日志、应用程序日志以及应用程序运行情况；如果发布日志与应用程序日志都没有报错信息，且应用程序运行正常，则验证机器发布成功，进行第07步；如果是应用程序问题则转到第04步；如果是机器环境问题，检查发布日志修复失败机器，回到第05步；

[0025] 步骤07：执行sh publish.sh left；

[0026] 步骤08：检查failIp.txt 和发布日志，以及应用程序运行情况；如果存在失败的机器，则根据failIp.txt里面提示的ip信息，修复发布失败的机器异常，然后到第步骤09；若无异常，则发布结束；

[0027] 步骤09：执行sh publish.sh fail，然后回到第08步；

[0028] 在所述步骤06中若分析出需要终止发布流程回滚到上一个版本，则执行sh handlePkg.sh （上个版本包名字参数），之后从步骤03开始执行，到步骤05结束，即回滚到上个版本，操作结束；

[0029] 在所述步骤08中若分析出需要终止发布流程回滚到上一个版本，则执行sh handlePkg.sh （上个版本包名字参数),之后从步骤03开始执行结束，即回滚到上个版本，操作结束。

[0030] 进一步，所述步骤S1中的程序包带product字样，代表生成包，程序包内包含有控制脚本的程序包类型包括zip; 程序包内未包含有控制脚本的程序包类型包括war;控制脚本包括start.sh、stop.sh、stop.sh force和status.sh，start.sh表示启动脚本，用于启动应用程序，stop.sh表示停止脚本，用于停止应用程序，stop.sh force表示强制停止脚本，用于强制停止应用程序，status.sh表示状态检查脚本，用于检查应用程序是启动还是停止。

[0031] 进一步，所述步骤S2和步骤S3中，

[0032] 若在一台远程服务器上发布程序包时，准备ssh免密登录环境的具体步骤为：

[0033] 公钥文件id_rsa.pub由发布机生成，生成后放置在账户deploy的home目录下的.ssh文件夹下的authorized_keys文件中，将公钥文件id_rsa.pub放置在authorized_keys文件中的对应命令是cat id_rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys;并在发布机上通过执行免密登录命令登录到远程服务器上将公钥文件id_rsa.pub添加到know_hosts文件中，用于实现免密登录时自动化连接；

[0034] 若在多台远程服务器上发布程序包时，准备ssh免密登录环境的具体步骤为：

[0035] 公钥文件id_rsa.pub由发布机生成，生成后复制到除发布机外的远程服务器上，并在除发布机外的远程服务器的deploy的home目录下建立.ssh目录，并执行chmod 700 .ssh，chmod 700 .ssh 是给.ssh目录授权为700，即拥有读、写、执行权限，将各远程服务器上的公钥文件id_rsa.pub放置在账户deploy的home目录下的.ssh文件夹下的authorized_keys文件中，将公钥文件id_rsa.pub放置在authorized_keys文件中的对应命令是cat id_rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys;并在发布机上通过执行免密登录命令登录到所有远程服务器，将公钥文件id_rsa.pub添加到各远程服务器的know_hosts文件中，用于实现免密登录时自动化连接。

[0036] 进一步，所述步骤S2和步骤S3中的发布脚本包括handlepkg.sh和pulish.sh，handlepkg.sh用于打包处理用户上传的程序包，pulish.sh用于将handlePkg.sh处理后的程序包发布发到各远程服务器;所述步骤S3中的外部容器包括tomcat容器和web容器。

[0037] 进一步，所述handlepkg.sh的实现逻辑为：

[0038] 步骤1、判断用户上传的程序包所在pkgDir配置目录下是否有回滚包参数，若没有，转到步骤2，若有，转到步骤4，其中，回滚包参数是指pkgDir配置目录下，程序包的包名中的时间后缀为离当前时刻最近的日期；

[0039] 步骤2、判断pkgDir配置目录下是否有pkgName匹配的唯一包，若有，转到步骤3，若没有，结束逻辑，其中，pkgName表示用户上传到pkgDir配置目录下的程序包的包名的正则表达式，此正则表达式是用户自定义的；

[0040] 步骤3、判断pkgDir配置目录下是否已有包名为当前包的程序包，若有，去掉当前包字样，将其加上时间后缀，并重命名成备份包，重名后将用户上传的程序包加上current后缀后重命名成当前包，再结束逻辑，若没有，将用户上传的程序包加上current后缀后重命名成当前包，再结束逻辑；

[0041] 步骤4、去掉当前包的当前包字样，将其加上时间后缀，并重命名成备份包，将需要回滚的程序包重命名为当前包，并结束逻辑。

[0042] 进一步，所述pulish.sh的实现逻辑为：

[0043] 步骤(1)、若在一台远程服务器上发布程序包时，根据sh publish.sh参数读取ip列表ipList，即通过sh publish.sh check参数读取ip配置文件checkIp.txt和sh publish.sh fail读取ip配置文件failIp.txt放置到列表ipList；若在多台远程服务器上发布程序包时，根据sh publish.sh参数读取ip列表ipList，即通过sh publish.sh check参数读取ip配置文件checkIp.txt、sh publish.sh left参数读取ip配置文件leftIp.txt和sh publish.sh fail读取ip配置文件failIp.txt放置到列表ipList，其中，列表ipList是指远程服务器的ip列表，checkIp.txt是指配置验证的ip文档，指一个远程服务器的ip，即发布机的ip，leftIp.txt是指除发布机外要发布的远程服务器的ip文档，failIp.txt是记录失败的ip的文档，其中，sh pulish.sh check参数是指会读取checkIp.txt里面配置的ip进行远程服务器发布程序包，sh publish.sh left参数是指会读取leftIp.txt里面配置的ip对相应的远程服务器发布程序包，sh publish.sh fail参数是指会读取failIp.txt中记录失败的ip的远程服务器；

[0044] 步骤(2)、从ipList中的checkIp.txt中获取一ip，执行远程停止命令停止所取ip上的应用程序，若停止成功，转到步骤(3)，若停止失败，记录失败的ip到failIp.txt；

[0045] 步骤(3)、执行远程备份命令备份ip上publish Dir/appDir的应用程序，若备份成功转到步骤(4)，否则，记录失败的ip到failIp.txt，其中，publish Dir是指远程服务器上用户配置的程序包的发布目录，appDir是指用户配置的应用程序目录名，其中，“/”表示publish Dir目录下；

[0046] 步骤(4)、执行远程复制命令复制pkgDir配置目录下的带current的程序包到ip对应的远程服务器上的publish Dir目录下，若成功，转到步骤(5)，否则，记录失败的ip到failIp.txt；

[0047] 步骤(5)、执行远程解压重命名命令解压ip对应的远程服务器的publish Dir目录下带current的程序包，删除源程序包并重命名成appDir配置的目录名，若成功，转到步骤(6)，否则，记录失败的ip到failIp.txt；

[0048] 步骤(6)、执行远程启动命令启动ip上的应用程序，若成功，转到步骤(7)，否则，记录失败的ip到failIp.txt；

[0049] 步骤(7)、休眠startCheckSleepTime待应用程序启动完毕，启动完毕后，执行远程状态检查命令检查应用程序是否是启动状态，若是启动状态，转到步骤(8)，否则，记录失败的ip到failIp.txt，其中，startCheckSleepTime是指应用程序启动所需时间；

[0050] 步骤(8)、判断列表ipList中的ip是否执行完毕，若执行完毕，结束流程，否则，从列表ipList选取下一ip，再执行步骤(2)至步骤(8)。

[0051] 进一步，所述步骤(2)中执行远程停止命令停止所取ip上的应用程序的具体步骤为：

[0052] 步骤(2.1)、检查停止脚本的stopFilePath是否存在，若存在，转到步骤(2.2)，否则，返回0，即表示没有发布过应用程序，结束逻辑，其中，stopFilePath是指远程服务器上应用程序的停止脚本的路径，执行成功返回0，失败返回非零值；

[0053] 步骤(2.2)、通过stopFilePath执行停止脚本stop.sh，执行时休眠stopSleepTime，执行后远程执行statusFilePath对应的状态检查脚本status.sh检查状态，若检查状态为停止，即停止成功，结束逻辑，否则，转到步骤(2.3)，其中，statusFilePath是指远程服务器上应用程序状态检测脚本路径，能检测到程序是运行还是停止，返回不同的状态码；

[0054] 步骤(2.3)、判断是否超过stopTryLimit，若是，结束逻辑，若等于stopTryLimit，使用强杀，即强制停止，否则，转到步骤(2.2)，其中，stopTryLimit是指远程停止命令如果停止失败最多尝试的次数，其中，最后一次使用强制停止，即使用强制停止脚本stop.sh force。

[0055] 进一步，所述步骤S4的具体步骤为：

[0056] S4.1、基于ssh免密登录环境，从发布机使用ssh远程免密登录命令登录到远程服务器；

[0057] S4.2、根据sh handlepkg.sh参数执行发布脚本handlepkg.sh，执行后，若未报错，转到步骤S4.3，否则，修复包存在的问题再执行步骤S4.1；

[0058] S4.3、执行发布脚本pulish.sh中的sh pulish.sh check参数，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，转到步骤S4.4，若应用程序运行时有异常，修复包存在的问题再转到步骤S4.1，若存在其它异常，根据异常信息修复问题，再执行步骤S4.3，若存在回滚的异常，转到步骤S4.2执行sh handlepkg.sh包名参数再进行报错判断，其中，其它异常包括忘记提前预装jdk、应用程序依赖的数据库没有安装和写日志的目录权限不够；sh handlepkg.sh包名参数是指修改了一个bug刚刚上了一次线，然后发现引入了bug这个严重的问题，现要回退到上一个版本；

[0059] S4.4、若在一台远程服务器上发布程序包时，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，在远程服务器上发布后结束流程，若有除回滚外的异常，修复发布机失败的异常，执行sh publish.sh fail参数，再执行步骤S4.4，若存在回滚的异常，转到步骤S4.2执行sh handlepkg.sh包名参数再进行报错判断；若在多台远程服务器上发布程序包时，执行发布脚本pulish.sh中的sh publish.sh left参数，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，在远程服务器上发布后结束流程，若有除回滚外的异常，修复远程服务器失败的异常，执行sh publish.sh fail参数，再执行步骤S4.4，若存在回滚的异常，转到步骤S4.2执行发布脚本handlepkg.sh中的sh handlepkg.sh包名参数再进行报错判断。

[0060] 本发明同现有技术相比，其有益效果表现在：

[0061] 一、本发明节约了研发成本，实现超轻量；即使用本方案研发时间极端，人力消耗极少，只需一人两三天时间即可，所以这种方案特别适合没有成熟部署系统，人力时间都极端不足，又想短时间内实现自动部署的情况，整套脚本实现不足10K，却实现了所有自动化部署该做的工作，实现超轻量；

[0062] 二、本发明节约硬件资源，减轻运维成本；即这种方案没有server-agent一说，也就没有server与agent常驻进程，不会消耗任何资源，内有运维成本，如果是server-agent的模式，就需要关心进程是否存活；本发明无需安装其他任何第三方软件，只要系统支持ssh协议即可(linux 都是支持的，且自带，所以不需要安装)，这使得实现自动部署时减少

了很多工作量，如果使用server-agent模式，每个机器上至少要安装agent；

[0063] 三、本发明解决网络与资源受限问题，部署特别轻量和简单，易于复制，假如A局域网与B局域网不通，A局域网已经用了server-agent模式，因为网络不通，B环境现在不能用A环境的部署系统，B环境去申请网络环境权限可能时间太长或者就是允许打通，那么B环境就得部署一套A环境部署系统，但是并没有多余的资源，去部署这些服务，也不想维护这些服务，那么，使用本方案就能解决资源与网络受限问题，因为使用本方案时不需要独立的硬件资源，特别轻量，不需要运维，B环境复制一套A环境不到10K脚本部署，也是极为简单，不费吹灰之力；

[0064] 四、本发明在一台远程服务器上进行发布几十秒就搞定，不会出现人工可能敲错命令，遗忘步骤，时间也比较长，因为本发明不管在多少台远程服务器上进行发布，都只敲固定的几次命令即可，其它都通过脚本执行，脚本有回滚机制，万一程序有问题，可以方便直接回滚到上一个版本，可大大降低错误率，并能快速发布。

附图说明

[0065] 图1是本发明中特有的发布流程；

[0066] 图2是本发明中发布脚本handlePkg.sh的实现逻辑示意图；

[0067] 图3是本发明中发布脚本publish.sh的实现逻辑示意图；

[0068] 图4是本发明中执行远程命令停止ip上的应用的示意图，其中，执行正常的停止命令没法停掉应用程序时，就会带上force参数，强制杀死进程；

[0069] 图5为控制脚本使用tomcat做web容器的目录结构示意图；

[0070] 图6为控制脚本不使用外部容器的目录结构示意图。

具体实施方式

[0071] 下面将结合附图及具体实施方式对本发明作进一步的描述。

[0072] 一种基于ssh免密登录协议的轻量级部署方法，包括如下步骤：

[0073] S1、在远程服务器上发布程序包时，若程序包内包含有控制脚本，转到步骤S2，若程序包内未包含有控制脚本且控制脚本未部署在各远程服务器上，在远程服务器发布程序包前，将控制脚本部署在远程服务器上，再转到步骤S3，否则，转到步骤S3，其中，程序包即指安装包；

[0074] S2、若在一台远程服务器上发布程序包时，此远程服务器为发布机，在发布机上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到步骤S4发布程序包；若在多台远程服务器上发布程序包时，选一远程服务器作为发布机，在各远程服务器上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到步骤S4发布程序包；

[0075] S3、若在一台远程服务器上发布程序包时，此远程服务器为发布机，在发布机发布程序包前，若运行程序包的发布，需要在外部容器上执行时，在发布机上安装外部容器，否则不安装，外部容器安装与否处理后，在发布机上准备ssh免密登录环境，准备后在发布机上部署发布脚本和特有的发布流程，再上传程序包到发布机的pkgDir配置目录下，再转到

步骤S4发布程序包;若在多台远程服务器上发布程序包时,在各远程服务器上发布程序包前,若运行程序包的发布,需要在外部容器上执行时,在发布机上安装外部容器,否则不安装,外部容器安装与否处理后,选一远程服务器作为发布机,部署后在多台远程服务器上准备ssh免密登录环境,准备后在发布机上部署发布脚本和特有的发布流程,再上传程序包到发布机的pkgDir配置目录下,再转到步骤S4发布程序包;

[0076] S4、基于ssh免密登录环境,从发布机使用ssh远程免密登录命令登录到远程服务器,执行发布机中的特有的发布流程调用发布脚本,在远程服务器上进行程序包发布。

[0077] 进一步,所述步骤S1中的程序包带product字样,代表生成包,生产包如app-product-1.0.1.war.current,测试包如app-test-1.0.1.war.current,开发包如app-develop-1.0.1.war.current,这样约定就知道是生产包、开发包或测试包。因为生产、测试或开发通常都是对应不同的环境,会使用不同的配置,比如说数据库、生产环境、开发环境和测试环境配置的ip不同。这样就能防止把测试包搞成生产包等。

[0078] 程序包内包含有控制脚本的程序包类型包括zip; 程序包内未包含有控制脚本的程序包类型包括war;控制脚本包括start.sh、stop.sh、stop.sh force和status.sh,start.sh表示启动脚本,用于启动应用程序,stop.sh表示停止脚本,用于停止应用程序,stop.sh force表示强制停止脚本,用于强制停止应用程序,status.sh表示状态检查脚本,用于检查应用程序是启动还是停止,应用程序即是程序包解压后的应用。这些控制脚本是程序包自带的,或人为先部署的,其实现逻辑也是根据环境先准备好了的。启动脚本、停止脚本都是由程序包提供的(程序包成功发布后,对应的控制脚本即表示由应用程序提供)。若用的是tomcat作为web容器,那么启动停止tomcat都有对应的startup.sh(启动)与shutdown.sh(停止),封装成start.sh,stop.sh脚本即可,status.sh脚本检测一下进程是否存在即可。stop.sh force就是使用stop.sh不能停止是采用强制终止进程,linux系统就是直接调用kill命令。如图5所示为使用tomcat做web容器的目录结构。

[0079] 另外一种比较通用的java程序结构不使用外部容器,比如一个jar包。那么是没由现成的气筒脚本的。一般stop.sh的实现逻辑是先使用优雅停止(比如说停止前释放资源),停止不掉再使用强制停止,如图6所示。

[0080] 进一步,所述步骤S2中,

[0081] 若在一台远程服务器上发布程序包时,准备ssh免密登录环境的具体步骤为:

[0082] 公钥文件id_rsa.pub由发布机生成,生成后放置在账户deploy的home目录下的.ssh文件夹下的authorized_keys文件中,将公钥文件id_rsa.pub放置在authorized_keys文件中的对应命令是cat id_rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys;并在发布机上通过执行免密登录命令登录到远程服务器上将公钥文件id_rsa.pub添加到know_hosts文件中,用于实现免密登录时自动化连接;

[0083] 若在多台远程服务器上发布程序包时,准备ssh免密登录环境的具体步骤为:

[0084] 公钥文件id_rsa.pub由发布机生成,生成后复制到除发布机外的远程服务器上,并在除发布机外的远程服务器的deploy的home目录下建立.ssh目录,并执行chmod 700 .ssh,chmod 700 .ssh 是给.ssh目录授权为700,即拥有读、写、执行权限,将各远程服务器上的公钥文件id_rsa.pub放置在账户deploy的home目录下的.ssh文件夹下的authorized_keys文件中,将公钥文件id_rsa.pub放置在authorized_keys文件中的对应命令是cat id_

rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys;并在发布机上通过执行免密登录命令登录到所有远程服务器,将公钥文件id_rsa.pub添加到各远程服务器的know_hosts文件中,用于实现免密登录时自动化连接。具体如下:

[0085] 假如有A、B、C三台远程服务器。选A作为发布机,都是使用账户deploy执行部署。

[0086] 在A机器上执行“ssh-keygen -t rsa -P”生成公钥文件id_rsa.pub(在deploy的home目录下面的.ssh文件夹下面),并将id_rsa.pub复制到B、C远程服务器上;

[0087] 在B、C远程服务器deploy的home目录下建立.ssh目录,并执行chmod 700 .ssh;

[0088] 在A、B、C三台远程服务器上执行cat id_rsa.pub > .ssh/authorized_keys && chmod 600 .ssh/authorized_keys,将公钥文件id_rsa.pub复制到authorized_keys文件中;

[0089] 同时在A远程服务器上执行免密登录命令ssh A、ssh B、ssh C分别免密登录到A、B、C免密登录命令将A、B、C中的公钥文件id_rsa.pub添加到know_hosts文件中。注意这里为了统一,也需要免密登录到自身A远程服务器。

[0090] 所述步骤S2和步骤S3中的发布脚本包括handlepkg.sh和pulish.sh,handlepkg.sh用于打包处理用户上传的程序包,pulish.sh用于将handlePkg.sh处理后的程序包发布发到各远程服务器;所述步骤S3中的外部容器包括tomcat容器和web容器。

[0091] 进一步,所述handlepkg.sh的实现逻辑为:

[0092] 步骤1、判断用户上传的程序包所在pkgDir配置目录下是否有回滚包参数,若没有,转到步骤2,若有,转到步骤4,其中,回滚包参数是指pkgDir配置目录下,程序包的包名中的时间后缀为离当前时刻最近的日期,具体为:

[0093] 假如pkgDir配置目录为pkg,pkg目录下有:

[0094] app-product-1.0.1.war.current;

[0095] app-product-1.0.1.war.2019-06-01-11-20-11;

[0096] app-product-1.0.1.war.2019-05-01-11-19-15;

[0097] 那么 app-product-1.0.1.war.2019-06-01-11-20-11和pp-product-1.0.1.war.2019-05-01-11-19-15中,日期最近的一个就是上次发布的程序包,其日期即为回滚包参数,回滚一般就会滚上一个版本。

[0098] 步骤2、判断pkgDir配置目录下是否有pkgName匹配的唯一包,若有,转到步骤3,若没有,结束逻辑,其中,pkgName表示用户上传到pkgDir配置目录下的程序包的包名的正则表达式,此正则表达式是用户自定义的,比如说包名app-product-1.0.1.war,就可以配置成app-product*类似的linux脚本正则表达式;

[0099] 步骤3、判断pkgDir配置目录下是否已有包名为当前包的程序包,若有,去掉当前包字样,将其加上时间后缀,并重命名成备份包,重名后将用户上传的程序包加上current后缀后重命名成当前包,再结束逻辑,若没有,将用户上传的程序包加上current后缀后重命名成当前包,再结束逻辑;

[0100] 其中,回滚包即是从备份包而来,关于备份包的说明:

[0101] 假如pkgDir配置目录为pkg,pkg目录下有:

[0102] app-product-1.0.3.war.current;

[0103] app-product-1.0.2.war.2019-06-01-11-20-11;

[0104] app-product-1.0.1.war.2019-05-01-11-19-15;

[0105] 现在用户上传了一个程序包app-product-1.0.4.war,后变成

[0106] app-product-1.0.4.war;

[0107] app-product-1.0.3.war.current;

[0108] app-product-1.0.2.war.2019-06-01-11-20-11;

[0109] app-product-1.0.1.war.2019-05-01-11-19-15;

[0110] 假如现在时间是2019-07-01-16-19-15那么执行脚本sh handlPkg.sh后就会变成

[0111] app-product-1.0.4.war.current;

[0112] app-product-1.0.3.war.2019-07-01-16-19-15;

[0113] app-product-1.0.2.war.2019-06-01-11-20-11;

[0114] app-product-1.0.1.war.2019-05-01-11-19-15;

[0115] 其中,app-product-1.0.3.war.2019-07-01-16-19-15即变成了备份包。

[0116] 步骤4、去掉当前包的当前包字样,将其加上时间后缀,并重命名成备份包,将需要回滚的程序包重命名为当前包,并结束逻辑。

[0117] 具体为:

[0118] 假如用户上传到pkg目录(即pkgDir配置目录)下的程序包的包名为:app-product-1.0.1.war;

[0119] 那么发布几次应用之后pkgDir目录下面就会变成如下类似列表;

[0120] app-product-1.0.1.war.current;

[0121] app-product-1.0.1.war.2019-06-01-11-20-11;

[0122] app-product-1.0.1.war.2019-05-01-11-19-15;

[0123] 现在用户想回滚app-product-1.0.1.war.2019-05-01-11-19-15为当前包,那么执行的动作就是删除app-product-1.0.1.war.current中的当前包字样current,将其加上时间后缀为app-product-1.0.1.war.2019-06-01-11-20-11;

[0124] 然后重命名app-product-1.0.1.war.2019-05-01-11-19-15为app-product-1.0.1.war.current,最后结果为:

[0125] app-product-1.0.1.war.current;

[0126] app-product-1.0.1.war.2019-06-01-11-20-11;

[0127] 进一步,所述pulish.sh的实现逻辑为:

[0128] 步骤(1)、若在一台远程服务器上发布程序包时,根据sh publish.sh参数读取ip列表ipList,即通过sh publish.sh check参数读取ip配置文件checkIp.txt和sh publish.sh fail读取ip配置文件failIp.txt放置到列表ipList;若在多台远程服务器上发布程序包时,根据sh publish.sh参数读取ip列表ipList,即通过sh publish.sh check参数读取ip配置文件checkIp.txt、sh publish.sh left参数读取ip配置文件leftIp.txt和sh publish.sh fail读取ip配置文件failIp.txt放置到列表ipList,其中,列表ipList是指远程服务器的ip列表,checkIp.txt是指配置验证的ip文档,指一个远程服务器的ip,即发布机的ip,leftIp.txt是指除发布机外要发布的远程服务器的ip文档,failIp.txt是记录失败的ip的文档,其中,sh pulish.sh check参数是指会读取checkIp.txt里面配置的ip进行远程服务器发布程序包, sh publish.sh left参数是指会读取leftIp.txt里面配

置的ip对相应的远程服务器进行发布程序包,sh publish.sh fail参数是指会读取failIp.txt中记录失败的ip的远程服务器;

[0129] 步骤(2)、从ipList中的checkIp.txt中获取一ip,执行远程停止命令停止所取ip上的应用程序,若停止成功,转到步骤(3),若停止失败,记录失败的ip到failIp.txt;

[0130] 步骤(3)、执行远程备份命令备份ip上publish Dir/appDir的应用程序,若备份成功转到步骤(4),否则,记录失败的ip到failIp.txt,其中,publish Dir是指远程服务器上用户配置的程序包的发布目录,appDir是指用户配置的应用程序目录名其中,“/”表示publish Dir目录下,publish Dir/appDir的应用程序如:publishDir: server,appDir: myApp,erver/myApp,备份其实就是把myApp应用程序目录备份一下;myApp应用程序就是从发布机A上把用户上传的包(如app-product-1.0.1.war.current)copy到需要部署应用程序的远程服务器B上,然后解压,并改名成appDir配置的目录名,第一次发布时,B机器上appDir配置的目录是不存在的,pulish.sh脚本会判断,不存在就不会备份,第二次以后appDir目录就存在,pulish.sh脚本就会备份,备份完毕之后publish.sh脚本就会把用户上传的包copy到机器B上然后解压,并改名成appDir配置的的目录名。

[0131] 步骤(4)、执行远程复制命令复制pkgDir配置目录下的带current的程序包到ip对应的远程服务器上的publish Dir目录下,若成功,转到步骤(5),否则,记录失败的ip到failIp.txt;

[0132] 步骤(5)、执行远程解压重命名命令解压ip对应的远程服务器的publish Dir目录下带current的程序包,删除源程序包并重命名成appDir配置的目录名,若成功,转到步骤(6),否则,记录失败的ip到failIp.txt;

[0133] 重命名成appDir配置的目录名的具体例子为:

[0134] 用户的publish Dir配置目录名字叫 testApp目录;那么发布脚本publish.sh就会把包copy到testApp目录下;testApp/app-product-1.0.1.war.current.执行远程解压重命名命令之后,变成testApp/app-product-1.0.1(app-product-1.0.1是一个目录),假如用户配置的appDir叫 myapp那么就会把testApp/app-product-1.0.1变成testApp/myapp。

[0135] 步骤(6)、执行远程启动命令启动ip上的应用程序,若成功,转到步骤(7),否则,记录失败的ip到failIp.txt;

[0136] 步骤(7)、休眠startCheckSleepTime待应用程序启动完毕,启动完毕后,执行远程状态检查命令检查应用程序是否是启动状态,若是启动状态,转到步骤(8),否则,记录失败的ip到failIp.txt,其中,startCheckSleepTime是指应用程序启动所需时间;

[0137] 步骤(8)、判断列表ipList中的ip是否执行完毕,若执行完毕,结束流程,否则,从列表ipList选取下一ip,再执行步骤(2)至步骤(8)。

[0138] 命令如下:

[0139] 停止应用程序:

[0140] 远程停止命令参考:ssh -p $port $deployUser@ip "$stopFilePath $stopWay" >> $publishLog 2>&1;

[0141] 远程备份:

[0142] 如果appDir目录存在就备份该目录到appBkDir目录下

[0143] 远程备份命令参考:ssh -p $port $deployUser@$ip "test -d $appDir&& mv $appDir ${pkgBkDir} >> $publishLog 2>&1;

[0144] 远程复制:

[0145] 远程复制命令参考:scp -P $port -r $pkgDir/pkgName $deployUser@$ip:$publishDir >> $publishLog 2>&1;

[0146] pkgDir/pkgName是handlePkg.sh处理后的带current的程序包;

[0147] 解压重命名包:

[0148] 解压重命名命令参考:sh -p $port $deployUser@$ip "unzip $publishDir/pkgName -d $publishDir > /dev/null && rm -f $publishDir/ pkgName && mv $publishDir/ pkgName $publishDir/ appDir " >> $publishLog 2>&1;

[0149] 启动应用程序:

[0150] 命令参考ssh -p $port $deployUser@$ip "source ~/.bash_profile && $startFilePath" >> $publishLog 2>&1;

[0151] 添加source ~/.bash_profile这句话的目的是加载用户环境变量,如果不加的话就可能找不到用户环境变量,比如JAVA_HOME等。

[0152] 状态检查命令参考:ssh -p $port $deployUser@ip "$statusFilePath" >> $publishLog 2>&1。

[0153] 上述命令只是实现逻辑的参考(redhat linux),其他版本系统具体命令会有写差异。$port类似这要带$符号的表示是一个变量,表示端口号用户根据自己的环境配置。不同的linux版本具体命令可能会有差异,比如说ubuntu,与centos。

[0154] 进一步,所述步骤(2)中执行远程命令停止所取ip上的应用程序的具体步骤为:

[0155] 步骤(2.1)、检查停止脚本的stopFilePath是否存在,若存在,转到步骤(2.2),否则,返回0,即表示没有发布过应用程序,结束逻辑,其中,stopFilePath是指远程服务器上应用程序的停止脚本的路径,执行成功返回0,失败返回非零值;

[0156] 步骤(2.2)、通过stopFilePath执行停止脚本stop.sh,执行时休眠stopSleepTime,执行后远程执行statusFilePath对应的状态检查脚本status.sh检查状态,若检查状态为停止,即停止成功,结束逻辑,否则,转到步骤(2.3),其中,statusFilePath是指远程服务器上应用程序状态检测脚本路径,能检测到程序是运行还是停止,返回不同的状态码;

[0157] 步骤(2.3)、判断是否超过stopTryLimit,若是,结束逻辑,若等于stopTryLimit,使用强杀,即强制停止,否则,转到步骤(2.2),其中,stopTryLimit是指远程停止命令如果停止失败最多尝试的次数,其中,最后一次使用强制停止,即使用强制停止脚本stop.sh force。

[0158] 进一步,所述步骤S4的具体步骤为:

[0159] S4.1、基于ssh免密登录环境,从发布机使用ssh远程免密登录命令登录到远程服务器;

[0160] S4.2、根据sh handlepkg.sh参数执行发布脚本handlepkg.sh,执行后,若未报错,转到步骤S4.3,否则,修复包存在的问题再执行步骤S4.1,其中,修复包存在的问题,如应用程代码有错误,就修改代码之后重新打包上传到pkgDir配置的目录下;

[0161] S4.3、执行发布脚本pulish.sh中的sh pulish.sh check参数，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，转到步骤S4.4，若应用程序运行时有异常，修复包存在的问题再转到步骤S4.1，若存在其它异常，根据异常信息修复问题，再执行步骤S4.3，若存在回滚的异常，转到步骤S4.2执行sh handlepkg.sh包名参数再进行报错判断，其中，其它异常包括忘记提前预装jdk、应用程序依赖的数据库没有安装和写日志的目录权限不够；忘记提前预装jdk，如有三台远程服务器A,B,C，有个java程序包test.zip需要发布，其中，C远程服务器忘记提前预装jdk，那么C远程服务器就会发布失败，到C远程服务器安装好jdk之后再发布失败的机器就行了。失败的原因各种各样，这个要根据发布时报错的信息进行排查。这些异常情况一般都是第一次发布时，没有把环境弄好导致的，以后弄好了后就不会再发生了，其中，sh handlepkg.sh包名参数是指修改了一个bug刚刚上了一次线，然后发现引入了bug这个严重的问题，现要回退到上一个版本，假如pkgDir配置目录为pkg时，pkg目录如下：

[0162] app-product-1.0.1.war.current；

[0163] app-product-1.0.1.war.2019-06-01-11-20-11；

[0164] app-product-1.0.1.war.2019-05-01-11-19-15；

[0165] app-product-1.0.1.war.current就时刚发布的包；

[0166] app-product-1.0.1.war.2019-05-01-11-19-15就是上一个版本的包；

[0167] 如果你执行Sh handlePkg.sh app-product-1.0.1.war.2019-05-01-11-19-15，就会把app-product-1.0.1.war.2019-05-01-11-19-15变成app-product-1.0.1.war.current 重新执行pulish.sh走一次发布流程就会到上一个版本了；

[0168] S4.4、若在一台远程服务器上发布程序包时，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，在远程服务器上发布后结束流程，若有除回滚外的异常，修复发布机失败的异常，执行sh publish.sh fail参数，再执行步骤S4.4，若存在回滚的异常，转到步骤S4.2执行sh handlepkg.sh包名参数再进行报错判断；若在多台远程服务器上发布程序包时，执行发布脚本pulish.sh中的sh publish.sh left参数，执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况，若没有异常，在远程服务器上发布后结束流程，若有除回滚外的异常，修复远程服务器失败的异常，执行sh publish.sh fail参数，再执行步骤S4.4，若存在回滚的异常，转到步骤S4.2执行发布脚本handlepkg.sh中的sh handlepkg.sh包名参数再进行报错判断。sh publish.sh left 如果只有一台远程服务器，leftIp.txt就没有值可以不用执行，如果不只一台远程服务器，这里就至少还有一台远程服务器需要执行，如有三台远程服务器A,B,C那么chekIp.txt就有A的ip.leftIp.txt就有B,C的ip。

[0169] 执行后检查failIp.txt和发布日志、应用程序日志以及应用程序运行情况(即检查与failIp.txt中失败的Ip所对应的发布日志、应用程序日志以及应用程序运行情况)，发布日志是指发布脚本pulish.sh产生的发布日志，这个日志只会记录在那一步失败，比如执行停止，copy程序包到指定的远程服务器，执行启动、备份、复制等，应用程序日志是指从发布脚本pulish.sh打印的日志，日志中可发现A远程服务器的执行start.sh报错，那么登录到A远程服务器查看start.sh(比如说信息输出到start.log中，可以查看上次的目录内容)输出错误信息时，从而从日志查到错误原因是jdk没有安装，应用程序运行情况是通过应用

程序日志进行查看(java程序一般都是用log4j打的)有没有报错信息,具体如下:

[0170] 发布时 sh publish.sh check参数是读取checkIp.txt中的配置的ip进行发布,如果有发布失败的话,那failIp.txt就是checkIp.txt里面配置的ip,通过查看发布脚本publish.sh的输出日志(发布日志)或者应用程序(应用程序日志)自己打印出的日志找到错误原因之后,修复好failIp.txt里面Ip远程服务器的问题,再执行sh publish.sh fail;

[0171] 发布时sh publish.sh left参数是读取leftIp.txt中的配置的ip进行发布,如果有发布失败的话,那failIp.txt就是leftIp.txt里面配置的ip,通过查看发布脚本publish.sh的输出日志或者应用程序自己打印出的日志找到错误原因之后,修复好failIp.txt里面Ip机器的问题,再执行sh publish.sh fail;

[0172] 综上所述,本发明节约了大量研发成本,实现超快超轻量。该方案的实施一人一天(即编写脚本只需一天肯定是够的,但是部署实施需要根据机器的多少来衡量,比如说做免密登录的时候机器越多时间肯定就越长)完全可以搞定,脚本实现不超过10K,超轻量,却完成了所有自动部署该完成的工作。至今我还没有看见哪一个自动部署工具能完成此种任务,一天就能研发完毕的。很多大公司拥有的部署系统都是耗时几个月,再加上几年的维护才进入稳定阶段的。即使使用开源软件不要研发,光学习怎么使用都不可能一天轻松搞定。如果已经有应用准备上线,迫在眉睫,又没有现成的部署系统,那本方案绝对是唯一最佳选择。

[0173] 本发明部署超轻量。不足10k的部署脚本不管怎么拷贝都很方便,用到的命令和工具一般的linux系统都是自带的,不用额外安装任何软件。哪怕是一台机器也可以用自动部署脚本,进行发布和回滚操作,能防止手动操作的失误,这种情况下如果去装一个部署服务那就得不偿失了。

[0174] 本发明零消耗资源与零运维成本。部署脚本不是常驻进程,只有发布的时候才会存在。所以平时是不会消耗系统资源的,也不需要任何运维工作。

[0175] 以上仅是本发明众多具体应用范围中的代表性实施例,对本发明的保护范围不构成任何限制。凡采用变换或是等效替换而形成的技术方案,均落在本发明权利保护范围之内。

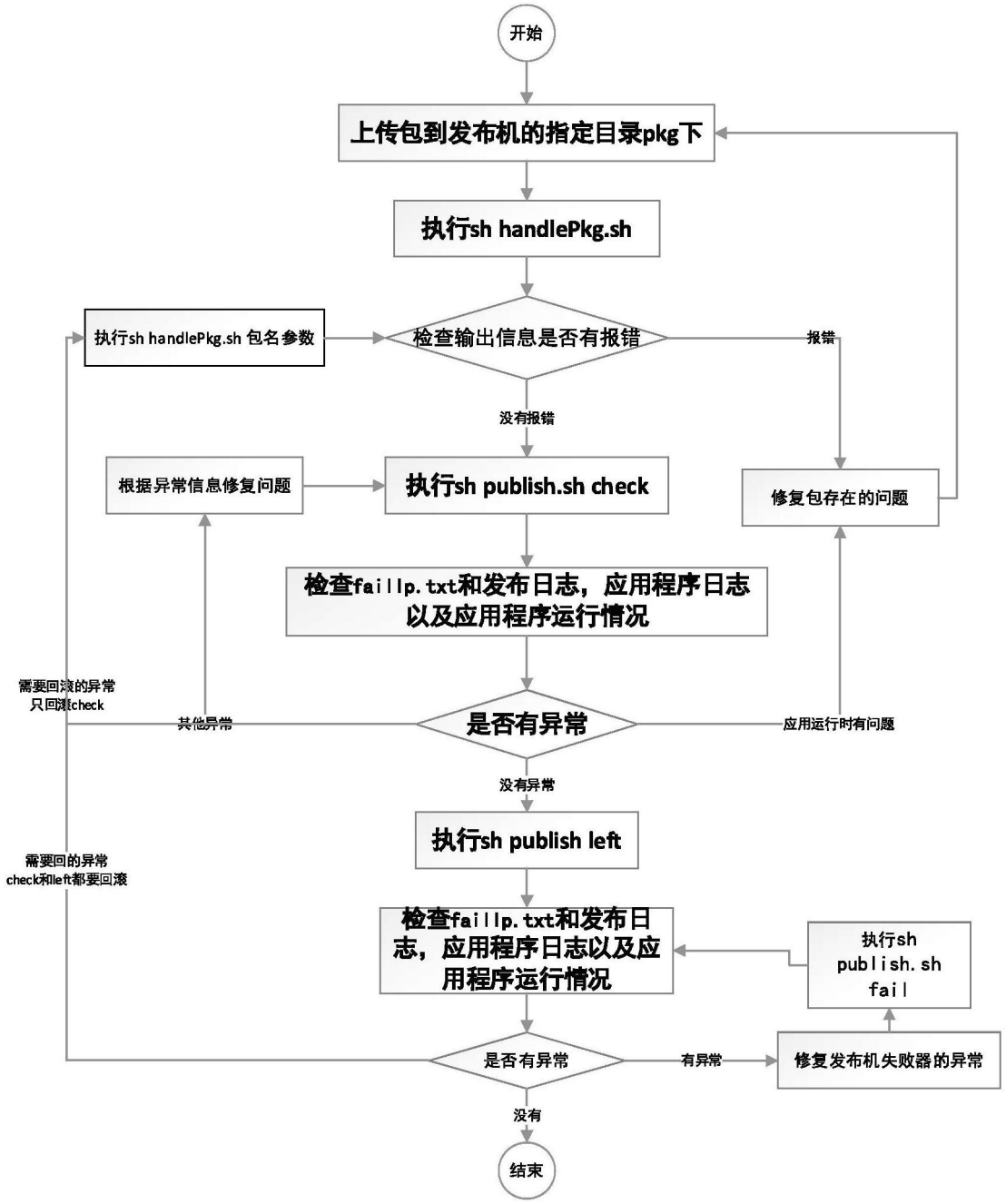

开始
上传包到发布机的指定目录pkg下
执行sh handlePkg.sh
执行sh handlePkg.sh 包名参数
检查输出信息是否有报错
报错
没有报错
根据异常信息修复问题
执行sh publish.sh check
修复包存在的问题
检查faillp.txt和发布日志，应用程序日志以及应用程序运行情况
需要回滚的异常
只回滚check
其他异常
是否有异常
应用运行时有问题
没有异常
执行sh publish left
需要回的异常
check和left都要回滚
检查faillp.txt和发布日志，应用程序日志以及应用程序运行情况
执行sh publish.sh fail
是否有异常
有异常
修复发布机失败器的异常
没有
结束

图1

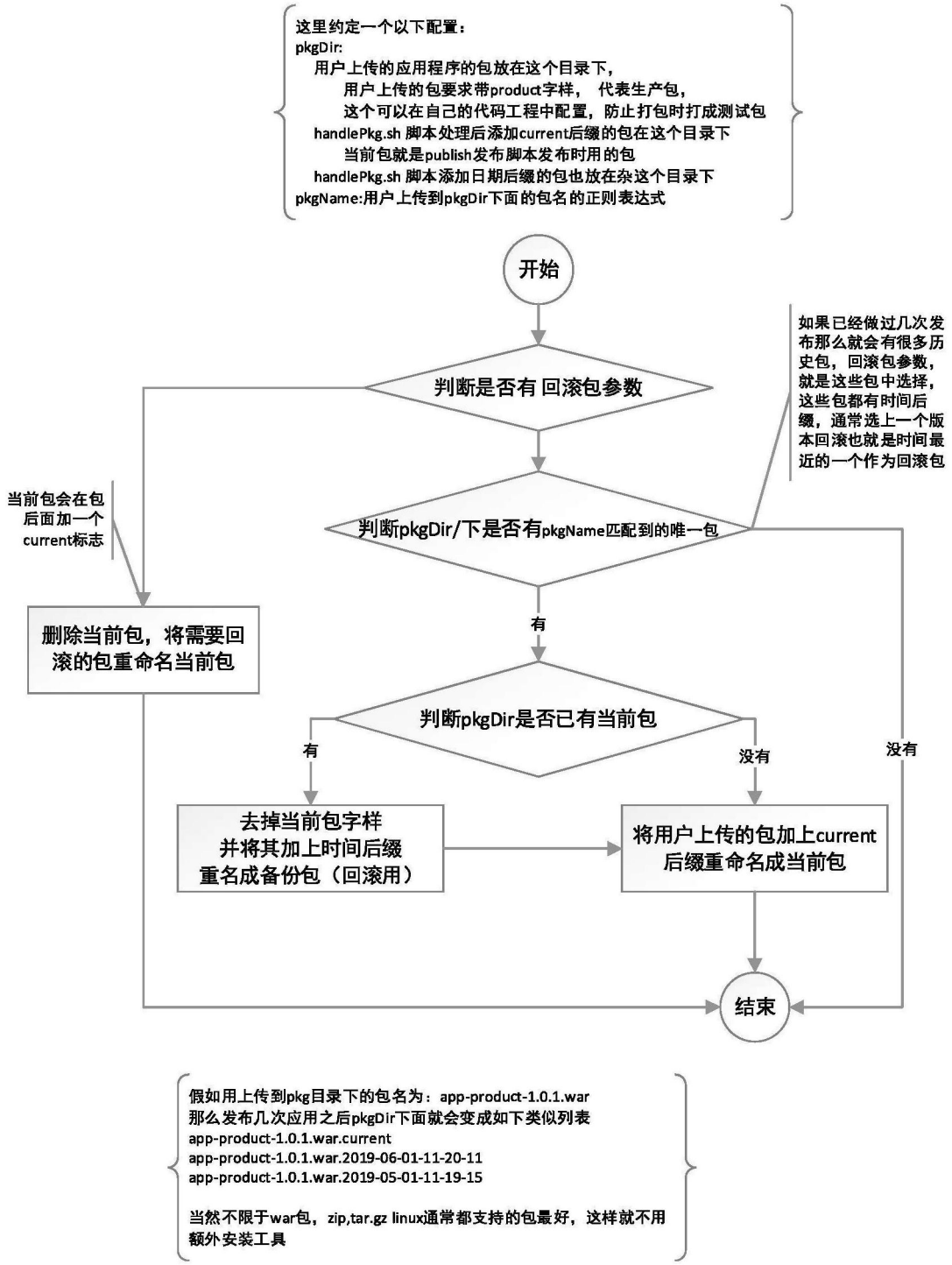

这里约定一个以下配置：
pkgDir:
用户上传的应用程序的包放在这个目录下，
用户上传的包要求带product字样， 代表生产包，
这个可以在自己的代码工程中配置，防止打包时打成测试包
handlePkg.sh 脚本处理后添加current后缀的包在这个目录下
当前包就是publish发布脚本发布时用的包
handlePkg.sh 脚本添加日期后缀的包也放在杂这个目录下
pkgName:用户上传到pkgDir下面的包名的正则表达式
开始
判断是否有 回滚包参数
如果已经做过几次发布那么就会有很多历史包，回滚包参数，就是这些包中选择，这些包都有时间后缀，通常选上一个版本回滚也就是时间最近的一个作为回滚包
当前包会在包后面加一个current标志
判断pkgDir/下是否有pkgName匹配到的唯一包
删除当前包，将需要回滚的包重命名当前包
有
判断pkgDir是否已有当前包
有
没有
没有
去掉当前包字样
并将其加上时间后缀
重名成备份包（回滚用）
将用户上传的包加上current
后缀重命名成当前包
结束
假如用上传到pkg目录下的包名为：app-product-1.0.1.war
那么发布几次应用之后pkgDir下面就会变成如下类似列表
app-product-1.0.1.war.current
app-product-1.0.1.war.2019-06-01-11-20-11
app-product-1.0.1.war.2019-05-01-11-19-15
当然不限于war包，zip,tar.gz linux通常都支持的包最好，这样就不用额外安装工具

图2

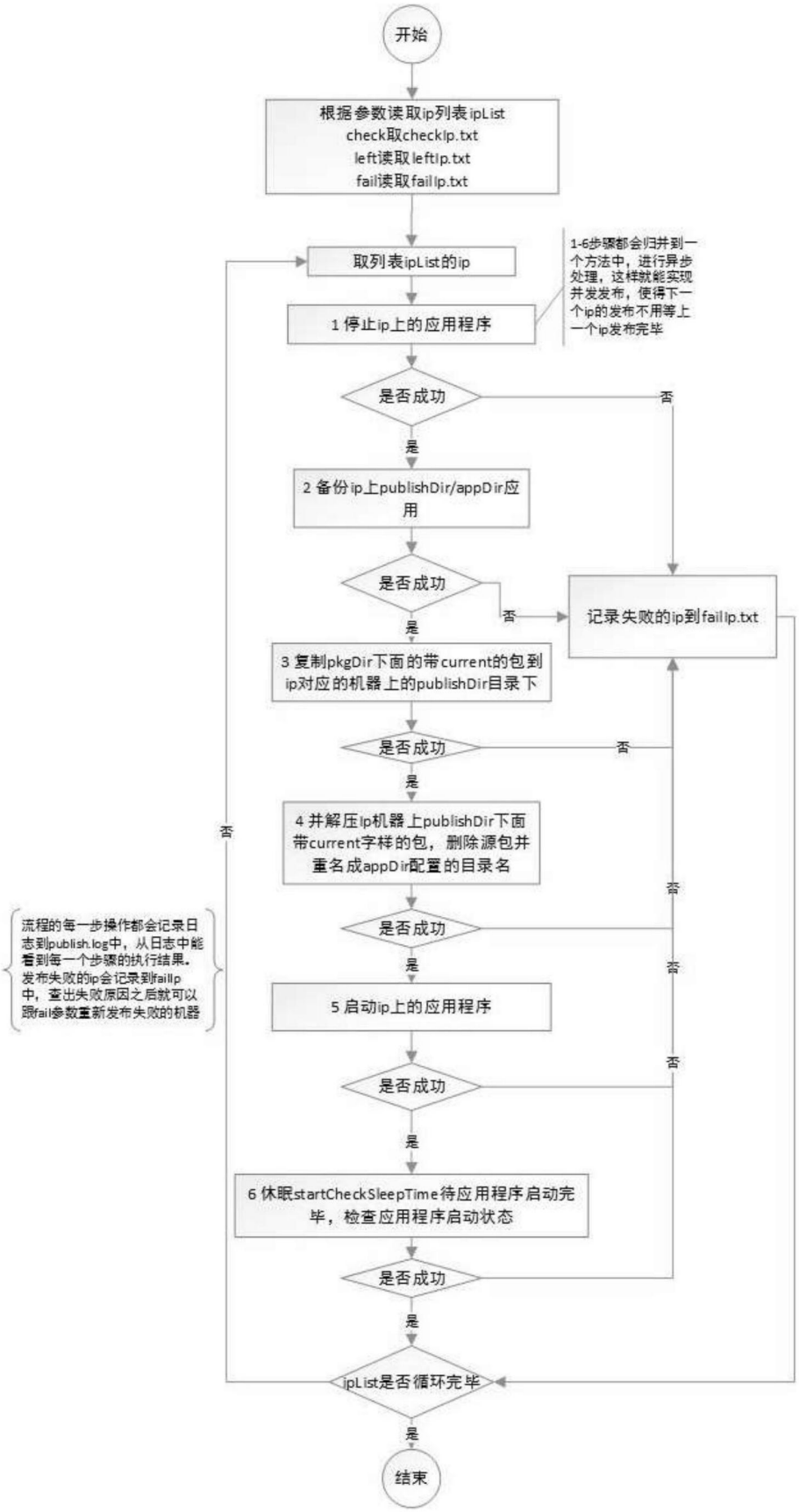

开始
根据参数读取ip列表ipList
check取checkIp.txt
left读取leftIp.txt
fail读取failIp.txt
取列表ipList的ip
1 停止ip上的应用程序
1-6步骤都会归并到一个方法中，进行异步处理，这样就能实现并发发布，使得下一个ip的发布不用等上一个ip发布完毕
是否成功
否
是
2 备份ip上publishDir/appDir应用
是否成功
否
记录失败的ip到failIp.txt
是
3 复制pkgDir下面的带current的包到ip对应的机器上的publishDir目录下
是否成功
否
是
4 并解压ip机器上publishDir下面带current字样的包，删除源包并重名成appDir配置的目录名
否
是否成功
否
是
5 启动ip上的应用程序
否
是否成功
否
是
6 休眠startCheckSleepTime待应用程序启动完毕，检查应用程序启动状态
是否成功
是
ipList是否循环完毕
否
是
结束
流程的每一步操作都会记录日志到publish.log中，从日志中能看到每一个步骤的执行结果。发布失败的ip会记录到failIp中，查出失败原因之后就可以跟fail参数重新发布失败的机器

图3

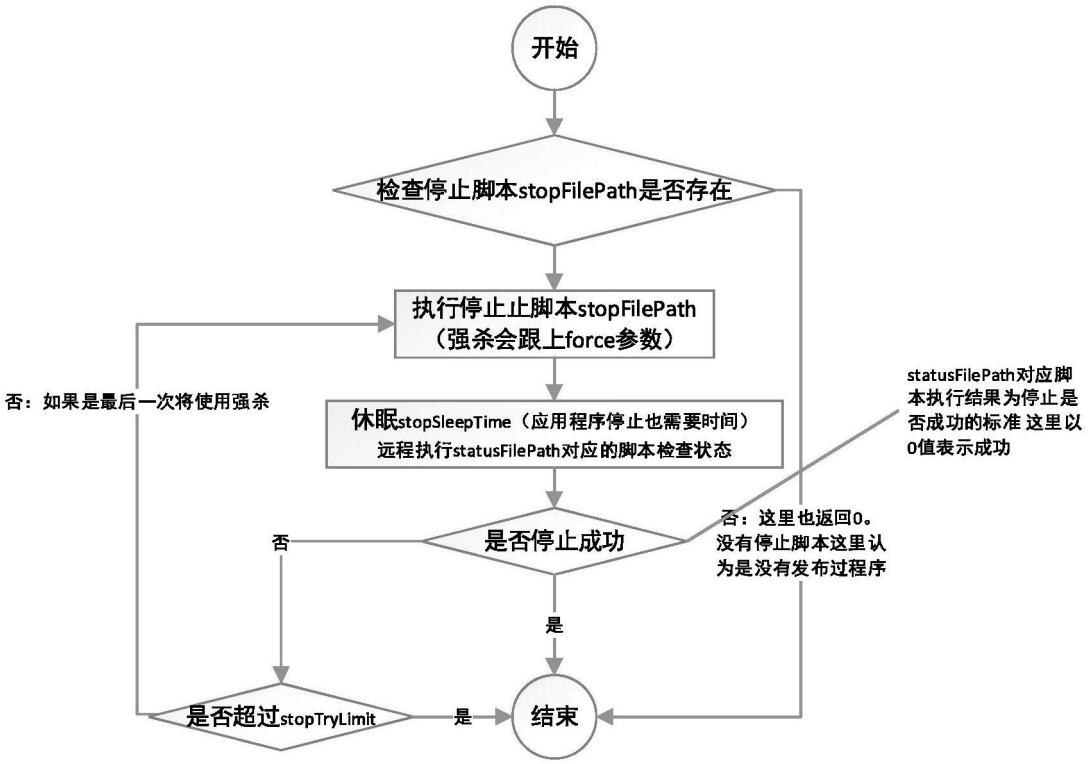

开始
检查停止脚本stopFilePath是否存在
执行停止止脚本stopFilePath
（强杀会跟上force参数）
休眠stopSleepTime（应用程序停止也需要时间）
远程执行statusFilePath对应的脚本检查状态
是否停止成功
否
是
否：如果是最后一次将使用强杀
是否超过stopTryLimit
是
结束
statusFilePath对应脚本执行结果为停止是否成功的标准 这里以0值表示成功
否：这里也返回0。没有停止脚本这里认为是没有发布过程序

图4

```
[root@fk-fqz01 ~]# tree caseSurvey -L 3
caseSurvey
├── appLog
│   └── root
│       └── root.log
├── bin
│   ├── common.sh
│   ├── start.sh
│   ├── status.sh
│   └── stop.sh
├── conf
│   ├── Catalina
│   │   └── localhost
│   ├── catalina.policy
│   ├── catalina.properties
│   ├── context.xml
│   ├── jaspic-providers.xml
│   ├── jaspic-providers.xsd
│   ├── logging.properties
│   ├── server.xml
│   ├── tomcat-users.xml
│   ├── tomcat-users.xsd
│   └── web.xml
├── logs
├── scriptLog
│   ├── scriptStart.log
│   ├── scriptStatus.log
│   └── scriptStop.log
├── temp
│   ├── hbase-root
│   │   └── local
│   └── safeToDelete.tmp
├── webapps
│   └── caseSurvey
│       ├── META-INF
│       ├── static
│       └── WEB-INF
└── work
    └── Catalina
        └── localhost

19 directories, 19 files
[root@fk-fqz01 ~]#
```

图5

```
[root@AFR-APP05 ~]# tree baiDuGps/
baiDuGps/
├── appLog
│   └── root.log
├── bin
│   ├── common.sh
│   ├── start.sh
│   ├── status.sh
│   └── stop.sh
├── conf
│   └── baiduGps.properties
├── lib
│   ├── baidu.gps-0.0.1.jar
│   ├── commons-codec-1.9.jar
│   ├── commons-lang3-3.6.jar
│   ├── commons-logging-1.2.jar
│   ├── commons-pool2-2.4.2.jar
│   ├── fastjson-1.1.19.jar
│   ├── httpclient-4.5.jar
│   ├── httpcore-4.4.1.jar
│   ├── jedis-2.9.0.jar
│   ├── log4j-1.2.17.jar
│   ├── log4j-1.2-api-2.7.jar
│   ├── log4j-api-2.7.jar
│   └── log4j-core-2.7.jar
└── scriptLog
    ├── java.log
    ├── jstack_12657_2018-10-24_19-35-57.log
    ├── jstack_20185_2018-06-01_16-48-17.log
    ├── start.log
    ├── status.log
    └── stop.log

5 directories, 25 files
[root@AFR-APP05 ~]#
```

图6