Slider Agent启动流程分析

Container的执行命令:

exec /bin/bash -c

"python ./infra/agent/slider-agent/agent/main.py

--label container\_1461811649094\_0010\_01\_000002\_\_\_KAFKA\_BROKER

--zk-quorum bditest3.cmss.com:2181,bditest2.cmss.com:2181,bditest1.cmss.com:2181

--zk-reg-path /registry/users/fys/services/org-apache-slider/kafka210

>/data11/hadoop/yarn/log/application\_1461811649094\_0010/container\_1461811649094\_0010\_01\_000002/slider-agent.out 2>&1 "

注：主要针对agent的参数配置生成

main.py#main

步骤1：初始化options

options.root\_folder <= AGENT\_WORK\_ROOT || 'PWD'

options.log\_folder <= AGENT\_LOG\_ROOT

options.label <= --label（见命令行）

options.zk-quorum <= --zk-quorum

options.zk-reg-path <= --zk-reg-path

步骤2：

agentConfig 初始化

options:包括AGENT\_WORK\_DIR,AGENT\_LOG\_DIR,label

config <= ConfigParser.RawConfigParser

（1）//默认值，AgentConfig.content

（2）//update\_config\_from\_file <= infra/conf/agent.ini //目前还没用到

生成类似于agent.conf文件的格式，，包括zk-quorum、zk-reg-path的设置

步骤3：从zk-reg-path中获取AM hostname和secured port

registry <= Registry(zk\_quorum,zk\_reg\_path)

amHost

amUnsecuredHost

amSecuredPort

//设置agentConfig中的值

hostname:amHost

secured\_port:amSecuredPort

port:amUnSecuredPort

//生成server\_url

https://hostname:port/check\_path

checkPath <= /ws/v1/slider/agents/

步骤4：

controller <=Controller(agentConfig)

controller.start



//运行架构图

步骤5：执行INSTALL命令

AgentProviderService通过Hearbeat发送INSTALL命令，对命令进行参数的赋值

addInstallCommand > response <=HeatBeatResponse

1) 设置基本参数

commandType

clusterName

roleCommand

serviceName

componentName

pkg

role

taskId

public\_hostname

appmaster\_hostname

commandId

2) 设置hostLevelParams

JAVA\_HOME

PACKAGE\_LIST

CONTAINER\_ID

3) 设置configurations

global

broker

server //buildCommandConfigurations

4) 设置componentConfiguration

slider-appmaster //buildComponentConfigurations

broker

步骤6：slider agent接收到SliderAppMaste命令后，

controller#heartbeatWithServer

addToQueue: executeCommand <= INSTALL

将命令加入到执行队列（ActionQueue）中

步骤7：

ActionQueue#execute，从队列中取出命令并执行，其中分为execute\_command和status\_command，以INSTALL命令为例：

#execute\_command：command

#CustomServiceOrchestrator.runCommand

1）初始化参数

script\_type

foreground\_start

task\_id

command\_name

script

timeout

python\_paths

script\_params:

[command\_name, //命令名称

json\_path, //command以json格式文件保存在本地

current\_base\_dir]

2)python\_executor执行脚本文件

PythonExecutor#run\_file

启动执行线程

subprocess.run: pythonCommand

pythonCommand:

script

script\_params

3)调用服务具体的脚本

以kakfa.py为例

调用形式：python -s kafka.py

[commandName]

[json\_path]

[current\_base\_dir]

[logger\_level]

[config.getWorkRootPath]

例如install

python -s kafka.py install command-11.json [env]

4)kafka.py 继承Script.py

Script.execute //解析command参数，并执行和command type相关的方法

执行流程如下：

(1)：

command\_name <= sys.argv[1] [str.lower]

command\_data\_file <= sys.arg[2] 命令执行脚本

basedir <= sys.arg[3] //当前目录

(2)：

*Script.config <= ConfigDictionary(json.load) command\_data\_file*

method <= choose\_method\_to\_execute: command\_name

(3)：执行method，例如install

(4)：script#install\_package

1) 参数初始化

name <= package[name]

type <= package[type]

tarball <= basedir+name

install\_location <= [global][app\_install\_dir]

2)将压缩包，解压至install\_location

步骤8：执行完install命令后，执行start

AgentProviderService#getNextCommand : Installed -> start

#addStartCommand //开始执行start命令

分析过程和INSTALL的执行过程相同，最终调用kafka.py#start

kafka.py start过程如下：

(1) 设置参数 parmas，根据command.json中获取

(2) environment.py#set\_params

(3) 设置kafkaLogConfig

*(4) 应用配置的初始化*

server\_conf <= {params.conf\_dir}/server.slider.properties

broker\_config <= {conf\_dir}/server.properties //并根据[configurations][server]去更新

PropertiesFile(server\_conf,broker\_config)

//resource\_managerment.libraries.providers.properties\_file.ProvidertiesFileProvider

先从server.properties中读取配置，然后根据command中的[configurations][server]中的内容完成替换

附录：

1) --label的生成

--label container\_1461811649094\_0010\_01\_000002\_\_\_KAFKA\_BROKER

label的生成过程

SliderAppMaster#onContainersAllocated

——> AppState#onContainersAllocated > assignment : role,container

——>RoleLaunchService#launchRole > assignment

——> AgentProviderService#buildContainerLaunchContext

——> label <= getContainerLable > container+LABEL\_MAKER+role

2) 命令脚本的生成,response的json //需要查看下

Controller#heartbeatWithServer

json.dumps(heartbeat.build(commandResult,xxx))

*CustomServiceOrchestrator.py#dump\_command\_to\_json*

3) PropertiesFileProvider，生成配置文件

filename

dir

config\_content <= InlineTemplate

properties\_dict : params.broker\_config

source.py#InlineTemplate，使用jinja2完成参数的替换

broker\_config中的参数来源，

broker\_config=

dict(line.strip().split('=') for line in open(format("{conf\_dir}/server.properties"))

if not (line.startswith('#') or re.match(r'^\s\*$', line)))

broker\_config.update(config['configurations']['server'])

先从server.properties中读取配置，然后根据command中的[configurations][server]中的内容完成替换

4）kafka broker.id的生成过程

如何保证其唯一性

根据上面的分析，broker.id是从command中的[configurations][server]中读取

原始的配置为：

appConfig.json

resources.json

其中metainfo-default中有server.xml

<configFiles>

<configFile>

<type>xml</type>

<fileName>server.xml</fileName>

<dictionaryName>server</dictionaryName>

</configFile>

</configFiles>

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>zookeeper.connect</name>

<value>${@//site/broker/zookeeper}/kafka/${@//site/broker/instance.name}</value>

</property>

<property>

<name>broker.id</name>

<value>${@//site/global/app\_container\_tag}</value>

</property>

<property>

<name>log.dirs</name>

<value>${@//site/global/app\_pid\_dir}</value>

</property>

</configuration>

在addInstallCommand中生成的command的配置包括

"configurations":{

"server": {

"log.dirs": "/data7/hadoop/yarn/local/usercache/fys/appcache/application\_1461811649094\_0010/container\_1461811649094\_0010\_01\_000003/app/run",

"broker.id": "2",

"zookeeper.connect": "bditest1.cmss.com,bditest2.cmss.com,bditest3.cmss.com/kafka/kafka210",

"port": "52301" //该端口由appmaster生成

}

}替换流程： //疑问，哪个阶段读取的server.xml，怎么替换的，其中metainfo-default中有

<configFiles>

<configFile>

<type>xml</type>

<fileName>server.xml</fileName>

<dictionaryName>server</dictionaryName>

</configFile>

</configFiles>

AgentProviderService#buildCommandConfigurations *//打日志分析其过程*

参数：

appConf

containerId

componentName

1）获取tokens: Map<String,String>

${NN\_URI}

${NN\_HOST}

${ZK\_HOST}

${DEFAULT\_ZK\_PATH}

${DEFAULT\_DATA\_DIR}

${JAVA\_HOME}

${COMPONENT\_HOME}

2）gettApplicationConfigurationType

*configFiles <= getMetaInfo().getApplication().getConfigFiles() //server.xml*

*configList.add <= GLOBAL\_CONFIG\_TAG*

*configList.add <= configFile.getDictionaryName //server*

"configurations":{

"global":{

},

"server":{

}

}

3) getSystemConfigurationsRequested

configTypes <= appConf.get(AgentKeys.SYSTEM\_CONFIG) //在kafka中是broker

经过2,3后

"configurations":{

"global":{},

"server":{}，

"broker":{}

}

4) addNamedConfiguration

参数：

configName <= global,server,broker

sourceConfig <= appConfig.json中global选项

global: addDefaultGlobalConfig

app\_log\_dir

app\_pid\_dir

app\_install\_dir

app\_input\_conf\_dir

app\_container\_id

*app\_container\_tag*  //ComponentTagProvider

pid\_file

app\_root

updateAllocatedPorts

addDefaultConfig //将kafka server.xml放入configurations(Digester)

AgentProviderService#buildMetaInfo

defaultConfig <= Map<String,DefaultConfig>

#initializeDefaultConfig

DefaultConfig <= AgentUtils.getDefaultConfigs

DefaultConfig <= DefaultConfigParser.parse(defaultConfigFile)

5) dereferencAllConfigs

将defaultConfig中的替换格式为"${@//site/%s/%s}"的配置项