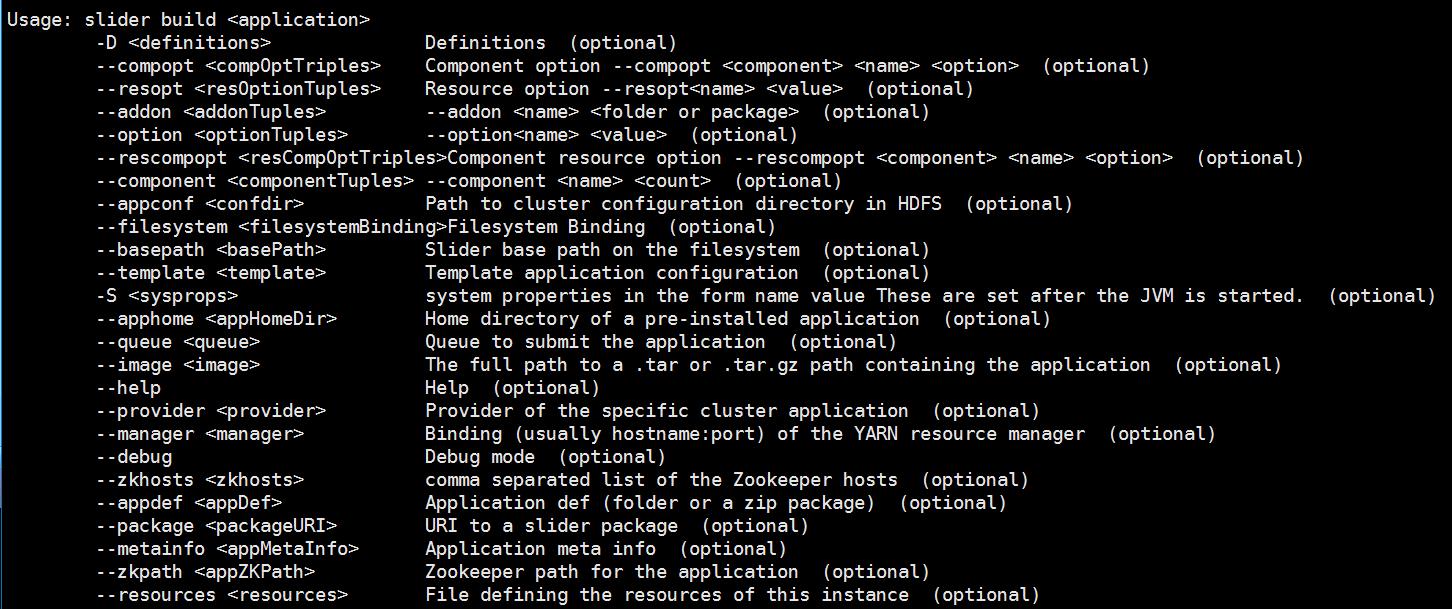
AppConfig和Resources解析过程源码分析

slider build过程分析，命令如下：



AbstractClusterBuildingActionArgs

对应参数，如命令行中所示

SliderClient#actionBuild

——> buildInstanceDefinition

——> lookupZKQuorum

registryQuorum <== hadoop.registry.zk.quorum

appConfDir <== --appConf ，hdfs目录

provider <== --provider，定义cluster app的provider //详细分析下provider

sliderAM :SliderAMClientProvider

provider: AbstractClientProvider =>AgentClientProvider

builder <== new InstanceBuilder

定义:

instanceDefinition: AggregateConf

appConf : ConfTreeOperations NULL

resources: ConfTreeOperations NULL

internal: ConfTreeOperations NULL //详细分析其的作用

sliderAM.prepareInstanceConfiguration

provider.prepareInstanceConfiguration

resources.json的初始化

ConfTreeOperations#mergeFile

resources.mergeFile <= resources.json

1）参数

buildInfo.resources ：resources文件，File

validator: ResourcesInputPropertiesValidator

2）过程分析

confTreeSerDeser: ConfTreeSerDeser

tree: ConfTree <= confTreeSerDeser.fromFile

//根据文件进行confTree的解析，json解析

ops <= new ConfTreeOperations(tree)

ops.validate(validator) //验证配置的是否正确

merge(ops.confTree)

3) mege(ops.confTree)

getGlobalOptions <= tree.global

metadata <= tree.metadata //分析其作用

credentials <= tree.credentials //分析其作用

components <= tree.components

ConfTreeOperations#components

#Map<String,Map<String,String>>

"HBASE\_MASTER": {

"yarn.role.priority": "1",

"yarn.component.instances": "1",

"yarn.placement.escalate.seconds": "10",

"yarn.memory": "1500"

}

<HBASE\_MASTER,

<yarn.role.priority,1>

<yarn.component.instances,1>

<yarn.placement.escalate.seconds,10>,

<yarn.memory,1500>

>

ConfTreeOperations#globalOptions

global <— Map<String,String>

#putAll <— that.global，包括appconfig及resources中的global配置项

ConfTreeOperations

#resolve

mergeInGlobal : component的配置与confTree.global merge

增加了CONF\_GROUP的处理逻辑

componentConf.containsKey(OptionKeys.CONF\_GROUP) ? true

componentConf <— comp.getValue

groupName <— componentConf.get

visited.add

if (confTree.components.containsKey(group)) {

Map<String, String> groupConf = confTree.components.get(group);

SliderUtils.mergeMapsIgnoreDuplicateKeys(componentConf, groupConf);

group = groupConf.get(OptionKeys.CONF\_GROUP);

} //将group对应的值赋给component

原格式：

{

"global": {

"global.property": "global.value"

},

"components": {

"A": {

"a.property": "a.value"

},

"B": {

"conf.group": "A",

"b.property": "b.value"

}

}

}

{

"global": {

"global.property": "global.value"

},

"components": {

"A": {

"global.property": "global.value"

"a.property": "a.value"

},

"B": {

"conf.group": "A",

"global.property": "global.value"

"a.property": "a.value"

"b.property": "b.value"

}

}

}