Slider Docker

# 1.简介

Slider支持Docker需要实现以下目标：

1）以docker image的形式定义应用，同时可以监控运行状态，获取exported configs和生命周期管理

2）用户可以在不进行任务改变的情况下使用已有的Docker images

3）Slider可以从Docker hubs中拉取Docker images

4）Slider可以从Docker containers中聚合日志 ，并将输入文件写入Docker

5）Slider可以发布包含多个Docker images的应用（在同一物理机上）

6）用户描述Docker image及配置等。使用Slider配置文件中的Docker命令启动、查询和停止Docker container

7）如果应用以安装模式运行，Slider可以将Security credentials放入Container中

但是相关的JIRA，不包括以下：

1）Application Docker containers和Ambari Client融合

2）以Secure mode的方式部署应用

3）其他OS，只能在Linux操作系统中执行

4）私有Docker hub中的Docker images

5）以Docker run -i命令启动的Application Docker containers需要和用户交互

6）Docker images需要多初始步骤

7）YARN仅能监控到Slider Agent使用的内存，对Docker images无法获取其内存使用情况

# 2.定义及命令

命令如下：

slider create [app-name] --template appConfig.json --metainfo metainfo.json --resources resources.json

下面是部署Dockerized应用的例子：

{

"schemaVersion": "2.1",

"application": {

"name": "NODEJS-REDIS",

"components": [

{

"name": "NODEJS",

"type": "docker",

"dockerContainers": [

{

"name": "nodejs",

"commandPath": "/usr/bin/docker",

"image": "rsahahw/centos-node-redis",

"ports": [{

"containerPort" : "8000"

}]

}

]

},

{

"name": "REDIS",

"type": "docker",

"dockerContainers": [

{

"name": "redis",

"commandPath": "/usr/bin/docker",

"image": "tutum/redis",

"ports": [{

"containerPort" : "6379",

"hostPort": "6379"

}]

}]

}

]

}

}

在Metainfo.json中定义了image如何运行，其字段和非Docker application被非全部不一样，仅其在dockerContainers描述中不同，包括以下部分：

1）name，container的名称，不影响发布的应用，但是在appConfig.json中标识覆盖的属性

2）image，docker image的全称

3）additionalParam，在启动Docker container中时将参数传递到命令中

4）commandPath，docker命令的路径

5）statusCommand，用于检查运行应用的健康状况，返回0时为healthy，非0为unhealthy。如果不定义，则slider执行docker top ${container\_ID}|grep来获取运行状况

6）port,containerPort和hostPort，container的port和hostPort字段相绑定，当docker run命令时通过-p hostPort:containerPort来转换

7）mount,containerMount和hostMount，在启动container时通过-v hostMount:container

Mount将主机的目录挂载到Container中

8）options，定义多个附加的docker run命令。在启动应用时，传到docker un中，如果没有定义，会使用-d

appConfig.json中配置会覆盖metainfo.json中的参数，包括commandPath,options,

statusCommand,inputFiles,mounts及ports

例如：

{

"schema": "http://example.org/specification/v2.0.0",

"metadata": {

},

"global": {

},

"components": {

"NODEJS": {

"nodejs.commandPath": "/user/local/bin/docker",

"nodejs.options":"-d -e REDIS\_HOST=${REDIS\_HOST} --net=host",

"nodejs.statusCommand":"docker inspect -f {{.State.Running}} ${CONTAINER\_ID} | grep true"

},

"REDIS": {

"redis.options":"-d -e REDIS\_PASS=\*\*None\*\*",

"redis.statusCommand":"docker top ${CONTAINER\_ID} | grep redis-server"

}

}

}

AppConfig.json中的参数覆盖metainfo.json中的值，这种方式可以提供一些运行时变量。在定义时变量名前面添加container name，例如：nodejs.commandPath，而不是直接使用command.path，通过这种方式区分metainfo.json的参数。

The config structure in appConfig.json(map) -> components(list) -> component(map) -> containers(list)-> container(map) have to match the one in metainfo.json.

resources.json中的定义如下：

{

"schema": "http://example.org/specification/v2.0.0",

"metadata": { },

"global": { },

"components": {

"slider-appmaster": { },

"REDIS": {

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

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

"yarn.memory": "512"

},

"NODEJS": {

"yarn.role.priority": "2",

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

"yarn.memory": "512"

}

}

}

该文件中用于定义从YARN中申请的containers数目。在每个YARN Containers中，Slider运行Docker run来下载在metainfo.json中定义的Docker images。例如：

/usr/bin/docker run -d borja/memcached

对于redis组件，运行的命令如下：

docker run -d -e REDIS\_PASS=\*\*None\*\* -p hostPort:containerPort -name ${CONTAINER\_ID} tutum/redis

对于nodejs组件，运行：

docker run -d -e REDIS\_HOST=${REDIS\_HOST} -name ${CONTAINER\_ID} --net=host rsahahw/centos-node-redis

# 3.使用举例

metainfo.json

{

"schemaVersion":"1.0",

"application":{

"name":"MEMCACHED",

"components":[

{

"name":"MEMCACHED",

"type":"docker",

"dockerContainers":[

{

"name":"memcached",

"commandPath:":"/usr/bin/docker",

"image":"memcached"

}

]

}

]

}

}

appConfig.json

{

"schema":"http://example.org/specification/v2.0.0",

"metadata":{},

"global":{},

"components":{

"MEMCACHED":{

"memcached.commandPath":"/usr/bin/docker"

}

}

}

resource.json

{

"schema":"http://example.org/specification/v2.0.0",

"metadata":{},

"global":{},

"components":{

"slider-appmaster":{},

"MEMCACHED":{

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

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

"yarn.memory":"512"

}

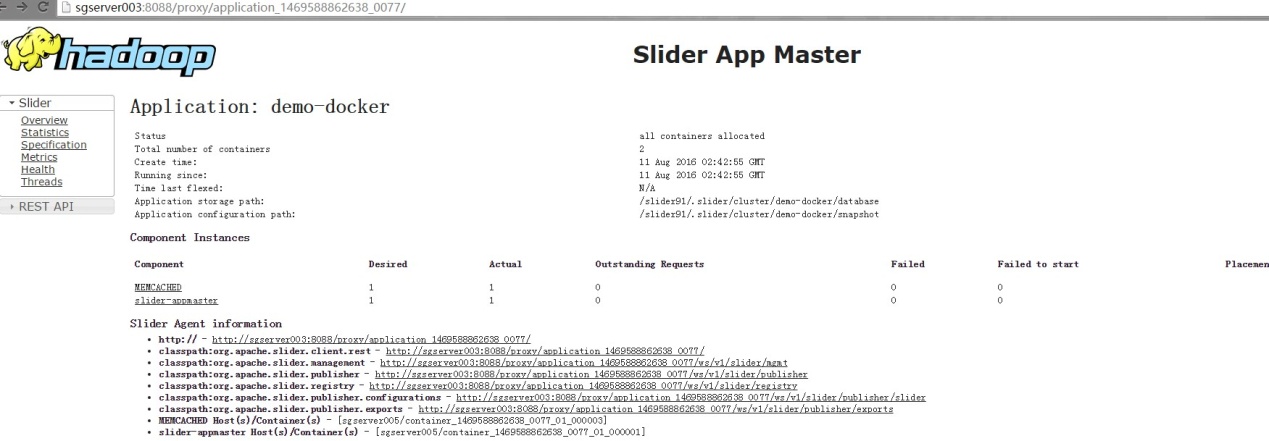
}

}

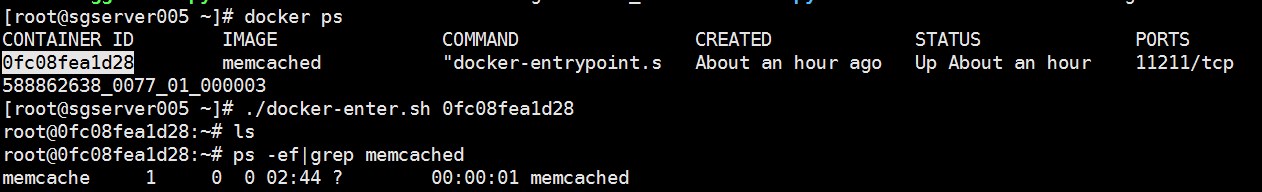
创建命令：

slider create demo-docker --template appConfig.json --resources resources.json --metainfo metainfo.json

结果如下：



在sgserver007上启动的docker container如下：



进程及日志分析

2031|nm

——|26857 container-executor

——|26859 /bin/bash -c python

——|26864(slider-agent)

1| Linux Dameon

——| 18819(/usr/bin/docker -d)

——|27405 (docker container)

stop(slider stop app)

AppMaster#

Triggering shutdown of the AM: stop command issued: exit code = 0, SUCCEEDED: stop command issued;

Releasing container. Log: http://sgserver001:19888/jobhistory/logs/sgserver005:45454/container\_1469588862638\_0077

impl.AMRMClientAsyncImpl - Interrupted while waiting for queue

Slider-Agent# stopCommand

main.py# signal received, exiting.

DcokerManager.py ## docker stop: ['/usr/bin/docker', 'stop', u'container\_1469588862638\_0077\_01\_000003']

# 附录：

参考文献：

https://slider.incubator.apache.org/docs/slider\_specs/application\_pkg\_docker.html

相关Jira及源码分析：

|  |  |  |
| --- | --- | --- |
| Jira号 | 解决问题 | 描述 |
| SLIDER-780 | Docker的部署，monitor running status，获取exported configs及维护生命周期 | 支持Docker的初始Jira |
| SLIDER-916 |  | Fix Docker相关的bug |
| SLIDER-1162 | 创建一个Docker providers，解决在Docker container中执行agent python的问题 | OPEN状态 |
| SLIDER-906  SLIDER-1109 | With first class YARN support来实现基于Docker的应用 | Slider 0.91 |
| SLIDER-921 | Slider stop命令不能kill掉docker containers | OPEN状态 |
| SLIDER-919 | Slider DockerMananger从repository来去images时，首先检查本地是否存在 | 不是问题 |
| SLIDER-1118 | YarnDockerManager使用python module而不是system module中获取ip和hostname | Patch Available(0.91) |