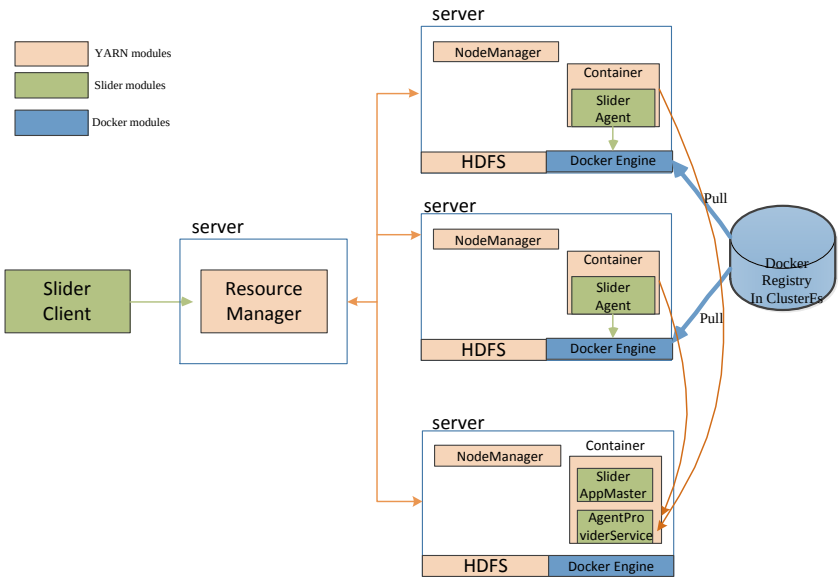
使用 Apache Slider 管理 Docker 集群

1.简介

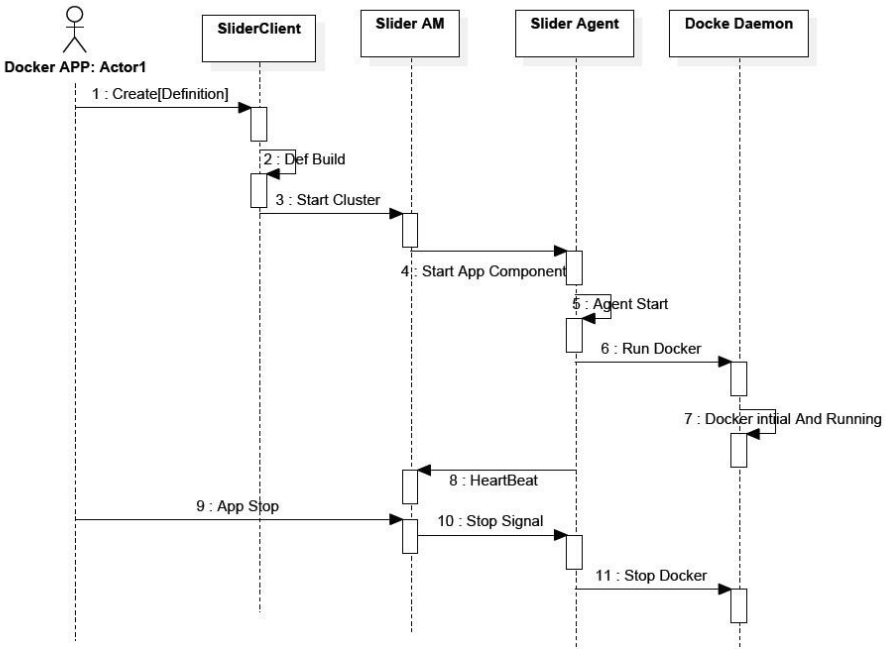
当前使用 Apache Slider 可以将 HBase、Storm 这些常驻服务运行在 YARN 上，但是这些常驻服务的启动与维护都有一定的难度；Docker 是一个开源的容器引擎，可以将应用及其依赖打包到一个可移植的容器中，然后发布到 Linux 之上。

在 YARN 中启动 Docker 有两种形式，使用 DockerContainerExecutor 将 YARN Containers 发布到 Docker Container 中，用户需要根据他们的需求定义 Docker images，将用户程序所依赖的软件环境打包到镜像中，但是在。但是使用 DockerContainerExecutor 时，需要 NM 的启动 Container 时以 root 来执行，另外一种方式是通过 YARN-3611 中的新特性，Support Docker Containers in LinuxContainerExecutor，使用 LinuxContainerExecutor 来启动 Docker，这两种 Docker 的执行方式不在这里介绍。

不管是通过哪种形式来执行 Docker 集群，都需要用户根据需求提前生成 image，然后实现应用程序的运行逻辑，这个过程工作量大而且过程比较复杂。Apache Slider 在 JIRA SLIDER-780 和 SLIDER-906 中实现了两个 docker 集群的执行方式，前一种的执行方式是通过 Slider Agent 中直接调用 docker 命令来启动 Docker Container，第二种是基于 YARN 的 DockerLinuxContainerRuntime(即 Support Docker Container In LinuxContainerExecutor)来实现 Docker 的启动和管理，其执行的总体框架基本相同（细节在以后会进行分析），如下图所示：



大致流程为通过 SliderClient 端启动程序的执行，在 YARN 中启动 SliderAppMaster，Slider AM 与 YARN 进行通信获取所需要的资源，在 Container 在启动 Slider Agent 后，Slider AM 向 Agent 发送执行命令，启动与管理 Docker，时序如下所示：



2. 运行实例

通过 Apache Slider 来运行和管理 Docker 集群，简化了开发者的工作，开发者仅需要在应用的定义文件中描述其执行逻辑即可，下面是一个例子，通过该例子介绍其定义及执行过程，Docker 集群的执行命令和其他应用程序的执行略有不同，如下：

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

在命令中要提供三个定义文件：

- metainfo.json，docker 集群的描述文件，定义了集群包含的 docker component 类型及默认使用的一些配置
- appConfig.json，docker 在运行过程中配置的文件，可以包括 mount,inputfile 及端口等信息，这些信息会覆盖 metainfo.json 中的属性
- resource.json，docker 集群启动的 docker 数目及资源等

实例 example，启动一个 memached server, 如下：

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.jso
```

启动后，通过 YARN AM Web 查看的结果如下：

找到具体的执行 host，查看 docker container 如下：

```
[root@sgserver005 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
0fc08fea1d28        memcached           "docker-entrypoint.s   About an hour ago   Up About an hour    11211/tcp
588862638_0077_01_000003
[root@sgserver005 ~]# ./docker-enter.sh 0fc08fea1d28
root@0fc08fea1d28:~# ls
root@0fc08fea1d28:~# ps -ef|grep memcached
memcache     1     0  0 02:44 ?        00:00:01 memcached
```

3. 定义描述

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

参考：

https://issues.apache.org/jira/browse/YARN-3611

https://issues.apache.org/jira/browse/YARN-3291

https://issues.apache.org/jira/browse/SLIDER-906

https://issues.apache.org/jira/browse/SLIDER-780