Support Docker Containers in LinuxContainerExecutor

**——YARN-3611**

# 概况，核心JIRA

在YARN中已LCE提供了资源本地化、基于Cgroups的资源管理（包括CPU，网络及磁盘等）、在container-executor utility中规范的特权用户操作来保证安全性。在LCE中支持docker containers的管理，可以避免用户通过修改配置来使用不同的ContainerExecutor。为了在LCE中支持Docker containers的执行，需要解决以下问题：

1. 客户端通过配置环境变量来请求docker功能，不影响client API
2. 安全：包括docker daemon和application
3. docker image的localization
4. 通过container-executor以特定用户来运行docker container
5. docker container的资源隔离，包括CPU/network/disk等
6. 与running container的交互，communicating with 或者signaling（确保正确的pid处理）
7. 某些特殊场景的处理，例如HDFS short-circuit reads

以上问题的解决不能改变当前LCE的所有操作。在YARN 3.0之前，为了启动不同类型的的Container，需要实现不同的ContainerExecutor，如下图所示：



为了支持Docker类型的应用的执行，需要实现DockerContainerExecutor（YARN-2466）。如果使用DockerContainerExecutor，需要修改参数<yarn.nodemanager.container-executor.class>，修改后会影响其他container type的job的执行。

在YARN-5388中，将会把DCE从YARN中删除，YARN 3.0以后不再有DCE。Docker应用通过LinuxContainerExecutor中的DockerRuntime来启动和维护，如下图所示：



在YARN 2.9（YARN-3611）后使用这种方式的docker 应用管理。

DockerContainerRuntime的基本框架实现涉及到的JIRA为YARN-3852和YARN-3853，Dokcer container的支持是通过DockerContainerRuntime来实现，DCR相对于LCE来说是low-level的接口，DCR启动Docker的操作是通过container-executor来实现。

应用的类型是通过环境变量来设置，以pi程序为例，执行命令如下：

$export YARN\_EXAMPLES\_JAR=./share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar

$bin/yarn jar $YARN\_EXAMPLES\_JAR pi -Dmapreduce.map.env="YARN\_CONTAINER\_RUNTIME\_TYPE=docker,YARN\_CONTAINER\_RUNTIME\_DOCKER\_IMAGE=ashahab/hadoop-trunk" -Dyarn.app.mapreduce.am.env="YARN\_CONTAINER\_RUNTIME\_TYPE=docker,YARN\_CONTAINER\_RUNTIME\_DOCKER\_IMAGE=ashahab/hadoop-trunk" -Dmapreduce.reduce.env="YARN\_CONTAINER\_RUNTIME\_TYPE=docker,YARN\_CONTAINER\_RUNTIME\_DOCKER\_IMAGE=ashahab/hadoop-trunk" 4 1000

在命令中通过设置下面变量来选择container的执行的runtime及使用的docker参数：

YARN\_CONTAINER\_RUNTIME\_TYPE

YARN\_CONTAINER\_RUNTIME\_DOCKER\_IMAGE

YARN\_CONTAINER\_RUNTIME\_DOCKER\_IMAGE\_FILE

YARN\_CONTAINER\_RUNTIME\_DOCKER\_RUN\_OVERRIDE\_DISABLE

YARN\_CONTAINER\_RUNTIME\_DOCKER\_CONTAINER\_NETWORK

YARN\_CONTAINER\_RUNTIME\_DOCKER\_RUN\_PRIVILEGED\_CONTAINER

YARN\_CONTAINER\_RUNTIME\_DOCKER\_LOCAL\_RESOURCE\_MOUNTS

在YARN-3853中，实现DockerContainerExecutorRuntime，同时改造LCE使之能够根据env中的配置，选择执行的CER(ContainerExecutorRuntime)，主要的改变如下所示:



NM使用DockerContainerExecutorRuntime启动Docker Container时，也是通过调用container-executor来实现，因此container-executor中需要增加调用docker daemon的方法（YARN-3852）。在container-executor.c中增加方法如下：

1. parse\_docker\_command\_file

从command file中解析docker命令，由DockerContainerRuntime生成

1. run\_docker，执行docker命令，参数为command\_file名

读取command\_file内容，解析出docker\_command

获取DOCKER\_BINARY\_KEY，docker的执行binary

1. launch\_docker\_container\_as\_user

参数包括:

user

app\_id

containter\_id

work\_dir

pid\_file

cred\_file

local\_dirs

log\_dirs

command\_file

resources\_key

resources\_value

在该方法中会初始化docker\_command,docker\_binary，可以执行的命令包括：

docker\_command\_with\_binary,

docker\_inspect\_command

docker\_wait\_command

docker\_rm\_command

YARN-3852和YARN-3853是支持docker的两个主要patch。为了优化docker的使用，有以下JIRA:

# 2. RESOLVED JIRA

**YARN-4267|YARN-4004**，Docker的执行可能遇到异常退出，返回的exit-code非0，这个时候需要输出docker log，使docker调试更容易，修改container-executor.c，增加docker\_log\_commands，当异常时，输出docker container的执行日志。

**YARN-4007**，设置用户可以配置Docker container的网络：



**YARN-4258**，Docker允许用户在启动时配置一个container的capabilities，该JIRA用于在docker containers启动时定义capabilites集合，如下所示：



**YARN-4262**，在一些场景下需要在特权containers中运行应用。因此需要：

1. 默认情况下disable 特权containers
2. enable时，配置允许启动这种containers的用户的whitelisted
3. whitelisted中的用户不一定启动特权containers，因此在使用时配置containers的模式



要符合三个条件：

1. container的启动环境变量中配置了：

YARN\_CONTAINER\_RUNTIME\_DOCKER\_RUN\_PRIVILEGED\_CONTAINER <= true

1. yarn-site中配置：

privileged-containers.allowed，允许container以特权方式启动

1. container的启动用户，submittingUser在privilegedContainesAcl中

privilegedContainersAcl由参数yarn-site中privileged-containers.acl来配置

**YARN-4553**，在YARN中启动docker不支持基于cgroup的资源隔离。由docker daemon启动的containers的进程不是container-executor进程的子进程。Docker支持—cgroup-parent标识，用于指定由nodemanager创建的基于container-specific cgroup目录。这样就可以使docker containers资源隔离，如下所示：



**YARN-4578**，将container相关的目录挂载到docker container中



**YARN-4595**，将主机上的文件或者目录挂载到docker container中是传递配置和其他信息的一种方式。该JIRA用于在ContainerLauchContext中允许用户在环境变量中挂在文件（只读）。考虑到用户和权限问题，仅挂在在distributed cache中的资源，如下所示：



**YARN-4759**，在DockerContainerRuntime中当前signal handling需要revisited docker containers。当NM重启需要根据用户进程的运行container中重新获取container。



**YARN-5042**，在containe中运行systemd需要访问/sys/fys/cgroup，需要将该模块mount成read only mount。



**YARN-5298**, Docker container中没有挂载usercache和NM filecache目录，如果container依赖public及application cope的资源会产生问题，将filecache和usercache挂载：



**YARN-5394**，LCE将/etc/passwd挂载到container中，但是使用了错误的文件名：

.addMountLocation("/etc/passwd", "/etc/password:ro")

该JIRA中，将该行代码删除掉了

**YARN-5443**，增加DockerInspectCommand

**YARN-5459**，增加DockerRmCommand

# 3. PATCH AVALIABLE

**YARN-5428(PATCH AVAILABLE)**，允许定义docker client配置目录，docker client允许配置包含docker client配置的目录，一般在docker client配置文件中会存储docker login证书,默认情况下该目录为Linux中的$HOME/.docker/config.json，同时该JIRA允许YARN在localizations时pull images（仿照marathon和kubernetes），如下所示：



**YARN-5366（PATH AVALIABLE）**，当docker containers结束或者failed后，container-executor会删掉该container。在DockerLinuxContainerRuntime中增加Job level的环境变量，运行用户可以选择保留container或者删除该container，JIRA如下：



**YARN-5669(PATCH AVAILABLE)**，增加DockerPullCommand

**YARN-5534（PATCH AVAILABLE）**，在YARN-4595中已经可以通过containerLaunchContext中将主机的文件和目录挂载到docker container中。但是有问题的挂载目录会引来安全问题。因此该JIRA通过配置目录的white-list，避免安全问题：



# 4. 待解决问题

**YARN-5360|YARN-4266**, Docker中通过—user机制使container的进程以该用户来执行，DockerCR中在启动docker containers时设置该参数，在non-secure模式下，该user通过yarn-site.xml中的参数: linux-container-executor.nonsecure-mode.local-user来配置。但是问题是pre-created的镜像中很多都没有该user，因此需要提供一种不配置—user的方式，该方式中需要考虑log-aggregation及artifact detection等。该JIAR和YARN-5360解决相同的问题。

LCE中的run\_as\_user通过下面两个参数：

yarn.nodemanager.linux-container-executor.nonsecure-mode.local-user

yarn.nodemanager.linux-container-executor.nonsecure-mode.limit-users

例如limit-users配置true，local-user配置yarn，那么LCE就会使用yarn来运行jobs。LCE会将launch\_containers.sh和/etc/passwd挂载到docker containers中，然后通过命令：

docker run –user=<run\_as\_user>

将/etc/passwd挂载后会覆盖掉docker image中的用户，通过设置UID，可以使用户在container中执行对应逻辑。但是这种方式对于解耦主机用户和container 用户是不够的。最终的方案是允许用户可以flexibly的执行docker，不依赖YARN。

YARN-4266的解决方案：

方式1：

修改whitelisted users所需要的目录权限（container-executor），包括以下目录：

container working dir，whitelisted users创建，所有的用户可读及可执行

container tokens dir，所有用户可读

application/container log目录，whitelisted users创建及写入

方式2：

在docker中进程启动前，修改docker containers中的UID，来达到操作目录的目的。 docker run命令以root用户启动docker container

获取主机run-as 用户的$UID

通过docker inspect [image]获取image中的$username

通过docker inspect获取image中的entrypoint

当container启动后，首先检查$username，如果是root，则执行lauch\_container.sh及entrypoint。否则：

在container中，通过usermod –u $UID $username修改用户的UID，通过su $username来执行lauch\_container.sh和$entrypoint

方式2是更合理的方式，patch如下所示：



但是两种方式都有缺点，因此该方式还是没有被接受

**YARN-3854，**在containter启动时，如果本地没有image，则会先下载image，方式有很多，例如从hdfs+docker load，docker pull等，同时也需要清除没用处的image。考虑使用哪种方式，需要解决下面的问题：

1. YARN如何根据启动的应用确认从什么地方拉取镜像
2. 在什么地方执行docker pull
3. 有哪些影响，例如清除images

解决方式：

1. 在LocalResourceType中增加DOCKERIMAGE，在URL中定义[registry/image:tag]
2. 在FSDownload.java中添加docker pull命令
3. 在DeletionService中添加docker iamge的删除

Docker pull操作设置timeout

输入docker credentials和store option

docker repository的安全性

最终选择使用docker pull，支持DockerPullCommand，扩展LocalResourceType，在FSDownlaod中使用docker pull命令。同时支持DockerRmiCommand，在Deletion Service中扩展删除无用镜像，如下所示：



目前该patch不完整，因此未被接受

**YARN-5670（OPEN）**，Add support for Docker image clean up，该JIRA未开始做

**YARN-5168（OPEN）**，当配置bridge模式：

增加-P参数，动态分配端口

增加-p，用户配置端口

为了支持bridge network，增加服务注册机制，应用可以通过服务发现机制获取端口信息

**YARN-5818**，Support the Docker Live restore feature

Docker 1.12以后引入了docker live restore功能，运行docker daemon重启或者升级时不重启docker containers。

https://issues.apache.org/jira/browse/YARN-5258 Docker的使用文档

http://www.slideshare.net/HadoopSummit/yarn-and-the-docker-container-runtime ，yarn&docker