iSulad 实验

实验介绍

关于本实验

本实验介绍如何在弹性云服务器上安装 iSulad 及其有关容器生命周期的基本操作。同时，尝试使用 iSula 容器镜像构建工具 isula-build 构建自己的容器镜像。

实验目的

学会安装 iSula 容器引擎 isulad；

学会如何建立、运行、停止以及删除一个容器；

学会用 isula-build 构建自己的容器镜像并运行之。

安装 isulad

安装

用 yum 命令安装

```
[root@openeuler ~]# yum install -y iSulad
```

1.1.1 启动并查看版本

用 systemctl start 命令启动

```
[root@openeuler ~]# systemctl start isulad
```

查看状态

```
[root@openeuler ~]# systemctl status isulad
● isulad.service - iSulad Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/isulad.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2020-09-14 15:53:43 CST; 4h 35min ago
 Main PID: 1248 (isulad)
    Tasks: 25
   Memory: 88.3M
   CGroup: /system.slice/isulad.service
           ├─1248 /usr/bin/isulad
           └─1315 isulad-img .....
```

按‘q’或‘Q’键退出信息显示。

查看版本

```
[root@openeuler ~]# isula version
Client:
   Version:          2.0.0
   Git commit:       5bf7c66ad4f7095156e87ca455da016f57c3e60f
   Built:            2020-03-23T21:35:55.821352180+00:00

Server:
   Version:          2.0.0
   Git commit:       5bf7c66ad4f7095156e87ca455da016f57c3e60f
   Built:            2020-03-23T21:35:55.821352180+00:00

OCI config:
   Version:          1.0.1
   Default file: /etc/default/isulad/config.json
```

查看帮助

```
[root@openeuler ~]# isula --help
```

1.2 容器与镜像管理

1.2.1 准备工作

安装 JSON 格式数据处理工具

```
[root@openeuler ~]# yum install –y jq
```

（虽然建议自己输入命令行，但如果拷贝粘贴此命令的话，注意参数-y 的“-”被正确粘贴。最好将命令拷贝至记事本再进行粘贴。）

修改 isulad 的配置文件

```
[root@openeuler ~]# mkdir iSula && cd iSula
[root@openeuler iSula]# cp /etc/isulad/daemon.json /etc/isulad/daemon.json.origin
[root@openeuler iSula]# vi /etc/isulad/daemon.json       #在此处修改
[root@openeuler iSula]# cat /etc/isulad/daemon.json
{
      "group": "isulad",
      "default-runtime": "lcr",
      "graph": "/var/lib/isulad",
      "state": "/var/run/isulad",
      "engine": "lcr",
      "log-level": "ERROR",
      "pidfile": "/var/run/isulad.pid",
```

```
    "log-opts": {
        "log-file-mode": "0600",
        "log-path": "/var/lib/isulad",
        "max-file": "1",
        "max-size": "30KB"
    },
    "log-driver": "stdout",
    "hook-spec": "/etc/default/isulad/hooks/default.json",
    "start-timeout": "2m",
    "storage-driver": "overlay2",
    "storage-opts": [
        "overlay2.override_kernel_check=true"
    ],
    "registry-mirrors": [
        "docker.io"
    ],
    "insecure-registries": [
    ],
    "pod-sandbox-image": "",
    "image-opt-timeout": "5m",
    "image-server-sock-addr": "unix:///var/run/isulad/isula_image.sock",
    "native.umask": "secure",
    "network-plugin": "",
    "cni-bin-dir": "",
    "cni-conf-dir": "",
    "image-layer-check": false,
    "use-decrypted-key": true,
    "insecure-skip-verify-enforce": false
}
```

在上述文件中，我们设"registry-mirrors"的值为"docker.io"。

检查配置文件合法性

```
[root@openeuler iSula]# cat /etc/isulad/daemon.json | jq
```

如果配置文件的内容能正确显示出来，即表示其格式合法。

重启 isulad 服务

```
[root@openeuler iSula]# systemctl restart isulad
```

1.2.2 运行容器 hello-world

用 isula run 命令直接运行

```
[root@openeuler iSula]# isula run hello-world
Unable to find image 'hello-world' locally
Image "hello-world" pulling
Image "a29f45ccde2ac0bde957b1277b1501f471960c8ca49f1588c6c885941640ae60" pulled

Hello from Docker!
```

This message shows that your installation appears to be working correctly.

由于这是第一次运行，所以会拉取 hello-world 的镜像，然后会运行它的一个实例，该实例会打印这样一个字符串：

Hello from Docker!

查看其镜像

```
[root@openeuler iSula]# isula images
REPOSITORY                    TAG         IMAGE ID            CREATED             SIZE
hello-world                   latest      a29f45ccde2a        2020-01-03 09:45:59 12.487 KB
```

以上输出表明这一阶段的安装成功了，下面继续验证其他的一些命令。

1.2.3 运行容器 busybox

创建容器并启动之

```
[root@openeuler iSula]# isula create -it busybox
Unable to find image 'busybox' locally
Image "busybox" pulling
Image "5a9d3c3dd268611214b65066c72d5fe73cf948f8b6195227de70119858bab258" pulled
bf76b1e5c1c624faaa8a1f81887607ca002ebe3939047508738988134dcd00ec

[root@openeuler iSula]# isula start bf76b1e5c1c6
```

直接运行

```
[root@openeuler iSula]# isula run -itd busybox
6c1d81467d3367a90dd6e388a16c80411d4ba76316d86b6f56463699306e1394
```

可以看出是以一个新的实例运行的。

交互式运行

```
[root@openeuler iSula]# isula run -it busybox
/ # ls
bin     dev     etc     home    proc    root    sys     tmp     usr     var
/ # uname -a
Linux localhost 4.19.90-2003.4.0.0036.oe1.aarch64 #1 SMP Mon Mar 23 19:06:43 UTC 2020 aarch64 GNU/Linux
/ # ifconfig
lo              Link encap:Local Loopback
                inet addr:127.0.0.1     Mask:255.0.0.0
                inet6 addr: ::1/128 Scope:Host
                UP LOOPBACK RUNNING     MTU:65536     Metric:1
                RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                collisions:0 txqueuelen:1000
                RX bytes:0 (0.0 B)     TX bytes:0 (0.0 B)
```

```
/ # busybox –help       #这是个错误命令，但是可以看到 busybox 的版本
BusyBox v1.32.0 (2020-07-27 19:11:43 UTC) multi-call binary.
……
/ # exit
```

以上略浅色的字体表示是在容器中运行的。

暂停/恢复一个容器

```
[root@openeuler iSula]# isula pause bf76b1e5c1c6
bf76b1e5c1c6
[root@openeuler iSula]# isula unpause bf76b1e5c1c6
bf76b1e5c1c6
```

先停止，再删除一个容器

```
[root@openeuler iSula]# isula stop bf76b1e5c1c6
bf76b1e5c1c6
[root@openeuler iSula]# isula rm bf76b1e5c1c6
bf76b1e5c1c624faaa8a1f81887607ca002ebe3939047508738988134dcd00ec
```

查看正在运行着的容器

```
[root@openeuler iSula]# isula ps
CONTAINER ID        IMAGE       COMMAND CREATED                 STATUS                  PORTS
NAMES
6c1d81467d33        busybox "sh"        37 minutes ago     Up 37 minutes
6c1d81467d3367a90dd6e388a16c80411d4ba76316d86b6f56463699306e1394
```

直接删除

```
[root@openeuler iSula]# isula rm -f 6c1d81467d33
6c1d81467d3367a90dd6e388a16c80411d4ba76316d86b6f56463699306e1394
```

查看镜像关联到容器没有

```
[root@openeuler iSula]# isula ps -a
CONTAINER ID        IMAGE       COMMAND CREATED                 STATUS
PORTS       NAMES
c17c95817f73        busybox "sh"        54 minutes ago              Exited (130) 52 minutes ago
c17c95817f7375b4b1bcafdff5c51442b96185ef598335a53a3f3f9ec8f4f2f6
```

先将关联到镜像的容器销毁

```
[root@openeuler iSula]# isula rm -f c17c95817f7375
c17c95817f7375b4b1bcafdff5c51442b96185ef598335a53a3f3f9ec8f4f2f6
```

然后删除镜像

```
[root@openeuler iSula]# isula rmi busybox
Image "busybox" removed
```

如果这一步骤失败请先将所有容器删除再运行此命令。

1.3 使用 isula-build 构建容器镜像

目前为止，我们使用的容器镜像都是从 docker.io 下载的已经构建好的容器镜像，下面我们尝试使用 iSula 提供的容器镜像构建工具 isula-build，构建自己的容器镜像并运行。

1.3.1 安装 isula-build

检查 yum 源

```
[root@openeuler iSula]# tail /etc/yum.repos.d/openEuler_aarch64.repo
[update]
name=update
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS/update/$basearch/
enabled=0
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS/OS/$basearch/RPM-GPG-KEY-openEuler
```

将文件中[update]节的 enable=0 改为 enable=1 并保存。

```
[root@openeuler ~]# yum repolist
repo id
repo name
EPOL
EPOL
OS
OS
debuginfo
debuginfo
everything
everything
source
source
update
```

可以从输出中看到 update 的字样。

用 yum 命令安装 isula-build

```
[root@openeuler iSula]# yum install -y isula-build
```

安装 docker-runc

```
[root@openeuler iSula]# yum install -y docker-runc
```

用 systemctl start 命令启动

```
[root@openeuler iSula]# systemctl start isula-build
```

查看版本

```
[root@openeuler iSula]# isula-build version
Client:
```

```
    Version:            0.9.2
    Go Version:         go1.13.3
    Git Commit:         a96cf18
    Built:              Thu Aug 20 02:16:07 2020
    OS/Arch:            linux/arm64

Server:
    Version:            0.9.2
    Go Version:         go1.13.3
    Git Commit:         a96cf18
    Built:              Thu Aug 20 02:16:07 2020
    OS/Arch:            linux/arm64
```

修改配置

修改/etc/isula-build/registries.toml，将 docker.io 加入到 isula-build 可搜索的镜像仓库列表里：

```
vi /etc/isula-build/registries.toml
……
[registries.search]
registries = ["docker.io"]
……
```

当然，如果有不同的镜像仓库也可以配置不同的镜像仓库。

重启 isula-build 服务：

```
[root@openeuler iSula]# systemctl restart isula-build
```

查看配置：

```
[root@openeuler iSula]# isula-build info -H
General:
    MemTotal:          3.12 GB
    MemFree:           2.07 GB
    SwapTotal:         0 B
    SwapFree:          0 B
    OCI Runtime:       runc
    DataRoot:          /var/lib/isula-build/
    RunRoot:           /var/run/isula-build/
    Builders:          0
    Goroutines:        11
Store:
    Storage Driver:          overlay
    Backing Filesystem: extfs
Registry:
    Search Registries:
        docker.io
    Insecure Registries:
```

在 Search Registries 配置项里看到”docker.io”赫然纸上。

至此，isula-build 安装完成。

1.3.2 构建容器镜像并导入到 isulad

创建 Dockerfile

创建 “Dockerfile”的文件：

```
[root@openeuler iSula]# mkdir –p /home/test/ && cd /home/test/
[root@openeuler test]# vi Dockerfile       #在此编辑文件内容
[root@openeuler iSula]# cat Dockerfile
FROM busybox
COPY hello.sh /usr/bin/
CMD ["sh", "-c", "/usr/bin/hello.sh"]
```

这里我们是在在/home/test 目录下创建 Dockerfile 文件的。

编辑 Dockerfile 中出现的 hello.sh 脚本，它将被加到原有的 busybox 镜像中以构建出我们自己的镜像：

```
[root@openeuler test]# vi hello.sh       #在此编辑文件
[root@openeuler test]# cat hello.sh
#!/bin/sh
echo "hello, isula-build!"
```

修改文件属性：

```
[root@openeuler test]# chmod +x hello.sh
```

验证：

```
[root@openeuler test]# ls -l
total 8
-rw------- 1 root root 76 Aug 21 12:31 Dockerfile
-rwx------ 1 root root 38 Aug 21 12:32 hello.sh
```

以上即是我们刚刚创建的 2 个文件。

构建容器镜像

用 isula-build 构建我们自己的容器镜像并导入到 isulad，镜像命名为 hello-isula-build:v0.1：

```
[root@openeuler test]# isula-build ctr-img build -f ./Dockerfile -o isulad:hello-isula-build:v0.1
STEP    1: FROM busybox
STEP    2: COPY hello.sh /usr/bin/
STEP    3: CMD ["sh", "-c", "/usr/bin/hello.sh"]
…
Build success with image id:
b195dbe534918dbb67694ab9ee9e15a02a7ae16018c93e212debd71072b4086b
```

查询构建出来的容器镜像

```
[root@openeuler test]# isula-build ctr-img images
--------------------------  ----------  ----------------  ------------------------  ------------
REPOSITORY                  TAG         IMAGE ID          CREATED                   SIZE
--------------------------  ----------  ----------------  ------------------------  ------------
hello-isula-build           v0.1        b195dbe53491      2020-08-11 08:01:27
1.45 MB
docker.io/library/busybox   latest      018c9d7b792b      2020-07-28 00:19:37
1.45 MB
------------------------------  ----------  ----------------  ------------------------  ------------
[root@openeuler test]# isula images
REPOSITORY                  TAG         IMAGE ID          CREATED                   SIZE
hello-isula-build           v0.1        b195dbe53491      2020-08-11 16:01:27
1.380 MB
```

可以看到，isula-build ctr-img images 可以查询到构建出来的容器镜像，同时，isula images 命令也可以查询到从 isula-build 导入到 isulad 的容器镜像

启动容器

我们使用自己构建出来的容器镜像来启动容器，按照预期，容器进程会打印出“hello, isula-build!”：

```
[root@openeuler test]# isula run hello-isula-build:v0.1
hello, isula-build!
```

1.3.3 扩展实验：isula-build 的其他镜像导出方式

在前述步骤中构建了自己的容器镜像 hello-isula-build:v0.1，并将容器镜像导出到 isulad 启动容器。isula-build 除了能将容器镜像导出到 isulad 之外，还可以：

- 导出到远端仓库
- 导出到 docker daemon
- 导出到本地压缩包
- 导出到 isula-build 的本地存储

可以通过 isula-build 的命令行帮助，查看这些导出方式：

```
[root@openeuler test]# isula-build ctr-img build --help
Build container images
Usage:
isula-build ctr-img build [FLAGS] PATH
Examples:
isula-build ctr-img build -f Dockerfile .
isula-build ctr-img build -f Dockerfile -o docker-archive:name.tar:image:tag .
isula-build ctr-img build -f Dockerfile -o docker-daemon:image:tag .
isula-build ctr-img build -f Dockerfile -o docker://registry.example.com/repository:tag .
isula-build ctr-img build -f Dockerfile -o isulad:image:tag .
isula-build ctr-img build -f Dockerfile --build-static='build-time=2020-06-30 15:05:05' .
```

可以按照命令行帮助信息中给出的详细示例进行练习。