Docker环境搭建

一. 版本及系统要求

1. Docker版本说明

2017/3/3，Docker官方发表了一篇博客，Docker版本从1.13.*直接跳入17.03，该版本的意思是17年3月。同时，还声明了Docker以后会以CE（Community Edition）和EE（Enterprise Edition）的形式发布。其中，CE版本每个月发布一次，也就是说，随后的版本就是17.03、17.04、17.05等，而EE的版本是没三个月发布一次，对应的就是17.03、17.06等。对于发布的每个EE版本，Docker官网都会提供一年的技术支持。

CE和EE在各发行版的支持情况

Platform	**Docker EE**	**Docker CE**
Ubuntu	yes	yes
Debian		yes
Red Hat Enterprise Linux yes	yes	
CentOS	yes	yes
Fedora		yes
Oracle Linux	yes	
SUSE Linux Enterprise Server	yes	
Microsoft Windows Server 2016	yes	
Microsoft Windows 10		yes
macOS		yes
Microsoft Azure	yes	yes
Amazon Web Services	yes	yes

2. 对linux内核版本的要求

docker对linux内核版本有要求，内核版本不能太低， 如果太低会导致docker的一些功能不能使用， docker官方文档要求至少3.8以上，建议3.10以上。

考虑到通用性，计划采用CentOS版本作为宿主机，目前Centos 7 使用的内核版本为3.10。

> 注：中兴新支点的CGSL来源于开源社区，由开源社区的源代码构建，在提供CGSL版本的时候，参考了CentOS (对应于RHEL的开源版本)所用的内核版本号和相关软件包， 并根据实际需要选取了他们所用的补丁集(毕竟红帽所用的方案是业界公认比较出色的)。因此，CGSL与RHEL是高度兼容的。
>
> CGSL V3 在今年生命周期已到最后阶段，后续不再发货，其内核版本为 2.6.18-164；
> CGSL V4 内核版本为2.6.32，V4系列版本现已进入成熟期，稳定商用多年，建议在没特殊要求的情

况下使用此版本。
CGSL V5 内核版本为3.10.0 ，此版本完美适配虚拟化，如环境需要用于虚拟化的场景，建议使用此版本。

3. CentOS的安装

版本：CentOS-7-x86_64-DVD-1511.iso

内核：3.10.0-327.el7.x86_64

安装过程略

4. 构建本地CentOS的YUM源

基于光盘ISO文件进行挂载

```
mount -t iso9660 /home/hello/CentOS-7-x86_64-DVD-1511.iso /mnt/cdrom/
```

在/etc/yum.repos.d/下新建repo文件，内容如下：

```
[local_server]
name=This is a local repo
baseurl=file:///mnt/cdrom
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

然后执行

```
yum clean
yum makecache
```

5. Docker的安装

受限公司的网络，采用官方文档中的如下方式进行离线安装：

Install from a package

If you cannot use Docker's repository to install Docker, you can download the .rpm file for your release and install it manually. You will need to download a new file each time you want to upgrade Docker.

1. Go to https://download.docker.com/linux/centos/7/x86_64/stable/Packages/ and download the .rpm file for the Docker version you want to install.

2. Install Docker CE, changing the path below to the path where you downloaded the Docker package.

```
$ sudo yum install /path/to/package.rpm
```

6. Docker官方镜像加速

6.1 修改镜像源

通过 Docker 官方镜像加速，中国区用户能够快速访问最流行的 Docker 镜像。该镜像托管于中国大陆，本地用户现在将会享受到更快的下载速度和更强的稳定性，从而能够更敏捷地开发和交付 Docker 化应用。

Docker 中国官方镜像加速可通过 registry.docker-cn.com 访问。该镜像库只包含流行的公有镜像。私有镜像仍需要从美国镜像库中拉取。

修改 /etc/docker/daemon.json 文件并添加上 registry-mirrors 键值，保存后重启docker以生效

```
{
"registry-mirrors": ["https://registry.docker-cn.com"]
}
```

6.2 内网代理设置

First, create a systemd drop-in directory for the docker service:

```
mkdir /etc/systemd/system/docker.service.d
```

Now create a file called /etc/systemd/system/docker.service.d/http-proxy.conf that adds the HTTP_PROXY environment variable:

```
[Service]
Environment="HTTP_PROXY=http://proxy.example.com:80/"
```

If you have internal Docker registries that you need to contact without proxying you can specify them via the NO_PROXY environment variable:

```
Environment="HTTP_PROXY=http://proxy.example.com:80/"
Environment="NO_PROXY=localhost,127.0.0.0/8,docker-
registry.somecorporation.com"
```

Flush changes:

```
$ sudo systemctl daemon-reload
```

Verify that the configuration has been loaded:

```
$ sudo systemctl show docker --property Environment
Environment=HTTP_PROXY=http://proxy.example.com:80/
```

Restart Docker:

```
$ sudo systemctl restart docker
```

7. 验证环境

7.1 从Docker仓库获取一个镜像

```
[root@localhost hello]# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
b04784fba78d: Pull complete
Digest:
sha256:f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f
Status: Downloaded newer image for hello-world:latest
```

7.2 查看本地镜像

```
[root@localhost hello]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
hello-world         latest              1815c82652c0        7 weeks ago
1.84kB
```

1. REPOSITORY：该字段包括两部分，第一部分是仓库名字（docker.io），第二部分是镜像包的名字。
2. TAG：用来表征镜像的版本号，默认情况下拖取最新版本（latest），也可以指定镜像的具体版本。
3. IMAGE_ID：一个哈希值，用来唯一标志一个镜像文件。
4. CREATED：镜像创建的时间。
5. VIRTUAL SIZE：镜像的大小。

7.3 利用镜像包启动一个容器

```
[root@localhost hello]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
```

8. 小结

目前公司访问外网，特别是在linux上直接访问外网，局限性太大，可以考虑申请一个可以直接上网的网口。

考虑搭建内网Docker私有仓库，后续单独开一章节来进行说明。