docker实战

初始化一个NodeJs程序

以下操作必须已经安装了NodeJS。

首先创建一个空文件夹。并创建以下文件：

- server.js
- package.json
- Dockerfile
- .dockerignore

```
mkdir docker_demo
cd docker_demo
touch server.js
touch package.json
touch Dockerfile
touch .dockerignore
```

然后在server.js写入

```
const Koa = require('koa');
const app = new Koa();

app.use(async ctx => {
   ctx.body = 'Hello docker';
});

app.listen(3000);
```

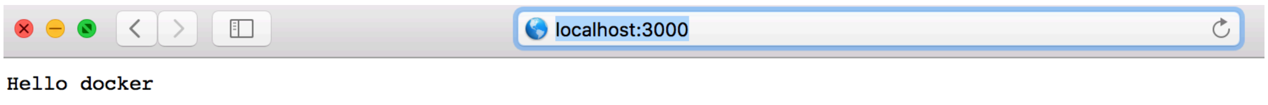
在package.json中写入

```json
{
    "name": "docker_demo",
    "version": "0.1.0",
    "private": true,
    "scripts": {
       "start": "node server.js"
    },
    "dependencies": {
        "koa": "^2.5.0"
    }
}
```

测试程序。控制台输入

```
npm start
```

浏览器打开本地测试,如果如图所示。表示demo创建成功。请继续往下。



```
Hello docker
```

创建dockerfile文件

Dockerfile是由一系列命令和参数构成的脚本，一个Dockerfile里面包含了构建整个image的完整命令。Docker通过docker build执行Dockerfile中的一系列命令自动构建image. 在.dockerignore文件里面写入代码。表示过滤该类型的文件。类似git的.gitignore

```
# Logs
logs
*.log
npm-debug.log*

# Runtime data
pids
*.pid
*.seed

# Directory for instrumented libs generated by jscoverage/JSCover
lib-cov

# Coverage directory used by tools like istanbul
```

```
coverage

# nyc test coverage
.nyc_output

# Grunt intermediate storage (http://gruntjs.com/creating-plugins#storing-task-files)
.grunt

# node-waf configuration
.lock-wscript

# Compiled binary addons (http://nodejs.org/api/addons.html)
build/Release

# Dependency directories
node_modules
jspm_packages

# Optional npm cache directory
.npm

# Optional REPL history
.node_repl_history
.idea
.node_modules
node_modules
.vscode
```

在Dockerfile文件中写入以下代码：

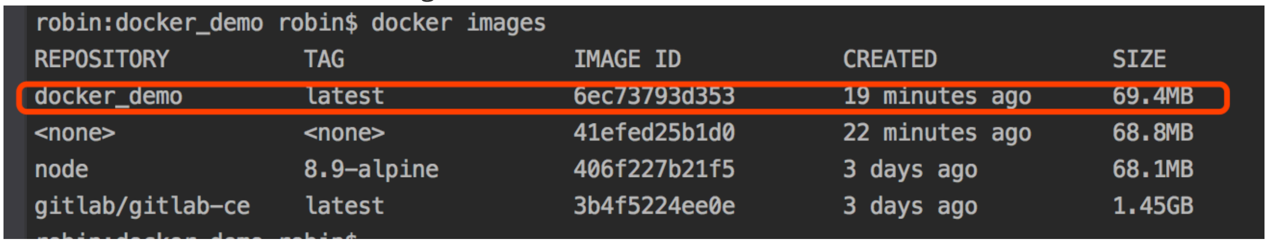
```
#制定node镜像的版本
FROM node:8.9-alpine
#声明作者
MAINTAINER evilboy
#移动当前目录下面的文件到app目录下
ADD . /app/
#进入到app目录下面，类似cd
WORKDIR /app
#安装依赖
RUN npm install
#对外暴露的端口
EXPOSE 3000
#程序启动脚本
CMD ["npm", "start"]
```

构建镜像

使用build命令构造镜像,注意后面那个“.”不能少。

```
[root@Sandbox-N ~]# docker build -t docker_demo .
Sending build context to Docker daemon  39.94kB
Step 1/7 : FROM node:8.9-alpine
---> 406f227b21f5
Step 2/7 : MAINTAINER robin
---> Using cache
---> 78d6cdbcfee2
Step 3/7 : ADD . /app/
---> 2cb30678612d
Step 4/7 : WORKDIR /app
Removing intermediate container e51377081039
---> c2b7d0f37d2d
Step 5/7 : RUN npm install
---> Running in da0c3946ca8d
npm notice created a lockfile as package-lock.json. You should commit
this file.
added 38 packages in 3.323s
Removing intermediate container da0c3946ca8d
---> eecee87f10e2
Step 6/7 : EXPOSE 3000
---> Running in f3973cc168a4
Removing intermediate container f3973cc168a4
---> 2671a4c6deb4
Step 7/7 : CMD ["npm", "start"]
---> Running in dec529f754aa
Removing intermediate container dec529f754aa
---> 6ec73793d353
Successfully built 6ec73793d353
Successfully tagged docker_demo:latest
```

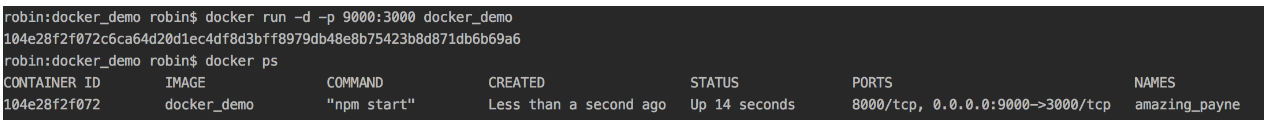
等待镜像构造完成。然后使用 images命令查看镜像。



```
robin:docker_demo robin$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker_demo         latest              6ec73793d353        19 minutes ago      69.4MB
<none>              <none>              41efed25b1d0        22 minutes ago      68.8MB
node                8.9-alpine          406f227b21f5        3 days ago          68.1MB
gitlab/gitlab-ce    latest              3b4f5224ee0e        3 days ago          1.45GB
```

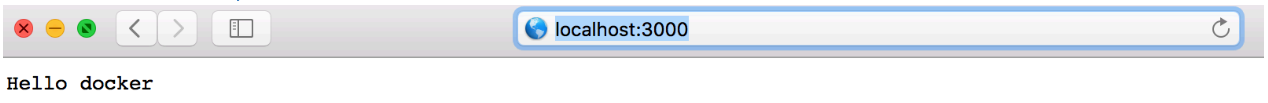
此时可以看到images已经构造完成。现在开始启动images，并测试。

```
#启动镜像 -d表示后台执行，-p 9000:3000表示指定本地的9000端口隐射到容器内的3000端口，
docker_demo为镜像名称
docker run -d -p 9000:3000 docker_demo
#查看容器
docker ps
```



```
robin:docker_demo robin$ docker run -d -p 9000:3000 docker_demo
104e28f2f072c6ca64d20d1ec4df8d3bff8979db48e8b75423b8d871db6b69a6
robin:docker_demo robin$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED                  STATUS              PORTS                              NAMES
104e28f2f072        docker_demo         "npm start"         Less than a second ago   Up 14 seconds       8000/tcp, 0.0.0.0:9000->3000/tcp   amazing_payne
```

此时浏览器打开http://localhost:9000/,如果如图所示。表示容器运行正常。



Hello docker

如果此时本地无法打开。可以使用log命令查看日志。根据日志修改对应出现的对方。