

Docker 应用部署-Nacos

参考链接

- 英文官方地址: <https://nacos.io/en-us/>
- 中文官方地址: <https://nacos.io/zh-cn/>
- 安装参考: <https://nacos.io/zh-cn/docs/quick-start-docker.html>

1 准备目录

在home目录下面创建目录

```
1 mkdir -p /home/nacos/nacos-docker
2 cd /home/nacos/nacos-docker
```

2 放行防火墙端口

注意: 如果是云服务器, 请到云服务器控制台开放端口, 不需要使用命令行来开启端口。

```
1 firewall-cmd --add-port 8848/tcp --add-port 9848/tcp --permanent
2 firewall-cmd --reload
```

3 启动单机模式

服务编排配置参考:

<https://github.com/nacos-group/nacos-docker/blob/master/example/standalone-derby.yaml>

在nacos-docker目录下新建standalone-derby.yaml文件, 然后写下面的内容。

TIP: 如果后期需要安装其它版本, 修改镜像版本号即可

```
1 version: "2"
2 services:
3   nacos:
4     image: nacos/nacos-server:v2.1.0
5     container_name: nacos-standalone
6     environment:
7       - PREFER_HOST_MODE=hostname
8       - MODE=standalone
9       - JVM_XMS=256m
10      - JVM_XMX=256m
11      - JVM_XMN=128m
12      - JVM_MS=32m
13      - JVM_MMS=80m
14     volumes:
15       - ./standalone-logs:/home/nacos/logs
16       - ./init.d/custom.properties:/home/nacos/init.d/custom.properties
17     ports:
18       - "8848:8848"
```

前台启动服务

```
1 docker-compose -f standalone-derby.yaml up
```

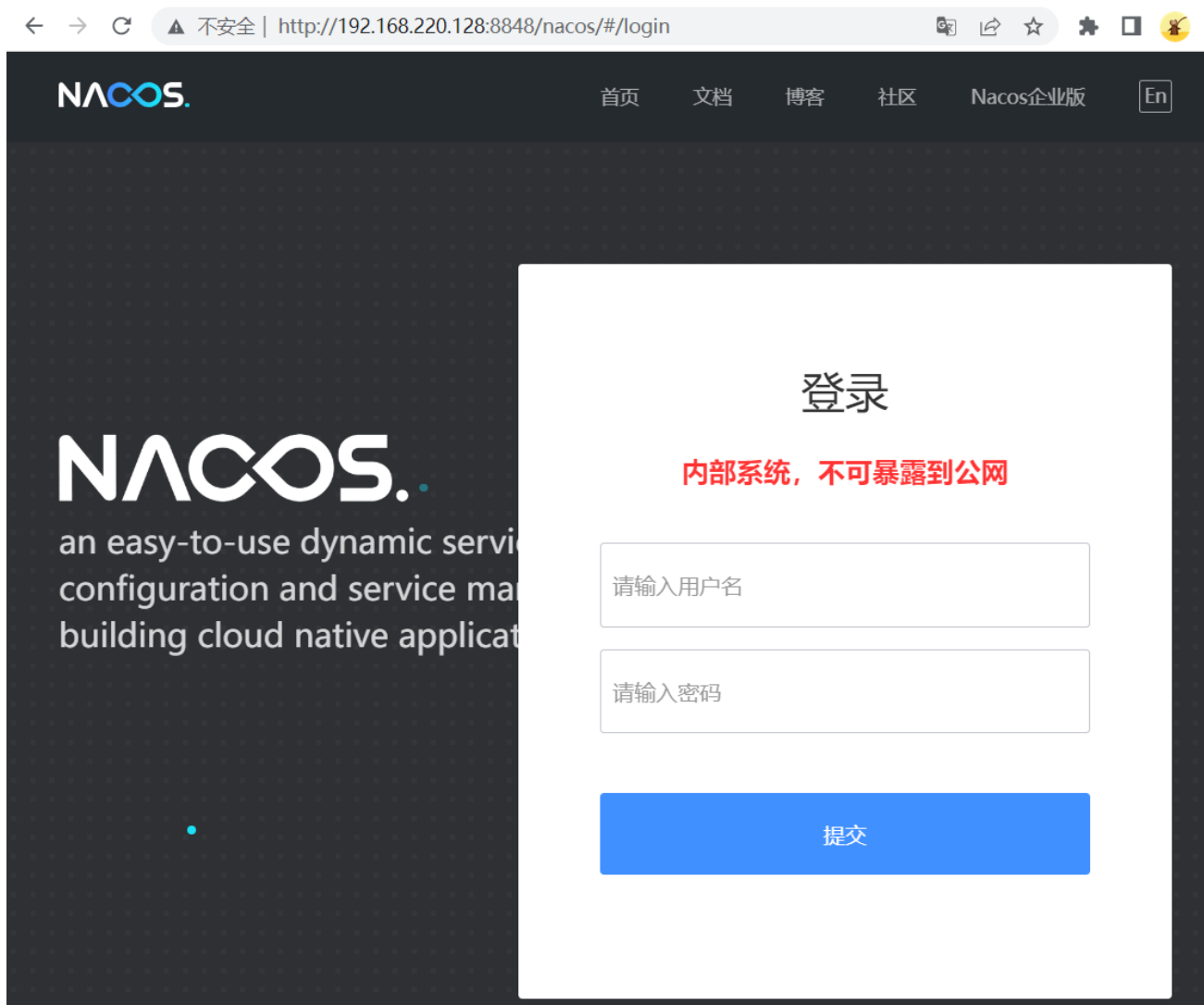
首次启动会拉取nacos相关镜像，如下图所示

```
[root@localhost nacos-docker]# docker-compose -f standalone-derby.yaml up
[+] Running 0/11
  -: nacos Pulling
    : 2d473b07cdd5 Downloading
    : 815fc4d5db6e Download complete
    : 7128bc78cbbc Downloading [=====] 66.13MB/210.1MB
    : 7d739257ae96 Downloading [=====] 52.96MB/117.5MB
    : 9b8d32566260 Downloading [=====] 48.73MB/117.5MB
    : a6223e2872bd Waiting
    : 267d1d69b4e3 Waiting
    : 98fbaef82461 Waiting
    : 8b43f8d72154 Waiting
    : 4f4fb700ef54 Waiting
```

启动成功后

```
nacos-standalone | Running in stand alone mode, All function
nacos-standalone | Port: 8848
nacos-standalone | Pid: 1
nacos-standalone | Console: http://1e8f28a3609b:8848/nacos/in
nacos-standalone | https://nacos.io
nacos-standalone |
nacos-standalone | 2022-07-29 16:14:55,662 INFO Bean 'org.springframework.security.access.expression.method.DefaultMe
of type [org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler] is not eligible
processors (for example: not eligible for auto-proxying)
nacos-standalone | 2022-07-29 16:14:55,671 INFO Bean 'methodSecurityMetadataSource' of type [org.springframework.secu
ityMetadataSource] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto
nacos-standalone | 2022-07-29 16:14:56,173 INFO Tomcat initialized with port(s): 8848 (http)
nacos-standalone | 2022-07-29 16:14:56,778 INFO Root WebApplicationContext: initialization completed in 6527 ms
nacos-standalone | 2022-07-29 16:15:05,573 INFO Initializing ExecutorService 'applicationTaskExecutor'
nacos-standalone | 2022-07-29 16:15:05,866 INFO Adding welcome page: class path resource [static/index.html]
nacos-standalone | 2022-07-29 16:15:06,526 INFO Creating filter chain: Ant [pattern='/*'], []
nacos-standalone | 2022-07-29 16:15:06,577 INFO Creating filter chain: any request, [org.springframework.security.web
tegrationFilter@3b2f4a93, org.springframework.security.web.context.SecurityContextPersistenceFilter@2015b2cd, org.spr
iterFilter@2416498e, org.springframework.security.web.csrf.CsrfFilter@4e73b552, org.springframework.security.web.auth
org.springframework.security.web.savedrequest.RequestCacheAwareFilter@2324bfe7, org.springframework.security.web.servl
ilter@4016ccc1, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@213bd3d5, org.springfram
mentFilter@240f6c41, org.springframework.security.web.access.ExceptionTranslationFilter@2cec704c]
nacos-standalone | 2022-07-29 16:15:06,723 INFO Initializing ExecutorService 'taskScheduler'
nacos-standalone | 2022-07-29 16:15:06,770 INFO Exposing 2 endpoint(s) beneath base path '/actuator'
nacos-standalone | 2022-07-29 16:15:06,968 INFO Tomcat started on port(s): 8848 (http) with context path '/nacos'
nacos-standalone | 2022-07-29 16:15:06,979 INFO Nacos started successfully in stand alone mode. use embedded storage
nacos-standalone |
```

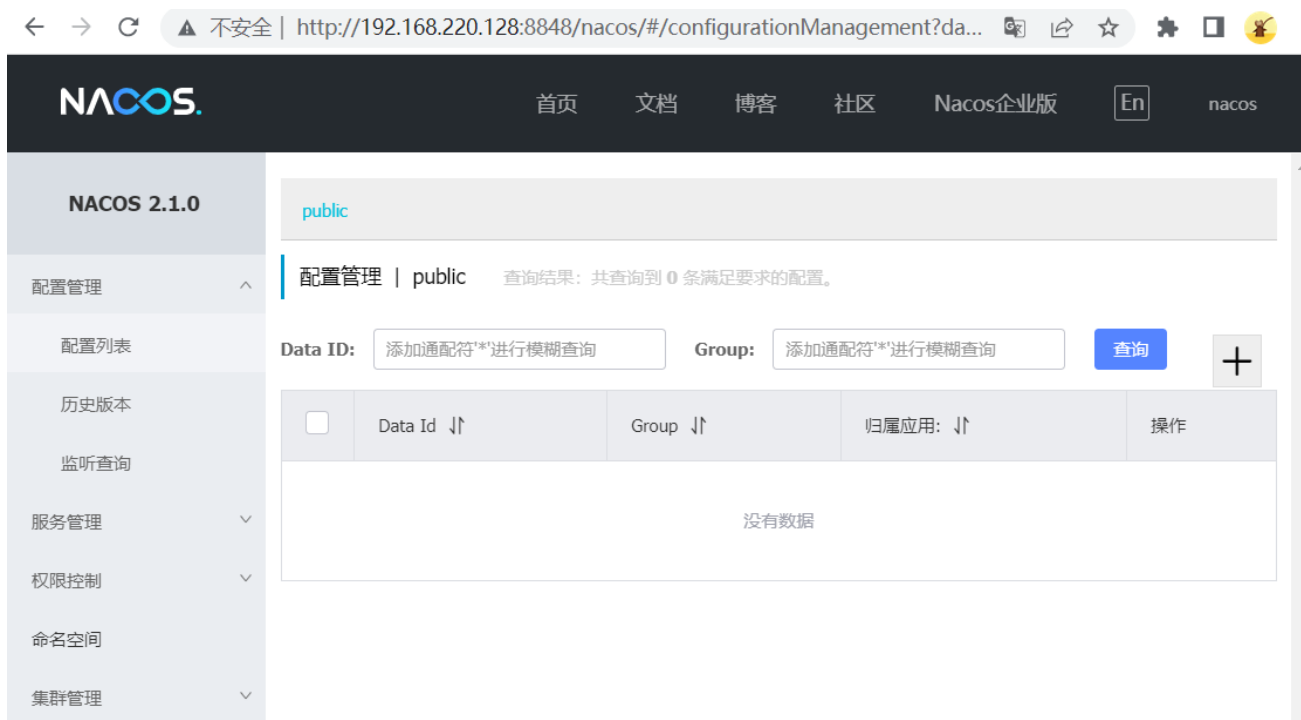
浏览器访问: <http://ip:8848/nacos/>



输入

账号：nacos

密码：nacos，进入管理中心。



接下来我们来新建一个配置，点击查询按钮后面的+图标，按照下图示意创建配置文件。

The screenshot shows the Nacos configuration creation interface. It includes the following elements:

- Data ID:** hello.properties (Step 1: 输入配置文件名称)
- Group:** DEFAULT_GROUP (Step 2: 输入配置文件分组)
- 更多高级选项** (More Advanced Options)
- 描述:** (Description)
- 配置格式:** TEXT, JSON, XML, YAML, HTML, Properties (Step 3: 选择配置文件类型)
- * 配置内容:** (Step 4: 输入配置文件内容)
 - 1 user.id=1
 - 2 user.age=18
 - 3 user.name=ddg
- 发布配置** (Step 5: 发布) and **返回** (Return)

发布配置后可以看到下面的结果

The screenshot shows the Nacos configuration management interface. It includes the following elements:

- NACOS 2.1.0**
- 配置管理** (Configuration Management)
- 配置列表** (Configuration List)
- 历史版本** (History Version)
- 监听查询** (Listen Query)
- 服务管理** (Service Management)
- 权限控制** (Permission Control)
- 命名空间** (Namespace)
- 集群管理** (Cluster Management)
- public** (Group)
- 配置管理 | public** (Configuration Management | public)
- 查询结果: 共查询到 1 条满足要求的配置。** (Query Result: 1 configuration found that meets the requirements.)
- Data ID:** 添加通配符 "*" 进行模糊查询 (Add wildcard "*" for fuzzy search)
- Group:** 添加通配符 "*" 进行模糊查询 (Add wildcard "*" for fuzzy search)
- 查询** (Query)
- 高级查询** (Advanced Query)
- 导入配置** (Import Configuration)
- +** (Add)
- | | Data Id ↓↑ | Group ↓↑ | 归属应用: ↓↑ | 操作 |
|--------------------------|------------------|---------------|----------|--------------------------|
| <input type="checkbox"/> | hello.properties | DEFAULT_GROUP | | 详情 示例代码 编辑 删除 更多 |
- 删除** (Delete)
- 克隆** (Clone)
- 导出** (Export)
- 每页显示:** 10 (Items per page)
- < 上一页** (Previous Page)
- 1** (Page 1)
- 下一页 >** (Next Page)

没有问题后，直接ctrl + c，结束前台启动服务，切换为后台方式启动。

```
1 docker-compose -f standalone-derby.yaml up -d
```

启动成功后可以执行下面命令查看启动日志

```
1 docker logs nacos-standalone
```

看到下面的结果表示后台启动成功

```
Running in stand alone mode, All function modules
Port: 8848
Pid: 1
Console: http://8a93c8a81e8a:8848/nacos/index.html
https://nacos.io

2022-07-29 16:30:55,875 INFO Bean 'org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler' is not eligible for get
e: not eligible for auto-proxying)

2022-07-29 16:30:55,882 INFO Bean 'methodSecurityMetadataSource' of type [org.springframework.security.access not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxyin

2022-07-29 16:30:56,331 INFO Tomcat initialized with port(s): 8848 (http)

2022-07-29 16:30:56,871 INFO Root WebApplicationContext: initialization completed in 5735 ms

2022-07-29 16:31:03,556 INFO Initializing ExecutorService 'applicationTaskExecutor'

2022-07-29 16:31:03,828 INFO Adding welcome page: class path resource [static/index.html]

2022-07-29 16:31:04,332 INFO Creating filter chain: Ant [pattern='/**'], []

2022-07-29 16:31:04,376 INFO Creating filter chain: any request, [org.springframework.security.web.context.d3d5, org.springframework.security.web.context.SecurityContextPersistenceFilter@5ec46cdd, org.springframework.org.springframework.security.web.csrf.CsrfFilter@3bdb2c78, org.springframework.security.web.authentication.se
security.web.savedrequest.RequestCacheAwareFilter@112dlc8e, org.springframework.security.web.servletapi.Se
springframework.security.web.authentication.AnonymousAuthenticationFilter@470a659f, org.springframework.se
org.springframework.security.web.access.ExceptionTranslationFilter@1c758545]

2022-07-29 16:31:04,506 INFO Initializing ExecutorService 'taskScheduler'

2022-07-29 16:31:04,533 INFO Exposing 2 endpoint(s) beneath base path '/actuator'

2022-07-29 16:31:04,689 INFO Tomcat started on port(s): 8848 (http) with context path '/nacos'

2022-07-29 16:31:04,695 INFO Nacos started successfully in stand alone mode. use embedded storage
```

4 常见问题解决参考

1 云服务器部署的nacos服务程序无法访问参考下面链接:

<https://nacos.io/zh-cn/docs/v2/upgrading/2.0.0-compatibility.html>

2 nacos服务重启数据丢失

导致这个问题的可能原因是, 你重启nacos服务的时候使用的是下列命令

```
1 docker-compose -f standalone-derby.yaml up -d
```

up指令会重新创建容器, 这样就导致数据丢失了。

如果服务器宕机后重启nacos建议使用下列命令。

```
1 docker-compose -f standalone-derby.yaml start
```

还有一种方式就是将数据持久化到数据库中, 关于持久化到数据库中这里不做说明, 参考官方对应的服务编排文件修改即可。