# □□□□

mirror.nju.edu.cn

- □□
- GCR
- GHCR
- GLCR
- NGC
- Quay
- K8s
- Go
- Maven
- npm
- □□□□

# □□

## □□□

| □□ | IPv4(A) | IPv6(AAAA) |
| --- | --- | --- |
| mirror.nju.edu.cn | Y | Y |
| mirrors.nju.edu.cn | Y | Y |
| ipv4.mirror.nju.edu.cn | Y | N |
| ipv4.mirrors.nju.edu.cn | Y | N |
| v4.mirror.nju.edu.cn | Y | N |
| v4.mirrors.nju.edu.cn | Y | N |
| ipv6.mirror.nju.edu.cn | N | Y |
| ipv6.mirrors.nju.edu.cn | N | Y |
| v6.mirror.nju.edu.cn | N | Y |
| v6.mirrors.nju.edu.cn | N | Y |

## □□□

| □□ | □□ |
| --- | --- |
| HTTP | 80 |
| HTTPS | 443 |
| RSYNC | 873 |
| RSYNC SSL | 874 |

# GCR

Google Container Registry

## gcr.io

□ `gcr.io` □□□ `gcr.nju.edu.cn` □□□

□

```
docker pull gcr.io/cadvisor/cadvisor:v0.39.3
```

□□□

```
docker pull gcr.nju.edu.cn/cadvisor/cadvisor:v0.39.3
```

## k8s.gcr.io

□ `k8s.gcr.io` □□□ `gcr.nju.edu.cn/google-containers` □□□

□

```
docker pull k8s.gcr.io/kube-proxy:v1.18.6
```

□□□

```
docker pull gcr.nju.edu.cn/google-containers/kube-proxy:v1.18.6
```

# GHCR

Github Container Registry

# ghcr.io

 `ghcr.io` □□□  `ghcr.nju.edu.cn` □□□

□

```
docker pull ghcr.io/github/super-linter:latest
```

□□□

```
docker pull ghcr.nju.edu.cn/github/super-linter:latest
```

# GLCR

GitLab Container Registry

# registry.gitlab.com

□ `registry.gitlab.com` □□□ `glcr.nju.edu.cn` □□□

□

```
docker pull registry.gitlab.com/pipeline-components/yamllint
```

□□□

```
docker pull glcr.nju.edu.cn/pipeline-components/yamllint
```

# NGC

NVIDIA NGC Catalog

# nvcr.io

□ `nvcr.io` □□□ `nvcr.nju.edu.cn` □ `ngc.nju.edu.cn` □□□

□

```
docker pull nvcr.io/nvidia/cuda
```

□□□

```
docker pull nvcr.nju.edu.cn/nvidia/cuda
```

# Quay

Quay Container Registry

# quay.io

 `quay.io` □□□ `quay.nju.edu.cn` □□□

□

```
docker pull quay.io/prometheus/prometheus:latest
```

□□□

```
docker pull quay.nju.edu.cn/prometheus/prometheus:latest
```

# K8s

Kubernetes's container image registry

# registry.k8s.io

 `registry.k8s.io` □□□ `k8s.nju.edu.cn` □□□

□

```
docker pull registry.k8s.io/busybox
```

□□□

```
docker pull k8s.nju.edu.cn/busybox
```

Go

注意，开源镜像同时提供 go 二进制文件下载，具体文件请 参考链接 。

go 配置方式比较简单，执行以下命令：

go env -w GO111MODULE=on # 启用 Go Modules 功能
go env -w GOPROXY="https://repo.nju.edu.cn/go/,direct" # 配置代理 go env | grep GOPROXY # linux 确认配置

go env |findstr "GOPROXY" # windows 确认配置

如有私有模块，可以通过以下命令特殊设置：

go env -w GOPRIVATE="*.nju.edu.cn"

# Maven

采用官方 maven central + gradle-plugin 混合形式。 gradle 和 maven 使用的是同一个仓库，不过配置方法略有区别。

# maven 配置指南

## 全局配置

配置当前登录用户使用南大镜像站，如果未修改过 maven 的默认配置文件位置，请如下操作:

- linux 用户在终端中输入以下命令 :

```
mkdir ~/.m2
vim ~/.m2/settings.xml
```

- windows 用户在 powershell (或者 pwsh中 ) 输入以下命令:

```
mkdir ~/.m2
cd ~/.m2
notepad settings.xml
```

如果打开的文件为空，粘贴以下内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <mirrors>
       <mirror>
            <id>nju_mirror</id>
            <mirrorOf>*</mirrorOf>
            <url>https://repo.nju.edu.cn/maven/</url>
        </mirror>
    </mirrors>
```

```
</settings>
```

如果已经有内容，则在 `mirrors` 节点加入子节点 (对于 mirrorOf 值相同的节点，只有首个子节点生效 ):

```
<mirror>
    <id>nju_mirror</id>
    <mirrorOf>*</mirrorOf>
    <url>https://repo.nju.edu.cn/maven/</url>
</mirror>
```

保存退出即可使用。

## 项目配置

如果只在某个项目中使用，则在项目的 `pom.xml` 中配置:

```
<project>
    ......[其他配置]

    <repositories>
        <repository>
            <id>nju</id>
            <url>https://repo.nju.edu.cn/maven/</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
</project>
```

# gradle-plugin 配置指南

注意，开源镜像同时提供 gradle 二进制文件下载，将项目文件下的 `gradle\wapper\gradle-wrapper.properties` 的 `distributionUrl` 修改为 `http://mirror.nju.edu.cn/gradle/gradle-[version].zip` 即可，详细版本请 参考链接_____ 。

# □□□

□□□□□□□□□□□□□□□□□□□□□□□□ gradle □□□□□□□□□□□□□□□□ :

- linux □□□□□□□□□□□□ :

```
mkdir ~/.gradle
vim ~/.gradle/init.gradle
```

- windows □□□ powershell (□□ pwsh□ ) □□□□□□ :

```
mkdir ~/.gradle
cd ~/.gradle
notepad init.gradle
```

□□□□□□□□ :

```
allprojects {
    buildscript {
        repositories {
            def NJU_REPOSITORY_URL = 'https://repo.nju.edu.cn/maven/'
            all { ArtifactRepository repo ->
                if (repo instanceof MavenArtifactRepository) {
                    def url = repo.url.toString()
                    if (url.startsWith('https://repo1.maven.org/maven2')) {
                        project.logger.lifecycle "Repository ${repo.url} replaced by
$NJU_REPOSITORY_URL."
                        remove repo
                    }
                    if (url.startsWith('https://jcenter.bintray.com/')) {
                        project.logger.lifecycle "Repository ${repo.url} deleted."
                        remove repo
                    }
                    if (url.startsWith('https://dl.google.com/dl/android/maven2/')) {
                        project.logger.lifecycle "Repository ${repo.url} deleted."
                        remove repo
                    }
                    if (url.contains('plugins.gradle.org/m2')) {
                        project.logger.lifecycle "Repository ${repo.url} deleted."
                        remove repo
                    }
```

```
                }
            }
            maven { url NJU_REPOSITORY_URL }
            mavenLocal()
        }
    }

    repositories {
        def NJU_REPOSITORY_URL = 'https://repo.nju.edu.cn/maven/'
        all { ArtifactRepository repo ->
            if (repo instanceof MavenArtifactRepository) {
                def url = repo.url.toString()
                if (url.startsWith('https://repo1.maven.org/maven2')) {
                    project.logger.lifecycle "Repository ${repo.url} replaced by
$NJU_REPOSITORY_URL."
                    remove repo
                }
                if (url.startsWith('https://jcenter.bintray.com/')) {
                    project.logger.lifecycle "Repository ${repo.url} deleted."
                    remove repo
                }
                if (url.startsWith('https://dl.google.com/dl/android/maven2/')) {
                    project.logger.lifecycle "Repository ${repo.url} deleted."
                    remove repo
                }
                if (url.contains('plugins.gradle.org/m2')) {
                    project.logger.lifecycle "Repository ${repo.url} deleted."
                    remove repo
                }
            }
        }
        maven { url NJU_REPOSITORY_URL }
        mavenLocal()
    }
}
```

□□□□□□□ gradle □□□□□□□ `gradle --stop` □□□□□□ gradle □ daemon□ □□□□□□□□□

# 项目配置

如果只在某个项目中使用，则在项目的 `build.gradle` 中配置 ：

```
buildscript {
    repositories {
        maven { url 'https://repo.nju.edu.cn/maven/' }
    }
}


plugins {
    ...[你需要用的 gradle 插件]
}

allprojects {
    repositories {
        maven { url 'https://repo.nju.edu.cn/maven/' }
    }
}
```

注意这样可能有时候无法通过镜像下载部分 gradle 插件。

# npm

Node Package Manager

南大源采用了腾讯云和官方源的混合代理形式。

# npm 配置指南

## 临时使用

```
npm install [packageName] --registry=https://repo.nju.edu.cn/repository/npm/
```

使用此命令可以临时使用南大源下载指定包。

## 设置为默认源

使用一下命令修改默认源:

```
npm config set registry https://repo.nju.edu.cn/repository/npm/ # 修改源
```

# npm 私服

南京大学 npm 仓库同时提供 **npm** 包托管服务，本校师生可以将自己编写的 js 脚本打包后发布到仓库中让其他人使用。此仓库文件不会同步到其他镜像源。

## 从仓库获取文件

将上面提到的仓库地址改为 `https://repo.nju.edu.cn/repository/npm-nju/` 即可。

使用此仓库请注意：

- 此仓库会优先从官方库和其他源中获取包，如果你发布了同名包，会被其他来源的同名包覆盖，因此请不要也禁止发布同名包或者名称相近包；

- 当仓库网络出现问题或者上游源出现问题时，可能会导致你下载到仓库其他人发布的同名包，这可能不是你想要的，甚至有风险（pypi 仓库则没有此问题）；
- 如果发现和上游仓库同名包，请 填写统计表_____，管理员将会定期删除；
- 如果发现恶意同名包或者类似包，请 填写统计表_____，管理员将核实后采取措施。

## 向仓库提交文件

1. 使用 npm 登陆:

   ```
   npm adduser --registry https://repo.nju.edu.cn/repository/npm-releases/
   ```

   其中，邮箱推荐使用南大邮箱，账号密码为统一身份认证账号密码。
2. 在项目的 `package.json` 中加入以下内容:

   ```
   "publishConfig": {
     "registry" : "https://repo.nju.edu.cn/repository/npm-releases/"
   },
   ```

3. 配置 `package.json` 中的其他配置项。
4. 配置 `.npmignore` 指定你不想发布的文件（夹）。
5. 构建并推送:

   ```
   npm run build
   npm publish
   ```

# yarn 配置指南

```
yarn config set registry https://repo.nju.edu.cn/repository/npm/ # 修改源
```

yarn 发布的过程和 npm 类似，不过 `yarn login` 的时候不需要输入密码，在每次发布的时候输入，具体内容请查看 yarn publish 的文档。

触发同步

原理
通过 SSH 触发镜像站立刻开始同步指定仓库，从而解决镜像站周期同步的延迟问题。可以将此集成到 CI/CD 中，从而实现自动实时发布。

使用方法
1. 专门生成一对 SSH 密钥，每个仓库均需使用独立的密钥
2. 将 ssh-ed25519 公钥发送至 my@yaoge123.com
，并注明需要同步的仓库名称，且需要证明自己对该仓库具有管理权限
3. 触发命令： ssh -T pushsync@mirror.nju.edu.cn -p 52222