

# DevOps



## Nexus制品库实践

# 讲师-泽阳



微信: `proc_code`

持续交付工程师

多年DevOps工具链运维经验  
参与持续交付标准项目改造  
参与项目端到端流水线实践  
打通端到端工具链提升效率  
基于容器设计DevOps流水线

做最好的自己

研究最新技术

提高工作效率

喜好读书电影

创新创新创新

GitHub: <http://github.com/zeyangli>

个人公众号: `devopsadmin` DevOps持续集成





**JFrog Artifactory**

制品库实践



# Jenkins & Artifactory



# 配置Artifactory插件

- 安装Artifactory插件
- 配置Artifactory仓库信息（仓库地址、用户认证信息）

可更新

可选插件

已安装

高级

安装 ↓

☒

[Artifactory](#)  
This plugin allows your build jobs to deploy artifacts and resolve dependencies. It includes a collection of features, including a rich pipeline API library and release management capabilities.

☐

[lambdatest-automation](#)  
Artifactory auto generated POM


直接安装


下载待重启后安装


20 小时 之前获取了更新信息


# 配置Artifactory插件


- 配置Artifactory仓库信息（仓库地址、用户认证信息）


☒ Use the Credentials Plugin 


 Artifactory


Server ID  

URL  



Connection Timeout  

Number of retries  

Number of threads for Generic uploads  

☐ Bypass HTTP Proxy 

Default Deployer Credentials

Credentials   添加 

Found Artifactory 6.16.0

☐ Use Different Resolver Credentials

Test Connection

删除

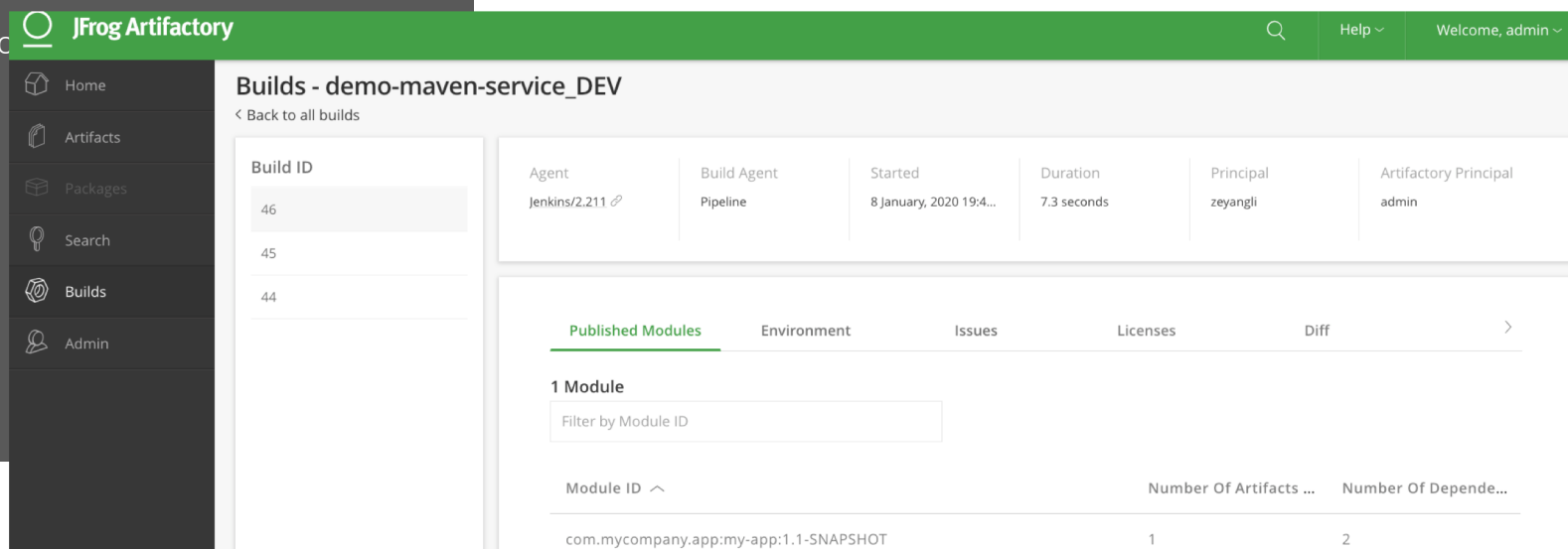
Add **Artifactory** Server

```
package org.devops

//Maven打包构建
def MavenBuild(buildShell){
    def server = Artifactory.newServer url: "http://192.168.1.200:30082/artifactory"
    def rtMaven = Artifactory.newMavenBuild()
    def buildInfo
    server.connection.timeout = 300
    server.credentialsId = 'artifactory-admin-user'
    //maven打包
    rtMaven.tool = 'M2'
    buildInfo = Artifactory.newBuildInfo()

    String newBuildShell = "${buildShell}".toString()
    println(newBuildShell)
    rtMaven.run pom: 'pom.xml', goals: newBuildShell, buildInfo
    //上传build信息
    server.publishBuildInfo buildInfo
}

def main(buildType,buildShell){
    if(buildType == "mvn"){
        MavenBuild(buildShell)
    }
}
```









The screenshot shows the JFrog Artifactory web interface. The top navigation bar is green with the JFrog Artifactory logo, a search icon, and user information. A left sidebar contains navigation links: Home, Artifacts, Packages, Search, Builds, and Admin. The main content area is titled 'Builds - demo-maven-service\_DEV' and includes a 'Back to all builds' link. Below the title is a table of build IDs (46, 45, 44). To the right of the build list is a detailed view for build 46, showing metadata such as Agent (Jenkins/2.211), Build Agent (Pipeline), Started time (8 January, 2020 19:4...), Duration (7.3 seconds), Principal (zeyangli), and Artifactory Principal (admin). Below this is a section for 'Published Modules' with a filter by Module ID and a table listing modules with their artifact and dependency counts.

Build ID	Agent	Build Agent	Started	Duration	Principal	Artifactory Principal
46	Jenkins/2.211	Pipeline	8 January, 2020 19:4...	7.3 seconds	zeyangli	admin
45						
44						

Module ID	Number Of Artifacts ...	Number Of Dependence...
com.mycompany.app:my-app:1.1-SNAPSHOT	1	2

- 仓库命名规范
  - 业务/项目-环境类型 例如: demo-dev
- 制品命名规范
  - 应用名称-版本号-构建ID. type
  - 例如: demo-myapp-service-1.jar
- 制品目录规范
  - 业务/项目
    - 应用名称
      - 版本号
        - 制品

Repository Key	Type
 demo	m Maven
 demo-dev	m Maven
 demo-prod	m Maven
 demo-stag	m Maven
 demo-uat	m Maven
 example-repo-local	g Generic

## Artifact Repository Browser

Tree Simple 



>  artifactory-build-info

>  demo

✓  demo-dev

└─  demo/demo-myapp-service/1.1-SNAPSHOT

└─ >  demo-myapp-service-1.1-20200108.130358-1-4.jar

└─ >  demo-myapp-service-1.1-20200108.130447-1-5.jar

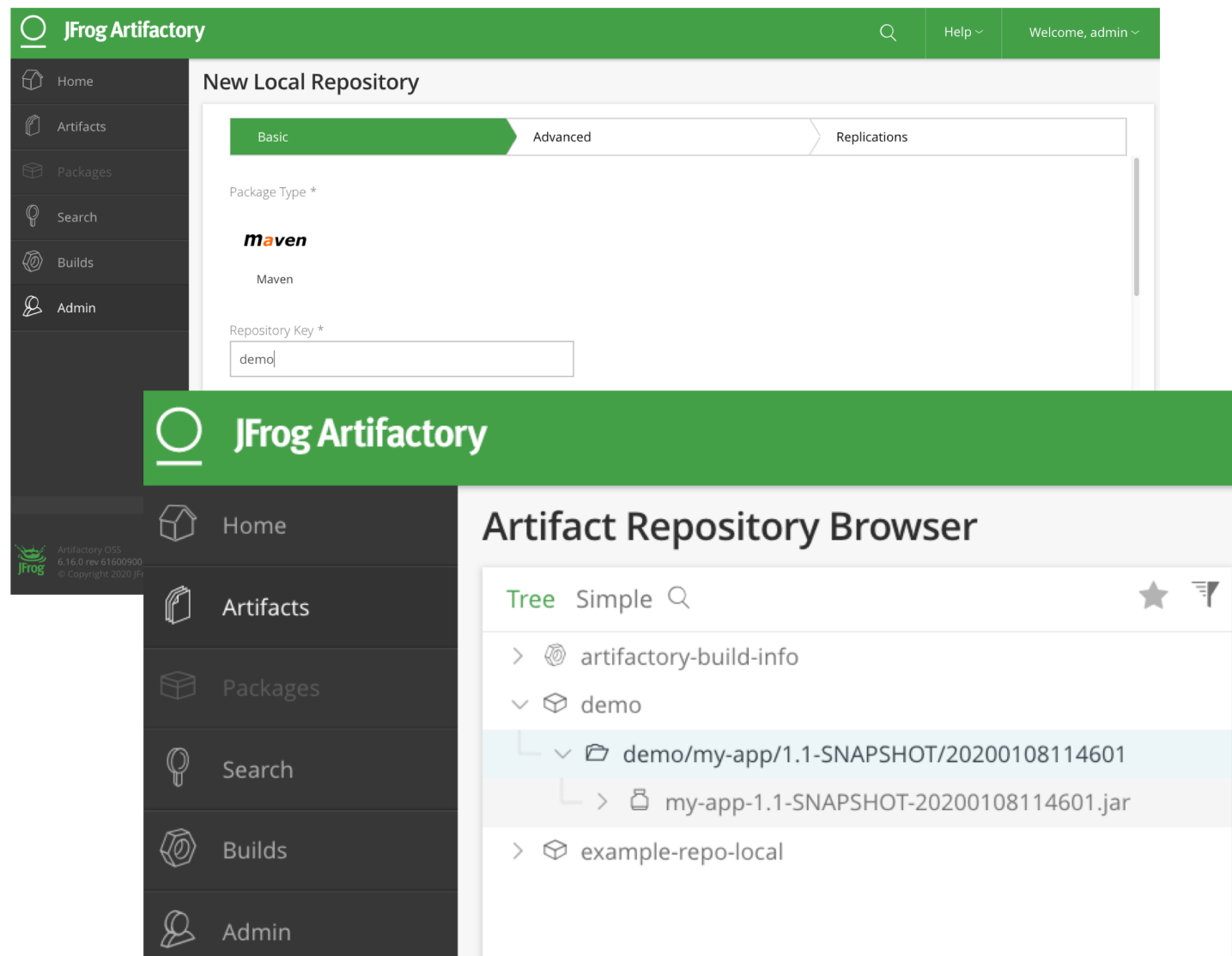
└─ >  demo-myapp-service-1.1-20200108.130449-1-6.jar

└─ >  demo-myapp-service-1.1-20200108.130452-1-7.jar

└─ >  demo-myapp-service-1.1-20200108.130455-1-8.jar



```
rtUpload (  
  serverId: "art1",  
  spec:  
    ""{  
      "files": [  
        {  
          "pattern": "target/${jarName}",  
          "target": "${uploadDir}/"  
        }  
      ]  
    }""  
)
```



The screenshot displays the JFrog Artifactory web interface. The top navigation bar is green with the JFrog Artifactory logo, a search icon, and a user profile dropdown showing 'Welcome, admin'. A left sidebar contains navigation links: Home, Artifacts, Packages, Search, Builds, and Admin. The main content area is divided into two sections. The upper section, titled 'New Local Repository', features three tabs: 'Basic' (selected), 'Advanced', and 'Replications'. Under the 'Basic' tab, the 'Package Type' is set to 'Maven', and the 'Repository Key' is 'demo'. The lower section, titled 'Artifact Repository Browser', shows a tree view of the repository structure. The tree includes a root node 'artifactory-build-info', a 'demo' repository, and a sub-repository 'demo/my-app/1.1-SNAPSHOT/20200108114601' which contains the artifact 'my-app-1.1-SNAPSHOT-20200108114601.jar'. Other repositories like 'example-repo-local' are also visible.



# 用实际经验制作最实用的教程！

个人公众号： devopsadmin DevOps持续集成 GitHub: <http://github.com/zeyangli>



讲师泽阳

微信：proc\_code

