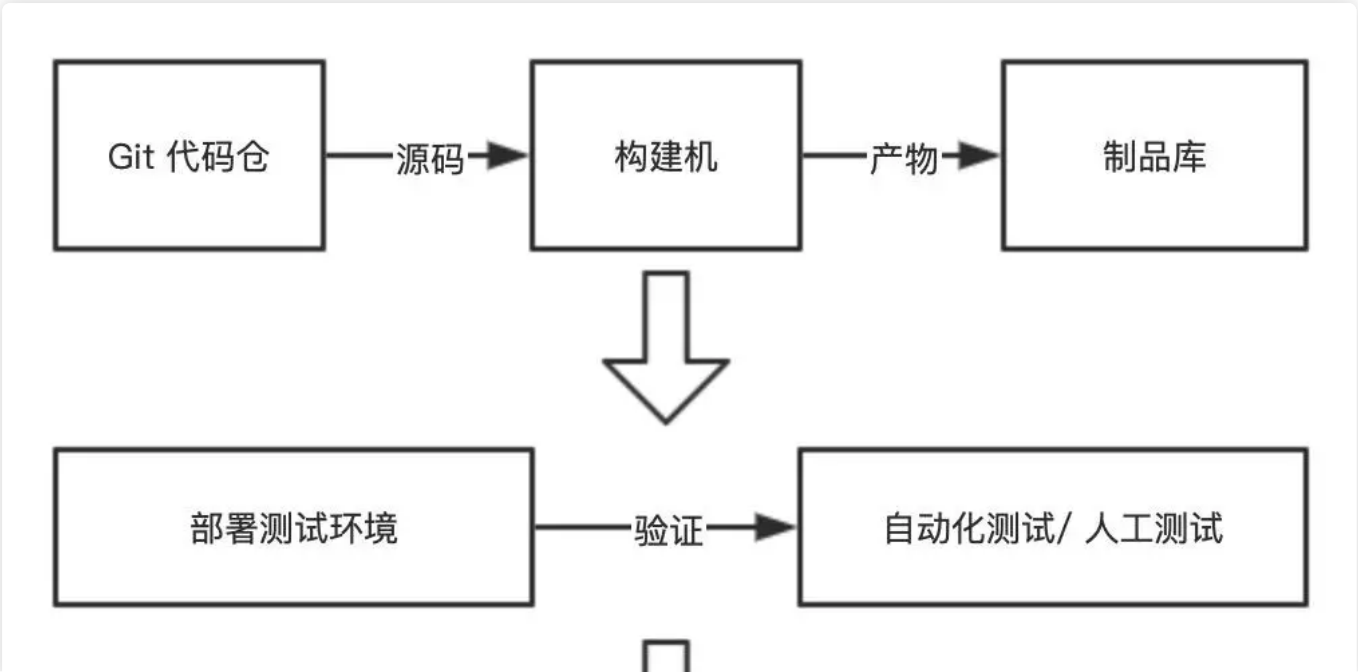
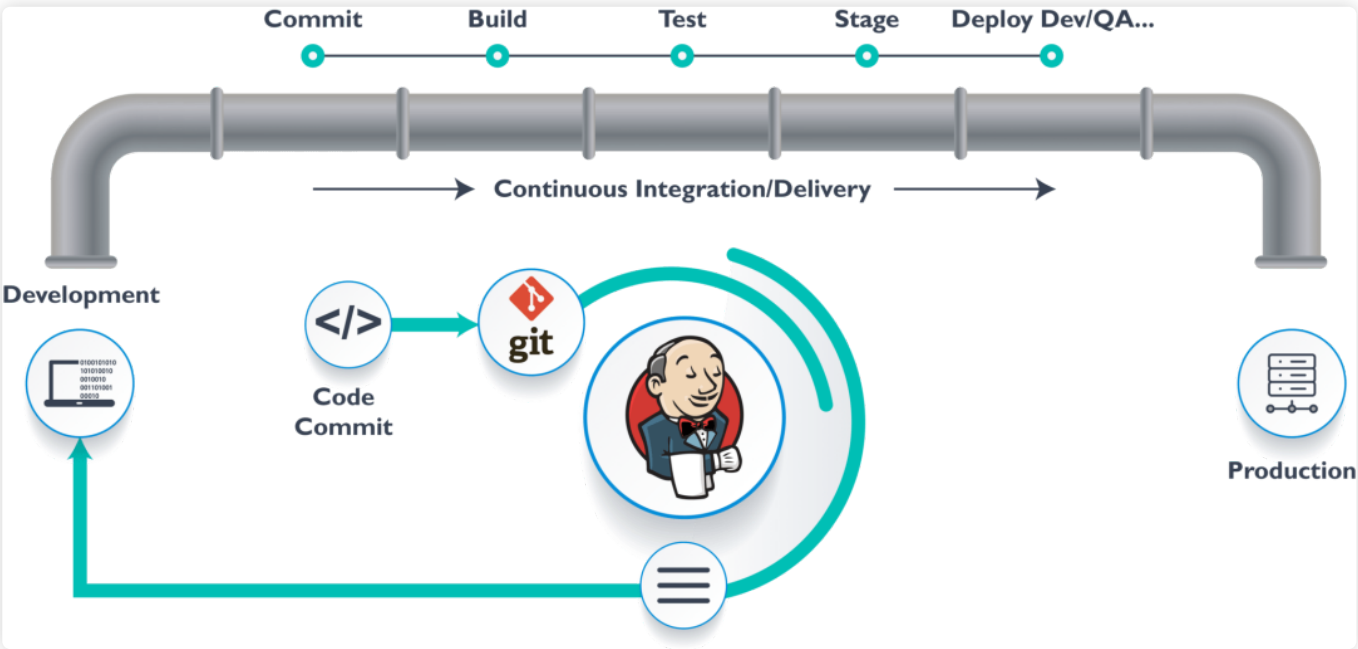


自动化流程

企业里常见的项目自动化流程应该构建机从代码仓拉取代码进行构建，构建完成后会将产物推送到制品库中，比如镜像仓，然后中间会有测试环境，用于进行自动化测试或人工测试，最后进行远程部署。





## 项目结构

这里我们用的 Go 的项目结构，它大概的结构应该是下面这样的：

```
my-app
├── .gitignore
├── README.md
├── LICENSE
├── go.mod
├── go.sum
├── main.go
├── pkg
└── ...
```

## 项目构建

因为这里构建的是 Go 的项目，如果用到私有库，在 go mod tidy 时会要求提供 Git 凭证，我们可以现在 Jenkins 的凭证管理中创建 Username with password 类型的凭证，其中 Username 就是 GitHub 的用户名，password 则是 GitHub 的 AccessToken，这里主要用到的是 AccessToken，Username 其实并不需要。但在 Jenkins Pipeline 中使用 usernamePassword 时要求同时定义用户名变量名 usernameVariable 和 密码变量名 passwordVariable 。

```
stage('Build') {
    steps {
        withCredentials(bindings: [
            usernamePassword(credentialsId: 'GITHUB_CREDENTIAL',
                             usernameVariable: 'GITHUB_USER',
                             passwordVariable: 'GITHUB_ACCESS_TOKEN'
            )
        ]) {
            sh '''
                git config --global url."https://${GITHUB_ACCESS_TOKEN}:x-oauth-basic@github.com/".insteadOf "https://github.com/"

                go mod tidy
                go build -o bin/my-app main.go
                ...
            '''
        }
    }
}
```

## 远程部署

在构建完成后，我们会将构建产物推送到制品库，然后我们可以从制品库中拉取构建产物进行部署测试环境并进行测试，在验证通过后，会从制品库中拉取验证通过的产物进行部署上线。

但在本文中，我们的应用相对简单，可以忽略推送产物到制品库以及中间的测试验证环节，目标是实现构建后立即部署上线。

一般来说，线上环境和构建环境不会是同一台机器，所以这个时候我们需要将构建产物复制到另一台服务器上，然后在另一台服务器上进行部署。

由于需要对另一台服务器进行操作，所以我们需要在 Jenkins 上配置 DEPLOY\_HOST、DEPLOY\_PORT 和 SSH\_CREDENTIAL 三个凭证，其中 DEPLOY\_HOST 和 DEPLOY\_PORT 是 Secret text 类型的凭证，SSH\_CREDENTIAL 是 SSH Username with private key 类型的凭证。

```
stage('Deploy') {
    environment {
```

```
DEPLOY_HOST = credentials('DEPLOY_HOST')
DEPLOY_PORT = credentials('DEPLOY_PORT')
}
steps {
    withCredentials([
        sshUserPrivateKey(credentialsId: 'SSH_CREDENTIAL',
                           keyFileVariable: 'SSH_KEY',
                           usernameVariable: 'SSH_USERNAME'),
    ]) {
        sh """
        mkdir -p ~/.ssh && chmod 700 ~/.ssh
        echo 'StrictHostKeyChecking no' >> /etc/ssh/ssh_config
        cat ${SSH_KEY} > ~/.ssh/id_rsa && chmod 400 ~/.ssh/id_rsa

        scp -P ${DEPLOY_PORT} bin/my-app ${SSH_USER}@${DEPLOY_HOST}:/data/my-app
        ssh -p ${DEPLOY_PORT} ${SSH_USER}@${DEPLOY_HOST} \
        "nohup /data/my-app >> /data/my-app.log 2>&1 &\
        """
    }
}
```

部署的步骤主要包括：

- 复制构建产物到部署服务器
- 在部署服务器上执行部署命令，比如 nohup /data/my-app >> /data/my-app.log 2>&1 &

其中简化了一些细节，比如在部署前，我们需要先备份数据。所以这里我们可以写一个复杂的部署脚本 `deploy.sh` 放在项目中，然后在 Jenkins Pipeline 中使用 `scp` 将部署脚本文件复制到部署服务器，假设放在 `/data/deploy.sh`，最后只需 `ssh -p ${DEPLOY_PORT} {DEPLOY_HOST} /bin/bash /data/deploy.sh` 即可。

### 完整的 Jenkins Pipeline

```
pipeline {
    agent {
        docker {
            image 'golang:1.15-alpine'
            args '-v /data/my-app-cache:/go/.cache'
        }
    }

    options {
        timeout(time: 20, unit: 'MINUTES')
        disableConcurrentBuilds()
    }

    stages {
        stage('Build') {
            steps {
                withCredentials(bindings: [
                    usernamePassword(credentialsId: 'GITHUB_CREDENTIAL',
                                    usernameVariable: 'GITHUB_USER',
                                    passwordVariable: 'GITHUB_ACCESS_TOKEN'
                                )
                ]) {
                    sh '''
                    git config --global url."https://${GITHUB_ACCESS_TOKEN}:x-oauth-basic@github.com/".insteadOf "https://github.com/"

                    go mod tidy
                    go build -o bin/my-app main.go
                    ...
                    '''
                }
            }
        }
    }
}
```



```
stage('Deploy') {
    environment {
        DEPLOY_HOST = credentials('DEPLOY_HOST')
        DEPLOY_PORT = credentials('DEPLOY_PORT')
    }
    steps {
        withCredentials([
            sshUserPrivateKey(credentialsId: 'SSH_CREDENTIAL',
                              keyFileVariable: 'SSH_KEY',
                              usernameVariable: 'SSH_USERNAME'),
        ]) {
            sh """
            mkdir -p ~/.ssh && chmod 700 ~/.ssh
            echo 'StrictHostKeyChecking no' >> /etc/ssh/ssh_config
            cat ${SSH_KEY} > ~/.ssh/id_rsa && chmod 400 ~/.ssh/id_rsa

            scp -P ${DEPLOY_PORT} bin/my-app ${SSH_USER}@${DEPLOY_HOST}:/data/my-app
            ssh -p ${DEPLOY_PORT} ${SSH_USER}@${DEPLOY_HOST} \"nohup /data/my-app >> /data/my-app.log 2>&1 &\"
            """
        }
    }
}
```

原文链接：<https://juejin.cn/post/6969968007690846238>

文章转载：DevOps技术栈  
(版权归原作者所有，侵权)



底层原理+项目实战双轮驱动，要学就学专业的


## Linux云计算SRE工程师

双轨教学/20+实战项目/引入多云案例/优化30+技能模块

系统稳定性建设 | 微服务 | Kubernetes | 阿里云  
ELK日志系统 | 持续集成CI/CD| 指标采集

更多课程内容  
请扫码咨询 >>





画中画

马永亮|马哥教育创始人

Linux核心专家  
计算机安全专业硕士  
《Kubernetes进阶实战》作者  
学员遍布一线互联网公司

00:00/00:00

下载视频

倍速

🔊


🔍

腾讯视频

大家都在看

Linux系统入门视频教程 推荐

用腾讯视频观看

 点击下方“阅读原文”查看更多

喜欢此内容的人还喜欢

自动化运维 | 用16 张图带你入门 Ansible，赶紧收藏~  
网络工程师阿龙



带你快速了解 Docker 和 Kubernetes  
石杉的架构笔记



Julia 为 Python 的王冠而来!  
HelloGitHub

