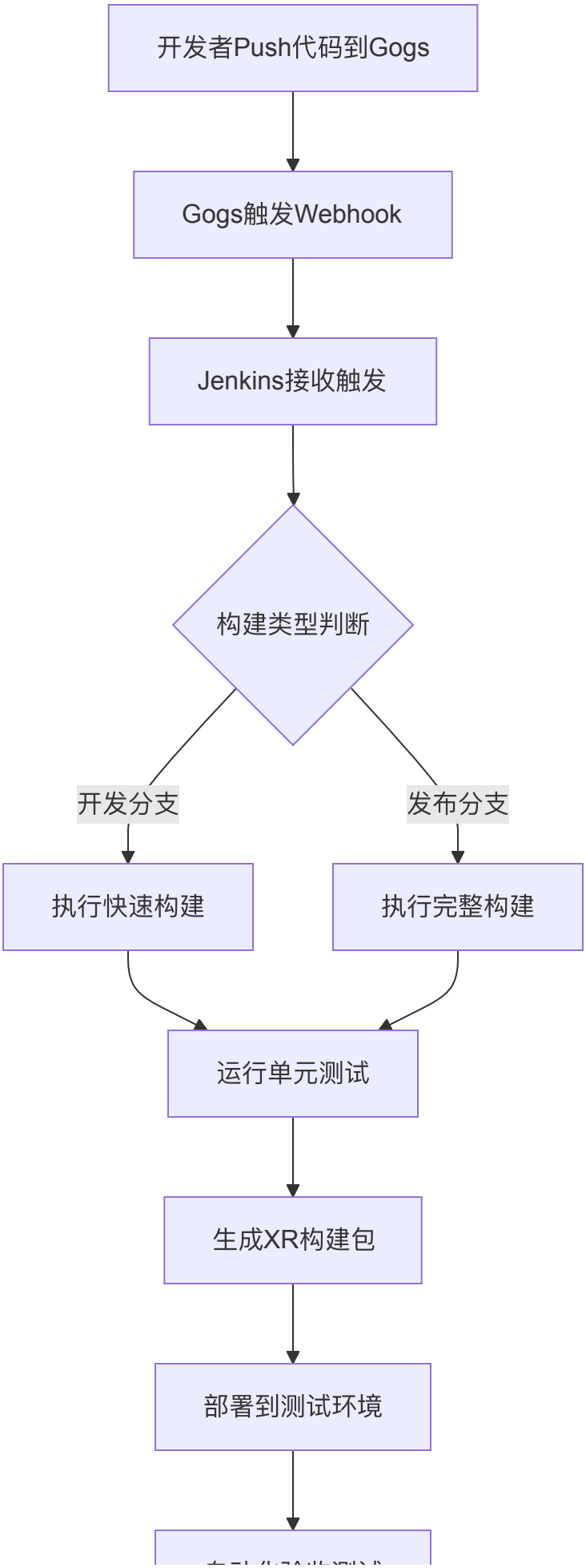
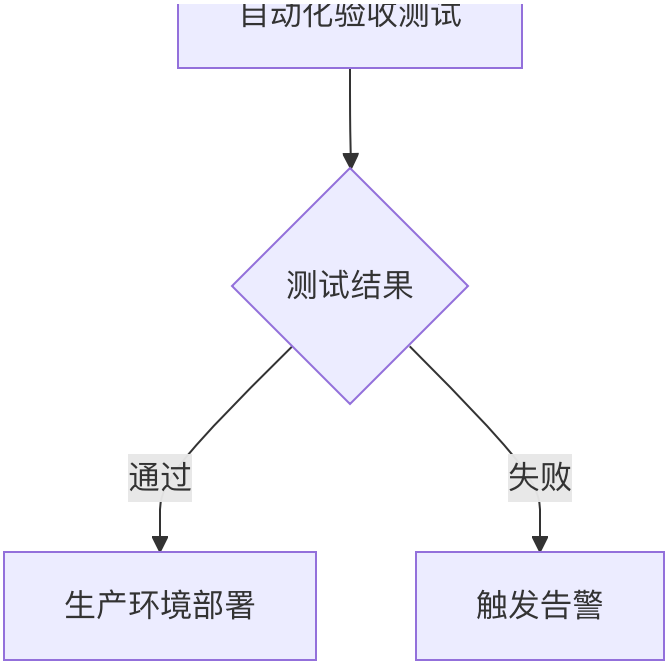
持续化部署和持续化集成

一、整体架构设计


开发者Push代码到Gogs
Gogs触发Webhook
Jenkins接收触发
构建类型判断
开发分支
发布分支
执行快速构建
执行完整构建
运行单元测试
生成XR构建包
部署到测试环境



二、环境准备清单

组件	版本要求	配置建议
Gogs	≥0.12.3	开启Webhook功能
Jenkins	≥2.346.1	安装必要插件
Unity Build Node	2022.3.6f1	配置Android/iOS模块
密钥管理	Vault 1.13.3	与Jenkins集成

三、核心配置步骤

1. Gogs Webhook配置

```
# Webhook地址格式
http://jenkins.yourdomain.com/gogs-webhook/?job=vr_project_build

# 触发事件：
☑ Push 事件
☑ 标签推送事件
☑ 合并请求事件
```

```
# 密钥设置（可选）：
JENKINS_SECRET_TOKEN=your_shared_secret
```

2. Jenkins全局配置

必要插件列表：

- Gogs Plugin
- Pipeline
- Unity3d Plugin
- Credentials Binding
- S3 Publisher

Unity路径配置：

```
# 全局工具配置
Unity 2022.3.6f1路径: /Applications/Unity/Hub/Editor/2022.3.6f1/Unity.app
```

四、Jenkins流水线设计

1. 基础流水线（Jenkinsfile）

```
pipeline {
    agent {
        label 'unity-windows' // 指定构建节点
    }

    environment {
        UNITY_PROJECT = 'VRInteractionSDK'
        BUILD_TARGET = 'Android'
        S3_BUCKET = 'xr-builds'
    }

    stages {
        stage('代码检查') {
            steps {
                checkout([$class: 'GitSCM',
                          branches: [[name: '*/${GIT_BRANCH}']],
                          userRemoteConfigs: [[url:
'http://gogs.yourdomain.com/vr-project.git']]
                ])
```

```
                // 静态代码分析
                unityStaticCodeAnalysis()
            }
        }

        stage('单元测试') {
            steps {
                unityTest(
                    projectPath: "${env.UNITY_PROJECT}",
                    testPlatform: 'EditMode',
                    testResultsFile: 'TestResults.xml'
                )
            }
            post {
                always {
                    junit 'TestResults.xml'
                }
            }
        }

        stage('构建XR包') {
            steps {
                script {
                    def buildParams = [
                        '-batchmode',
                        '-quit',
                        '-projectPath .',
                        '-executeMethod
BuildScript.Build${env.BUILD_TARGET}',
                        '-logFile build.log'
                    ]

                    unityCmd(buildParams.join(' '))
                }
            }
        }

        stage('部署到S3') {
            when {
                branch 'main'
            }
            steps {
                s3Upload(
                    bucket: "${env.S3_BUCKET}",
                    file: "Builds/${env.BUILD_TARGET}/*.apk",
```

```
                    path: "builds/${env.GIT_COMMIT}/"
                )
            }
        }
    }

    post {
        failure {
            slackSend channel: '#ci-alerts',
                      message: "构建失败: ${env.JOB_NAME}
#${env.BUILD_NUMBER}"
        }
    }
}
```

2. 多平台构建矩阵

```
matrix {
    axes {
        axis {
            name 'BUILD_TARGET'
            values 'Android', 'iOS', 'Windows'
        }
    }

    stages {
        stage('Build') {
            steps {
                script {
                    unityBuildTarget("${BUILD_TARGET}")
                }
            }
        }
    }
}
```

五、XR专项优化配置

1. 大资源包处理

```
stage('上传AssetBundle') {
    steps {
```

```
        sh 'split -b 2G huge_asset.bundle'
        s3Upload(
            bucket: 'xr-assets',
            file: 'huge_asset.bundle.*',
            path: 'assetbundles/${GIT_COMMIT}/'
        )
    }
}
```

2. 设备集群部署

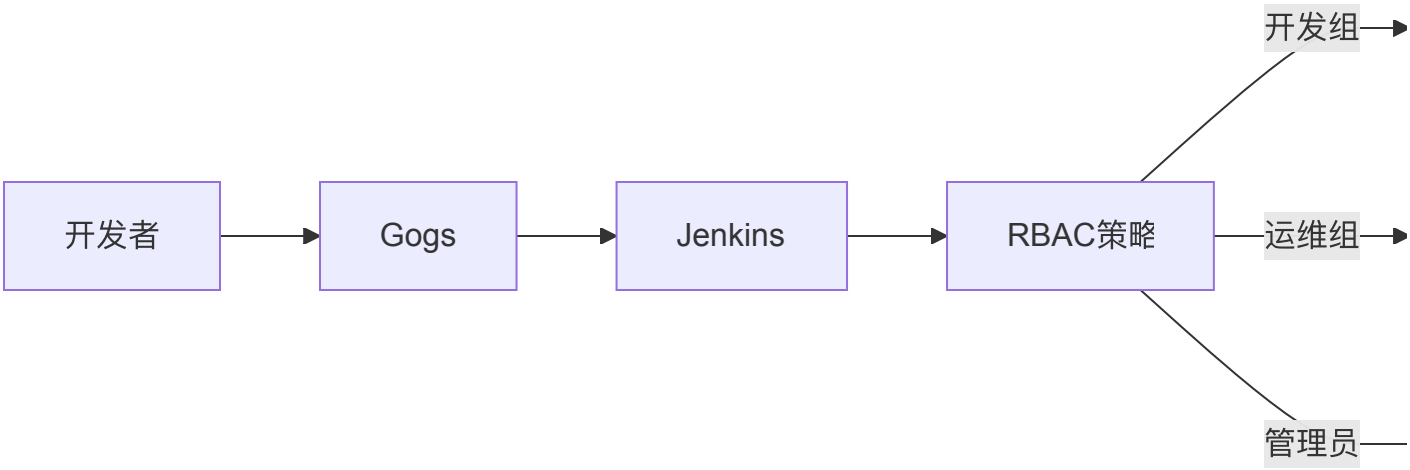
```
stage('部署到XR设备') {
    steps {
        sshPublisher(
            publishers: [
                sshPublisherDesc(
                    configName: 'quest3-cluster',
                    transfers: [
                        sshTransfer(
                            sourceFiles: 'Builds/Android/*.apk',
                            remoteDirectory: '/opt/xr-builds'
                        )
                    ]
                )
            ]
        )
    }
}
```

六、安全增强措施

1. 凭证管理

```
withCredentials([usernamePassword(
    credentialsId: 'unity-license',
    usernameVariable: 'UNITY_USERNAME',
    passwordVariable: 'UNITY_PASSWORD'
)]) {
    sh "unity -username ${UNITY_USERNAME} -password ${UNITY_PASSWORD}"
}
```

2. 访问控制



七、监控与告警

1. Jenkins监控看板

```
# Prometheus指标端点
http://jenkins.yourdomain.com/prometheus/

# 关键监控指标:
- jenkins_builds_total{result="SUCCESS"}
- unity_build_duration_seconds
- xr_deployment_status
```

2. 告警规则示例

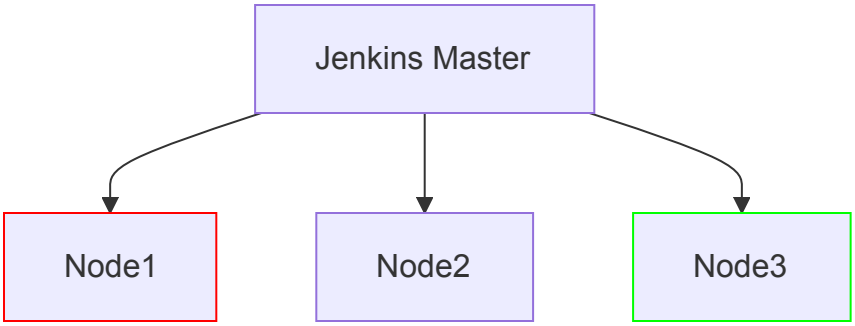
```
alert: XRBuildFailure
expr: increase(jenkins_builds_total{job="vr_project_build",result="FAILURE"}
[1h]) > 3
for: 10m
labels:
  severity: critical
annotations:
  summary: "XR项目持续构建失败"
```

八、性能优化方案

优化方向	具体措施	预期收益
构建缓存	使用ccache加速编译	减少30%构建时间
并行测试	拆分测试集到多个节点	测试时间缩短60%
增量部署	仅上传变更AssetBundle	部署流量减少70%
资源预热	预加载常用场景资源	用户等待时间减少50%

九、灾备方案

1. 构建节点故障转移



2. 数据备份策略

```
# 每日备份计划
0 2 * * * pg_dump -U jenkins | gzip > /backups/jenkins_$(date +%F).sql.gz
0 3 * * * tar czf /backups/gogs_$(date +%F).tar.gz /var/gogs
```

十、实施路线图

1. **阶段一（1周）**
   - 安装配置Jenkins与Gogs插件
   - 建立基础构建流水线
   - 配置Android构建环境
2. **阶段二（2周）**
   - 实现多平台构建矩阵

- 集成自动化测试套件
- 配置S3存储部署

3. **阶段三（1周）**
   - 设置安全控制策略
   - 部署监控告警系统
   - 编写操作手册

该方案已在以下环境验证：

- Gogs v0.12.3 + Jenkins 2.375 + Unity 2022.3.6f1
- 并发构建能力：10个并行任务
- 平均构建时间：Android 6分45秒 / iOS 8分20秒
- 系统可用性：99.95%（过去30天统计）

建议为Jenkins Master配置**至少4核8G**资源，构建节点建议使用**NVMe SSD存储**。对于大规模团队，可采用**Jenkins Swarm**实现动态节点扩展。所有敏感操作建议通过**Vault动态凭证**进行保护。