# Netkiller DevOps 手札

陈景峯 著

# Netkiller DevOps 手札

目录


- 自述
  - 1. 写给读者
  - 2. 作者简介
  - 3. 如何获得文档
  - 4. 打赏（Donations）
  - 5. 联系方式
- I. 软件项目管理工具
  - 1. Gitlab 项目管理
    - 1. GitLab
      - Docker 方式安装 Gitlab
      - docker-compose 安装
      - CentOS 8 Stream
      - Yum 安装 GitLab
        - GitLab Runner
      - 绑定SSL证书
    - 2. 用户管理
      - 创建用户
    - 3. 组管理
      - 创建组与项目
    - 4. 项目管理
    - 5. 分支管理
    - 6. Issue
      - Milestones 里程碑
      - Labels 标签
    - 7. 合并
      - 代码审查
    - 8. WebHook
    - 9. CI / CD
      - 远程服务器配置
      - 配置 CI / CD

        - 安装 GitLab Runner
        - 注册 gitlab-runner
    - Shell 执行器
        - 注册 Gitlab Runner 为 Shell 执行器
        - 生成 SSH 证书
        - 数据库环境
        - Java 环境
        - NodeJS
        - 远程执行 sudo 提示密码
    - tags 的使用方法
    - Docker 执行器
10. Pipeline 流水线
    - cache
    - before_script / after_script
    - stages
    - variables
        - 列出所有环境变量
    - only and except
    - 允许失败
    - 定义何时开始job
    - services
    - tags
    - rules 规则
    - Git submodule
11. 应用案例
    - Java
        - 使用 Docker 编译并收集构建物
        - Shell 执行器，远程部署物理机/虚拟机
        - Shell 执行器，远程部使用容器启动项目
        - Docker 执行器
    - Node
    - vue.js android
    - docker
12. FAQ
    - 查看日志
    - debug runner

gitolite 向 gitlab 迁移
修改主机名
ERROR: Uploading artifacts as "archive" to coordinator... too large archive

2. Jenkins
    1. 安装 Jenkins
        OSCM 一键安装
        Mac
        CentOS
        Ubuntu
        Docker
        Minikube
    2. 配置 Jenkins
    3. Jenkinsfile
        Jenkinsfile - Declarative Pipeline
            stages
            script
            junit
            withEnv
            parameters
            options
            triggers
            tools
            post
            when 条件判断
            抛出错误
            withCredentials
                token
            withMaven
            isUnix() 判断操作系统类型
            Jenkins pipeline 中使用 sshpass 实现 scp, ssh 远程运行
                后台运行
        Jenkinsfile - Scripted Pipeline
            git
            切换 JDK 版本

- groovy
- Groovy code
  - Groovy 函数
- Ansi Color
- 写文件操作
- modules 实现模块
- docker
- input
- if 条件判断
- Docker
- conditionalSteps
- nexus

设置环境变量

- 系统环境变量

agent

- label
- docker
  - 指定docker 镜像
  - args 参数
  - Docker outside of Docker (DooD)
  - 挂在宿主主机目录
  - 构建镜像
- Dockerfile

Steps

- parallel 平行执行
- echo
- catchError 捕获错误
- 睡眠
- 限制执行时间
- 时间截

版本控制

- checkout
- Git

节点与过程

- sh
- Windows 批处理脚本

        - 分配工作空间
        - node
    - 工作区
        - 变更目录
        - 判断文件是否存在
        - 分配工作区
        - 清理工作区
        - 递归删除目录
        - 写文件
        - 读文件
4. Jenkins Job DSL / Plugin
5. Jenkins Plugin
    - Blue Ocean
    - Locale Plugin (国际化插件)
    - github-plugin 插件
    - Docker
        - 设置 Docker 主机和代理
        - 持久化
    - JaCoCo
        - Pipeline
    - SSH Pipeline Steps
    - Rancher
    - Kubernetes 插件
        - Kubernetes
        - Kubernetes :: Pipeline :: Kubernetes Steps
        - Kubernetes Continuous Deploy
        - Kubernetes Cli
    - HTTP Request Plugin
    - Skip Certificate Check plugin
    - Android Sign Plugin
6. Jenkinsfile Pipeline Example
    - Maven 子模块范例
    - 使用指定镜像构建
    - 命令行制作 Docker 镜像
    - Yarn
    - Android

3. SonarQube
4. Phabricator - an open source, software engineering platform
5. Gerrit
6. TeamCity
7. TRAC
    1. Ubuntu 安装
        - source code
        - easy_install
        - Apache httpd
    2. CentOS 安装
        - trac.ini
        - standalone
        - Using Authentication
        - trac-admin
            - Permissions
            - Resync
    3. Project Environment
        - Sqlite
        - MySQL
        - Plugin
            - AccountManagerPlugin
            - Subtickets
    4. trac.ini
        - repository
        - attachment 附件配置
    5. trac-admin
        - adduser script
    6. Trac 项目管理
        - Administration
            - General
            - Ticket System
            - Version Control
        - Wiki
        - Timeline
        - Roadmap

Ticket
7. FAQ
TracError: Cannot load Python bindings for MySQL
8. Apache Bloodhound
8. Redmine
1. CentOS 安装
2. Redmine 运行
3. 插件
workflow
9. TUTOS
10. Open Source Requirements Management Tool
II. 软件版本控制
11. Git - Fast Version Control System
1. Repositories 仓库管理
1.1. initial setup
1.2. checkout
1.3. Creating and Commiting
1.4. Manager remote
1.5. Status
1.6. Diff
--name-only 仅显示文件名
1.7. Cloning
1.8. Push
1.9. Pull
1.10. fetch
1.11. Creating a Patch
1.12. reset
2. 分支管理
2.1. 查看本地分支
2.2. 创建分支
2.3. 删除分支
2.4. 切换分支
2.5. 重命名分支
2.6. git-show-branch - Show branches and their commits

3. Sharing Repositories with others
    3.1. Setting up a git server
    3.2. 修改 origin
    3.3. 删除 origin
4. 合并分支
    4.1. 合并分支
    4.2. rebase
    4.3. 合并分支解决冲突
5. Submodule 子模块
    5.1. 添加模块
    5.2. checkout 子模块
    5.3. 删除子模块
6. Git Large File Storage
    6.1. 安装 LFS 支持
    6.2. LFS lock
7. command
    7.1. hash-object
    7.2. git-add - Add file contents to the index
    7.3. git-status - Show the working tree status
    7.4. git-commit - Record changes to the repository
    7.5. git-show - Show various types of objects
    7.6. git-checkout - Checkout and switch to a branch
        checkout master
        checkout branch
    7.7. git config
    7.8. git log
8. git-daemon 服务器
    8.1. git-daemon - A really simple server for git repositories
    8.2. git-daemon-sysvinit
    8.3. inet.conf / xinetd 方式启动
    8.4. git-daemon-run
    8.5. Testing
9. git config
    9.1. core.sshCommand

9.2. fatal: The remote end hung up unexpectedly
9.3. 忽略 SSL 检查
10. git-svn - Bidirectional operation between a single Subversion branch and git
11. .gitignore
12. .gitattributes
12.1. SVN Keywords
13. gitolite - SSH-based gatekeeper for git repositories
13.1. gitolite-admin
gitolite.conf
staff
repo
14. Web Tools
14.1. viewgit
15. FAQ
15.1. 导出最后一次修改过的文件
15.2. 导出指定版本区间修改过的文件
15.3. 回撤提交
15.4. 每个项目一个证书
12. Subversion
1. Invoking the Server
1.1. Installing
Ubuntu
CentOS 5
classic Unix-like xinetd daemon
WebDav
项目目录结构
CentOS 6
1.2. standalone “daemon” process
starting subversion for debian/ubuntu
starting subversion daemon script for CentOS/Radhat
1.3. classic Unix-like inetd daemon
1.4. hooks
post-commit
1.5. WebDav

davfs2 - mount a WebDAV resource as a regular file system

2. repository 管理
    2.1. create repository
    2.2. user admin
    2.3. authz
    2.4. dump
3. 使用Subversion
    3.1. Initialized empty subversion repository for project
    3.2. ignore
    3.3. 关键字替换
    3.4. lock 加锁/ unlock 解锁
    3.5. import
    3.6. export 指定版本
    3.7. 修订版本关键字
    3.8. 恢复旧版本
4. branch
    4.1. create
    4.2. remove
    4.3. switch
    4.4. merge
    4.5. relocate
5. FAQ
    5.1. 递归添加文件
    5.2. 清除项目里的所有.svn目录
    5.3. color diff
    5.4. cvs2svn
    5.5. Macromedia Dreamweaver MX 2004 + WebDAV +Subversion
    5.6. 指定用户名与密码

13. cvs - Concurrent Versions System
    1. installation
        1.1. chroot
    2. cvs login | logout
    3. cvs import

4. cvs checkout
5. cvs update
6. cvs add
7. cvs status
8. cvs commit
9. cvs remove
10. cvs log
11. cvs annotate
12. cvs diff
13. rename file
14. revision
15. cvs export
16. cvs release
17. branch
    17.1. milestone
    17.2. patch branch
18. keywords

14. Miscellaneous
    1. 代码托管
        1.1. sourceforge.net
            http://netkiller.users.sourceforge.net/ 页面
        1.2. Google Code
        1.3. GitHub
            首次操作
            clone 已经存在的仓库
    2. GUI
        2.1. TortoiseSVN
        2.2. TortoiseGit
    3. Browser interface for CVS and SVN version control repositories

## 范例清单

2.1. Shell Docker 示例
12.1. authz

# Netkiller DevOps 手札

## Software engineering platform, Integrated SCM & Project Management, Version Control System，Continuous Integration & Delivery

## 《Netkiller DevOps 手札》视频教程

**Mr. Neo Chan, 陈景峯(BG7NYT)**

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

<netkiller@msn.com>

$Date: 2013-04-10 15:03:49 +0800 (Wed, 10 Apr 2013) $

电子书最近一次更新于 2021-09-20 11:37:31

版权 © 2009-2021 Neo Chan

版权声明

转载请与作者联系，转载时请务必标明文章原始出处和作者信息及本声明。

http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net
微信订阅号 netkiller-ebook
微信：13113668890 请注明“读者”
QQ：13721218 请注明“读者”

QQ群：128659835 请注明“读者”

[知乎专栏](#) | [多维度架构](#)

---

# 致读者

Netkiller 系列手札 已经被 Github 收录，并备份保存在北极地下250米深的代码库中，备份会保留1000年。

Preserving open source software for future generations

The world is powered by open source software. It is a hidden cornerstone of modern civilization, and the shared heritage of all humanity.

The GitHub Arctic Code Vault is a data repository preserved in the Arctic World Archive (AWA), a very-long-term archival facility 250 meters deep in the permafrost of an Arctic mountain.

We are collaborating with the Bodleian Library in Oxford, the Bibliotheca Alexandrina in Egypt, and Stanford Libraries in California to store copies of 17,000 of GitHub's most popular and most-depended-upon projects—open source's "greatest hits"—in their archives, in museum-quality cases, to preserve them for future generations.

https://archiveprogram.github.com/arctic-vault/

# 自述

《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是 nickname 从1999 开使用至今，“手札” 是札记，手册的含义。

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随着内容增加，不断修订，开始发布第一版，第二版.....

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用 Microsoft Office Word 慢慢随着文档尺寸增加 Word 开始表现出力不从心。

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习 Docbook SGML。使用Docbook写的第一部电子书是《Netkiller Postfix Integrated Solution》这是Netkiller 系列手札的原型。

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》 的 SGML文件已经不存在; Docbook SGML存在很多缺陷 UTF-8支持不好，转而使用Docbook XML.

目前技术书籍的价格一路飙升，动则￥80，￥100，少则￥50，￥60. 技术书籍有时效性，随着技术的革新或淘汰，大批书记成为废纸垃圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所有我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

# 1. 写给读者

*为什么写这篇文章*

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

*写作动力:*

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

*没有内容的章节:*

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本

章写其它的.
文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.
理论的东西,网上大把,我这里就不写了,需要可以去网上查.
我爱写错别字,还有一些是打错的,如果发现请指正.
文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

**写给读者**

至读者:

我不知道什么时候，我不再更新文档或者退出IT行业去从事其他工作，我必须给这些文档找一个归宿，让他能持续更新下去。

我想捐赠给某些基金会继续运转，或者建立一个团队维护它。

我用了20年时间坚持不停地写作，持续更新，才有今天你看到的《Netkiller 手扎》系列文档，在中国能坚持20年，同时没有任何收益的技术类文档，是非常不容易的。

有很多时候想放弃，看到外国读者的支持与国内社区的影响，我坚持了下来。

中国开源事业需要各位参与，不要成为局外人，不要让外国人说：中国对开源索取无度，贡献却微乎其微。

我们参与内核的开发还比较遥远，但是进个人能力，写一些文档还是可能的。

**系列文档**

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札
Netkiller Developer 手札
Netkiller PHP 手札
Netkiller Python 手札
Netkiller Testing 手札
Netkiller Cryptography 手札

Netkiller Linux 手札
Netkiller FreeBSD 手札
Netkiller Shell 手札
Netkiller Security 手札
Netkiller Web 手札
Netkiller Monitoring 手札
Netkiller Storage 手札
Netkiller Mail 手札
Netkiller Docbook 手札
Netkiller Version 手札
Netkiller Database 手札
Netkiller PostgreSQL 手札
Netkiller MySQL 手札
Netkiller NoSQL 手札
Netkiller LDAP 手札
Netkiller Network 手札
Netkiller Cisco IOS 手札
Netkiller H3C 手札
Netkiller Multimedia 手札
Netkiller Management 手札
Netkiller Spring 手札
Netkiller Perl 手札
Netkiller Amateur Radio 手札

# 2. 作者简介

陈景峯 (ㄔㄣˇ ㄐㄧㄥˇ ㄈㄥ)

Nickname： netkiller | English name: Neo chen | Nippon name: ちんけいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: BG7NYT | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工, Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

**成长阶段**

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Trubo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作 UCDOS 中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

## 职业生涯

2001 年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002 年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003 年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入分布式计算团队,目前成绩，工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL，相隔几年发现PHP进步很大。在前台展现方面无人能敌，于是便前台使用PHP，后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机，休息了4个月（其实是找不到工作），关外很难上439.460中继，搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,2009-6-13学车，年底拿到C1驾照

2010 对电子打击乐产生兴趣，计划学习爵士鼓。由于我对Linux热爱，我轻松的接管了公司的运维部，然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜，我们用买服务器纸箱的钱改善伙食。我将40多台服务器安装BOINC做压力测试，获得了中国第二的名次。

2011 平凡的一年，户外运动停止，电台很少开，中继很少上，摄影主要是拍女儿与家人，年末买了一辆山地车

2012 对油笔画产生了兴趣，活动基本是骑行银湖山绿道，

2013 开始学习民谣吉他，同时对电吉他也极有兴趣；最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola

MOTOTRBO XIR P8668，Netkiller 系列手札从Sourceforge向Github迁移；年底对MYSQL UDF，Engine与PHP扩展开发产生很浓的兴趣，拾起遗忘10+年的C，写了几个mysql扩展（图片处理，fifo管道与ZeroMQ），10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴，由于这家钢琴是合成器电钢，里面有打击乐，我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游，对中国道教文化与音乐产生了兴趣，10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣（https://github.com/SheetMusic/Piano），给女儿做了几首钢琴伴奏曲，MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练，经过反复琢磨加上之前学钢琴的乐理知识，终于在02号晚上，打出了简单的基本节奏，迈出了第一步。

2016 对弓箭（复合弓）产生兴趣，无奈天朝法律法规不让玩。每周游泳轻松1500米无压力，年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台，连接Xbox打游戏爽翻了，入Kindle电子书，计划学习蝶泳，果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦，丢弃了多年攒下的家底。11 月开始玩 MMDVM，使用 Yaesu FT-7800 发射，连接MMDVM中继板，树莓派，覆盖深圳湾，散步骑车通联两不误。

2019 卖了常德的房子，住了5次院，哮喘反复发作，决定停止电子书更新，兴趣转到知乎，B站

2020 准备找工作

职业生涯路上继续打怪升级

# 3. 如何获得文档

**下载 Netkiller 手札 (epub,kindle,chm,pdf)**
EPUB https://github.com/netkiller/netkiller.github.io/tree/master/download/epub
MOBI https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi
PDF https://github.com/netkiller/netkiller.github.io/tree/master/download/pdf
CHM https://github.com/netkiller/netkiller.github.io/tree/master/download/chm

**通过 GIT 镜像整个网站**

https://github.com/netkiller/netkiller.github.com.git

$ git clone https://github.com/netkiller/netkiller.github.com.git

**镜像下载**

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

**Yum 下载文档**

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

http://netkiller.sourceforge.net/pub/repo/

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
```

```
name=Netkiller Free Books
baseurl=http://netkiller.sourceforge.net/pub/repo/
enabled=1
gpgcheck=0
gpgkey=
EOF
```

查找包

```
# yum search netkiller

netkiller-centos.x86_64 : Netkiller centos Cookbook
netkiller-cryptography.x86_64 : Netkiller cryptography Cookbook
netkiller-docbook.x86_64 : Netkiller docbook Cookbook
netkiller-linux.x86_64 : Netkiller linux Cookbook
netkiller-mysql.x86_64 : Netkiller mysql Cookbook
netkiller-php.x86_64 : Netkiller php Cookbook
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook
netkiller-python.x86_64 : Netkiller python Cookbook
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```

# 4. 打赏（Donations）

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

**银行(Bank)**

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

**微信（Wechat）**

**支付宝（Alipay）**

**PayPal Donations**

https://www.paypal.me/netkiller

# 5. 联系方式

主站 http://www.netkiller.cn/

备用 http://netkiller.github.io/

繁体网站 http://netkiller.sourceforge.net/

联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222

注：请不要问我安装问题！

博客 **Blogger**

知乎专栏 https://zhuanlan.zhihu.com/netkiller

LinkedIn: http://cn.linkedin.com/in/netkiller

OSChina: http://my.oschina.net/neochen/

Facebook: https://www.facebook.com/bg7nyt

Flickr: http://www.flickr.com/photos/bg7nyt/

Disqus: http://disqus.com/netkiller/

solidot: http://solidot.org/~netkiller/

SegmentFault: https://segmentfault.com/u/netkiller

Reddit: https://www.reddit.com/user/netkiller/

Digg: http://www.digg.com/netkiller

Twitter: http://twitter.com/bg7nyt

weibo: http://weibo.com/bg7nyt

**Xbox club**

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部 netkiller 欢迎加入。

**Radio**

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

**MMDVM Hotspot:**

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

# 部分 I. 软件项目管理工具

## project management tool

## 1. Nexus Repository OSS

https://www.sonatype.com/download-oss-sonatype

```
wget https://www.sonatype.com/oss-thank-you-tar.gz
```

### 1.1. 安装 Nexus

#### Docker

```
docker run -d -p 8081:8081 --restart=always --name nexus sonatype/nexus3
```

### 1.2. Nexus UI

http://localhost:8081/ 登陆用户名 admin 默认密码 admin123

### 1.3. maven 设置

maven在settings.xml中配置如下，下次maven就会通过访问电脑上的私服来获取jar包

```
<mirrors>
    <mirror>
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-public/</url>
    </mirror>
```

```
</mirrors>
```

## 1.4. Node.js

输入命令登陆远程仓库

```
npm login --registry=http://nexus.netkiller.cn/repository/npm/
```

在项目中输入

```
npm pack
```

上传

```
npm publish --registry=http://nexus.netkiller.cn/repository/npm/
```

## 1.5. Ruby

安装 nexus 包

```
$ gem install nexus
```

打包

```
gem build project.gemspec
```

上传，系统会提示上传URL

```
gem nexus project-1.0.0.gem
```

# 第 1 章 Gitlab 项目管理

实施DEVOPS首先我们要有一个项目管理工具。

我建议使用 Gitlab，早年我倾向使用Trac，但Trac项目一直处于半死不活状态，目前来看Trac 对于 Ticket管理强于Gitlab，但Gitlab发展的很快，我们可以看到最近的一次升级中Issue 加入了 Due date 选项。Gitlab 已经有风投介入，企业化运作，良性发展，未来会超越Redmine等项目管理软件，成为主流。所以我在工具篇采用Gitlab，BTW 我没有使用 Redmine，我认为 Redmine 的发展方向更接近传统项目管理思维。

软件项目管管理，我需要那些功能，Ticket/Issue管理、里程碑管理、内容管理Wiki、版本管理、合并分支、代码审查等等

关于Gitlib的安装配置请参考
http://www.netkiller.cn/project/project/gitlab/index.html

## 1. GitLab

https://github.com/gitlabhq

GitLab是一个利用 Ruby on Rails 开发的开源应用程序，实现一个自托管的Git项目仓库，可通过Web界面进行访问公开的或者私人项目。

它拥有与Github类似的功能，能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序(Wall)进行交流。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

GitLab 5.0以前版本要求服务器端采用 Gitolite 搭建，5.0版本以后不再使用 Gitolite ，采用自己开发的 gitlab-shell 来实现。如果你觉得安装麻烦可以使用 GitLab Installers 一键安装程序。

**Docker 方式安装 Gitlab**

```
export GITLAB_HOME=/srv/gitlab
```

```
sudo docker run --detach \
  --hostname gitlab.example.com \
  --publish 443:443 --publish 80:80 --publish 22:22 \
  --name gitlab \
  --restart always \
  --volume $GITLAB_HOME/config:/etc/gitlab \
  --volume $GITLAB_HOME/logs:/var/log/gitlab \
  --volume $GITLAB_HOME/data:/var/opt/gitlab \
  gitlab/gitlab-ce:latest
```

配置对外url，域名或者ip，公网能访问即可

```
vim /mnt/gitlab/etc/gitlab.rb
添加一下配置:
external_url    'http://127.0.0.1'  (你的域名或者ip地址)
```

配置邮箱

```
vim /mnt/gitlab/etc/gitlab.rb
gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtp.qq.com"
gitlab_rails['smtp_port'] = 465
gitlab_rails['smtp_user_name'] = "13721218@qq.com"    (替换成自己
的QQ邮箱)
gitlab_rails['smtp_password'] = "xxxxx"
gitlab_rails['smtp_domain'] = "smtp.qq.com"
gitlab_rails['smtp_authentication'] = "login"
gitlab_rails['smtp_enable_starttls_auto'] = true
gitlab_rails['smtp_tls'] = true
```

```
gitlab_rails['gitlab_email_from'] = '13721218@qq.com'  (替换成自己的QQ邮箱，且与smtp_user_name一致)
```

重新启动gitlab

```
docker restart gitlab-ce
sudo docker logs -f gitlab
```

**docker-compose 安装**

安装 gitlab

```
version: '3.9'
services:
    gitlab:
      image: 'gitlab/gitlab-ce:latest'
      container_name: "gitlab"
      restart: unless-stopped
      privileged: true
      hostname: 'gitlab.example.com'
      environment:
        TZ: 'Asia/Shanghai'
        GITLAB_OMNIBUS_CONFIG: |
          external_url 'https://gitlab.example.com'
          gitlab_rails['time_zone'] = 'Asia/Shanghai'
          gitlab_rails['smtp_enable'] = true
          gitlab_rails['smtp_address'] = "smtp.netkiller.cn"
          gitlab_rails['smtp_port'] = 465
          gitlab_rails['smtp_user_name'] =
"netkiller@netkiller.cn"
          gitlab_rails['smtp_password'] = "******"
          gitlab_rails['smtp_domain'] = "netkiller.cn"
          gitlab_rails['smtp_authentication'] = "login"
          gitlab_rails['smtp_enable_starttls_auto'] = true
          gitlab_rails['smtp_tls'] = true
          gitlab_rails['gitlab_email_from'] =
```

```
'netkiller@netkiller.cn'
          gitlab_rails['gitlab_shell_ssh_port'] = 22
      ports:
        - '80:80'
        - '443:443'
        - '22:22'
      volumes:
        - /opt/gitlab/config:/etc/gitlab
        - /opt/gitlab/logs:/var/log/gitlab
        - /opt/gitlab/data:/var/opt/gitlab
```

安装 gitlab-runner

```
version: '3.9'
services:
  gitlab-runner:
    image: gitlab/gitlab-runner:alpine
    restart: unless-stopped
    depends_on:
      - gitlab
    privileged: true
    volumes:
      - ./config/gitlab-runner:/etc/gitlab-runner
      - /var/run/docker.sock:/var/run/docker.sock
      - /bin/docker:/bin/docker
```

启动 Gitlab runner

```
sudo chmod 666 /var/run/docker.sock
sudo usermod -aG docker $USER
docker-compose up -d
```

注册 gitlab-runner 到 Gitlab

```
docker exec -it gitlab-runner gitlab-runner register
```

## CentOS 8 Stream

```
dnf install langpacks-en glibc-all-langpacks -y
localectl set-locale LANG=en_US.UTF-8

sudo systemctl status firewalld
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo systemctl reload firewalld

sudo dnf install postfix
sudo systemctl enable postfix
sudo systemctl start postfix


curl -s
https://packages.gitlab.com/install/repositories/gitlab/gitlab-
ce/script.rpm.sh | bash

EXTERNAL_URL="http://gitlab.example.com"

export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
export LC_CTYPE=UTF-8

dnf install -y gitlab-ce

cp /etc/gitlab/gitlab.rb{,.original}

gitlab-ctl reconfigure
```

查看 root 密码

```
[root@localhost ~]# cat  /etc/gitlab/initial_root_password
# WARNING: This value is valid only in the following conditions
#          1. If provided manually (either via
`GITLAB_ROOT_PASSWORD` environment variable or via
`gitlab_rails['initial_root_password']` setting in `gitlab.rb`,
it was provided before database was seeded for the first time
(usually, the first reconfigure run).
#          2. Password hasn't been changed manually, either via
UI or via command line.
#
#          If the password shown here doesn't work, you must
reset the admin password following
https://docs.gitlab.com/ee/security/reset_user_password.html#res
et-your-root-password.

Password: dpzQFzltaq0BhAwDnugMf6MCFbvItXDvC+RcuN9R+wg=

# NOTE: This file will be automatically deleted in the first
reconfigure run after 24 hours.
```

GitLab Runner

```
curl -sL
"https://packages.gitlab.com/install/repositories/runner/gitlab-
runner/script.rpm.sh" | sudo bash
dnf install gitlab-runner
```

配置文件 /etc/gitlab-runner/config.toml

```
[root@localhost ~]# systemctl restart gitlab-runner
```

**Yum 安装 GitLab**

```
yum localinstall -y https://downloads-
packages.s3.amazonaws.com/centos-6.6/gitlab-ce-7.10.0~omnibus.2-
1.x86_64.rpm

gitlab-ctl reconfigure

cp /etc/gitlab/gitlab.rb{,.original}
```

停止 GitLab 服务

```
# gitlab-ctl stop
ok: down: logrotate: 1s, normally up
ok: down: nginx: 0s, normally up
ok: down: postgresql: 0s, normally up
ok: down: redis: 0s, normally up
ok: down: sidekiq: 1s, normally up
ok: down: unicorn: 0s, normally up
```

启动 GitLab 服务

```
# gitlab-ctl start
ok: run: logrotate: (pid 3908) 0s
ok: run: nginx: (pid 3911) 1s
ok: run: postgresql: (pid 3921) 0s
ok: run: redis: (pid 3929) 1s
ok: run: sidekiq: (pid 3933) 0s
ok: run: unicorn: (pid 3936) 1s
```

配置gitlab

```
# vim /etc/gitlab/gitlab.rb
external_url 'http://gitlab.example.com'
```

SMTP配置

```
gitlab_rails['gitlab_email_enabled'] = true
gitlab_rails['gitlab_email_from'] = 'openunix@163.com'
gitlab_rails['gitlab_email_display_name'] = 'Neo'
gitlab_rails['gitlab_email_reply_to'] = 'noreply@example.com'

gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtp.163.com"
gitlab_rails['smtp_user_name'] = "openunix@163.com"
gitlab_rails['smtp_password'] = "password"
gitlab_rails['smtp_domain'] = "163.com"
gitlab_rails['smtp_authentication'] = "login"
```

任何配置文件变化都需要运行 # gitlab-ctl reconfigure

WEB管理员

```
# Username: root
# Password: 5iveL!fe
```

**GitLab Runner**

```
curl -L
https://packages.gitlab.com/install/repositories/runner/gitlab-
ci-multi-runner/script.rpm.sh | sudo bash
sudo yum install gitlab-ci-multi-runner
```

进入 CI 配置页面
http://git.netkiller.cn/netkiller.cn/www.netkiller.cn/settings/ci_cd

Specific Runners 你将看到 CI 的URL和他的Token

Specify the following URL during the Runner setup:
http://git.netkiller.cn/ci

Use the following registration token during setup:
wRoz1Y_6CXpNh2JbxN_s

现在回到 GitLab Runner

```
# gitlab-ci-multi-runner register
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g.
https://gitlab.com/):
http://git.netkiller.cn/ci
Please enter the gitlab-ci token for this runner:
wRoz1Y_6CXpNh2JbxN_s
Please enter the gitlab-ci description for this runner:
[iZ62yln3rjjZ]: gitlab-ci-1
Please enter the gitlab-ci tags for this runner (comma
separated):
test
Whether to run untagged builds [true/false]:
[false]:
Registering runner... succeeded
runner=wRoz1Y_6
Please enter the executor: docker, docker-ssh, shell, ssh,
virtualbox, docker+machine, docker-ssh+machine, kubernetes,
parallels:
shell
Runner registered successfully. Feel free to start it, but if
it's running already the config should be automatically
reloaded!
```

回到 Gitlab 页你将看到 Pending 状态变成 Running 状态

升级 GitLab Runner

```
yum install gitlab-ci-multi-runner
```

## 绑定SSL证书

编辑 /etc/gitlab/gitlab.rb 文件

```
```

```
external_url 'https://git.netkiller.cn'

nginx['enable'] = true
nginx['redirect_http_to_https'] = true
nginx['ssl_certificate'] =
"/etc/gitlab/ssl/git.netkiller.cn.crt"
nginx['ssl_certificate_key'] =
"/etc/gitlab/ssl/git.netkiller.cn.key"
nginx['listen_https'] = true
nginx['http2_enabled'] = true
```

## 2. 用户管理

初始化GitLab，进入Admin area，单击左侧菜单Users，在这里为gitlab添加用户

### 创建用户

过程 1.1. 企业内部使用的 Gitlab 初始化

1. 关闭在线用户注册
2. Step 3.
    a. Substep a.
    b. Substep b.

## 3. 组管理

初始化GitLab组，我比较喜欢使用“域名”作为组名，例如example.com

### 创建组与项目

过程 1.2. Gitlab 初始化 - 创建组

1. 点击 New Group 按钮新建一个组，我习惯每个域一个组，所以我使用 netkiller.cn 作为组名称

2. 输入 netkiller.cn 然后单击 Create group

3. 组创建完毕

创建组后接下来创建项目

过程 1.3. Gitlab 初始化 - 创建项目

1. 单击 New Project 创建项目

2. 输入 www.netkiller.cn 并点击 Create project 按钮创建项目

3. 项目创建完毕

W

www.netkiller.cn

Star 0

HTTP http://git.netkiller.cn/netkiller.cn/www.netki

Global

## 4. 项目管理

创建项目，我通常会在组下面创建项目，每个域名对应一个项目，例如www.example.com,images.example.com

版本库URL如下

```
http: http://192.168.0.1/example.com/www.example.com.git
ssh: git@192.168.0.1:example.com/www.example.com.git
```

## 5. 分支管理

起初我们应对并行开发在Subversion上创建分支，每个任务一个分支，每个Bug一个分支，完成任务或修复bug后合并到开发分支(development)内部测试，然后再进入测试分支(testing)提交给测试组，测试组完成测试，最后进入主干(trunk)。对于Subverion来说每一个分支都是一份拷贝，SVN版本库膨胀的非常快。

Git 解决了Svn 先天不足的分支管理功能，分支在GIT类似快照，同时GIT还提供了 pull request 功能。

我们怎样使用git 的分支功能呢？ 首先我们不再为每个任务创建一个分支，将任务分支放在用户自己的仓库下面，通过 pull request 合并，同时合并过程顺便code review。

master：是主干，只有开发部主管/经理级别拥有权限，只能合并来自testing的代码

testing: 测试分支，测试部拥有权限，此分支不能修改，只能从开发分支合并代码。

development：开发组的分支，Team拥有修改权限，可以合并，可以接受pull request, 可以提交代码

tag 是 Release 本版，开发部主管/经理拥有权限

分支的权限管理：

master: 保护

testing：保护

development：保护

过程 1.4. Gitlab 分支应用 - 创建分支

1. 首先，点击左侧 Commits 按钮，然后点击 Branches 按钮进入分支管理

2. 点击 New branch 创建分支

在 Branch name 中输入分支名称，然后点击 Create branch 创建分支

3. 分支已经创建

重复上面步骤，完成development分支的创建。

保护分支：锁定分支，只允允许合并，不允许提交

过程 1.5. 保护分支

1. master

   testing

2. Step 2.

   a.

   b. Substep b.

## 6. Issue

Issues 任务

**Milestones 里程碑**

敏捷开发中可以每周一个里程碑，或者每个月一个里程碑。

**Labels 标签**

参考 github 设置

bug Something isn't working
documentation Improvements or additions to documentation
duplicate This issue or pull request already exists
enhancement New feature or request
good first issue Good for newcomers
help wanted Extra attention is needed
invalid This doesn't seem right
question Further information is requested
wontfix This will not be worked on

通常定义四个状态，开发，测试，升级，完成

## 7. 合并

### 代码审查

# 8. WebHook

## 9. CI / CD

https://gitlab.com/gitlab-examples

```
Gitlab(仓库) -> Gitlab Runner（持续集成/部署） -> Remote host（远程部署主机）
```

### 远程服务器配置

为远程服务器创建 www 用户，我们将使用该用户远程部署，远程启动程序。

```
[root@netkiller ~]# groupadd -g 80 www
[root@netkiller ~]# adduser -o --uid 80 --gid 80 -G wheel -c "Web Application" www
[root@netkiller ~]# id www
uid=80(www) gid=80(www) groups=80(www),10(wheel)
[root@netkiller ~]# PASSWORD=$(cat /dev/urandom | tr -dc [:alnum:] | head -c 32)
[root@netkiller ~]# echo www:${PASSWORD} | chpasswd
[root@netkiller ~]# echo "www password: ${PASSWORD}"
www password: 0Uz1heY9v9KJyRKbvTi0VlAzfEoFW9GH
```

```
mkdir -p /opt/netkiller.cn/www.netkiller.cn
chown www:www -R /opt/netkiller.cn
```

### 配置 CI / CD

进入项目设置界面，点击 Settings，再点击 CI / CD

点击 Expand 按钮 展开 Runners

这时可以看到 Set up a specific Runner manually, 后面会用到 http://192.168.1.96/ 和 zASzWwffenos6Jbbfsgu

**安装 GitLab Runner**

**Install GitLab Runner**

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.rpm.sh" |
sudo bash
dnf install gitlab-runner

cp /etc/gitlab-runner/config.toml{,.original}

systemctl enable gitlab-runner
```

**注册 gitlab-runner**

使用 SSH 登录 Gitlab runner 服务器，运行 gitlab-runner register

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=92925
revision=ac2a293c version=11.11.2
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://192.168.1.96/
Please enter the gitlab-ci token for this runner:
zASzWwffenos6Jbbfsgu
Please enter the gitlab-ci description for this runner:
[localhost.localdomain]:
Please enter the gitlab-ci tags for this runner (comma separated):
```

```
Registering runner... succeeded                     runner=zASzWwff
Please enter the executor: docker, docker-ssh, shell, ssh, docker-ssh+machine, parallels,
virtualbox, docker+machine, kubernetes:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

返回 gitlab 查看注册状态

## Shell 执行器

### Registering Runners

**注册 Gitlab Runner 为 Shell 执行器**

```
[root@gitlab ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=1020084
revision=c1edb478 version=14.0.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
DyKdKyaJaq5KN-irgNGz
Enter a description for the runner:
[gitlab]:
Enter tags for the runner (comma-separated):

Registering runner... succeeded                     runner=DyKdKyaJ
Enter an executor: parallels, virtualbox, docker+machine, custom, docker, docker-ssh, shell,
ssh, docker-ssh+machine, kubernetes:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

/etc/gitlab-runner/config.toml 配置文件

```
[root@gitlab ~]# cat /etc/gitlab-runner/config.toml
concurrent = 1
```

```
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "gitlab"
  url = "http://git.netkiller.cn/"
  token = "kVkzjDM74xZUN-aKbdPp"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

#### 生成 SSH 证书

持续集成和部署运行在 gitlab-runner 用户下，切换到 gitlab-runner 用户

```
[root@gitlab ~]# su - gitlab-runner
Last login: Mon Jul 19 19:01:37 CST 2021
```

生成 SSH 证书

```
[gitlab-runner@gitlab ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/gitlab-runner/.ssh/id_rsa):
Created directory '/home/gitlab-runner/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gitlab-runner/.ssh/id_rsa.
Your public key has been saved in /home/gitlab-runner/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:l90LYBeSF9l9JHXJUHeO+IyvscCziz4C8vFNpJoKEjo gitlab-runner@gitlab
The key's randomart image is:
+---[RSA 3072]----+
|          ..o===B|
|          ..oo.**|
|          o.o . o|
|        .. = =   |
|.      oS o + +  |
|... o . .o   o . |
|E  o * o  + . o  |
|.o  + o o. + +   |
|  ..   oo.o.o    |
+----[SHA256]-----+
[gitlab-runner@gitlab ~]$
```

正常情况下，当我们链接一个 SSH 主机，会让我们输入 yes 确认继续链接。

```
[gitlab-runner@gitlab ~]$ ssh www@192.168.40.10
The authenticity of host '192.168.40.10 (192.168.40.10)' can't be established.
ECDSA key fingerprint is SHA256:xmFF266MPdXhnlAljS+QWhQsw6jOw1sOwQXRr/PHi2w.
```

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

配置 SSH

```
[gitlab-runner@gitlab ~]$ cat > ~/.ssh/config <<'EOF'
Host *
        ServerAliveInterval=30
        StrictHostKeyChecking no
        UserKnownHostsFile=/dev/null
EOF

chmod 600 -R ~/.ssh/config
```

授权远程执行 Shell

```
[gitlab-runner@gitlab ~]$ ssh-copy-id www@www.netkiller.cn
```

#### 数据库环境

在构建过程中，我们需要备份数据库/同步数据库，下面安装了一些所需的工具

```
[root@localhost ~]# dnf install -y mysql
```

设置数据库备份账号和密码，这里偷懒使用了 root 账号，生产环境请创建专用的备份账号。

```
[root@localhost ~]# su - gitlab-runner
Last login: Wed Sep  1 19:17:48 CST 2021
[gitlab-runner@localhost ~]$ vim ~/.my.cnf
[gitlab-runner@localhost ~]$ cat ~/.my.cnf
[mysql]
user=root
password=test

[mysqldump]
user=root
password=test
```

测试数据库是否畅通

```
[gitlab-runner@localhost ~]$ mysql -h mysql.netkiller.cn
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 37602
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

**Java 环境**

JRE：java-11-openjdk

JDK：java-11-openjdk-devel

```
[root@gitlab ~]# dnf install -y java-11-openjdk java-11-openjdk-devel
[root@gitlab ~]# dnf install -y maven
```

修改 Maven 镜像路

```
[root@gitlab ~]# vim /etc/maven/settings.xml
  <mirrors>
    <mirror>
      <id>aliyun</id>
      <name>aliyun maven</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
```

如果需要安装最新版本 maven 使用下面脚本。

```
#!/bin/bash

cd /usr/local/src/
wget https://mirrors.bfsu.edu.cn/apache/maven/maven-3/3.8.2/binaries/apache-maven-3.8.2-
bin.tar.gz
tar zxf apache-maven-3.8.2-bin.tar.gz
mv apache-maven-3.8.2 /srv/
rm -f /srv/apache-maven
ln -s /srv/apache-maven-3.8.2 /srv/apache-maven

alternatives --install /usr/local/bin/mvn apache-maven-3.8.2 /srv/apache-maven-3.8.2/bin/mvn 0
```

```
[root@localhost src]# mvn -v
Apache Maven 3.8.2 (ea98e05a04480131370aa0c110b8c54cf726c06f)
Maven home: /srv/apache-maven-3.8.2
Java version: 17-ea, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-17-openjdk-17.0.0.0.26-
0.2.ea.el8.x86_64
```

```
Default locale: en_US, platform encoding: ANSI_X3.4-1968
OS name: "linux", version: "4.18.0-338.el8.x86_64", arch: "amd64", family: "unix"
```

apache-maven-3.8.2 配置

```
[root@localhost ~]# vim /srv/apache-maven/conf/settings.xml
  <mirrors>
    <!-- mirror
     | Specifies a repository mirror site to use instead of a given repository. The repository
that
     | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are
used
     | for inheritance and direct lookup purposes, and must be unique across the set of
mirrors.
     |
    <mirror>
      <id>mirrorId</id>
      <mirrorOf>repositoryId</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
      <url>http://my.repository.com/repo/path</url>
    </mirror>
     -->
    <mirror>
      <id>maven-default-http-blocker</id>
      <mirrorOf>external:http:*</mirrorOf>
      <name>Pseudo repository to mirror external repositories initially using HTTP.</name>
      <url>http://0.0.0.0/</url>
      <blocked>true</blocked>
    </mirror>
  </mirrors>
```

apache-maven-3.8.2 默认会阻止其他镜像，需要会去掉 maven-default-http-blocker 配置

切换到 gitlab-runner 用户，随便运行一下 mvn 命令，这样就会产生 ~/.m2 文件夹

```
[root@gitlab ~]# su - gitlab-runner
[gitlab-runner@gitlab ~]$ mvn -v
```

**NodeJS**

```
[root@netkiller ~]# dnf install -y nodejs
```

安装 cnpm

```
[root@netkiller ~]# npm config set registry https://registry.npm.taobao.org
[root@netkiller ~]# npm config get registry
https://registry.npm.taobao.org/
[root@netkiller ~]# npm install -g cnpm
```

yarn

```
[root@netkiller ~]# curl -sL https://dl.yarnpkg.com/rpm/yarn.repo -o /etc/yum.repos.d/yarn.repo
[root@netkiller ~]# dnf install -y yarn
```

pm2 进程管理

```
[root@netkiller ~]# npm install -g pm2
```

设置 pm2 启动开启

```
[root@netkiller ~]# pm2 startup
[root@netkiller ~]# pm2 save --force
[root@netkiller ~]# systemctl enable pm2-root
[root@netkiller ~]# systemctl start pm2-root
[root@netkiller ~]# systemctl status pm2-root
```

**远程执行 sudo 提示密码**

```
[gitlab-runner@gitlab api.sfzito.com]$ ssh www@192.168.40.10 "sudo ls"
Warning: Permanently added '192.168.40.10' (ECDSA) to the list of known hosts.
sudo: a terminal is required to read the password; either use the -S option to read from
standard input or configure an askpass helper
```

解决方案一

```
ssh -t www@www.netkiller.cn "echo <yourpassword> |sudo -S <yourcommand>"
```

解决方案二

```
cat > /etc/sudoers.d/www <<-EOF
www    ALL=(ALL)    NOPASSWD: ALL
EOF
```

## tags 的使用方法

tags 是给 Gitlab Runner 打个标签，我的用法是多次注册，例如 shell 执行器的标签是 shell, Docker 执行器的标签是 docker，这样便可以在.gitlab-ci.yml文件中来选择使用那个执行器来触发操作。

下面是 Shell 执行器

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=268363
revision=58ba2b95 version=14.2.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
k_SsvMQV397gAMaP_q1v
Enter a description for the runner:
[localhost.localdomain]: development
Enter tags for the runner (comma-separated):
shell
Registering runner... succeeded                     runner=k_SsvMQV
Enter an executor: docker, docker-ssh, virtualbox, docker-ssh+machine, kubernetes, custom,
parallels, shell, ssh, docker+machine:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

下面是 Docker 执行器

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=268397
revision=58ba2b95 version=14.2.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
k_SsvMQV397gAMaP_q1v
Enter a description for the runner:
[localhost.localdomain]: development
Enter tags for the runner (comma-separated):
docker
Registering runner... succeeded                     runner=k_SsvMQV
Enter an executor: custom, docker-ssh, parallels, shell, ssh, docker-ssh+machine, docker,
virtualbox, docker+machine, kubernetes:
docker
Enter the default Docker image (for example, ruby:2.6):
maven:latest
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

注册后的效果

```
[root@localhost ~]# cat /etc/gitlab-runner/config.toml
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "development"
  url = "http://git.netkiller.cn/"
  token = "EztTBypKRW5ibtC5rs2h"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]

[[runners]]
  name = "development"
  url = "http://git.netkiller.cn/"
  token = "51948sQbQsXGV-RxFMty"
  executor = "docker"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "maven:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache"]
    shm_size = 0
```

## Docker 执行器

注册 Docker 执行器

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=268397
revision=58ba2b95 version=14.2.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
k_SsvMQV397gAMaP_q1v
Enter a description for the runner:
[localhost.localdomain]: development
Enter tags for the runner (comma-separated):
docker
Registering runner... succeeded                     runner=k_SsvMQV
Enter an executor: custom, docker-ssh, parallels, shell, ssh, docker-ssh+machine, docker,
virtualbox, docker+machine, kubernetes:
docker
Enter the default Docker image (for example, ruby:2.6):
maven:latest
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

配置缓存

```
[root@localhost ~]# cat /etc/gitlab-runner/config.toml
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "development"
  url = "http://192.168.30.5/"
  token = "EztTBypKRW5ibtC5rs2h"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]

[[runners]]
  name = "development"
  url = "http://192.168.30.5/"
  token = "GP-ozvd6uw2nDxyRohZ-"
  executor = "docker"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "maven:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
```

```
    volumes = ["/cache","/root/.m2"]
    pull_policy = ["never"]
    shm_size = 0
```

volumes = ["/cache","/root/.m2"] 将 Maven 仓库缓存

.gitlab-ci.yaml 编排脚本

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  tags:
    - docker
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  image: maven:3.8.2-openjdk-17
  stage: test
  variables:
    GIT_STRATEGY: none
  tags:
    - docker
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
    DOCKER_HOST: unix:///var/run/docker.sock mvn clean install docker:build
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    # - mvn docker:push
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
```

```
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart"
```

## 10. Pipeline 流水线

### cache

Java 缓存设置

```
image: maven:3.5.0-jdk-8

variables:
  MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

cache:
  paths:
    - .m2/repository/
    - target/

stages:
  - build
  - test
  - package

build:
  stage: build
  script: mvn compile

unittest:
  stage: test
  script: mvn test


package:
  stage: package
  script: mvn package
  artifacts:
    paths:
      - target/java-project-0.0.1-SNAPSHOT.jar
```

Node 缓存设置

```
cache:
  paths:
```

```
    - node_modules
    - dist

# variables:
  # GIT_STRATEGY: clone
  # GIT_STRATEGY: fetch
  # GIT_CHECKOUT: "false"

stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  only:
    - master
    - testing
    - development
  script:
    - echo "Compiling the code..."
    # - cnpm cache verify
    - cnpm install
    - cnpm run build:stage
    # - cnpm run build:prod
    - echo "Compile complete."

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - master
    - testing
    - development
  script:
    - echo "Running unit tests..."
    - sed -i 's#192.168.20.180#192.168.30.4#g' dist/umi.*.js
    - ls dist/*
    # - rm -rf *.tar.gz
    # - tar zcvf www.netkiller.cn.$(date -u +%Y-%m-%d.%H%M%S).tar.gz
dist
    # - ls *.tar.gz
    - echo "Test complete."
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - dist/*
      # - ./*.tar.gz
```

```
deploy-test-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - testing
    - development
  script:
    - echo "Deploying application..."
    - rsync -auzv dist/* www@192.168.30.10:/opt/www.netkiller.cn/
    - echo "Application successfully deployed."

deploy-prod-job:
  stage: deploy
  only:
    - master
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/*
www@192.168.30.10:/opt/www.netkiller.cn/
    - echo "Application successfully deployed."
```

**before_script / after_script**

before_script: 在 pipeline 运行前执行脚本

after_script: 在 pipeline 完成之后执行脚本

```
cache:
  paths:
    - node_modules
    - dist

before_script:
  - cnpm install

stages:
  - build
  - test
  - deploy

build-dev-job:
  stage: build
  only:
    - development
```

```
  script:
    - npm run build:dev

build-test-job:
  stage: build
  only:
    - testing
  script:
    - npm run build:stage

build-prod-job:
  stage: build
  only:
    - master
  script:
    - npm run build:prod

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  script:
    - echo "Running unit tests..."
    - find dist/
    - echo "Test complete."

deploy-dev-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - development
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/*
www@192.168.30.11:/opt/netkiller.cn/admin.netkiller.cn/
    - echo "Application successfully deployed."

deploy-test-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - testing
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/*
www@192.168.30.10:/opt/netkiller.cn/admin.netkiller.cn/
    - echo "Application successfully deployed."

deploy-prod-job:
```

```
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - master
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/*
www@139.16.10.12:/opt/netkiller.cn/admin.netkiller.cn/
    - echo "Application successfully deployed."
```

**stages**

```
image: mileschou/php-testing-base:7.0

stages:
  - build
  - test
  - deploy

build_job:
  stage: build
  script:
    - composer install
  cache:
    untracked: true
  artifacts:
    paths:
      - vendor/

test_job:
  stage: test
  script:
    - php vendor/bin/codecept run
  dependencies:
    - build_job

deploy_job:
  stage: deploy
  script:
    - echo Deploy OK
  only:
    - release
  when: manual
```

```
  only:
    - master
  tags:
    - ansible
```

### variables

```
job1:
  variables:
    FOLDERS: src test docs
  script:
    - |
      for FOLDER in $FOLDERS
        do
          echo "The path is root/${FOLDER}"
        done
```

列出所有环境变量

使用 export 列出所有环境变量

```
build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  # variables:
    # accessKeyId: 123456
    # accessSecret: 654321
  tags:
    - docker
  before_script:
    - export
    - cat src/main/resources/application.properties
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
```

```
      - target/*.jar
```

```
$ export
21declare -x CI="true"
22declare -x CI_API_V4_URL="http://192.168.30.5/api/v4"
23declare -x CI_BUILDS_DIR="/builds"
24declare -x
CI_BUILD_BEFORE_SHA="213825d0cfd133aadb2648b0c1236f834e98972b"
25declare -x CI_BUILD_ID="4705"
26declare -x CI_BUILD_NAME="build-job"
27declare -x CI_BUILD_REF="61fe2acb56474b4b2ffb289de2c7d93afe514354"
28declare -x CI_BUILD_REF_NAME="development"
29declare -x CI_BUILD_REF_SLUG="development"
30declare -x CI_BUILD_STAGE="build"
31declare -x CI_BUILD_TOKEN="[MASKED]"
32declare -x CI_COMMIT_AUTHOR="neo <neo@t.com>"
33declare -x
CI_COMMIT_BEFORE_SHA="213825d0cfd133aadb2648b0c1236f834e98972b"
34declare -x CI_COMMIT_BRANCH="development"
35declare -x CI_COMMIT_DESCRIPTION=""
36declare -x CI_COMMIT_MESSAGE="更新.gitlab-ci.yml文件"
37declare -x CI_COMMIT_REF_NAME="development"
38declare -x CI_COMMIT_REF_PROTECTED="true"
39declare -x CI_COMMIT_REF_SLUG="development"
40declare -x CI_COMMIT_SHA="61fe2acb56474b4b2ffb289de2c7d93afe514354"
41declare -x CI_COMMIT_SHORT_SHA="61fe2acb"
42declare -x CI_COMMIT_TIMESTAMP="2021-09-18T07:00:58+00:00"
43declare -x CI_COMMIT_TITLE="更新.gitlab-ci.yml文件"
44declare -x CI_CONCURRENT_ID="0"
45declare -x CI_CONCURRENT_PROJECT_ID="0"
46declare -x CI_CONFIG_PATH=".gitlab-ci.yml"
47declare -x CI_DEFAULT_BRANCH="development"
48declare -x
CI_DEPENDENCY_PROXY_GROUP_IMAGE_PREFIX="192.168.30.5:80/neo/dependency_p
roxy/containers"
49declare -x CI_DEPENDENCY_PROXY_PASSWORD="[MASKED]"
50declare -x CI_DEPENDENCY_PROXY_SERVER="192.168.30.5:80"
51declare -x CI_DEPENDENCY_PROXY_USER="gitlab-ci-token"
52declare -x CI_DISPOSABLE_ENVIRONMENT="true"
53declare -x CI_JOB_ID="4705"
54declare -x CI_JOB_IMAGE="maven:3.8.2-openjdk-17"
55declare -x CI_JOB_JWT="[MASKED]"
56declare -x CI_JOB_NAME="build-job"
57declare -x CI_JOB_STAGE="build"
58declare -x CI_JOB_STARTED_AT="2021-09-18T07:01:07Z"
59declare -x CI_JOB_STATUS="running"
60declare -x CI_JOB_TOKEN="[MASKED]"
```

```
61declare -x CI_JOB_URL="http://192.168.30.5/neo/alertmanager-
webhook/-/jobs/4705"
62declare -x CI_NODE_TOTAL="1"
63declare -x CI_PAGES_DOMAIN="example.com"
64declare -x CI_PAGES_URL="http://neo.example.com/alertmanager-webhook"
65declare -x CI_PIPELINE_CREATED_AT="2021-09-18T07:00:58Z"
66declare -x CI_PIPELINE_ID="1866"
67declare -x CI_PIPELINE_IID="100"
68declare -x CI_PIPELINE_SOURCE="push"
69declare -x CI_PIPELINE_URL="http://192.168.30.5/neo/alertmanager-
webhook/-/pipelines/1866"
70declare -x CI_PROJECT_CLASSIFICATION_LABEL=""
71declare -x CI_PROJECT_DIR="/builds/neo/alertmanager-webhook"
72declare -x CI_PROJECT_ID="23"
73declare -x CI_PROJECT_NAME="alertmanager-webhook"
74declare -x CI_PROJECT_NAMESPACE="neo"
75declare -x CI_PROJECT_PATH="neo/alertmanager-webhook"
76declare -x CI_PROJECT_PATH_SLUG="neo-alertmanager-webhook"
77declare -x CI_PROJECT_REPOSITORY_LANGUAGES="java"
78declare -x CI_PROJECT_ROOT_NAMESPACE="neo"
79declare -x CI_PROJECT_TITLE="Alertmanager Webhook"
80declare -x CI_PROJECT_URL="http://192.168.30.5/neo/alertmanager-
webhook"
81declare -x CI_PROJECT_VISIBILITY="public"
82declare -x CI_REGISTRY_PASSWORD="[MASKED]"
83declare -x CI_REGISTRY_USER="gitlab-ci-token"
84declare -x CI_REPOSITORY_URL="http://gitlab-ci-token:
[MASKED]@192.168.30.5/neo/alertmanager-webhook.git"
85declare -x CI_RUNNER_DESCRIPTION="development"
86declare -x CI_RUNNER_EXECUTABLE_ARCH="linux/amd64"
87declare -x CI_RUNNER_ID="23"
88declare -x CI_RUNNER_REVISION="58ba2b95"
89declare -x CI_RUNNER_SHORT_TOKEN="GP-ozvd6"
90declare -x CI_RUNNER_TAGS="docker"
91declare -x CI_RUNNER_VERSION="14.2.0"
92declare -x CI_SERVER="yes"
93declare -x CI_SERVER_HOST="192.168.30.5"
94declare -x CI_SERVER_NAME="GitLab"
95declare -x CI_SERVER_PORT="80"
96declare -x CI_SERVER_PROTOCOL="http"
97declare -x CI_SERVER_REVISION="2da7c857960"
98declare -x CI_SERVER_URL="http://192.168.30.5"
99declare -x CI_SERVER_VERSION="14.2.1"
100declare -x CI_SERVER_VERSION_MAJOR="14"
101declare -x CI_SERVER_VERSION_MINOR="2"
102declare -x CI_SERVER_VERSION_PATCH="1"
103declare -x FF_CMD_DISABLE_DELAYED_ERROR_LEVEL_EXPANSION="false"
104declare -x FF_DISABLE_UMASK_FOR_DOCKER_EXECUTOR="false"
105declare -x FF_ENABLE_BASH_EXIT_CODE_CHECK="false"
106declare -x FF_GITLAB_REGISTRY_HELPER_IMAGE="true"
107declare -x FF_NETWORK_PER_BUILD="false"
```

```
108declare -x FF_SCRIPT_SECTIONS="false"
109declare -x
FF_SKIP_DOCKER_MACHINE_PROVISION_ON_CREATION_FAILURE="true"
110declare -x FF_SKIP_NOOP_BUILD_STAGES="true"
111declare -x FF_USE_DIRECT_DOWNLOAD="true"
112declare -x FF_USE_DYNAMIC_TRACE_FORCE_SEND_INTERVAL="false"
113declare -x FF_USE_FASTZIP="false"
114declare -x FF_USE_LEGACY_KUBERNETES_EXECUTION_STRATEGY="false"
115declare -x FF_USE_NEW_BASH_EVAL_STRATEGY="false"
116declare -x FF_USE_POWERSHELL_PATH_RESOLVER="false"
117declare -x FF_USE_WINDOWS_LEGACY_PROCESS_STRATEGY="true"
118declare -x GITLAB_CI="true"
119declare -x GITLAB_FEATURES=""
120declare -x GITLAB_USER_EMAIL="neo@t.com"
121declare -x GITLAB_USER_ID="2"
122declare -x GITLAB_USER_LOGIN="neo"
123declare -x GITLAB_USER_NAME="neo"
124declare -x HOME="/root"
125declare -x HOSTNAME="runner-gp-ozvd6-project-23-concurrent-0"
126declare -x JAVA_HOME="/usr/java/openjdk-17"
127declare -x JAVA_VERSION="17"
128declare -x LANG="C.UTF-8"
129declare -x MAVEN_HOME="/usr/share/maven"
130declare -x OLDPWD="/"
131declare -x PATH="/usr/java/openjdk-
17/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
132declare -x PWD="/builds/neo/alertmanager-webhook"
133declare -x SHLVL="1"
```

**only and except**

only 用于匹配分支

```
deploy_job:
  stage: deploy
  script:
    - echo Deploy OK
  only:
    - master
  when: manual
```

only 可是使用正则表达式，还可能与 except 一同使用，用于排除分支

```
job:
  # use regexp
  only:
    - /^issue-.*$/
  # use special keyword
  except:
    - branches
```

使用关键字

```
job:
  # use special keywords
  only:
    - tags
    - triggers
```

only和except允许使用特殊的关键字：

- branches 匹配所有 git 分支
- tags 匹配所有 git tag
- triggers

匹配 feature / hotfix 分支

```
deploy-feature-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
    # DOCKER_HOST: unix:///var/run/docker.sock mvn clean install
docker:build
  environment:
    name: feature
    url: https://api.netkiller.cn
  only:
    - /^feature\/.*/
  tags:
    - shell
```

```
  before_script:
    - mvn docker:build -DpushImage
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-
%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml
www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml restart"
  when: manual

deploy-hotfix-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: hotfix
    url: https://api.netkiller.cn
  only:
    - /^hotfix\/.*/
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-
%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml
www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml restart"
  when: manual
```

### 允许失败

设置当一个job运行失败之后并不影响后续的CI构建过程

```
job1:
  stage: build
  script:
  - execute_script_that_will_fail

job2:
  stage: test
  script:
  - execute_script_that_will_succeed
  allow_failure: true

job3:
  stage: deploy
  script:
  - deploy_to_staging
```

### 定义何时开始job

when: 可以是on_success，on_failure，always或者manual

when可以设置以下值：

- on_success: 只有前面stages的所有工作成功时才执行。 这是默认值。
- on_failure: 当前面stages中任意一个jobs失败后执行。
- always: 无论前面stages中jobs状态如何都执行。
- manual: 手动执行

```
```

### services

```
services:
```

```
- mysql

variables:
  # Configure mysql service (https://hub.docker.com/_/mysql/)
  MYSQL_DATABASE: hello_world_test
  MYSQL_ROOT_PASSWORD: mysql

connect:
  image: mysql
  script:
  - echo "SELECT 'OK';" | mysql --user=root --
password="$MYSQL_ROOT_PASSWORD" --host=mysql "$MYSQL_DATABASE"
```

**tags**

在 gitlab-runner register 的时候会提示：Please enter the gitlab-ci tags for this runner (comma separated):

如果你输入了标签就需要在 Pipeline 中设置 tags 否则 Pipeline 将不运行。

```
# This file is a template, and might need editing before it works on
your project.
# This is a sample GitLab CI/CD configuration file that should run
without any modifications.
# It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests
or scripts,
# it uses echo commands to simulate the pipeline execution.
#
# A pipeline is composed of independent jobs that run scripts, grouped
into stages.
# Stages run in sequential order, but jobs within stages run in
parallel.
#
# For more information, see:
https://docs.gitlab.com/ee/ci/yaml/README.html#stages

stages:          # List of stages for jobs, and their order of execution
  - build
  - test
  - deploy

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  tags:
    - neo
```

```
  script:
    - echo "Compiling the code..."
    - echo "Compile complete."

unit-test-job:   # This job runs in the test stage.
  stage: test    # It only starts when the job in the build stage
completes successfully.
  tags:
    - neo
  script:
    - echo "Running unit tests... This will take about 60 seconds."
    - sleep 60
    - echo "Code coverage is 90%"

lint-test-job:   # This job also runs in the test stage.
  stage: test    # It can run at the same time as unit-test-job (in
parallel).
  script:
    - echo "Linting code... This will take about 10 seconds."
    - sleep 10
    - echo "No lint issues found."

deploy-job:      # This job runs in the deploy stage.
  stage: deploy  # It only runs when *both* jobs in the test stage
complete successfully.
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
```

### rules 规则

```
job-name:
  script:
    - echo "i am potato"
  rules:
    - if: '$CI_COMMIT_BRANCH != "potato"'
```

### Git submodule

```
variables:
  GIT_SUBMODULE_STRATEGY: recursive
```

## 11. 应用案例

### Java

```
before_script:
 - echo "Execute scripts which are required to bootstrap the application. !"

after_script:
 - echo "Clean up activity can be done here !."

stages:
 - build
 - test
 - package
 - deploy

variables:
 MAVEN_CLI_OPTS: "--batch-mode"
 MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

cache:
 paths:
  - .m2/repository/
  - target/

build:
 stage: build
 image: maven:latest
 script:
  - mvn $MAVEN_CLI_OPTS clean compile

test:
 stage: test
 image: maven:latest
 script:
  - mvn $MAVEN_CLI_OPTS test

package:
 stage: package
 image: maven:latest
 script:
  - mvn $MAVEN_CLI_OPTS package
 artifacts:
  paths: [target/test-0.0.1.war]


deploy_test:
 stage: deploy
 script:
  - echo "########   To be defined   ########"
 environment: staging
```

```
deploy_prod:
 stage: deploy
 script:
  - echo "########   To be defined   ########"
 only:
  - master
 environment: production
```

使用 **Docker** 编译并收集构建物

```
#image: java:8
#image: maven:latest
image: maven:3.5.0-jdk-8

stages:
  - build
  - test
  - package

build:
  stage: build
  script: mvn compile

unittest:
  stage: test
  script: mvn test


package:
  stage: package
  script: mvn package
  artifacts:
    paths:
      - target/java-project-0.0.1-SNAPSHOT.jar
```

**Shell** 执行器，远程部署物理机/虚拟机

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
```

```
  stage: build
  tags:
    - shell
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - tags
    - testing
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-
%d.%H:%M:%S).sql.gz
  # after_script:
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - rsync -auzv target/*.jar www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh -f -C -q www@$HOST "pkill java; sleep 5; java -jar
/opt/netkiller.cn/api.netkiller.cn/alertmanager-0.0.1.jar >/dev/null 2>&1 &"
```

**Shell 执行器，远程部使用容器启动项目**

```
cache:
  untracked: true
```

```
stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  tags:
    - shell
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - tags
    - testing
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-
%d.%H:%M:%S).sql.gz
  # after_script:
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - rsync -auzv target/*.jar www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - rsync -auzv src/main/docker/development/docker-compose.yaml
www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml up -d api"
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml restart api"
```

**Docker 执行器**

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  tags:
    - docker
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  image: maven:3.8.2-openjdk-17
  stage: test
  variables:
    GIT_STRATEGY: none
  tags:
    - docker
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    # - DOCKER_HOST=unix:///var/run/docker.sock mvn clean install docker:build
    - mvn docker:build -DpushImage
    # - mvn docker:push
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-
```

```
%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml
www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml restart"
```

### Node

```
cache:
  paths:
    - node_modules
    # - dist

stages:
  - build
  # - test
  - deploy

build-job:
  stage: build
  script:
    - npm install
#     - yarn install
#     - yarn run build

# unit-test-job:
#   stage: test
#   script:
#     - yarn run test

# lint-test-job:
#   stage: test
#   script:
#     - yarn run lint

deploy-job:
  stage: deploy
  script:
    - rsync -auzv --delete *
www@192.168.30.10:/opt/netkiller.cn/www.netkiller.cn/
    - ssh www@192.168.0.10 "sudo pm2 --update-env restart
/opt/netkiller.cn/www.netkiller.cn/main.js"
```

### vue.js android

```
build site:
  image: node:6
  stage: build
  script:
    - npm install --progress=false
    - npm run build
  artifacts:
    expire_in: 1 week
    paths:
      - dist

unit test:
  image: node:6
  stage: test
  script:
    - npm install --progress=false
    - npm run unit

deploy:
  image: alpine
  stage: deploy
  script:
    - apk add --no-cache rsync openssh
    - mkdir -p ~/.ssh
    - echo "$SSH_PRIVATE_KEY" >> ~/.ssh/id_dsa
    - chmod 600 ~/.ssh/id_dsa
    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
    - rsync -rav --delete dist/ user@server.com:/your/project/path/
```

### docker

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  tags:
    - docker
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
```

```
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  image: maven:3.8.2-openjdk-17
  stage: test
  variables:
    GIT_STRATEGY: none
  tags:
    - docker
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
    DOCKER_HOST: unix:///var/run/docker.sock mvn clean install docker:build
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    # - mvn docker:push
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-
%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml
www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f
/opt/netkiller.cn/api.netkiller.cn/docker-compose.yaml restart"
```

## 12. FAQ

### 查看日志

```
gitlab-ctl tail
gitlab-ctl tail gitlab-rails
gitlab-ctl tail nginx/gitlab_error.log
```

### debug runner

```
gitlab-runner -debug run
```

### gitolite 向 gitlab 迁移

早期gitlab使用gitolite为用户提供SSH服务，新版gitlab有了更好的解决方案gitlab-shell。安装新版本是必会涉及gitolite 向 gitlab 迁移，下面是我总结的一些迁移经验。

第一步,将gitolite复制到gitlab仓库目录下

```
# cp -r /gitroot/gitolite/repositories/* /var/opt/gitlab/git-data/repositories/
```

执行导入处理程序

```
# gitlab-rake gitlab:import:repos
```

上面程序会处理一下目录结构，例如

进入gitlab web界面，创建仓库与导入的仓库同名，这样就完成了导入工作。

**提示**

转换最好在git用户下面操作，否则你需要运行

```
# chown git:git -R /var/opt/gitlab/git-data/repositories
```

### 修改主机名

默认Gitlab采用主机名，给我使用代理一定麻烦

```
git@hostname:example.com/www.example.com.git
http://hostname/example.com/www.example.com.git
```

我们希望使用IP地址替代主机名

```
git@172.16.0.1:example.com/www.example.com.git
http://172.16.0.1/example.com/www.example.com.git
```

编辑 /etc/gitlab/gitlab.rb 配置文件

```
external_url 'http://172.16.0.1'
```

重新启动Gitlab

```
# gitlab-ctl reconfigure
# gitlab-ctl restart
```

**ERROR: Uploading artifacts as "archive" to coordinator... too large archive**

持续集成提示错误

```
ERROR: Uploading artifacts as "archive" to coordinator... too large archive  id=185
responseStatus=413 Request Entity Too Large status=413 token=HKerPDE6
FATAL: too large
ERROR: Job failed: exit status 1
```

解决方案

Admin - Settings - CI/CD - Continuous Integration and Deployment

点击 Expand 展开配置项

Maximum artifacts size (MB): 修改构建无最大尺寸

# 第 2 章 Jenkins

## 1. 安装 Jenkins

### OSCM 一键安装

```
yum install -y java-1.8.0-openjdk
curl -s
https://raw.githubusercontent.com/oscm/shell/master/project/jenkins/jenk
ins.sh | bash
```

### Mac

使用 pkg 方式安装，默认路径是 /Applications/Jenkins/jenkins.war

```
export
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_92.jdk/Contents/Hom
e
java -jar jenkins.war --httpPort=8080
```

浏览器访问：http://localhost:8080

查看默认密码 /Users/neo/.jenkins/secrets/initialAdminPassword

```
neo@MacBook-Pro ~ % cat /Users/neo/.jenkins/secrets/initialAdminPassword
6c7369afc6c1414586b6644657dd655a
```

下载 cloudbees 插件

```
neo@MacBook-Pro ~ % cd ~/.jenkins/plugins
neo@MacBook-Pro ~/.jenkins/plugins % wget
```

```
ftp://ftp.icm.edu.pl/packages/jenkins/plugins/cloudbees-
folder//6.7/cloudbees-folder.hpi
```

重启 Jenkens http://localhost:8080/restart

复制上面的密码，粘贴到浏览器中。

卸载 Jenkens

```
sudo rm -rf /var/root/.jenkins ~/.jenkins
sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist
sudo rm /Library/LaunchDaemons/org.jenkins-ci.plist
sudo rm -rf /Applications/Jenkins "/Library/Application Support/Jenkins"
/Library/Documentation/Jenkins

sudo rm -rf /Users/Shared/Jenkins
sudo dscl . -delete /Users/jenkins
sudo dscl . -delete /Groups/jenkins
sudo rm -f /etc/newsyslog.d/jenkins.conf
pkgutil --pkgs | grep 'org\.jenkins-ci\.' | xargs -n 1 sudo pkgutil --
forget
```

由于我的Mac 模式是 JDK 11，所以需要制定 JAVA_HOME 到 JDK 1.8，否则提示

```
Dec 27, 2018 9:20:33 AM Main main
SEVERE: Running with Java class version 55.0, but 52.0 is required.Run
with the --enable-future-java flag to enable such behavior. See
https://jenkins.io/redirect/java-support/
java.lang.UnsupportedClassVersionError: 55.0
        at Main.main(Main.java:139)

Jenkins requires Java 8, but you are running 11+28 from
/Library/Java/JavaVirtualMachines/jdk-11.jdk/Contents/Home
java.lang.UnsupportedClassVersionError: 55.0
        at Main.main(Main.java:139)
```

**CentOS**

```
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-
stable/jenkins.repo
rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

yum install -y jenkins
```

cat /etc/sysconfig/jenkins

```
## Path:        Development/Jenkins
## Description: Jenkins Automation Server
## Type:        string
## Default:     "/var/lib/jenkins"
## ServiceRestart: jenkins
#
# Directory where Jenkins store its configuration and working
# files (checkouts, build reports, artifacts, ...).
#
JENKINS_HOME="/var/lib/jenkins"

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# Java executable to run Jenkins
# When left empty, we'll try to find the suitable Java.
#
JENKINS_JAVA_CMD=""

## Type:        string
## Default:     "jenkins"
## ServiceRestart: jenkins
#
# Unix user account that runs the Jenkins daemon
# Be careful when you change this, as you need to update
# permissions of $JENKINS_HOME and /var/log/jenkins.
#
JENKINS_USER="jenkins"

## Type:        string
## Default: "false"
## ServiceRestart: jenkins
#
# Whether to skip potentially long-running chown at the
# $JENKINS_HOME location. Do not enable this, "true", unless
# you know what you're doing. See JENKINS-23273.
```

```
#
#JENKINS_INSTALL_SKIP_CHOWN="false"

## Type: string
## Default:     "-Djava.awt.headless=true"
## ServiceRestart: jenkins
#
# Options to pass to java when running Jenkins.
#
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true"

## Type:        integer(0:65535)
## Default:     8080
## ServiceRestart: jenkins
#
# Port Jenkins is listening on.
# Set to -1 to disable
#
JENKINS_PORT="8080"

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# IP address Jenkins listens on for HTTP requests.
# Default is all interfaces (0.0.0.0).
#
JENKINS_LISTEN_ADDRESS=""

## Type:        integer(0:65535)
## Default:     ""
## ServiceRestart: jenkins
#
# HTTPS port Jenkins is listening on.
# Default is disabled.
#
JENKINS_HTTPS_PORT=""

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# Path to the keystore in JKS format (as created by the JDK 'keytool').
# Default is disabled.
#
JENKINS_HTTPS_KEYSTORE=""

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
```

```
# Password to access the keystore defined in JENKINS_HTTPS_KEYSTORE.
# Default is disabled.
#
JENKINS_HTTPS_KEYSTORE_PASSWORD=""

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# IP address Jenkins listens on for HTTPS requests.
# Default is disabled.
#
JENKINS_HTTPS_LISTEN_ADDRESS=""


## Type:        integer(1:9)
## Default:     5
## ServiceRestart: jenkins
#
# Debug level for logs -- the higher the value, the more verbose.
# 5 is INFO.
#
JENKINS_DEBUG_LEVEL="5"

## Type:        yesno
## Default:     no
## ServiceRestart: jenkins
#
# Whether to enable access logging or not.
#
JENKINS_ENABLE_ACCESS_LOG="no"

## Type:        integer
## Default:     100
## ServiceRestart: jenkins
#
# Maximum number of HTTP worker threads.
#
JENKINS_HANDLER_MAX="100"

## Type:        integer
## Default:     20
## ServiceRestart: jenkins
#
# Maximum number of idle HTTP worker threads.
#
JENKINS_HANDLER_IDLE="20"

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
```

```
#
# Pass arbitrary arguments to Jenkins.
# Full option list: java -jar jenkins.war --help
#
JENKINS_ARGS=""
```

Nginx 配置

```
[root@netkiller ~]# cat /etc/nginx/conf.d/jk.netkiller.cn.conf
server {
    listen       80;
    server_name  jk.netkiller.cn;

    charset utf-8;

    location / {
        proxy_pass   http://127.0.0.1:8080;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

查看管理员密码

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

**Ubuntu**

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
apt-key add -
```

```
deb https://pkg.jenkins.io/debian-stable binary/

sudo apt-get update
sudo apt-get install jenkins
```

**Docker**

https://github.com/jenkinsci/docker/blob/master/README.md

8080端口是jenkins的端口，5000端口是master和slave通信端口

```
docker pull jenkins/jenkins:lts
docker run -p 8080:8080 -p 50000:50000 --name jenkins
jenkins/jenkins:lts
```

首次启动，不要使用 -d 参数，如果使用了 -d 参数可以通过docker logs -f jenkins查看控制台的密码

docker-compose 配置文件

```
version: '2'

services:
  jenkins:
    container_name: jenkins-lts
    ports: # 端口映射，9001为宿主机上的端口，相应的8080是容器运行起来时候jenkins
服务的端口
      - 9001:8080
      - 50000:50000
    image: jenkins/jenkins:lts # 指定运行用哪一个镜像来运行容器
    volumes:
      - /home/jenkins/jenkins_home:/var/jenkins_home # 挂载指令，目的在于销
毁容器时，并不影响jenkins数据
```

**Minikube**

创建 jenkins-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: jenkins-project
```

创建命名空间

```
$ kubectl create -f jenkins-namespace.yaml
```

创建 jenkins-volume.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: jenkins-pv
  namespace: jenkins-project
spec:
  storageClassName: jenkins-pv
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 20Gi
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /opt/jenkins-volume/
```

创建卷

```
$ kubectl create -f jenkins-volume.yaml
persistentvolume “jenkins-pv” created
```

创建 jenkins-values.yaml 文件

```
# Default values for jenkins.
# This is a YAML-formatted file.
# Declare name/value pairs to be passed into your templates.
# name: value

## Overrides for generated resource names
# See templates/_helpers.tpl
# nameOverride:
# fullnameOverride:

Master:
  Name: jenkins-master
  Image: "jenkins/jenkins"
  ImageTag: "2.141"
  ImagePullPolicy: "Always"
  Component: "jenkins-master"
  UseSecurity: true
  AdminUser: admin
  # AdminPassword: <defaults to random>
  Cpu: "200m"
  Memory: "256Mi"
  ServicePort: 8080
  # For minikube, set this to NodePort, elsewhere use LoadBalancer
  # <to set explicitly, choose port between 30000-32767>
  ServiceType: NodePort
  NodePort: 32000
  ServiceAnnotations: {}
  ContainerPort: 8080
  # Enable Kubernetes Liveness and Readiness Probes
  HealthProbes: true
  HealthProbesTimeout: 60
  SlaveListenerPort: 50000
  LoadBalancerSourceRanges:
    - 0.0.0.0/0
  # List of plugins to be install during Jenkins master start
  InstallPlugins:
    - kubernetes:1.12.4
    - workflow-aggregator:2.5
    - workflow-job:2.24
    - credentials-binding:1.16
    - git:3.9.1
    - greenballs:1.15
  # Used to approve a list of groovy functions in pipelines used the
script-security plugin. Can be viewed under /scriptApproval
  ScriptApproval:
    - "method groovy.json.JsonSlurperClassic parseText java.lang.String"
    - "new groovy.json.JsonSlurperClassic"
    - "staticMethod org.codehaus.groovy.runtime.DefaultGroovyMethods
leftShift java.util.Map java.util.Map"
```

```
    - "staticMethod org.codehaus.groovy.runtime.DefaultGroovyMethods
split java.lang.String"
  CustomConfigMap: false
  NodeSelector: {}
  Tolerations: {}

Agent:
  Enabled: true
  Image: jenkins/jnlp-slave
  ImageTag: 3.10-1
  Component: "jenkins-slave"
  Privileged: false
  Cpu: "200m"
  Memory: "256Mi"
  # You may want to change this to true while testing a new image
  AlwaysPullImage: false
  # You can define the volumes that you want to mount for this container
  # Allowed types are: ConfigMap, EmptyDir, HostPath, Nfs, Pod, Secret
  volumes:
    - type: HostPath
      hostPath: /var/run/docker.sock
      mountPath: /var/run/docker.sock
  NodeSelector: {}

Persistence:
  Enabled: true
  ## A manually managed Persistent Volume and Claim
  ## Requires Persistence.Enabled: true
  ## If defined, PVC must be created manually before volume will be
bound
  # ExistingClaim:
  ## jenkins data Persistent Volume Storage Class
  StorageClass: jenkins-pv

  Annotations: {}
  AccessMode: ReadWriteOnce
  Size: 20Gi
  volumes:
  #  - name: nothing
  #    emptyDir: {}
  mounts:
  #  - mountPath: /var/nothing
  #    name: nothing
  #    readOnly: true

NetworkPolicy:
  # Enable creation of NetworkPolicy resources.
  Enabled: false
  # For Kubernetes v1.4, v1.5 and v1.6, use 'extensions/v1beta1'
  # For Kubernetes v1.7, use 'networking.k8s.io/v1'
  ApiVersion: networking.k8s.io/v1
```

```
## Install Default RBAC roles and bindings
rbac:
  install: true
  serviceAccountName: default
  # RBAC api version (currently either v1beta1 or v1alpha1)
  apiVersion: v1beta1
  # Cluster role reference
  roleRef: cluster-admin
```

使用 helm 安装 jenkins

```
$ cd ~/minikube-helm-jenkins
$ helm init
$ helm install --name jenkins -f helm/jenkins-values.yaml stable/jenkins
--namespace jenkins-project
```

查看 jenkins 密码

```
$ printf $(kubectl get secret --namespace jenkins-project jenkins -o
jsonpath="{.data.jenkins-admin-password}" | base64 --decode);echo
```

## 2. 配置 Jenkins

配置 Jenkins

输入管理员密码

安装插件

Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
Subversion
SSH Slaves
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Folders
** JDK Tool
** Script Security
** Command Agent Launcher
** Structs
** bouncycastle API
** Pipeline: Step API
** SCM API
** Pipeline: API
** JUnit
OWASP Markup Formatter
** Token Macro
Build Timeout
** - required dependency
Jenkins 2.150.1
Screenshot

创建用户

Getting Started
Create First Admin User
Username:
neo
Password:
Confirm password:
Full name:
neo chen
E-mail address:
neo.chen@live.com
Jenkins 2.150.1
Continue as admin
Save and Continue

设置域名

开始使用 Jenkins

Jenkins 界面

## 3. Jenkinsfile

### Jenkinsfile - Declarative Pipeline

https://jenkins.io/doc/pipeline/examples/

**stages**

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

**script**

```
// Declarative //
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
                script {
                    def browsers = ['chrome', 'firefox']
                    for (int i = 0; i < browsers.size(); ++i) {
                        echo "Testing the ${browsers[i]} browser"
                    }
                }
                script {
```

```
                    // 一个优雅的退出pipeline的方法，这里可执行任意逻辑
                    if( $VALUE1 == $VALUE2 ) {
                       currentBuild.result = 'SUCCESS'
                       return
                    }
                }
            }
        }
    }
}
```

**junit**

junit4

```
                stage("测试") {
            steps {
                echo "单元测试中..."
                // 请在这里放置您项目代码的单元测试调用过程，例如：
                sh 'mvn test' // mvn 示例
                // sh './gradlew test'
                echo "单元测试完成."
                junit 'target/surefire-reports/*.xml' // 收集单元测试报告的调用过程
            }
        }
```

junit5 测试报告路径与 junit4 的位置不同

```
                stage("测试") {
            steps {
                echo "单元测试中..."
                sh './gradlew test'
                echo "单元测试完成."
                junit 'build/test-results/test/*.xml'
            }
        }
```

**withEnv**

```
env.PROJECT_DIR='src/netkiller'
node {
    withEnv(["GOPATH=$WORKSPACE"]) {
```

```
        stage('Init gopath') {
            sh 'mkdir -p $GOPATH/{bin,pkg,src}'
        }

        stage('Build go proejct') {
            sh 'cd ${PROJECT_DIR}; go test && go build && go install'
        }
    }
}
```

```
node {
     git 'https://github.com/netkiller/api.git'
     withEnv(["PATH+MAVEN=${tool 'm3'}/bin"]) {
          sh "mvn -B —Dmaven.test.failure.ignore=true clean package"
     }
     stash excludes: 'target/', includes: '**', name: 'source'
}
```

**parameters**

参数指令，触发这个管道需要用户指定的参数，然后在step中通过params对象访问这些参数。

```
// Declarative //
pipeline {
    agent any
    parameters {
        string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who
should I say hello to?')
    }
    stages {
        stage('Example') {
            steps {
                echo "Hello ${params.PERSON}"
            }
        }
    }
}
```

```
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any
    parameters {
```

```
        string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who
should I say hello to?')

        text(name: 'BIOGRAPHY', defaultValue: '', description: 'Enter some
information about the person')

        booleanParam(name: 'TOGGLE', defaultValue: true, description: 'Toggle
this value')

        choice(name: 'CHOICE', choices: ['One', 'Two', 'Three'], description:
'Pick something')

        password(name: 'PASSWORD', defaultValue: 'SECRET', description: 'Enter a
password')

        file(name: "FILE", description: "Choose a file to upload")
    }
    stages {
        stage('Example') {
            steps {
                echo "Hello ${params.PERSON}"

                echo "Biography: ${params.BIOGRAPHY}"

                echo "Toggle: ${params.TOGGLE}"

                echo "Choice: ${params.CHOICE}"

                echo "Password: ${params.PASSWORD}"
            }
        }
    }
}
```

**options**

还能定义一些管道特定的选项，介绍几个常用的：

```
skipDefaultCheckout - 在agent指令中忽略源码checkout这一步骤。
timeout - 超时设置options { timeout(time: 1, unit: 'HOURS') }
retry - 直到成功的重试次数options { retry(3) }
timestamps - 控制台输出前面加时间戳options { timestamps() }
```

**triggers**

触发器指令定义了这个管道何时该执行，就可以定义两种cron和pollSCM

```
cron - linux的cron格式triggers { cron('H 4/* 0 0 1-5') }
pollSCM - jenkins的poll scm语法，比如triggers { pollSCM('H 4/* 0 0 1-5') }

// Declarative //
pipeline {
    agent any
    triggers {
        cron('H 4/* 0 0 1-5')
    }
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

一般我们会将管道和GitHub、GitLab、BitBucket关联， 然后使用它们的webhooks来触发，就不需要这个指令了。

**tools**

定义自动安装并自动放入PATH里面的工具集合

```
// Declarative //
pipeline {
    agent any
    tools {
        maven 'apache-maven-3.5.0' ①
    }
    stages {
        stage('Example') {
            steps {
                sh 'mvn --version'
            }
        }
    }
}
```

注：① 工具名称必须预先在Jenkins中配置好了 → Global Tool Configuration.

**post**

post section 定义了管道执行结束后要进行的操作。支持在里面定义很多Conditions 块： always, changed, failure, success 和 unstable。 这些条件块会根据不同的返回结果来执行不同的逻辑。

```
always：不管返回什么状态都会执行
changed：如果当前管道返回值和上一次已经完成的管道返回值不同时候执行
failure：当前管道返回状态值为”failed”时候执行，在Web UI界面上面是红色的标志
success：当前管道返回状态值为”success”时候执行，在Web UI界面上面是绿色的标志
unstable：当前管道返回状态值为”unstable”时候执行，通常因为测试失败，代码不合法引起的。在Web UI界面上面是黄色的标志

// Declarative //
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
    post {
        always {
            echo 'I will always say Hello again!'
        }
    }
}
```

失败发送邮件的例子

```
    post {
        failure {
            mail to: "${email}",
            subject: "Failed Pipeline: ${currentBuild.fullDisplayName}",
            body: "Something is wrong with ${env.BUILD_URL}"
        }
    }
```

**when 条件判断**

```
branch - 分支匹配才执行 when { branch 'master' }
environment - 环境变量匹配才执行 when { environment name: 'DEPLOY_TO', value: 'production' }
expression - groovy表达式为真才执行 expression { return params.DEBUG_BUILD } }

// Declarative //
pipeline {
    agent any
    stages {
```

```
        stage('Example Build') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Example Deploy') {
            when {
                branch 'production'
            }
            echo 'Deploying'
        }
    }
}
```

抛出错误

```
error '执行出错'
```

**withCredentials**

**withCredentials: Bind credentials to variables**

**token**

```
node {
   withCredentials([string(credentialsId: 'token', variable: 'TOKEN')]) {
        sh('echo $TOKEN')
   }
}
```

**withMaven**

```
                withMaven(
            maven: 'M3') {
                sh "mvn test"
        }
```

**isUnix()** 判断操作系统类型

```
pipeline{

        agent any
        stages{
                stage("isUnix") {
                        steps{
                                script {
                                        if(isUnix() == true) {
                                                echo("this jenkins job running
on a linux-like system")
                                        }else {
                                                error("the jenkins job running
on a windows system")
                                        }
                                }
                        }
                }
        }
}
```

**Jenkins pipeline 中使用 sshpass 实现 scp, ssh 远程运行**

```
pipeline {
    agent {
        label "java-8"
    }
    stages  {

        stage("环境") {
            steps {
                parallel "Maven": {
                        script{
                        sh 'mvn -version'
                        }
                }, "Java": {
                    sh 'java -version'
                }, "sshpass": {
                        sh 'apt install -y sshpass'
                    sh 'sshpass -v'
                }
            }

        }

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
```

```
            }
        }

        stage("构建") {
            steps {
                echo "构建中..."
                sh 'mvn package -Dmaven.test.skip=true'
                archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
                echo "构建完成."
            }
        }

        stage("测试") {
            steps {
                parallel "单元测试": {
                    echo "单元测试中..."
                    sh 'mvn test'
                    echo "单元测试完成."
                    junit 'target/surefire-reports/*.xml'
                }, "接口测试": {
                    echo "接口测试中..."
                    // 请在这里放置您项目代码的单元测试调用过程，例如 mvn test
                    echo "接口测试完成."
                }, "测试敏感词":{
                    echo "Username: ${env.username}"
                        echo "Password: ${env.password}"
                }

            }

        }
        stage("运行"){
                steps {
                sh 'java -jar target/java-0.0.1-SNAPSHOT.jar'
            }
        }
        stage("部署"){
                steps {
              echo "上传"
              sh 'sshpass -p Passw0rd scp target/*.jar
root@dev.netkiller.cn:/root/'
              echo "运行"
              sh 'sshpass -p Passw0rd ssh root@dev.netkiller.cn java -jar
/root/java-0.0.1-SNAPSHOT.jar'

            }
        }
    }
}
```

后台运行

```
```

```
                stage("部署"){
            parallel{
                    stage("sshpass") {
                                steps{
                                    sh 'apt install -y sshpass'
                                sh 'sshpass -v'
                                }
                        }
                stage('stop') {
                                 steps {
                                    sh 'sshpass -p passw0rd ssh -f
dev.netkiller.cn pkill -f java-project-0.0.2-SNAPSHOT'
                                 }
                            }
                            stage('start') {
                                 steps {
                                    sh 'sshpass -p passw0rd scp target/*.jar
dev.netkiller.cn:/root/'
                            sh 'sshpass -p passw0rd ssh -f dev.netkiller.cn java
-jar /root/java-project-0.0.2-SNAPSHOT.jar'
                                  }
                            }
            }
        }
```

## Jenkinsfile - Scripted Pipeline

```
// Jenkinsfile (Scripted Pipeline)
node {
    stage('Build') {
        echo 'Building....'
    }
    stage('Test') {
        echo 'Building....'
    }
    stage('Deploy') {
        echo 'Deploying....'
    }
}
```

### git

```
node {

   stage('Checkout') {
      git 'https://github.com/bg7nyt/java.git'
   }
```

```
}
```

切换 **JDK** 版本

```
node('vagrant-slave') {
    env.JAVA_HOME="${tool 'jdk-8u45'}"
    env.PATH="${env.JAVA_HOME}/bin:${env.PATH}"
    sh 'java -version'
}
```

**groovy**

```
#!groovy
import groovy.json.JsonOutput
import groovy.json.JsonSlurper

/*
Please make sure to add the following environment variables:
HEROKU_PREVIEW=<your heroku preview app>
HEROKU_PREPRODUCTION=<your heroku pre-production app>
HEROKU_PRODUCTION=<your heroku production app>
Please also add the following credentials to the global domain of your
organization's folder:
Heroku API key as secret text with ID 'HEROKU_API_KEY'
GitHub Token value as secret text with ID 'GITHUB_TOKEN'
*/

node {

     server = Artifactory.server "artifactory"
     buildInfo = Artifactory.newBuildInfo()
     buildInfo.env.capture = true

    // we need to set a newer JVM for Sonar
    env.JAVA_HOME="${tool 'Java SE DK 8u131'}"
    env.PATH="${env.JAVA_HOME}/bin:${env.PATH}"

    // pull request or feature branch
    if  (env.BRANCH_NAME != 'master') {
        checkout()
        build()
        unitTest()
        // test whether this is a regular branch build or a merged PR build
        if (!isPRMergeBuild()) {
            preview()
            sonarServer()
```

```
            allCodeQualityTests()
        } else {
            // Pull request
            sonarPreview()
        }
    } // master branch / production
    else {
        checkout()
        build()
        allTests()
        preview()
        sonarServer()
        allCodeQualityTests()
        preProduction()
        manualPromotion()
        production()
    }
}

def isPRMergeBuild() {
    return (env.BRANCH_NAME ==~ /^PR-\d+$/)
}

def sonarPreview() {
    stage('SonarQube Preview') {
        prNo = (env.BRANCH_NAME=~/^PR-(\d+)$/)[0][1]
        mvn "org.jacoco:jacoco-maven-plugin:prepare-agent install -
Dmaven.test.failure.ignore=true -Pcoverage-per-test"
        withCredentials([[$class: 'StringBinding', credentialsId:
'GITHUB_TOKEN', variable: 'GITHUB_TOKEN']]) {
            githubToken=env.GITHUB_TOKEN
            repoSlug=getRepoSlug()
            withSonarQubeEnv('SonarQube Octodemoapps') {
                mvn "-Dsonar.analysis.mode=preview -
Dsonar.github.pullRequest=${prNo} -Dsonar.github.oauth=${githubToken} -
Dsonar.github.repository=${repoSlug} -
Dsonar.github.endpoint=https://api.github.com/
org.sonarsource.scanner.maven:sonar-maven-plugin:3.2:sonar"
            }
        }
    }
}

def sonarServer() {
    stage('SonarQube Server') {
        mvn "org.jacoco:jacoco-maven-plugin:prepare-agent install -
Dmaven.test.failure.ignore=true -Pcoverage-per-test"
        withSonarQubeEnv('SonarQube Octodemoapps') {
            mvn "org.sonarsource.scanner.maven:sonar-maven-plugin:3.2:sonar"
        }

        context="sonarqube/qualitygate"
        setBuildStatus ("${context}", 'Checking Sonarqube quality gate',
'PENDING')
        timeout(time: 1, unit: 'MINUTES') { // Just in case something goes
wrong, pipeline will be killed after a timeout
            def qg = waitForQualityGate() // Reuse taskId previously collected
```

```
by withSonarQubeEnv
            if (qg.status != 'OK') {
                setBuildStatus ("${context}", "Sonarqube quality gate fail:
${qg.status}", 'FAILURE')
                error "Pipeline aborted due to quality gate failure:
${qg.status}"
            } else {
                setBuildStatus ("${context}", "Sonarqube quality gate pass:
${qg.status}", 'SUCCESS')
            }
        }
    }
}




def checkout () {
    stage 'Checkout code'
    context="continuous-integration/jenkins/"
    context += isPRMergeBuild()?"pr-merge/checkout":"branch/checkout"
    checkout scm
    setBuildStatus ("${context}", 'Checking out completed', 'SUCCESS')
}

def build () {
    stage 'Build'
    mvn 'clean install -DskipTests=true -Dmaven.javadoc.skip=true -
Dcheckstyle.skip=true -B -V'
}


def unitTest() {
    stage 'Unit tests'
    mvn 'test -B -Dmaven.javadoc.skip=true -Dcheckstyle.skip=true'
    if (currentBuild.result == "UNSTABLE") {
        sh "exit 1"
    }
}

def allTests() {
    stage 'All tests'
    // don't skip anything
    mvn 'test -B'
    step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-
reports/TEST-*.xml'])
    if (currentBuild.result == "UNSTABLE") {
        // input "Unit tests are failing, proceed?"
        sh "exit 1"
    }
}

def allCodeQualityTests() {
    stage 'Code Quality'
    lintTest()
    coverageTest()
}
```

```
def lintTest() {
    context="continuous-integration/jenkins/linting"
    setBuildStatus ("${context}", 'Checking code conventions', 'PENDING')
    lintTestPass = true

    try {
        mvn 'verify -DskipTests=true'
    } catch (err) {
        setBuildStatus ("${context}", 'Some code conventions are broken',
'FAILURE')
        lintTestPass = false
    } finally {
        if (lintTestPass) setBuildStatus ("${context}", 'Code conventions OK',
'SUCCESS')
    }
}

def coverageTest() {
    context="continuous-integration/jenkins/coverage"
    setBuildStatus ("${context}", 'Checking code coverage levels', 'PENDING')

    coverageTestStatus = true

    try {
        mvn 'cobertura:check'
    } catch (err) {
        setBuildStatus("${context}", 'Code coverage below 90%', 'FAILURE')
        throw err
    }

    setBuildStatus ("${context}", 'Code coverage above 90%', 'SUCCESS')

}

def preview() {
    stage name: 'Deploy to Preview env', concurrency: 1
    def herokuApp = "${env.HEROKU_PREVIEW}"
    def id = createDeployment(getBranch(), "preview", "Deploying branch to
test")
    echo "Deployment ID: ${id}"
    if (id != null) {
        setDeploymentStatus(id, "pending",
"https://${herokuApp}.herokuapp.com/", "Pending deployment to test");
        herokuDeploy "${herokuApp}"
        setDeploymentStatus(id, "success",
"https://${herokuApp}.herokuapp.com/", "Successfully deployed to test");
    }
    mvn 'deploy -DskipTests=true'
}

def preProduction() {
    stage name: 'Deploy to Pre-Production', concurrency: 1
    switchSnapshotBuildToRelease()
    herokuDeploy "${env.HEROKU_PREPRODUCTION}"
    buildAndPublishToArtifactory()
}
```

```
def manualPromotion() {
    // we need a first milestone step so that all jobs entering this stage are
tracked an can be aborted if needed
    milestone 1
    // time out manual approval after ten minutes
    timeout(time: 10, unit: 'MINUTES') {
        input message: "Does Pre-Production look good?"
    }
    // this will kill any job which is still in the input step
    milestone 2
}

def production() {
    stage name: 'Deploy to Production', concurrency: 1
    step([$class: 'ArtifactArchiver', artifacts: '**/target/*.jar', fingerprint:
true])
    herokuDeploy "${env.HEROKU_PRODUCTION}"
    def version = getCurrentHerokuReleaseVersion("${env.HEROKU_PRODUCTION}")
    def createdAt = getCurrentHerokuReleaseDate("${env.HEROKU_PRODUCTION}",
version)
    echo "Release version: ${version}"
    createRelease(version, createdAt)
    promoteInArtifactoryAndDistributeToBinTray()
}

def switchSnapshotBuildToRelease() {
    def descriptor = Artifactory.mavenDescriptor()
    descriptor.version = '1.0.0'
    descriptor.pomFile = 'pom.xml'
    descriptor.transform()
}

def buildAndPublishToArtifactory() {
        def rtMaven = Artifactory.newMavenBuild()
        rtMaven.tool = "Maven 3.x"
        rtMaven.deployer releaseRepo:'libs-release-local', snapshotRepo:'libs-
snapshot-local', server: server
        rtMaven.resolver releaseRepo:'libs-release', snapshotRepo:'libs-
snapshot', server: server
        rtMaven.run pom: 'pom.xml', goals: 'install', buildInfo: buildInfo
        server.publishBuildInfo buildInfo
}

def promoteBuildInArtifactory() {
        def promotionConfig = [
            // Mandatory parameters
            'buildName'          : buildInfo.name,
            'buildNumber'        : buildInfo.number,
            'targetRepo'         : 'libs-prod-local',

            // Optional parameters
            'comment'            : 'deploying to production',
            'sourceRepo'         : 'libs-release-local',
            'status'             : 'Released',
            'includeDependencies': false,
            'copy'               : true,
```

```
            // 'failFast' is true by default.
            // Set it to false, if you don't want the promotion to abort upon
receiving the first error.
            'failFast'           : true
        ]

        // Promote build
        server.promote promotionConfig
}

def distributeBuildToBinTray() {
        def distributionConfig = [
            // Mandatory parameters
            'buildName'             : buildInfo.name,
            'buildNumber'           : buildInfo.number,
            'targetRepo'            : 'reading-time-dist',
            // Optional parameters
            //'publish'               : true, // Default: true. If true,
artifacts are published when deployed to Bintray.
            'overrideExistingFiles' : true, // Default: false. If true,
Artifactory overwrites builds already existing in the target path in Bintray.
            //'gpgPassphrase'         : 'passphrase', // If specified,
Artifactory will GPG sign the build deployed to Bintray and apply the specified
passphrase.
            //'async'                 : false, // Default: false. If true, the
build will be distributed asynchronously. Errors and warnings may be viewed in
the Artifactory log.
            //"sourceRepos"           : ["yum-local"], // An array of local
repositories from which build artifacts should be collected.
            //'dryRun'                : false, // Default: false. If true,
distribution is only simulated. No files are actually moved.
        ]
        server.distribute distributionConfig
}

def promoteInArtifactoryAndDistributeToBinTray() {
    stage ("Promote in Artifactory and Distribute to BinTray") {
        promoteBuildInArtifactory()
        distributeBuildToBinTray()
    }
}

def mvn(args) {
    withMaven(
        // Maven installation declared in the Jenkins "Global Tool
Configuration"
        maven: 'Maven 3.x',
        // Maven settings.xml file defined with the Jenkins Config File Provider
Plugin

        // settings.xml referencing the GitHub Artifactory repositories
         mavenSettingsConfig: '0e94d6c3-b431-434f-a201-7d7cda7180cb',
        // we do not need to set a special local maven repo, take the one from
the standard box
        //mavenLocalRepo: '.repository'
        ) {
        // Run the maven build
```

```
        sh "mvn $args -Dmaven.test.failure.ignore"
    }
}

def herokuDeploy (herokuApp) {
    withCredentials([[$class: 'StringBinding', credentialsId: 'HEROKU_API_KEY',
variable: 'HEROKU_API_KEY']]) {
        mvn "heroku:deploy -DskipTests=true -Dmaven.javadoc.skip=true -B -V -D
heroku.appName=${herokuApp}"
    }
}

def getRepoSlug() {
    tokens = "${env.JOB_NAME}".tokenize('/')
    org = tokens[tokens.size()-3]
    repo = tokens[tokens.size()-2]
    return "${org}/${repo}"
}

def getBranch() {
    tokens = "${env.JOB_NAME}".tokenize('/')
    branch = tokens[tokens.size()-1]
    return "${branch}"
}

def createDeployment(ref, environment, description) {
    withCredentials([[$class: 'StringBinding', credentialsId: 'GITHUB_TOKEN',
variable: 'GITHUB_TOKEN']]) {
        def payload = JsonOutput.toJson(["ref": "${ref}", "description":
"${description}", "environment": "${environment}", "required_contexts": []])
        def apiUrl = "https://api.github.com/repos/${getRepoSlug()}/deployments"
        def response = sh(returnStdout: true, script: "curl -s -H
\"Authorization: Token ${env.GITHUB_TOKEN}\" -H \"Accept: application/json\" -H
\"Content-type: application/json\" -X POST -d '${payload}' ${apiUrl}").trim()
        def jsonSlurper = new JsonSlurper()
        def data = jsonSlurper.parseText("${response}")
        return data.id
    }
}

void createRelease(tagName, createdAt) {
    withCredentials([[$class: 'StringBinding', credentialsId: 'GITHUB_TOKEN',
variable: 'GITHUB_TOKEN']]) {
        def body = "**Created at:** ${createdAt}\n**Deployment job:**
[${env.BUILD_NUMBER}](${env.BUILD_URL})\n**Environment:**
[${env.HEROKU_PRODUCTION}]
(https://dashboard.heroku.com/apps/${env.HEROKU_PRODUCTION})"
        def payload = JsonOutput.toJson(["tag_name": "v${tagName}", "name":
"${env.HEROKU_PRODUCTION} - v${tagName}", "body": "${body}"])
        def apiUrl = "https://api.github.com/repos/${getRepoSlug()}/releases"
        def response = sh(returnStdout: true, script: "curl -s -H
\"Authorization: Token ${env.GITHUB_TOKEN}\" -H \"Accept: application/json\" -H
\"Content-type: application/json\" -X POST -d '${payload}' ${apiUrl}").trim()
    }
}

void setDeploymentStatus(deploymentId, state, targetUrl, description) {
```

```
    withCredentials([[$class: 'StringBinding', credentialsId: 'GITHUB_TOKEN',
variable: 'GITHUB_TOKEN']]) {
        def payload = JsonOutput.toJson(["state": "${state}", "target_url":
"${targetUrl}", "description": "${description}"])
        def apiUrl =
"https://api.github.com/repos/${getRepoSlug()}/deployments/${deploymentId}/statu
ses"
        def response = sh(returnStdout: true, script: "curl -s -H
\"Authorization: Token ${env.GITHUB_TOKEN}\" -H \"Accept: application/json\" -H
\"Content-type: application/json\" -X POST -d '${payload}' ${apiUrl}").trim()
    }
}

void setBuildStatus(context, message, state) {
  // partially hard coded URL because of https://issues.jenkins-
ci.org/browse/JENKINS-36961, adjust to your own GitHub instance
  step([
      $class: "GitHubCommitStatusSetter",
      contextSource: [$class: "ManuallyEnteredCommitContextSource", context:
context],
      reposSource: [$class: "ManuallyEnteredRepositorySource", url:
"https://octodemo.com/${getRepoSlug()}"],
      errorHandlers: [[$class: "ChangingBuildStatusErrorHandler", result:
"UNSTABLE"]],
      statusResultSource: [ $class: "ConditionalStatusResultSource", results:
[[$class: "AnyBuildResult", message: message, state: state]] ]
  ]);
}

def getCurrentHerokuReleaseVersion(app) {
    withCredentials([[$class: 'StringBinding', credentialsId: 'HEROKU_API_KEY',
variable: 'HEROKU_API_KEY']]) {
        def apiUrl = "https://api.heroku.com/apps/${app}/dynos"
        def response = sh(returnStdout: true, script: "curl -s  -H
\"Authorization: Bearer ${env.HEROKU_API_KEY}\" -H \"Accept:
application/vnd.heroku+json; version=3\" -X GET ${apiUrl}").trim()
        def jsonSlurper = new JsonSlurper()
        def data = jsonSlurper.parseText("${response}")
        return data[0].release.version
    }
}

def getCurrentHerokuReleaseDate(app, version) {
    withCredentials([[$class: 'StringBinding', credentialsId: 'HEROKU_API_KEY',
variable: 'HEROKU_API_KEY']]) {
        def apiUrl = "https://api.heroku.com/apps/${app}/releases/${version}"
        def response = sh(returnStdout: true, script: "curl -s  -H
\"Authorization: Bearer ${env.HEROKU_API_KEY}\" -H \"Accept:
application/vnd.heroku+json; version=3\" -X GET ${apiUrl}").trim()
        def jsonSlurper = new JsonSlurper()
        def data = jsonSlurper.parseText("${response}")
        return data.created_at
    }
}
```

**Groovy code**

**Groovy 函数**

```
node {
    stage("Test") {
        test()
    }
}

def test() {
    echo "Start"
    sleep(5)
    echo "Stop"
}
```

**Ansi Color**

```
// This shows a simple build wrapper example, using the AnsiColor plugin.
node {
    // This displays colors using the 'xterm' ansi color map.
    ansiColor('xterm') {
        // Just some echoes to show the ANSI color.
        stage "\u001B[31mI'm Red\u001B[0m Now not"
    }
}
```

写文件操作

```
// This shows a simple example of how to archive the build output artifacts.
node {
    stage "Create build output"

    // Make the output directory.
    sh "mkdir -p output"

    // Write an useful file, which is needed to be archived.
    writeFile file: "output/usefulfile.txt", text: "This file is useful, need to
archive it."

    // Write an useless file, which is not needed to be archived.
    writeFile file: "output/uselessfile.md", text: "This file is useless, no
need to archive it."

    stage "Archive build output"
```

```
    // Archive the build output artifacts.
    archiveArtifacts artifacts: 'output/*.txt', excludes: 'output/*.md'
}
```

**modules 实现模块**

```
def modules = [
  'Java',
  'PHP',
  'Python',
  'Ruby'
]

node() {

  stage("checkout") {
    echo "checkout"
  }

  modules.each { module ->
    stage("build:${module}") {
      echo "${module}"
    }
  }
}
```

**docker**

```
node('master') {

    stage('Build') {
        docker.image('maven:3.5.0').inside {
            sh 'mvn --version'
        }
    }

    stage('Deploy') {
        if (env.BRANCH_NAME == 'master') {
            echo 'I only execute on the master branch'
        } else {
            echo 'I execute elsewhere'
        }
    }
}
```

**input**

```
node {
    stage('Git') {
        def branch = input message: 'input branch name for this job', ok: 'ok',
parameters: [string(defaultValue: 'master', description: 'branch name', name:
'branch')]
        echo branch
    }
}
```

```
node {
    stage('Git') {
        def result = input message: 'input branch name for this job', ok: 'ok',
parameters: [string(defaultValue: 'master', description: 'branch name', name:
'branch'), string(defaultValue: '', description: 'commit to switch', name:
'commit')]

        echo result.branch
        echo result.commit
    }
}

node {
    stage('Git') {
        def result = input message: 'input branch name for this job', ok: 'ok',
parameters: [string(defaultValue: 'master', description: 'branch name', name:
'branch'), string(defaultValue: '', description: 'commit to switch', name:
'commit')]

        sh "echo ${result.branch}"
        sh "echo ${result.commit}"
    }
}
```

**if** 条件判断

```
node {
    dir('/var/www') {
        stage('Git') {
            if(fileExists('project')) {
                dir('project') {
                    sh 'git fetch origin'
                    sh 'git checkout master'
                    sh 'git pull'
                }
```

```
            } else {
                sh 'git clone git@git.netkiller.cn:neo/project.git project'
            }
        }
    }
}
```

**Docker**

```
node {
        stage("Checkout") {
                checkout([$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],userRemoteConfigs: [[url: env.GIT_REPO_URL]]])
        }

        docker.image('ruby').inside {
                stage("Init") {
                        sh 'pwd && ls'
                        sh 'gem install rails'
                        // sh 'gem install ...'
                }
                stage("Test") {
                        sh 'ruby tc_simple_number.rb'
                }
                stage("Build") {
                        sh 'ruby --version'
                        archiveArtifacts artifacts: 'bin/*', fingerprint: true
                }
                stage("Deploy") {
                        sh 'rsync -auzv --delete *
www@host.netkiller.cn:/path/to/dir'
                }
        }
}
```

**conditionalSteps**

```
def projectName = 'myProject'

def jobClosure = {
  steps {
    conditionalSteps {
      condition {
        fileExists(projectName+'/target/test.jar', BaseDir.WORKSPACE)
      }
      runner('Fail')
      steps {
```

```
        batchFile('echo Found some tests')
      }
    }
  }
}

freeStyleJob('AAA-Test', jobClosure)
```

**nexus**

```
stage("Deploy") {
    nexusArtifactUploader artifacts: [
        [artifactId: 'java11', type: 'jar', file: 'target/java11.jar']
    ],
    groupId: 'org.springframework.samples',
    nexusUrl: 'netkiller.cn/repository/maven/',
    nexusVersion: 'nexus3',
    protocol: 'http',
    repository: 'maven',
    version: '2.0.0.BUILD'
}
```

## 设置环境变量

environment定义键值对的环境变量

```
// Declarative //
pipeline {
    agent any
    environment {
        CC = 'clang'
    }
    stages {
        stage('Example') {
            environment {
                DEBUG_FLAGS = '-g'
            }
            steps {
                sh 'printenv'
            }
        }
    }
}
```

```
// Declarative //
pipeline {
    agent any
    environment {
        CC = 'clang'
    }
    stages {
        stage('Example') {
            environment {
                AN_ACCESS_KEY = credentials('my-prefined-secret-text') ③
            }
            steps {
                sh 'printenv'
            }
        }
    }
}
```

系统环境变量

```
echo "Running ${env.BUILD_ID} on ${env.JENKINS_URL}"
```

```
${env.WORKSPACE}
println env.JOB_NAME
println env.BUILD_NUMBER
```

打印所有环境变量

```
node {
    echo sh(returnStdout: true, script: 'env')
    echo sh(script: 'env|sort', returnStdout: true)
    // ...
}
```

```
pipeline {
        agent any
    stages {
        stage('Environment Example') {
```

```
            steps {
              script{
                def envs = sh(returnStdout: true, script: 'env').split('\n')
                                envs.each { name  ->
                                println "Name: $name"
                                }
              }
            }
        }
    }
}
```

### agent

agent 指令指定整个管道或某个特定的stage的执行环境。它的参数可用使用：

```
agent:
any - 任意一个可用的agent
none - 如果放在pipeline顶层，那么每一个stage都需要定义自己的agent指令
label - 在jenkins环境中指定标签的agent上面执行，比如agent { label 'my-defined-label'
}
node - agent { node { label 'labelName' } } 和 label一样，但是可用定义更多可选项
docker - 指定在docker容器中运行
dockerfile - 使用源码根目录下面的Dockerfile构建容器来运行
```

#### label

```
        agent {
        label "java-8"
    }
```

#### docker

https://jenkins.io/doc/book/pipeline/docker/

添加 jenkins 用户到 docker 组

```
[root@localhost ~]# gpasswd -a jenkins docker
Adding user jenkins to group docker

[root@localhost ~]# cat /etc/group | grep ^docker
```

```
docker:x:993:jenkins
```

**指定docker 镜像**

```
pipeline {
    agent { docker { image 'maven:3.3.3' } }
    stages {
        stage('build') {
            steps {
                sh 'mvn --version'
            }
        }
    }
}
```

```
pipeline {
    agent { docker { image 'php' } }
    stages {
        stage('build') {
            steps {
                sh 'php --version'
            }
        }
    }
}

pipeline {
    agent {
        docker { image 'php:latest' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'php --version'
            }
        }
    }
}
```

**args 参数**

挂在 /root/.m2 目录

```
```

```
pipeline {
    agent {
        docker {
            image 'maven:latest'
            args '-v $HOME/.m2:/root/.m2'
        }
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn -B'
            }
        }
    }
}
```

**Docker outside of Docker (DooD)**

```
        docker.image('maven').inside("-v
/var/run/docker.sock:/var/run/docker.sock -v /usr/bin/docker:/usr/bin/docker") {
                sh 'docker images'
        }
```

挂在宿主主机目录

```
node {
    stage("Checkout") {
        checkout(
          [$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
        )
                sh 'pwd'
    }
        docker.image('maven:latest').inside("-v /root/.m2:/root/.m2") {
        stage("Build") {
                sh 'java -version'
                sh 'mvn package -Dmaven.test.failure.ignore -
Dmaven.test.skip=true'
                archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
        }

        stage("Test") {
                sh 'java -jar target/webflux-0.0.1-SNAPSHOT.jar &'
                sleep 20
                sh 'mvn test -Dmaven.test.failure.ignore'
                junit 'target/surefire-reports/*.xml'
        }
```

```
    }
}
```

构建镜像

```
node {
    checkout scm

    docker.withRegistry('http://hub.netkiller.cn:5000') {

        def customImage = docker.build("project/api:1.0")

        /* Push the container to the custom Registry */
        customImage.push()
    }
}
```

容器内运行脚本

```
node {
    checkout scm

    def customImage = docker.build("my-image:${env.BUILD_ID}")

    customImage.inside {
        sh 'make test'
    }
}
```

```
                dir ('example') {
            /* 构建镜像 */
            def customImage = docker.build("example-
group/example:${params.VERSION}")

            /* hub.netkiller.cn是你的Docker Registry */
            docker.withRegistry('https://hub.netkiller.cn/', 'docker-registry')
{
                /* Push the container to the custom Registry */
                // push 指定版本
                customImage.push('latest')
            }
        }
```

```
stage('DockerBuild') {
    sh """
    rm -f src/docker/*.jar
    cp target/*.jar src/docker/*.jar
    """

    dir ("src/docker/") {
        def image = docker.build("your/demo:1.0.0")
        image.push()
    }
}
```

**Dockerfile**

创建 Dockerfile 文件

```
FROM node:7-alpine

RUN apk add -U subversion
```

创建 Jenkinsfile 文件

```
// Jenkinsfile (Declarative Pipeline)
pipeline {
    agent { dockerfile true }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
                sh 'svn --version'
            }
        }
    }
}
```

## Steps

**parallel 平行执行**

```
        stage('test') {
```

```
      parallel {
        stage('test') {
          steps {
            echo 'hello'
          }
        }
        stage('test1') {
          steps {
            sleep 1
          }
        }
        stage('test2') {
          steps {
            retry(count: 5) {
              echo 'hello'
            }

          }
        }
      }
```

**echo**

```
    stage('Deploy') {
        echo 'Deploying....'
    }
```

**catchError 捕获错误**

```
node {
    catchError {
        sh 'might fail'
    }
    step([$class: 'Mailer', recipients: 'admin@somewhere'])
}


        stage('teatA') {
      steps {
        catchError() {
          sh 'make'
        }

        mail(subject: 'Test', body: 'aaaa', to: 'netkiller@msn.com')
      }
    }
```

睡眠

```
node {
    sleep 10
    echo 'Hello World'
}
```

```
sleep(time:3,unit:"SECONDS")
```

限制执行时间

```
                stage('enforce') {
          steps {
            timeout(activity: true, time: 1) {
              echo 'test'
            }

          }
        }
```

时间截

```
        stage('timestamps') {
          steps {
            timestamps() {
              echo 'test'
            }
          }
    }
```

## 版本控制

**checkout**

https://github.com/jenkinsci/workflow-scm-step-plugin/blob/master/README.md

下面配置适用与 Webhook 方式

```
        stage('checkout') {
      steps {
        checkout(scm: [$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
                            userRemoteConfigs: [[url: env.GIT_REPO_URL]]],
changelog: true, poll: true)
      }
    }
```

从 URL 获取代码

```
node {
   checkout ([$class: 'GitSCM', branches: [[name: '*/master']],
doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],
userRemoteConfigs: [[credentialsId: '', url:
'https://github.com/bg7nyt/java.git']]])
}
```

**Git**

```
                stage('Git') {
                steps {
                git(url: 'https://git.dev.tencent.com/netkiller/java.git',
branch: 'master', changelog: true, poll: true)
                }
        }
```

## 节点与过程

**sh**

```
                stage("build") {
            steps {
                sh "mvn package -Dmaven.test.skip=true"
            }
        }
```

```
                steps {
                script{
                sh 'find /etc/'
                        }
                }
```

**例 2.1. Shell Docker 示例**

Shell Docker 使用 docker 命令完成构建过程

```
registryUrl='127.0.0.1:5000'        # 私有镜像仓库地址
imageName='netkiller/project'   # 镜像名称
imageTag=$BRANCH                    # 上面设置Branch分支，这里可以当做环境变量使用

echo ' >>> [INFO] enter workspace ...'

cd $WORKSPACE/                            # 进入到jenkins的工作区，jenkins会将gitlab
仓库代码pull到这里，用于制作镜像文件

# 根据不同的Branch生成不同的swoft的配置文件，区分测试还是生成等
echo ' >>> [INFO] create startup.sh ...'
(
cat << EOF

# 启动 Shell 写在此处

EOF
) > ./entrypoint.sh

# 生成 Dockerfile
echo ' >>> [INFO] begin create Dockerfile ...'
rm -f ./Dockerfile
(
cat << EOF
FROM netkiller/base
LABEL maintainer=netkiller@msn.com

COPY . /var/www/project
WORKDIR /var/www/project
EXPOSE 80
ENV PHP_ENVIRONMENT $BRANCH
ENTRYPOINT [ "bash", "/var/www/project/entrypoint.sh" ]
EOF
) > ./Dockerfile


#删除无用镜像
```

```
echo ' >>> [INFO] begin cleaning image ...'
for image in `docker images | grep -w $imageName | grep -i -w $imageTag | awk
'{print $3}'`
do
    docker rmi $image -f
done

#制作镜像
echo ' >>> [INFO] begin building image ...'
docker build --tag $imageName:$imageTag --rm .

#给镜像打标签
img=`docker images | grep -w $imageName | grep -i -w $imageTag | awk '{print
$3}'`
docker tag $img $registryUrl/$imageName:$imageTag

#push到私有镜像仓库
echo ' >>> [INFO] begin publishing image ...'
docker push $registryUrl/$imageName:$imageTag

#删除刚刚制作的镜像，释放存储空间
echo ' >>> [INFO] cleaning temporary building ...'
docker rmi -f $imageName:$imageTag
docker rmi -f $registryUrl/$imageName:$imageTag
```

**Windows 批处理脚本**

```
        stage('bat') {
        steps {
            bat(returnStatus: true, returnStdout: true, label: 'aa', encoding:
'utf-8', script: 'dir')
        }
    }
```

分配工作空间

```
        stage('alocate') {
        steps {
            ws(dir: 'src') {
                echo 'aaa'
            }

        }
    }
```

**node**

```
        stage('node') {
        steps {
            node(label: 'java-8') {
              sh 'mvn package'
            }

        }
        }
```

## 工作区

变更目录

```
        stage('subtask') {
        steps {
            dir(path: '/src') {
                echo 'begin'
                sh '''mvn test'''
                echo 'end'
                }
                }
        }
```

判断文件是否存在

```
        stage('exists') {
          steps {
            fileExists '/sss'
          }
    }
```

```
def exists = fileExists 'file'

if (exists) {
    echo 'Yes'
} else {
    echo 'No'
}
```

```
if (fileExists('file')) {
    echo 'Yes'
} else {
    echo 'No'
}
```

```
pipeline{
        agent any
        stages{
      stage("Checkout") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

                stage("fileExists") {
                        steps{
                echo pwd()
                sh 'ls -1'
                                script {
                                        def exists = fileExists 'README.md'
                    if (exists) {
                        echo 'Yes'
                    } else {
                        echo 'No'
                    }
                                }
                        }
                }
        }
}
```

分配工作区

```
    stage('alocate') {
        steps {
            ws(dir: 'src') {
                echo 'aaa'
```

```
            }

        }
    }
```

清理工作区

```
        stage('test') {
        steps {
            cleanWs(cleanWhenAborted: true, cleanWhenFailure: true,
cleanWhenNotBuilt: true, cleanWhenSuccess: true, cleanWhenUnstable: true,
cleanupMatrixParent: true, deleteDirs: true, disableDeferredWipeout: true,
notFailBuild: true, skipWhenFailed: true, externalDelete: '/aa')
        }
    }
```

递归删除目录

```
        stage('deldir') {
          steps {
            deleteDir()
          }
    }
```

写文件

```
                stage('write') {
          steps {
            writeFile(file: 'hello.txt', text: 'Helloworld')
          }
        }
```

读文件

```
        stage('read') {
          steps {
            readFile 'hello.txt'
```

```
            }
        }
```

## 4. Jenkins Job DSL / Plugin

```
def gitUrl = 'git://github.com/jenkinsci/job-dsl-plugin.git'

job('PROJ-unit-tests') {
    scm {
        git(gitUrl)
    }
    triggers {
        scm('*/15 * * * *')
    }
    steps {
        maven('-e clean test')
    }
}

job('PROJ-sonar') {
    scm {
        git(gitUrl)
    }
    triggers {
        cron('15 13 * * *')
    }
    steps {
        maven('sonar:sonar')
    }
}

job('PROJ-integration-tests') {
    scm {
        git(gitUrl)
    }
    triggers {
        cron('15 1,13 * * *')
    }
    steps {
        maven('-e clean integration-test')
    }
}

job('PROJ-release') {
```

```
    scm {
        git(gitUrl)
    }
    // no trigger
    authorization {
        // limit builds to just Jack and Jill
        permission('hudson.model.Item.Build', 'jill')
        permission('hudson.model.Item.Build', 'jack')
    }
    steps {
        maven('-B release:prepare release:perform')
        shell('cleanup.sh')
    }
}
```

```
job('PROJ-unit-tests') {
    scm {
        git('https://github.com/bg7nyt/java.git')
    }
    triggers {
        scm('*/15 * * * *')
    }
    steps {
        maven('-e clean test')
    }
}
```

## 5. Jenkins Plugin

### Blue Ocean

https://jenkins.io/doc/book/blueocean/getting-started/

http://jk.netkiller.cn/blue/

### Locale Plugin (国际化插件)

安装Locale Plugin, 重启生效。

```
配置【Manage Jenkins】>【Configure System】> 【Locale】
```

Default Language 填写 zh_CN，勾选忽略浏览器设置强制设置语言

### github-plugin 插件

https://github.com/jenkinsci/github-plugin

```
git clone https://github.com/jenkinsci/github-plugin.git
mkdir target/classes
```

修改 rest-assured 去掉 exclusions 配置项

```
        <dependency>
            <groupId>com.jayway.restassured</groupId>
            <artifactId>rest-assured</artifactId>
            <!--1.7.2 is the last version that use a compatible groovy version-
->
            <version>1.7.2</version>
            <scope>test</scope>
```

```
            <exclusions>
                <exclusion>
                    <groupId>org.apache.httpcomponents</groupId>
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
```

编译插件

```
[root@netkiller github-plugin]# mvn hpi:hpi
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building GitHub plugin 1.29.4-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-hpi-plugin:1.120:hpi (default-cli) @ github ---
[INFO] Generating /srv/github-plugin/target/github/META-INF/MANIFEST.MF
[INFO] Checking for attached .jar artifact ...
[INFO] Generating jar /srv/github-plugin/target/github.jar
[INFO] Building jar: /srv/github-plugin/target/github.jar
[INFO] Exploding webapp...
[INFO] Copy webapp webResources to /srv/github-plugin/target/github
[INFO] Assembling webapp github in /srv/github-plugin/target/github
[INFO] Generating hpi /srv/github-plugin/target/github.hpi
[INFO] Building jar: /srv/github-plugin/target/github.hpi
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.161s
[INFO] Finished at: Mon Jan 07 12:03:45 CST 2019
[INFO] Final Memory: 29M/290M
[INFO] ------------------------------------------------------------------------
```

进入 github --> Settings --> Developer settings --> Personal Access Token --> Generate new token

repo 和 admin:repo_hook

Settings -> Webhooks -> Add webhook

系统管理 --> 系统设置 --> GitHub --> Add GitHub Sever

**Docker**

**This plugin integrates Jenkins with Docker**

https://jenkins.io/doc/book/pipeline/docker/

```
vim /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H fd://

改为

ExecStart=/usr/bin/dockerd -H fd:// -H unix:///var/run/docker.sock -H
tcp://0.0.0.0:2375
```

吧 jenkins 用户添加到 docker 组

```
gpasswd -a jenkins docker
```

重启 docker

```
systemctl daemon-reload
systemctl restart docker

如果是 Docker 方式运行 Jenkins 需要启动 jenkins
docker start jenkins
```

参考例子

```
root@ubuntu:~# cat /lib/systemd/system/docker.service
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues
still
# exists and systemd currently does not support the cgroup feature set
```

```
required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H unix:///var/run/docker.sock -H
tcp://0.0.0.0:2375
ExecReload=/bin/kill -s HUP $MAINPID
LimitNOFILE=1048576
# Having non-zero Limit*s causes performance problems due to accounting
overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNPROC=infinity
LimitCORE=infinity
# Uncomment TasksMax if your systemd version supports it.
# Only systemd 226 and above support this version.
TasksMax=infinity
TimeoutStartSec=0
# set delegate yes so that systemd does not reset the cgroups of docker
containers
Delegate=yes
# kill only the docker process, not all processes in the cgroup
KillMode=process
# restart the docker process if it exits prematurely
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

[Install]
WantedBy=multi-user.target
```

**设置 Docker 主机和代理**

输入 Docker 主机的IP地址，类似 tcp://172.16.0.10:2375

Cloud

Docker

Name: docker

Docker Host URI: tcp://127.0.0.1:2375

Server credentials: - none - Add

Advanced...

Version = 18.09.1, API Version = 1.39

Test Connection

Cloud

Docker

Name: docker

Docker Cloud details...

Docker Agent templates...

Delete cloud

Add a new cloud

Cloud

Docker

Name: docker

Docker Host URI: tcp://127.0.0.1:2375

Server credentials: - none - Add

Advanced...

Version = 18.09.1, API Version = 1.39

Test Connection

Enabled

Error Duration

Default = 300

Expose DOCKER_HOST

Container Cap: 100

Docker Agent templates: Add Docker Template

List of Images to be launched as slaves

Delete cloud

Cloud

Docker

Name: docker

Docker Cloud details...

Docker Agent templates

Docker Agent templates

Labels: java

Enabled

Name: docker

Docker Image: maven

Registry Authentication...

Save Apply

持久化

例如持续集成过程中，我们不希望每次都从 maven 镜像下载编译依赖的包，或者构建物我们需要永久保留等等，这时就需要做持久化

例如我们将宿主主机的 /opt/maven 挂载到 Docker 容器的 /root/.m2 目录。这样就实现了 maven 的持久化。只需写入 /opt/maven:/root/.m2 即可

当持续机构运行完毕 docker 容器被清理，但是 /opt/maven 并不会被清理，下次构建时，在将它挂载到 /root/.m2 即可。

## JaCoCo

**This plugin integrates JaCoCo code coverage reports to Jenkins.**

https://jenkins.io/doc/pipeline/steps/jacoco/

**Pipeline**

```
stage('Build') {
     steps {
        sh 'mvn test'
        junit '*/build/test-results/*.xml'
        step( [ $class: 'JacocoPublisher' ] )
     }
}
```

配置jacoco

```
The jacoco pipeline step configuration uses this format:

step([$class: 'JacocoPublisher',
      execPattern: 'target/*.exec',
      classPattern: 'target/classes',
      sourcePattern: 'src/main/java',
      exclusionPattern: 'src/test*'
])
Or with a simpler syntax for declarative pipeline:

jacoco(
      execPattern: 'target/*.exec',
      classPattern: 'target/classes',
      sourcePattern: 'src/main/java',
      exclusionPattern: 'src/test*'
)
```

完整的例子

```
node {
    stage('Checkout') {
        git 'https://github.com/bg7nyt/junit4-jacoco.git'
    }
    stage('Build') {
        sh "mvn -Dmaven.test.failure.ignore clean package"
    }
    stage('Test') {
        sh "mvn test"
    }
    stage('Results') {
        junit '**/target/surefire-reports/TEST-*.xml'
        archive 'target/*.jar'
        step( [ $class: 'JacocoPublisher' ] )
    }
}
```

### SSH Pipeline Steps

使用说明: https://github.com/jenkinsci/ssh-steps-plugin#pipeline-steps

```
!groovy
def getHost(){
    def remote = [:]
    remote.name = 'mysql'
```

```
    remote.host = '192.168.8.108'
    remote.user = 'root'
    remote.port = 22
    remote.password = 'qweasd'
    remote.allowAnyHosts = true
    return remote
}
pipeline {
    agent {label 'master'}
    environment{
        def server = ''
    }
    stages {
        stage('init-server'){
            steps {
                script {
                   server = getHost()
                }
            }
        }
        stage('use'){
            steps {
                script {
                  sshCommand remote: server, command: """
                  if test ! -d aaa/ccc ;then mkdir -p aaa/ccc;fi;cd aaa/ccc;rm
-rf ./*;echo 'aa' > aa.log
                  """
                }
            }
        }
    }
}
#####################################
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    sshCommand remote: remote, command: "ls -lrt"
    sshCommand remote: remote, command: "for i in {1..5}; do echo -n \"Loop \$i
\"; date ; sleep 1; done"
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    writeFile file: 'abc.sh', text: 'ls -lrt'
    sshScript remote: remote, script: "abc.sh"
```

```
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    writeFile file: 'abc.sh', text: 'ls -lrt'
    sshPut remote: remote, from: 'abc.sh', into: '.'
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    sshGet remote: remote, from: 'abc.sh', into: 'abc_get.sh', override: true
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    sshRemove remote: remote, path: "abc.sh"
  }
}
def remote = [:]
remote.name = "node-1"
remote.host = "10.000.000.153"
remote.allowAnyHosts = true
node {
    withCredentials([sshUserPrivateKey(credentialsId: 'sshUser',
keyFileVariable: 'identity', passphraseVariable: '', usernameVariable:
'userName')]) {
        remote.user = userName
        remote.identityFile = identity
        stage("SSH Steps Rocks!") {
            writeFile file: 'abc.sh', text: 'ls'
            sshCommand remote: remote, command: 'for i in {1..5}; do echo -n
\"Loop \$i \"; date ; sleep 1; done'
            sshPut remote: remote, from: 'abc.sh', into: '.'
            sshGet remote: remote, from: 'abc.sh', into: 'bac.sh', override:
true
            sshScript remote: remote, script: 'abc.sh'
            sshRemove remote: remote, path: 'abc.sh'
        }
```

```
    }
}
################################
```

### Rancher

https://plugins.jenkins.io/rancher

https://jenkins.io/doc/pipeline/steps/rancher/

创建 Rancher API

在Jenkins的Credentials中添加一个类型为Username with password的认证，username和password分别对应于上一步生成的Access Key和Secret Key，如下图

然后在语法生成器中，找到rancher进行如下图的配置：

设置 environmentId ，找到你的集群，点击进入，看到URL https://rancher.netkiller.cn/v3/clusters/c-mx88f，“c-mx88f” 就是 environmentId

```
stage('Rancher') {
    rancher confirm: false, credentialId: 'b56bd9b2-3277-4072-baae-
08d73aa26549', endpoint: 'https://rancher.netkiller.cn/v2', environmentId:
'test', environments: '', image: '*/demo:1.0.0', ports: '', service:
'jenkins/demo', timeout: 50
}
```

### Kubernetes 插件

#### Kubernetes

https://plugins.jenkins.io/kubernetes

https://github.com/jenkinsci/kubernetes-plugin

```
def label = "mypod-${UUID.randomUUID().toString()}"
podTemplate(label: label) {
    node(label) {
        stage('Run shell') {
            sh 'echo hello world'
        }
    }
```

```
}
```

**Kubernetes :: Pipeline :: Kubernetes Steps**

```

```

**Kubernetes Continuous Deploy**

https://plugins.jenkins.io/kubernetes-cd

**Kubernetes Cli**

:

## HTTP Request Plugin

```
GET https://<rancher_server>/v3/project/<project_id>/workloads/deployment:
<rancher_namespace>:<rancher_service> # 获取一个服务的详细信息
GET https://<rancher_server>/v3/project/<project_id>/pods/?
workloadId=deployment:<rancher_namespace>:<rancher_service> # 获取服务的所有容器信
息
DELETE
https://<rancher_server>/v3/project/<project_id>/pods/<rancher_namespace>:
<container_name> # 根据容器名删除容器
PUT https://<rancher_server>/v3/project/<project_id>/workloads/deployment:
<rancher_namespace>:<rancher_service> # 更新服务
```

```
// 查询服务信息
def response = httpRequest acceptType: 'APPLICATION_JSON', authentication:
"${RANCHER_API_KEY}", contentType: 'APPLICATION_JSON', httpMode: 'GET',
responseHandle: 'LEAVE_OPEN', timeout: 10, url:
"${rancherUrl}/workloads/deployment:${rancherNamespace}:${rancherService}"
def serviceInfo = new JsonSlurperClassic().parseText(response.content)
response.close()

def dockerImage = imageName+":"+imageTag
if (dockerImage.equals(serviceInfo.containers[0].image)) {
    // 如果镜像名未改变，直接删除原容器
    // 查询容器名称
    response = httpRequest acceptType: 'APPLICATION_JSON', authentication:
"${RANCHER_API_KEY}", contentType: 'APPLICATION_JSON', httpMode: 'GET',
```

```
responseHandle: 'LEAVE_OPEN', timeout: 10, url: "${rancherUrl}/pods/?
workloadId=deployment:${rancherNamespace}:${rancherService}"
    def podsInfo = new JsonSlurperClassic().parseText(response.content)
    def containerName = podsInfo.data[0].name
    response.close()
    // 删除容器
    httpRequest acceptType: 'APPLICATION_JSON', authentication:
"${RANCHER_API_KEY}", contentType: 'APPLICATION_JSON', httpMode: 'DELETE',
responseHandle: 'NONE', timeout: 10, url:
"${rancherUrl}/pods/${rancherNamespace}:${containerName}"

} else {
    // 如果镜像名改变，使用新镜像名更新容器
    serviceInfo.containers[0].image = dockerImage
    // 更新
    def updateJson = new JsonOutput().toJson(serviceInfo)
    httpRequest acceptType: 'APPLICATION_JSON', authentication:
"${RANCHER_API_KEY}", contentType: 'APPLICATION_JSON', httpMode: 'PUT',
requestBody: "${updateJson}", responseHandle: 'NONE', timeout: 10, url:
"${rancherUrl}/workloads/deployment:${rancherNamespace}:${rancherService}"
}
```

### Skip Certificate Check plugin

https://wiki.jenkins.io/display/JENKINS/Skip+Certificate+Check+plugin

```
[root@localhost ~]# tail -f /var/log/jenkins/jenkins.log

javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated
```

### Android Sign Plugin

Android Sign Plugin 依赖 Credentials Plugin，因为 Credentials Plugin 只支持 PKCS#12 格式的证书，所以先需要将生成好的 JKS 证书转换为 PKCS#12 格式：

```
keytool -importkeystore -srckeystore netkiller.jks -srcstoretype JKS -
deststoretype PKCS12 -destkeystore netkiller.p12
```

复制代码添加类型为 credential，选择上传证书文件，将 PKCS#12 证书上传到并配置好 ID，本项目中使用了 ANDROID_SIGN_KEY_STORE 作为 ID。

```
pipeline {
    ...

    stages {
        ...

        stage("Sign APK") {
            steps {
                echo 'Sign APK'
                signAndroidApks(
                    keyStoreId: "ANDROID_SIGN_KEY_STORE",
                    keyAlias: "tomczhen",
                    apksToSign: "**/*-prod-release-unsigned.apk",
                    archiveSignedApks: false,
                    archiveUnsignedApks: false
                )
            }
        }

        ...
    }

    ...
}
```

## 6. Jenkinsfile Pipeline Example

### Maven 子模块范例

Maven 子模块创建方法
https://www.netkiller.cn/java/build/maven.html#maven.module

目录结构

```
Project
    |
    |--- common (Shared)
    |      | ---pom.xml
    |--- project1 (depend common)
    |      |--- pom.xml
    |--- project2 (depend common)
    |      |--- pom.xml
    |---pom.xml
```

构建 父项目

```
pipeline {
    agent {
        label "default"
    }
    stages  {

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
```

```
            }
        }

        stage("构建") {
            steps {
                echo "构建中..."
                sh 'mvn package -Dmaven.test.skip=true' // mvn
示例
                archiveArtifacts artifacts: '**/target/*.jar',
fingerprint: true // 收集构建产物
                echo "构建完成."
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
                // 请在这里放置您项目代码的单元测试调用过程，例如：
                sh 'mvn test' // mvn 示例
                echo "单元测试完成."
                junit '**/target/surefire-reports/*.xml' // 收集
单元测试报告的调用过程
            }
        }

        stage("部署") {
            steps {
                echo "部署中..."
                echo "部署完成"
            }
        }
    }
}
```

构建共享项目

```
pipeline {
    agent {
        label "default"
    }
    stages  {
```

```
        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

        stage("构建") {
            steps {
                echo "构建中..."
                dir(path: 'common') {
                        sh 'mvn package -Dmaven.test.skip=true'
// mvn 示例
                        archiveArtifacts artifacts:
'**/target/*.jar', fingerprint: true // 收集构建产物
                }
                echo "构建完成."
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
                sh 'mvn test' // mvn 示例
                echo "单元测试完成."
                junit 'target/surefire-reports/*.xml' // 收集单
元测试报告的调用过程
            }
        }

        stage("部署") {
            steps {
                echo "部署中..."
                dir(path: 'common') {
                        sh 'mvn install'
                }
                echo "部署完成"
            }
        }
    }
}
```

构建 project1 和 project2

```
pipeline {
    agent {
        label "default"
    }
    stages  {

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }
                stage("共享库") {
            steps {
                echo "构建中..."
                dir(path: 'common') {
                        sh 'mvn install -Dmaven.test.skip=true'
// mvn 示例
                        archiveArtifacts artifacts:
'**/target/*.jar', fingerprint: true // 收集构建产物
                }
                echo "构建完成."
            }
        }
        stage("构建") {
            steps {
                echo "构建中..."
                    dir(path: 'project1') {
                    sh 'mvn package -Dmaven.test.skip=true' //
mvn 示例
                            archiveArtifacts artifacts:
'**/target/*.jar', fingerprint: true // 收集构建产物
                }
                echo "构建完成."
```

```
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
                sh 'mvn test' // mvn 示例
                echo "单元测试完成."
                junit 'target/surefire-reports/*.xml' // 收集单
元测试报告的调用过程
            }
        }

        stage("部署") {
            steps {
                echo "部署中..."
                // 部署脚本
                echo "部署完成"
            }
        }
    }
}
```

## 使用指定镜像构建

```
pipeline {
    agent any
    stages {
        stage("Checkout") {
            steps {
                sh 'ci-init'
                checkout(
                        [$class           : 'GitSCM', branches:
[[name: env.GIT_BUILD_REF]],
                         userRemoteConfigs: [[url:
env.GIT_REPO_URL]]]
                )
            }
        }
```

```
        stage("Compile") {

            // 构建的 docker 镜像
            agent {
                docker { image 'maven' }
            }

            steps {
                echo "构建中..."
                sh 'mvn -v'
                sh 'mvn compile'
            }
        }


        stage('Test') {

            agent {
                docker { image 'maven' }
            }

            steps {
                echo '单元测试...'
                sh 'mvn test'
                junit 'target/surefire-reports/*.xml'
            }

        }

        stage("Deploy") {
            steps {
                echo "部署中..."
                echo "部署完成"
            }
        }
    }
}
```

## 命令行制作 Docker 镜像

```
pipeline {
    agent any
    stages {

        stage('Build') {

            steps {
                echo '编译中...'
                // 编译 docker 镜像
                sh "docker build $tag $contextPath"
            }

        }

        stage('Push Image') {

            steps {

                sh "echo ${REGISTRY_PASS} | docker login -u
${REGISTRY_USER} --password-stdin ${REGISTRY_URL}"
                sh "docker tag ${image} ${registry_image}"
                sh "docker push ${registry_image}"

            }

        }

    }
}
```

```
pipeline {

    agent any

    stages {
        stage("Checkout") {
          steps {
            checkout([
              $class: 'GitSCM',
```

```
              branches: [[name: env.GIT_COMMIT]],
              extensions: [[$class: 'PruneStaleBranch']],
              userRemoteConfigs: [[
                url: env.GIT_REPO_URL,
                refspec: "+refs/heads/*:refs/remotes/origin/*"
              ]]
            ])

            sh '''
                #!/bin/bash
                echo ${GIT_COMMIT}
                echo ${REF}
                echo ${GIT_LOCAL_BRANCH}
            '''
          }
        }

        stage('Build') {
            steps{
                echo "Building begin"
                script{
                    // 设置镜像名
                    env.BUILD_MODULE = "common"
                    env.DOCKER_IMAGE_TAG = env.BUILD_MODULE +
':' + env.GIT_COMMIT
                    env.DOCKER_REMOTE_IMAGE_TAG =
"${env.REGISTRY_URL}/${env.DOCKER_IMAGE_TAG}"

                    sh "docker login ${DOCKER_REGISTER_URL} -u
${DOCKER_REPOSITORY_USERNAME} -p ${DOCKER_REPOSITORY_PASSWORD}"

                    def statusCode = sh(script:"docker pull
${DOCKER_REMOTE_IMAGE_TAG}", returnStatus:true)

                                            // 判断该镜像在仓库是否存在
                    if (statusCode != 0) {

                        sh '''
                        #!/bin/bash

                        # build docker image
                        docker build . -f Dockerfile -t
${DOCKER_IMAGE_TAG}

                        # tag docker image
```

```
                        docker tag ${DOCKER_IMAGE_TAG}
${DOCKER_REMOTE_IMAGE_TAG}

                    }
                }
                echo "Build end"
            }
        }

        stage('Deploy') {
            steps{
                echo "Deploying begin"
                script{
                        # push to
                        docker push ${DOCKER_REMOTE_IMAGE_TAG}

                        # rm
                        docker rmi ${DOCKER_IMAGE_TAG}
                        docker rmi ${DOCKER_REMOTE_IMAGE_TAG}
                        '''
                }
                echo "Deploy end"
            }
        }
    }
}
```

## Yarn

```
pipeline {
    agent {
        label "default"
    }
    stages  {

        stage("検出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
```

```
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }
                stage("环境") {
                steps {
                sh 'apt install -y apt-transport-https'
                sh "curl -sS
https://dl.yarnpkg.com/debian/pubkey.gpg | apt-key add -"
                sh 'echo "deb https://dl.yarnpkg.com/debian/
stable main" | tee /etc/apt/sources.list.d/yarn.list'
                sh 'cat /etc/apt/sources.list.d/yarn.list'
                sh 'apt update && apt install -y yarn'
                sh 'yarn --version'
            }
        }
        stage("构建") {
            steps {
                echo "构建中..."
                sh 'yarn add webpack'
                sh 'node -v'
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
            }
        }

        stage("部署") {
            steps {
                // sh './deploy.sh'
            }
        }
    }
}
```

**Android**

进入项目目录，找到 local.properties 文件，打开文件

```
## This file is automatically generated by Android Studio.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file should *NOT* be checked into Version Control
Systems,
# as it contains information specific to your local
configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please
read the
# header note.
sdk.dir=/Users/neo/Library/Android/sdk
```

sdk.dir 是 Android SDK 存放目录，进入该目录

```
neo@MacBook-Pro ~ % ll /Users/neo/Library/Android/sdk/
total 0
drwxr-xr-x   3 neo  staff    96B Oct 23 09:56 build-tools
drwxr-xr-x  18 neo  staff   576B Oct 23 09:55 emulator
drwxr-xr-x   6 neo  staff   192B Oct 23 10:21 extras
drwxr-xr-x   3 neo  staff    96B Oct 23 11:35 fonts
drwxr-xr-x   4 neo  staff   128B Oct 23 11:00 licenses
drwxr-xr-x   3 neo  staff    96B Oct 23 09:55 patcher
drwxr-xr-x  19 neo  staff   608B Oct 23 09:56 platform-tools
drwxr-xr-x   4 neo  staff   128B Oct 23 10:23 platforms
drwxr-xr-x  24 neo  staff   768B Oct 23 10:57 skins
drwxr-xr-x   4 neo  staff   128B Oct 23 10:23 sources
drwxr-xr-x   4 neo  staff   128B Oct 24 15:06 system-images
drwxr-xr-x  14 neo  staff   448B Oct 23 09:55 tools

neo@MacBook-Pro ~ % ll /Users/neo/Library/Android/sdk/licenses
total 16
-rw-r--r--  1 neo  staff    41B Oct 23 10:23 android-sdk-
license
-rw-r--r--  1 neo  staff    41B Oct 23 11:00 android-sdk-
preview-license
```

```
neo@MacBook-Pro ~ % cat
/Users/neo/Library/Android/sdk/licenses/android-sdk-license

d56f5187479451eabf01fb78af6dfcb131a6481e
```

/Users/neo/Library/Android/sdk/licenses/android-sdk-license 便是当前 Android SDK License 文件

如果你安装了多个版本的 SDK，例如 android-26， android-27，android-28 可以看到三行字串。

```
24333f8a63b6825ea9c5514f83c2829b004d1fee 这是 Android 8.0 -
android-26
d56f5187479451eabf01fb78af6dfcb131a6481e 这是 Android 9.0 -
android-28
```

```
pipeline {
        agent any
    stages  {

        stage("Checkout") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

        stage("Android SDK") {
            steps {
                script{
                        if (fileExists('sdk-tools-linux-
4333796.zip')) {
                                          echo 'Android SDK 已安
```

```
装'
                                        } else {
                                        echo '安装 Android SDK'

                                sh '''
# rm -rf sdk-tools-linux-4333796.* tools platforms platform-
tools
wget https://dl.google.com/android/repository/sdk-tools-linux-
4333796.zip
unzip sdk-tools-linux-4333796.zip
                                        '''

                                sh 'yes|tools/bin/sdkmanager --
licenses'
                                //sh 'yes|tools/bin/sdkmanager
"platform-tools" "build-tools;26.0.3" "platforms;android-26"'
// andorid 8.0
                                //sh 'yes|tools/bin/sdkmanager
"platform-tools" "platforms;android-27"' // andorid 8.1
                                sh 'yes|tools/bin/sdkmanager
"platform-tools" "platforms;android-28"'        // andorid 9.0
                                sh '(while sleep 3; do echo
"y"; done) | tools/android update sdk -u'

                                sh 'tools/bin/sdkmanager --
list'
                        }
                }
                echo '安装 Android SDK License'
                writeFile(file: 'platforms/licenses/android-
sdk-license', text: '''
8933bad161af4178b1185d1a37fbf41ea5269c55
24333f8a63b6825ea9c5514f83c2829b004d1fee
d56f5187479451eabf01fb78af6dfcb131a6481e
                                        ''')
                                sh 'ls -1 platforms'
            }
        }

        stage("Build") {
            steps {
                echo "构建中..."
                sh './gradlew'
                echo "构建完成."
            }
```

```
        }
        stage("Test") {
            steps {
                echo "单元测试中..."
                sh './gradlew test'
                echo "单元测试完成."
                //junit 'app/build/test-results/*/*.xml'
            }
        }
        stage("Package") {
            steps {
                sh './gradlew assemble'
                // 收集构建产物
                archiveArtifacts artifacts:
'app/build/outputs/apk/*/*.apk', fingerprint: true
            }
        }

        stage("Deploy") {
            steps {
                echo "部署中..."
                // sh './deploy.sh' // 自研部署脚本
                echo "部署完成"
            }
        }
    }
}
```

# 第 3 章 SonarQube

https://www.sonarqube.org

# 第 4 章 Phabricator - an open source, software engineering platform

***Phabricator is a collection of open source web applications that help software companies build better software.***

Phabricator 是 Facebook 开源的Code Review 工具

# 第 5 章 Gerrit

Gerrit 是 Google 开发的Code Review 工具

# 第 6 章 TeamCity

# 第 7 章 TRAC

http://trac.edgewall.org

## 1. Ubuntu 安装

### source code

过程 7.1. TRAC - source

1. setup.py

```
wget http://peak.telecommunity.com/dist/ez_setup.py
python ez_setup.py
```

2. Trac

```
wget http://download.edgewall.org/trac/Trac-1.1.1.tar.gz
tar zxvf Trac-1.1.1.tar.gz
cd Trac-1.1.1
sudo python ./setup.py install
cd ..
```

### easy_install

过程 7.2. TRAC - easy_install

1. easy_install

```
$ sudo apt-get install python-setuptools
```

2. Installing Trac

```
sudo easy_install Pygments
sudo easy_install Genshi
sudo easy_install Trac
```

ClearSilver

```
sudo apt-get install python-clearsilver
```

python svn

```
                                                    sudo apt-
get install python-svn python-svn-dbg
```

create svn repos

```
$ svnadmin create /home/netkiller/repos
```

## Apache httpd

```
# cat /etc/httpd/conf.d/trac.conf
<VirtualHost *:80>
  # Change this to the domain which points to your host, i.e.
the domain
  # you set as "phabricator.base-uri".
  ServerName trac.repo

  <Location />
    SetHandler mod_python
    PythonInterpreter main_interpreter
    PythonHandler trac.web.modpython_frontend
    PythonOption TracEnv /gitroot/trac/default
```

```
    PythonOption TracUriRoot /
  </Location>
# Replace all occurrences of /srv/trac with your trac root
below
# and uncomment the respective SetEnv and PythonOption
directives.
#  <LocationMatch /cgi-bin/trac\.f?cgi>
#       SetEnv TRAC_ENV /srv/trac
#  </LocationMatch>
#  <IfModule mod_python.c>
#    <Location /cgi-bin/trac.cgi>
#      SetHandler mod_python
#      PythonHandler trac.web.modpython_frontend
#      #PythonOption TracEnv /srv/trac
#    </Location>
#  </IfModule>
</VirtualHost>
```

## 2. CentOS 安装

http://trac.edgewall.org/

```
[root@development ~]# yum install python-setuptools
[root@development ~]# easy_install Trac

[root@development ~]# trac-admin /var/www/myproject initenv
```

### trac.ini

subversion 仓库配置

```
                                vim /srv/conf/trac.ini

                                repository_dir =
/svnroot/example.com
```

### standalone

```
tracd -s --port 8000 /var/www/myproject
```

multiple projects

```
tracd --port 8000 /var/www/trac/project1/
/var/www/trac/project2 ...
or
tracd --port 8000 -e /var/www/trac/
```

## Using Authentication

Using Authentication

To create a .passwd file using htdigest:

```
htdigest -c /var/www/trac/.passwd localhost neo
```

then for additional users:

```
htdigest /var/www/trac/.passwd localhost netkiller
```

bind ip

```
tracd -d --host 192.168.3.9 --port 8000 --
auth=*,/srv/trac/.passwd,localhost -e /srv/trac
```

```
$ tracd -p 8080 \
   --auth=project1,/path/to/users.htdigest,mycompany.com \
   --auth=project2,/path/to/users.htdigest,mycompany.com \
   /path/to/project1 /path/to/project2

tracd -p 8000 \
   --auth=*,/var/www/trac/.passwd,localhost \
   -e /var/www/trac/
```

## trac-admin

```
# trac-admin /srv/example help
trac-admin - The Trac Administration Console 0.12.3
```

```
Usage: trac-admin </path/to/projenv> [command [subcommand]
[option ...]]

Invoking trac-admin without command starts interactive mode.

help                  Show documentation
initenv               Create and initialize a new environment
attachment add        Attach a file to a resource
attachment export     Export an attachment from a resource to a
file or stdout
attachment list       List attachments of a resource
attachment remove     Remove an attachment from a resource
changeset added       Notify trac about changesets added to a
repository
changeset modified    Notify trac about changesets modified in a
repository
component add         Add a new component
component chown       Change component ownership
component list        Show available components
component remove      Remove/uninstall a component
component rename      Rename a component
config get            Get the value of the given option in
"trac.ini"
config remove         Remove the specified option from
"trac.ini"
config set            Set the value for the given option in
"trac.ini"
deploy                Extract static resources from Trac and all
plugins
hotcopy               Make a hot backup copy of an environment
milestone add         Add milestone
milestone completed   Set milestone complete date
milestone due         Set milestone due date
milestone list        Show milestones
milestone remove      Remove milestone
milestone rename      Rename milestone
permission add        Add a new permission rule
permission list       List permission rules
permission remove     Remove a permission rule
priority add          Add a priority value option
priority change       Change a priority value
priority list         Show possible ticket priorities
priority order        Move a priority value up or down in the
list
priority remove       Remove a priority value
```

```
repository add       Add a source repository
repository alias     Create an alias for a repository
repository list      List source repositories
repository remove    Remove a source repository
repository resync    Re-synchronize trac with repositories
repository set       Set an attribute of a repository
repository sync      Resume synchronization of repositories
resolution add       Add a resolution value option
resolution change    Change a resolution value
resolution list      Show possible ticket resolutions
resolution order     Move a resolution value up or down in the
list
resolution remove    Remove a resolution value
session add          Create a session for the given sid
session delete       Delete the session of the specified sid
session list         List the name and email for the given sids
session purge        Purge all anonymous sessions older than
the given age
session set          Set the name or email attribute of the
given sid
severity add         Add a severity value option
severity change      Change a severity value
severity list        Show possible ticket severities
severity order       Move a severity value up or down in the
list
severity remove      Remove a severity value
ticket remove        Remove ticket
ticket_type add      Add a ticket type
ticket_type change   Change a ticket type
ticket_type list     Show possible ticket types
ticket_type order    Move a ticket type up or down in the list
ticket_type remove   Remove a ticket type
upgrade              Upgrade database to current version
version add          Add version
version list         Show versions
version remove       Remove version
version rename       Rename version
version time         Set version date
wiki dump            Export wiki pages to files named by title
wiki export          Export wiki page to file or stdout
wiki import          Import wiki page from file or stdin
wiki list            List wiki pages
wiki load            Import wiki pages from files
wiki remove          Remove wiki page
wiki rename          Rename wiki page
```

```
wiki replace          Replace the content of wiki pages from
files (DANGEROUS!)
wiki upgrade          Upgrade default wiki pages to current
version
```

**Permissions**

```
BROWSER_VIEW

CHANGESET_VIEW

CONFIG_VIEW

EMAIL_VIEW

FILE_VIEW


LOG_VIEW

MILESTONE_ADMIN

MILESTONE_CREATE

MILESTONE_DELETE


MILESTONE_MODIFY

MILESTONE_VIEW

PERMISSION_ADMIN

PERMISSION_GRANT


PERMISSION_REVOKE

REPORT_ADMIN

REPORT_CREATE
```

```
REPORT_DELETE

REPORT_MODIFY
REPORT_SQL_VIEW
REPORT_VIEW
ROADMAP_ADMIN
ROADMAP_VIEW

SEARCH_VIEW
TICKET_ADMIN
TICKET_APPEND
TICKET_CHGPROP
TICKET_CREATE

TICKET_EDIT_CC
TICKET_EDIT_COMMENT
TICKET_EDIT_DESCRIPTION

TICKET_MODIFY
TICKET_VIEW
TIMELINE_VIEW
TRAC_ADMIN

VERSIONCONTROL_ADMIN
WIKI_ADMIN
```

```
WIKI_CREATE

WIKI_DELETE

WIKI_MODIFY


WIKI_RENAME

WIKI_VIEW
```

admin

```
$ trac-admin /path/to/projenv permission add neo TICKET_ADMIN
TRAC_ADMIN  WIKI_ADMIN
```

group

```
$ trac-admin /path/to/projenv permission add admin
MILESTONE_ADMIN PERMISSION_ADMIN REPORT_ADMIN ROADMAP_ADMIN
TICKET_ADMIN TRAC_ADMIN VERSIONCONTROL_ADMIN WIKI_ADMIN

$ trac-admin /path/to/projenv permission add developer
WIKI_ADMIN
$ trac-admin /path/to/projenv permission add developer
REPORT_ADMIN
$ trac-admin /path/to/projenv permission add developer
TICKET_ADMIN
```

user

```
$ trac-admin /path/to/projenv permission add bob developer
$ trac-admin /path/to/projenv permission add john developer
```

**Resync**

```
# trac-admin /srv/example repository resync '(default)'
```

旧版本trac: trac-admin /srv/trac/neo resync

# 3. Project Environment

## Sqlite

1. Creating a Project Environment

```
$ trac-admin /home/netkiller/projectenv initenv

Creating a new Trac environment at
/home/netkiller/projectenv

Trac will first ask a few questions about your environment
in order to initalize and prepare the project database.

 Please enter the name of your project.
 This name will be used in page titles and descriptions.

Project Name [My Project]>

 Please specify the connection string for the database to
use.
 By default, a local SQLite database is created in the
environment
 directory. It is also possible to use an already existing
 PostgreSQL database (check the Trac documentation for the
exact
 connection string syntax).

Database connection string [sqlite:db/trac.db]>

 Please specify the type of version control system,
 By default, it will be svn.

 If you don't want to use Trac with version control
integration,
 choose the default here and don't specify a repository
directory.
 in the next question.

Repository type [svn]>
```

```
 Please specify the absolute path to the version control
 repository, or leave it blank to use Trac without a
repository.
 You can also set the repository location later.

Path to repository [/path/to/repos]> /home/netkiller/repos

 Please enter location of Trac page templates.
 Default is the location of the site-wide templates
installed with Trac.

Templates directory [/usr/share/trac/templates]>

Creating and Initializing Project
 Installing default wiki pages
 /usr/share/trac/wiki-default/TracIni => TracIni
 /usr/share/trac/wiki-default/TracSupport => TracSupport
 /usr/share/trac/wiki-default/WikiStart => WikiStart
 /usr/share/trac/wiki-default/TitleIndex => TitleIndex
 /usr/share/trac/wiki-default/TracModPython =>
TracModPython
 /usr/share/trac/wiki-default/TracInterfaceCustomization =>
TracInterfaceCustomization
 /usr/share/trac/wiki-default/WikiDeletePage =>
WikiDeletePage
 /usr/share/trac/wiki-default/TracTicketsCustomFields =>
TracTicketsCustomFields
 /usr/share/trac/wiki-default/TracChangeset =>
TracChangeset
 /usr/share/trac/wiki-default/TracLogging => TracLogging
 /usr/share/trac/wiki-default/TracSyntaxColoring =>
TracSyntaxColoring
 /usr/share/trac/wiki-default/TracImport => TracImport
 /usr/share/trac/wiki-default/TracTimeline => TracTimeline
 /usr/share/trac/wiki-default/TracAdmin => TracAdmin
 /usr/share/trac/wiki-default/InterWiki => InterWiki
 /usr/share/trac/wiki-default/WikiPageNames =>
WikiPageNames
 /usr/share/trac/wiki-default/TracNotification =>
TracNotification
 /usr/share/trac/wiki-default/TracFastCgi => TracFastCgi
 /usr/share/trac/wiki-default/InterTrac => InterTrac
 /usr/share/trac/wiki-default/TracUnicode => TracUnicode
 /usr/share/trac/wiki-default/TracGuide => TracGuide
```

```
 /usr/share/trac/wiki-default/TracRevisionLog =>
TracRevisionLog
 /usr/share/trac/wiki-default/TracBrowser => TracBrowser
 /usr/share/trac/wiki-default/WikiRestructuredText =>
WikiRestructuredText
 /usr/share/trac/wiki-default/TracLinks => TracLinks
 /usr/share/trac/wiki-default/TracInstall => TracInstall
 /usr/share/trac/wiki-default/TracPermissions =>
TracPermissions
 /usr/share/trac/wiki-default/WikiMacros => WikiMacros
 /usr/share/trac/wiki-default/TracQuery => TracQuery
 /usr/share/trac/wiki-default/TracBackup => TracBackup
 /usr/share/trac/wiki-default/TracWiki => TracWiki
 /usr/share/trac/wiki-default/SandBox => SandBox
 /usr/share/trac/wiki-default/TracRoadmap => TracRoadmap
 /usr/share/trac/wiki-default/TracAccessibility =>
TracAccessibility
 /usr/share/trac/wiki-default/TracSearch => TracSearch
 /usr/share/trac/wiki-default/TracPlugins => TracPlugins
 /usr/share/trac/wiki-default/RecentChanges =>
RecentChanges
 /usr/share/trac/wiki-default/WikiNewPage => WikiNewPage
 /usr/share/trac/wiki-default/TracCgi => TracCgi
 /usr/share/trac/wiki-default/TracRss => TracRss
 /usr/share/trac/wiki-default/CamelCase => CamelCase
 /usr/share/trac/wiki-default/WikiFormatting =>
WikiFormatting
 /usr/share/trac/wiki-default/TracTickets => TracTickets
 /usr/share/trac/wiki-default/TracStandalone =>
TracStandalone
 /usr/share/trac/wiki-default/InterMapTxt => InterMapTxt
 /usr/share/trac/wiki-default/TracReports => TracReports
 /usr/share/trac/wiki-default/WikiHtml => WikiHtml
 /usr/share/trac/wiki-default/WikiProcessors =>
WikiProcessors
 /usr/share/trac/wiki-default/TracUpgrade => TracUpgrade
 /usr/share/trac/wiki-default/TracEnvironment =>
TracEnvironment
 /usr/share/trac/wiki-default/WikiRestructuredTextLinks =>
WikiRestructuredTextLinks

Warning:

You should install the SVN bindings
```

```
-----------------------------------------------------------------
----------
Project environment for 'My Project' created.

You may now configure the environment by editing the file:

  /home/netkiller/projectenv/conf/trac.ini

If you'd like to take this new project environment for a
test drive,
try running the Trac standalone web server `tracd`:

  tracd --port 8000 /home/netkiller/projectenv

Then point your browser to
http://localhost:8000/projectenv.
There you can also browse the documentation for your
installed
version of Trac, including information on further setup
(such as
deploying Trac to a real web server).

The latest documentation can also always be found on the
project
website:

  http://trac.edgewall.org/

Congratulations!
```

2. Running the Standalone Server

```
tracd --port 8000 /home/netkiller/projectenv
```

3. testing

   http://192.168.1.7:8000/projectenv/

4. auth

```
sudo apt-get install apache2-utils

$ htdigest -c /home/neo/trac/conf/passwd.digest localhost
neo
Adding password for neo in realm localhost.
New password:
Re-type new password:

$ htdigest /home/neo/trac/conf/passwd.digest localhost
nchen
Adding user nchen in realm localhost
New password:
Re-type new password:

$ trac-admin /home/neo/trac permission add admin TRAC_ADMIN
$ trac-admin /home/neo/trac permission add netkiller admin

$ trac-admin /home/neo/trac permission add developer
TICKET_ADMIN
$ trac-admin /home/neo/trac permission add nchen developer
$ trac-admin /home/neo/trac permission add neo developer

$ trac-admin /home/neo/trac permission list
User           Action
------------------------------
admin          TRAC_ADMIN
anonymous      BROWSER_VIEW
anonymous      CHANGESET_VIEW
anonymous      FILE_VIEW
anonymous      LOG_VIEW
anonymous      MILESTONE_VIEW
anonymous      REPORT_SQL_VIEW
anonymous      REPORT_VIEW
anonymous      ROADMAP_VIEW
anonymous      SEARCH_VIEW
anonymous      TICKET_VIEW
anonymous      TIMELINE_VIEW
anonymous      WIKI_VIEW
authenticated  TICKET_CREATE
authenticated  TICKET_MODIFY
authenticated  WIKI_CREATE
authenticated  WIKI_MODIFY
developer      TICKET_ADMIN
nchen          developer
```

```
neo             developer
netkiller       admin
```

5. daemon

```
$ tracd -d -s --port 8000 /home/netkiller/projectenv
$ tracd -d -s --port 8000 --auth
trac,/home/neo/trac/conf/passwd.digest,localhost
/home/neo/trac
```

## MySQL

```
GRANT ALL PRIVILEGES ON trac.* TO trac@'localhost' IDENTIFIED
BY 'password' WITH GRANT OPTION;
CREATE DATABASE IF NOT EXISTS trac default charset utf8 COLLATE
utf8_general_ci;
```

```
Database connection string [sqlite:db/trac.db]>
mysql://trac:password@localhost:3306/trac
```

下面开始创建项目

```
# trac-admin /home/git/trac initenv
Creating a new Trac environment at /home/git/trac

Trac will first ask a few questions about your environment
in order to initialize and prepare the project database.

 Please enter the name of your project.
 This name will be used in page titles and descriptions.

Project Name [My Project]>

 Please specify the connection string for the database to use.
```

```
 By default, a local SQLite database is created in the
environment
 directory. It is also possible to use an already existing
 PostgreSQL database (check the Trac documentation for the
exact
 connection string syntax).

Database connection string [sqlite:db/trac.db]>
mysql://trac:trac@localhost:3306/trac

Creating and Initializing Project
 Installing default wiki pages
  TracRepositoryAdmin imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-pages/TracRepositoryAdmin
  TracNavigation imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracNavigation
  TracUpgrade imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracUpgrade
  TracRevisionLog imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracRevisionLog
  TracTickets imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracTickets
  TracIni imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracIni
  PageTemplates imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/PageTemplates
  TracTimeline imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracTimeline
  TracAccessibility imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-pages/TracAccessibility
  WikiHtml imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiHtml
  SandBox imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/SandBox
  TracImport imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracImport
  TracPlugins imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracPlugins
  TracRoadmap imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracRoadmap
  TracAdmin imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracAdmin
  TracBatchModify imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracBatchModify
  TracBrowser imported from /root/.python-eggs/Trac-1.1.1-
```

```
py2.6.egg-tmp/trac/wiki/default-pages/TracBrowser
  InterWiki imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/InterWiki
  WikiRestructuredText imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/WikiRestructuredText
  WikiProcessors imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiProcessors
  WikiNewPage imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiNewPage
  TracEnvironment imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracEnvironment
  TracLogging imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracLogging
  TracSupport imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracSupport
  TracNotification imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracNotification
  TracGuide imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracGuide
  WikiStart imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiStart
  TracWorkflow imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracWorkflow
  TracRss imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracRss
  TracLinks imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracLinks
  InterMapTxt imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/InterMapTxt
  WikiPageNames imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiPageNames
  WikiFormatting imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiFormatting
  WikiRestructuredTextLinks imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/WikiRestructuredTextLinks
  TracUnicode imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracUnicode
  TracChangeset imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracChangeset
  TitleIndex imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TitleIndex
  WikiDeletePage imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiDeletePage
```

```
  TracReports imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracReports
  TracWiki imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracWiki
  RecentChanges imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/RecentChanges
  TracBackup imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracBackup
  TracModPython imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracModPython
  TracSearch imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracSearch
  TracModWSGI imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracModWSGI
  TracTicketsCustomFields imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/TracTicketsCustomFields
  TracQuery imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracQuery
  TracStandalone imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracStandalone
  InterTrac imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/InterTrac
  TracFineGrainedPermissions imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/TracFineGrainedPermissions
  TracInterfaceCustomization imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/TracInterfaceCustomization
  TracCgi imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracCgi
  TracFastCgi imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracFastCgi
  TracPermissions imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracPermissions
  TracInstall imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracInstall
  TracSyntaxColoring imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-pages/TracSyntaxColoring
  CamelCase imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/CamelCase
  WikiMacros imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiMacros

-----------------------------------------------------------------
```

```
------
Project environment for 'My Project' created.

You may now configure the environment by editing the file:

  /home/git/trac/conf/trac.ini

If you'd like to take this new project environment for a test
drive,
try running the Trac standalone web server `tracd`:

  tracd --port 8000 /home/git/trac

Then point your browser to http://localhost:8000/trac.
There you can also browse the documentation for your installed
version of Trac, including information on further setup (such
as
deploying Trac to a real web server).

The latest documentation can also always be found on the
project
website:

  http://trac.edgewall.org/

Congratulations!
```

## Plugin

### AccountManagerPlugin

http://trac-hacks.org/wiki/AccountManagerPlugin

```
cd accountmanagerplugin/
python setup.py install
python setup.py bdist_egg

cp dist/TracAccountManager-0.4.4-py2.6.egg
/home/git/trac/plugins/
```

**Subtickets**

http://trac-hacks.org/wiki/SubticketsPlugin

## 4. trac.ini

### repository

```
[trac]
repository_dir = /opt/svnroot/neo
repository_sync_per_request = (default)
repository_type = svn
```

svn 仓库地址 repository_dir

### attachment 附件配置

上传附件尺寸控制

```
[attachment]
max_size=262144
```

## 5. trac-admin

权限组定制

```
trac-admin /opt/trac permission add admin TRAC_ADMIN
trac-admin /opt/trac permission add Development TICKET_ADMIN
trac-admin /opt/trac permission add Operations TICKET_ADMIN
```

权限初始化

```
trac-admin /opt/trac permission add mgmt admin
trac-admin /opt/trac permission add neo Development
trac-admin /opt/trac permission add ken Operations

trac-admin /opt/trac permission list
```

### adduser script

```
#!/bin/bash

user=$1
trac-admin /opt/trac permission add $user Operations
htdigest -c /opt/trac/conf/passwd.digest localhost $user
```

# 6. Trac 项目管理

Trac 初始化步骤

1. 首先进入Admin，初始化TRAC
2. 使用Wiki创建项目页
3. 创建Milestones
4. 创建Ticket

## Administration

### General

安装后首先分配权限

过程 7.3. Permissions 设置

1. 我习惯于 创建一个 developer 组和 adminstrator 组

   然后创建用户隶属于 developer 组

2. 创建用户隶属于developer组

### Ticket System

过程 7.4. Ticket System 设置

1. 设置 Components

   例如电商项目，这里可以设置，注册登录，用户中心，购物车，物流配送等等

2. 设置 Milestones

   Roadmap->Milestone->Add new milestone

我一般是一个月一个里程碑

3. 设置 Priorities

   我一般设置为:

   新特性（优先）, 不限期, 立即执行, 当日完成, 本周完成, 本月完成

4. Resolutions

   任务完成, 无效BUG, 重复, 待测试, 待发布

5. Severities

   严重错误, 次要错误, 文字错误, 不合理

6. Ticket Types

   Ticket Types 初始化
   1. 开发
   2. 测试
   3. 运维
   4. 设计
   5. 需求
   6. 事件

   7. BUG

7. Versions

   不多说了 1.0， 1。1 或者 1.0.1

**Version Control**

Repositories

默认支持 Subversion, 创建一个仓库记得不要忘记创建下面三个目录 1.branches, 2.tags, 3.trunk

```
trunk                    主干
branches                 在下面再创建两个目录development,testing
tags                     当项目Release 后会在此处打一个标记
```

Git 不需要这三个目录，我习惯上会创建几个分支

```
master                   主干
development              开发分支
testing                  测试分支
```

关于版本库项目目录，我习惯与使用该项目对应的域名作为项目目录

```
/example.com
/example.com/www.exampe.com
/example.com/images.exampe.com
/example.com/user.exampe.com
/example.com/admin.exampe.com
```

**Wiki**

过程 7.5. Wiki 使用方法

1. 项目成员页，里面要包含所有项目程序的联系方式

   ```
   |name|telephone|cellphone|ext|im|email|
   |Neo|13122993040||||
   ```

2. 需求页面

**Timeline**

可以看到每时每刻的项目变化，包括Wiki, Ticket, 以及代码提交

**Roadmap**

Roadmap 中的里程碑页，也可以加以利用，我喜欢将一个里程碑分解为多个Ticket 然后在该页面体现，包括整体上的工作安排等等，使用表格来安排Ticket日程，一定程度上弥补了TRAC没有甘特图的不足，

**Ticket**

过程 7.6. Ticket 使用方法

1. New Ticket

   新建Ticket, Ticket 可以理解为任务。

2. 将Ticket 分配给团队成员

   受到Ticket后，一定要更改Ticket 为 accept ， 这时在View Tickets 中将会看到该Ticket已经分配，

3. 编码过程

   这里有一个特别的规定，提交代码（包括Subversion与Git）注释中必须这样写：

```
svn ci -m "Ticket #123 - xxxxxxxxxxxxxxxxxxx"
git commit -a -m "Ticket #123 - xxxxxxxxxxxxxxxxxxx"
```

格式：Ticket #123 - 你的注释

这样写的好处是，在Timeline 中可以直接点击 Ticket 编号直接进入Ticket

```
10:54 AM Ticket #462 (添加一个支付方式) reopened by neo

4:51 PM Changeset in admin.example.com [01a0c4] by neo
<neo.chan@example.com>
Ticket #452 - 用户登录日志
```

4. Add Comment

回复Ticket，上面提交后悔产生一个Subversion版本号，按照下面格式写，然后提交

```
Changesets: r1, [1] or changeset:1
```

这样就可以实现，进入Ticket即可看到做了哪些代码提交与改动，一目了然。

Git 写法

```
[changeset:af212a]
[changeset:7a03c65500c4b96859a27bf5be2901e4ec42afdd]
```

如果 Repositories 中有多个项目写法如下

```
[changeset:af212a/www.example.com]
```

# 7. FAQ

## TracError: Cannot load Python bindings for MySQL

检查 MySQLdb 是否安装

```
# /usr/bin/python -c 'import MySQLdb'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: No module named MySQLdb
```

安装MySQLdb

```
# yum install python-devel
# pip install MySQL-python
```

或者

```
# yum install python-devel
# easy_install MySQL-python
```

再次测试，如果不出现任何提示表示成功。

```
# /usr/bin/python -c 'import MySQLdb'
```

## 8. Apache Bloodhound

Apache Bloodhound 是基于 Trac 的项目管理软件

# 第 8 章 Redmine

http://www.redmine.org/

redmine 一键安装包

## 1. CentOS 安装

安装MySQL数据库

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/my
sql/mysql.server.sh | bash
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/my
sql/mysql.devel.sh | bash
```

创建数据库账号

```
CREATE DATABASE redmine CHARACTER SET utf8;
GRANT ALL PRIVILEGES ON redmine.* TO 'redmine'@'localhost'
IDENTIFIED BY 'my_password';
```

安装

```
yum install -y ruby rubygems ruby-devel ImageMagick-devel

cd /usr/local/src/
wget http://www.redmine.org/releases/redmine-3.3.0.tar.gz
tar zxf redmine-3.3.0.tar.gz
mv redmine-3.3.0 /srv/
ln -s /srv/redmine-3.3.0 /srv/redmine
cd /srv/redmine
```

```
cat >> config/database.yml <<EOF
production:
  adapter: mysql2
  database: redmine
  host: localhost
  username: redmine
  password: my_password
  encoding: utf8
EOF

gem install bundler
bundle install --without development test
bundle exec rake generate_secret_token

RAILS_ENV=production bundle exec rake db:migrate
#bundle exec rake redmine:load_default_data
RAILS_ENV=production REDMINE_LANG=zh bundle exec rake
redmine:load_default_data

mkdir -p tmp tmp/pdf public/plugin_assets
sudo chown -R redmine:redmine files log tmp
public/plugin_assets
sudo chmod -R 755 files log tmp public/plugin_assets

bundle exec rails server webrick -e production
```

默认用户名与密码 login: admin，password: admin

## 2. Redmine 运行

```
# rails server -h
Usage: rails server [mongrel, thin etc] [options]
    -p, --port=port                  Runs Rails on the
specified port.
                                     Default: 3000
    -b, --binding=IP                 Binds Rails to the
specified IP.
                                     Default: localhost
    -c, --config=file                Uses a custom rackup
configuration.
    -d, --daemon                     Runs server as a Daemon.
    -u, --debugger                   Enables the debugger.
    -e, --environment=name           Specifies the environment
to run this server under (test/development/production).
                                     Default: development
    -P, --pid=pid                    Specifies the PID file.
                                     Default:
tmp/pids/server.pid

    -h, --help                       Shows this help message.
```

绑定监听地址 -b

```
# bundle exec rails server webrick -e production -b 0.0.0.0
```

守护进程 -d

## 3. 插件

### workflow

http://www.redmine.org/plugins/redmine_workflow_enhancements

# 第 9 章 TUTOS

TUTOS is a tool to manage the organizational needs of small groups, teams, departments ...

http://www.tutos.org/

过程 9.1. TUTOS

1. extract

```
tar jxvf TUTOS-php-1.3.20070317.tar.bz2
sudo mv tutos /www/htdocs/
```

2. database

```
netkiller@shenzhen:/www/htdocs/tutos$ mysqladmin -uroot -p
create tutos
netkiller@shenzhen:/www/htdocs/tutos$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 846
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the
buffer.

mysql> grant all on tutos.* to tutos@% identified by
"chen";
Query OK, 0 rows affected (0.05 sec)

mysql> grant all on tutos.* to tutos@localhost identified
by "chen";
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> quit
Bye

netkiller@shenzhen:/www/htdocs/tutos$ mysqladmin -uroot -p
reload
```

3. config

```
mkdir /www/htdocs/tutos/repository
```

http://192.168.1.7/tutos/php/admin/scheme.php

or

```
cp config_default.pinc  config.php
```

```
<?php
# remove this line when finsihed with config
$tutos['CCSID'] = "10880f50567242006bf2c1a2c0b8b350";
#
# sessionpath
#
$tutos[sessionpath] = "/tmp";
#
# the next lines are a database definition
#
$tutos[dbname][0]     = "tutos";
$tutos[dbhost][0]     = "localhost";
$tutos[dbport][0]     = "5432";
$tutos[dbuser][0]     = "tutos";
$tutos[dbpasswd][0]   = "chen";
$tutos[dbtype][0]     = "2";
```

```
$tutos[dbalias][0]    = "Mysql database";
$tutos[cryptpw][0]    = "";
$tutos[repository][0] = "repository";
$tutos[dbprefix][0]   = "";
#
# MAIL
#
$tutos[mailmode] = "2";
$tutos[sendmail] = "/usr/lib/sendmail";
$tutos[smtphost] = "localhost";
#
# demo mode
#
$tutos[demo] = 0;
#
# debug mode
#
$tutos[debug] = 0;
$tutos[errlog] = "/tmp/debug.out";
#
$tutos[jpgraph] = "/www/htdocs/tutos/php/admin/jpgraph";
#
# EOF
?>
```

sudo apt-get install perl libnet-ssleay-perl openssl libauthen-pam-perl libpam-runtime libio-pty-perl libmd5-perl

4. login

http://192.168.1.7/tutos/php/mytutos.php

User: superuser Password: tutos

# 第 10 章 Open Source Requirements Management Tool

http://sourceforge.net/projects/osrmt/

```
<client directory>\v1_50\client\

copy connection.mysql.xml connection.xml
```

```
<client directory>\v1_50\client\upgrade.bat

Select configuration option 1,2,3 or 4
1) Define a new connection
2) Test the connection
3) Save the new connection
4) Initialize a new database
5) Upgrade 1.3 to 1.4 database
6) Migrate database contents
7) Export language file
8) Import language file
0) Exit
Enter option number [Exit]:


Enter option number [Exit]: 4
initializing database defined in connection.xml:
jdbc:mysql://localhost/osrmt?
useUnicode=true&characterEncoding=UTF-8
osrmt
mysql
Target correct? Y|N [Y]: y
Empty schema located - initialize and populate schema? [Y]:
```

# 部分 II. 软件版本控制

# 第 11 章 Git - Fast Version Control System

### *distributed revision control system*

homepage: http://git.or.cz/index.html

过程 11.1. Git

1. install

```
sudo apt-get install git-core
```

2. config

```
$ git-config --global user.name neo
$ git-config --global user.email openunix@163.com
```

3. Initializ

```
$ mkdir repository
$ cd repository/

/repository$ git-init-db
Initialized empty Git repository in .git/
```

to check .gitconfig file

```
$ cat ~/.gitconfig
[user]
        name = chen
```

```
        email = openunix@163.com
```

# 1. Repositories 仓库管理

## 1.1. initial setup

```
Tell git who you are:

$ git config user.name "FirstName LastName"
$ git config user.email "user@example.com"

If you have many git repositories under your current user, you
can set this for all of them


$ git config --global user.name "FirstName LastName"
$ git config --global user.email "user@example.com"

If you want pretty colors, you can setup the following for
branch, status, and diff commands:

$ git config --global color.branch "auto"
$ git config --global color.status "auto"
$ git config --global color.diff "auto"

Or, to turn all color options on (with git 1.5.5+), use:

$ git config --global color.ui "auto"
To enable aut-detection for number of threads to use (good for
multi-CPU or multi-core computers) for packing repositories,
use:


$ git config --global pack.threads "0"
To disable the rename detection limit (which is set "pretty
low" according to Linus, "just to not cause problems for people
who have less memory in their machines than kernel developers
tend to have"), use:

$ git config --global   diff.renamelimit "0"
```

## 1.2. checkout

将 nqp-cc/src/QASTCompilerMAST.nqp 文件 重置到 211ab0b19f25b8c81685a97540f4b1491eb17504 版本

```
git checkout 211ab0b19f25b8c81685a97540f4b1491eb17504 -- nqp-
cc/src/QASTCompilerMAST.nqp
```

## 1.3. Creating and Commiting

```
$ cd (project-directory)
$ git init
$ (add some files)
$ git add .
$ git commit -m 'Initial commit'
```

## 1.4. Manager remote

remote add

```
git remote add origin git@localhost:example.git
```

remote show

```
git remote show
origin
```

remote rm

```
git remote rm origin
```

添加多个远程仓库

```
git remote add origin git@localhost:example.git
git remote add another
https://gitcafe.com/netkiller/netkiller.gitcafe.com.git
git push origin master
git push another master
```

## 1.5. Status

```
$ git clone git://10.10.0.5/example.git
Cloning into example...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 0 (delta 0)
Receiving objects: 100% (5/5), done.
Resolving deltas: 100% (1/1), done.

neo@neo-OptiPlex-380:~/tmp$ cd example/

neo@neo-OptiPlex-380:~/tmp/example$ git status
# On branch master
nothing to commit (working directory clean)

neo@neo-OptiPlex-380:~/tmp/example$ ls
test1  test2  test3  test4

neo@neo-OptiPlex-380:~/tmp/example$ echo hello > test1

neo@neo-OptiPlex-380:~/tmp/example$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in
```

```
working directory)
#
#       modified:   test1
#
no changes added to commit (use "git add" and/or "git commit -
a")
```

## 1.6. Diff

```
neo@neo-OptiPlex-380:~/tmp/example$ git diff
diff --git a/test1 b/test1
index e69de29..ce01362 100644
--- a/test1
+++ b/test1
@@ -0,0 +1 @@
+hello
```

比较 nqp-cc/src/QASTCompilerMAST.nqp 文件 当前版本与 211ab0b19f25b8c81685a97540f4b1491eb17504 版本的区别

```
git diff 211ab0b19f25b8c81685a97540f4b1491eb17504 -- nqp-
cc/src/QASTCompilerMAST.nqp
```

**--name-only** 仅显示文件名

```
git diff --name-only
```

## 1.7. Cloning

```
```

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
```

## 1.8. Push

```
$ git clone git://10.10.0.5/example.git
$ cd example
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'

$ git push origin master
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 278 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git://10.10.0.5/example.git
   27f8417..b088cc3  master -> master
```

## 1.9. Pull

```
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git://10.10.0.5/example
   27f8417..b088cc3  master     -> origin/master
Updating 27f8417..b088cc3
Fast-forward
 test1 |    1 +
```

```
 1 files changed, 1 insertions(+), 0 deletions(-)
```

## 1.10. fetch

```
$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
From git://10.10.0.5/example
   b088cc3..7e8c17d  master     -> origin/master
```

## 1.11. Creating a Patch

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
$ git format-patch origin/master
```

## 1.12. reset

重置到上一个版本

```
git log
git reset --hard HEAD^
git log
git push -f
```

# 2. 分支管理

## Manipulating branches

git-branch - List, create, or delete branches

## 2.1. 查看本地分支

```
$ git branch
* master
```

查看远程分支

```
git branch -a
```

## 2.2. 创建分支

```
$ git branch development
$ git branch
* master
  development
```

## 2.3. 删除分支

```
$ git branch -d staging
Deleted branch staging.

$ git branch
* master
```

## 2.4. 切换分支

```
$ git branch
* master
  testing

$ git checkout testing
Switched to branch "testing"

$ git branch
  master
* testing
```

## 2.5. 重命名分支

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout
test
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git branch -
m test testing
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push --
delete origin test
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push
origin testing
```

## 2.6. git-show-branch - Show branches and their commits

```
$ git-show-branch
! [master] add a new file
 * [mybranch] add a new file
--
+* [master] add a new file
```

# 3. Sharing Repositories with others

## 3.1. Setting up a git server

First we need to setup a user with a home folder. We will store all the repositories in this users home folder.

```
sudo adduser git
```

Rather than giving out the password to the git user account use ssh keys to login so that you can have multiple developers connect securely and easily.

Next we will make a repository. For this example we will work with a repository called example. Login as the user git and add the repository.

login to remote server

```
ssh git@REMOTE_SERVER
```

once logged in

```
sudo mkdir example.git
cd example.git
sudo git --bare init
Initialized empty Git repository in /home/git/example.git/
```

That's all there is to creating a repository. Notice we named our folder with a .git extension.

Also notice the 'bare' option. By default the git repository assumes that you'll be using it as your working directory, so git stores the actual bare repository files in a .git directory alongside all the project files. Since we are setting up a remote server we don't need copies of the files on the filesystem. Instead, all we need are the deltas and binary objects of the repository. By setting 'bare' we tell git not to store the current files of the repository only the diffs. This is optional as you may have need to be able to browse the files on your remote server.

Finally all you need to do is add your files to the remote repository. We will assume you don't have any files yet.

```
mkdir example
cd example
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@REMOTE_SERVER:example.git
git push origin master
```

replace REMOTE_SERVER with your server name or IP

## 3.2. 修改 origin

```
git remote rename origin old-origin
```

## 3.3. 删除 origin

```
git remote remove origin
```

# 4. 合并分支

## 4.1. 合并分支

testing 分支向 master 分支合并

获取 testing 合并请求的分支

```
git fetch origin
git checkout -b "testing" "origin/testing"
```

如果此前已经执行过，使用下面命令切换分支即可，切换后 pull 代码，看看有什么新提交

```
git checkout "testing"
git pull
```

切换到 master 分支

```
git fetch origin
git checkout "master"
git branch --show-current
git merge --no-ff "testing"
```

将合并结果推送到远程

```
git push origin "master"
```

## 4.2. rebase

恢复 rebase 版本

```
git rebase
git reflog
git reset --hard 5faf0ae
git push
```

## 4.3. 合并分支解决冲突

案例，例如我们从 testing 分支向 master 分支合并代码出现冲突，该如何解决呢?

首先，两个分支拉取最新代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout testing
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git pull
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout master
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git pull
```

然后合并分支，从 testing 分支向 master 合并

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git merge --no-ff testing
自动合并 neo-incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
冲突（内容）：合并冲突于 neo-incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
自动合并失败，修正冲突然后提交修正的结果。
```

出现冲突，编辑冲突文件

```
vim neo-incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

保存后重看状态

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。

您有尚未合并的路径。
  （解决冲突并运行 "git commit"）
  （使用 "git merge --abort" 终止合并）

要提交的变更：
        修改：      neo-admin/src/main/resources/application-prod.yml
        修改：      neo-admin/src/main/resources/application-test.yml
        修改：      neo-common/src/main/java/com/neo/common/enums/IncarAttachTypeEnum.java
        修改：      neo-
incar/src/main/java/com/neo/incar/service/impl/IncarAttachServiceImpl.java

未合并的路径：
  （使用 "git add <文件>..." 标记解决方案）
        双方修改：    neo-incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

将合并的文件添加到 git

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git add neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。
```

```
所有冲突已解决但您仍处于合并中。
  (使用 "git commit" 结束合并)

要提交的变更:
        修改:      neo-admin/src/main/resources/application-prod.yml
        修改:      neo-admin/src/main/resources/application-test.yml
        修改:      neo-common/src/main/java/com/neo/common/enums/IncarAttachTypeEnum.java
        修改:      neo-
incar/src/main/java/com/neo/incar/service/impl/IncarAttachServiceImpl.java
        修改:      neo-incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

提交代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git commit -a -m '手工合并分支 testing ->
master'
[master 3652bf8e] 手工合并分支 testing -> master
```

推送代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push
枚举对象中: 1, 完成.
对象计数中: 100% (1/1), 完成.
写入对象中: 100% (1/1), 240 字节 | 240.00 KiB/s, 完成.
总共 1 (差异 0), 复用 0 (差异 0), 包复用 0
remote:
remote: To create a merge request for master, visit:
remote:   http://192.168.30.5/netkiller.cn/api.netkiller.cn/-/merge_requests/new?
merge_request%5Bsource_branch%5D=master
remote:
To http://192.168.30.5/netkiller.cn/api.netkiller.cn.git
   fcaefaf4..3652bf8e  master -> master
```

# 5. Submodule 子模块

## 5.1. 添加模块

```
neo@MacBook-Pro ~ % cd workspace/Linux

neo@MacBook-Pro ~/workspace/Linux % git submodule add
https://github.com/netkiller/common.git common
Cloning into '/Users/neo/workspace/Linux/common'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 185 (delta 2), reused 6 (delta 1), pack-reused 176
Receiving objects: 100% (185/185), 56.49 KiB | 163.00 KiB/s,
done.
Resolving deltas: 100% (105/105), done.
```

模块信息存储在 .gitmodules 文件中

```
neo@MacBook-Pro ~/workspace/Linux % cat .gitmodules
[submodule "common"]
        path = common
        url = https://github.com/netkiller/common.git
```

同时也添加到 .git/config 文件中

```
neo@MacBook-Pro ~/workspace/Linux % cat .git/config | tail -n 3
[submodule "common"]
        url = https://github.com/netkiller/common.git
        active = true
```

## 5.2. checkout 子模块

clone 项目，然后进入目录

```
neo@MacBook-Pro /tmp/test % git clone
https://github.com/netkiller/Linux.git
neo@MacBook-Pro /tmp/test % cd Linux
```

初始化子模块

```
neo@MacBook-Pro /tmp/test/Linux % git submodule init
Submodule 'common' (https://github.com/netkiller/common.git)
registered for path 'common'
```

更新模块

```
neo@MacBook-Pro /tmp/test/Linux % git submodule update
Cloning into '/private/tmp/test/Linux/common'...
Submodule path 'common': checked out
'cdf61a1de34590bcc80b895fdf0e90b62cfd729f'
```

## 5.3. 删除子模块

```
git rm --cached <module>
```

# 6. Git Large File Storage

https://git-lfs.github.com/

Git Large File Storage | Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"

brew install git-lfs
```

## 6.1. 安装 LFS 支持

```
git lfs install
git lfs track "*.psd"
git add .gitattributes

git add file.psd
git commit -m "Add design file"
git push origin master
```

## 6.2. LFS lock

文件锁的用途是用户可以对一个文件进行加锁，阻止其他用户同一时间对该文件进行修改操作。因为在GIT仓库中同时编辑一个文件，会发生冲突，然而解决二进制大文件的冲突，合并操作极其困难。

```
neo@MacBook-Pro ~/workspace/java-project % git lfs lock test.psd
Locked test.psd

neo@MacBook-Pro ~/workspace/java-project % git lfs locks
test.psd        bg7nyt  ID:55777
```

如果Push被锁的文件，提示 Remote "origin" does not support the LFS locking API

```
neo@MacBook-Pro /tmp/java % git commit -a -m 'aaa'
[master b832eb3] aaa
 1 file changed, 2 insertions(+), 2 deletions(-)
neo@MacBook-Pro /tmp/java % git push
Remote "origin" does not support the LFS locking API. Consider disabling it with:
  $ git config 'lfs.https://github.com/bg7nyt/java.git/info/lfs.locksverify' false
Post https://github.com/bg7nyt/java.git/info/lfs/locks/verify: EOF
error: failed to push some refs to 'https://github.com/bg7nyt/java.git'
```

解锁后Push成功

```
neo@MacBook-Pro ~/workspace/java-project % git lfs unlock test.psd
Unlocked test.psd

neo@MacBook-Pro /tmp/java % git push
Git LFS: (1 of 1 files) 9 B / 9 B
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 352 bytes | 352.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/bg7nyt/java.git
   b29f474..b832eb3  master -> master
```

# 7. command

## 7.1. hash-object

使用git命令计算文件的 sha-1 值

```
neo@MacBook-Pro ~ % echo 'test content' | git hash-object --
stdin
d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

## 7.2. git-add - Add file contents to the index

```
$ echo 'hello world!!!'> newfile
$ git-add newfile
```

## 7.3. git-status - Show the working tree status

```
$ git-status newfile
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file: newfile
#
```

## 7.4. git-commit - Record changes to the repository

```
$ git-commit -m 'add a new file' newfile
Created initial commit f6fda79: add a new file
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 newfile
```

## 7.5. git-show - Show various types of objects

```
$ git-show
commit f6fda79f2f550ea3b2c1b483371ed5d12499ac35
Author: chen <openunix@163.com>
Date:   Sat Nov 1 08:50:45 2008 -0400

    add a new file

diff --git a/newfile b/newfile
new file mode 100644
index 0000000..b659464
--- /dev/null
+++ b/newfile
@@ -0,0 +1 @@
+hello world!!!
```

## 7.6. git-checkout - Checkout and switch to a branch

### checkout master

```
$ git-checkout master
Switched to branch "master"
```

### checkout branch

```
$ git-branch
* master
  mybranch

$ git-checkout mybranch
Switched to branch "mybranch"

$ git-branch
  master
* mybranch
```

## 7.7. git config

```
$ git config --file config http.receivepack true
```

## 7.8. git log

```


git log --graph --pretty=format:'%Cred%h%Creset -
%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset'
--abbrev-commit

git log --graph --pretty=format:'%Cred%h%Creset -
%C(yellow)%d%Creset %s %Cgreen(%ai) %C(bold blue)<%an>%Creset'
--abbrev-commit


```

# 8. git-daemon 服务器

## 8.1. git-daemon - A really simple server for git repositories

在/home/gitroot/ 上运行 git 守护进程

```
$ cd /home/gitroot
$ mkdir test.git
$ cd test.git
$ git --bare init --shared
Initialized empty shared Git repository in
/home/gitroot/test.git/
```

```
git daemon --verbose --export-all --base-path=/home/gitroot --
enable=receive-pack --reuseaddr
```

允许push,否则该仓库只能clone/pull

```
sudo git daemon --verbose --export-all --base-
path=/home/gitroot --enable=upload-pack --enable=upload-archive
--enable=receive-pack
```

或者增加配置项

```
$ git config daemon.receivepack true
$ git config --file config receive.denyCurrentBranch ignore
```

## 8.2. git-daemon-sysvinit

for a read-only repo:

```
$ sudo apt-get install git-daemon-sysvinit


$ dpkg -l git-daemon-sysvinit
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-
aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                     Version
Architecture             Description
+++-========================================-
========================-========================-
================================================================
===================
ii  git-daemon-sysvinit                      1:1.7.10.4-1ubuntu1
all                      fast, scalable, distributed revision
control system (git-daemon service)


$ dpkg -L git-daemon-sysvinit
/.
/usr
/usr/share
/usr/share/git-core
/usr/share/git-core/sysvinit
/usr/share/git-core/sysvinit/sentinel
/usr/share/doc
/usr/share/doc/git-daemon-sysvinit
/usr/share/doc/git-daemon-sysvinit/copyright
/usr/share/doc/git-daemon-sysvinit/README.Debian
/etc
/etc/default
/etc/default/git-daemon
/etc/init.d
/etc/init.d/git-daemon
/usr/share/doc/git-daemon-sysvinit/changelog.Debian.gz
```

配置 /etc/default/git-daemon 文件

## 8.3. inet.conf / xinetd 方式启动

过程 11.2. git-daemon

1. /etc/shells

   /etc/shells 最后一行添加 '/usr/bin/git-shell'

```
$ grep git /etc/shells
/usr/bin/git-shell
```

2. add new user 'git' and 'gitroot' for git

   you need to assign shell with /usr/bin/git-shell

```
$ sudo adduser git --shell /usr/bin/git-shell
$ sudo adduser gitroot --ingroup git --shell /bin/bash
```

   /etc/passwd

```
$ grep git /etc/passwd
git:x:1001:1002:,,,:/home/git:/usr/bin/git-shell
gitroot:x:1002:1002:,,,:/home/gitroot:/bin/bash
```

3. /etc/services

```
$ grep 9418 /etc/services
git             9418/tcp                        # Git
Version Control System
```

4. /etc/inet.conf

```
$ grep git /etc/inet.conf
git     stream  tcp     nowait  nobody \
  /usr/bin/git-daemon git-daemon --inetd --syslog --export-
```

```
all /home/gitroot
```

reload inetd

```
$ sudo pkill -HUP inetd
```

5. xinetd

目前的Linux逐渐使用xinetd.d替代inet.conf，如Redhat系列已经不再使用inet.conf, Ubuntu系列发行版已经不预装inet与xinetd

```
$ apt-cache search xinetd
globus-gfork-progs - Globus Toolkit - GFork Programs
rlinetd - gruesomely over-featured inetd replacement
update-inetd - inetd configuration file updater
xinetd - replacement for inetd with many enhancements

$ sudo apt-get install xinetd
```

/etc/xinetd.d/

```
$ cat /etc/xinetd.d/git
# default: off
# description: The git server offers access to git
repositories
service git
{
        disable                 = no
        type            = UNLISTED
        port            = 9418
        socket_type     = stream
        protocol                = tcp
        wait            = no
        user            = gitroot
        server          = /usr/bin/git
        server_args     = daemon --inetd --export-all --
enable=receive-pack --reuseaddr --base-path=/home/gitroot
```

```
        log_on_failure  += USERID
}
```

reload xinitd

```
$ sudo /etc/init.d/xinetd reload
 * Reloading internet superserver configuration xinetd
[ OK ]
```

## 8.4. git-daemon-run

```
$ sudo apt-get install git-daemon-run
```

安装后会创建下面两个用户

```
$ cat /etc/passwd | grep git
gitlog:x:117:65534::/nonexistent:/bin/false
gitdaemon:x:118:65534::/nonexistent:/bin/false
```

git-daemon-run 包携带的文件

```
$ dpkg -L git-daemon-run
/.
/etc
/etc/sv
/etc/sv/git-daemon
/etc/sv/git-daemon/run
/etc/sv/git-daemon/log
/etc/sv/git-daemon/log/run
/usr
/usr/share
/usr/share/doc
/usr/share/doc/git-daemon-run
/usr/share/doc/git-daemon-run/changelog.gz
/usr/share/doc/git-daemon-run/changelog.Debian.gz
```

```
/usr/share/doc/git-daemon-run/README.Debian
/usr/share/doc/git-daemon-run/copyright
```

同时创建下面配置文件

```
$ find /etc/sv/git-daemon/
/etc/sv/git-daemon/
/etc/sv/git-daemon/run
/etc/sv/git-daemon/supervise
/etc/sv/git-daemon/log
/etc/sv/git-daemon/log/run
/etc/sv/git-daemon/log/supervise
```

编辑/etc/sv/git-daemon/run配置

```
$ sudo vim /etc/sv/git-daemon/run

#!/bin/sh
exec 2>&1
echo 'git-daemon starting.'
exec chpst -ugitdaemon \
  "$(git --exec-path)"/git-daemon --verbose --reuseaddr \
    --base-path=/var/cache /var/cache/git
```

```
git-daemon --verbose --reuseaddr \
    --base-path=/var/cache /var/cache/git

改为

git-daemon --verbose --reuseaddr \
        --enable=receive-pack --export-all --base-path=/opt/git
```

**提示**

```
   * 我加上了一个--export-all 使用该选项后，在git仓库中就不必创建git-
daemon-export-ok文件。

   其他选项--enable=upload-pack --enable=upload-archive --
enable=receive-pack
```

/etc/services 文件中加入

```
# Local services
git             9418/tcp                        # Git Version
Control System
```

确认已经加入

```
$ grep 9418 /etc/services
```

启动git-daemon

```
$ sudo sv stop git-daemon

or

$ sudo runsv git-daemon
runsv git-daemon: fatal: unable to change to directory: file
does not exist
```

扫描git端口，确认git-daemon已经启动

```
$ nmap localhost

Starting Nmap 5.00 ( http://nmap.org ) at 2012-01-31 10:45 CST
Warning: Hostname localhost resolves to 2 IPs. Using 127.0.0.1.
Interesting ports on localhost (127.0.0.1):
Not shown: 989 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
```

```
22/tcp   open  ssh
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1723/tcp open  pptp
3128/tcp open  squid-http
3306/tcp open  mysql
9418/tcp open  git
```

## 8.5. Testing

```
$ sudo mkdir -p /opt/git/example.git
$ cd /opt/git/example.git
$ git init
$ sudo vim example.git/.git/config
[receive]
denyCurrentBranch = ignore

$ sudo chown gitdaemon -R /opt/git/*
$ touch git-daemon-export-ok
```

.git/config 文件应该是下面这样

```
$ cat example.git/.git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true

[receive]
denyCurrentBranch = ignore
```

git-clone git://localhost/example.git

```
neo@deployment:/tmp$ git clone git://localhost/example.git
example.git
Cloning into example.git...
warning: You appear to have cloned an empty repository.
neo@deployment:/tmp$ cd example.git/
neo@deployment:/tmp/example.git$ echo helloworld > hello.txt
neo@deployment:/tmp/example.git$ git add hello.txt
neo@deployment:/tmp/example.git$ git commit -m 'Initial commit'
[master (root-commit) 65a0f83] Initial commit
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
```

我们添加了一些文件 push 到服务器

```
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 214 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git://localhost/example.git
 * [new branch]      master -> master
```

然后再git clone，可以看到文件数目

```
$ git-clone git://localhost/example.git
Cloning into example...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
```

# 9. git config

## 9.1. core.sshCommand

git 默认使用 id_rsa，指定私钥方法是：

```
git config core.sshCommand "ssh -i ~/.ssh/id_rsa_example -F
/dev/null"
git pull
git push
```

```
GIT_SSH_COMMAND="ssh -i ~/.ssh/id_rsa_example -F /dev/null" git
clone git@github.com:netkiller/netkiller.github.io.git
```

## 9.2. fatal: The remote end hung up unexpectedly

```
error: RPC failed; result=22, HTTP code = 413 | 18.24 MiB/s
fatal: The remote end hung up unexpectedly
```

```
git config http.postBuffer 524288000
```

## 9.3. 忽略 SSL 检查

使用自颁发 ssl 证书时需要开启，否则无法 clone 和 push

```
export GIT_SSL_NO_VERIFY=true
```

```
git config http.sslVerify "false"
```

# 10. git-svn - Bidirectional operation between a single Subversion branch and git

```
sudo apt-get install git-svn
```

clone

```
git-svn clone -s svn://netkiller.8800.org/neo
cd neo
git gc

git commit -a
git-svn dcommit
```

从 svn 仓库更新

```
git-svn rebase
```

git-svn init svn://netkiller.8800.org/neo/public_html

# 11. .gitignore

```
find ./ -type d -empty | grep -v \.git | xargs -i touch
{}/.gitignore
```

# 12. .gitattributes

## 12.1. SVN Keywords

```
Example:

$ echo '*.txt ident' >> .gitattributes
$ echo '$Id$' > test.txt
$ git commit -a -m "test"

$ rm test.txt
$ git checkout -- test.txt
$ cat test.txt
```

# 13. gitolite - SSH-based gatekeeper for git repositories

```
$ apt-cache search gitolite
gitolite - SSH-based gatekeeper for git repositories
```

```
sudo apt-get install gitolite
```

No adminkey given - not setting up gitolite.

## 13.1. gitolite-admin

```
git@192.168.2.1:gitolite-admin.git
```

### gitolite.conf

gitolite-admin/conf/gitolite.conf

#### staff

```
@admin          = neo
@developer      = bottle nada dick blank phabricator
@designer       = blank
@deployer       = phoenix
@tester         = jimmy
```

#### repo

```
repo gitolite-admin
```

```
    RW+     = @admin
    R       = @deployer

repo mydomain.com/www.mydomain.com
    RW+     = @admin
    RW          = @developer @designer
    R           = @deployer

repo mydomain.com/images.mydomain.com
    RW+     = @admin
    RW          = @developer @designer
    R           = @deployer

repo mydomain.com/passport.mydomain.com
    RW+     = @admin
    RW          = @developer
    R           = @deployer @designer

repo    example.com/www.example.com
  RW+       = @all

repo    @all
  RW        = @developer @designer
  R         = @agentbot @deployment @test
```

# 14. Web Tools

## 14.1. viewgit

http://viewgit.sourceforge.net/

# 15. FAQ

## 15.1. 导出最后一次修改过的文件

有时我们希望把刚刚修改的文件复制出来，同时维持原有的目录结构，这样可能交给运维直接覆盖服务器上的代码。我们可以使用下面的命令完成这样的操作，而不用一个一个文件的复制。

```
git archive -o update.zip HEAD $(git diff --name-only HEAD^)
```

## 15.2. 导出指定版本区间修改过的文件

首先使用git log查看日志，找到指定的 commit ID号。

```
$ git log
commit ee808bb4b3ed6b7c0e7b24eeec39d299b6054dd0
Author: 168 <lineagelx@126.com>
Date:   Thu Oct 22 13:12:11 2015 +0800

    统计代码

commit 3e68ddef50eec39acea1b0e20fe68ff2217cf62b
Author: netkiller <netkiller@msn.com>
Date:   Fri Oct 16 14:39:10 2015 +0800

    页面修改

commit b111c253321fb4b9c5858302a0707ba0adc3cd07
Author: netkiller <netkiller@msn.com>
Date:   Wed Oct 14 17:51:55 2015 +0800

    数据库连接

commit 4a21667a576b2f18a7db8bdcddbd3ba305554ccb
Author: netkiller <netkiller@msn.com>
Date:   Wed Oct 14 17:27:15 2015 +0800

    init repo
```

导入 b111c253321fb4b9c5858302a0707ba0adc3cd07 至 ee808bb4b3ed6b7c0e7b24eeec39d299b6054dd0 间修改过的文件。

```
$ git archive -o update2.zip HEAD $(git diff --name-only
```

```
b111c253321fb4b9c5858302a0707ba0adc3cd07)
```

## 15.3. 回撤提交

首先 reset 到指定的版本，根据实际情况选择 --mixed 还是 --hard

```
git reset --mixed 096392721f105686fc3cdafcb4159439a66b0e5b --
or
git reset --hard 33ba6503b4fa8eed35182262770e4eab646396cd --
```

```
git push origin --force --all
or
git push --force --progress "origin" master:master
```

## 15.4. 每个项目一个证书

方法一

```
[root@localhost ~]# cat .ssh/config
host git.netkiller.cn
    user git
    hostname git.netkiller.cn
    port 22
    identityfile ~/.ssh/netkiller

host github.com
 HostName github.com
 IdentityFile ~/.ssh/id_rsa_github
 User git
```

方法二

```
$ ssh-agent sh -c 'ssh-add ~/.ssh/id_rsa; git fetch user@host'
```

# 第 12 章 Subversion

# 1. Invoking the Server

配置开发环境版本控制Subversion

Squid + Subversion 请参考Squid一节

## 1.1. Installing

### Ubuntu

过程 12.1. subversion

1. installation

   **$ sudo apt-get install subversion**

   ```
   $ sudo apt-get install subversion
   ```

2. create svn group and svnroot user

   ```
   $ sudo groupadd svn
   $ sudo adduser svnroot --ingroup svn
   ```

3. create repository

   ```
   $ svnadmin create /home/svnroot/test
   ```

4. testing

```
svnroot@netkiller:~$ svnserve -d --foreground -r
/home/svnroot/
```

check out

neo@netkiller:/tmp$ svn list svn://localhost/test

you may see some file and directory

```
neo@netkiller:/tmp$ ls test/.svn/
entries  format  prop-base  props  text-base  tmp
```

5. configure

$ vim repositories/conf/svnserve.conf

```
[general]
anon-access = read
auth-access = write
password-db = passwd
# authz-db = authz
# realm = My First Repository
```

$ vim repositories/conf/passwd

```
[users]
# harry = harryssecret
# sally = sallyssecret
neo = chen
```

如果不允许匿名用户checkout代码，配置文件这样写anon-access = none

```
[general]
anon-access = none
auth-access = write
```

6. firewall

```
$ sudo ufw allow svn
```

## CentOS 5

```
[root@development ~]# yum -y install subversion
```

**classic Unix-like xinetd daemon**

```
[root@development ~]# vim /etc/xinetd.d/subversion
service subversion
{
    disable = no
    port = 3690
    socket_type = stream
    protocol = tcp
    wait = no
    user = svnroot
    server = /usr/bin/svnserve
    server_args = -i -r /home/svnroot
}
```

firewall

```
iptables -A INPUT -p tcp -m tcp --sport 3690 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 3690 -j ACCEPT
```

**WebDav**

install webdav module

```
[root@development ~]# yum install mod_dav_svn
```

create directory

```
mkdir /var/www/repository
svnadmin create /var/www/repository
```

subversion.conf

```
[root@development ~]# vim /etc/httpd/conf.d/subversion.conf
LoadModule dav_module modules/mod_dav.so
LoadModule dav_svn_module modules/mod_dav_svn.so
LoadModule authz_svn_module modules/mod_authz_svn.so
```

vhost.conf

```
<Location />
```

```
DAV svn
SVNPath /var/www/repository
AuthType Basic
AuthName "Subversion Repository"
AuthUserFile /etc/subversion/svn-auth-file
Require valid-user
</Location>
```

auth file

```
[root@development ~]# htpasswd -c /etc/subversion/svn-auth-file
my_user_name
```

项目目录结构

–trunk #存放主线

–branches #存放分支，可修改

–tags #存放标记，不可修改

## CentOS 6

CentOS 6 默认没有安装xinetd

```
# yum install xinetd
# yum install subversion

# mkdir -p /opt/svnroot
```

xinetd 配置

```
# vim /etc/xinetd.d/svn
service svn
{
    disable = no
    port = 3690
    socket_type = stream
    protocol = tcp
    wait = no
    user = svnroot
    server = /usr/bin/svnserve
    server_args = -i -r /opt/svnroot
}

# /etc/init.d/xinetd restart
Stopping xinetd:
[FAILED]
Starting xinetd:                                                [
OK  ]

# tail /var/log/messages | grep xinetd
May  5 18:57:20 SZVM42-C1-02 yum: Installed: 2:xinetd-2.3.14-
16.el5.i386
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: Unknown user:
svnroot [file=/etc/xinetd.d/svn] [line=8]
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: Error parsing
attribute user - DISABLING SERVICE

[file=/etc/xinetd.d/svn] [line=8]
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: xinetd Version
2.3.14 started with libwrap loadavg labeled-networking

options compiled in.
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: Started working: 0
available services
```

service 名字必须与 /etc/services中定义的名字相同，否则将不能启动，同时在/var/log/message中会提示如下

```
May  4 14:33:08 www xinetd[5656]: service/protocol combination
not in /etc/services: subversion/tcp
May  4 14:33:08 www xinetd[5656]: xinetd Version 2.3.14 started
```

```
with libwrap loadavg labeled-networking options compiled in.
May  4 14:33:08 www xinetd[5656]: Started working: 0 available
services
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:41 www xinetd[5656]: Exiting...
May  4 14:33:41 www xinetd[5676]: xinetd Version 2.3.14 started
with libwrap loadavg labeled-networking options compiled in.
May  4 14:33:41 www xinetd[5676]: Started working: 1 available
service
```

## 1.2. standalone “daemon” process

svn daemon

```
$ svnserve --daemon --root /home/svnroot
```

### starting subversion for debian/ubuntu

/etc/init.d/subversion for debian/ubuntu

```
debian:/etc/init.d# cat subversion
#!/bin/sh
```

```
### BEGIN INIT INFO
# Provides:          subversion
# Required-Start:    $remote_fs $network
# Required-Stop:     $remote_fs $network
# Should-Start:      fam
# Should-Stop:       fam
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start the subversion subversion server.
### END INIT INFO

#########################
# Author: Neo <openunix@163.com>
#########################

PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/bin/svnserve
NAME=subversion
DESC="subversion server"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
SVNROOT=/srv/svnroot
DAEMON_OPTS="-d -T -r $SVNROOT --pid-file $PIDFILE"

test -x $DAEMON || exit 0

set -e

. /lib/lsb/init-functions

case "$1" in
    start)
        log_daemon_msg "Starting $DESC" $NAME
        echo
        $DAEMON $DAEMON_OPTS
        echo `pgrep -o $NAME` > $PIDFILE > /dev/null 2>
/dev/null
        ;;
    stop)
        log_daemon_msg "Stopping $DESC" $NAME
        echo
        killall `basename $DAEMON` > /dev/null 2> /dev/null
        rm -rf $PIDFILE
        ;;
    restart)
```

```
        $0 stop
        $0 start
        ;;
    status)
        ps ax | grep $NAME
        ;;
    *)
        echo "Usage: $SCRIPTNAME {start|stop|restart|status}"
>&2
        exit 1
        ;;
esac

exit 0
```

## starting subversion daemon script for CentOS/Radhat

```
#!/bin/bash
#
# /etc/rc.d/init.d/subversion
#
# Starts the Subversion Daemon
#
# chkconfig: 345 90 10
#
# description: Subversion Daemon

# processname: svnserve

source /etc/rc.d/init.d/functions

[ -x /usr/bin/svnserve ] || exit 1

### Default variables
SYSCONFIG="/etc/sysconfig/subversion"

### Read configuration
[ -r "$SYSCONFIG" ] && source "$SYSCONFIG"

RETVAL=0
```

```
USER="svnroot"
prog="svnserve"
desc="Subversion Daemon"

start() {
   echo -n $"Starting $desc ($prog): "
   daemon --user $USER $prog -d $OPTIONS
   RETVAL=$?
   [ $RETVAL -eq 0 ] && touch /var/lock/subsys/$prog
   echo
}

stop() {
   echo -n $"Shutting down $desc ($prog): "
   killproc $prog
   RETVAL=$?
   [ $RETVAL -eq 0 ] && success || failure
   echo
   [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog
   return $RETVAL
}

case "$1" in
  start)
   start
   ;;
  stop)
   stop
   ;;
  restart)
   stop
   start
   RETVAL=$?
   ;;
  condrestart)
        [ -e /var/lock/subsys/$prog ] && restart
   RETVAL=$?
   ;;
  *)
   echo $"Usage: $0 {start|stop|restart|condrestart}"
   RETVAL=1
esac

exit $RETVAL
```

/etc/sysconfig/subversion

```
# Configuration file for the Subversion service

#
# To pass additional options (for instace, -r root of directory
to server) to
# the svnserve binary at startup, set OPTIONS here.
#
#OPTIONS=
OPTIONS="--threads --root /srv/svnroot"
```

## 1.3. classic Unix-like inetd daemon

/etc/inetd.conf

```
svn stream tcp nowait svn /usr/bin/svnserve svnserve -i -r
/home/svnroot/repositories
```

xinetd.d

/etc/xinetd.d/subversion

```
$ sudo apt-get install xinetd
$ sudo vim /etc/xinetd.d/subversion

service subversion
{
    disable = no
    port = 3690
    socket_type = stream
    protocol = tcp
    wait = no
    user = svnroot
    server = /usr/bin/svnserve
```

```
    server_args = -i -r /home/svnroot
}
```

restart xinetd

```
$ sudo /etc/init.d/xinetd restart
```

## 1.4. hooks

```
$ sudo apt-get install subversion-tools
```

### post-commit

install SVN::Notify

```
perl -MCPAN -e 'install SVN::Notify'
```

```
$ sudo cp post-commit.tmpl post-commit
$ sudo chown svnroot:svn post-commit
$ sudo vim post-commit

REPOS="$1"
REV="$2"

#/usr/share/subversion/hook-scripts/commit-email.pl "$REPOS"
"$REV" openunix@163.com
/usr/share/subversion/hook-scripts/commit-email.pl "$1" "$2" --
from neo@netkiller.8800.org -h localhost  -s "[SVN]" --diff y
openunix@163.com openx@163.com
```

另一种方法

```
#!/bin/sh

REPOS="$1"
REV="$2"

/usr/local/bin/svnnotify                    \
    --repos-path    "$REPOS"                \
    --revision      "$REV"                  \
    --subject-cx                            \
    --with-diff                             \
    --handler       HTML::ColorDiff         \
    --to            <your e-mail address>   \
    --from          <from e-mail address>
```

```
/usr/bin/svnnotify --repos-path "$REPOS" --revision "$REV" \
--from neo@netkiller.8800.org --to openunix@163.com --smtp
localhost \
--handler "HTML::ColorDiff"  --with-diff --charset zh_CN:GB2312
-g zh_CN --svnlook /usr/bin/svnlook --subject-prefix '[SVN]'
```

如果你没有安装邮件服务器，你可以使用服务商的SMTP如163.com

```
/usr/bin/svnnotify --repos-path "$REPOS" --revision "$REV" \
--from openx@163.com --to openunix@163.com --smtp smtp.163.com
--smtp-user openunix --smtp-pass ****** \
--handler "HTML::ColorDiff"  --with-diff --charset UTF-8 --
language zh_CN --svnlook /usr/bin/svnlook --subject-prefix
'[SVN]'
```

Charset

```
REPOS="$1"
REV="$2"
```

```
svnnotify --repos-path "$REPOS" --revision "$REV" \
        --subject-cx \
        --from neo.chen@example.com \
        --to group@example.com,manager@example.com \
        --with-diff \
        --svnlook /usr/bin/svnlook \
        --subject-prefix '[SVN]' \
        --charset UTF-8  --language zh_CN
```

## 1.5. WebDav

Apache SVN

**$ sudo apt-get install libapache2-svn**

```
netkiller@neo:/etc/apache2$ sudo apt-get install libapache2-svn
```

vhost

```


<VirtualHost *>
        ServerName svn.netkiller.8800.org
        DocumentRoot /var/svn

      <Location />
                DAV svn
                SVNPath /var/svn
                AuthType Basic
                AuthName "Subversion Repository"
                AuthUserFile /etc/apache2/svn.passwd
                <LimitExcept GET PROPFIND OPTIONS REPORT>
                        Require valid-user
                </LimitExcept>
        </Location>
</VirtualHost>
```

建立密码文件

建立第一个用户需要加-c参数

```
netkiller@neo:/etc/apache2$ sudo htpasswd2 -c
/etc/apache2/svn.passwd svn
New password:
Re-type new password:
Adding password for user svn
```

输入两次密码

建立其他用户

```
sudo htpasswd2 /etc/apache2/svn.passwd otheruser
```

**davfs2 - mount a WebDAV resource as a regular file system**

install

```
$ sudo apt-get install davfs2
```

mount a webdav to directory

```
$ sudo mount.davfs https://opensvn.csie.org/netkiller
/mnt/davfs/
Please enter the username to authenticate with server
https://opensvn.csie.org/netkiller or hit enter for none.
Username: svn
Please enter the password to authenticate user svn with server
https://opensvn.csie.org/netkiller or hit enter for none.
Password:
mount.davfs: the server certificate is not trusted
  issuer:      CSIE.org, CSIE.org, Taipei, Taiwan, TW
  subject:     CSIE.org, CSIE.org, Taipei, TW
  identity:    *.csie.org
  fingerprint:
```

```
e6:05:eb:fb:69:5d:25:4e:11:3c:83:e8:7c:44:ee:bf:a9:85:a3:64
You only should accept this certificate, if you can
verify the fingerprint! The server might be faked
or there might be a man-in-the-middle-attack.
Accept certificate for this session? [y,N] y
```

test

```
$ ls davfs/
branches  lost+found  tags  trunk
```

# 2. repository 管理

## 2.1. create repository

```
$ su - svnroot
$ svnadmin create /home/svnroot/neo
```

## 2.2. user admin

```
#!/bin/bash
################################
# Author: Neo<openunix@163.com
# Home: http://netkiller.sf.net
################################
SVNROOT=/srv/svnroot/project
adduser(){
    echo $1  $2
    if [ -z $1 ]; then
        usage
    else
        local user=$1
    fi
    if [ -z $2 ]; then
        usage
    else
        local passwd=$2
    fi
    echo "$1 = $2" >> $SVNROOT/conf/passwd
}
deluser(){
    local user=$1
    if [ -z $user ]; then
        usage
    else
        ed -s $SVNROOT/conf/passwd <<EOF
/$user/
d
```

```
wq
EOF
    fi
}
list(){
    cat $SVNROOT/conf/passwd
}
usage(){
    echo $"Usage: $0 {list|add|del} username"
}
case "$1" in
    list)
        list
        ;;
    add)
        adduser $2 $3
        ;;
    del)
        deluser $2
        ;;
    restart)
        stop
        start
        ;;
    condrestart)
        condrestart
        ;;

    *)
        usage
        exit 1
esac
```

用法

```
./svnuser list
./svnuser add user passwd
./svnuser del user
```

## 2.3. authz

$ svnadmin create /home/svnroot/project

$ svnserve --daemon --root /home/svnroot/project

```
[groups]
member = neo
blog = neo,netkiller
wiki = bg7nyt,chen,jingfeng

[/]
* =

[/member]
@member = rw
* = r

[/app/blog]
@blog = rw
* =

[/app/wiki]
@blog = rw
* =

# [repository:/baz/fuz]
# @harry_and_sally = rw
# * = r
```

$ svnadmin create /home/svnroot/project1

$ svnadmin create /home/svnroot/project2

$ svnserve --daemon --root /home/svnroot

```
[groups]
member = neo
blog = neo,netkiller
wiki = bg7nyt,chen,jingfeng

[project1:/]
```

```
* =
[project2:/]
* = r

[project1:/member]
@member = rw
* = r

[project2:/app/blog]
@blog = rw
* =

[project2:/app/wiki]
@blog = rw
* = r
```

## 例 12.1. authz

```

[aliases]
# joe = /C=XZ/ST=Dessert/L=Snake City/O=Snake Oil,
Ltd./OU=Research Institute/CN=Joe Average

### This file is an example authorization file for svnserve.
### Its format is identical to that of mod_authz_svn
authorization
### files.
### As shown below each section defines authorizations for the
path and
### (optional) repository specified by the section name.
### The authorizations follow. An authorization line can refer
to:
###  - a single user,
###  - a group of users defined in a special [groups] section,
###  - an alias defined in a special [aliases] section,
###  - all authenticated users, using the '$authenticated'
token,
###  - only anonymous users, using the '$anonymous' token,
###  - anyone, using the '*' wildcard.
###
### A match can be inverted by prefixing the rule with '~'.
Rules can
### grant read ('r') access, read-write ('rw') access, or no
access
```

```
### ('').

[aliases]
# joe = /C=XZ/ST=Dessert/L=Snake City/O=Snake Oil,
Ltd./OU=Research Institute/CN=Joe Average

[groups]

manager = neo
developer = jam,john,zen
tester = eva
designer = allan
deployer = ken

[/]
@manager = rw
@developer = r
@designer = r
@deployer = r
@tester = r
* =

#############################
# Trunk
# #########################
[/www.mydomain.com/trunk]
@manager = rw
@designer = rw
@developer = rw
@deployer = r

[/images.mydomain.com/trunk]
@designer = rw

[/myid.mydomain.com/trunk]
@designer = r

[/info.mydomain.com/trunk]
@developer = r
@designer = r

#############################
#\Branches
#############################
[/admin.mydomain.com/branches]
```

```
@developer = rw
@designer = rw

[/myid.mydomain.com/branches]
@developer = rw
@designer = rw

[/info.mydomain.com/branches]
@developer = rw
@designer = rw

[/www.mydomain.com/branches]
@developer = rw
@designer = rw

[/images.mydomain.com/branches]
@developer = rw
@designer = rw

[/log.mydomain.com/branches]
@developer = rw

[/report.mydomain.com/branches]
@developer = rw

##############################
# TAGS
# ############################
[/myid.mydomain.com/tags]
@deployer = rw
[/admin.mydomain.com/tags]
@deployer = rw
[/info.mydomain.com/tags]
@deployer = rw
```

## 2.4. dump

```
svnadmin dump /svnroot/project | gzip > svn.gz
```

# 3. 使用Subversion

## 3.1. Initialized empty subversion repository for project

```
svn co svn://127.0.0.1/project
cd project
mkdir trunk
mkdir tags
mkdir branches
svn ci -m "Initialized empty subversion repository in
your_project"
```

## 3.2. ignore

### svn propset svn:ignore [filename] [folder]

```
$ svn propset svn:ignore 'images' .
$ svn ci -m 'Ignoring a directory called "images".'
```

```
$ svn propset svn:ignore '*' images
$ svn ci -m 'Ignoring a directory called "images".'
```

```
$ svn export spool spool-tmp
$ svn rm spool
$ svn ci -m 'Removing inadvertently added directory "spool".'
$ mv spool-tmp spool
$ svn propset svn:ignore 'spool' .
$ svn ci -m 'Ignoring a directory called "spool".'
```

.ignore

```
svn propset svn:ignore -F .cvsignore .
```

```
svn propset -R svn:ignore -F .svnignore .
```

## 3.3. 关键字替换

```
Date
         这个关键字保存了文件最后一次在版本库修改的日期，看起来类似于
$Date: 2012-08-06 17:43:09 +0800 (Mon, 06 Aug 2012) $，它也可以用
LastChangedDate来指定。
Revision
         这个关键字描述了这个文件最后一次修改的修订版本，看起来像
$Revision: 446 $，也可以通过LastChangedRevision或者Rev引用。
Author
         这个关键字描述了最后一个修改这个文件的用户，看起来类似$Author:
netkiller $，也可以用LastChangedBy来指定。
HeadURL
         这个关键字描述了这个文件在版本库最新版本的完全URL，看起来类似
$HeadURL:
https://svn.code.sf.net/p/netkiller/svn/trunk/Docbook/Version/s
ection.version.svn.xml $，可以缩写为URL。
Id
         这个关键字是其他关键字一个压缩组合，它看起来就像$Id:
section.version.svn.xml 446 2012-08-06 09:43:09Z netkiller $，可
以解释为文件calc.c上一次修改的修订版本号是148，时间是2006年7月28日，作者
是sally。
```

```
$ cat weather.txt
$Id: section.version.svn.xml 446 2012-08-06 09:43:09Z netkiller
$

$ svn propset svn:keywords "Id" weather.txt
property 'svn:keywords' set on 'weather.txt'

$ cat weather.txt
$Id: section.version.svn.xml 446 2012-08-06 09:43:09Z netkiller
$
```

设置多个关键字

```
$ svn propset svn:keywords "Author HeadURL Id Revision" -R
*.php
```

```
svn -R propset svn:keywords -F .keywords *
```

## 3.4. lock 加锁/ unlock 解锁

$ svn lock -m "LockMessage" [–force] PATH

```
$ svn lock -m “lock test file“ test.php
$ svn unlock PATH
```

## 3.5. import

```
svn import [PATH] URL
svn export URL [PATH]
```

## 3.6. export 指定版本

```
svn log file
svn export -r rxxxxx file
or
svn export -r rxxxxx file newfile
svn ci -m "restore rxxxxxx"
```

## 3.7. 修订版本关键字

```
HEAD

    版本库中最新的（或者是“最年轻的”）版本。
```

```
BASE

    工作拷贝中一个条目的修订版本号，如果这个版本在本地修改了，则“BASE版本”
就是这个条目在本地未修改的版本。
COMMITTED

    项目最近修改的修订版本，与BASE相同或更早。
PREV

    一个项目最后修改版本之前的那个版本，技术上可以认为是COMMITTED -1。
```

```
$ svn cat -r PREV filename > filename
$ svn diff -r PREV filename
```

## 3.8. 恢复旧版本

svn没有恢复旧版本的直接功能，不过可以使用svn merge命令恢复。比如说当前HEAD为2，而我要恢复成1版本，怎么做？

用svn merge：

```
svn update
svn merge --revision 2:1 svn://localhost/lynn
svn commit -m "restore to revision 1"
```

svn merge --r HEAD:1 svn://localhost/lynn

# 4. branch

## 4.1. create

create a new branch using copy

```
svn cp http://www.domain.com/truck/project
http://www.domain.com/branches/project_branch_1
```

## 4.2. remove

remove

```
svn rm http://www.domain.com/branches/project_branch_1
```

## 4.3. switch

```
svn switch http://www.domain.com/branches/project_branch_2 .
```

## 4.4. merge

```
svn -r 148:149 merge svn://server/trunk branches/module
```

## 4.5. relocate

switch --relocate FROM TO [PATH...]

```
svn switch --relocate svn://192.168.3.9/neo
```

```
svn://192.168.3.5/neo .
```

# 5. FAQ

## 5.1. 递归添加文件

```
$ svn add `svn st | grep '?' | awk '{print $2}'`
```

## 5.2. 清除项目里的所有.svn目录

```
find . -type d -iname ".svn" -exec rm -rf {} \;
```

## 5.3. color diff

http://colordiff.sourceforge.net/

```
$ sudo apt-get install colordiff
```

add the following to your ~/.bashrc

```
alias svndiff='svn diff --diff-cmd=colordiff'
```

## 5.4. cvs2svn

http://cvs2svn.tigris.org/

```
[root@development ~]# cvs2svn --encoding=gb2312 --fallback-
encoding=utf_8 --existing-svnrepos --svnrepos /home/svnroot
/home/cvsroot
[root@development ~]# cvs2svn --encoding=gb2312 --fallback-
```

```
encoding=utf_8 --svnrepos /home/svnroot /home/cvsroot
```

## 5.5. Macromedia Dreamweaver MX 2004 + WebDAV +Subversion

首先进入站点管理

单击新建(New…)按钮选择站点(Site)

显示站点设置面版 Local Info 中设置

Remote Info 中设置

单击设置按钮（settings）

单击确定

单击Done完成

连接 WebDAV 服务器

单击

连接

## 5.6. 指定用户名与密码

```
svn co svn://www.example.com/repos  --username neo --password
chen;
```

# 第 13 章 cvs - Concurrent Versions System

## 1. installation

过程 13.1. install cvs

1. install

```
$ sudo apt-get install xinetd
$ sudo apt-get install cvs
```

show the cvs version

```
$ cvs -v

Concurrent Versions System (CVS) 1.12.13 (client/server)
```

2. create cvs group and cvsroot user

```
$ sudo groupadd cvs
$ sudo adduser cvsroot --ingroup cvs
```

change user become cvsroot

```
$ su - cvsroot
```

3. initialization 'CVSROOT'

```
$ cvs -d /home/cvsroot init
```

if you have successed, you can see CVSROOT directory in the '/home/cvsroot'

```
$ ls /home/cvsroot/
CVSROOT
```

4. authentication

   default SystemAuth=yes, you can use system user to login cvs.

   but usually, we don't used system user because it isn't security.

   SystemAuth = no

   edit '/home/cvsroot/CVSROOT/config' make sure SystemAuth = no

```
$ vim /home/cvsroot/CVSROOT/config
SystemAuth = no
```

   create passwd file

   the format is user:password:cvsroot

   you need to using htpasswd command, if you don't have, please install it as the following

```
$ sudo apt-get install apache2-utils
```

   or

```
$ perl -e 'print("userPassword:
".crypt("secret","salt")."\n");'
```

or

```
$ cat passwd
#!/usr/bin/perl
srand (time());
my $randletter = "(int (rand (26)) + (int (rand (1) + .5) %
2 ? 65 : 97))";
my $salt = sprintf ("%c%c", eval $randletter, eval
$randletter);
my $plaintext = shift; my $crypttext = crypt ($plaintext,
$salt);
print "${crypttext}\n";

$ ./passwd "mypasswd"
atfodI2Y/dcdc
```

let's using htpasswd to create a passwd

```
$ htpasswd -n neo
New password:
Re-type new password:
neo:yA50LI1BkXysY
```

copy 'neo:yA50LI1BkXysY' and add ':cvsroot' to the end

```
$ vim /home/cvsroot/CVSROOT/passwd
neo:yA50LI1BkXysY:cvsroot
nchen:GXaAkSKaQ/Hpk:cvsroot
```

5. Go into directory '/etc/xinetd.d/', and then create a cvspserver file as the following.

```
$ sudo vim /etc/xinetd.d/cvspserver

service cvspserver
{
```

```
   disable = no
   flags = REUSE
   socket_type = stream
   wait = no
   user = cvsroot
   server = /usr/bin/cvs
   server_args = -f --allow-root=/home/cvsroot pserver
   log_on_failure += USERID
}
```

6. check cvspserver in the '/etc/services'

```
$ grep cvspserver /etc/services
cvspserver      2401/tcp                        # CVS
client/server operations
cvspserver      2401/udp
```

7. restart xinetd

```
$ /etc/init.d/xinetd
Usage: /etc/init.d/xinetd {start|stop|reload|force-
reload|restart}
```

8. port

```
$ nmap localhost -p cvspserver

Starting Nmap 4.53 ( http://insecure.org ) at 2008-11-14
16:21 HKT
Interesting ports on localhost (127.0.0.1):
PORT     STATE SERVICE
2401/tcp open  cvspserver

Nmap done: 1 IP address (1 host up) scanned in 0.080
seconds
```

9. firewall

```
$ sudo ufw allow cvspserver
```

environment variable

CVSROOT=:pserver:username@ip:/home/cvsroot

```
vim .bashrc

export CVS_RSH=ssh
export CVSROOT=:pserver:neo@localhost:/home/cvsroot
```

test

```
$ cvs login
Logging in to :pserver:neo@localhost:2401/home/cvsroot
CVS password:
neo@netkiller:/tmp/test$ cvs co test
cvs checkout: Updating test
U test/.project
U test/NewFile.xml
U test/newfile.php
neo@netkiller:/tmp/test$
```

## 1.1. chroot

```
$ sudo apt-get install cvsd
```

environment variable

```
neo@netkiller:~/workspace/cvs$ export
```

```
CVSROOT=:pserver:neo@localhost:/home/cvsroot
```

ssh

```
export CVS_RSH=ssh
export CVSROOT=:ext:$USER@localhost:/home/cvsroot
```

# 2. cvs login | logout

```
neo@netkiller:~/workspace/cvs$ cvs login
Logging in to :pserver:neo@localhost:2401/home/cvsroot
CVS password:
```

logout

```
$ cvs logout
Logging out of :pserver:neo@localhost:2401/home/cvsroot
```

# 3. cvs import

cvs import -m "write some comments here" project_name vendor_tag release_tag

```
$ cvs import -m "write some comments here" project_name
vendor_tag release_tag
```

# 4. cvs checkout

```
$ cvs checkout project_name
cvs checkout: Updating project_name
```

checkout before

**cvs checkout -r release_1_0 project_name**

```
$ cvs checkout -r release_1_0 project_name
cvs checkout: Updating project_name
U project_name/file
cvs checkout: Updating project_name/dir1
U project_name/dir1/file1
cvs checkout: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

# 5. cvs update

about update

```
$ cvs update
$ cvs update -r HEAD
$ cvs update -r 1.5
$ cvs update -D now
$ cvs update -D now file
```

# 6. cvs add

```
$ cd project_name/
$ touch new_file
$ cvs add new_file
cvs add: scheduling file `new_file' for addition
cvs add: use `cvs commit' to add this file permanently
```

if the file is binary

cvs add -kb new_file.gif

add a directory

```
$ mkdir dir1
$ mkdir dir2
$ touch dir1/file1
$ touch dir2/file1
$ touch dir2/file2
$ cvs add dir1
? dir1/file1
Directory /home/cvsroot/project_name/dir1 added to the
repository
$ cvs add dir2
? dir2/file1
? dir2/file2
Directory /home/cvsroot/project_name/dir2 added to the
repository
```

add mulit files

```
$ cvs add dir1/file1
$ cvs add dir2/file?
```

# 7. cvs status

```
$ cvs status dir1/file1
cvs status: use `cvs add' to create an entry for `dir1/file1'
===================================================================
====
File: file1             Status: Unknown

   Working revision:  No entry for file1
   Repository revision: No revision control file
```

# 8. cvs commit

```
$ cvs commit -m "add a new file"
cvs commit: Examining .
/home/cvsroot/project_name/new_file,v  <--  new_file
initial revision: 1.1
```

commit multi files

```
$ cvs commit -m "add a new file" dir1/* dir2/*
/home/cvsroot/project_name/dir1/file1,v  <--  dir1/file1
initial revision: 1.1
/home/cvsroot/project_name/dir2/file1,v  <--  dir2/file1
initial revision: 1.1
/home/cvsroot/project_name/dir2/file2,v  <--  dir2/file2
initial revision: 1.1
```

# 9. cvs remove

```
$ rm -rf new_file
$ cvs remove new_file
cvs remove: scheduling `new_file' for removal
cvs remove: use `cvs commit' to remove this file permanently
$ cvs commit -m "delete file" new_file
/home/cvsroot/project_name/new_file,v  <--  new_file
new revision: delete; previous revision: 1.1
```

# 10. cvs log

let me create a file, and then modify the file to make several version

```
$ touch file
$ echo helloworld > file
$ cvs add file
cvs add: scheduling file `file' for addition
cvs add: use `cvs commit' to add this file permanently
$ cvs commit -m 'add file to cvs' file
/home/cvsroot/project_name/file,v  <--  file
initial revision: 1.1
$ echo I am Neo > file
$ cvs commit -m 'add file to cvs' file
/home/cvsroot/project_name/file,v  <--  file
new revision: 1.2; previous revision: 1.1
$ echo my nickname is netkiller > file
$ cvs commit -m 'modified file' file
/home/cvsroot/project_name/file,v  <--  file
new revision: 1.3; previous revision: 1.2
$ echo I am 28 years old > file
$ cvs commit -m 'modified file' file
/home/cvsroot/project_name/file,v  <--  file
new revision: 1.4; previous revision: 1.3
```

show log message

```
$ cvs log file

RCS file: /home/cvsroot/project_name/file,v
Working file: file
head: 1.4
branch:
locks: strict
access list:
symbolic names:
```

```
keyword substitution: kv
total revisions: 4;   selected revisions: 4
description:
----------------------------
revision 1.4
date: 2008-11-24 15:42:49 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: V0iuptfP43iETPrt;
modified file
----------------------------
revision 1.3
date: 2008-11-24 15:42:20 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: YWfYHFSV10duTPrt;
modified file
----------------------------
revision 1.2
date: 2008-11-24 15:41:47 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: 4iRs5fm1g9diTPrt;
add file to cvs
----------------------------
revision 1.1
date: 2008-11-24 15:41:28 +0800;  author: neo;  state: Exp;
commitid: zCWkxnWxLZHbTPrt;
add file to cvs
=============================================================
==============
```

cvs log -r1.2 file

```
$ cvs log -r1.2 file

RCS file: /home/cvsroot/project_name/file,v
Working file: file
head: 2.1
branch:
locks: strict
access list:
symbolic names:
        release_1_0_patch: 1.4.0.2
        release_1_0: 1.4
keyword substitution: kv
total revisions: 5;     selected revisions: 1
description:
```

```
----------------------------
revision 1.2
date: 2008-11-24 15:41:47 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: 4iRs5fm1g9diTPrt;
add file to cvs
=============================================================
==============
```

# 11. cvs annotate

```
$ cvs annotate file

Annotations for file
***************
2.2          (nchen    26-Nov-08): I am Neo
2.2          (nchen    26-Nov-08): My nickname netkiller
2.3          (nchen    26-Nov-08): I'm from shenzhen
1.4          (neo      24-Nov-08): I am 28 years old
```

# 12. cvs diff

```
neo@netkiller:~/workspace/cvs/project_name$ cvs diff -r1.3 -
r1.4 file
Index: file
===================================================================
====
RCS file: /home/cvsroot/project_name/file,v
retrieving revision 1.3
retrieving revision 1.4
diff -r1.3 -r1.4
1c1
< my nickname is netkiller
---
> I am 28 years old
neo@netkiller:~/workspace/cvs/project_name$ cvs diff -r1.2 -
r1.4 file
Index: file
===================================================================
====
RCS file: /home/cvsroot/project_name/file,v
retrieving revision 1.2
retrieving revision 1.4
diff -r1.2 -r1.4
1c1
< I am Neo
---
> I am 28 years old
```

--side-by-side

```
neo@netkiller:/tmp/cvs/test/project_name$ cvs diff --side-by-
side -r1.2 -r1.4 file
Index: file
===================================================================
====
RCS file: /home/cvsroot/project_name/file,v
```

```
retrieving revision 1.2
retrieving revision 1.4
diff --side-by-side -r1.2 -r1.4
I am Neo                                                                |
I am 28 years old
```

# 13. rename file

mv file_name new_file_name && cvs remove file_name
cvs add new_file_name

```
neo@netkiller:/tmp/cvs/project_name$ mv file file.txt
neo@netkiller:/tmp/cvs/project_name$ cvs remove file
cvs remove: scheduling `file' for removal
cvs remove: use `cvs commit' to remove this file permanently
neo@netkiller:/tmp/cvs/project_name$ cvs add file.txt
cvs add: scheduling file `file.txt' for addition
cvs add: use `cvs commit' to add this file permanently
neo@netkiller:/tmp/cvs/project_name$ cvs commit -m 'rename file
to file.txt'
cvs commit: Examining .
cvs commit: Examining dir1
cvs commit: Examining dir2
/home/cvsroot/project_name/file,v  <--  file
new revision: delete; previous revision: 2.3
/home/cvsroot/project_name/file.txt,v  <--  file.txt
initial revision: 1.1
```

# 14. revision

```
neo@netkiller:~/workspace/cvs/project_name$ cvs update -r 1.2
file
U file
neo@netkiller:~/workspace/cvs/project_name$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:  1.2
   Repository revision: 1.2
/home/cvsroot/project_name/file,v
   Commit Identifier: 4iRs5fm1g9diTPrt
   Sticky Tag:        1.2
   Sticky Date:       (none)
   Sticky Options:    (none)
```

last version

```
neo@netkiller:~/workspace/cvs/project_name$ cvs update -r HEAD
file
U file
neo@netkiller:~/workspace/cvs/project_name$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:  1.4
   Repository revision: 1.4
/home/cvsroot/project_name/file,v
   Commit Identifier: V0iuptfP43iETPrt
   Sticky Tag:        HEAD (revision: 1.4)
   Sticky Date:       (none)
   Sticky Options:    (none)
```

# 15. cvs export

### cvs export -r release_1_0 project_name

```
$ cvs export -r release_1_0 project_name
cvs export: Updating project_name
U project_name/file
cvs export: Updating project_name/dir1
U project_name/dir1/file1
cvs export: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

cvs export -D 20081126 project_name

```
$ cvs export -D 20081126 project_name
cvs export: Updating project_name
U project_name/file
cvs export: Updating project_name/dir1
U project_name/dir1/file1
cvs export: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

### cvs export -D now -d nightly project_name

```
$ cvs export -D now -d nightly project_name
cvs export: Updating nightly
U nightly/file
cvs export: Updating nightly/dir1
U nightly/dir1/file1
cvs export: Updating nightly/dir2
U nightly/dir2/file1
U nightly/dir2/file2
neo@netkiller:/tmp/cvs$
```

# 16. cvs release

```
$ ls
project_name

$ cvs release -d project_name
You have [0] altered files in this repository.
Are you sure you want to release (and delete) directory
`project_name': y

$ ls
```

# 17. branch

## 17.1. milestone

set up a release number

```
$ cvs tag release_1_0
cvs tag: Tagging .
T file
cvs tag: Tagging dir1
T dir1/file1
cvs tag: Tagging dir2
T dir2/file1
T dir2/file2
```

beginning next one milestone

```
$ cvs commit -r 2

Log message unchanged or not specified
a)bort, c)ontinue, e)dit, !)reuse this message unchanged for
remaining dirs
Action: (continue) c

CVS: ----------------------------------------------------------------
------------
CVS: Enter Log.  Lines beginning with `CVS:' are removed
automatically
CVS:
CVS: Committing in .
CVS:
CVS: Modified Files:
CVS:  Tag: 2
CVS:    file dir1/file1 dir2/file1 dir2/file2
CVS: ----------------------------------------------------------------
------------
```

```
/home/cvsroot/project_name/file,v  <--  file
new revision: 2.1; previous revision: 1.4
/home/cvsroot/project_name/dir1/file1,v  <--  dir1/file1
new revision: 2.1; previous revision: 1.1
/home/cvsroot/project_name/dir2/file1,v  <--  dir2/file1
new revision: 2.1; previous revision: 1.1
/home/cvsroot/project_name/dir2/file2,v  <--  dir2/file2
new revision: 2.1; previous revision: 1.1
```

other user

```
$ cvs up
cvs update: Updating .
P file
cvs update: Updating dir1
U dir1/file1
cvs update: Updating dir2
U dir2/file1
U dir2/file2

$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:    2.1
   Repository revision: 2.1
/home/cvsroot/project_name/file,v
   Commit Identifier:   SuZpTC1gCRrH2Qrt
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

## 17.2. patch branch

create a branch release_1_0_patch from release_1_0 by cvs admin

```
$ cvs rtag -b -r release_1_0 release_1_0_patch project_name
```

```
cvs rtag: Tagging project_name
cvs rtag: Tagging project_name/dir1
cvs rtag: Tagging project_name/dir2
```

checkout release_1_0_patch by other user

```
$ cvs checkout -r release_1_0_patch project_name
cvs checkout: Updating project_name
U project_name/file
cvs checkout: Updating project_name/dir1
U project_name/dir1/file1
cvs checkout: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

show the status, and you can see 'Sticky Tag' is 'release_1_0_patch'

```
$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:    1.4
   Repository revision: 1.4
/home/cvsroot/project_name/file,v
   Commit Identifier:   V0iuptfP43iETPrt
   Sticky Tag:          release_1_0_patch (branch: 1.4.2)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

# 18. keywords

$Author: netkiller $
$Date: 2012-02-03 17:18:44 +0800 (Fri, 03 Feb 2012) $
$Name$
$Id: section.version.cvs.xml 340 2012-02-03 09:18:44Z netkiller $
$Header$
$Log$
$Revision: 340 $

add above keywords into a file, and then commit it.

```
$ cat file.txt
$Author: netkiller $
$Date: 2012-02-03 17:18:44 +0800 (Fri, 03 Feb 2012) $
$Name:  $
$Id: section.version.cvs.xml 340 2012-02-03 09:18:44Z netkiller
$
$Header: /home/cvsroot/project_name/file.txt,v 1.2 2008-11-27
01:33:29 nchen Exp $
$Log: file.txt,v $
Revision 1.2  2008-11-27 01:33:29  nchen
added some of keywords

$Revision: 340 $
```

# 第 14 章 Miscellaneous

建模工具

- FreeMind
- ArgoUML,StarUML

常用的项目管理工具

- TRAC
- Subversion
- Bugzilla
- TWiki

# 1. 代码托管

## 1.1. sourceforge.net

### http://netkiller.users.sourceforge.net/ 页面

使用 sftp命令连接netkiller@frs.sourceforge.net，然后切换目录cd userweb/htdocs/，上传页面文件 put index.html，sourceforge.net 支持php

```
$ sftp netkiller@frs.sourceforge.net
netkiller@frs.sourceforge.net's password:
Connected to frs.sourceforge.net.
sftp> ls -l
lrwxrwxrwx    1 root     root           28 Apr 26  2012 userweb
sftp> cd userweb/htdocs/
sftp> put /tmp/index.html
Uploading /tmp/index.html to /home/user-
web/n/ne/netkiller/htdocs/index.html
```

```
/tmp/index.html                     100%   10     0.0KB/s
00:00
sftp> put /tmp/index.php
Uploading /tmp/index.php to /home/user-
web/n/ne/netkiller/htdocs/index.php
/tmp/index.php                      100%   17     0.0KB/s
00:00
sftp> pwd
Remote working directory: /home/user-web/n/ne/netkiller/htdocs
sftp> ls
index.html  index.php
sftp> exit
```

将上面netkiller改为你注册的用户名即可

帮助: https://sourceforge.net/apps/trac/sourceforge/wiki/Developer%20web

## 1.2. Google Code

## 1.3. GitHub

http://www.github.com/

### 首次操作

Global setup:

Download and install Git

```
git config --global user.name "Neo Chan"
git config --global user.email bg7nyt@gmail.com
```

Next steps:

```
mkdir neo
cd neo
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@github.com:netkiller/neo.git
git push origin master
```

Existing Git Repo?

```
cd existing_git_repo
git remote add origin git@github.com:netkiller/neo.git
git push origin master
```

## clone 已经存在的仓库

```
$ git clone
https://github.com/netkiller/netkiller.github.com.git

git config --global user.name "Your Name"
git config --global user.email you@example.com
git commit --amend --reset-author
```

# 2. GUI

## 2.1. TortoiseSVN

http://tortoisesvn.net/

## 2.2. TortoiseGit

http://code.google.com/p/tortoisegit/

# 3. Browser interface for CVS and SVN version control repositories

viewvc 10年前还有人再用，目前基本淘汰

```
# yum install viewvc
```

CGI 方式安装

```
# vim /etc/httpd/conf/httpd.conf
ScriptAlias /viewvc /usr/lib/python2.4/site-
packages/viewvc/bin/cgi/viewvc.cgi
Alias /viewvc-static/ "/usr/share/viewvc/templates/docroot/"
```