版本控制及代码托管平台综述

版本控制是协助软件开发的重要工具之一，根据维基百科中“版本控制”词条的描述，版本控制（英语：Version control）是维护工程蓝图的标准作法，能追踪工程蓝图从诞生一直到定案的过程。此外，版本控制也是一种软件工程技巧，借此能在软件开发的过程中，确保由不同人所编辑的同一程序文件都得到同步。

在程式设计中，分散式版本控制（英语：distributed revision control 或 distributed version control，又译为分布式版本控制），又称去中心化版本控制（decentralized version control），是一种版本控制的方式，它允许软件开发者可以共同参与一个软件开发专案，但是不必在相同的网络系统下工作。以分散式版本控制方法，作出的软件版本控制系统，称为分散式版本控制系统（distributed revision control system，缩写为 DRCS，或是 distributed version control system，缩写为 DVCS）。著名的分散式版本控制系统有 Monotone、Git 等。版本控制可辅助开发者精确记录自己对代码的每一次修改；若在团队开发中则辅助整个团队中的每一个人管理同一项目的每一个版本，保证整个团队共同开发的项目随时保持同步。

综上，版本控制是工程中常用做法，而在软件工程中常用分布式版本控制，在分布式版本控制系统中， Git 最早是根据 Monotone 改写，但通过 Git 进行版本控制的软件源代码托管服务平台 GitHub 已经成为了世界上最大的代码存放网站和开源社区。目前许多大型项目和著名项目开源到 GitHub 且使用 Git 做版本控制，如 Linux 内核。因此团队进行软件开发使用分布式版本控制系统是很有必要的，而当前使用 Git 做分布式版本控制系统并搭建基于 Git 的代码仓库是很先进的。

相关背景

上文论证了使用 Git 做分布式版本控制系统的必要性和先进性，在这里介绍相关技术、产品的背景，内容主要来自维基百科。

Git

git（/gɪt/）是一个分布式版本控制软件，最初由林纳斯·托瓦兹创作，于 2005 年以 GPL 发布。最初目的是为更好地管理 Linux 内核开发而设计。

git 是用于 Linux 内核开发的版本控制工具。与 CVS、Subversion 一类的集中式版本控制工具不同，它采用了分布式版本库的作法，不需要服务器端软件，就可以运作版本控制，使得源代码的发布和交流极其方便。git 的速度很快，这对于诸如

Linux 内核这样的大项目来说自然很重要。git 最为出色的是它的合并追踪（merge tracing）能力。

GitHub

GitHub 是通过 Git 进行版本控制的软件源代码托管服务平台，由 GitHub 公司（曾称 Logical Awesome）的开发者 Chris Wanstrath、PJ Hyett 和 Tom Preston-Werner 使用 Ruby on Rails 编写而成。

GitHub 同时提供付费账户和免费账户。这两种账户都可以创建公开或私有的代码仓库，但付费用户支持更多功能。根据在 2009 年的 Git 用户调查，GitHub 是最流行的 Git 访问站点。除了允许个人和组织创建和访问保管中的代码以外，它也提供了一些方便社会化共同软件开发的功能，即一般人口中的社群功能，包括允许用户追踪其他用户、组织、软件库的动态，对软件代码的改动和 bug 提出评论等。GitHub 也提供了图表功能，用于概观显示开发者们怎样在代码库上工作以及软件的开发活跃程度。

截止到 2015 年，GitHub 已经有超过两千八百万注册用户和 5700 万代码库。事实上已经成为了世界上最大的代码存放网站和开源社区。

2018 年 6 月 4 日晚上，美国科技公司微软宣布以 75 亿美元的股票收购 GitHub。

GitLab

GitLab 是由 GitLab Inc.开发，使用 MIT 许可证的基于网络的 Git 仓库管理工具，且具有 wiki 和 issue 跟踪功能。

GitLab 由乌克兰程序员 Dmitriy Zaporozhets 和 Valery Sizov 开发，它由 Ruby 写成。后来，一些部分用 Go 语言重写。截止 2018 年 5 月，该公司约有 290 名团队成员，以及 2000 多名开源贡献者。GitLab 被 IBM，Sony，Jülich Research Center，NASA，Alibaba，Invincea，O'Reilly Media，Leibniz-Rechenzentrum (LRZ)，CERN，SpaceX 等组织使用。

总结

宗上，若公司内部搭建基于版本控制的代码存放网站，可使用私有服务器搭建 GitLab 实现代码托管及权限控制，在客户端使用 Git 进行代码版本控制及代码的拉取和提交。下面介绍搭建 GitLab 所需的软硬件环境。

GitLab 环境需求

硬件要求

- CPU
- 1 core supports up to 100 users
- 2 cores is the recommended minimum number of cores and supports up to 100 users（推荐配置）
- 4 cores supports up to 500 users
- 8 cores supports up to 1,000 users
- 32 cores supports up to 5,000 users
- 内存
- 4GB RAM + 4GB swap supports up to 100 users but it will be very slow
- 8GB RAM is the recommended minimum memory size for all installations and supports up to 100 users（推荐配置）
- 16GB RAM supports up to 500 users
- 32GB RAM supports up to 1,000 users
- 128GB RAM supports up to 5,000 users
- 硬盘 >= 所有代码仓库所占空间 并为数据库预留 5-10 GB 存储空间

数据库要求

- PostgreSQL >= 9.6

操作要求

GitLab 支持以下操作系统

- Ubuntu
- Debian
- CentOS
- openSUSE
- Red Hat Enterprise Linux (please use the CentOS packages and instructions)
- Scientific Linux (please use the CentOS packages and instructions)
- Oracle Linux (please use the CentOS packages and instructions)

软件要求

- Ruby >= 2.6
- Go >= 1.12
- Git >= 2.21.x
- Node.js >= 8.10.0

总结

综上所述，GitLab 的推荐硬件配置为 2C8GB，接下来介绍 GitLab 的安装。

安装流程

本教程仅简单介绍 GitLab Omnibus package 方式在一台 安装 Ubuntu 操作系统的服务器/弹性云服务器 上一键安装的方法，更多方法请移步官网文档安装部分 https://docs.gitlab.com/ee/install/README.html。

代码托管服务器配置（GitLab）

第一步、安装并配置必要的依赖项

```
$ sudo apt-get update
$ sudo apt-get install -y curl openssh-server ca-certificates
```

第二步、添加 GitLab 软件包存储库并安装软件包

```
$ curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.deb.sh | sudo bash
```

接下来使用自己为 GitLab 准备的域名安装 GitLab

bash $ sudo EXTERNAL_URL="https://gitlab.example.com" apt-get install gitlab-ee 注：若使用 https:// 访问需要验证域名，您也可以使用自己的证书或是使用 http://。

也可不指定域名，若不指定安装完成后直接使用服务器 ip 地址 访问 GitLab。

```
$ sudo apt-get install gitlab-ee
```

第三步、使用域名/IP 访问并进行相关配置

第一次进入会要求输入密码，此时设置的密码为 root 账户的密码，此账户为本 GitLab 的最高权限管理员，请妥善管理账号密码。

注册后使用 root 登陆，即可进行添加用户、添加项目、仓库及各种权限管理操作。

本教程到此结束，更多详细内容请移步 GitLab 官方文档：https://docs.gitlab.com/。也可来到我的博客访问 GitLab 部分。

客户端配置（Git）

在 Linux 上安装

如果你想在 Linux 上用二进制安装程序来安装 Git，可以使用发行版包含的基础软件包管理工具来安装。 如果以 Fedora 上为例，你可以使用 yum：

```
$ sudo yum install git
```

如果你在基于 Debian 的发行版上，请尝试用 apt-get：

```
$ sudo apt-get install git
```

要了解更多选择，Git 官方网站上有在各种 Unix 风格的系统上安装步骤，网址为 http://git-scm.com/download/linux。

在 Mac 上安装

在 Mac 上安装 Git 有多种方式。 最简单的方法是安装 Xcode Command Line Tools。 Mavericks （10.9） 或更高版本的系统中，在 Terminal 里尝试首次运行 git 命令即可。 如果没有安装过命令行开发者工具，将会提示你安装。

如果你想安装更新的版本，可以使用二进制安装程序。 官方维护的 OSX Git 安装程序可以在 Git 官方网站下载，网址为 http://git-scm.com/download/mac。

你也可以将它作为 GitHub for Mac 的一部分来安装。 它们的图形化 Git 工具有一个安装命令行工具的选项。 你可以从 GitHub for Mac 网站下载该工具，网址为 http://mac.github.com。

在 Windows 上安装

在 Windows 上安装 Git 也有几种安装方法。 官方版本可以在 Git 官方网站下载。 打开 http://git-scm.com/download/win，下载会自动开始。 要注意这是一个名为 Git for Windows 的项目（也叫做 msysGit），和 Git 是分别独立的项目；更多信息请访问 http://msysgit.github.io/。

另一个简单的方法是安装 GitHub for Windows。 该安装程序包含图形化和命令行版本的 Git。 它也能支持 Powershell，提供了稳定的凭证缓存和健全的换行设置。 稍后我们会对这方面有更多了解，现在只要一句话就够了，这些都是你所需要的。 你可以在 GitHub for Windows 网站下载，网址为 http://windows.github.com。

注：内容来自 git 官网教程：https://git-scm.com/book/zh/v2/%E8%B5%B7%E6%AD%A5-%E5%AE%89%E8%A3%85-Git

总结

至此，完整交代了 GitLab 及 Git 的安装配置流程，其中 Git 是版本控制工具，用于拉取、提交及对代码的版本控制；而 GitLab 则是远程的代码托管平台，管理员可对代码仓库进行权限控制保证代码安全。用户在需要的使用使用 Git 从 GitLab 拉取代码仓库的代码，并将编写好的代码通过 Git 提交到远程的 GitLab 代码仓库。使用这一组合可以完美实现代码的管理、版本控制、团队协作及代码仓库的权限管理。

参考文献/推荐阅读

- 版本控制
- WikiPedia/分布式版本控制
- WikiPedia/GitHub
- WikiPedia/GitLab
- Install GitLab using the Omnibus GitLab package
- git/1.5 起步 - 安装 Git
- gitlab/Administrator Docs
- git
- monotone
- Git 安装配置