

# Linux高级运维

**NSD OPERATION**

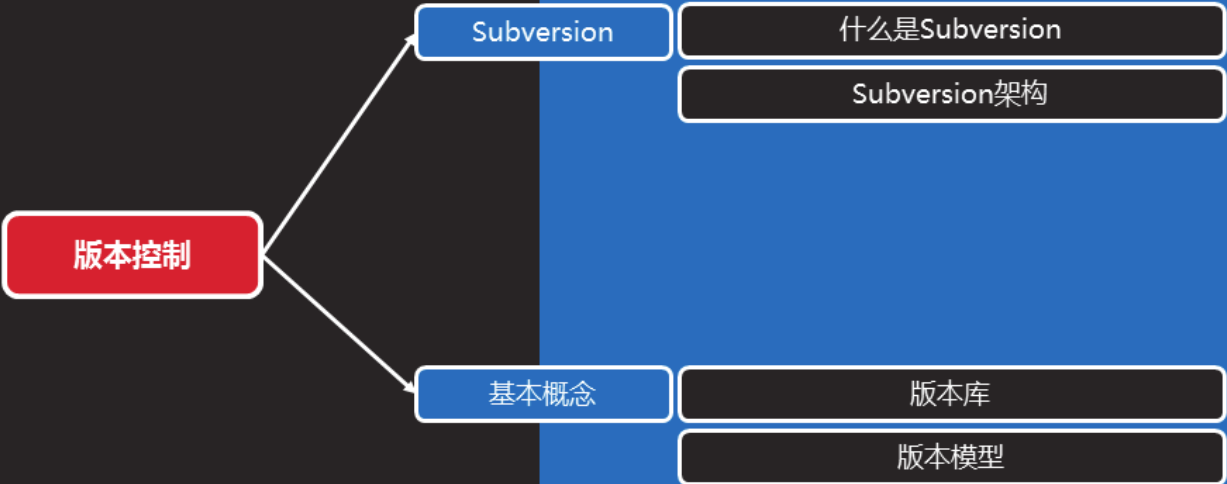
**DAY06**

# 内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	版本控制
	10:30 ~ 11:20	SVN基础
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	实战案例
	15:00 ~ 15:50	RPM打包
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑



## 版本控制



# Subversion

## 什么是Subversion

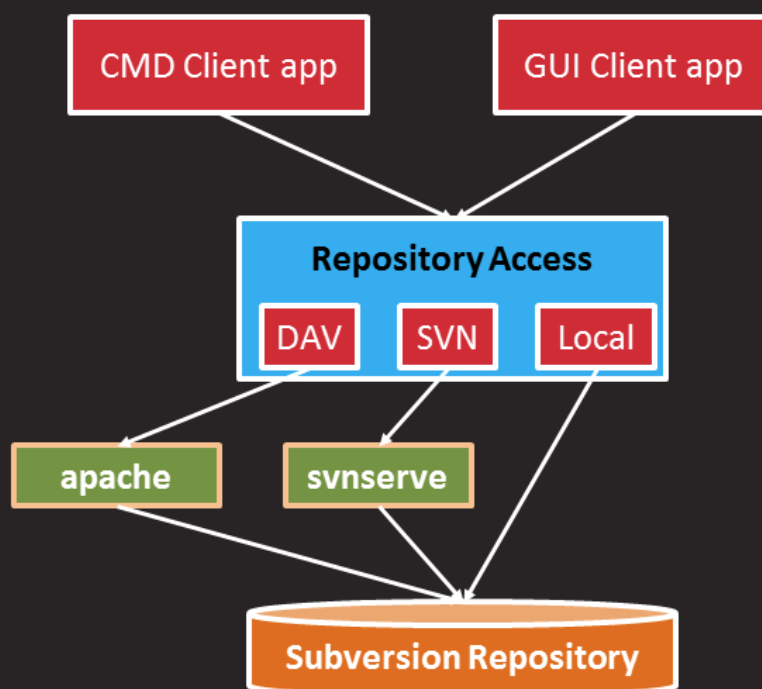
- Subversion 是一个自由/开源的**版本控制系统**
  - 在Subversion管理下，文件和目录可以超越时空
  - Subversion**允许你数据恢复到早期版本**
  - 或者是检查数据修改的历史
  - 许多人将版本控制系统当作一种神奇的“时间机器”
  - 允许你**和别人协作文档**并跟踪所做的修改



# Subversion架构

## 知识讲解

- 客户端
  - 命令行
  - 图形
- 通信方式
  - 本地访问
  - SVN服务器
  - Web服务

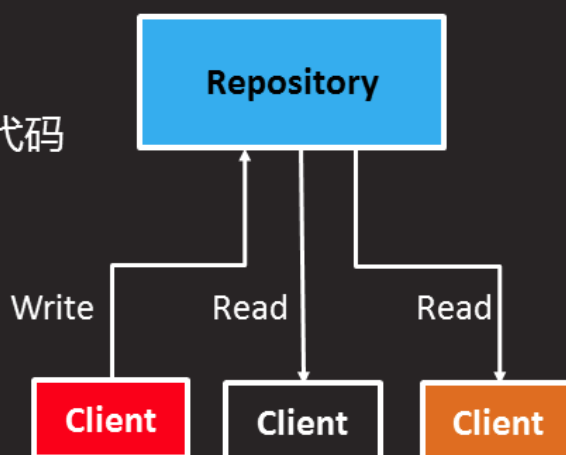


## 基本概念

# 版本库

知识讲解

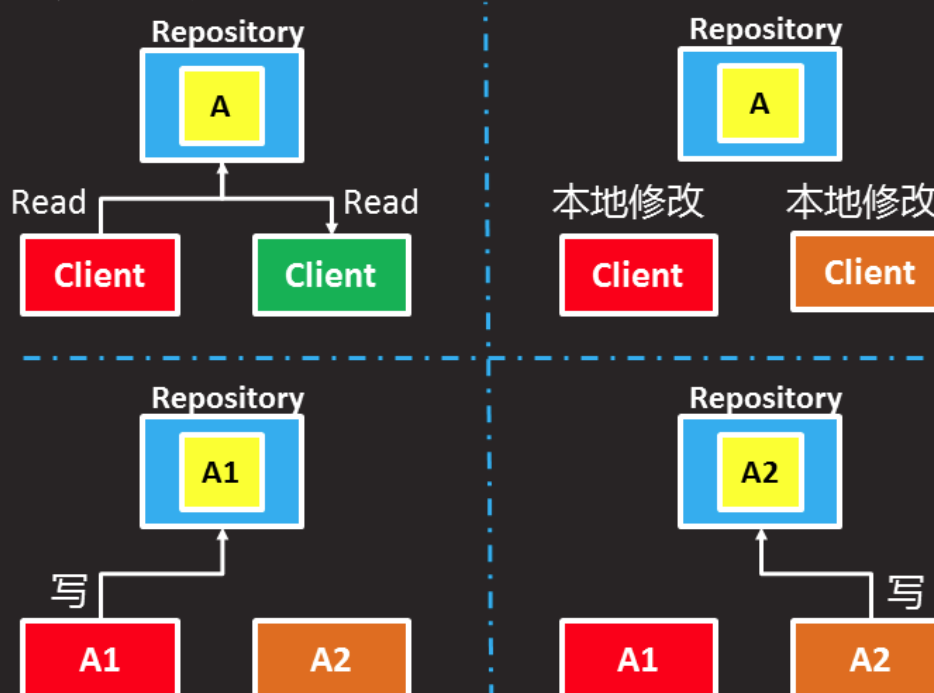
- 典型的客户/服务器系统
  - 版本库是版本控制的核心
  - 任意数量客户端
  - 客户端通过写数据库分享代码
- Subversion特点
  - 记录每一次改变



# 版本模型

知识讲解

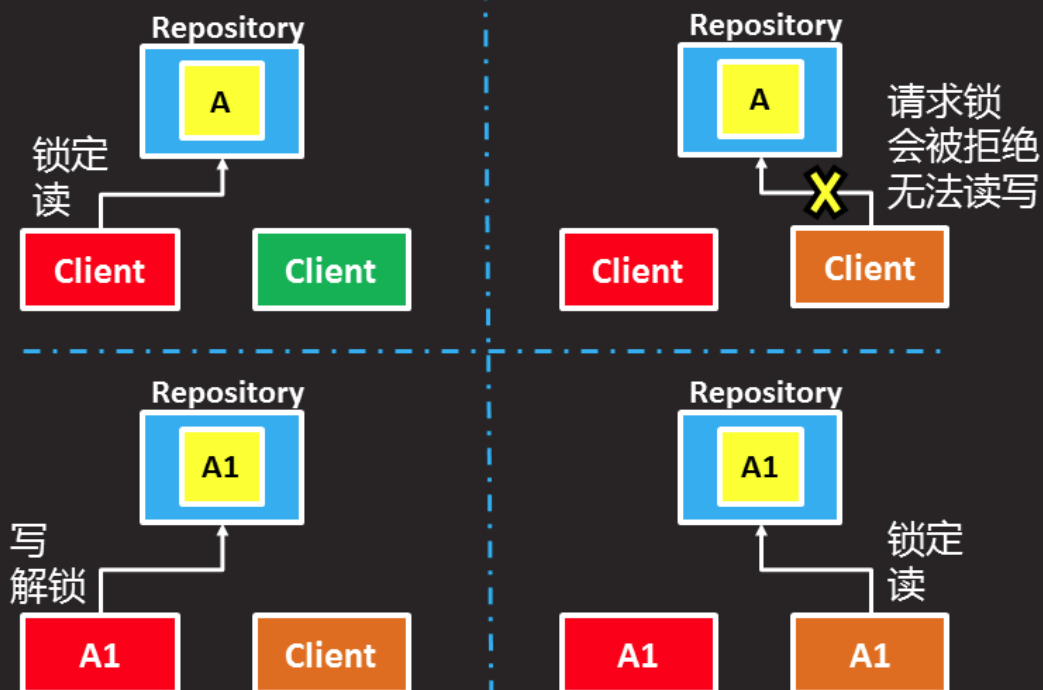
- 需要避免的问题



## 版本模型（续1）

- 锁定-修改-解锁的方案

知识讲解



## 版本模型（续2）

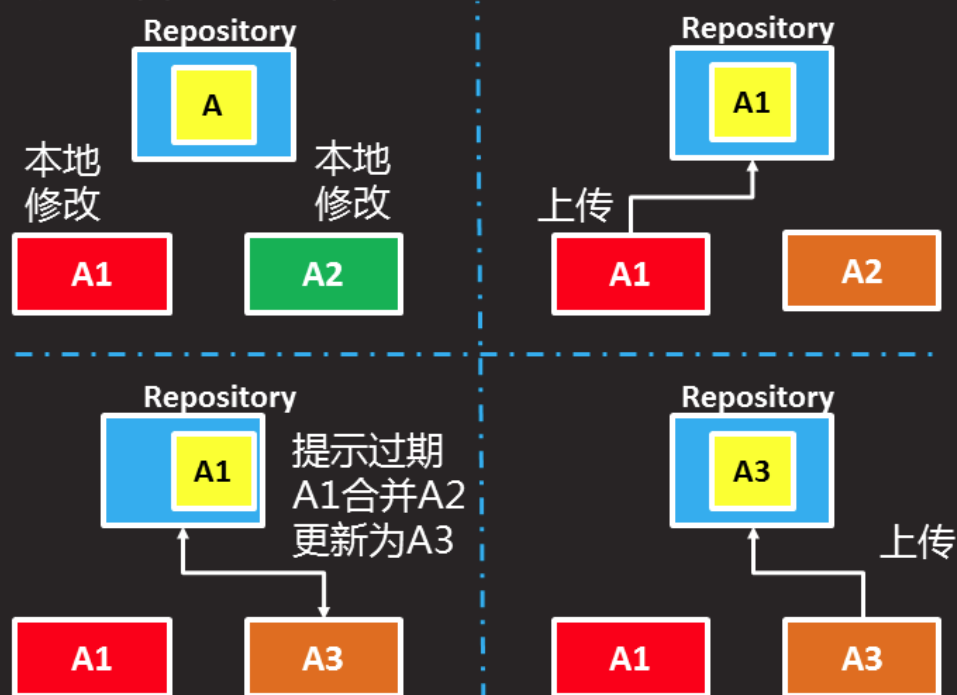
- 锁定-修改-解锁的问题
  - 锁定可能导致管理问题
    - A锁定文件后忘记解锁等问题
  - 锁定导致不必要的串行开发
    - A想修改一个文件的开始，B想修改一个文件的结尾
    - 如果能进行正确的合并，则可以更轻松的工作，没必要轮流工作

知识讲解



## 版本模型（续3）

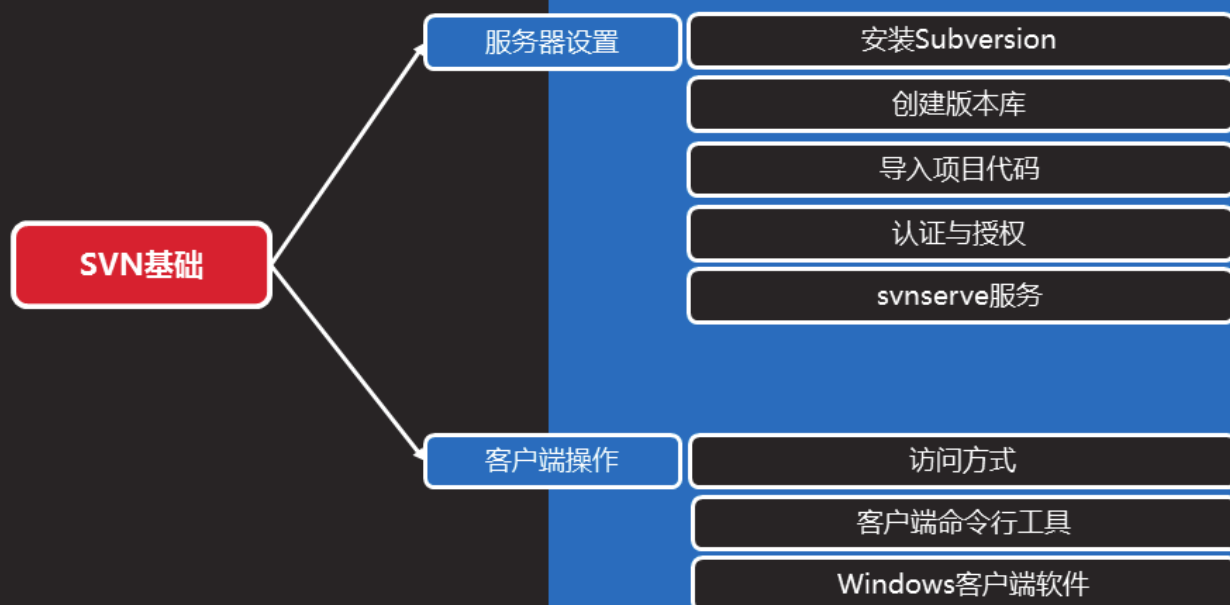
- 拷贝-修改-合并的方案



## 版本模型（续4）

- Subversion控制系统使用**拷贝-修改-合并模型**
  - 每个客户连接版本库，并建立个人工作副本
  - 用户并行工作，修改自己的副本
  - 最终，合并版本！
  - 个别冲突问题，需要人为手动解决
- 流程
  - Harry拷贝副本，Sally拷贝副本
  - Harry修改A1,Sally修改A2
  - Harry上传A1后，Sally上传A2，会提示Sally的文件已过期
  - Sally更新文件（合并）后上传新的A3

# SVN基础



## 服务器设置



# 安装Subversion

- 安装Subversion

知识讲解

```
[root@svr5 ~]# yum -y install subversion
[root@svr5 ~]# rpm -q subversion
subversion-1.7.14-10.el7.x86_64
```



# 创建版本库

- 利用svnadmin命令可创建服务器版本库
  - subversion版本库管理工具，帮助：svnadmin help
  - 用法：svnadmin 命令 /版本库路径 [选项]
  - 命令：create 创建一个新的版本库

知识讲解

```
[root@svr5 ~]# mkdir /var/svn
[root@svr5 ~]# svnadmin create /var/svn/project1
[root@svr5 ~]# svnadmin create /var/svn/project2
[root@svr5 ~]# svnadmin create /var/svn/devel
```



## 导入项目代码

知识讲解

- 使用svn命令将项目代码导入到版本库中
  - import指令执行导入操作
  - -m选项设置注释性的字符串

```
[root@svr5 ~]# cd /usr/lib/systemd/system  
[root@svr5 ~]# svn import ./ file:///var/svn/project1 -m "Init"
```



## 认证与授权

知识讲解

- 使用SVN内置的认证机制可以有效地增强客户端访问版本库的安全性
  - 当客户端访问版本库服务器时，服务器会根据版本库目录下的conf/svnserve.conf文件中定义的认证与授权策略实现权限的控制



## 认证与授权（续1）

- 使用SVN内置的认证机制可以有效地增强客户端访问版本库的安全性

知识讲解

```
[root@svr5 ~]# cat /var/svn/web_project/conf/svnserve.conf
[general]
anon-access = none
//设置拒绝匿名账户访问，此处可以设置为none、read、write
auth-access = write
//经过认证的账户权限为可写权限
password-db = passwd
//账户名称与密码的存放文件名，该文件在conf目录下
authz-db = authz
//基于路径的访问控制文件名（可以对文件或目录设置权限）
```



## 认证与授权（续2）

- 在passwd文件中需要设置账户信息
- 在authz文件中需要设置访问控制权限

知识讲解

```
[root@svr5 ~]# cat /var/svn/web_project/conf/passwd
[users]
harry = harryssecret //用户名为harry，密码为harryssecret
sally = sallyssecret //用户名为sally，密码为sallyssecret
[root@svr5 ~]# cat /var/svn/web_project/conf/authz
[groups]
admins = harry,sally //定义组，组成员有harry与sally
[/] //根路径设置权限，也可以设置其他路径
@admin = rw //admins组中的用户可读、可写权限
* = r //其他所有的人只读
//权限列别：只读('r')、读写('rw')、无权限('')
```



# svnserve服务

知识讲解

- svnserve命令即可启动SVN服务进程
  - -d //以守护进程方式运行svnserve
  - --listen-port=port //监听的端口，默认端口号为3690
  - -r root //设置一个虚拟路径，默认客户端要指定绝对路径访问库

[root@svr5 ~]# svnserve -d -r /var/svn/project1 //共享单个仓库  
或者  
[root@svr5 ~]# svnserve -d //共享所有仓库



## 客户端操作

## 访问方式

知识讲解

- 本地磁盘
- SVN
- Web

URL 格式	含义
file:///	直接访问本地磁盘上的版本库（客户端与服务器端在一台机器上）
http://	配置 Apache 的 WebDAV 协议，通过网页访问版本库
https://	与 http://相似，但使用了 SSL 进行数据加密
svn://	通过 svnserve 定义的协议访问版本库
svn+ssh://	与 svn://相似，但使用了 SSH 封装加密数据



## 客户端命令行工具

知识讲解

- svn 命令 [选项]
  - --password 密码
  - --username 用户名
  - --revision(-r) 指定要检查的特定版本



## 客户端命令行工具（续1）

知识讲解

- checkout命令(初始化检出)
  - checkout URL [PATH]
  - 从服务器版本库中复制一份副本至本地
  - URL定位版本库
  - PATH为本地工作副本路径

```
[root@svr5 ~]# cd /tmp/  
[root@svr5 ~]# svn co svn://10.47.214.131/ code
```



## 客户端命令行工具（续2）

知识讲解

- commit命令（提交修改）
  - 在本地修改本地副本中的代码后，commit可以提交该修改，原子事务提交
- update命令
  - 将服务器上其他人的修改的代码更新到本地

```
[root@svr5 mine]# svn ci -m "node"  
[root@svr5 mine]# svn update
```



## 客户端命令行工具（续3）

知识讲解

- 查看版本库信息
  - info命令  
查看版本仓库信息
  - log命令  
查看版本修改历史

```
[root@svr5 ~]# svn info svn://192.168.2.100/  
[root@svr5 ~]# svn log svn://192.168.2.100/
```



## 客户端命令行工具（续4）

知识讲解

- add命令（本地版本库添加新文件）
  - 在本地版本库副本，添加新的文件
  - 注意，add不会自动提交版本库服务器
  - 需要使用commit命令提交服务器

```
[root@svr5 ~]# echo "echo test" > test.sh  
[root@svr5 ~]# svn add test.sh  
[root@svr5 ~]# svn list svn://192.168.2.100/  
[root@svr5 ~]# svn ci -m "add test file"  
[root@svr5 ~]# svn list svn://192.168.2.100/
```

//服务器无test.txt文件

//服务器有test.txt文件



## 客户端命令行工具（续5）

知识讲解

- del、rm命令（本地版本库删除文件）
  - 删除本地版本库副本文件
  - 注意，del不会自动提交版本库服务器
  - 需要使用commit命令提交服务器

```
[root@svr5 ~]# svn rm test.sh
```

```
[root@svr5 ~]# svn ci -m "delete test.sh"
```



## 客户端命令行工具（续6）

知识讲解

- mkdir命令（本地版本库创建目录）
  - 在本地版本库副本，添加新目录

```
[root@svr5 ~]# svn mkdir test
```

```
[root@svr5 ~]# svn add abc.txt
```

```
[root@svr5 ~]# svn ci -m "mkdir and add file"
```





## 客户端命令行工具（续7）

知识讲解

- diff命令（数据对比）
  - 对比本地副本与服务器数据

```
[root@svr5 ~]# echo "add test file" > killall
[root@svr5 ~]# svn diff killall
[root@svr5 ~]# svn diff
```

//对比单个文件  
//对比所有文件

- cat命令
  - 查看版本库数据的内容
  - 本地副本可以直接cat查看

```
[root@svr5 ~]# svn cat killall
```



## 客户端命令行工具（续8）

知识讲解

- 版本回滚
  - revert命令  
本地副本修改后，但未commit提交修改时回滚数据
  - merge命令  
本地副本修改commit提交后，使用该命令回滚

```
[root@svr5 ~]# sed -i '1d' killall
[root@svr5 ~]# head killall
[root@svr5 ~]# svn revert killall
[root@svr5 ~]# head killall
```

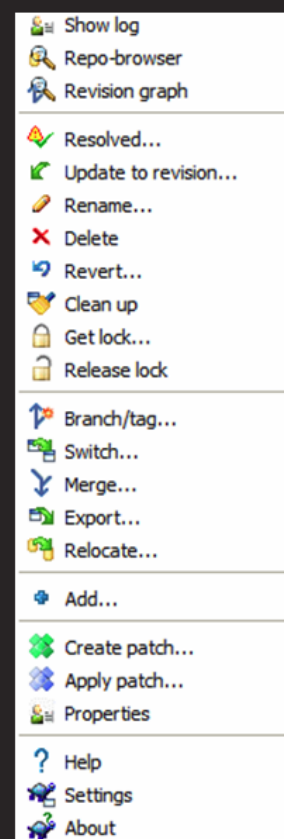
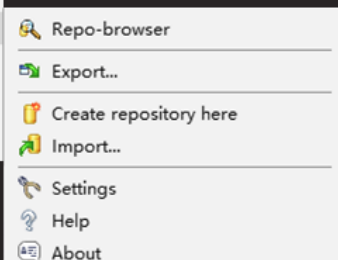
```
[root@svr5 ~]# sed -i '1d' netfs
[root@svr5 ~]# svn ci -m "modify netf"
[root@svr5 ~]# svn merge -r 10:5 netfs
```

//将netfs从10版本还原回5版本



# Windows客户端软件

知识讲解

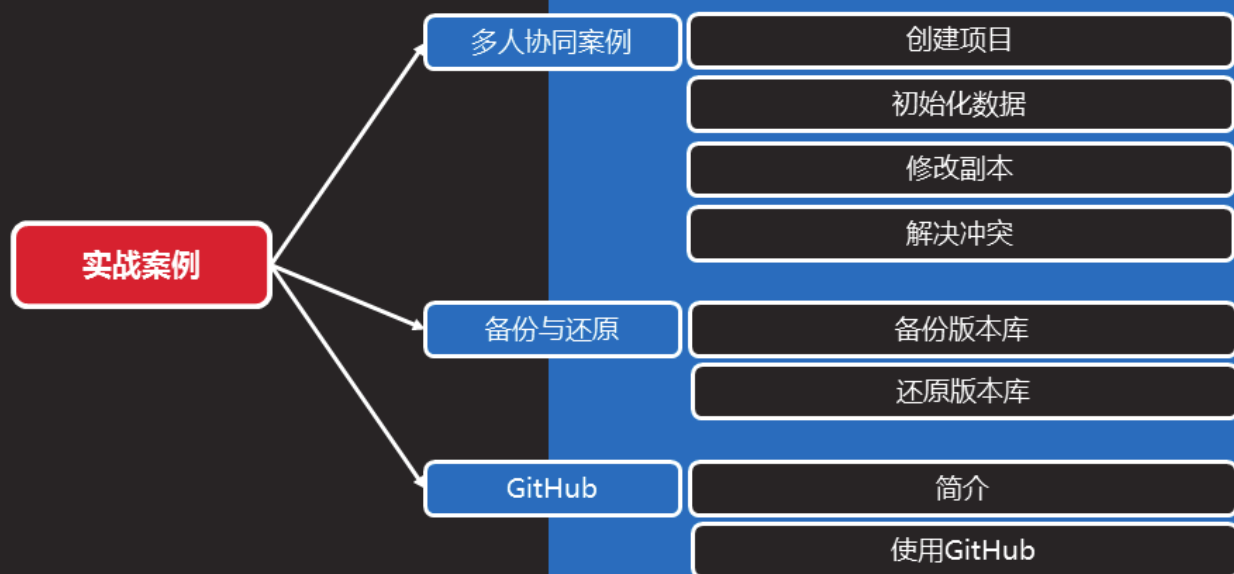


## 案例1：Subversion基本操作

- 安装subversion软件
  - 启动svnserver服务
- 客户端使用svn连接服务器进行简单操作
  - 对版本库中的文件进行增、删、改、查等操作

课堂练习

## 实战案例



## 多人协同案例

## 创建项目

知识讲解

```
[root@svr5 ~]# yum -y install subversion
[root@svr5 ~]# mkdir /var/svn
[root@svr5 ~]# svnadmin create /var/svn/project
[root@svr5 ~]# cd /var/svn/project/conf
[root@svr5 ~]# cat svnserve.conf
[general]
anon-access = none
auth-access = write
password-db = passwd
authz-db = authz
[root@svr5 ~]# cat passwd
[users]
harry = pass
tom = pass
[root@svr5 ~]# vim authz
[/]
harry = rw
tom = rw
```



## 初始化数据

知识讲解

- 初始化服务器版本库并启动SVN服务

```
[root@svr5 ~]# cd /etc/rc.d/init.d/
[root@svr5 ~]# svn import file:///var/svn/project/ -m "Init Data"
[root@svr5 ~]# service svnserve start
```

- 开发者初始化本地副本（user1和user2两个用户）

```
[user1@svr5 ~]# cd /var/tmp
[user1@svr5 ~]# svn --username harry --password pass co
svn://127.0.0.1/var/svn/project harry
[user2@svr5 ~]# cd /var/tmp
[user2@svr5 ~]# svn --username tom --password pass co
svn://127.0.0.1/var/svn/project tom
```



## 修改副本

知识讲解

- Harry修改自己的副本（不同的文件）

```
[user1@svr5 ~]# cd /var/tmp/harry
[user1@svr5 ~]# sed -i '1a####test####' sshd
[user1@svr5 ~]# svn ci -m "sshd was modified"
```

- Tom修改自己的副本（不同的文件）

```
[user2@svr5 ~]# cd /var/tmp/tom
[user2@svr5 ~]# sed -i '1a####test####' atd
[user2@svr5 ~]# svn ci -m "atd was modified"
```



## 修改副本（续1）

知识讲解

- Harry修改自己的副本（相同的文件）

```
[user1@svr5 ~]# cd /var/tmp/harry
[user1@svr5 ~]# sed -i '3a####test####' svnserve
[user1@svr5 ~]# svn ci -m "svnserve add one line test for harry"
```

- Tom修改自己的副本（相同的文件）

```
[user2@svr5 ~]# cd /var/tmp/tom
[user2@svr5 ~]# sed -i '5a####test2####' svnserve
[user2@svr5 ~]# svn ci -m "svnserve add one line test for tom"
Sending      svnserve
Transmitting file data.svn: Commit failed (details follow):
svn: File '/svnserve' is out of date
[user2@svr5 ~]# svn update
[user2@svr5 ~]# svn ci -m "svnserve add one line test for tom"
```



## 修改副本（续2）

知识讲解

- Harry修改自己的副本（相同的行）

```
[user1@svr5 ~]# cd /var/tmp/harry
[user1@svr5 ~]# sed -i '1c#!/bin/sh' halt
[user1@svr5 ~]# svn ci -m "harry changed the compiler for halt"
```

- Tom修改自己的副本（相同的行）

```
[user2@svr5 ~]# cd /var/tmp/tom
[user2@svr5 ~]# sed -i '1c#!/bin/ksh' halt
[user2@svr5 ~]# svn ci -m "harry changed the compiler for halt"
[user2@svr5 ~]# svn update
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```



## 解决冲突

- update时会提示冲突，需要手动处理

```
[user2@svr5 ~]# svn update
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
//df          对比不同
//edit        直接修改文件,修改后选择r
//mine-conflict 冲突以本地为准
//theirs-conflict 冲突以服务器为准
//postpone    标记冲突，稍后解决
```

知识讲解



## 解决冲突（续1）

- 选择postpone标记后，本地副本会多几个文件

```
[user2@svr5 ~]# ls
halt                //本地副本文件
halt.mine           //我的修改副本
halt.r5             //第五个版本
halt.r6             //第六个版本
```

知识讲解

- 保留需要的文件覆盖掉halt后，rm删除其他即可

```
[user2@svr5 ~]# cd /var/tmp/tom
[user2@svr5 ~]# rm -rf halt.r5; rm -rf halt.r6
[user2@svr5 ~]# mv halt.mine halt; chmod +x halt.mine
[user2@svr5 ~]# svn ci -m "harry changed the compiler for halt"
```



## 备份与还原

## 备份版本库

- 使用dump指令备份

```
[root@svr5 ~]# svnadmin dump /var/svn/project/ > project.bak
```

知识讲解



## 还原版本库

- 使用load指令还原数据

```
[root@svr5 ~]# svnadmin create /var/svn/project2  
[root@svr5 ~]# svnadmin load /var/svn/project2 < project.bak
```

知识讲解





# GitHub



## 简介

- GitHub是一个面向开源及私有软件项目的托管平台，因为只支持git 作为唯一的版本库格式进行托管，故名GitHub。
- GitHub于2008年4月10日正式上线

# 使用GitHub

- 注册、上传代码

知识讲解

Username

Email

Password

Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)



## 使用GitHub (续1)

- 注册、上传代码

知识讲解

Owner: [redhatedu](#) / Repository name:

Great repository names are short and memorable. Need inspiration?

Description (optional)

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

☒ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip

Add .gitignore: [None](#) Add a license: [None](#) ⓘ

[Create repository](#)

1 commit 2 branches 0 releases 1 contributor

Branch: master New pull request

[Create new file](#) [Upload files](#) [Find file](#)

[redhatedu](#) Initial commit Latest commit

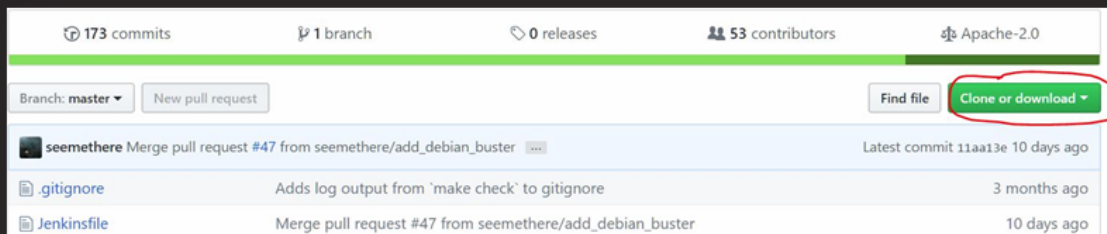
[README.md](#) Initial commit



## 使用GitHub (续2)

- 下载代码

知识讲解

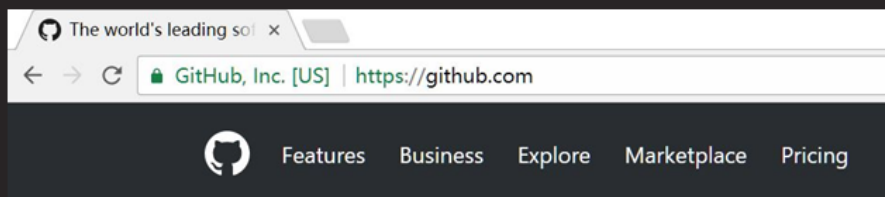


```
[root@test ~]# git clone https://github.com/docker/docker-install
Cloning into 'docker-install'...
remote: Counting objects: 493, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 493 (delta 13), reused 13 (delta 6), pack-reused 467
Receiving objects: 100% (493/493), 184.10 KiB | 242.00 KiB/s, done.
Resolving deltas: 100% (207/207), done.
```



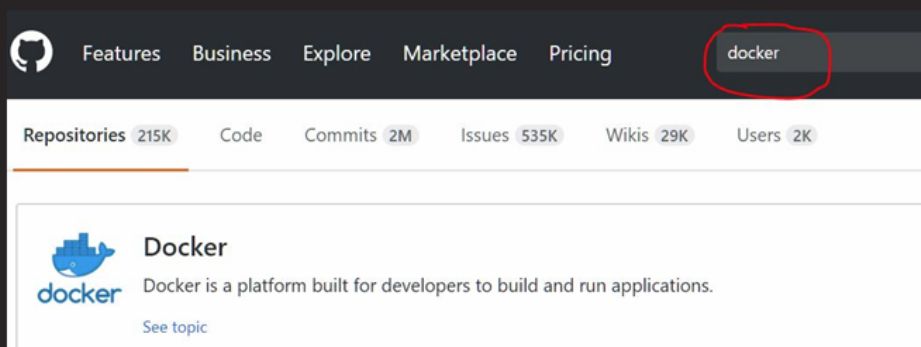
## 使用GitHub (续3)

- 登录网址 ( <https://github.com/> )



知识讲解

- 搜索项目



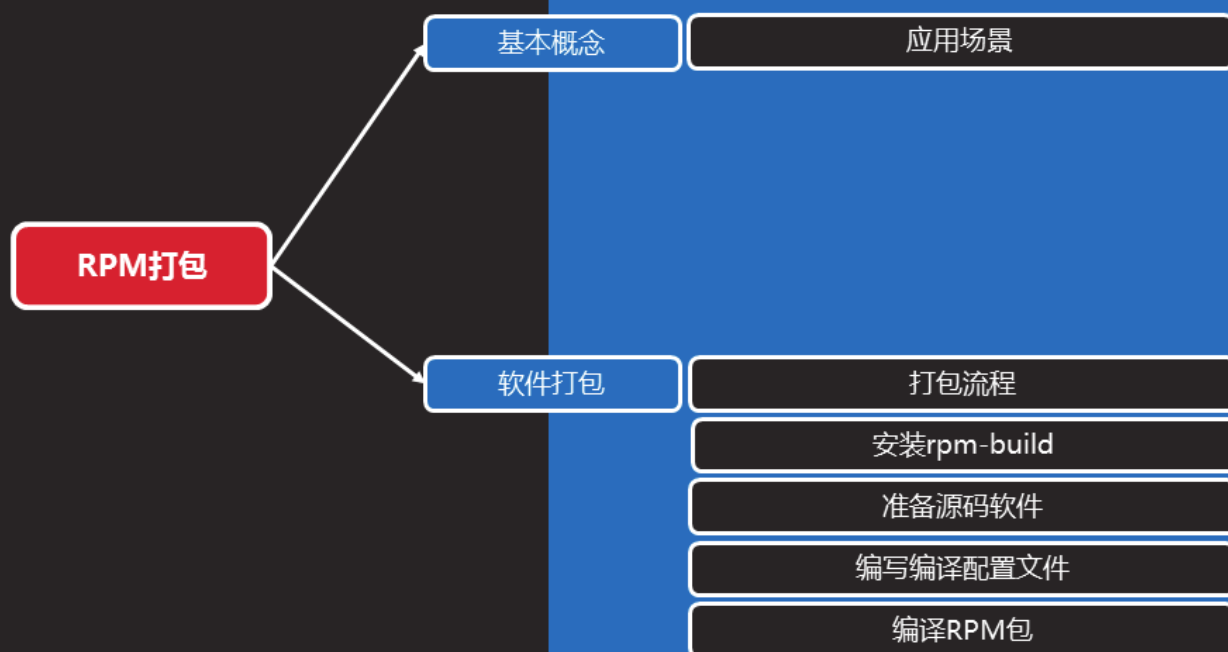
## 案例2：使用Subversion协同工作

课堂练习

- 版本库支持多个账户同时协作编辑文件
  - 测试演示多人协作编辑的具体操作
  - 手动解决版本冲突问题
- 备份版本库数据



### RPM打包



# 基本概念



## 应用场景

- 官方未提供RPM包
- 官方RPM无法自定义
- 大量源码包，希望提供统一的软件管理机制

知识讲解



# 软件打包



## 打包流程

- 准备源码软件
- 安装rpm-build
- 编写编译配置文件
- 编译RPM包

知识讲解



# 安装rpm-build

- 编译打包RPM的命令工具

```
[root@svr5 ~]# yum -y install rpm-build  
[root@svr5 ~]# rpmbuild -ba test.spec  
[root@svr5 ~]# ls
```

//生成rpmbuild目录

知识讲解



# 准备源码软件

- 将源码包复制到rpmbuild子目录

```
[root@svr5 ~]# cp nginx-1.12.2.tar.gz rpmbuild/SOURCES/
```

知识讲解



# 编写编译配置文件

- 新建SPEC文件

知识讲解

```
[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec
```

```
Name:hello           //软件名称
Version:             //软件版本
Release: 1           //RPM版本
Summary:             //描述
Group:               //软件组
License:             //协议
URL:                 //网址
Source0:             //源码文件
BuildRoot:           %(mktemp -ud %[_tmppath]/%{name}-%{version}-%{release})
                      //临时编译目录
```



# 编写编译配置文件（续1）

- 新建SPEC文件

知识讲解

```
[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec
```

```
... ..
BuildRequires:       //编译时依赖包
Requires:            //安装时依赖包
%description         //详细描述
%prep                //安装前准备，解压
%setup -q            //系统使用setup自动解压，安静模式
%build               //编译需要执行的命令
make
%configure           //配置时需要执行的命令
make %{?_smp_mflags}
%install              //安装时需要执行的指令
rm -rf %{buildroot}
make install DESTDIR=%{buildroot}
```





## 编写编译配置文件（续2）

- 新建SPEC文件

```
[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec
```

知识讲解

```
... ..  
%clean                                //清理时需要执行的指令  
rm -rf %{buildroot}  
%files                                //定义打包文件列表  
%defattr(-,root,root,-)  
%doc  
%changelog                            //软件修改历史
```



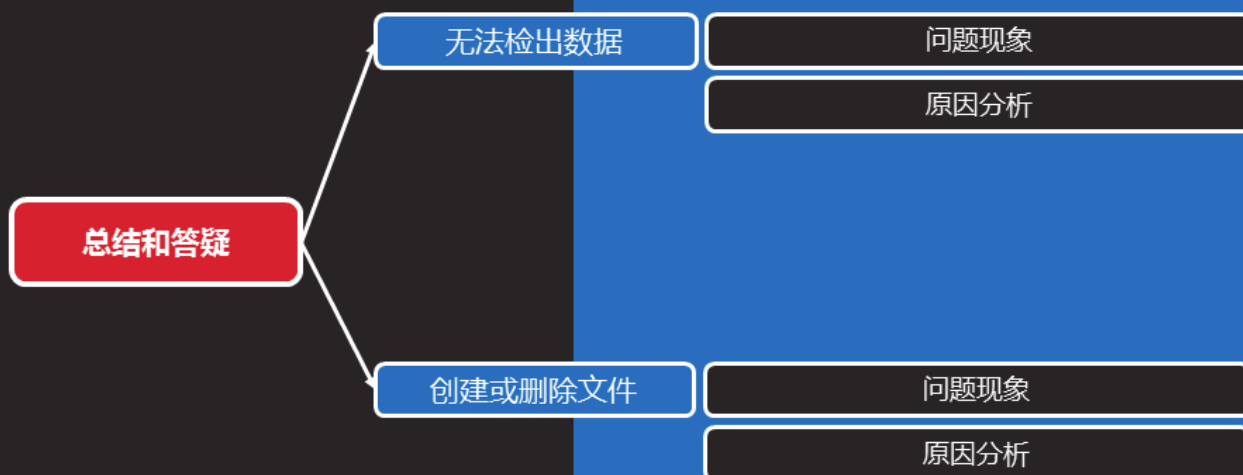
## 案例3：制作nginx的RPM包

课堂练习

- 使用nginx源码制作对应的RPM包
  - nginx源码版本为1.12.2
  - 编写spec配置文件
  - 使用rpmbuild工具进行rpm打包
- 使用RPM包安装nginx软件



### 总结和答疑



# 无法检出数据

## 问题现象

- 故障错误信息

```
[root@svr5 ~]# svn --username harry --password pass \  
> checkout svn://127.0.0.1/harry  
No repository found in 'svn://127.0.0.1'
```

```
[root@svr5 ~]# svn --username harry --password pass \  
> checkout svn://127.0.0.1/harry  
No access allowed to this repository
```

```
[root@svr5 ~]# svn --username harry --password pass \  
> checkout svn://127.0.0.1/harry  
Authorization failed
```

## 原因分析

知识讲解

- 分析故障信息
  - No repository found in 'svn://127.0.0.1'
  - No access allowed to this repository
  - Authorization failed
- 分析故障原因
  - 无法找到代码库，启动服务器时没有-r，客户端连接需要指定绝对路径
  - 账户无权限访问代码库
  - authz文件没有正确定义账户对项目的权限
  - svn配置文件顶头不可以有空格



## 创建或删除文件

# 问题现象

- 故障错误信息

```
[root@svr5 ~]# rm httpd  
[root@svr5 ~]# mkdir test; touch test.txt  
[root@svr5 ~]# svn commit -m "aa"
```

知识讲解



