Automation Factory

Version Control Systems

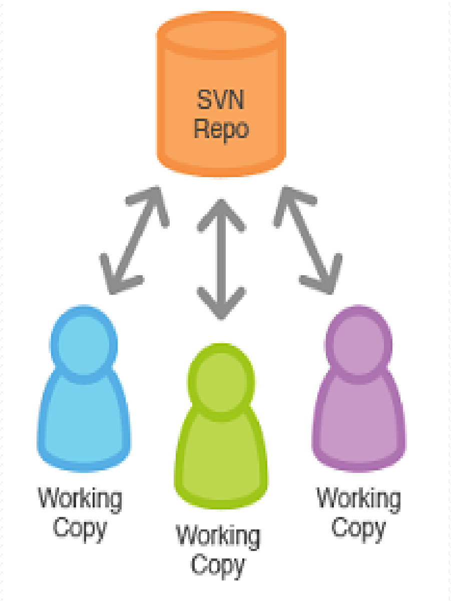
By, Ganesh Palnitkar

Agenda

- Importance of Version Control System
- Types of Version Control Systems
  - Distributed VCS
  - Centralized VCS
- Importance of Branching and merging
- Branching and Merging strategies
- Learning GIT – Common
- Learning Subversion (SVN)

About Version Control Systems

- **Version control software** is an essential part of the every-day of the modern software team's professional practices.
  - Version control software keeps track of every modification to the code in a special kind of database.
  - Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.
  - Helps teams to solve problems like, tracking every individual change by each contributor and helping prevent concurrent work from conflict.
  - Conflicts if any should be solved using help of tool or manually.



Benefits:

- No need to keep multiple backups.
- Allows multiple people to work on same file / project.
- A complete long-term change history of every file.
- Branching and merging – maintain code as per projects / release / functionality etc.
- Traceability - ability to trace each change made to the software and connect it to project management and bug tracking software.
- Easily switch / work on earlier file versions.

Types of Version Control System

Centralized VCS:

Workflow:

- Pull down any changes other people have made from the central server.
- Make your changes, and make sure they work properly.
- Commit your changes to the central server, so other programmers can see them.

Benefits:

- Project history and database is maintained on centralized server, beneficial incase of large project size and history.

Disadvantages:

- Internet / network connection is must.
- Frequent (daily) database sync is necessary.
- Pull and push to centralized server can be time taking.

e.g. : SVN, TFS, Perforce

Distributed VCS

Workflow:

- Clone entire repository on local machine.
- Maintain all code changes on local machine.
- Incase of sharing the code with others, merge the code on central GIT repo.

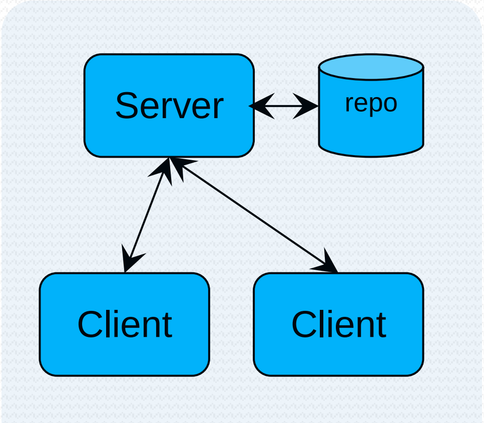
Benefits:

- Extremely fast, as database resides on local drive.
- Committing to a changeset can be done locally. Group of changesets then can be pushed to remote repository.
- No need of internet connection.

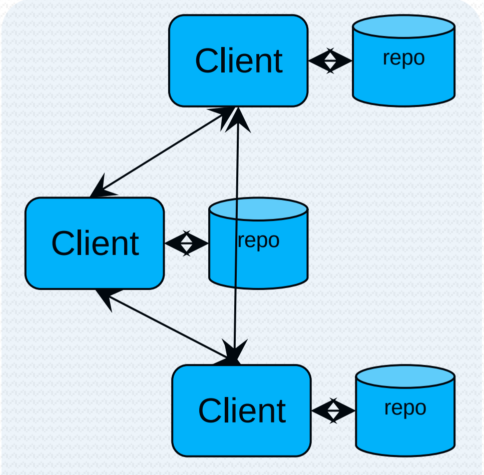
Disadvantages:

- The only major one can be because of large project history / size. Initial cloning and maintaining huge database can eat up space on local machine.
- Database backup should be done locally.

e.g. GIT, Mercurial

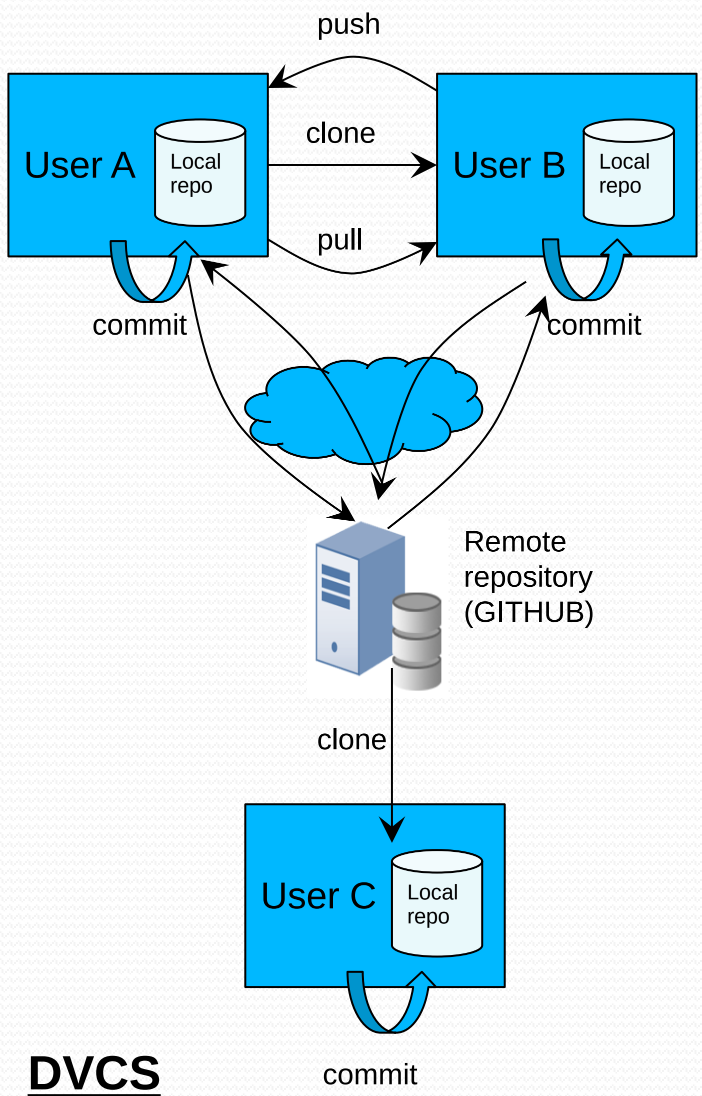


Centralized VCS

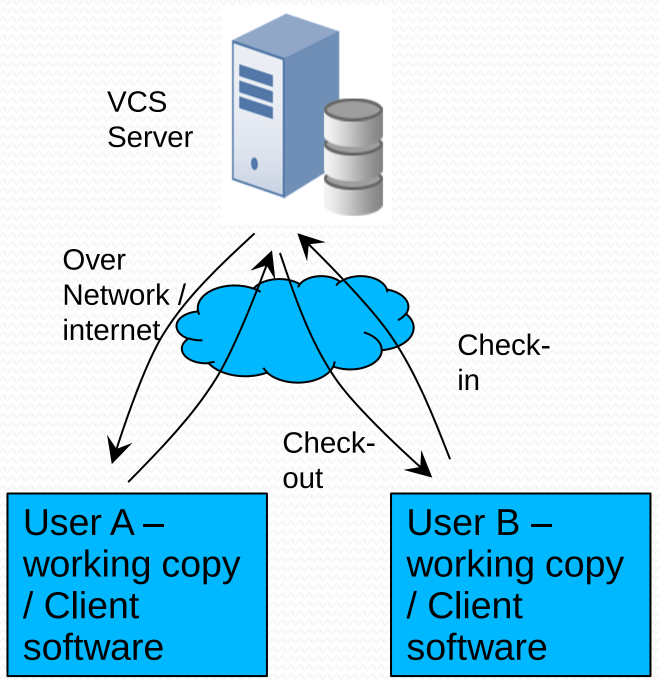


Distributed VCS

Distributed VCS & Centralized VCS



DVCS


VCS Server
Over Network / internet
Check-in
Check-out
User A – working copy / Client software
User B – working copy / Client software

CVCS

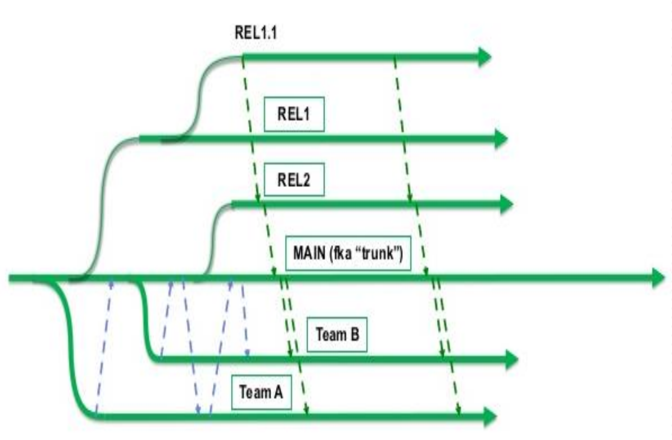
Branching and Merging strategies

- ➢ Brach by release, “staircase model”.
- ➢ Branch by feature. Each distinct branch.
- ➢ Hotfix branch.

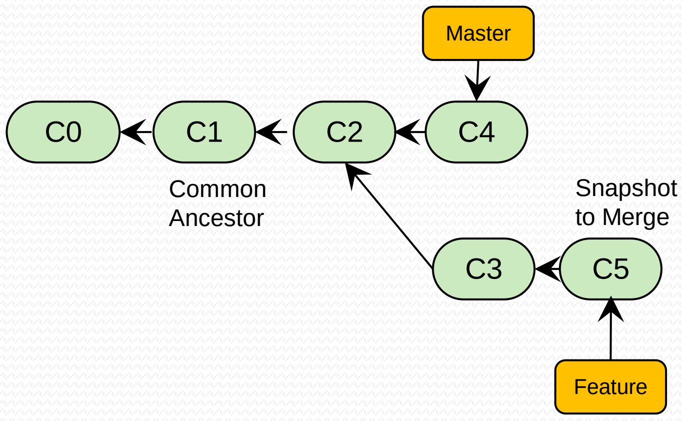
Best practices:

- Keep option of branching only if required, keep it simple.
- Incase of centralized VCS, merging should be done frequently / sooner (daily recommended)
- Use tagging as and when required.
- Delete unwanted branches.
- Identify shared components in details and in advance.

- **Merge only when your branch is compliable and stable.**



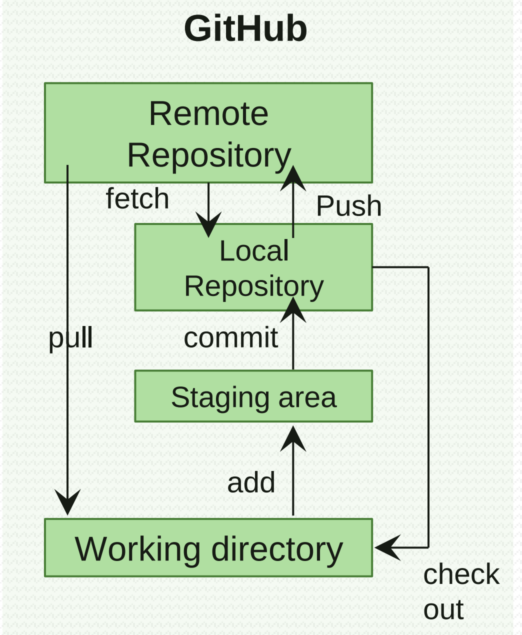
Branching and merging scenario



Simple Branching and merging in GIT project

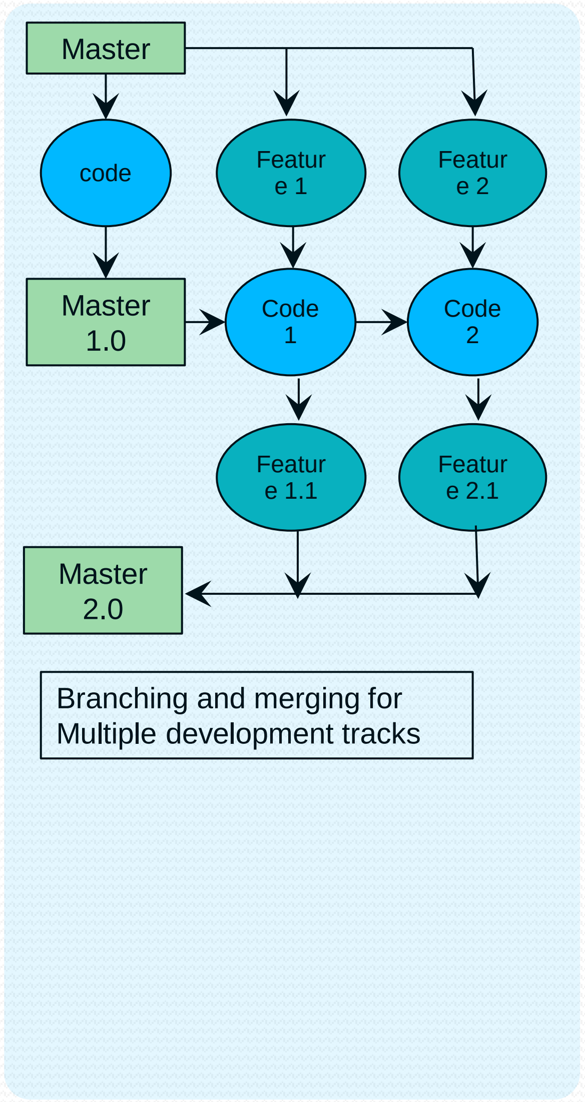
GIT - DVCS

- GIT has bash /command-line interface and GUI as well.
- Git is a distributed VCS. Provides extremely fast operation. Does not need centralized server. Once the entire repository is pulled on local machine, network / internet connection is not required for most of the VCS operations.
  - **Working area:** similar to work area, development environment.
  - **Staging area:** where you store a place a snapshot before committing changes to the local repo.
  - **Local repo:** is a copy of remote repository. This is where you have all versioning of data/code/artifacts maintained.
  - **Remote repository:** this can be the GitHub repository or a remote repository maintained on a server on intranet.
- GIT works on most of the OS platforms (OSx, Windows, Unix / Linux).
- On installation, the GIT config should be run to configure user name, email ID, etc.
- In order to start using the VCS, the folder that you want to version control, run the command 'git init'. This enables the VCS and tracking of file changes in the folder.

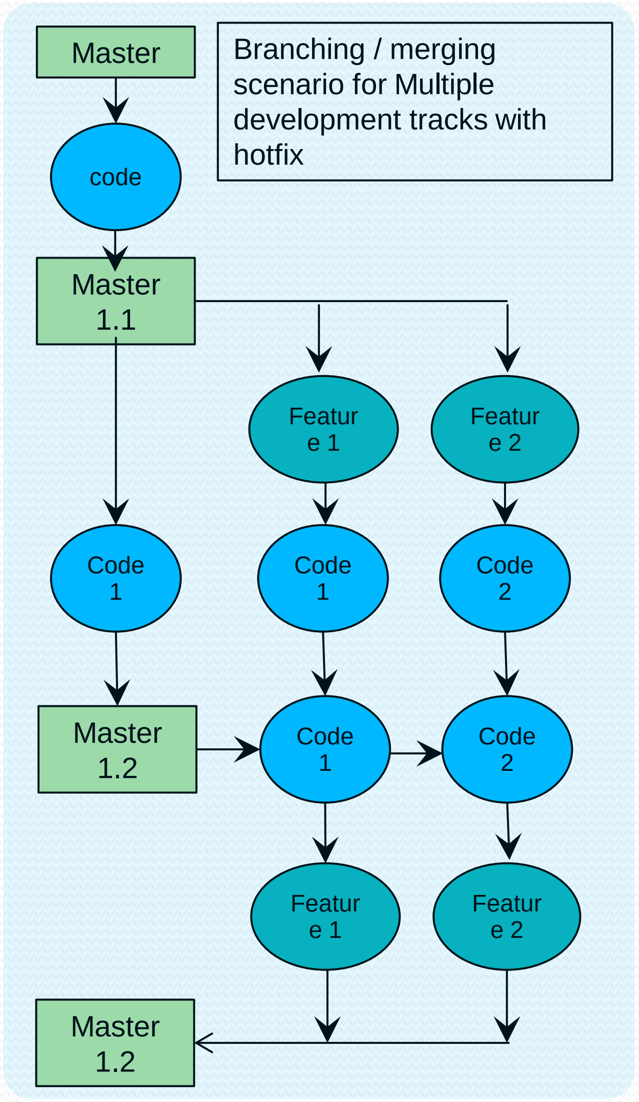


Download 'git' from, http://git-scm.com

GIT – Branching Strategies



Branching and merging for Multiple development tracks



Branching / merging scenario for Multiple development tracks with hotfix

To rename a remote repository,

```
git remote rename <remote-name><URL>
```

Verify after renaming,

```
git remote –v
```

To delete existing remote,

```
git remote rm <url>
git remote –v
```

To push commits made to local repo onto remote repo,

```
git push origin master
git push <remote name> <branch>
```

To push local branch to remote branch and rename remote,

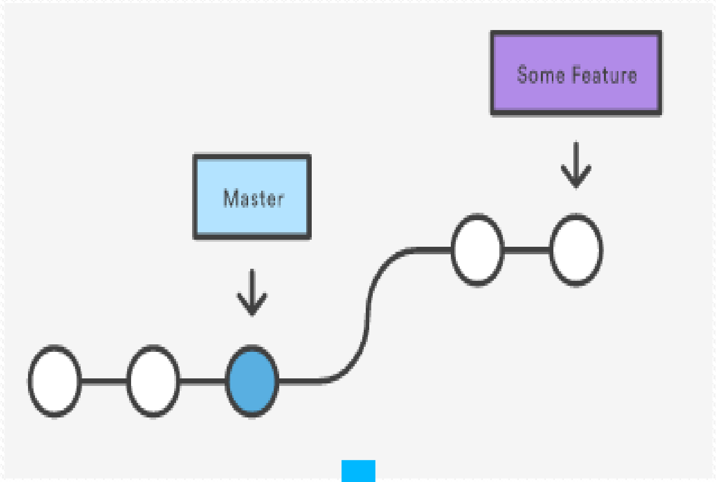
```
git push <remote-name><local branch>:<remote branch-name>
```

GIT common Commands

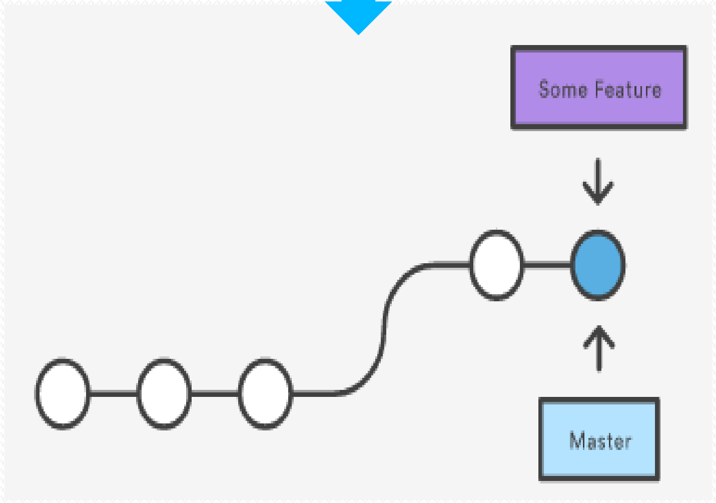
- Below are certain commands that are used often while we use ‘GIT’.
  - git status
  - git add
  - git rm –cached -- to remove file added to staging area
  - git commit –m “comment”
  - git log --- use ‘shift+z’ to come out of the log
  - git diff -- provides info about what has changed in the file
  - git diff –cached   --- diff for the files in the staging area.
  - git branch -- provides list of all branches
  - git branch <branch name> -- create a branch
  - git remote add <name> <url> -- to add remote repo.
  - git checkout <branch name> -- switch to mentioned branch
  - git clone <https://remote repo> -- clone from remote repo.
  - git pull <repo name> <branch name>

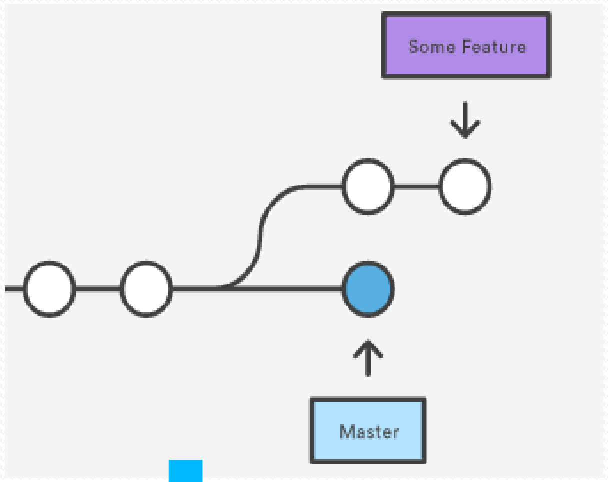
Always perform a ‘Pull’ action before ‘pushing’ the code to remote repo.

GIT Merge operations

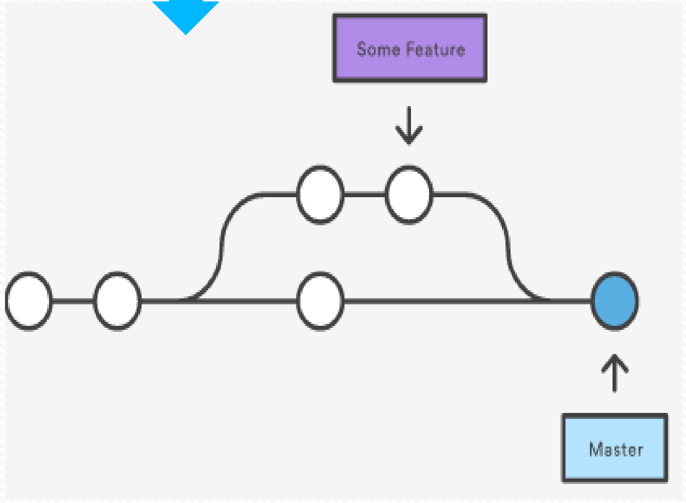


<u>After Fast-forward Merge</u>





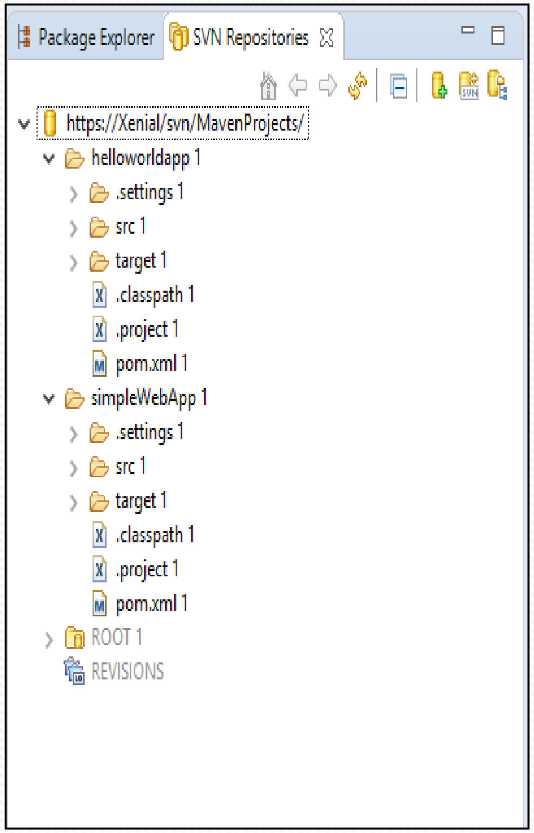
<u>3-way Merge</u>



Subversion (SVN) - VCS

- Easy to use and most favorite among developer community.
- Open Source under apache project.
- To start with,
    - Identify a machine to install SVN server.
    - Tortoise SVN is a client software used for connecting to SVN server.
    - The client has repository browser.
    - Default folder structure involves,
        - Trunk --- development activities
        - Tags ---- a label for released version
        - Branches --- release activities
- All development work is carries out in trunk folder. Once the code is ready for release a branch is created in branches as ‘release-1’ (example) and once the code is moved to production a tag is created as a label for the released version.
- For various hotfix activities a branch is derived from the tag.
- SVN has PowerShell plugin for scripting and automation, enables automated version control operation while integrating or moving through development activities.
- VisualSVN supports HTML5, repositories can be accessed over web.

Subversive SVN plug-in: helps to provide team explorer view of project repository



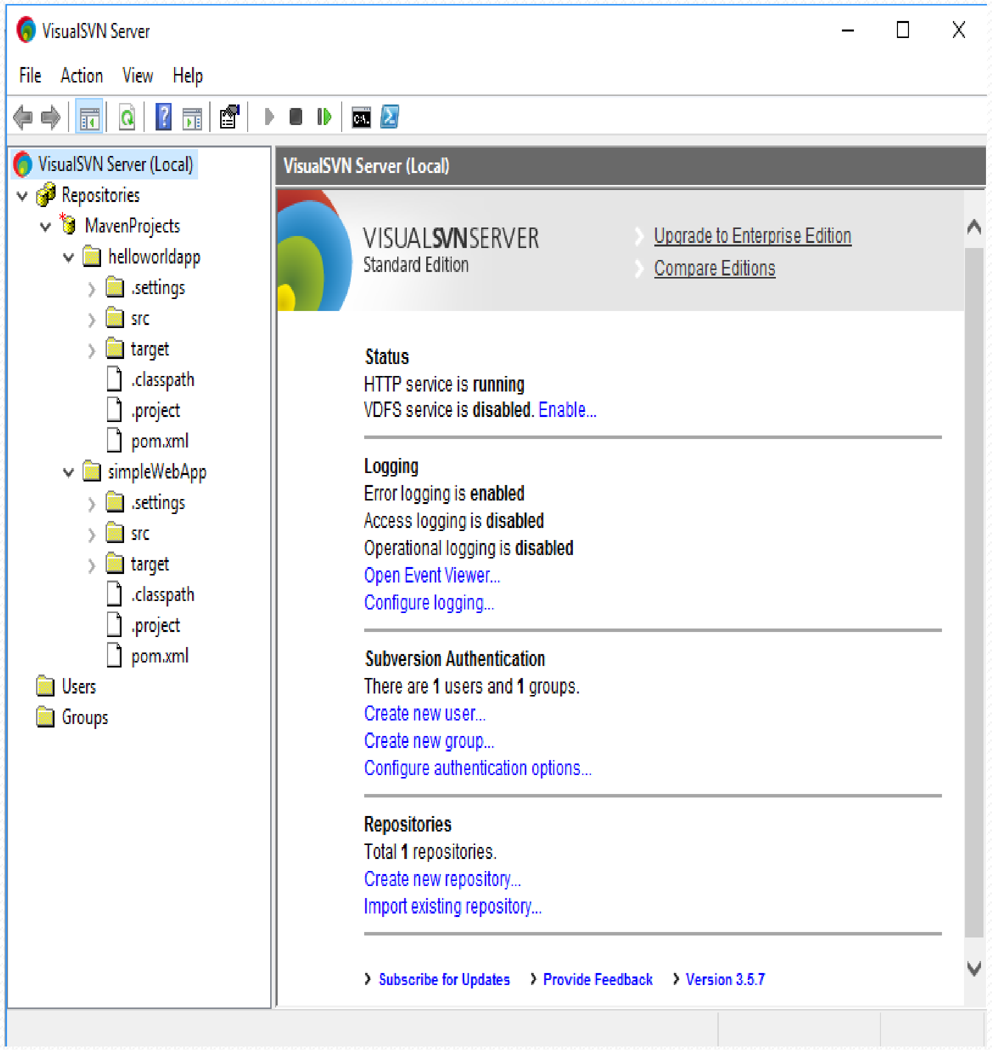
```
https://Xenial/svn/MavenProjects/
  helloworldapp 1
    .settings 1
    src 1
    target 1
    .classpath 1
    .project 1
    pom.xml 1
  simpleWebApp 1
    .settings 1
    src 1
    target 1
    .classpath 1
    .project 1
    pom.xml 1
  ROOT 1
  REVISIONS
```

Subversion (SVN) - VCS

VisualSVN Server is a freeware Apache Subversion **server** package for Windows.

Features:

- **Active directory single sign-on**: allows user to access SVN repository with their windows active directory credentials.
- Provides **remote server administration** access.
- **Multisite repository replication**: Using SVN distributed file system to fast replicate VCS database across geographically separated servers.
- Repository **management delegation**: can delegate repository management to non-administrator user.



Status
HTTP service is **running**
VDFS service is **disabled**. Enable...

Logging
Error logging is **enabled**
Access logging is **disabled**
Operational logging is **disabled**
Open Event Viewer...
Configure logging...

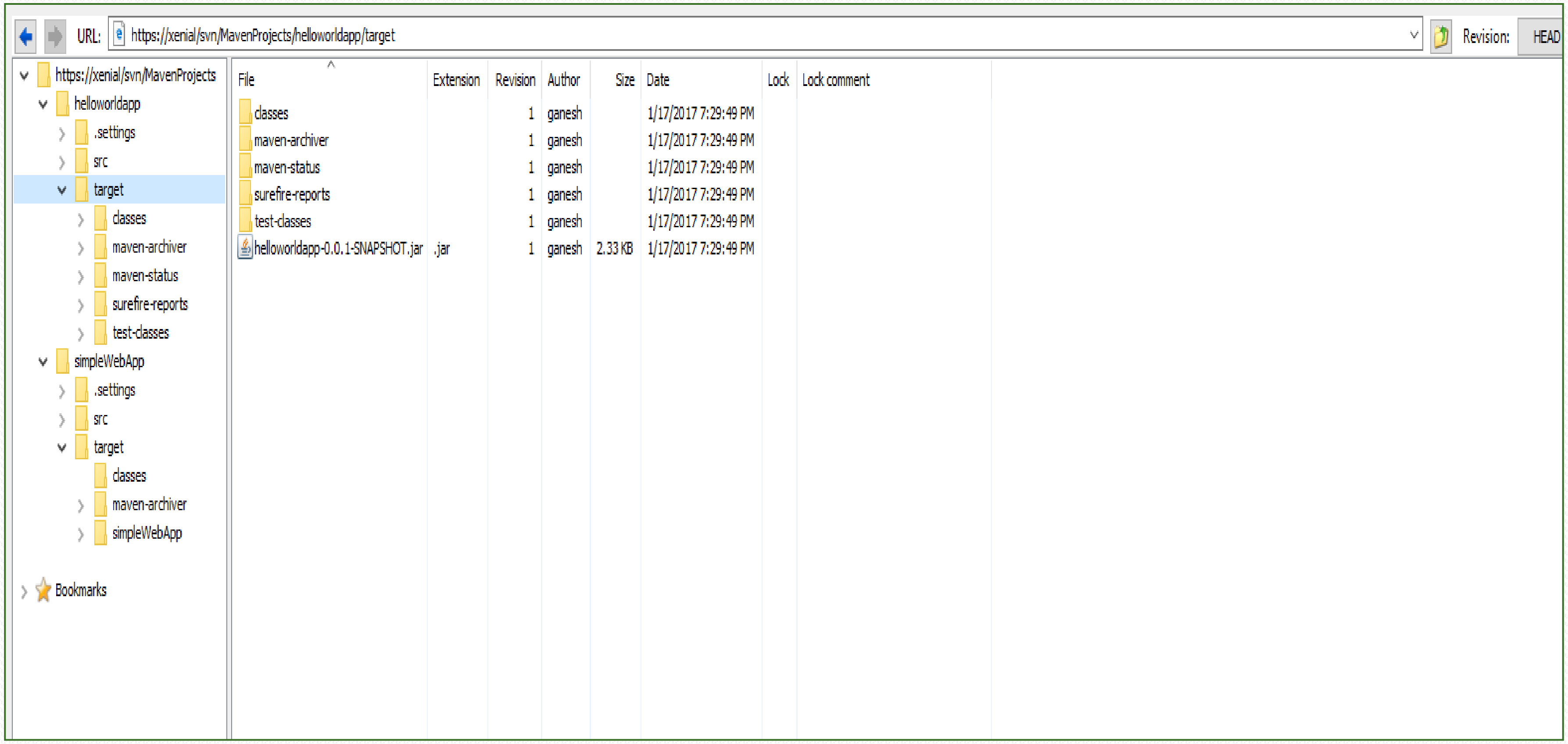
Subversion Authentication
There are **1** users and **1** groups.
Create new user...
Create new group...
Configure authentication options...

Repositories
Total **1** repositories.
Create new repository...
Import existing repository...

Subscribe for Updates · Provide Feedback · Version 3.5.7

Subversion (SVN) - VCS

Repository Browser View of a project repository


URL: https://xenial/svn/MavenProjects/helloworldapp/target

Revision: HEAD

- https://xenial/svn/MavenProjects
  - helloworldapp
    - .settings
    - src
    - target
      - classes
      - maven-archiver
      - maven-status
      - surefire-reports
      - test-classes
  - simpleWebApp
    - .settings
    - src
    - target
      - classes
      - maven-archiver
      - simpleWebApp
- Bookmarks

File	Extension	Revision	Author	Size	Date	Lock	Lock comment
classes		1	ganesh		1/17/2017 7:29:49 PM		
maven-archiver		1	ganesh		1/17/2017 7:29:49 PM		
maven-status		1	ganesh		1/17/2017 7:29:49 PM		
surefire-reports		1	ganesh		1/17/2017 7:29:49 PM		
test-classes		1	ganesh		1/17/2017 7:29:49 PM		
helloworldapp-0.0.1-SNAPSHOT.jar	.jar	1	ganesh	2.33 KB	1/17/2017 7:29:49 PM		

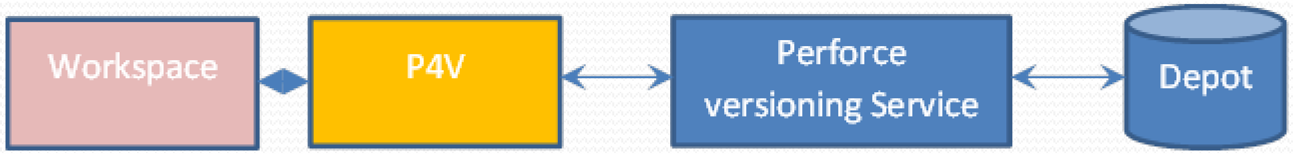
Perforce (P4V)

What is Perforce Visual Client (P4V)?

P4V, the Perforce Visual Client, is Perforce's cross-platform graphical user interface. You can use P4V on Windows, Mac, UNIX, and Linux computers and benefit from an identical interface regardless of platform.

To use Perforce to manage files, you typically connect to the Perforce versioning service using an application like P4V.

P4V enables you to check files in and out, and perform various other versioning tasks.



Above diagram shows how one can use P4V to connect to the versioning service and in-turn the depot.

Thank You