What do "branch", "tag" and "trunk" mean in Subversion repositories?

Asked 16 years, 4 months ago Modified 3 years, 11 months ago

Viewed 465k times

1208

I've seen these words a lot around Subversion (and I guess general repository) discussions.
I have been using **SVN** for my projects for the last few years, but I've never grasped the complete concept of these directories.

What do they mean?

svn branch terminology trunk

Share

Improve this question

Follow

edited Jan 6, 2021 at 8:26

Wolf

10.2k ● 8 ● 66 ● 112

asked Aug 19, 2008 at 13:22

grapefrukt

27k ● 6 ● 51 ● 74

29 Here's a good article I ran across explaining how/when to use trunk, branch, and tags. I'd not used source control before, but this article made it pretty easy for a noob like me to understand. Day-to-day with Subversion – badmoon Aug 19, 2008 at 14:53

16 Answers

Sorted by: Highest score (default)

924

Hmm, not sure I agree with Nick re tag being similar to a branch. A tag is just a marker

- **Trunk** would be the main body of development, originating from the start of the project until the present.
- **Branch** will be a copy of code derived from a certain point in the trunk that is used for applying major changes to the code while preserving the integrity of the code in the trunk. If the major changes work according to plan, they are usually merged back into the trunk.
- **Tag** will be a point in time on the trunk or a branch that you wish to preserve. The two main reasons for preservation would be that either this is a major release of the software, whether alpha, beta, RC or RTM, or this is the most stable point of the software before major revisions on the trunk were applied.

In open source projects, major branches that are not accepted into the trunk by the project stakeholders can

become the bases for *forks* -- e.g., totally separate projects that share a common origin with other source code.

The branch and tag subtrees are distinguished from the trunk in the following ways:

Subversion allows sysadmins to create *hook scripts* which are triggered for execution when certain events occur; for instance, committing a change to the repository. It is very common for a typical Subversion repository implementation to treat any path containing "/tag/" to be write-protected after creation; the net result is that tags, once created, are immutable (at least to "ordinary" users). This is done via the hook scripts, which enforce the immutability by preventing further changes if **tag** is a parent node of the changed object.

Subversion also has added features, since version 1.5, relating to "branch merge tracking" so that changes committed to a **branch** can be merged back into the trunk with support for incremental, "smart" merging.

Share Improve this answer

Follow

edited Apr 4, 2019 at 1:12

JoGusto

973 ● 9 ● 8

answered Aug 19, 2008 at 13:35

Jon Limjap

95.3k ● 15 ● 103 ● 153

288 The confusion with tags and branches is that in svn there really is no distinction between them, besides the name of

the directory. In svn you are able to commit changes to a tag, and in fact it is difficult to prevent this. Most other VCSes treat tags as immutable snapshots (points in time). – Ken Liu Oct 9, 2009 at 18:36

4 `Tags` directory is also often used for milestones testing and verification by the regular user. This would be a good place to put a prototype too (just some ideas on top of my head). – Jeff Noel Oct 26, 2012 at 7:09

7 @KenLiu There are hooks that can make tags immutable. That is, you can create, and checkout a tag, but not make any changes. Of course, a tag being just part of the repository means the full history is available. If someone changes a tag, you can track that and why. In many VCS, if you modify a tag, there may not be any way to know. – David W. Mar 1, 2015 at 19:33

3 Maybe **stable branches** should be mentioned: changes made there are normally not *merged back into the trunk*. – Wolf May 6, 2015 at 12:59

6 My understanding is that in a "perfect world" no development should ever happen in the trunk, the trunk should always be either the exact code that is in live or code that is about to be released into live. as such that would make the branches the main body of development. – MikeT Aug 3, 2015 at 9:20

558

First of all, as @AndrewFinnell and @KenLiu point out, in SVN the directory names themselves mean nothing -- "trunk, branches and tags" are simply a common convention that is used by most repositories. Not all projects use all of the directories (it's reasonably common not to use "tags" at all), and in fact, nothing is stopping

you from calling them anything you'd like, though breaking convention is often confusing.

I'll describe probably the most common usage scenario of branches and tags, and give an example scenario of how they are used.

- **Trunk**: The main development area. This is where your next major release of the code lives, and generally has all the newest features.
- **Branches**: Every time you release a major version, it gets a branch created. This allows you to do bug fixes and make a new release without having to release the newest - possibly unfinished or untested - features.
- **Tags**: Every time you release a version (final release, release candidates (RC), and betas) you make a tag for it. This gives you a point-in-time copy of the code as it was at that state, allowing you to go back and reproduce any bugs if necessary in a past version, or re-release a past version exactly as it was. Branches and tags in SVN are lightweight - on the server, it does not make a full copy of the files, just a marker saying "these files were copied at this revision" that only takes up a few bytes. With this in mind, you should never be concerned about creating a tag for any released code. As I said earlier, tags are often omitted and instead, a changelog or other document clarifies the revision number when a release is made.

For example, let's say you start a new project. You start working in "trunk", on what will eventually be released as version 1.0.

- **trunk/ - development version, soon to be 1.0**
- branches/ - empty

Once 1.0.0 is finished, you branch trunk into a new "1.0" branch, and create a "1.0.0" tag. Now work on what will eventually be 1.1 continues in trunk.

- trunk/ - development version, **soon to be 1.1**
- **branches/1.0 - 1.0.0 release version**
- **tags/1.0.0 - 1.0.0 release version**

You come across some bugs in the code, and fix them in trunk, and then merge the fixes over to the 1.0 branch. You can also do the opposite, and fix the bugs in the 1.0 branch and then merge them back to trunk, but commonly projects stick with merging one-way only to lessen the chance of missing something. Sometimes a bug can only be fixed in 1.0 because it is obsolete in 1.1. It doesn't really matter: you only want to make sure that you don't release 1.1 with the same bugs that have been fixed in 1.0.

- trunk/ - development version, soon to be 1.1
- branches/1.0 - **upcoming 1.0.1 release**
- tags/1.0.0 - 1.0.0 release version

Once you find enough bugs (or maybe one critical bug), you decide to do a 1.0.1 release. So you make a tag "1.0.1" from the 1.0 branch, and release the code. At this point, trunk will contain what will be 1.1, and the "1.0" branch contains 1.0.1 code. The next time you release an update to 1.0, it would be 1.0.2.

- trunk/ - development version, soon to be 1.1
- branches/1.0 - **upcoming 1.0.2 release**
- tags/1.0.0 - 1.0.0 release version
- **tags/1.0.1 - 1.0.1 release version**

Eventually you are almost ready to release 1.1, but you want to do a beta first. In this case, you likely do a "1.1" branch, and a "1.1beta1" tag. Now, work on what will be 1.2 (or 2.0 maybe) continues in trunk, but work on 1.1 continues in the "1.1" branch.

- trunk/ - development version, **soon to be 1.2**
- branches/1.0 - upcoming 1.0.2 release
- **branches/1.1 - upcoming 1.1.0 release**
- tags/1.0.0 - 1.0.0 release version
- tags/1.0.1 - 1.0.1 release version
- **tags/1.1beta1 - 1.1 beta 1 release version**

Once you release 1.1 final, you do a "1.1" tag from the "1.1" branch.

You can also continue to maintain 1.0 if you'd like, porting bug fixes between all three branches (1.0, 1.1, and trunk). The important takeaway is that for every main version of the software you are maintaining, you have a branch that contains the latest version of code for that version.

Another use of branches is for features. This is where you branch trunk (or one of your release branches) and work on a new feature in isolation. Once the feature is completed, you merge it back in and remove the branch.

- trunk/ - development version, soon to be 1.2
- branches/1.1 - upcoming 1.1.0 release
- **branches/ui-rewrite - experimental feature branch**

The idea of this is when you're working on something disruptive (that would hold up or interfere with other people from doing their work), something experimental (that may not even make it in), or possibly just something that takes a long time (and you're afraid if it holding up a 1.2 release when you're ready to branch 1.2 from trunk), you can do it in isolation in branch. Generally you keep it up to date with trunk by merging changes into it all the time, which makes it easier to re-integrate (merge back to trunk) when you're finished.

Also note, the versioning scheme I used here is just one of many. Some teams would do bug fix/maintenance releases as 1.1, 1.2, etc., and major changes as 1.x, 2.x,

etc. The usage here is the same, but you may name the branch "1" or "1.x" instead of "1.0" or "1.0.x". (Aside, semantic versioning is a good guide on how to do version numbers).

Share Improve this answer Follow

edited Apr 10, 2012 at 6:05

answered Sep 20, 2008 at 19:00

gregmac
25.2k • 11 • 90 • 120

1 It is not true that in SVN tags are lightweight. In SVN, tags are a full code export, and lose any reference to where in the source tree they cam from. – Baruch Sep 27, 2011 at 11:07

6 @baruch - That's completely wrong. Tags are lightweight and are (as far as Subversion itself is concerned) identical to branches. – Josh Kelley Jul 14, 2014 at 14:08

2 Can I get a quote on where it's stated that tags/branches are lightweight? It doesn't seem that way.. – Cardin May 8, 2015 at 2:45

4 @Cardin I don't have reference right now, but important to note tags are lightweight on the server, but not client. If you checkout all tags, you'll get that many full copies. However, if you look at repository size on server, it will only increase by a few bytes per tag. This is why you should not checkout the root directory, generally speaking. – gregmac Feb 2, 2016 at 21:19

@Cardin Since you asked for a reference, take a look at the "Cheap Copies" block on this page: svnbook.red-bean.com/en/1.7/…. – Darryl Apr 28, 2020 at 16:26

97

In addition to what Nick has said you can find out more at [Streamed Lines: Branching Patterns for Parallel Software Development]

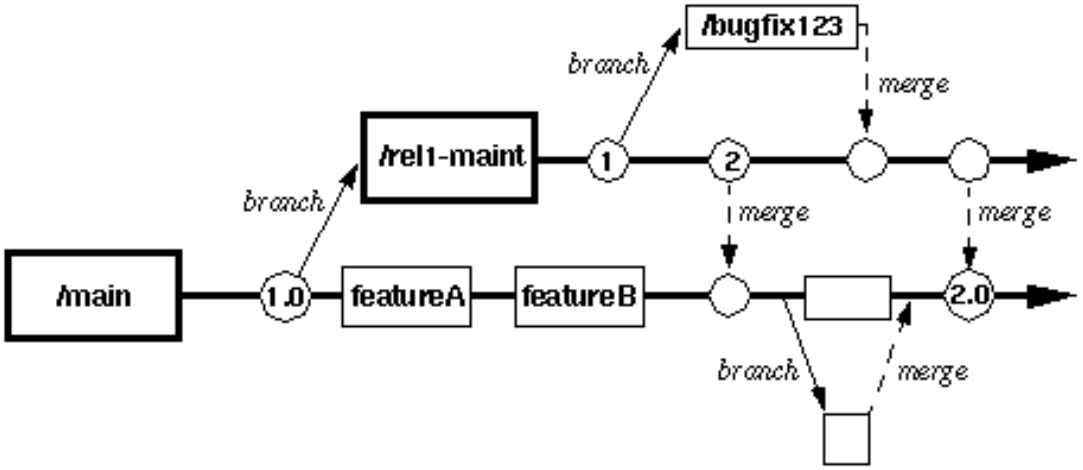

Figure 5: A sample version tree diagram for a project or system

In this figure `main` is the trunk, `rel1-maint` is a branch and `1.0` is a tag.

Share Improve this answer

Follow

edited May 3, 2012 at 0:46

Ben

57k • 50 • 183 • 228

answered Aug 19, 2008 at 13:58

grom

16.1k • 19 • 65 • 67

1 @Wolf he could be - that diagram is pretty generic regardless of the tooling. All SCMs use different words but the same concepts, there's no difference between trunk and Main; or trunk and master. That diagram shows how my current company uses SVN. – gbjbaanb May 27, 2015 at 10:58

@gbjbaanb Thanks for sharing. ...and *tags* seem not to be addressed by the question. Is it pure coincidence (also in

your current company) that that no merges go from main to from maintained branches? – Wolf May 27, 2015 at 11:17

@Wolf No coincidence - only branch from trunk, do work, merge back onto trunk. Then branch off trunk to a tag branch. We're considering another 'trunk' called Integration that has finished branches merged to it for testing that doesn't constitute a release, trunk is still used for those branches that we decide to put in the next release. The only time you merge from trunk to a branch is to update a long-running branch, but its better (and easier) to simply create a new branch off trunk and merge your old branch's changes to it if you need that. – gbjbaanb May 27, 2015 at 12:10

76

In general (tool agnostic view), a branch is the mechanism used for parallel development. An SCM can have from 0 to n branches. Subversion has 0.

- **Trunk** is a main branch **recommended** by Subversion, but you are in no way forced to create it. You could call it 'main' or 'releases', or not have one at all!
- **Branch** represents a development effort. It should never be named after a resource (like 'vonc_branch') but after:
  - a purpose 'myProject_dev' or 'myProject_Merge'
  - a release perimeter 'myProjetc1.0_dev'or myProject2.3_Merge' or 'myProject6..2_Patch1'...
- **Tag** is a snapshot of files in order to easily get back to that state. The problem is that tag and branch is

the same in Subversion. And I would definitely recommend the paranoid approach:

> you can use one of the access control scripts provided with Subversion to prevent anyone from doing anything but creating new copies in the tags area.

A tag is final. Its content should never change. NEVER. Ever. You forgot a line in the release note? Create a new tag. Obsolete or remove the old one.

Now, I read a lot about "merging back such and such in such and such branches, then finally in the trunk branch". That is called **merge workflow** and there is **nothing mandatory here**. It is not because you have a trunk branch that you *have to merge back* anything.

By convention, the trunk branch can represent the current state of your development, but that is for a simple sequential project, that is a project which has:

- no 'in advance' development (for the preparing the next-next version implying such changes that they are not compatible with the current 'trunk' development)
- no massive refactoring (for testing a new technical choice)
- no long-term maintenance of a previous release

Because with one (or all) of those scenario, you get yourself four 'trunks', four 'current developments', and not all you do in those parallel development will necessarily have to be merged back in 'trunk'.

Share Improve this answer

Follow

edited Jan 3, 2016 at 20:07

community wiki
6 revs, 5 users 77%
VonC

39

In SVN a tag and branch are really similar.

Tag = a defined slice in time, usually used for releases

Branch = also a defined slice in time that development can continue on, usually used for major version like 1.0, 1.5, 2.0, etc, then when you release you tag the branch. This allows you to continue to support a production release while moving forward with breaking changes in the trunk

Trunk = development work space, this is where all development should happen, and then changes merged back from branch releases.

Share Improve this answer

Follow

answered Aug 19, 2008 at 13:25

Nick Berardi
54.8k ● 15 ● 117 ● 136

33

They don't really have any formal meaning. A folder is a folder to SVN. They are a generally accepted way to organize your project.

- The trunk is where you keep your main line of developmemt. The branch folder is where you might create, well, branches, which are hard to explain in a short post.
- A branch is a copy of a subset of your project that you work on separately from the trunk. Maybe it's for experiments that might not go anywhere, or maybe it's for the next release, which you will later merge back into the trunk when it becomes stable.
- And the tags folder is for creating tagged copies of your repository, usually at release checkpoints.

But like I said, to SVN, a folder is a folder. `branch`, `trunk` and tag are just a convention.

I'm using the word 'copy' liberally. SVN doesn't actually make full copies of things in the repository.

Share Improve this answer

Follow

edited Jan 3, 2016 at 20:01

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 19, 2008 at 13:27

Eric Z Beard

38.4k ● 27 ● 101 ● 147

13

The **trunk** is the development line that holds the latest source code and features. It should have the latest bug fixes in it as well as the latest features added to the project.

The **branches** are usually used to do something away from the trunk (or other development line) that would otherwise *break* the build. New features are often built in a branch and then merged back into the trunk. Branches often contain code that are not necessarily approved for the development line it branched from. For example, a programmer could try an optimization on something in a branch and only merge back in the development line once the optimization is satisfactory.

The **tags** are snapshots of the repository at a particular time. No development should occur on these. They are most often used to take a copy of what was released to a client so that you can easily have access to what a client is using.

Here's a link to a very good guide to repositories:

- Source Control HOWTO

The articles in Wikipedia are also worth reading.

Share Improve this answer

Follow

answered Aug 19, 2008 at 13:37

mbillard

38.8k ● 18 ● 75 ● 98

12

Now that's the thing about software development, there's no consistent knowledge about anything, everybody seems to have it their own way, but that's because it is a relatively young discipline anyway.

Here's my plain simple way,

trunk - The trunk directory contains the most current, approved, and merged body of work. Contrary to what many have confessed, my trunk is only for clean, neat, approved work, and not a development area, but rather a release area.

At some given point in time when the trunk seems all ready to release, then it is tagged and released.

branches - The branches directory contains experiments and ongoing work. Work under a branch stays there until is approved to be merged into the trunk. For me, this is the area where all the work is done.

For example: I can have an *iteration-5* branch for a fifth round of development on the product, maybe a *prototype-9* branch for a ninth round of experimenting, and so on.

tags - The tags directory contains snapshots of approved branches and trunk releases. Whenever a branch is approved to merge into the trunk, or a release is made of the trunk, a snapshot of the approved branch or trunk release is made under tags.

I suppose with tags I can jump back and forth through time to points interest quite easily.

Share Improve this answer

Follow

answered Jun 30, 2009 at 11:50

BakerTheHacker

354 ● 2 ● 2

10

I found this great tutorial regarding SVN when I was looking up the website of the author of the OpenCV 2 Computer Vision Application Programming Cookbook and I thought I should share.

He has a tutorial on how to use SVN and what the phrases 'trunk', 'tag' and 'branch' mean.

Cited directly from his tutorial:

> The current version of your software project, on which your team is currently working is usually located under a directory called **trunk**. As the project evolves, the developer updates that version fix bugs, add new features) and submit his changes under that directory.
>
> At any given point in time, you may want to freeze a version and capture a snapshot of the software as it is at this stage of the development. This generally corresponds to the official versions of your software, for example, the ones you will deliver to your clients. These snapshots

> are located under the **tags** directory of your project.
>
> Finally, it is often useful to create, at some point, a new line of development for your software. This happens, for example, when you wish to test an alternative implementation in which you have to modify your software but you do not want to submit these changes to the main project until you decide if you adopt the new solution. The main team can then continue to work on the project while other developer work on the prototype. You would put these new lines of development of the project under a directory called **branches**.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Jul 26, 2011 at 8:56

Vince

111 ● 1 ● 2

9

The trunk directory is the directory that you're probably most familiar with, because it is used to hold the most recent changes. Your main codebase should be in trunk.

The branches directory is for holding your branches, whatever they may be.

The tags directory is basically for tagging a certain set of files. You do this for things like releases, where you want "1.0" to be these files at these revisions and "1.1" to be these files at these revisions. You usually don't modify tags once they're made. For more information on tags, see *Chapter 4. Branching and Merging* (in *Version Control with Subversion*).

Share Improve this answer

Follow

edited Apr 9, 2012 at 18:08

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 19, 2008 at 13:28

bradtgmurray

14.3k ● 10 ● 39 ● 36

9

One of the reasons why everyone has a slightly different definition is because Subversion implements **zero** support for branches and tags. Subversion basically says: *We looked at* full-featured *branches and tags in other systems and did not found them useful, so we did not implement anything. Just make a copy into a new directory with a name* convention *instead*. Then of course everyone is free to have slightly different conventions. To understand the difference between a *real* tag and a mere copy + naming convention see the Wikipedia entry *Subversion tags & branches*.

Share Improve this answer

Follow

edited Apr 9, 2012 at 18:23

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Nov 17, 2010 at 20:39

MarcH

19.7k ● 1 ● 31 ● 26

8

> Tag = a defined slice in time, usually used for releases

I think this is what one typically means by "tag". But in Subversion:

> They don't really have any formal meaning. A folder is a folder to SVN.

which I find rather confusing: a revision control system that knows nothing about branches or tags. From an implementation point of view, I think the Subversion way of creating "copies" is very clever, but me having to know about it is what I'd call a *leaky abstraction*.

Or perhaps I've just been using CVS far too long.

Share Improve this answer

Follow

edited Apr 9, 2012 at 18:37

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 4, 2008 at 14:39

sme

5,671 ● 7 ● 34 ● 30

An alternative perspective is that the opposite is true, that imposing the concept of tags on subversion's object model would be leaky abstraction in the opposite direction. As I'm guessing you know, subversion was a reaction to CVS, an attempt to "do CVS right." I couldn't find the reference, but the original subversion designers have said they threw out the concept of tags 100% deliberately, that the distinction between branches and tags is purely a policy issue. If teams want to impose policy and convention on top of subversion's object model, so be it. That's exactly what we have today.
– Darryl May 28, 2013 at 18:11

6

I think that some of the confusion comes from the difference between the concept of a tag and the implementation in SVN. To SVN a tag is a branch which is a copy. Modifying tags is considered wrong and in fact tools like TortoiseSVN will warn you if you attempt to modify anything with ../tags/.. in the path.

Share Improve this answer

Follow

answered Aug 19, 2008 at 17:24

denis phillips

12.7k ● 5 ● 34 ● 47

I'm not really sure what 'tag' is, but branch is a fairly common source control concept.

Basically, a branch is a way to work on changes to the code without affecting trunk. Say you want to add a new feature that's fairly complicated. You want to be able to check in changes as you make them, but don't want it to affect trunk until you're done with the feature.

First you'd create a branch. This is basically a copy of trunk as-of the time you made the branch. You'd then do all your work in the branch. Any changes made in the branch don't affect trunk, so trunk is still usable, allowing others to continue working there (like doing bugfixes or small enhancements). Once your feature is done you'd integrate the branch back into trunk. This would move all your changes from the branch to trunk.

There are a number of patterns people use for branches. If you have a product with multiple major versions being supported at once, usually each version would be a branch. Where I work we have a QA branch and a Production branch. Before releasing our code to QA we integrate changes to the QA branch, then deploy from there. When releasing to production we integrate from the QA branch to the Production branch, so we know the code running in production is identical to what QA tested.

Here's the Wikipedia entry on branches, since they probably explain things better than I can. :)

Share Improve this answer

Follow

edited Apr 9, 2012 at 18:26

Peter Mortensen

31.6k • 22 • 109 • 133

answered Aug 19, 2008 at 13:34

Herms

38.7k • 13 • 79 • 104

4

Trunk : After the completion of every sprint in agile we come out with a partially shippable product. These releases are kept in trunk.

Branches : All parallel developments codes for each ongoing sprint are kept in branches.

Tags : Every time we release a partially shippable product kind of beta version, we make a tag for it. This gives us the code that was available at that point of time, allowing us to go back at that state if required at some point during development.

Share Improve this answer

Follow

answered Apr 25, 2017 at 11:43

Ujjwal

2,533 • 2 • 12 • 8

This is *your* particular workflow, it is not applicable in general. – Jason S Sep 18, 2018 at 16:55

For people familiar with GIT, master in GIT is equivalent to trunk in SVN.

4

Branch and tag has same terminology in both GIT and SVN.

Share Improve this answer

Follow

answered Apr 4, 2018 at 5:07

Desert Rose

3,404 ● 1 ● 31 ● 36

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.