Subject: GIT
Posted by unodgs on Wed, 21 May 2008 20:22:57 GMT
View Forum Message <> Reply to Message

I found GIT to be very fast. Importing 1500 files of uppsrc to git repo took only few seconds (on win32, p4 3ghz and msysgit). And Git dosn't create .git directory in every subdirectory. The only con right now is git needs to be installed on remote machine if we want our repo to be shared with others (bazaar dosn't have this restriction). I think we should move from uvs/svn to git. It's much more powerful. For example I can commit to my local repo not working code and sync it with remote repo if my changes will be finished. I have started to work on UltimateGit app (as there is no really good gui for git IMO) but I think we should finally add to TheIDE some kind of VCS interface so anyone willing could write plugin to his favourite VCS.

Subject: Re: GIT
Posted by mr_ped on Thu, 22 May 2008 07:15:10 GMT
View Forum Message <> Reply to Message

yes, yes, yes, yes...

I'm thinking about trying out GIT for several months, but I never had the mood+time to really try, so if U++ will move to it, at least I will have more motivation.

How about GIT and windows? Did you think about this? (it doesn't scare *me* too much, as I'm closing all my old relations to windows over years and now I'm working more at linux or embedded platforms, but Mirek looks like he has to support lot of legacy applications even for W98 .)

Subject: Re: GIT
Posted by unodgs on Thu, 22 May 2008 12:58:59 GMT
View Forum Message <> Reply to Message

As I wrote I used msysgit which is native port of GIT for win32 platform
BTW: here you'll find nice git commands wallpaper :
http://zrusin.blogspot.com/2007/09/git-cheat-sheet.html

Subject: Re: GIT
Posted by mdelfede on Thu, 22 May 2008 13:13:36 GMT
View Forum Message <> Reply to Message

well, I was just thinking to start an svn add-on for theide... I'm in favour of a server-based repository, imho shared repos depends too much on how often people shares them.
And don't see very much advantage on use a shared system as git and then setup a server based repository... But that's only my opinion. I looked also over Mercurial some time ago... nice stuff too, in some ways superior to git, but also a shared repository.

Ciao

Max

Subject: Re: GIT
Posted by masu on Thu, 22 May 2008 13:52:50 GMT
View Forum Message <> Reply to Message

For me a huge advantage is as Uno mentioned the ability to work with the repo offline.

The next second advantage is that you also have a natural backup system, because chances are high somebody else has a repo copy in case of local data loss.

I had a huge project I put into a GIT repo out of curiosity and I found it to be much smaller in total in comparison to the original SVN working copy.

Matthias

Subject: Re: GIT
Posted by mdelfede on Thu, 22 May 2008 14:15:12 GMT
View Forum Message <> Reply to Message

I don't think GIT is bad, I only think that it can bring troubles if people don't share often enough their local repositories....
Linus Thorvalds uses it for Linux Kernel, but that's an obvious choice because he wants to have the full control of repository and decide which patches to apply. Let's say that he uses it like a central repo, and people who wants their patches applied must share with him.

With svn, if you make a patch, you put on central repo and all people CAN fetch it when they wants. With git, you must update your repo, remember to share with all other people (OR setup some sort of central repository, but then, what's the advantage of a distributed one???) an other people must fetch AND sync their local copy. There's no more an "official" copy on which people works, if you don't setup a central public repository.

I see git much more useful to keep some sort of local history of your working copy.

BTW, the fact that svn puts "hidden" .svn folders everywhere (which are usually not hidden on windows...) disturbs me too, but it has the advantage to keep all stuff in a singkle tree.
I don't know how git works, but I gess that it must keep management data somewhere too.

Ciao

Max

Subject: Re: GIT
Posted by Novo on Thu, 22 May 2008 14:20:26 GMT
View Forum Message <> Reply to Message

CVS, SVN, and ClearCase. GIT seems to be closer to ClearCase. At least it has a familiar

Subject: Re: GIT
Posted by tojocky on Mon, 09 Jun 2008 07:03:24 GMT
View Forum Message <> Reply to Message

I thing that it is a good link for comparation GIT and SVN. For my view I'd like GIT!

http://git.or.cz/gitwiki/GitSvnComparsion

Quote:For example the Mozilla repository is reported to be almost 12 GiB when stored in SVN using the fsfs backend. The fsfs backend also requires over 240,000 files in one directory to record all 240,000 commits made over the 10 year project history. The exact same history is stored in Git by only two files totaling just over 420 MiB. SVN requires 30x the disk space to store the same history.

Subject: Re: GIT
Posted by mirek on Mon, 09 Jun 2008 11:28:32 GMT
View Forum Message <> Reply to Message

tojocky wrote on Mon, 09 June 2008 03:03I thing that it is a good link for comparation GIT and SVN. For my view I'd like GIT!

http://git.or.cz/gitwiki/GitSvnComparsion

Quote:For example the Mozilla repository is reported to be almost 12 GiB when stored in SVN using the fsfs backend. The fsfs backend also requires over 240,000 files in one directory to record all 240,000 commits made over the 10 year project history. The exact same history is stored in Git by only two files totaling just over 420 MiB. SVN requires 30x the disk space to store the same history.

Actually, as far as disk usage goes, I could not care less...

Our infrastructure server will have no less than 300GB free (probably much more).

Mirek

Subject: Re: GIT
Posted by unodgs on Mon, 09 Jun 2008 19:19:21 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 09 June 2008 07:28
Our infrastructure server will have no less than 300GB free (probably much more).
Mirek
Much more important to me than space is speed of git. Listing all files from repository, commits, updates are noticable faster than in svn. Opening upp project in SmartSvn takes a while (my UltimateGit does the same in few seconds. We should really consider git or bazaar in the near future.

Subject: Re: GIT
Posted by mirek on Mon, 09 Jun 2008 19:34:52 GMT
View Forum Message <> Reply to Message

unodgs wrote on Wed, 21 May 2008 16:22
I have started to work on UltimateGit app (as there is no really good gui for git IMO) but I think we should finally add to TheIDE some kind of VCS interface so anyone willing could write plugin to his favourite VCS.

Will you make UltimateGit uvs2-like?

(I mean single sync for multiple repositories?)

Mirek

Subject: Re: GIT
Posted by tojocky on Mon, 09 Jun 2008 20:06:52 GMT
View Forum Message <> Reply to Message

Will be great for a single GUI application and more technologies as uvs2, git, and svn! GIT is more faster than SVN.

Subject: Re: GIT
Posted by unodgs on Mon, 09 Jun 2008 20:11:31 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 09 June 2008 15:34
Will you make UltimateGit uvs2-like?
(I mean single sync for multiple repositories?)

Mirek

Each repository has assigned a tab, so you can open many repos at once. Grouped operations are on to-do list

---

Subject: Re: GIT
Posted by mirek on Mon, 09 Jun 2008 21:18:39 GMT
View Forum Message <> Reply to Message

unodgs wrote on Mon, 09 June 2008 16:11luzr wrote on Mon, 09 June 2008 15:34
Will you make UltimateGit uvs2-like?
(I mean single sync for multiple repositories?)
Mirek

Each repository has assigned a tab, so you can open many repos at once. Grouped operations are on to-do list

I do not care about tabs, I need to do everything - commit/update ~7 repositories - in single click (Well, dialog listing what is going to happen would be nice as well).

Mirek

Subject: Re: GIT
Posted by sapiency on Sat, 12 Sep 2009 21:40:16 GMT
View Forum Message <> Reply to Message

hi

what about git now?
I cannot find any package with a ui for git?

reinhard

Subject: MERCURIAL
Posted by Didier on Sun, 13 Sep 2009 08:44:16 GMT
View Forum Message <> Reply to Message

I've been using MERCURIAL for a few years now and it works perfectly. It's works like GIT but has simple commands and great GUI both on linux (hgtk) and windows (TotoiseHg).

I even use it to manage my UPP src code localy: the 'hgpullsvn' allows me to pull the upp svn repository into my hg (meens mercurial) reposirory.

Passing from svn to Mercurial is just as easy as that.

The main advantage Mercurial, GIT, R.I.P. TeamWare (for those who knew it), and generally SCMs that store the repository data with is that you are free to do what you wan't localy WITHOUT disturbing the main repository (One is of course needed as central changeset repository).
==> Every one can therefore start managing their code locally and submit a 'bundle of changesets' for acceptation if they don't have direct deliver rigths.
==> Patches would be delivered as bundles ==> no merge error possible: all is managed by mercurial and complete history is always kept !!

In addition, there is a video where Linus Thorvalds presents GIT. He also talks about MERCURIAL as the only SCM close to GIT
Many open source projects are also swithing to mercurial

Mercurial is REALLY WORTH it !!!!!

Please take a look at it:
http://mercurial.selenic.com/wiki/

Subject: Re: MERCURIAL
Posted by andrei_natanael on Sun, 13 Sep 2009 19:30:12 GMT
View Forum Message <> Reply to Message

I don't think that hg is the choice for everyone. Seems to me that big projects are using git. Gnome is using it, Qt too and also KDE is thinking to switch to it (amarok made the move). See the list from git site for almost every project using git.
I wish U++ to use git too

Subject: Re: GIT
Posted by cocob on Sun, 13 Sep 2009 20:10:09 GMT
View Forum Message <> Reply to Message

is u++ used for big projects ??? no but it is a great tool ! This is not a argument.

cocob

Subject: Re: GIT
Posted by Didier on Sun, 13 Sep 2009 21:31:51 GMT
View Forum Message <> Reply to Message

Quote:I don't think that hg is the choice for everyone. Seems to me that big projects are using git.

Gnome is using it, Qt too and also KDE is thinking to switch to it (amarok made the move). See the list from git site for almost every project using git.
I wish U++ to use git too

Open Solaris, Open JDK, Mozilla, Python, Xen, ... I think are big enough projects.

Anyway it seems that GIT has done some progress on documentation since the last time I looked at it.

GIT and mercurial have exactly the same syntax for the main commands, but GIT seems more complicated for the other ones: that is what is usually said about GIT: complicated commands (at least what I read last year).

Anyway the point is to figure out what peaple will, and wan't to use. It seems many upp users work with windows so good windows integration is requested (I haven't tested the GIT Windows GUI).
Documentation is a central point: Mercurial has a complete, and yet simple documentation as well as many interesting plugins.
Simplicity of use is also a primary point: it seems that GIT and HG are just as easy to use (at least for the main commands)

I'm not saying GIT is better or worst than HG, I'm only saying it's worth looking closely at it before deciding to use GIT.

Subject: Re: GIT
Posted by andrei_natanael on Mon, 14 Sep 2009 08:45:40 GMT
View Forum Message <> Reply to Message

cocob wrote on Sun, 13 September 2009 23:10is u++ used for big projects ??? no but it is a great tool ! This is not a argument.

cocob
Well, don't get me wrong. I have nothing against mercurial, i really like it because it's written in C/C++ and Python and it's not written in a mixture of languages like git(C, shell script, perls, tcl, python, etc.). It have the same features as git and and proves to be better supported on different platforms.

Didier wrote
Anyway the point is to figure out what peaple will, and wan't to use. It seems many upp users work with windows so good windows integration is requested

Both scm have svn integration so it's possible now to use both in development of U++, the only missing part is theIDE integration(using hg and git from theIDE).
I really like to see U++ going distributed using hg or git. Right now hg is better supported on Windows than git so it's by far the right choice to use it if Mirek think ever to move to a distributed

scm.
The only thing remaining is the git popularity vs hg. Do we care about that?

Subject: Re: GIT
Posted by koldo on Mon, 14 Sep 2009 14:36:40 GMT
View Forum Message <> Reply to Message

Hello all

I use Mercurial (hg) as user because it is used in Eigen math library.

It is so easy that I use it through the command line.

Best regards
Koldo

Subject: Re: GIT
Posted by mirek on Mon, 14 Sep 2009 20:10:10 GMT
View Forum Message <> Reply to Message

cocob wrote on Sun, 13 September 2009 16:10is u++ used for big projects ???

Define "big"....

Mirek

Subject: Re: GIT
Posted by sapiency on Tue, 15 Sep 2009 12:01:25 GMT
View Forum Message <> Reply to Message

nevertheless,

but what about the idea to support git from theide?

reinhard

Subject: Re: GIT
Posted by andrei_natanael on Sat, 10 Oct 2009 13:41:09 GMT
View Forum Message <> Reply to Message

Hi,

I've created a upp mirror on gitorious.org so everyone who want to use git for upp development may clone the repo from there (I'll try to keep it in sync with upp svn mirror).
Why?
Doing programs sometimes require to modify some packages from upp and cannot push them to upstream, so it's simple to keep them in a separate branch and git make branch merge and creation more simple than svn. Also the cloning of git repository take few seconds compared to svn checkout which take minutes. If i want to create a program with some customized code based on upp upstream code is simple to create a branch and work on it.

Quote:nevertheless,

but what about the idea to support git from theide?

reinhard

I would like to see that too

Subject: Re: GIT
Posted by andrei_natanael on Sat, 10 Oct 2009 13:45:28 GMT
View Forum Message <> Reply to Message

I forget the link
http://gitorious.org/upp-mirror

Subject: GIT essentials
Posted by kohait00 on Mon, 11 Jan 2010 11:00:35 GMT
View Forum Message <> Reply to Message

hi people,

i have been using svn in my company now for some time and it was ok, sometiimes it was a pain. and one time we even had a crash on the server, boom, the repo was gone, the history too. my fault, though. should have done backup before. well, life continued with the revision that was on my work laptop. with git, that cant happen.

the trunk/branches/tags model in svn (which is not obligatory though, but used widely) is kinda strict in a sense. it does not motivate a developper much to try and experiment with the codebase, because one fears the whole commit/revert/reread and filter the changes stuff.. and you always need the repo connection for looking up the history, the graph etc.. with git that is far esier.

admittable, git has a hard learning curve in the beginning, since the concepts are remarkably different from svn, so svn users espacially have their what?s and how?s at the beginning. but one could simply present them some tutorials and explain the basic things, so they get the picture.

the fact that everyone has its own repository, and mirek would need to pull is both an advantage

and a disadvantage. he would not need to care much about commit right of other users, because *HE* needs to decide from whom to pull or not, this is all. he does not need to give anyone the commit right to its own git repo at all, basicly. so user management is franly different. mirek could be able try out some ideas and reject them without comment or polluting its own repo master branch. having a bunch of remote branches from *trusted* co contributers would make it easy to keep track. but probably this would make more work for him (more decision work especially), whereas now, he grants commit right to some users, trusting them not to pollute the trunk and does not need to take care of it anymore (but this is also possible in git). but if we consider that linus has as little time as mirek and manages to handle kernel development with git (it actually was developped for that purpose), then it should be even easier.

i'd personally prefer to switch from svn to git, it is way superior to svn and many other scms (as far as i understood) and it should even be prefered over mercurial and other commercial projects, cause git will have a much wider accaptance and be wider used in future due to its open source nature. one can even say its more *standard* now. (look at all the major projects one by one dropping svn and heading for git).

we clearly have to destinct 2 things concerning usage of git for u++:

1) usage of git as scm for the Ultimate++ project *itself*
2) providing a git interface (among others, just like svn) in TheIDE to sync the user's project codebase when he uses git for it (has nothing to do with the ultimate++ project itself at all)

i think that was not clear cut apart in the preceeding posts

beeing using svn for quite some time, i started to switching to git, and still learning, have strived some of the tutrials and infos on the net over git. and so far, became more and more fixed in my decision to stay with git.

hope U++ will consider to move to git too. sorry for the long post

PS: Linus, the godfather of GIT making a really enjoyable talk about GIT itself, at google talk, focusing and adressing most of the topics we descuss here. is worth watching, makes some harsh jokes . this video made me think about switching. is quite long, but woth it.
http://www.youtube.com/watch?v=4XpnKHJAok8

Subject: Re: GIT essentials
Posted by andrei_natanael on Mon, 11 Jan 2010 15:07:11 GMT
View Forum Message <> Reply to Message

I would like too that U++ use git

When I've made a full checkout last time I've measured checkout time for subversion and clone time for git. I've used my U++ git clone repo from gitorious (it's not updated since 6 December, 2009) and the result's are quite impressive.

svn
real 37m23.956s
user 0m49.850s
sys 0m29.460s

git
real 1m27.702s
user 0m10.170s
sys 0m2.400s

The clone of U++ repository take only 1 minute and half compared to more than half hour for svn.

It's simply when you work on something to create a new branch, work on that and if you have right results merge it to master else drop it. Branching is more easier than with svn. I didn't done a svn branch merging but I've heard that's pure pain.

Here comes another advantage. If we use git as scm it's simple to use the same nest(let's say uppsrc) and test other changes because git branches use the same location but modify content of files related to current branch.

IMO git is just great (I'm mainly on Linux, but on Windows should not be that much trouble using it) and switching to git will be a step forward for U++.

Andrei

P.S.: If you want to test git, you may use my mirror from gitorious.org, even if it's not updated i think it will help you take a decision to use git or not.

Subject: Re: GIT essentials
Posted by mr_ped on Mon, 11 Jan 2010 15:13:50 GMT
View Forum Message <> Reply to Message

those 30+ minutes... was that from (internal) main UPP SVN or from google mirror? The internal SVN is running behind ordinary ADSL line, so the upload is horrible. If you want fair comparison, you must time checkout from google's SVN. (I still bet the GIT would win, but I don't think the difference will be so huge)

Subject: Re: GIT essentials
Posted by andrei_natanael on Mon, 11 Jan 2010 15:32:46 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Mon, 11 January 2010 17:13those 30+ minutes... was that from (internal) main UPP SVN or from google mirror? The internal SVN is running behind ordinary ADSL line, so the upload is horrible. If you want fair comparison, you must time checkout from google's SVN. (I still

bet the GIT would win, but I don't think the difference will be so huge)
Yes, you're right. It was main UPP SVN. Anyway i think git offer many more advantages than svn and you could easily mimic svn work-flow with git.

Subject: Re: GIT essentials
Posted by andrei_natanael on Mon, 11 Jan 2010 15:42:13 GMT
View Forum Message <> Reply to Message

I've tested with googlecode.

svn
real 5m0.305s
user 0m25.060s
sys 0m10.330s

Anyway i think it's not that much because we don't to a full checkout too often perhaps only when local repo get compromised or we checkout on a new system.

http://git.or.cz/gitwiki/GitSvnComparsion

Subject: Re: GIT essentials
Posted by mr_ped on Mon, 11 Jan 2010 16:19:35 GMT
View Forum Message <> Reply to Message

Well, the problem is that cloning(branching) with GIT is cheap, while with SVN it will take those 5+min. That's not good.
Then again as almost all U++ devs have limited access to SVN, and only very few people change the core of it (most of them used to old non-SVN way of syncing), they rarely run into conflicts or need of branches. Rest of us work mostly in Bazaar, which are reasonably small projects to rarely need a branch during development, and even then we would more likely create package_2 in bazaar in the very same trunk, mimicking branch behavior.
So I don't think right now there's huge benefit from using GIT.

If Mirek and other core developers have some spare time, it would probably make sense to try it out. If U++ core contributors will grow more in the future, the benefit from switch may be even greater. But right now I personally don't see this as big priority (then again I'm not core dev and rarely contributing even into bazaar, so who cares what I think).

Also I think if you are proposing this, you should maybe try also design the work-flow model, i.e. how Mirek will remain the master of U++, yet other contributors will submit patches with GIT to him. So once he will want to try it out, he can read some ideas how it should work in U++ community. (because GIT allows many ways of cooperation)

What's puzzling *me* as non-user of git is the "pull". I understand anyone can clone repo, do his

changes, prepare public commit (patch) and tell maintainer of project it's really great improvement and he should adopt it. Then comes the "pull" by maintainer from the contributor's repo? So everyone's personal repository has to be on public IP? (I find this unlikely with current U++ contributors)
I know this can be worked around by submitting patches for example trough e-mail, or by letting contributor to instead push into central repo, I'm just asking if I understand this part correctly. I think for really democratic/decentralized development (I'm NOT saying the U++ needs this, I think current way works quite ok right now, maybe in future we will need change, but not yet?) everyone should have his own repo and basically Mirek as site owner will use very likely his own repo to create official distribution, so if he does like something from somebody else, he will pull it and add to official U++. I wonder how well that works in current age of IPv4 and NATs. Maybe I misunderstood something important about DVCS/GIT?

Subject: Re: GIT essentials
Posted by andrei_natanael on Mon, 11 Jan 2010 16:59:32 GMT
View Forum Message <> Reply to Message

mr_ped wrote Well, the problem is that cloning(branching) with GIT is cheap, while with SVN it will take those 5+min. That's not good.

Actually branching in git takes almost not time.
real 0m0.044s
user 0m0.000s
sys 0m0.010s

Cloning the main repository takes 1 minute and equivalent of clone is checkout in svn I guess(in git checkout is a different operation).
mr_ped wrote
Also I think if you are proposing this, you should maybe try also design the work-flow model, i.e. how Mirek will remain the master of U++, yet other contributors will submit patches with GIT to him. So once he will want to try it out, he can read some ideas how it should work in U++ community. (because GIT allows many ways of cooperation)

Mirek will remain the master of U++. We may follow Linus work-flow, Mirek as dictator and perhaps if U++ will grow more other persons as lieutenants. Right now Mirek being single person who will decide what goes in repo is the right choice.
mr_ped wrote
What's puzzling *me* as non-user of git is the "pull". I understand anyone can clone repo, do his changes, prepare public commit (patch) and tell maintainer of project it's really great improvement and he should adopt it. Then comes the "pull" by maintainer from the contributor's repo? So everyone's personal repository has to be on public IP? (I find this unlikely with current U++ contributors)
I know this can be worked around by submitting patches for example trough e-mail, or by letting contributor to instead push into central repo, I'm just asking if I understand this part correctly. [...] Maybe I misunderstood something important about DVCS/GIT?
You shouldn't have a public IP, for that exists sites like gitorious, googlecode, sourceforge. If someone wants to let Mirek see his changes he will push changes to his own repo from one of

those sites and send to Mirek a link to it.
Perhaps if we take a realistic situation it will be more easier to understand that. I have a clone of U++ at gitorious, and also a local clone of if. If i will make changes to local clone and merge changes with the master branch from gitorious my repo will diverge from current U++ repo (supposing it's using git). Now i may send to Mirek a link to my repo, he will clone it (or only clone last revision) and if he like it he will merge mine repo with U++ main repo. I don't have to tell to anyone my IP (my IP is dynamic anyway and telling one will have no effect next time because it may get changed). No need to use a central repo, no emailed patches.

I don't understand yet very well git but as i understand now it makes more sense to me to use it than using svn. If I'll make changes to Chameleon with git i have to do simple operations:
git branch cham-changes
git checkout cham-changes
// change the code

git commit -a
git checkout master
// if i'm glad with changes

git merge cham-changes
git commit -a
git branch -d cham-changes

// if i'm not pleased with changes

git branch -D cham-changes

With svn i'll have to change the files... if i dislike the changes... revert them... in the meantime i cannot work on something else because the repo is blocked by my changes to Chameleon. To solve this i have to use branches in svn too, it take time and i don't know how hard is to merge them... perhaps someone with merging experience will tell me. In git merging is so simple .

Subject: Re: GIT essentials
Posted by Novo on Tue, 12 Jan 2010 05:07:00 GMT
View Forum Message <> Reply to Message

Just my two cents.

A couple years ago I had the same problem: my CVS server broke down and I lost one year history. Fortunately, I had backups from previous years.

After a short research I chose "monotone".

Cons:
1) It is just one executable (plus a couple of dlls). You don't really need to install it.
2) It is quite powerful. Actually, GIT borrowed a lot of ideas from monothone.
3) It is easy to use. Monotone has not that big set of commands.

4) Repository and workspace can be located in different places. (You can have different workplaces for the same branch. Actually, I'm using this for a different purpose.)

Pros:
1) No usable GUI. Nothing similar to TortoiseSVN or TortoiseGIT.
2) Not very popular. Actually, the same can be told about U++. Though, that doesn't make it bad.

The only thing I'm missing about monotone is GUI, which can be easily developed using U++, because monotone stores repository in SQLITE database.

A week ago I tried to migrate from monotone to GIT because of lack of GUI. That didn't happen because I couldn't resolve several work-flow issues.

1) GIT keeps repository and workspace in the same directory. In order to checkout another branch you need to clone repository. Basically, this means that you cannot store several projects in one repository. This is not a problem with monotone. In my case "branch" is often equal to "project".
2) U++ is a set of "packages". When I create a new project I want to assemble it from several "packages". This is possible with GIT, but in this case each package should be represented as a separate GIT repository. This seems to be way too complicated.

I'd like to get more feedback about work-flows with distributed VCS from you guys.

TIA

Subject: Re: GIT essentials
Posted by cbpporter on Tue, 12 Jan 2010 08:18:45 GMT
View Forum Message <> Reply to Message

Are you sure about "Cons" and "Pros"?

Subject: Re: GIT essentials
Posted by andrei_natanael on Tue, 12 Jan 2010 09:58:25 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 12 January 2010 07:07
After a short research I chose "monotone".

Pros:
1) It is just one executable (plus a couple of dlls). You don't really need to install it.
2) It is quite powerful. Actually, GIT borrowed a lot of ideas from monothone.
3) It is easy to use. Monotone has not that big set of commands.
4) Repository and workspace can be located in different places. (You can have different workplaces for the same branch. Actually, I'm using this for a different purpose.)

I like your writing mistake about pros and cons, IMO your pros are cons .

1) How extensible is it? If we want to create a hook for some operation or replace one part with our i.e. diff?
2) ... (the counterparts are too) ...
3) Easy to use not always means less commands, monotone stay in my way because it's doing simple things complicated. I've tried once to help pidgin development and implement some features which i needed but I've hit monotone wall and I quited. A scm should not stay in your way, it should be expressive enough not to impose limits but not too expressive to let you doing same thing in thousand different ways.
4) IMO it just complicate my life

Novo wrote
The only thing I'm missing about monotone is GUI, which can be easily developed using U++, because monotone stores repository in SQLITE database.

While others want a GUI on top of scm tool I want it to be integrated with my tools. I.e. I create project in theIDE (or other ide), i'm adding new source files, do some codding, then i initialize a repository (from theIDE) and do initial commit. Supposing that i have a working program and want to test new stuff. I'm creating a branch (from theIDE AnySCM->NewBranch), theIDE reloads all opened files and these contain data from new branch, note that i've not changed project location, so the branch reside in same directory and i may switch to other branch if i want and that's cool because i don't have to create a different package in theIDE just to try new things (i'm reusing the same interface, just switching from one branch to another). I'm doing some modification to this branch, i'm testing changes and if it's ok i'm merging it in master branch (AnySCM->Merge<branch-name>).
IMO that's a nice way to get stuff done.
Novo wrote
1) GIT keeps repository and workspace in the same directory. In order to checkout another branch you need to clone repository. Basically, this means that you cannot store several projects in one repository. This is not a problem with monotone. In my case "branch" is often equal to "project".
2) U++ is a set of "packages". When I create a new project I want to assemble it from several "packages". This is possible with GIT, but in this case each package should be represented as a separate GIT repository. This seems to be way too complicated.

I'd like to get more feedback about work-flows with distributed VCS from you guys.

1) I think you don't have to clone the repo to checkout another branch. You do the checkout and reuse the same space and if you want to switch to master branch just checkout it.

```
~$ git branch
  master *
  your-cool-branch
~$ git checkout your-cool-branch
# now where your project is, you have your-cool-branch source code in place
# working...
~$ git commit -a
~$ git checkout master
# yay, we are on master branch on the same location
# and i think that's cool
```

2) Not every package should be a separate repository, i would make assembly(nest?) a repository, perhaps only for MyApps i wouldn't do that because packages from it may not be related one to other.

Just my 2 cents.

I think we should consider others dvcs for scm but IMO the leading scm which will be used for a long time from now will be git, hg and bzr.
Git - surfer from code fragmentation(shell code, perl, python, C, tcl, etc.), making it a bit unportable.
Bzr - a bit slow, it's written in python but cross-platform is achieved.
Hg - i don't have an opinion, i've used it only once (tried the tutorial) so no real life experience with it.

Subject: Re: GIT essentials
Posted by kohait00 on Tue, 12 Jan 2010 10:36:47 GMT
View Forum Message <> Reply to Message

the basic question is indeed, as somone above mentioned: does it matter to migrate *right now*? the workflow seems to be alright for now, but could significantly change in future.

it s really the question of workflow models, so far nobody can imagine a truely practicable solution. maybe it's due to the fact that git *is* different from svn, it stores repo/history locally and completely (if you clone, or partially if you pull). as far as everyone can pull from anyother, there is a different user handling model.

now, as far as i understand, there is almost the dictator/leutnant model, using svn. mirek beeing master commiter, having some trusties, who also have commit rights, but use them with care (paying head and pants to mirek if things go wrong), so this is bit of security issue. people contributing and lacking commit rights do this by supplying test projects in forum, or like me most times, just some lines in plain text with the file:line pattern so mirek needs to sit down, search the file, *think*, change the line and commit. he is actually a busy master/dictator.

with git this could go just as before, but somehow ease it for mirek. he could have his own git repo, the "MAIN" repo, creating releases from, often times refered as the "blessed repository", publically readable, but commit rights only for mirek.
his leutnants, just as the rest of the world, keep synced with the blesses repo, always pulling from it (as the origin repo, in git terms).
the leutnants are just helpers in u++ case, and either have an own and somehow accessable repo for mirek, where *he* can pull from in his branches, and then he can merge. or it would be easier for now, if they also had comit rights to the blessed repo as well, so this would be just the same situation as before with svn.
he would mainly benefit from the easyness of branch merging with git.

when we expect u++ users to supply patches, just as in kernel issues, this would be too much. they are not that experienced, and neither that engaged with u++ . so even the workflow with

forum bugfix posting and contributions would go on for quite a while just the same, as long as we dont provide some better means for contributions, i.e. a howto setup own public git repo, where a engaged contributer can commit his changes and mirek pull from.

so its mainly a ideologic dessision for now, it *would* have some benefits, but they are not that extraweighted as to justify a migration to git. i personally would still prefer git, for even more but presently for u++ unimportant reasons. maybe just one still to mention: it seems not to be that much fixed in structure, is really flexible, where every developper is responsable for itself. this is not the case with svn, where 'anyone' can pollute the repo.

attached is a one popular model of how things are drive, bold arrows is commit, other is pull. all are repos, either private on working machine or public on github or gitorious.

if we really want mirekt to be convinced, we need to more clearly elaborate a better workflow model, though. so far, he'd understandably not see the need to migrate.

File Attachments

1) gitmodel.JPG, downloaded 314 times

Subject: Re: GIT essentials
Posted by Didier on Tue, 12 Jan 2010 20:28:55 GMT
View Forum Message <> Reply to Message

Hi I've been using HG for more than two years now and it works perfectly: fast and easy to use.
It's available under:

1. Windows
2. Mac OS X
3. Linux (.deb)
4. Linux (.rpm)
5. Linux (others)
6. Solaris
7. AIX
8. BSD

The base commands of HG are very simple and it's almost impossible to do "bad" things you can always rollback if last action was wrong.

The main GUI, TortoiseHG, has done lots of progress, is very good now and is the same under linux or windows (I haven't tried on other OSs).

I have no experience with GIT so I won't say anything about it, but does it have good GUI's ?

Anyway before changing from svn to another SCM people can try HG, and probably GIT or another SCM, by using it over SVN:

Currently I use an HG repository for the UPP code.
I update it directly from the UPP SVN
repository by using the hgpullsvn utility.
The pending hgpushsvn utility also exists.

So I use HG locally while the repo is still under SVN.

This approach could be a good test before deciding SCM A or B

Subject: Re: GIT essentials
Posted by Novo on Wed, 13 Jan 2010 04:36:37 GMT
View Forum Message <> Reply to Message

Didier wrote on Tue, 12 January 2010 15:28 Currently I use an HG repository for the UPP code.
I update it directly from the UPP SVN
repository by using the hgpullsvn utility.
The pending hgpushsvn utility also exists.

So I use HG locally while the repo is still under SVN.

This approach could be a good test before deciding SCM A or B

I completely agree with this. GIT can be easily synchronized with SVN. If your favorite DVCS doesn't have native synchronization support, then you can use Tailor (http://progetti.arstecnica.it/tailor).

Subject: Re: GIT essentials
Posted by Novo on Wed, 13 Jan 2010 04:49:24 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 12 January 2010 03:18Are you sure about "Cons" and "Pros"?

Sorry. I shouldn't be allowed to write anything after 12pm.

Subject: Re: GIT essentials
Posted by Novo on Wed, 13 Jan 2010 05:40:33 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Tue, 12 January 2010 04:58Novo wrote on Tue, 12 January 2010 07:07
After a short research I chose "monotone".

Pros:

1) It is just one executable (plus a couple of dlls). You don't really need to install it.
2) It is quite powerful. Actually, GIT borrowed a lot of ideas from monothone.
3) It is easy to use. Monotone has not that big set of commands.
4) Repository and workspace can be located in different places. (You can have different workplaces for the same branch. Actually, I'm using this for a different purpose.)

I like your writing mistake about pros and cons, IMO your pros are cons .
1) How extensible is it? If we want to create a hook for some operation or replace one part with our i.e. diff?
2) ... (the counterparts are too) ...
3) Easy to use not always means less commands, monotone stay in my way because it's doing simple things complicated. I've tried once to help pidgin development and implement some features which i needed but I've hit monotone wall and I quited. A scm should not stay in your way, it should be expressive enough not to impose limits but not too expressive to let you doing same thing in thousand different ways.
4) IMO it just complicate my life

1) Monotone uses Lua for scripting. It allows you to create hooks. Though, I haven't tried to do that myself.
2) I just wanted to say that they are similar.
3) Could you, please, describe the problem in more details? I'm trying to learn different workflows.
4) You need to synchronize your repository only once in this case. May be I'm wrong, but if my project is build from a dozen of independent submodules, and I want to update the code, then I need to update all these repositories manually (or to develop a script).

Quote:
Novo wrote
The only thing I'm missing about monotone is GUI, which can be easily developed using U++, because monotone stores repository in SQLITE database.

While others want a GUI on top of scm tool I want it to be integrated with my tools. I.e. I create project in theIDE (or other ide), i'm adding new source files, do some codding, then i initialize a repository (from theIDE) and do initial commit. Supposing that i have a working program and want to test new stuff. I'm creating a branch (from theIDE AnySCM->NewBranch), theIDE reloads all opened files and these contain data from new branch, note that i've not changed project location, so the branch reside in same directory and i may switch to other branch if i want and that's cool because i don't have to create a different package in theIDE just to try new things (i'm reusing the same interface, just switching from one branch to another). I'm doing some modification to this branch, i'm testing changes and if it's ok i'm merging it in master branch (AnySCM->Merge<branch-name>).
IMO that's a nice way to get stuff done.

IMHO all VCS can be integrated with TheIDE. IMHO all DVCS let you use the same directory to checkout different branches. Not all DVCS let you checkout into multiple places. And DVCS's do not let you do partial checkout. You get all or nothing.

I personally need GUI to be able to browse big logs, and to be able easily compare different versions of a file. It is hard for me to type long SHA1 keys. I'm not using mouse.

Quote:
Novo wrote
1) GIT keeps repository and workspace in the same directory. In order to checkout another branch you need to clone repository. Basically, this means that you cannot store several projects in one repository. This is not a problem with monotone. In my case "branch" is often equal to "project".
2) U++ is a set of "packages". When I create a new project I want to assemble it from several "packages". This is possible with GIT, but in this case each package should be represented as a separate GIT repository. This seems to be way too complicated.

I'd like to get more feedback about work-flows with distributed VCS from you guys.

1) I think you don't have to clone the repo to checkout another branch. You do the checkout and reuse the same space and if you want to switch to master branch just checkout it.

```
~$ git branch
  master *
  your-cool-branch
~$ git checkout your-cool-branch
# now where your project is, you have your-cool-branch source code in place
# working...
~$ git commit -a
~$ git checkout master
# yay, we are on master branch on the same location
# and i think that's cool
```

2) Not every package should be a separate repository, i would make assembly(nest?) a repository, perhaps only for MyApps i wouldn't do that because packages from it may not be related one to other.

Just my 2 cents.

1) Yes, GIT is designed to use only one directory and work with only one project. It is a fixed workflow. Actually, the way you use GIT in your example is not exactly a GIT way. You completely bypass Index. Index is a nice feature to get rid of temporary branches.

2) I personally have ~30 packages in my MyApps. I do not want to checkout this mess all the time. I want to checkout a project and get only necessary packages. And when I update my project, I want these packages get updated. But in case when I use an external code, I do not want it to be updated all the time. I want to "rebase" it manually.

Probably, I want too much.

This all is about workflows.

I personally trying to figure out which DVCS i need to use (and how to use it) in case of complicated code structure, many projects, many external projects, many versions of same code, plus U++, plus e.t.c.

All your suggestions are welcome. I appreciate your feedback.

Subject: Re: GIT essentials
Posted by kohait00 on Wed, 13 Jan 2010 08:47:34 GMT
View Forum Message <> Reply to Message

concerning the GUI parts of git (with usability in mind when thinking of switching u++ project repo to git):

The Git GUI is quite usefull, supporting most of the command line features. a nice history few (one needs to get familiar with the view though, its due to different model than svn graph)

furthermore, there is TortoiseGIT for windows, which is in appearance relative to TortoiseSVN. it seems quite same in usage, though i havent tried it extensively.

concerning integration of git in TheIDE, just like SlikSVN, one can use the well defined command line operations in cygwin environment, though this is not trivial, i think.

Subject: GIT tryout
Posted by kohait00 on Wed, 13 Jan 2010 13:47:26 GMT
View Forum Message <> Reply to Message

for all who want to play around with git and u++:
in a post above, there was a link, but is not quite current anymore.

download and install git for windows i.e.
http://msysgit.googlecode.com/files/Git-1.6.5.1-preview20091022.exe
when installing, choose with the radio buttons, to modify your PATH, should make no problems.
and choose to commit lineendings as is.

then from git bash:

```
mkdir uppgit
cd uppgit
git svn clone -s http://upp-mirror.googlecode.com/svn .
```

takes around 4 hours to import the whole history.
total size 260 MB.

first time starting the GitGUI and opening the repo sinals 7000 loose objects, just allow to

compress, and its gone, leaving you with a nice, clean clone of uppsvn.

normally, one can run

git svn fetch

or something like that, to keep in touch with svn repo, and
git svn rebase
to apply the changes (dont sure)

from git help:

"...After a repository is cloned, the fetch command will be able to update revisions without affecting the working tree; and the rebase command will be able to update the working tree with the latest changes. ..."

enjoy

Subject: Re: GIT tryout
Posted by Didier on Wed, 13 Jan 2010 21:55:01 GMT
View Forum Message <> Reply to Message

Hi Novo,

Quote:This all is about workflows.

I personally trying to figure out which DVCS i need to use (and how to use it) in case of complicated code structure, many projects, many external projects, many versions of same code, plus U++, plus e.t.c.

All your suggestions are welcome. I appreciate your feedback.

Well in HG there there are lot's of extensions
http://mercurial.selenic.com/wiki/UsingExtensions#Extensions _Bundled_with_Mercurial

And in particular the Forest extension which allows to manage sub repositories inside a global repo: maybe this sounds a bit like what you need.
This plugin is still in beta state but it's very promising and it would fit very well to the Upp package organisation Or at least in bazaar

---

Subject: Re: GIT tryout
Posted by andrei_natanael on Thu, 14 Jan 2010 00:52:17 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 13 January 2010 15:47for all who want to play around with git and u++: in a post above, there was a link, but is not quite current anymore.

If you're talking about http://gitorious.org/upp-mirror it wasn't updated since 6 December because when i've reinstalled the system i've lost my ssh key, now I have backup for it and the repository will get updated every time I'm starting my laptop (that happen often). Also i've changed the user too, that's why is repo only 2 hours old. I think should be safe to use the repo from now on as I'm doing my graduation thesis with U++ and i'm using git for scm.

kohait00 wrotetakes around 4 hours to import the whole history.
I don't know what have you measured but it took me only half an hour to import whole history from upp-mirror please don't scare new users

Andrei

Subject: Re: GIT tryout
Posted by kohait00 on Thu, 14 Jan 2010 11:19:09 GMT
View Forum Message <> Reply to Message

thank you for your efforts,

well, i started at 8 am and was done at 11:40 am or something, might have been due to slow connection also, or mi laptop is kinda slow, a lot of maybes, but it was that long, nice to hear it can go faster.

Subject: Re: GIT tryout
Posted by Novo on Sun, 17 Jan 2010 19:25:15 GMT
View Forum Message <> Reply to Message

Didier wrote on Wed, 13 January 2010 16:55Hi Novo,

Quote:This all is about workflows.

I personally trying to figure out which DVCS i need to use (and how to use it) in case of complicated code structure, many projects, many external projects, many versions of same code, plus U++, plus e.t.c.

All your suggestions are welcome. I appreciate your feedback.

Well in HG there there are lot's of extensions
http://mercurial.selenic.com/wiki/UsingExtensions#Extensions _Bundled_with_Mercurial

And in particular the Forest extension which allows to manage sub repositories inside a global

repo: maybe this sounds a bit like what you need.
This plugin is still in beta state but it's very promising and it would fit very well to the Upp package organisation Or at least in bazaar

Hi Didier,

Thanks for pointing out. It looks like Mercurial has a build in support for sub-projects since v 1.3.

I checked sub-project support in both Git and Mercurial. In both cases it seems to be somewhat confusing to me. There are many "don't do that" cases in both systems. Right now my preferences lean to Mercurial. I'll keep experimenting.

Below are links for those, who is looking for a new distributed VCS.

A Guide to Branching in Mercurial (pretty cool analysis, comments are also cool) -
http://stevelosh.com/blog/2009/08/a-guide-to-branching-in-me rcurial/

Git workflows:
http://osteele.com/archives/2008/05/my-git-workflow
http://osteele.com/archives/2008/05/commit-policies

Subject: Re: GIT tryout
Posted by andrei_natanael on Sun, 17 Jan 2010 22:37:47 GMT
View Forum Message <> Reply to Message

With all this movement around DVCS in U++ forum, i think we have to start making our own DVCS, named Sit (Simple Information Tracker or Stay In Touch [with the code, of course]), sure it may not be wide used but anyway it's made by us for us, anyone for reviving UVS? Just kidding.

Andrei

Subject: Re: GIT tryout
Posted by Novo on Mon, 18 Jan 2010 00:21:21 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Sun, 17 January 2010 17:37With all this movement around DVCS in U++ forum, i think we have to start making our own DVCS, named Sit (Simple Information Tracker or Stay In Touch [with the code, of course]), sure it may not be wide used but anyway it's made by us for us, anyone for reviving UVS? Just kidding.

Andrei

Actually, I would try to reimplement monotone using U++. Database needs refactoring (text sha1

shouldn't be used within primary keys). As a result of reimplementation size of a binary should shrink from 9M to 1.5M. Unfortunately, this task doesn't have highest priority for me. For the next half a year I'm going to stick to the old-fashioned monotone.

Subject: Re: GIT tryout
Posted by andrei_natanael on Sun, 24 Jan 2010 11:53:48 GMT
View Forum Message <> Reply to Message

Just new links about git, and yes, you should read this
http://hades.name/blog/2010/01/17/git-your-friend-not-foe/
 http://hades.name/blog/2010/01/22/git-your-friend-not-foe-vo l-2-branches/

Subject: Re: GIT tryout
Posted by Novo on Mon, 25 Jan 2010 04:32:14 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Sun, 24 January 2010 06:53Just new links about git, and yes, you should read this
http://hades.name/blog/2010/01/17/git-your-friend-not-foe/
  http://hades.name/blog/2010/01/22/git-your-friend-not-foe-vo l-2-branches/

I like that:

"it is confusing ..."
"It will simply destroy the remote history ..."

And this is in an introduction for the beginners ...

Think several times before you start using this system.

Subject: Re: GIT tryout
Posted by andrei_natanael on Mon, 25 Jan 2010 09:37:53 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 25 January 2010 06:32andrei_natanael wrote on Sun, 24 January 2010 06:53Just new links about git, and yes, you should read this
http://hades.name/blog/2010/01/17/git-your-friend-not-foe/
   http://hades.name/blog/2010/01/22/git-your-friend-not-foe-vo l-2-branches/

I like that:

"it is confusing ..."
"It will simply destroy the remote history ..."

And this is in an introduction for the beginners ...

Think several times before you start using this system.

With a great power come a great responsibility. Right now the parts which i like at a DVCS are:

cloning speed
it's easy to create a branch and merge with another
new branch use the same place (i really like that because i'm not forced to create different assemblies in theide for different branches)

For me right now doesn't matter which DVCS but it have to have these features and probable more as i need them. I know that hg have better cross-platform implementation (because python) but i'm using git right now because it's supported better than others on some sites (gitorious, github) and there are many projects using it. I want a tool that doesn't stay in my way (frankly that happen with git sometimes).

Subject: Re: GIT
Posted by mr_ped on Mon, 25 Jan 2010 09:45:27 GMT
View Forum Message <> Reply to Message

I'm trying to get the grasp with GIT too, at least reading such articles, but I have difficult times as well.

I never hit any merge problems with SVN hard (thanks to my limited usage of it mostly, and little to my understanding how to do it in proper SVN way). My biggest problem with SVN is accidentally hitting commit with unfinished log message.

That said, the git does solve very little of my problems, as you can see. I think if I would join a more OSS project with more distributed decentralized development, I would hit limits of SVN much harder and welcome GIT.
And that's probably true for future of U++, I think in the long term (10+ years) it's inevitable to further loose the connection to mirek so the U++ would survive even without him later. If there will be still a point to have C++ framework at all. But right now I feel very confused by GIT. I can see easily how Linus would be unable to work with SVN, but for the scale of project I work on, SVN fits it almost perfectly. :/

Yes, I'm sort of ranting I did expect bigger magic from GIT. And at the same time admitting it's superior. Looks like I'm crazy or momentarily confused too much.

But keep these things coming, I think at this rate I will try to use GIT in next months, after all it never hurts to learn something new, although in my case it keeps pushing the older things out of my head, so I have to choose wisely what set of technologies I do support at the same time.

Subject: Re: GIT
Posted by Didier on Mon, 25 Jan 2010 22:19:37 GMT
View Forum Message <> Reply to Message

Hi,

I started using DVCS's more than 10 years ago with RIP TEAMWARE, the Solaris (I think) version for DVCS wich doesn't exist any more (Their build environment is not maintained any more).

Since then I used Clearcase/CVS/SVN and now HG.

The main concept about DVCS that you need to catch is the cloning part:
When you clone a project you don't only get a copy, you get a LINKED copy with local history, with which you can exchange history data (push/pull) and make merges if you want.

You can also exchange changesets (==history data) with any other clone, through http for example, as long as it has the same base history.
==> this allows you to give or receive from another developer selected changesets/branch or complete history update WITHOUT disturbing the main repository.
==> this gives a handy way for a distributed team to share changes before they are actually pushed to the main repository.
And when the changes are finalised, all that has to be done is to push them up to the main repository and there will be no conflicts !
This is the other great part about DVCS, all the conflicts are solved locally.

Of course with enhancements like in HG and GIT you can do exactly the same with local branches. But even then, these branches can also be exchanged with other clones.

My experience shows me that this is the only way that allows an easy management of complex merges between many developers where history can get very complex.

Once you have tasted DVCS and really learned to use it at its full power, you cannot come back no matter which DVCS you are using !!!

Subject: Re: GIT
Posted by Novo on Tue, 26 Jan 2010 03:08:30 GMT
View Forum Message <> Reply to Message

Didier wrote on Mon, 25 January 2010 17:19
Since then I used Clearcase/CVS/SVN and now HG.

In my case it is CVS/Clearcase/SVN/Perforce and now I'm using monotone at home.

Quote:
Once you have tasted DVCS and really learned to use it at its full power, you cannot come back no matter which DVCS you are using !!!

Imagine a project, which employs several hundreds developers, and VCS has several hundred thousands revisions. You will not be able to clone a repository in this case.

Actually, I figured out how to switch to a different branch in monotone.

1) Check out a branch1:

mtn --db=database --branch=branch1 co .

2) Update to a branch branch2:

mtn --revision=h:branch2 update

So, now I know how to work with different branches using only one directory with three different DVCS: GIT, Mercurial, and Monotone.

Subject: Re: GIT
Posted by andrei_natanael on Tue, 26 Jan 2010 10:33:50 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 26 January 2010 05:08
1) Check out a branch1:

mtn --db=database --branch=branch1 co .

2) Update to a branch branch2:

mtn --revision=h:branch2 update

That's something I personally dislike at monotone, it's showing you his internals. Using it you know you're using a VCS which use a database.
Another problem i have it's about generating RSA. mtn force you to have RSA key even if you're using it's repository locally. I really don't want a RSA key for my private repository, because i'm not sowing it to anyone over the network. I think it should be optionally.
If you'll follow monotone tutorial from http://www.monotone.ca/monotone.html you'll see how simple is to deal with it as a VCS
I'm dreaming at the day when we will make usable software, the day when help manuals will disappear. Btw, i think bugs get into source becaus we don't know what we're doing and we don't know what other was doing.

Subject: Re: GIT
Posted by Novo on Wed, 27 Jan 2010 19:12:46 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Tue, 26 January 2010 05:33Novo wrote on Tue, 26 January 2010 05:08
1) Check out a branch1:

mtn --db=database --branch=branch1 co .

2) Update to a branch branch2:

mtn --revision=h:branch2 update

That's something I personally dislike at monotone, it's showing you his internals. Using it you know you're using a VCS which use a database.

You need to specify a database only once when you create a workspace. This nice feature lets you create multiple workspaces using only one repository/database.

I personally find monotone much easier to use comparing to GIT and Mercurial. Though I miss good implementation of sub-modules.

Subject: Re: GIT essentials
Posted by kohait00 on Tue, 16 Feb 2010 13:35:00 GMT
View Forum Message <> Reply to Message

[quote title=Novo wrote on Wed, 13 January 2010 06:40][quote title=andrei_natanael wrote on Tue, 12 January 2010 04:58]Novo wrote on Tue, 12 January 2010 07:07
After a short research I chose "monotone".

2) I personally have ~30 packages in my MyApps. I do not want to checkout this mess all the time. I want to checkout a project and get only necessary packages. And when I update my project, I want these packages get updated. But in case when I use an external code, I do not want it to be updated all the time. I want to "rebase" it manually.

this is indeed a major issue with git. you cant say to one of your projects in your git repo "update to a certain revision", while leaving the rest of your packages beeing up to head revision i.e. thats because git is trating the entire base directory as content as mentionend by someone. you could check out earlier commits, but that would probably also change the state of the other packages, f they had been worked on meanwhile. in that case, svn is usefull actually..

i'm not that experienced in git to provide a workaround to this. probably the solution lies in an inteligent branching workflow. so one could checkout branch and rebase there some stuff needed

from other packages or the like.

but since git is cheap, one could use a repo for each project as well. this is the apporach Xorg is doing, if you take a look inside the buildscript, the pull a bunch of repos, not just one.

i still believe, git is to be favored above others, since it's the most promising technology, not neccesarily because its the best, but because its a de facto standard in Open Source now and is pushed there. new promising technologies like u++ can profit A LOT, just by sticking to some stadard interfaces. the time of svn is running out.

Subject: Re: GIT essentials
Posted by fudadmin on Tue, 16 Feb 2010 13:42:54 GMT
View Forum Message <> Reply to Message

could anyone explain why Uvs2 was bad? And what features it was lacking comparing to git and/or other version c. systems?

Subject: Re: GIT essentials
Posted by andrei_natanael on Tue, 16 Feb 2010 14:45:16 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Tue, 16 February 2010 15:42could anyone explain why Uvs2 was bad? And what features it was lacking comparing to git and/or other version c. systems?
I've used it only few times but if i remember correctly reasons were:
- not a wide used tool so every developers wanting to commit to U++ had to learn it and it wasn't released together with U++ or TheIDE, so one had to build it from uppbox folder... IIRC initially it was existing only on private repository
- compared to git, uvs isn't a DVCS
- i don't know if branching and merging was possible
- looked like U++ developer suffer from Not Invented Here syndrome and abandoning Uvs and choosing an external tool for SCM spread any rumors (no, it's not a reason, but we have our own IDE, our(STL) NTL, our own Image format [with hot-spots], site is made with U++ made tools)

Perhaps one should make a comparison between Uvs and SVN, GIT, BZR, etc. maybe we get back to it

Andrei

Subject: Re: GIT essentials
Posted by fudadmin on Tue, 16 Feb 2010 15:52:02 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Tue, 16 February 2010 14:45fudadmin wrote on Tue, 16 February 2010 15:42could anyone explain why Uvs2 was bad? And what features it was lacking comparing to git

and/or other version c. systems?

- looked like U++ developer suffer from Not Invented Here syndrome and abandoning Uvs and choosing an external tool for SCM spread any rumors (no, it's not a reason, but we have our own IDE, our(STL) NTL, our own Image format [with hot-spots], site is made with U++ made tools)

Andrei

So, Linus can suffer from this sindrom, we not???

Subject: Re: GIT essentials
Posted by fudadmin on Tue, 16 Feb 2010 16:34:12 GMT
View Forum Message <> Reply to Message

As I understand, Uvs2 is distributed.

Quote From wikipedia:

1. There may be many "central" repositories.
2. Code from disparate repositories are merged based on a web of trust, i.e., historical merit or quality of changes.
3. Lieutenants are project members who have the power to dynamically decide which branches to merge.
4. Network is not involved in most operations.
5. A separate set of "sync" operations are available for committing or receiving changes with remote repositories.

P.S. If a mankind hadn't made and/or used something better those who doesn't suffer from the syndrom would be still living on trees...

Subject: Re: GIT essentials
Posted by andrei_natanael on Tue, 16 Feb 2010 21:54:18 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Tue, 16 February 2010 18:34As I understand, Uvs2 is distributed.

Quote From wikipedia:

1. There may be many "central" repositories.
2. Code from disparate repositories are merged based on a web of trust, i.e., historical merit or quality of changes.
3. Lieutenants are project members who have the power to dynamically decide which branches to merge.
4. Network is not involved in most operations.

5. A separate set of "sync" operations are available for committing or receiving changes with remote repositories.

P.S. If a mankind hadn't made and/or used something better those who doesn't suffer from the syndrom would be still living on trees...
Sorry about NIH syndrome, maybe i made a bad joke
Another reason is that Uvs is GUI only so people who want to use CLI cannot use it.
However, i'll try Uvs again and maybe it may gain more love from me . I have to check if all your points are applying to Uvs and maybe then we will forget about git waves... long live Uvs. Btw, last release is from last year (12/08/2009) so it's not so far. What about creating a poll "Uvs reloaded" ? That would drive Mirek crazy . From Uvs to SVN and back.

Subject: Re: GIT essentials
Posted by andrei_natanael on Tue, 16 Feb 2010 22:07:49 GMT
View Forum Message <> Reply to Message

I found another reason while playing with Uvs. It's server dependent. One could not use it only locally without a ftp server. Hmm, i think i need a tutorial for Uvs.

Subject: Re: GIT essentials
Posted by fudadmin on Tue, 16 Feb 2010 22:57:14 GMT
View Forum Message <> Reply to Message

Personally for me it's not too big problem to download and/or exchange code even via forums.

Subject: Re: GIT essentials
Posted by sergeynikitin on Wed, 17 Feb 2010 15:14:15 GMT
View Forum Message <> Reply to Message

I want to say thank you to the topic, for what interested me using GIT. It was so convenient than other version control systems for personal development.

Subject: Re: GIT essentials
Posted by kohait00 on Thu, 19 Aug 2010 06:57:52 GMT
View Forum Message <> Reply to Message

Quote:
it's not a reason, but we have our own IDE, our(STL) NTL, our own Image format [with hot-spots], site is made with U++ made tools

think of google, they also have a whole bunch of things they offer, but some things they simply use from others (keep established standards) i.e. the googlecode svn repositories, only provide a

mask for it, just as u++ does, or think of youtube / google video.

it's not only important to redesign and 'reinvent' things, but also to know what is worth keeping.

BTW: GIT
i'm using it for quite a while now, and can't without anymore i enjoy the flexibility of branching merging and handling iin general. i still imagine mirek could profit from that more than from svn.

Subject: Re: GIT essentials
Posted by kohait00 on Fri, 27 May 2011 12:45:26 GMT
View Forum Message <> Reply to Message

to warm this one up a bit more:

the last git repository of upp somewhat died
so i decided to make a new one.

https://gitorious.org/upp

now how to handle this best?

* setup a gitorious account for yourself, clone my repo in gitorious, just search for 'upp'. you will be able to commit to your clone, and send me merge requests, which so far does not make sense until it's based on svn. it'd breake history of svn, but for patch discussion it's quite nice

* for commiting to your repository, you will need to generate a ssh key pair 'ssh-keygen -t rsa' and paste the content of the pub file in the settings.

* based on your username, it will also be possible to define collaborators (those that can commit as well) to the upp original project, not only your clone.

cheers

Subject: Re: GIT essentials
Posted by kohait00 on Tue, 07 Jun 2011 10:27:28 GMT
View Forum Message <> Reply to Message

i think it would be great to make that git repository run on mireks server alongside google-mirror updater.
thats how that git could be refreshed hourly as well, and not like now every now and then (since i dont have a 24/7 running server sitting around)

i think it should be possible to even transfer ownership of that girorious repo to some more official mantainer i could provide help with the scripts for that..

what do you think?