Version control of deliverables

Asked 16 years, 3 months ago Modified 7 years, 6 months ago

Viewed 498 times

4

We need to regularly synchronize many dozens of binary files (project executables and DLLs) between many developers at several different locations, so that every developer has an up to date environment to build and test at. Due to nature of the project, updates must be done often and on-demand (overnight updates are not sufficient). This is not pretty, but we are stuck with it for a time.

We settled on using a regular version (source) control system: put everything into it as binary files, get-latest before testing and check-in updated DLL after testing.

It works fine, but a version control client has a lot of features which don't make sense for us and people occasionally get confused.

Are there any tools better suited for the task? Or may be a completely different approach?

Update:

I need to clarify that it's not a tightly integrated project - more like extensible system with a heap of "plugins", including thrid-party ones. We need to make sure those modules-plugins works nicely with recent versions of

each other and the core. Centralised build as was suggested was considered initially, but it's not an option.

version-control deployment

Share

Improve this question

Follow

edited Jun 9, 2017 at 9:53

gprathour

15.3k ● 5 ● 71 ● 92

asked Sep 12, 2008 at 8:43

ima

8,255 ● 3 ● 23 ● 19

I think your updated comment to me seems like you still could do with some kind of build process that maybe pulls in those external binaries and runs sets of tests on them etc. I think I've been in your situation before and it's hard to see CI working until you've put something in place. – Shaun Austin Sep 12, 2008 at 20:43

We have something of that kind for final testing, but it doesn't save from the need for developers to have recent libraries. Anyway, I understand the urge to point out faulty process, but the question concerns tools and techniques for versioning and distributing binary files. – ima Sep 13, 2008 at 6:48

This issue sounds awfully like what we are currently using at my company. There is even possibility that we are working for the same company. Thanks for asking this question.
– Ady Romantika Sep 16, 2008 at 5:35

6 Answers

Sorted by: Highest score (default)

4

I'd probably take a look at rsync.

Just create a .CMD file that contains the call to rsync with all the correct parameters and let people call that. rsync is very smart in deciding what part of files need to be transferred, so it'll be very fast even when large files are involved.

What rsync doesn't do though is conflict resolution (or even detection), but in the scenario you described it's more like reading from a central place which is what rsync is designed to handle.

Share Improve this answer

Follow

answered Sep 12, 2008 at 8:56

pilif

12.7k ● 5 ● 36 ● 31

Thanks, it's a possible solution. But I better like even what we have now. I'm actually impressed by performance of our version control (SourceGear) with huge binary project.
– ima Sep 12, 2008 at 10:33

3

Another option is unison

Share Improve this answer

Follow

answered Sep 16, 2008 at 5:24

user11087

292 ● 2 ● 6

Looks like, we'll give it a try. – ima Sep 16, 2008 at 13:02

1

You should look into continuous integration and having some kind of centralised build process. I can only imagine the kind of hell you're going through with your current approach.

Obviously that doesn't help with the keeping your local files in sync, but I think you have bigger problems with your process.

Share Improve this answer

Follow

answered Sep 12, 2008 at 8:55

Shaun Austin

3,822 ● 3 ● 24 ● 27

The process is awfull, agreed, but there are important legacy and practical reasons for it. Total centralization is not possible - basically, you don't want to build your core together with some flimsy interface plugin. – ima Sep 12, 2008 at 10:29

0

Building the project should be a centralized process in order to allow for better control soon your solution will be caos in the long run. Anyway here is what I'd do.

- Create the usual repositories for source files, resources, documentation, etc for each project.
- Create a repository for resources. There will be the latest binary versions for each project as well as any

required resources, files, etc. Keep a good folder structure for each project so developers can "reference" the files directly.

- Create a repository for final buidls which will hold the actual stable release. This will get the stable files, done in an automatic way (if possible) from the checked in sources. This will hold the real product, the real version for integration testing and so on.

While far from being perfect you'll be able to define well established protocols. Check in your latest dll here, generate the "real" versión from latest source here.

Share Improve this answer

Follow

answered Sep 12, 2008 at 9:46

Jorge Córdoba

52.1k ● 11 ● 82 ● 130

I edited question to explain why it's not centralized - otherwise what you described is exactly what we do now. We are not happy with how source control tools handle binary files and deployment scenarious though. – ima Sep 12, 2008 at 10:23

0

What about embedding a 'what' string in the executables and libraries. Then you can synchronise the desired list of versions with a manifest.

We tend to use CVS id strings as a part of the what string.

```
const char cvsid[] =
"@(#)INETOPS_filter_ip_$Revision: 1.9 $";
```

Entering the command

```
what filter_ip | grep INETOPS
```

returns

```
INETOPS_filter_ip_$Revision: 1.9 $
```

We do this for all deliverables so we can see if the versions in a bundle of libraries and executables match the list in a associated manifest.

HTH.

cheers,

Rob

Share Improve this answer

Follow

answered Sep 12, 2008 at 9:48

Rob Wells

37k • 13 • 84 • 147

0

Subversion handles binary files really well, is pretty fast, and scriptable. VisualSVN and TortoiseSVN make dealing with Subversion very easy too.

You could set up a folder that's checked out from Subversion with all your binary files (that all developers can push and update to) then just type "svn update" at

the command line, or use TortoiseSVN: right click on the folder, click "SVN Update" and it'll update all the files and tell you what's changed.

Share Improve this answer

Follow

answered Sep 16, 2008 at 5:19

Garo Yeriazarian

2,533 ● 18 ● 29

That's what we do now (with a different version control system, but it's better than SVN in that aspect), as described in question. – ima Sep 16, 2008 at 13:05