# Build and Source Control - Support #6863

## implement CI/CD servers and the rules for each set of tests

10/18/2022 07:01 PM - Greg Shah

| | | | |
|---|---|---|---|
| **Status:** | WIP | **Start date:** | |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Hynek Cihlar | **% Done:** | 0% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |
| **billable:** | No | **case_num:** | |
| **vendor_id:** | GCD | | |

**Description**

### History

#### #1 - 10/25/2022 05:12 AM - Dominik Zalewski

I had some experiences with various CI/CD server. Put a braindump here: Choosing_CICD_Server. Hope this helps.

#### #2 - 10/25/2022 06:22 AM - Dominik Zalewski

There is also another point worth considering and discussion. All of the CI/CD servers I worked with had some looser or tighter integration/optimizations with the VCS. Basically it boils down to the point that as soon as build jobs get triggered automatically, the number of downloads of source code repository increases. If on every build run you do the equivalent of 'bzr checkout p2j', then you transfer around 300MB, which soon will become a performance issue.

CI/CD servers have close integration with 'git' (due to it being so popular). They solve that problem by NOT issuing 'git clone' on each build, but keeping the repo cloned and just making sure that the 'diff' is downloaded with 'git pull'.

However, GCD keeps the source code in bazaar. Due to the fact that this VCS is practically deprecated, we cannot expect that there will be plugins to do similar thing as for git, but for bazaar. If GCD stays with bazaar, it probably means that on the long run one would need to code similar optimizations but for bazaar internally.

Perhaps it's worth doing a comparison table of the pros and cons of doing one vs the other:

- investing time into doing bazaar-related developer for CI/CD server integration
- investing time into migrating source code from bazaar to git

#### #3 - 10/25/2022 06:54 AM - Marian Edu

Dominik Zalewski wrote:

> However, GCD keeps the source code in bazaar. Due to the fact that this VCS is practically deprecated, we cannot expect that there will be plugins to do similar thing as for git, but for bazaar.

Not a fan of Bazaar by any means, though there seems to be a plugin available for Jenkins - [[https://plugins.jenkins.io/bazaar/]]

#### #4 - 10/25/2022 07:11 PM - Greg Shah

This is very useful.  I've read through the wiki page.  It is clear that we won't be using Bamboo. :)

As you deduced, we indeed are not going to use cloud systems for this.  We will implement our own hardware.  This is far better from a cost perspective and has significant security, privacy/confidentiality and control benefits.  Of course, the downside is the extra effort to manage the systems ourselves.  That is the "price" we pay for control.

I have been leaning toward Jenkins as a popular, open source option.  We can evaluate TeamCity as an option.

- Using open source is a big benefit.  If there are problems, we can fix them ourselves or leverage the larger relative community that exists.
- We must consider the pool of people that know how to use the solution.  The more popular, the easier it is for us to support over time.
- Technical features and integration with our environment (bzr, Linux, Ansible...).
- Annual cost.  This will have to be calculated for the scale of our usage, which will be at least one CI/CD server per customer (on their own siloed server hardware) plus some number of servers for our internal usage (I'm pretty sure the 4GL compatibility testing will be run across multiple servers).

In regard to bzr vs git, please see Why-Do-We-Use-bzr-Instead-of-git.  I don't see moving to git in the near future.  Let's plan with bzr in mind, even with all its flaws.

#### #5 - 10/27/2022 12:00 PM - Dominik Zalewski

> In regard to bzr vs git, please see Why-Do-We-Use-bzr-Instead-of-git.

May I ask when this document (the section that you refer to) has been composed? Has this been reviewed since?

#### #6 - 10/27/2022 12:50 PM - Greg Shah

Dominik Zalewski wrote:

> > In regard to bzr vs git, please see Why-Do-We-Use-bzr-Instead-of-git.
>
> May I ask when this document (the section that you refer to) has been composed? Has this been reviewed since?

Original contents came from some years ago and there have been edits since (for example the Breezy stuff was written by me, within the last year). To my knowledge it is all still accurate, but if there is something that is incorrect I'd like to know.

#### #7 - 10/27/2022 01:25 PM - Roger Borrello

Greg Shah wrote:

> Original contents came from some years ago and there have been edits since (for example the Breezy stuff was written by me, within the last year).  To my knowledge it is all still accurate, but if there is something that is incorrect I'd like to know.

One thing I will add... Breezy isn't that straightforward to get working on Windows. They don't even provide a standalone installer. I tried for about an hour on one client system yesterday... finally got it to build and run today, but that might have been because I blind squirrel can find a nut by accident. Probably a Windows thing.

#### #8 - 10/28/2022 05:08 AM - Dominik Zalewski

Greg Shah wrote:

> Original contents came from some years ago and there have been edits since (for example the Breezy stuff was written by me, within the last year).  To my knowledge it is all still accurate, but if there is something that is incorrect I'd like to know.

Thanks. I'm not sure whether what I did is not an 'overdoing' of what you requested, but here it is: https://proj.goldencode.com/projects/internal-documentation/wiki/VCS_reevaluation. If you also feel like you want to park this discussion, for some other reasons, just communicate that and I'll stop asking questions. Thanks for your patience.

#### #9 - 10/28/2022 05:10 AM - Dominik Zalewski

Roger Borrello wrote:

> One thing I will add... Breezy isn't that straightforward to get working on Windows. They don't even provide a standalone installer. I tried for about an hour on one client system yesterday... finally got it to build and run today, but that might have been because I blind squirrel can find a nut by accident. Probably a Windows thing.

Thanks for your comment. Sorry that I will ask what sounds to be 'unrelated' question, but why do you need to install VCS tools on customer's machines?

#### #10 - 10/28/2022 05:31 AM - Tijs Wickardt

I presume Roger is talking about our FWD Windows builds, which is a valid point. They must be automated (builds and tests) as well.
A few tips on bzr speed (not a fan on bzr, but also hesitant about a 'need to switch'):

1. If you use the original bzr (python2) instead of Breezy (python3): performance gain is about 1.7x . (it is common for old python2 code to become less performant after a 'plain' port to python3, since string logic goes from raw to interpreted)
2. If you use bzr co --lightweight instead of bzr co: performance gain is about 10x

Not quite as speedy and versatile as git, but it might as well be acceptable.

#### #11 - 10/28/2022 05:36 AM - Hynek Cihlar

Dominik, this is an interesting discussion. I'm a proponent of moving to git myself. There are a few things, which bug me on bzr.

- The revision numbers are moving targets. Any reference of a revision number in Redmine will become invalid after a rebase.
- Miserable tooling support: IDE integration, reviews, etc.
- Poor documentation.

#### #12 - 10/28/2022 05:51 AM - Tijs Wickardt

A few more 'pro' considerations for moving to git:

- https://www.openhub.net/repositories/compare shows bzr is dead.
- We should make it easy on our customers and future collaborators, by using tools that are recognized as a standard
- The lack of bzr maintenance is a security risk
- The lack of revision numbers, if really needed by GCD, could be tackled by git hooks (with a central authority for a sequence per branch, with temporary revision numbers in disconnected mode). It is generally regarded as an anti-pattern though. *Opinionated*: Bzr is quite schizophrenic/mps in its design, it tries to be centralized and distributed at the same time.
- **If** we make the switch, it could be more efficient to do that **before** implementing CI/CD

#### #13 - 10/28/2022 06:14 AM - Dominik Zalewski

Tijs Wickardt wrote:

> I presume Roger is talking about our FWD Windows builds, which is a valid point. They must be automated (builds and tests) as well.

I'm not sure if I got you wrong, but are you building the version of FWD on customer machines from source?

I am guessing, but correct me if I'm wrong. Your build for Windows looks like this:

- On a Windows box someone manually triggers it
- The script does a VCS checkout
- The script does a gradle build

But the truth is, that it could possibly be setup that way:

- The Build Server (Jenkins/TeamCity/etc.) builds the versioned jar for the distribution. That is the agent on the build server (Linux-based - always) does the checkout and the gradle build. The output is a jar file that get's pushed to a jar repository (nexus/etc.).

Or that way:

- The Build Server (Jenkins/TeamCity/etc.) builds the versioned docker image for the distribution. That is the agent on the build server (Linux-based - always) does the checkout, the gradle build and docker build. The output is a docker image file that get's pushed to a docker

> repository.

Either 'new' way would not require the build process to be supported on different platforms? There is of course this bit with 'gcc' where you build natively. But I presume that these components don't have the same changing patterns/frequency as the java code? These libraries could be build/published separately and just referenced from a java build. What I want to say is that if you change the 'C' code once a month and java code practically every day, maybe it's better to split the build/publication process for 'C' code and 'Java' code?

I probably don't understand your deployment procedure, as I cannot guess how later on you are using what is being built on Windows. Do you have the build/deployment process Redmine documentation, so that I can get myself up to speed?

#### #14 - 10/28/2022 06:19 AM - Tijs Wickardt

Dominik, thank you for your input. You can leave out 'my/your' build.
I'm talking about our builds (as the Golden Code company) which need to be covered by our CI/CD. The topic of this redmine issue.
That involves Linux builds and Windows builds.
FWD customers are not involved in this (at least: it's not a requirement).
I myself don't currently do Windows builds, but that's not important.

#### #15 - 10/28/2022 06:28 AM - Dominik Zalewski

Tijs Wickardt wrote:

> Dominik, thank you for your input. You can leave out 'my/your' build.
> I'm talking about our builds (as the Golden Code company) which need to be covered by our CI/CD. The topic of this redmine issue.

Thanks for correcting me. I just re-read what I've written and it's full of 'you/yours'. I think I still haven't switched my brain to thinking that now I'm internal not external and it's 'mine' as well - whatever is being 'inherited' ;)

#### #16 - 10/28/2022 08:42 AM - Roger Borrello

Just to clarify my back-story... in using SikuliX for testing the customer's application, the screen grabs and development are being performed on a Windows 10 workstation. Rather than gather my PNG and PY files and transfer them to my GCD laptop, I could save a step by working with a local repository on the customer's Windows 10 workstation.

#### #17 - 10/28/2022 09:29 AM - Greg Shah

As to why people use bzr on Windows:

Users of this open source project must be able to check out and build FWD on their systems, which might be Linux, Windows, Solaris, AIX, MacOS... whatever.  Right now we only support Linux, Windows and Solaris.

Providing access to our source repo is an important part of enabling external people/organizations to work with this project.

> https://www.openhub.net/repositories/compare shows bzr is dead.
> ...
> The lack of bzr maintenance is a security risk

The Breezy project is active, though it is admittedly not strategic. Please see my comments about this in Why-Do-We-Use-bzr-Instead-of-git. FYI, Ubuntu is a project that still uses bzr so it is not like we are the only ones.

git is a monoculture of source control systems right now, which is a security problem in a different way (it becomes a target and any flaw can be exploited everywhere).

> The lack of revision numbers, if really needed by GCD, could be tackled by git hooks (with a central authority for a sequence per branch, with temporary revision numbers in disconnected mode). It is generally regarded as an anti-pattern though.

I disagree with this idea that it is an antipattern. Revision numbers, especially the concept that they are sequential, allow one to easily and instantly reason about what is exactly in any branch. git hashes are atrocious for this. If you apply changes to a given branch in git, determining exactly what changes are in that branch will take some non-trivial effort. Even just having to reference specific changes by git has is extra work and the hash itself intrinsically does not tell you anything.

Yes, in a rebase situation the revision numbers are not stable. But once something is committed to trunk, it is completely stable and can be easily remembered by humans. For example, I recall (without any assistence) that we have a custom build line for a customer's production environment which is built off of trunk 10680. That tells me a great deal of information immediately. The git hash tells you nothing and you can't remember it.

I'd have to understand more about the implications of simulating revision numbers in git. The system itself is not built on the concept so I doubt it is as easy as just tagging a revision number with each check in. But perhaps I don't understand.

> Opinionated: Bzr is quite schizophrenic/mps in its design, it tries to be centralized and distributed at the same time.

Yes. Our GCD development process has aspects of this as well. So, we resemble this problem.

Of course it would be easier for people to use git. People already have it installed, they understand it and there are good tools/GUI interfaces... which can be used with it.

I do understand people having a burning desire not to deal with another source management system.

> If we make the switch, it could be more efficient to do that before implementing CI/CD

Maybe. But then again, the protocol limitations of git don't work well in our customer silo concept (each customer silo will be an isolated system only accessible via ssh), while bzr can be handled very cleanly in that secure environment. Also, there is a serious effort involved with reworking all of our development processes and migrating everything over to git. It was not trivial to move from cvs to bzr and I'm quite sure moving to git would be a similar exercise. We have enough on our plate right now without adding something that is largely a matter of preference for people but which also brings issues for which there is no simple answer.

In a scenario where we had answers for all the technical limitations of git, we would still need to carefully think about when the timing is right for the shift. That timing might not be now.

#### #18 - 10/28/2022 09:40 AM - Hynek Cihlar

Greg Shah wrote:

> Yes, in a rebase situation the revision numbers are not stable.  But once something is committed to trunk, it is completely stable and can be easily remembered by humans. For example, I recall (without any assistence) that we have a custom build line for a customer's production environment which is built off of trunk 10680.  That tells me a great deal of information immediately.  The git hash tells you nothing and you can't remember it.

For the matter of fairness. While it is true that trunk ids are stable, most of the references are created on the task branches where active development takes place. It's happened to me many times looking for a particular revision in the history, which either took untrivial amount of effort or didn't bring the answer at all.

Any important revisions deserve tags and so having cryptic ids in a form of hash shouldn't pose issues.

#### #19 - 10/28/2022 10:14 AM - Tijs Wickardt

Greg Shah wrote:

> The Breezy project is active, though it is admittedly not strategic.  Please see my comments about this in Why-Do-We-Use-bzr-Instead-of-git. FYI, Ubuntu is a project that still uses bzr so it is not like we are the only ones.

It seems Ubuntu changed from bzr to git, if I'm not mistaken. See: https://code.launchpad.net/ubuntu

> > If we make the switch, it could be more efficient to do that before implementing CI/CD
>
> Maybe.  But then again, the protocol limitations of git don't work well in our customer silo concept (each customer silo will be an isolated system only accessible via ssh), while bzr can be handled very cleanly in that secure environment.  Also, there is a serious effort involved with reworking all of our development processes and migrating everything over to git.  It was not trivial to move from cvs to bzr and I'm quite sure moving to git would be a similar exercise.  We have enough on our plate right now without adding something that is largely a matter of preference for people but which also brings issues for which there is no simple answer.
>
> In a scenario where we had answers for all the technical limitations of git, we would still need to carefully think about when the timing is right for the shift.  That timing might not be now.

Yes, I agree on that. See my previous remark "not a fan on bzr, but also hesitant about a 'need to switch'" . It's probably a considerable effort.

#### #20 - 10/28/2022 10:27 AM - Hynek Cihlar

Btw. I found this https://github.com/felipec/git-remote-bzr. Perhaps it could be used to employ some of the git tools.

#### #21 - 10/28/2022 10:34 AM - Tijs Wickardt

Hynek Cihlar wrote:

> Btw. I found this https://github.com/felipec/git-remote-bzr. Perhaps it could be used to employ some of the git tools.

Interesting. I used an almost identical setup for mercurial to git in the past, which worked well.
However, it's also an extra link in the chain, which is cumbersome.
If only used for one way sync, it can work.
(for sync back, you run into technical limitations of the source, like unsupported octopus merges).
**iff** it solves a problem.

#### #22 - 10/28/2022 10:37 AM - Greg Shah

> It seems Ubuntu changed from bzr to git, if I'm not mistaken. See: https://code.launchpad.net/ubuntu

My understanding was that they still use bzr but they have some kind of export to git.

#### #23 - 10/28/2022 10:43 AM - Tijs Wickardt

Greg Shah wrote:

> It seems Ubuntu changed from bzr to git, if I'm not mistaken. See: https://code.launchpad.net/ubuntu

> My understanding was that they still use bzr but they have some kind of export to git.

Thanks, that could well be. Possibly even by git-remote-bzr? (no answer needed, just a brain twist)

#### #24 - 03/08/2023 03:25 PM - Greg Shah

*- Assignee set to Hynek Cihlar*

#### #25 - 03/08/2023 03:54 PM - Greg Shah

We will use Jenkins.

I intend to implement at least two FWD build systems, one for Linux and one for Windows. However, I don't think we want all CI/CD builds going through that system. The build system idea is more so that we can automatically publish binary builds of various branches. For the purposes of this task, assume that FWD will be built locally on the system under test. This will often be a customer-specific system (see Customer Silo Design) but for our 4GL compatibility tests it will be dedicated internal test machines. We can assume that each test environment will have its own FWD checkout and build, even if more than one environment on the same system uses the same branch and revision of FWD. This will keep the environments independent of one another.

We have these use cases:

- customer silos (automated regression tests for customer-applications)
  - automated, nightly runs will occur in the staging instances of each environment
  - one-off runs can be done in dev or even in prod
    - for dev this might be requested on a specific branch by a developer
    - for prod this would be needed when changes happen to replicate the field conditions
- internal test systems
  - 4GL Compatibility using the Testcases project, split into many separate runs since the overall project is too big to run in one set
  - internal application regression tests
    - Hotel GUI sikuli tests
    - Hotel ChUI harness tests
    - other over time
  - performance and load tests (not yet built)

See Testing Process for more details.

For each application under test, I'm expecting that we will have:

- a FWD checkout and build
- conversion
- conversion regression testing against a known good build
- analytics
  - run reporting
  - bring up the report server for external access)
- runtime testing
  - deployment of a test environment
  - reset database to a known state
  - execute testing (the exact mechanism depends on Automated Testing Approaches)

**#26 - 04/25/2023 02:38 PM - Hynek Cihlar**

*- Status changed from New to WIP*