



## Git White Paper

Is Git ready for the Enterprise.....and is the Enterprise ready for Git?

The concept of a Distributed Version Control System (DVCS) is alien to the traditional Enterprise environment. The idea of not having a central server or a rigidly structured hierarchy just doesn't mesh with the established views on what Enterprise development and Software Change & Configuration Management (SCCM) should be.

But as with 'Agile development' before it, this once 'fringe' area previously relegated to small non-critical projects, or written off as 'just that open source thing', is becoming accepted and is challenging the mainstream.

But let's step back a bit, DVCS's are not new. It is not some overnight sensation or young upstart but is actually a venerable old man of computing having been with us in one form or another for decades. So why only now is it challenging for centre stage? And why is Git at the forefront of this movement?

Firstly the king of 'geek chic' Linus Torvalds gave us Git, primarily to secure the Linux kernel and to facilitate the continued development of it in the open source community when free access to the version control tool, Bitkeeper was withdrawn. Not finding a tool that met his needs, Linus developed his own. Git is a purpose built DVCS designed from the ground up to be fast, open, and support collaboration for a large number of distributed parallel developments.

One step closer but not there yet. Linux kernel development is at the pinnacle of open source community projects but not generally worked on internally in an Enterprise environment. Not to say Companies do not contribute, they do, but most work is directed back to the community and the Enterprise just makes use of the results, maintaining clear boundaries between the two.

What has brought Git to the attention of the masses is the adoption of it by other high profile projects, such as Ruby on Rails and Fedora, but it's the choice by Google to use it for the Android Operating System that we believe has been the catalyst.

Google Android is shaking up the mobile phone world and its adoption of Git as its version control system has thrust the tool into the spotlight and 'forced' the adoption of it with in that sector, in a way that the Linux Kernel itself did not and could not do.

At a stroke all Mobile Phone software and hardware manufacturers need to work directly with an open source project to integrate their own bespoke proprietary developments. The choice facing Enterprises, is whether to work with multiple versioning and Configuration management tools and to deal with the challenge of converting and synchronising code between them or to go 'native ' and use Git.

Its not all been plain sailing, in moving to Git, Google realised some of its limitations in terms of security, and the single repository focus and had to develop some additional tools to fully support their desired working model, which whilst not developed with the Enterprise in mind has effectively plugged the gaps enabling enterprise adoption.



Google's additions are:

- Gerrit; a code review tool which addresses access control and security and has rapidly expanded to cover rudimentary project management and administration;
- Repo, a command line tool for manipulating a 'forest' of Git repositories. The forest concept is also something which Google has brought to Git, as a means of implementing component based development by isolating each component in its own Git repository. To manage and maintain this approach a tool was needed which could work at the forest level as opposed to the individual repository, and so Repo was born.

So the question is, is this triumvirate of Open Source tools ready to take centre stage in your Enterprise and replace the expensive commercial offerings?

Yes, these tools provide a credible alternative and when you add a change management tool you have a full SCCM solution. Admittedly they may not look as polished or as slick as the commercial competition but for raw functionality and performance Git holds it own and puts some commercial systems to shame.

But what about those oft publicised Git shortcomings:

### **Is Git secure enough? Isn't there a lack of access control?**

Yes this is a common criticism of the core Git application, but by applying file level permissions and using secure SSH connections for any remote access, these can be implemented. In the same way Subversion is completely open until you implement an authentication and authorisation policy, either via apache or svnserve. And this is not limited to open source tools; IBM Rational ClearCase is completely permissive out of the box and requires the creation of LDAP or Active Directory User Groups before you can secure artefacts. The key point is the securing of Git is a configuration step you need to take to implement your desired Security Model, and whether this is implemented via a third party tool such as Gerrit, Gitosis or Gitolite or by file level permissions is immaterial.

### **Isn't Git just Linux only, it won't run on Windows?**

Git certainly does have a Linux bias, no surprises given its origins, the windows support has often been called or classed as second tier. But that is not the case anymore; the msysgit implementation has now become a stable windows tool and provides the required Git functionality. The creative forces behind Tortoise SVN, the defacto Subversion Windows client, have released a solid version of their windows shell integration for Git, TortoiseGit. With the maturing of J-git and E-git, integration via IDEs such as Eclipse allows interactivity without having to run a native Git command. And just check any online demonstration of Git to see that most are being run from Macs and I think it's fair to say that Git to all extent and purposes is now platform independent.

Leaving Google's offerings aside and looking further afield, perhaps Open Source is still not completely trusted to house intellectual property. There are more and more choices for Git in the commercial marketplace, from Hosting Companies such as codebasehq (<http://www.codebasehq.com/>), codaset (<http://www.codaset.com/>) and Github, ([www.github.com](http://www.github.com)), most now offering an internal version of their application to run behind your firewall, to fully featured Collaborative ALM solutions which support and can manage Git such as Intlands CodeBeamer (<http://www.clearvision-cm.com/products/codebeamer.html>), as well as tools to help integrate Git into existing environments.



Clearvision specialise in hybrid Software Configuration and Change management solutions taking the best of the commercial and open source communities and providing a fully integrated best of breed solution. Our CM Bridge tool ([http://www.clearvision-cm.com/version-control-connectors/cmbridge/ash\\_flypage.tpl.html](http://www.clearvision-cm.com/version-control-connectors/cmbridge/ash_flypage.tpl.html)) integrates Git into a legacy IBM Rational ClearCase deployment, synchronising changes between the two and acting as a multi-site tool if required. This flexible approach allows you to maintain vendor products whilst freeing projects to work in a more agile Git environment where appropriate. The CM Bridge also allows you to perform a migration from ClearCase to Git or vice versa to your own timescales i.e. projects can move across to Git as they are ready rather than a “big bang” approach for all teams. Full migration from ClearCase to Git is also possible using the Migrate2Git tool.

Git is not the only DVCS out there and it would not be fair to ignore the other players in this space. Mercurial (<http://www.clearvision-cm.com/mercurial/mercurial.html>) and Bazaar (<http://bazaar.canonical.com/en/>) offer credible alternatives and advantages to Git in some areas. They both have better native Windows support and have placed more emphasis on usability from the outset rather than raw functionality. Git is undeniably more powerful, but that power comes with the cost of complexity; whilst a lot of this complexity can be hidden, for some the simpler and arguably more intuitive interfaces provided by these alternatives is preferable. But they still lag behind in terms of adoption and popularity.

So is Git ready for the Enterprise? Yes it is and judging by the growth in interest and new tools, the Enterprise is starting to take notice. The more relevant question is, is the Enterprise ready for Git? Is it ready to embrace a fully distributed, agile working model? We are quite there yet but to use Git as the core of a SCCM system, we believe it is.

If you are interested in exploring Git as an option for your SCCM system then Clearvision can help with all aspects of evaluation through to the implementation of Git. From specialised migration and bridging tools, CM Bridge and Migrate2Git, through to enterprise Git support, comprehensive Git training programmes and Git consultancy packages, Clearvision can help you every step of the way.