FREEFORM DYNAMICS

Inside Track Research Note

In association with

PERFORCE

# Enterprise Git in Perspective

Don’t let the tail wag the dog

March 2016

## In a nutshell

**If asked to name a distributed version control system (DVCS), most people in the software development world would probably think of Git. Indeed, some would spontaneously rave about how Git has released them from the shackles of the traditional centralised VCS model. While developers find it liberating, however, letting their enthusiasm drive your whole versioning strategy may not be wise. When looked at through an enterprise lens, Git's open source, open community roots become clear, as do its limitations from a visibility, security, control and scalability perspective. The trick is therefore to apply the same thinking as in the end user computing space, regarding the front-end and back-end parts of the equation as related but separate.**

## A question of balance

*A dangerous adage we hear in the world of business is "The customer is always right."*

A dangerous adage we hear in the world of business is "The customer is always right." If you work to this principle, you risk disappointing your clients and never doing anything truly innovative. Many of the products and services we enjoy from Apple, for example, would never exist if the company took its lead from what people said they wanted. Indeed products designed based on input from focus groups frequently fail.

*Another dangerous notion is that employees should have total freedom to choose whatever technology they want to do their job, based on the equally risky assumption that they are best placed to define their own requirements.*

Along similar lines, another dangerous notion is that employees should have total freedom to choose whatever technology they want to do their job, based on the equally risky assumption that they are best placed to define their own requirements. This is worth exploring further as it provides a useful parallel for considering how to deal with tooling decisions in a software development context.

### Freedom versus anarchy

*If you work in IT, you will be familiar with the consequences of letting people have their own way without any constraints.*

If you work in IT, you will be familiar with the consequences of letting people have their own way without any constraints. Users tend to take a short-term, parochial view, and are too often driven by desire and fashion rather than genuine need. In practical terms, they don't know what they don't know, and are, in fact, not great at assessing even their own requirements, let alone those of the broader team and the organisation overall. The result is that they often create as many problems as they solve, for themselves, for others and the business.

In our paper entitled "**IT-Business Alignment Revisited**"[1], we discuss user-driven technology decisions and make the observation that *"Benefits such as flexibility and productivity are always emphasised by users, but the additional time and distraction of working around limitations and constraints is often downplayed. Over the years, it has never ceased to amaze us how much effort people will make, and how tolerant they are willing to be, in order to use the latest desirable tech in a corporate environment."* At more of an organisational level, concerns range from collaborative friction caused by disjoints and incompatibilities, through elevated procurement and support costs, to issues and uncertainties around security and compliance.

*Success comes from balancing needs across the individual, team and business levels.*

But none of this should be taken as an excuse to go to the other extreme of a locked-down, dictatorial approach. In another of our papers entitled "**Freedom Without Anarchy**"[2], we note that *"Organisations with a strong culture of empowerment get more from their people and generally have a happier and more content workforce."* Based on input from over 500 IT and business professionals, it was clear that success comes from balancing needs across the individual, team and business levels.

## Focus on enabling-infrastructure

*The trick is to create a robust and well-controlled set of capabilities at the core, supporting a high degree of flexibility at the edge.*

From a practical perspective, the abovementioned research went on to tell us that higher performing organisations put a lot of focus on 'enabling-infrastructure', i.e. systems and tools to enable user freedom, but within a structured, policy-driven framework. The trick is to create a robust and well-controlled set of capabilities at the core, supporting a high degree of flexibility at the edge.

With regard to specific functions, the core takes care of things like storage, data protection, information management, integration, workflow, security, compliance and overall monitoring. With the right architecture, users can plug whatever devices and tools they want into this (within reason), and everybody's happy.

# What's good for the goose…

*How much do you think in the same way when it comes to enabling teams within the IT department?*

As an IT professional, particularly if you work in a management role, everything we have discussed so far will make sense to you. You or your colleagues may even be implementing the kind of end user computing environment we have outlined. But how much do you think in the same way when it comes to enabling teams within the IT department itself, given that a similar, perhaps even stronger, desire for freedom and flexibility exists?

## A case of double standards?

*The arguments heard from individuals in both the end user and IT camps are almost identical.*

The arguments heard from individuals in both the end user and IT camps are almost identical. The case made by a developer telling you why they want to use their favourite open source tool, or manage their code through an open community repository, for example, sounds no different to a business professional trying to justify use of their desired mobile device or personal Dropbox account. Phrases like "It's a lot more flexible than the company standard", "I can be more creative and productive", and "Everyone else I know is doing it" are used in each case. Yet we are far less likely to challenge this logic when the words come from IT pros rather than end users.

One reason for what on the surface looks like a case of double-standards is the assumption that IT professionals know a lot more than the average user, so are much better placed to make tech-related decisions. But that doesn't stop them taking that same parochial view and being susceptible to fashion and personal interest. It may not be the latest device bling, but at any one time, some tools and techniques are more trendy and in demand than others. And just as with end users, then there is the temptation to force fit the wrong solution into the wrong situation, and invent all kinds of workarounds to smooth over the mismatch.

*There is the temptation to force fit the wrong solution into the wrong situation, and invent all kinds of workarounds to smooth over the mismatch.*

## If all you have is a hammer…

We then have a risk we highlighted in our paper "Orchestrating the DevOps Tool Chain" [3], where we noted: *"There is a tendency for enthusiasts to sometimes get carried away with the power of their chosen tool, in turn leading to overextension of solutions. With enough scripting it's perfectly possible to use build-automation tools to configure hardware, or ops automation tools to run software builds, but neither is necessarily a good idea."*

We went on to conclude that it of course makes sense to encourage experimentation and allow IT practitioners to select the right tool for the job in many circumstances, but at the same time someone needs to have an eye on the bigger picture to ensure broader coherence, scalability and management of costs and risks.

*Someone needs to have an eye on the bigger picture to ensure broader coherence, scalability and management of costs and risks.*

## Are developers really a special case?

It can be hard to take some of these realities on board given all of the buzz at the moment around the need to empower developers. The imperative here is reinforced by publications like "The New Kingmakers"[4], as well as research such as our recent global study (based on over 1,400 respondents) captured in our report entitled "Exploiting the Software Advantage"[5]. The latter, for example, shows a strong correlation between success with digital business initiatives and the adoption of modern software delivery techniques. All of this would seem to support the notion of encouraging developers to constantly push the boundaries.

But again, boring as it might sound, unless you strike the right balance, the danger is that you undermine your ability to contribute broader business value in a scalable, risk aware and sustainable manner over the longer term. At this point, another old adage starts to seem very apt - "What's good for the goose is good for the gander."

*What we have here is fundamentally an infrastructure discussion.*

Empowerment is important, but unchecked freedom can be dangerous, disruptive and costly, so just as with end user computing, what we have here is fundamentally an infrastructure discussion.

# Git as a real world example of the principles and practicalities in action

*Git has started to find its way into enterprise development environments, often introduced into the organisation by developers.*

The march of Git in developer circles for distributed version control provides us with a good example of some of the challenges, principles and practicalities we have been discussing. Originally developed by Linus Torvalds to support Linux kernel development, Git is widely used by the open source and independent developer communities. Even some major commercial IT vendors have integrated Git capabilities into their tooling and embraced Git as a mechanism for making their own code available to developer communities.

Many (though not all) developers love Git because it empowers the individual through the way in which the distributed model has been implemented. You can take work offline and enjoy the benefits of a local repository, freely cloning and branching code lines, then contributing/merging your efforts back into the community environment when and how you choose. Of course, owners of 'master' repositories are able to control what makes it into that sacred space, but on the whole, Git encapsulates the spirit of developer freedom and open collaboration very well.

*Git can feel like a breath of fresh air.*

## Git in the enterprise

Not surprisingly, Git has started to find its way into enterprise development environments, often introduced into the organisation by developers themselves. And if teams have been used to old-fashioned, highly-centralised version control systems, Git can feel like a breath of fresh air, and in a more fast-moving environment, e.g. one focused on rapidly evolving digital solutions, it can be extremely liberating.

*Git 'as it comes', or with a few appropriate extensions, works fine and delivers on the promise for many enterprises.*

And indeed Git 'as it comes', or with a few appropriate extensions, works fine and delivers on the promise for many enterprises. This is especially the case if you run your development organisation as a collection of small, self-contained teams and projects.

Native Git will find its natural place in such an environment, just as it does in the independent developer community.

But in a more typical enterprise context with a pipeline of many tens or even hundreds of inter-dependent development streams, and a need for centralised coordination and management, a range of limitations and constraints can make Git a much less natural fit. While it's great at what it was designed for, i.e. enabling distributed developer-friendly workflow capability, the fact that it wasn't conceived with enterprise control, security, availability, recovery and other requirements in mind becomes clear as you try to use it in a broader and more diverse manner.

*In a more typical enterprise context with a pipeline of many tens or even hundreds of inter-dependent development streams, and a need for centralised coordination and management, a range of limitations and constraints can make Git a much less natural fit.*

It's not that Git ignores the things we have mentioned, but as discussed in "7 Tips for Success with Git in the Enterprise"[6], you frequently need to implement quite a few bolt-ons and workarounds to make it operate anywhere near acceptably.

## String and sticky tape

By the time you have implemented tools such GitHub, GitLab, Atlassian, BitBucket and other tools required to plug the gaps and strengthen the environment, it's easy to end up with a much more complex, fragmented and fragile environment than anyone originally envisaged. The elegance of Git is lost, and you may still not have even achieved the level and granularity of control, coordination and visibility you want.

*It's easy to end up with a much more complex, fragmented and fragile environment than anyone originally envisaged.*

And when things break or hit a wall or integration disjoint, it's fine for developers to dismiss the significance of having to resort to clever command-line remedies, but this isn't sustainable at scale. It really isn't healthy, in fact it can be quite dangerous, for a large, business-critical development function to be held together by the tooling equivalent of 'string and sticky tape'.

The reality is that the native Git simply represents the wrong starting point for dealing with many enterprise needs. Life becomes a lot easier when you acknowledge this and switch your focus from dealing with the symptoms one at a time to addressing the underlying cause, i.e. your reliance on the fundamentally limited core Git engine.

## Back to the enabling-infrastructure discussion

This brings us back full circle to where we started with the end user computing discussion. The trick is to separate the front-end and back-end parts of the equation. Letting either end-user or developer tools that weren't conceived with enterprise needs in mind dictate the nature of the core infrastructure is a recipe for unnecessary complexity, cost and risk that will ultimately work against the flexibility, efficiency and productivity objectives you probably set out with.

And it isn't just the limitations of current capabilities as activity is scaled-up across more projects and teams. You also need to consider how the nature and scope of requirements are evolving over time. As a simple example, in our research report "Assembling the DevOps jigsaw" [7], we note that digital business is making it increasingly harder to divorce software development from broader product development. Even if we aren't talking about pure digital offerings, many products today have a digital service component to them.

*The question then arises of how to coordinate version control across software and other digital assets such as specifications, designs, multi-media content, supporting documentation, and so on.*

Given this, the question then arises of how to coordinate version control across software and other digital assets such as specifications, designs, multi-media content, supporting documentation, and so on. Solutions such as Git that were never designed to work optimally with anything other than text-based source files, and force artificial

*The answer to many of the challenges is to implement a Git-friendly infrastructure that is enterprise grade.*

fragmentation of repositories to work around size constraints, struggle to deal with this extension of scope. Even handling binary artefacts generated as part of the software build process to simplify and streamline DevOps orchestration processes can be a challenge.

## Hardening the core

The answer to many of the challenges is to implement a Git-friendly infrastructure that is enterprise grade. This means acknowledging that community editions of tools only get you so far, and looking for vendors who can provide solutions with the appropriate enterprise-level functionality and support. Here are some capabilities to look out for as you explore the options:

- Support for the distributed workflow experience of Git, allowing developers to pull, push, branch, merge, etc. using familiar Git-based clients and techniques.
- Mirroring of Git repositories to a robust, policy-driven back-end environment, with a rich set of controls that can be centrally administered, e.g. controlling who can access what, recording who has accessed what (for auditing and compliance purposes), and so on.
- Extension of version control beyond text-based source files to other digital assets, including large binary objects and multi-media files. Related to this is easy access from clients other than Git, so non-technical staff involved in broader product development are supported via the tools that are natural for them.
- Capabilities that allow large, aggregated and/or inter-dependent repositories to be dipped into selectively, in turn allowing easier navigation, more efficient synchronisation, and better support for automated build and release processes in a DevOps environment.
- Enterprise-grade scalability and recoverability features to enhance performance and availability of business-critical development environments.

*Solutions exist nowadays that allow you to adopt or scale up the use of Git on your terms.*

This list is by no means exhaustive, and it's clearly possible that some of the capabilities we mention are not that relevant in your environment. Our aim is simply to highlight that solutions exist nowadays that allow you to adopt or scale up the use of Git on your terms.

*Be careful not to let developer desires and preferences inadvertently define the infrastructure underpinning your development and/or DevOps environment.*

## The bottom line

Be careful not to let developer desires and preferences inadvertently define the infrastructure underpinning your development and/or DevOps environment. This would be the equivalent of the tail wagging the dog. Meeting developer needs is of course critical, and supporting Git may be an important part of this, but you have to think more broadly. If you look beyond native Git and associated community solutions, you will discover that you don't have to sacrifice enterprise-class performance, scalability, control and security in order to benefit from modern, distributed version control techniques.

*It is possible to keep developers happy, even make them happier, while at the same time driving increased productivity, keeping costs and risks under control, and laying good foundations for the future.*

Working with the right vendor(s), it is possible to keep developers happy, even make them happier, while at the same time driving increased productivity, keeping costs and risks under control, and laying good foundations for the future as your requirements evolve.

## References and further reading

1. IT-Business Alignment Revisited
   *Accommodating increased user influence*
   http://www.freeformdynamics.com/fullarticle.asp?aid=1766

2. Freedom without Anarchy
   *Empowering your users while keeping control*
   http://www.freeformdynamics.com/fullarticle.asp?aid=1630

3. Orchestrating the DevOps Tool Chain
   *Continuous delivery for the Enterprise*
   http://www.freeformdynamics.com/fullarticle.asp?aid=1853

4. The New Kingmakers
   *How Developers Conquered the World*
   http://thenewkingmakers.com/

5. Exploiting the Software Advantage
   *Lessons from Digital Disrupters*
   http://www.freeformdynamics.com/fullarticle.asp?aid=1867

6. Seven Tips for Success with Git in The Enterprise
   *An eBook by Perforce*
   https://www.perforce.com/blog/150923/seven-tips-success-git-enterprise

7. Assembling the DevOps Jigsaw
   *Do you have all the right pieces in place?*
   http://www.freeformdynamics.com/fullarticle.asp?aid=1868

## About Freeform Dynamics

Freeform Dynamics is an IT industry analyst firm. Through our research and insights, we aim to help busy IT and business professionals get up to speed on the latest technology developments, and make better-informed investment decisions.

For more information, and access to our library of free research, please visit www.freeformdynamics.com.

## About Perforce Software

Perforce Software helps companies build complex products more collaboratively and securely. Its highly scalable source code management (SCM) and collaboration platform, Perforce Helix, enables global teams to collaborate on any type or size of file. It supports both centralized and distributed (DVCS) workflows while safeguarding intellectual property with advanced behavioral analytics. Perforce is trusted by the world's most innovative brands, including adidas, Samsung, NVIDIA, Intuit, Pixar, Salesforce, EA, Ubisoft, and VMware. The company is headquartered in Eden Prairie, Minnesota, with offices in California, the United Kingdom, Canada and Australia, and sales partners around the globe.

### Helix GitSwarm

Helix GitSwarm is where developer preferences meet enterprise needs. Based on the popular GitLab collaboration suite, it gives developers the pure Git-based workflow they love while making projects easy to manage. Your team's work is automatically synchronized to the Perforce Helix mainline repository for DevOps efficiency and security.

For more information on enterprise-grade Git solutions from Perforce Software, please visit www.perforce.com/gitswarm.

## Terms of use