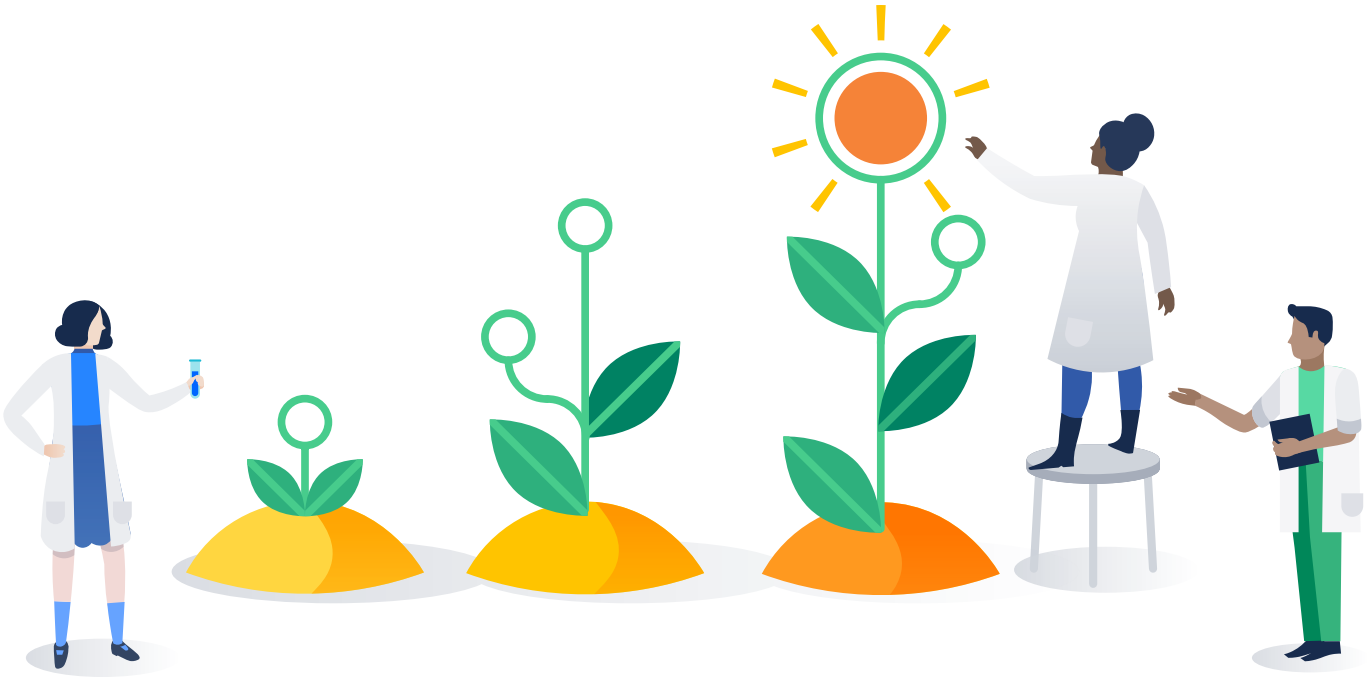
ATLASSIAN



Scaling Git with Bitbucket Data Center

Considerations for large teams switching to Git

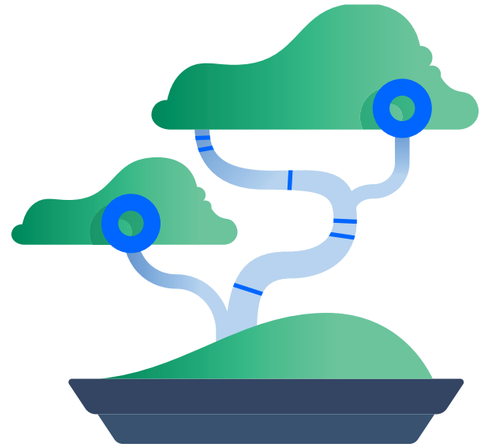
Contents

01

What is Git, why do I want it, and why is it hard to scale?

So. Your software team is expanding and taking on more high-value projects. That's great news! The bad news, however, is that your centralized version control system isn't really cutting it anymore. For growing IT organizations, moving to a distributed version control system is now considered an inevitable shift. This paper outlines some of the benefits of Git as a distributed version control system and how Bitbucket Data Center can help your company scale its Git-powered operations smoothly.

As software development increases in complexity, and development teams become more globalized, centralized version control systems like Subversion stop meeting the needs of their users. But distributed version control systems like Git thrive in this environment. Rather than confining developers to one centralized location for the full version history of the software, distributed version control allows each developer's working copy of the code to act as a complete history of all changes made to the software. Today, Git is far and away the most widely used distributed version control system by developers– and for good reason.



Some of the key benefits of adopting Git are:

Distributed development

Distributed development gives each developer a working copy of the full repository history, making development faster by speeding up the commit process and reducing developers' interdependence, as well as their dependence on a network connection.

Feature branching

Feature branching gives developers an isolated environment for every change they make to the code base, insulating the main line of code from any new features or bug fixes before they're reviewed and ready to be implemented.

Flexibility

Git offers flexibility in several respects including development workflows and deployment environments allowing your organization to create its optimal development ecosystem.

Codebase safety

Git is designed with maintaining the integrity of managed source code as a top priority, using secure algorithms to preserve your code, change history, and traceability against both accidental and malicious change.

Community

Git has become the expected version control systems in many circles, and is very popular among open source projects. This means its easy to take advantage of third party libraries and encourage others to use your open source code.

Pull requests

Pull requests create an intuitive way for project leads to track changes and for developers to discuss and review each other's work before new branches are integrated into the rest of the codebase.

Faster release cycles

The ultimate result of distributed development, feature branches, a strong community, and pull requests is a more agile workflow and faster release cycles!

IMPLEMENTING GIT

Implementing Git, just like implementing any organization-wide change, comes with its own set of roadblocks that can be easily mitigated through a Git management solution like Bitbucket:

Learning to use Git can be hard

While some of the terminology in Git can be cryptic for newcomers, Git is a very robust system and teams that take the time to learn it can substantially increase their development speed. The Git maintainers have also been steadily releasing new improvements like sensible command defaults and contextual help messages that have made the on-boarding process much easier.

Git doesn't come with administrative controls

There aren't any native methods in Git to control your workflow, environment, or how your team operates in your development ecosystem. Without a repository management tool to harness its power and establish these protocols, Git can seem like a development free-for-all.

No native access management

While Git repositories and the integrity of their histories are cryptographically secure, there is no out of the box solution with Git for setting up permission schemes for your projects and repositories. In order to grant specific users read-only, read/write, and sys admin privileges at various levels, a Git repository management solution like Bitbucket is necessary.

No compliance checks by default

Git alone does not offer any intuitive ways to monitor commits, audit changes, or track development efforts from project management software. Neither can Git enforce specific paradigms of development (e.g. only the author of a commit can push said commit). These functions are most efficiently executed by outside applications.

Despite these hurdles, Git has been crowned the industry standard and the most popular choice for version control among developers. Git is favored by significant portions of both experienced developers and college students, which means some of your team has probably used it already. Git's distributed development environment also grants users freedom and makes it great for remote teams with increased process speed and less dependence on network access. Git's workflow options, feature branches, and pull request system encourage collaboration among your team in the way that best suits them.

Choosing to adopt Git is the first step...but what about all the other considerations?

Integrations

How will your Git tooling integrate with the rest of the tools in your software development ecosystem like your issue tracker, continuous integration solution, and communication tool?

Customization

Will your software team need to make customizations to how they interact with Git to address specific needs or requirements?

If you're a large enterprise, or in a heavily-regulated industry, Git adoption can become even trickier and come with more points of consideration:

Scale

How do you ensure that your version control system and development ecosystem can scale to support your users as they grow in number and tax your hardware more?

Service-level agreements

Are you confident that your systems will stay functioning and available to users and customers at the necessary rates?

Disaster recovery

Do you have a recovery plan in place to quickly get up and running if your entire system is down?

Compliance requirements

Are there mechanisms in place to establish a required workflow for your development ecosystem and meet the necessary traceability and accountability requirements?

Each of these issues is important to address when establishing a version control system at your organization. It's much easier to plan for these needs ahead of time than address them after they become a problem. So how do you overcome these challenges and still reap the benefits of adopting Git for your team?

SOLUTION

Select a tool that supports Git at scale, supports compliance and security needs, integrates with the rest of your development ecosystem, and, most importantly, keeps your developers happy.

> "Happy developers are the lifeblood of any software organization and keeping them happy means removing obstacles that get between them and coding before they realize they even exist.

Git can quickly become a very computation-intensive system when used by teams of hundreds of developers (especially distributed development teams) so it's important that your Git management solution can keep the lights on and the wheels spinning quickly.

02

Scaling Git with Bitbucket Data Center

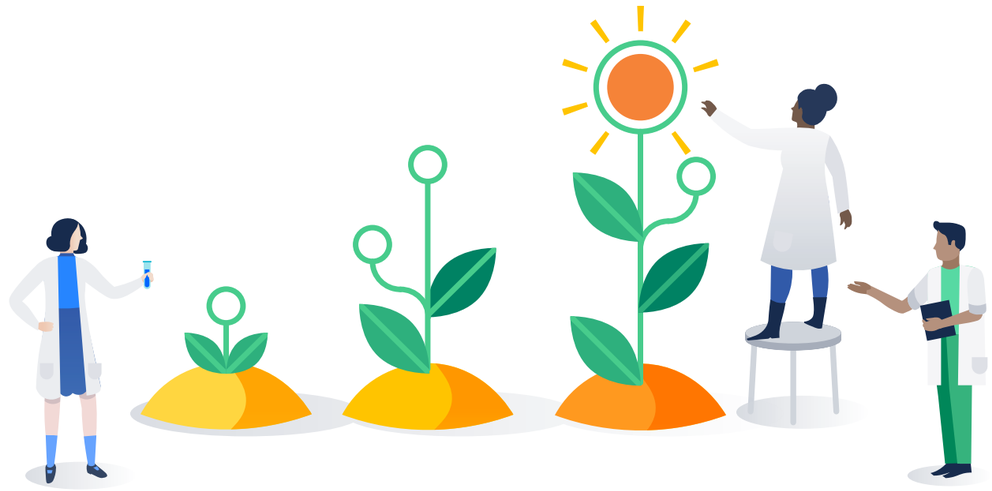
Bitbucket Server fits the needs of many development teams, but as growing software organizations take on more projects, hire remote developers, and encounter new compliance requirements, additional power may be required. That's why we specifically built Bitbucket Data Center to scale amicably with the growth and evolving needs of your development team. Data Center sports numerous features and background processes that ensure performance and reliability. Some of these include:

Source code management (SCM) cache

While redundancy is a great way for software teams to ensure their code is producing the desired results, you don't want Git performing operations over and over when it doesn't have to. Bitbucket Data Center's SCM cache is designed to relieve the burden of repetitive Git processes. Build agents can automatically trigger torrents of repetitive Git clone and fetch commands, so overtaxing your system's hardware happens all too easily. The SCM cache prompts Bitbucket to analyze each Git hosting request, and, if it's a cacheable clone request, Bitbucket will save a copy on local fast storage. The next time an identical clone request is received for the same repository, Bitbucket will stream the files from fast local storage instead of starting another Git process, using far less processing power and memory.

Adaptive throttling

If your organization is already using Git, you've probably seen this scene before: frustrated developers twiddling their thumbs, waiting forever for clones or builds that are taking much longer than they should. Even after implementing processes designed to reduce Git's CPU and memory footprint, every system hits its capacity at some point. For Git management systems under high stress, you can start seeing performance issues as Git process vie for scarce CPU and memory. Bitbucket's throttle service addresses this problem with a simple solution: by limiting the number of Git hosting operations that are allowed to run concurrently. If many users (or, more likely, build agents) are attempting to perform Git hosting operations all at once, only some



are processed immediately while the rest wait in a queue until the the number of Git hosting operations falls back below the threshold you've set. Bitbucket also recognizes that not all Git operations are alike and can vary considerably in how much power they use. This is where adaptive throttling comes into play. Adaptive throttling lets Bitbucket constantly monitor the system load and adjust the number of allowable Git hosting operations accordingly. This keeps the request throttling system from needlessly queuing large numbers of simple operations and from executing too many large operations at once. Bitbucket's adaptive throttling system optimizes your system's capabilities, minimizes thumb twiddling, and maximizes development time.

Git LFS (large file storage)

Git LFS is a Git extension that makes it easier for developers to work with large files in their Git repositories. One of the cool things about Git LFS is that it makes Git a viable option for certain developers who work with large binary files like designers and game development teams. Historically, they've needed to use version control systems like Perforce to handle their large files or just haven't used a version control system. But with Git LFS, those days are over.

Git alone can really struggle to handle large files. Since every version of a file has to be copied in the cloning process, large files can substantially increase clone and push times. Git LFS solves this problem by replacing large files with tiny pointer files that reference their location on a large file storage cache. Copies of large files' histories are stored locally and remotely with the main repository. This system of pointers significantly decreases operation times by only pushing or cloning new versions of large files, and simply storing pointers for the rest. These pointers also work in the background, so as far as developers can see, they have direct access to their large files. With Git LFS support, your team can take on a much wider spectrum of projects that may not have been possible otherwise!

PERFORMANCE AT SCALE

Your software team is growing, so isn't it natural that your development ecosystem should be able to grow with it? Systems that aren't designed to scale see massive performance decreases when the number of users and amount of data they process increases beyond their capabilities. Bitbucket Data Center solves this issue by not only providing performance at scale, but instant scaleability as well. Data Center's turnkey active/active clustering allows you to seamlessly add nodes to your cluster without any system downtime or additional licensing fees. Simply add a new node to your cluster and the system will incorporate it into the structure.

Rather than one server running a Git management solution, a typical Bitbucket Data Center cluster comprises a number of machines, each with their own purpose:

Application nodes

Each of these nodes runs a copy of Bitbucket and processes the requests that your users make. The more nodes you install, the more users and requests your system can handle.

Load balancer

The load balancer autonomously distributes requests from your users to your nodes. If a cluster node goes down, the load balancer automatically reroutes requests to the other nodes within seconds, minimizing performance loss and providing true high availability.

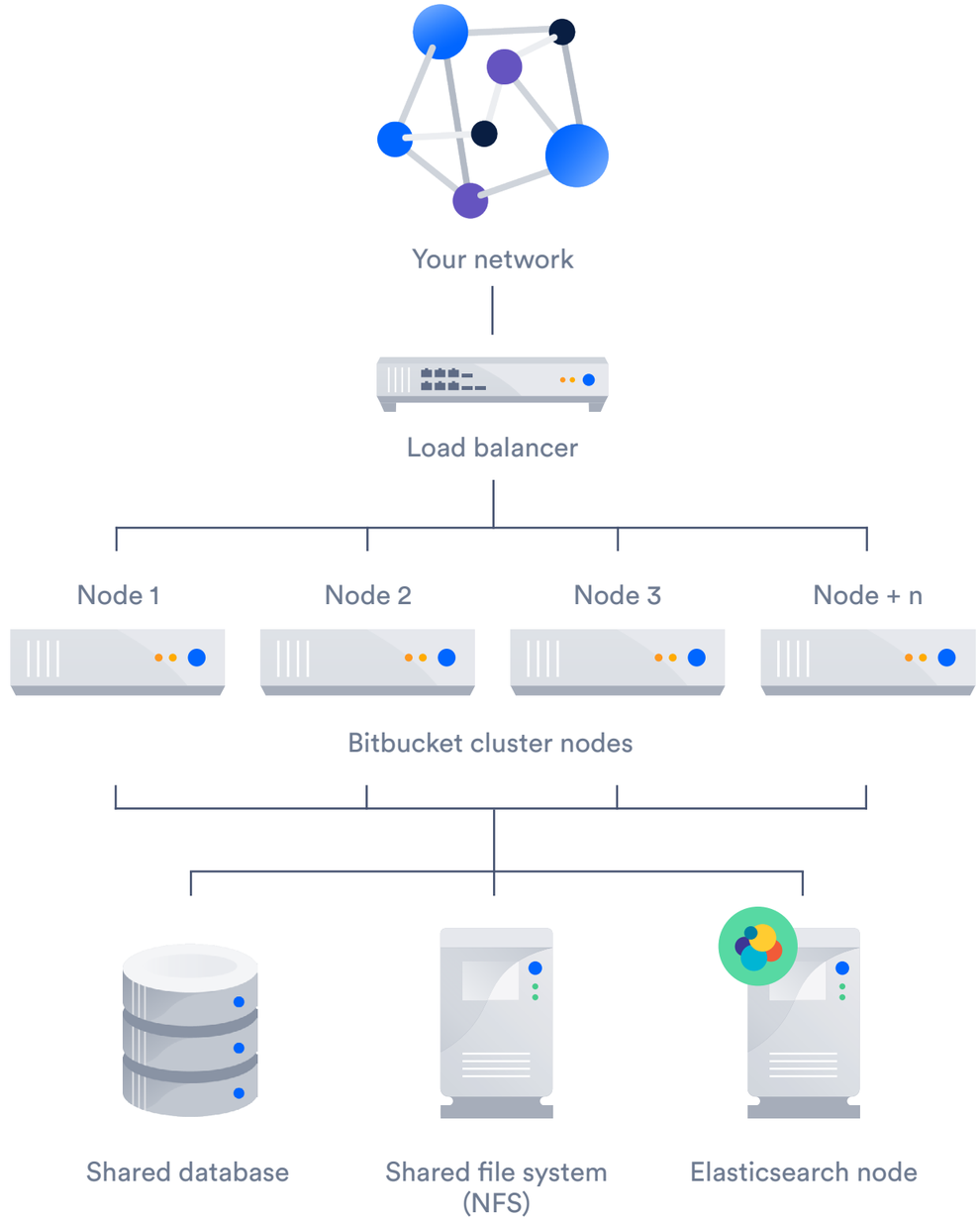
Shared database and NFS file system

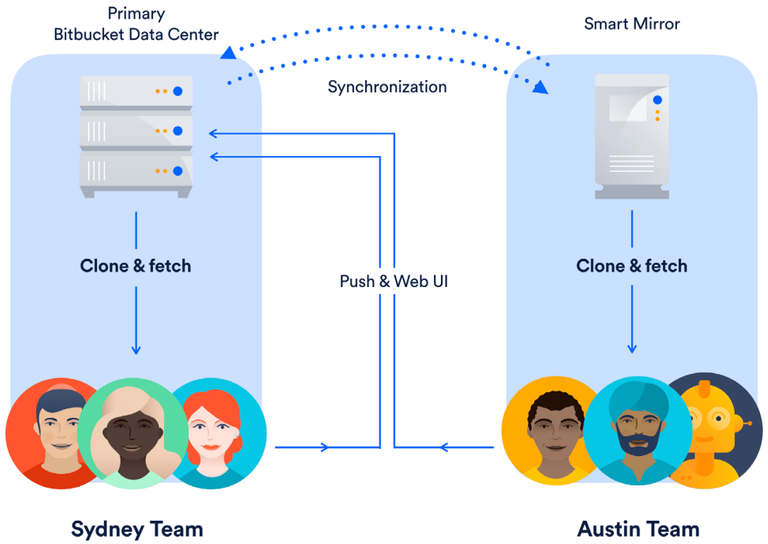
Bitbucket Data Center supports the same databases as Bitbucket Server (except for MySQL). A dedicated high performance NFS filesystem stores repository, attachment, and avatar data, allowing for quick access from any of your cluster nodes.

Elasticsearch node

Elasticsearch provides a fast, full-text search engine that powers searches at the project, repository, and source code levels within Bitbucket.

These components work in unison to maximize the performance of your system and allow you to scale Bitbucket seamlessly with the growth of your development team.





Smart mirroring for global teams

It's pretty exciting when you can bring some of the best developers in the world to your team. The only problem is that they might be on the other side of the world. With development teams becoming increasingly globalized (Atlassian has developers on three continents, for example), companies have less control over the performance of the networks that connect their devs. Additionally, many software development teams who use Git end up with large repositories as a result of storing extensive historical information, utilizing monolithic repositories, or storing large binaries (often all three at once). These compounding issues lead to lost development time when developers have to wait hours to clone a large repository stored on a server across the world. This is where Bitbucket Data Center's smart mirroring comes into play.

Smart mirroring restores lost development time by allowing you to set up live mirror nodes with automatically synced, read-only copies of repositories locally for your distributed developers. Local and remote developers simply push their commits to the primary system and the mirrors automatically sync to incorporate these commits. Additionally, these mirrors automatically delegate authentication and authorization of users' credentials back to the primary instance, so no additional user management is necessary for the mirrors. Smart mirroring cuts down wait times for remote developers while keeping your permission schemes in place, allowing you to expand your development teams abroad with little overhead.

> **Here at Atlassian, we've seen 25x faster clone times for 5GB repositories between San Francisco and Sydney using smart mirroring. 25x faster. Seriously.**

Deployment flexibility

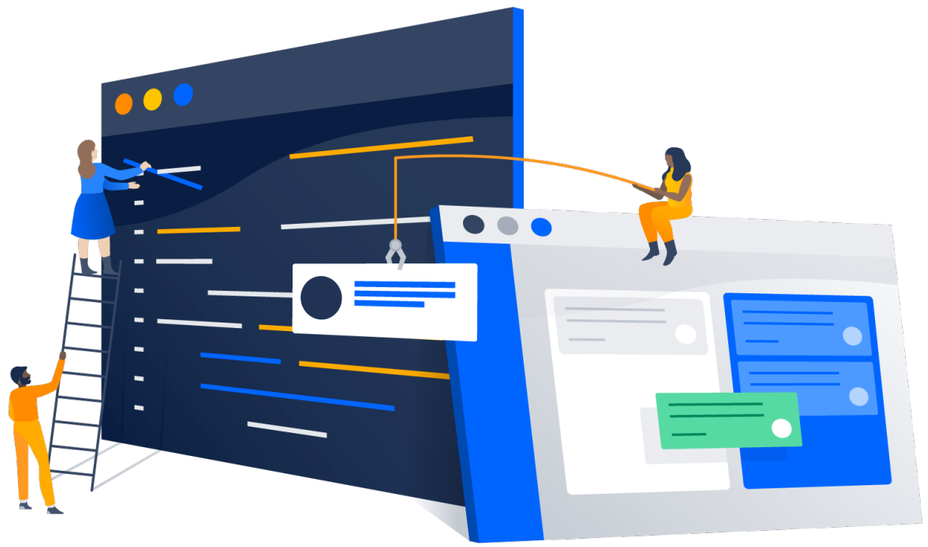
Bitbucket Data Center leaves it up to you to choose the infrastructure to host your deployment on. Whether it's on bare metal servers, virtual machines, or a hosted environment, Data Center runs in whatever environment that best suits your team's needs. In a recent survey of Atlassian Data Center customers, 85% of installations were at least partially virtualized. As infrastructure as a service becomes more popular, it becomes increasingly important that these offsite systems maintain compatibility with the systems IT teams want to put in place. To this end, Bitbucket Data Center offers official support for Amazon Web Services.

You can quickly deploy Data Center clusters to AWS, including multiple application nodes, a shared database, and a load balancer. To make the process as seamless as possible, we collaborated with Amazon to develop CloudFormation templates and Quickstart Guides for all of Atlassian's available Data Center products. With capabilities like auto-scaling and instant provisioning of nodes, scaling Data Center to meet growing demand is simple. When using infrastructure-as-a-service (IaaS) products like AWS, make sure your nodes are as collocated as possible to ensure optimal functionality. This means locating nodes in the same geographical location.

Integrations

Having a high-performance and scalable system is only half the battle. Creating a reusable, secure, and speedy development workflow makes your Git management tool even more powerful. There are several ways Bitbucket Data Center can help in this area, but deep integrations with the rest of your development ecosystem just may be the most important. Your teams can save time, boost visibility and transparency, and reduce project overhead with out of the box integrations with Jira Software, Bamboo, and Hipchat. Bi-directional communication allows developers to view and transition Jira issues, get the status of their continuous integration builds, and update Hipchat rooms without leaving Bitbucket. Likewise, development information is piped back into these tools for easy access to information by interested parties.

If your company is one of the many organizations adopting DevOps, these integrations will play an important role. There are likely several other tools in your ecosystem that must be integrated with your version control, or perhaps you have specific workflow requirements that are not available out of the box. To solve this, Bitbucket Data Center offers plugin capabilities, in which there are hundreds of add-ons already available at the Atlassian Marketplace. Additional extensibility is also available through the Bitbucket REST APIs.



03

What about compliance?

Developing and operating software when constrained by compliance requirements (be they rooted in a regulated industry, a service-level agreement, or elsewhere) comes with its own specific needs. Bitbucket Data Center addresses these needs with a number of features designed to help your team operate within these constraints:

- High availability ensures that your system will stay functional and available to users and customers should one of your cluster nodes go down.
- Disaster recovery lets your team to set up an offsite cold standby array of machines ready to be immediately implemented if your entire system goes down.
- Support for various branching models enables your team to choose their optimal workflow, and Git hooks compatibility can enforce that workflow so everyone is accountable and on the same page.
- Merge checks protect your codebase by allowing your sys admins to place strict requirements (like a set number of peer approvals) before new code is merged into the main branch.
- Bitbucket's integrations with Bamboo and other third party continuous integration solutions let your team view build statuses in Bitbucket, respond faster to build failures, and reduce the need to switch applications.
- Sys admins can bestow accessibility permissions at global, project, and repository levels, ensuring that data is only available to those who should have access.
- Complete logs of every single event that occurs on your servers allow you to audit each commit, pull request, user creation, and more should the need arise.
- SAML 2.0 support enables single sign-on, easing the user management burden on your sys admins and giving your users quicker and easier access to Bitbucket.

Performance monitoring and ease of testability

Bitbucket Server and Data Center both support JMX monitoring for a wide array of statistics including thread pools, thread pool attributes, repositories, tickets, events, and more. Our build engineers, however, wanted to take performance monitoring to the next level. We wanted to construct a method to test the expansion capabilities better than the "try and see" approach we were using, and solve performance issues before they became issues. Thus, the Elastic Experiment Executor (E3) performance tool was born.

E3 lets you run controlled performance experiments on theoretical infrastructure configurations. E3 spins up a configuration of Bitbucket, throws a repeatable workload of any desired size and shape at it, and observes its throughput, response times, and other vital statistics while it runs. We've used it ourselves to analyze whether our Bitbucket configuration could handle additional build agents on Bamboo before implementing them. This kind of information is vital to monitoring your system's performance and, more importantly, knowing when its time to upgrade your system before it becomes a problem.

04

Why choose Bitbucket Data Center?

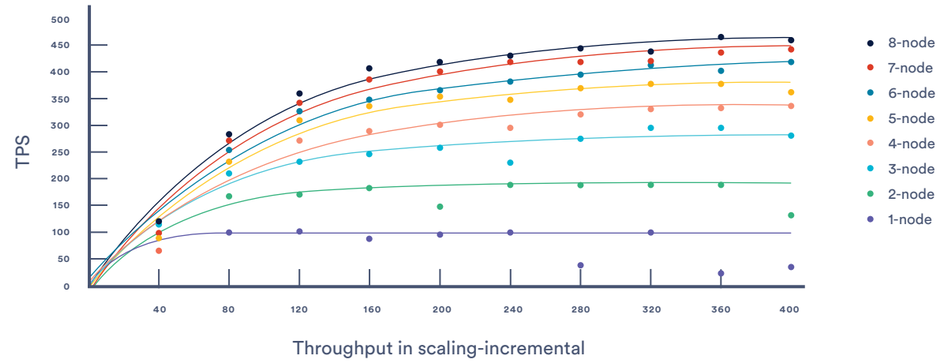
It meets all the requirements

Bitbucket Data Center was designed to unleash software teams' potential and to meet the increasingly rigorous needs of growing teams. Whether your team is increasing in size, distribution, or both, Bitbucket Data Center will ensure that your development team is operating as efficiently as possible. In addition to all these features, we're working to make Bitbucket Data Center meet even more of your teams' needs, releasing updates multiple times a year.

Elastic experiment executor

For a better idea of just how well Bitbucket Data Center scales with the growth of your team, take a look at results we found after using the "Elastic Experiment Executor" to test the loads that our Data Center configuration would be able to handle at various numbers of nodes. This graph shows Data Center's sustained tests per second (TPS) at various cluster sizes and load levels. Tests in this context refer to any operations (such as a Git clone or loading a page) that are successfully executed. The operations are taken from a typical mix of interactions executed by Bitbucket.

As you can see, each node added substantially increases the number of operations Bitbucket Data Center can handle in a timely manner. This is sort of scalability and the ability to predict when it's needed are imperative for growing software teams. Similar results can also be reproduced for your own systems by using E3 on your own instance.



Throughput in scaling-incremental

Simplified pricing model

Bitbucket Data Center offers a simplified pricing model that scales only with the number of users accessing the application. License pricing does not increase with the number of nodes you wish to install in your cluster or with different deployment environments. This way, you can create a system for your whole development team without worrying about incurring more costs.

Data Center is an annual term license and requires renewal each year. The renewal cost is identical to that of a new license purchase for the user tier you require. Additionally, the cost per user at each tier is identical to that of Bitbucket Server and the lowest among comparable Git enterprise solutions. The table below shows exact pricing. Please note, however, when renewing a license with an upgraded tier, you are not adding that number of users to your license, but simply upgrading to that tier. This pricing model is designed to make scaling Bitbucket Data Center with the growth of your development team as straightforward as possible.

Bitbucket Data Center Pricing

Users	Commercial Annual Price
11 - 25 users	US $1,800
26 - 50 users	US $3,300
51 - 100 users	US 6,000
101 - 250 users	US $12,000
251 - 500 users	US $16,000
501 - 1,000 users	US $24,000
Each additional 1,000 users	US $24,000

> Features such as clustering for load-balancing and reliability, fine-grain permissions, branching and forking for large-scale coordinated development efforts and the availability of source code for customizations make Bitbucket Data Center a no-brainer for bringing Git into the Enterprise.
>
> **Frederic Ros**
> Head of software development engineering, Amadeus

Teams at Amadeus are succeeding with Bitbucket Data Center

Amadeus is a leader in technology solutions for the global travel and tourism. Through its flagship product, Altéa Customer Management System, Amadeus connects airlines, hotels, railways, cruise lines, and other travel providers to over 100,000 travel agents worldwide.

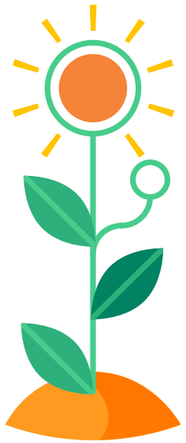
Amadeus leverages the entire Atlassian productivity stack including Jira Software, Hipchat, Confluence, and Bitbucket for collaboration between their 15,000 employees. When they saw rapid growth in their development operations, Amadeus started to notice some performance issues with their Git operations. That's when they made the decision to upgrade from Bitbucket Server to Bitbucket Data Center. With Bitbucket Data Center, they were able to add cluster nodes, improve system performance, and provide better latency for their developers working in distributed teams across multiple locations.

With Bitbucket Data Center, Amadeus is able to handle:

1,000+
pull requests a day

5,500+
developers collaborating on projects from

20+
locations



Sources and resources

You can find more information on the topics covered in this paper in our Atlassian Documentation: www.confluence.atlassian.com

Friend, Richard. “How we built Bitbucket Data Center to scale.” Atlassian Blog
https://developer.atlassian.com/blog/2016/12/how-we-built-bitbucket-data-center-to-scale/?_ga=2.81805782.1965541543.1498494935-1052122247.1498494935

“Git Hooks.” Atlassian Website
https://www.atlassian.com/git/tutorials/git-hooks

“Git LFS.” Atlassian Website
https://www.atlassian.com/git/tutorials/git-lfs?_ga=2.57180650.1965541543.1498494935-1052122247.1498494935

Heemskerk, Michael. “How we built Bitbucket Data Center to scale (part 3).” Atlassian Blog,
https://developer.atlassian.com/blog/2016/12/bitbucket-caches/

Paz, John. “Adding cluster nodes to Bitbucket Data Center.” Atlassian Documentation, https://confluence.atlassian.com/bitbucketserver/adding-cluster-nodes-to-bitbucket-data-center-776640178.html.

Paz, John. “Check for merging pull requests.” Atlassian Documentation
https://confluence.atlassian.com/bitbucketserver/checks-for-merging-pull-requests-776640039.html

Paz, John. “Disaster recovery guide for Bitbucket Data Center.” Atlassian Documentation
https://confluence.atlassian.com/bitbucketserver/disaster-recovery-guide-for-bitbucket-data-center-833940586.html.

Paz, John. “Failover for Bitbucket Data Center.” Atlassian Documentation
https://confluence.atlassian.com/enterprise/failover-for-bitbucket-data-center-687022231.html

Paz, John. “Git Large File Storage.” Atlassian Documentation
https://confluence.atlassian.com/bitbucketserver/git-large-file-storage-794364846.html

Paz, John. “Global permissions.” Atlassian Documentation
https://confluence.atlassian.com/bitbucketserver/global-permissions-776640369.html

Paz, John. “Smart Mirroring.” Atlassian Documentation
https://confluence.atlassian.com/bitbucketserver/smart-mirroring-776640046.html

Paz, John. “Workflow strategies in Bitbucket Server.” Atlassian Documentation
https://confluence.atlassian.com/bitbucketserver/workflow-strategies-in-bitbucket-server-776639944.html

“Server to Data Center: The Tipping Point.” Atlassian Website
https://www.atlassian.com/enterprise/data-center/server-to-data-center-the-tipping-point

Studman, Michael. “How we built Bitbucket Data Center to scale (part 2).” Atlassian Blog
https://developer.atlassian.com/blog/2016/12/bitbucket-adaptive-throttling/

“SVN to Git - prepping for the migration.” Atlassian Website
https://www.atlassian.com/git/tutorials/svn-to-git-prepping-your-team-migration

“What is Git.” Atlassian Website
https://www.atlassian.com/git/tutorials/what-is-git

“Why Git for your organization.” Atlassian Website
https://www.atlassian.com/git/tutorials/why-git

ATLASSIAN