ATLASSIAN + valiantys

Performance at Scale

10 million issues and beyond
with Jira Data Center

Introduction

Like many Jira Software customers, you might find yourself dealing with an instance (or multiple instances) of Jira that has grown to massive scale. Administration of the application and hardware has become a full-time job as you struggle to keep up with the demands of this mission critical application. You find yourself in need of a solution that makes your life easier and can grow with your organization. In this paper, we discuss the challenges of a growing Jira Software instance and how Atlassian's Data Center deployment option provides enterprises with a stable and scalable platform to combat these growth pains.

The testing outlined later in this paper was designed to replicate specific challenges that come with a growing instance and investigate how Data Center would help. We've outlined our tests and results below.

> **valiantys**
>
> **Much of the research, testing and methodologies referenced in this document was conducted directly and in partnership with Atlassian Platinum Solution Partner, Valiantys.**
>
> Their contributions, based on many successful years of helping Enterprise Customers succeed at scale helped guide realistic example installation sizing, as well an authentic collection of actions that happen in real time production environments. For guidance with your own installation at scale, please visit them at valiantys.com.

To summarize, Jira Data Center allows you to:

- Grow your hardware to meet the needs of teams of any size
- Maintain performance beyond 5,000 active users
- Sustain performance and response times as you increase the number of users, issues, projects, and custom fields
- Improve capacity by separating API traffic and administrative tasks, like indexing, to dedicated nodes
- Upgrade the Jira Software application without downtime
- Deploy quickly, efficiently, and powerfully in public clouds like AWS & Azure

Contents

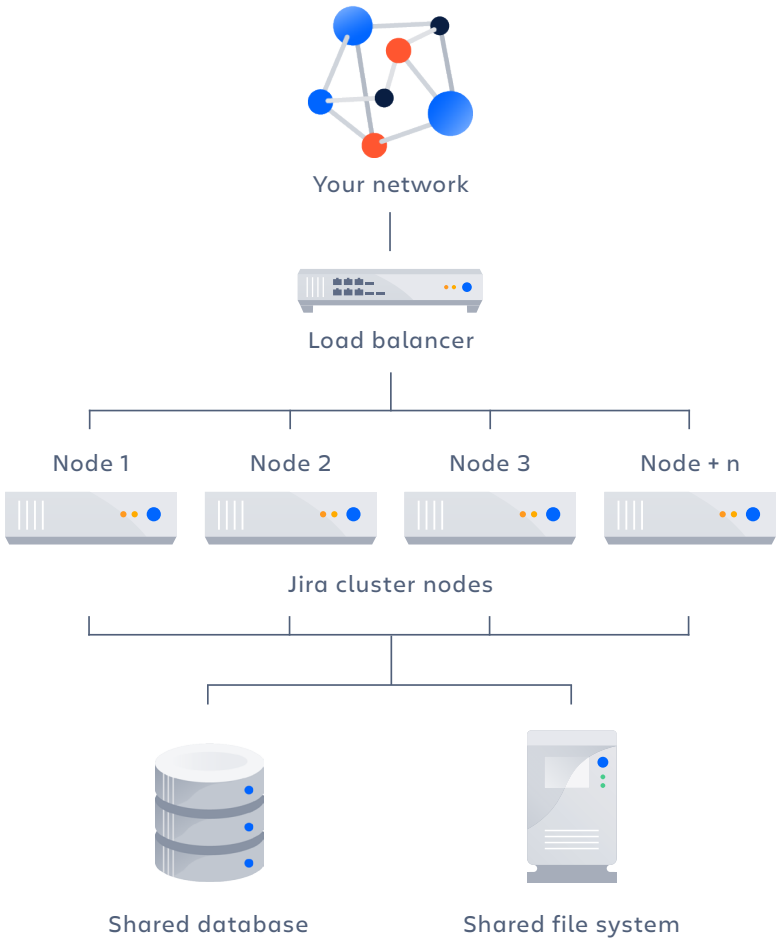
INTRODUCTION TO ATLASSIAN

Atlassian unleashes the potential of every team.

Our collaboration software helps teams organize, discuss, and complete shared work. Teams at more than 112,000 large and small organizations - including Citigroup, eBay, Coca-Cola, Visa, BMW and NASA - use Atlassian's project tracking, content creation, real-time communication, and service management products to work better together and deliver quality results on time. Products like Jira Software and Confluence help teams plan and coordinate work. Bitbucket and Bamboo give developers the platform they need to write and test great code. While Stride gives teams of all types the communication they need. These products are available to customers in three different deployment options. The cloud deployment option is a traditional SaaS based offering while Server and Data Center are both on-premise, self-hosted deployments.

INTRODUCTION TO DATA CENTER

Data Center is a self-hosted deployment option available for many Atlassian tools. Purpose built for Atlassian's largest customers, Data Center provides high availability, consistent performance at scale, and admin control and flexibility. The key to these features lies in the architecture of Data Center. With Data Center, admins can deploy an Atlassian application on an active-active cluster of servers, as opposed to the traditional single-node deployment. This clustering technology gives the application scale and resiliency. As the use of the application grows, the cluster of servers can grow with it to provide increased throughput for requests. In addition, if one server experiences downtime, there are other nodes in the cluster to take over the load, resulting in a significant reduction in downtime for users. And far fewer angry emails in your admin's inbox. Finally, Data Center gives your admins the flexibility to deploy in the way they want to, whether that's on-premise or hosting in the cloud (IaaS).



With the Data Center deployment option, user traffic enters the system through the load balancer and is distributed accordingly to the active nodes in the application cluster. Before returning the request to the user, the application will retrieve the needed data from the database and files from the shared file system.

Take a closer look at the architecture of the Data Center deployment option (above) to add color to the testing and results outlined later.

The specifics:

- Any load balancer, hardware or software based, can be used. The only requirement is that cookie-based session affinity be enabled (also known as “sticky sessions”).
- Your Data Center license does not dictate how many nodes you can have in your cluster, you can have as many or as few as you’d like. Later we’ll cover our findings on node configurations that performed best in our testing to help you make this decision.
- For the most part, the same databases can be used with Data Center as with a single server deployment. Odds are you’d be able to use the same database configuration in Data Center that you’re currently using.
- The shared file system contains plugins, attachments, avatars, profiles, and indexes. This is a NFS, SAN, or NAS file directory that is mounted to each node.

Scaling Jira Software

JOURNEY TO MISSION CRITICAL

You've probably seen or heard stories of how Jira can start in a small team and grow like wildfire. Typically, a developer installs Jira on a server under their desk and gets their team to start using it for their projects. Then, usage grows to other teams and before you know it you have an application that is critical to your teams' ability to do their jobs. It's at this point that Jira is considered mission critical to the organization, and when resiliency and performance become a priority.

THE CHALLENGE OF SCALING

When it comes to scaling Jira's user growth, many admins struggle to find the perfect balance of autonomy and governance, not to mention ensuring optimal hardware performance alongside this growth. And as usage increases, more administrative tasks are required to maintain the performance users expect.

Here's a snapshot of how some of our customers have managed growth at massive scale:

Did you know?

- The number of Jira issues among our largest customers grows an average of **23% year-over-year**
- One customer has more than **5 million** issues in a single instance
- The average number of issues per Jira instance for our large customers is **1,425,772**

But that's just Jira issues...

- The number of projects in Jira are growing an average of **17%** year-over-year
- The highest number of Jira projects in an instance is more than **2,600**
- The average number of projects for large Jira customers is **1,294**

High availability and active-active clustering

High availability is arguably the biggest reason that customers choose Data Center.

Because of the clustered architecture of Data Center, if any application node experiences an outage (planned or otherwise) the system remains up, running, and performant. This resiliency reduces the risk of downtime dramatically. For a mission critical application, downtime is a huge risk; not only is there loss in productivity but there are opportunity costs and IT costs as the team scrambles to get the system back online. A single hour of downtime for 1,000 developers could cost your organization tens-of-thousands of dollars. Data Center is the only way to achieve active-active clustering for Atlassian applications. You'll notice several references to this functionality throughout this paper as it is foundational to the value that Data Center provides.

Some of the following scenarios might not apply to your implementation (and we sure hope you aren't experiencing all of these problems).

Identify the proof points that apply most directly to you and focus on the findings in those sections to make the document a little easier to digest.

PROOF POINT | 1

Reaching the limits of your hardware

When administrators notice suboptimal performance, troubleshooting starts on the application layer. If there is no other obvious problem (like a bug) causing the degraded performance, it's likely you will need to allocate more resources to the system. Dedicating additional resources is relatively easy on virtualized hardware, but system administrators will eventually hit the limits of a single server. Furthermore, in many organizations, requesting additional hardware can be a time consuming and bureaucratic procedure - not ideal when reacting to production performance-related incidents. So, what are the signs that indicate you may have reached your current hardware limits? Let's take a look.

Signs that hardware limits might be an issue for you

When it comes to Jira's application layer, the three main system resources you need to keep your eye on are:

- **Processors or CPU allocation** | Number of CPU cores and their computing capacity
- **Memory or RAM** | Amount of physical memory allocated to the hosting environment and the corresponding allocation of Java heap space to the application itself
- **Storage or disk** | Physical capacity of the file system used by the application and more importantly the concurrent I/O throughput and latency available at any given time

MONITORING

While Jira's administration interface offers a general overview of used and available system resources, don't rely on this alone as the main source of truth. In mission critical environments, infrastructure and system administrators should have native monitoring implemented for all three main components. Should any of these be nearing their limits, causing a resource bottleneck, the standard operating procedure should already trigger alerts. If you notice any of the system resources consistently being utilized to 90-100%, this should be a clear indicator that you're reaching those limits. Typically this corresponds with a dip in application performance at times of peak usage.

At the application layer, administrators should also monitor throughput, or transactions per second (TPS), and response times. You can set up your own monitoring for the application layer either with end-to-end functionality tests or through the JMX monitoring interface.

REMEDIATION

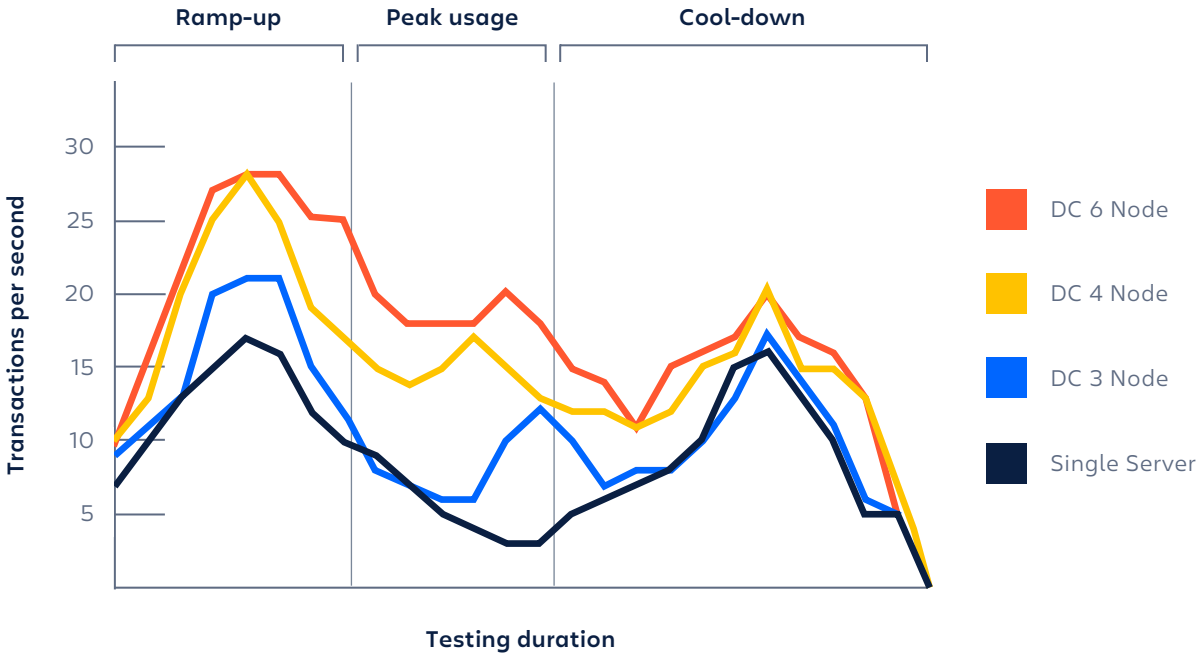
On a single-server instance you should experiment with allocating more system resources. But keep in mind that each attempt to find and remedy the bottleneck will require a brief period of downtime so that the new settings to be applied.

How Data Center can help

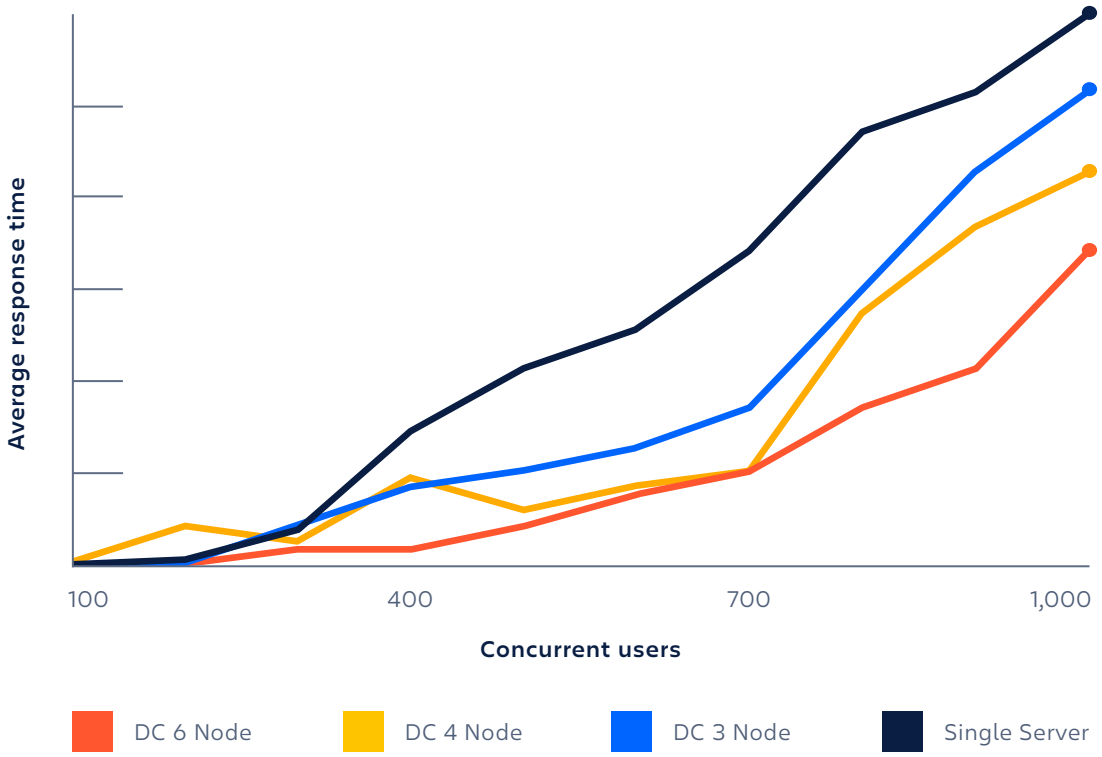
In a Jira Software Data Center environment, you can instantly react to resource bottlenecks. Instead of having to adjust the settings for the server being used, you can add additional nodes instantly - extending the overall resource capacity without affecting user downtime. Allocating additional resources to existing nodes lets you perform reconfigurations node by node, taking down each server to apply the new settings while the rest of the nodes handle transactions. The best part is that users are unaware of the operations; there's no downtime or impact to performance.

Take a look at the results from stress-tests we ran on Jira Data Center

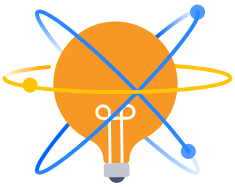
We set out to better understand the difference in performance between a single Jira Software server instance and a Jira Software Data Center instance, while examining how additional nodes in a Data Center cluster affect performance.



In this test, we ramped up the instance by adding users until 1,000 concurrent users were hammering the system. We found that Data Center consistently returned more requests per second, even during peak usage. You can see that as we added additional nodes to the cluster, the application was able to handle even more traffic.



Another way of looking at these results is in terms of how long each request takes. In other words, how long the user has to wait to get their result. We found that as the usage increases, on average Data Center returns requests faster than a single server instance. When you add additional nodes to the cluster the response times improve.



Additional references and documentation

- See a more detailed guide through the process of scaling with Data Center here
- Learn more about tuning performance for Atlassian tools from the CTO of one of Atlassian's largest Solutions Partners

With Jira Software Data Center, administrators can improve application performance and capacity by adding more nodes to the cluster. The application can be dynamically scaled to cater to increased user load, thus maintaining quick response times for your users. Now that you understand how Data Center gives your hardware the ability to grow, let's investigate a few of the most common underlying problems that cause an instance to reach its limits.

PROOF POINT | 2

Concurrent users

One primary issue with scaling Jira is the number of users on the system at one time. Usage can grow through organic adoption and consolidation. Organic adoption typically happens by word of mouth and user growth is often gradual. One problem many customers find is that administrators of various deployments often don't know one another. In response to this unmanaged growth, it is common that different instances are brought together into one consolidated instance. This kind of consolidation is advantageous for organizations aiming to improve governance while simplifying system maintenance, but will increase concurrent users instantly and dramatically.

Signs that concurrent usage might be an issue for you

The signs may become painfully obvious for administrators - and even worse, end users - when concurrent usage starts to take a toll on your system. As more users login to the system, a single server can start to get overloaded resulting in worsening response times, frequent service degradation, and even outages. While there is no single way to accurately estimate the tipping point for a specific instance, concurrent usage will cause problems sooner or later as Jira Software grows on a single server.

Below, take a look at how a single server instance handles an increasing number of active users. Even with 500 concurrent users, 1 in 5 requests to a single server can take over 30 seconds to complete. Once we scale to 750 and 1,000 users, the server reaches the point where users begin to experience unacceptably slow response times. In the context of our testing, we marked 1,000 users as the point we wanted to dig into.

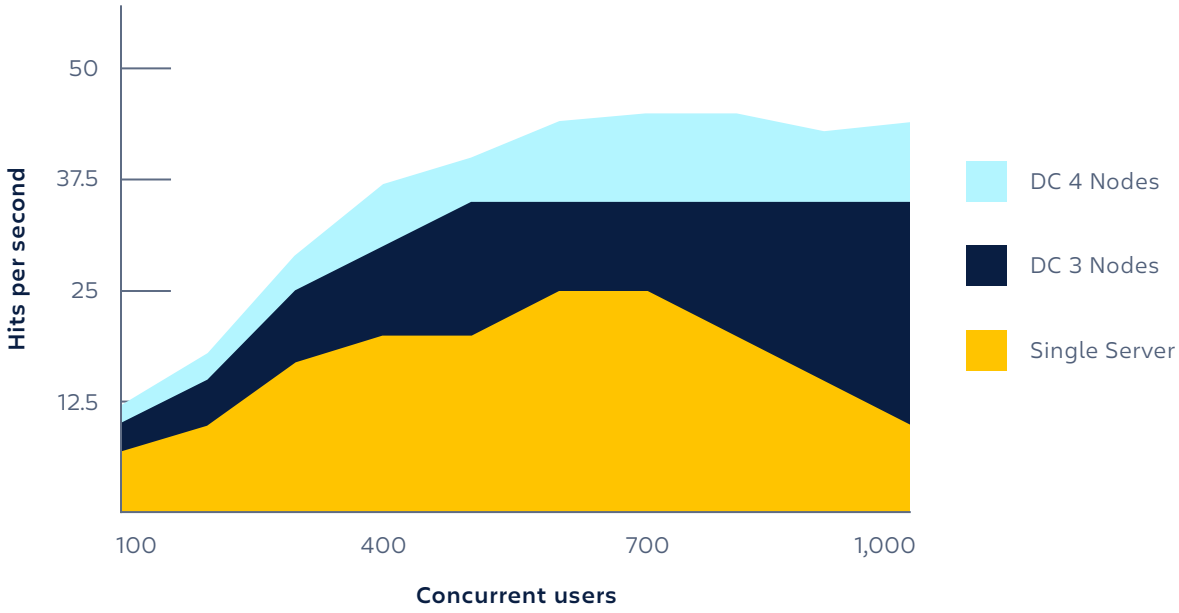
Single server response times

Concurrent users	Median response time(s)	Worst 20% of response times are over (s)
500	0.99	30.97
750	1.73	53.39
1,000	14.77	543.17

How Data Center can help

Jira Software Data Center allows you to react to the early warning signs and scale for more users without requiring downtime. When you need the extra capacity for users, the instance can be scaled up with additional nodes. Your system can grow during peak times and shrink during slow periods - scaling your Jira Software Data Center deployment up and down is seamless for your end users and the results speak for themselves.

> Keep in mind that your Data Center license does not limit or specify the number of nodes you can have, this along with the ability to grow and shrink your cluster size keeps your costs predictable.



One takeaway from this data is that organizations who want to consolidate Jira instances can do so without the threat of bogging the system down. There are a lot of benefits to consolidating onto one single instance rather than having multiple instances spread around your organization. Security, complexity, and access can all be governed much more effectively, and the required hardware is all in one place. Just remember that when you consolidate your Jira instances you are putting all of your eggs in one basket, making it even more essential that the system always be up and performing properly. Consolidating onto Jira Software Data Center gives you the governance and security you need with the resiliency you require.

PROOF POINT | 3

Succeeding as data grows

Having a high number of users is only one way that a Jira Software instance can grow. With those users come additional data: issues, projects, custom fields, workflows, attachments, and more. Both users and the associated data can have a negative impact on performance at scale. As the data inside your instance grows, every user request becomes more resource intensive. **Just rendering a single issue or board will require significantly more memory and computing power on your application server as your deployment grows in complexity.** In other words, even if the number of users remains the same, the increasing complexity will have performance and resource-consumption implications on your Jira application instance.

Signs that data complexity might be an issue for you

To simulate how complexity can grow at an enterprise, we created four scenarios of large instances. See the table below to identify where your instance might fall. Your instance won't match a scenario exactly, but find the one that's closest.

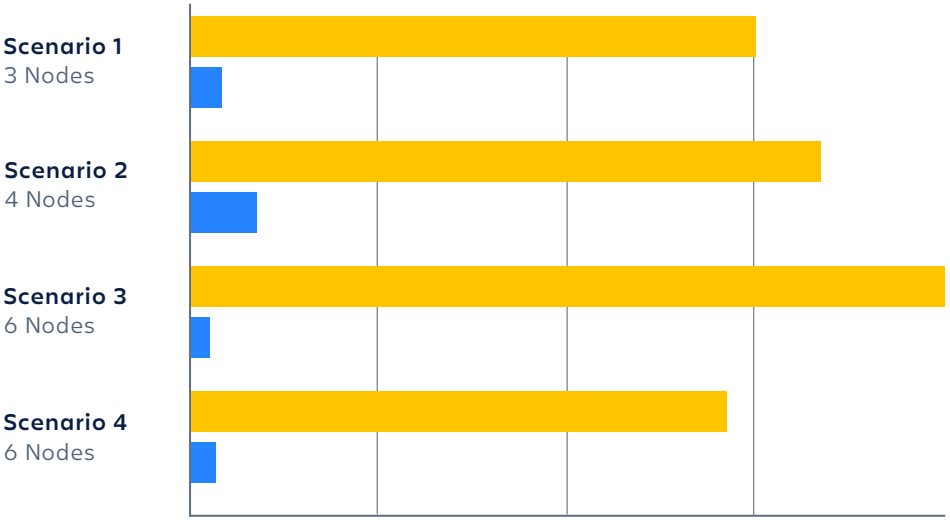
	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Issues	2,000,000	4,000,000	8,000,000	10,000,000
Projects	1,800	2,520	3,528	4,233
Custom fields	1,680	2,352	3,292	3,950
Workflows	540	756	1,058	1,269
Attachments	1,320,000	2,640,000	5,280,000	10,560,000
Comments	5,800,000	9,744,000	19,488,000	38, 976,000
Agile Boards	1,740	2,436	3,410	4,092
Users	120,000	168,000	235,000	282,000
Groups	22,500	37,000	51,000	61,000
Security levels	170	244	292	350
Permission schemes	200	288	345	414

As you grow to this scale, you may have noticed that your single server instance has started to experience degraded performance. Testing single server instances at these scales showed that average response times increased dramatically. This can cause a significant impact to the user

experience and cause the application to become nearly unusable. This is the point at which the complexity of the data has caused each request to become resource intensive and therefore, requests take longer. Additional hardware will be needed to alleviate this bottleneck and spread the requests over more resources.

How Data Center can help

The scenarios outlined above are on the larger end of customer instances and therefore perfect when comparing performance of a single server to performance on Data Center for a growing enterprise. While operating at this scale can be unsustainable for a single server instance, Data Center is able to maintain performance by spreading the added complexity across more computing resources so the system doesn't get overloaded.



Average response time

As you can see in the figure above, Data Center significantly outperformed singe server instances across every scale scenario we tested.

It's important to note that in our testing we altered the architecture of the Data Center instance to be appropriate for the scale being tested. For Scenario 1, Data Center only needed 3 nodes to return requests nearly 20 times faster than a single server, on average. Scenario 2 required 4 nodes, and for the massive scales of Scenarios 3 and 4 we ramped Data Center up to 6 Nodes. Data Center provides the horizontal scalability required to keep response times where your users expect them.

PROOF POINT | 4

Upgrades

To continue to receive the latest features and functionality, as well as fixes, version updates are a necessary process to keep an enterprise deployment thriving. The more frequently you upgrade means your users always have the latest version.

Enterprise Releases

> For those customers who take a more measured and planned approach to version upgrades, Atlassian has introduced Enterprise Releases. The goal is to allow you to upgrade to a feature release (e.g. Jira Software 7.6) that you know will get critical bug fixes, without requiring you to upgrade to the most recent feature release. There will be at least one Enterprise Release for Jira Software every year and these releases will receive bug-fix support for a longer period of time than a standard release. In other words, if you only upgrade once in a year, the Enterprise Release is probably the way to go.

Zero Downtime Upgrades with Jira Software Data Center

Jira Software Data Center lets administrators perform application upgrades without downtime or service degradation - this is called Zero Downtime Upgrades. See a visual depiction of how Zero Downtime Upgrades function to the right. The steps are similar to those in a Jira Server environment, but do no impact your users. Here's how it works:

1. Put your Jira Software Data Center instance into upgrade mode
2. *Optional* Add additional application node(s) to the cluster to sustain performance
3. Upgrade each node individually
4. Verify the upgrade via the Zero Downtime Upgrade administration screen, and in the application logs on each node
5. Commit the upgrade and commence smoke tests
6. Make a go/no-go decision. Rollback if needed or continue business as usual on the new version

The above diagram shows how rolling upgrades in Jira Software Data Center work. The upgrade to the system takes place one node at a time, without any downtime for the users. In this case, the top node has been upgraded to a new version, the middle node is currently being upgraded, and the bottom node is still on the old version. Note that user traffic can continue to be served during the upgrade process.

Benefits of Zero Downtime Upgrades

As you can see in the comparison matrix below, Jira Data Center will save you downtime during your production upgrade procedure. However, service availability is only one of the main benefits of Zero Downtime Upgrades since your administrators can now perform the upgrade during business hours. This not only eliminates the need for overtime and weekend coverage, but also ensures that your external support teams are available without added costs or planning. If something goes wrong during your upgrade you can contact Atlassian Support during regular hours, and involve teams supporting remote integrations in your smoke testing.

With the ability to dynamically add more nodes to your Jira Software Data Center deployment prior to the upgrade, you can rest assured that performance and service quality will be sustained throughout the procedure, even when you take nodes offline one-by-one to launch them with the new version.

A comparison of the single-server upgrade process vs. zero downtime upgrades in Jira Software Data Center

Step	Details	Single Server	Data Center
Preparation & testing	• Test upgrade procedure, update documentation and SOP • Perform UAT and performance tests, document smoke test steps • Finalize upgrade procedure	Partial downtime	No downtime
Backup	• Create a consistent backup from production	Planned downtime	No downtime
Upgrade	• Perform Jira upgrade	Planned downtime	No downtime
Verification	• Smoke testing • Go/no-go decision	Partial downtime	No downtime

Need a better way of communicating with your users and customers?

Atlassian uses Statuspage to communicate planned maintenance and unplanned service outages to both internal and external customers.

PROOF POINT | 5

Integrations & indexing

Integrations often become a crucial functionality of Jira Software instances. Development tools, automation, monitoring and reporting all depend on remote API calls. Since such queries are usually issued by remote systems, Jira administrators cannot rely on self-throttling of those requests. A single source malfunctioning and issuing thousands of calls within minutes can easily overload your instance. If this happens on a single-server deployment, you have very little control over service quality and performance.

When it comes to administrative tasks, such as performing a full reindex on your Jira instance, you also need to be mindful of performance implications. You obviously can't just kick off one of the most resource-intensive operations during business hours, it would severely impact usability for all users. Administrators usually schedule such tasks outside of business hours, likely over weekends, to keep the application functioning when the users need it.

Integrations and administrative tasks can take a toll on the performance of a single server instance in the same way concurrent usage can. Thanks to the administrative control that Data Center provides, organizations can alleviate this burden from their system and for their admins.

Signs that integrations & indexing might be an issue for you

INTEGRATIONS

Jira administrators can enable user access logging on their application. Having these logs available, you can use various log analyzer tools (such as the Jira access log analyzer) to classify traffic and report on it over time. If you see peaks of API traffic from specific sources which correlate to performance problems on your production deployment, you may have a smoking gun and need to investigate further.

A common troubleshooting technique is to deny traffic from a specific source on the reverse-proxy in front of Jira. This will isolate a specific problem source, but sacrifice the integration altogether during the test. Another, less intrusive method is to throttle API requests from the remote system and prevent your application from being overloaded which can causing service degradation only for the integration itself, not the entire Jira Software application.

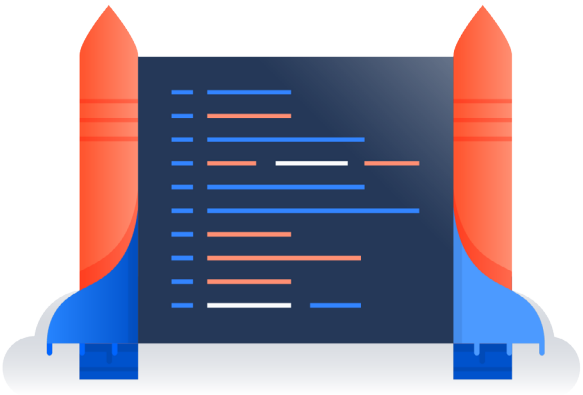
THE RE-INDEXING PROCESS

If your instance size and complexity has reached a certain point, a full reindex operation can easily take multiple hours and utilize the vast majority of available system resources on your server. You can test the resource-implications and length of this procedure on a staging environment, cloned from your production instance. This will give you a good estimation of what to expect in production and help you to better understand the effort required to maintain your instance.

How Data Center can help

Using Jira Software Data Center not only makes it easier to troubleshoot API related issues, but can also ensure that your instance won't get overloaded with API requests. Using custom load-balancer rules, you can isolate API traffic to a dedicated node. Using this technique, even if thousands of API requests flood your Jira Software instance, only the node dedicated to API traffic will be affected. This ensures uninterrupted performance for your users on the rest of the application nodes. Learn more about this type of traffic distribution here.

Similarly, you can spin up an administrative node connected to your Jira application cluster (but not your load-balancer) that you can dedicate to maintenance tasks. Your users and integrations will work on the nodes dedicated to them, while your re-index operation is performed on a separate node, not interfering with the rest of your system. Once the index is created on the dedicated node, the snapshot is distributed to the rest of the cluster through the shared storage, without causing any hiccups in performance or availability. To save on costs, you can simply turn off the administrative node when not in use - instant scalability in action.



A few stories from customers

One of the best examples of an enterprise customer taking full advantage of the administrative capacity of Data Center is Cerner's segmentation of API traffic.

Cerner noticed that their system was being overloaded with API traffic, automated calls running every second along with countless custom integrations that produced a massive number of requests. By distributing this traffic onto its own node in Data Center, Cerner was able to lighten the load for the other nodes in the cluster that were serving normal user traffic. This allowed the system to continue to scale without end users being negatively impacted by API integrations.

A Fortune 500 financial organization streamlined their indexing process by dedicating one particular node for online indexing.

Instead of having to do a background index (which can slow down the end user experience) they were able to leverage Data Center's clustering capability to have an active node perform the index for the whole system. This meant there would be no negative impact on end users and the indexes would be created as fast and as reliably as possible.

In both of these examples, the customers are working with a Technical Account Manager (TAM).

With the help of their TAMs, these customers are on the cutting edge of best practices with Atlassian tools. Keep TAMs in mind to help you make the most of your Atlassian investment.

A note on running Data Center in the cloud (Infrastructure as a Service)

A growing trend in the software industry is prioritizing cloud technology. Leveraging cloud technology can mean different things to different customers. Using Atlassian software as a service offerings is not the only way Atlassian customers are taking collaboration to new heights with cloud computing. Customers are more commonly choosing to deploy Atlassians self-hosted tools using IaaS providers. Let's quickly cover your options if leveraging the cloud is important to your organization.

The benefits of IaaS for your Data Center deployment

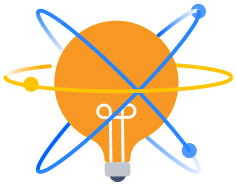
Currently Jira Software Data Center is supported in AWS and Azure. In addition to the benefits of Data Center we have already discussed, infrastructure providers like AWS and Azure can provide additional functionality such as:

- **Monitoring** | We've already showed the value of being able to add nodes instantly to your Data Center cluster, but in many on-premise environments this process still requires manual processes. Monitoring services from AWS and Azure keep an eye on the instance and provide the information you need to know when to adjust the architecture.

- **Modeling with Templates** | Admittedly, there are quite a few moving parts in a humming Data Center deployment. To get up and running you need to deploy a load balancer, database, and shared file system, in addition to the application nodes. What if you could create all of the needed architecture with one click? Cloud Formation Templates in AWS and Resource Management Templates in Azure allow you to create infrastructure from code. Configure a script to create all of your needed architecture elements at once based on the specifications you outline.

- **Architecture-level redundancy** | Migrating to Data Center ensures that your application nodes are redundant, but if you deploy on a cloud platform you can ensure that the underlying architecture is also fail-proof. Provisioning new or additional application nodes only takes seconds and you don't need to worry about hardware failures.

- **Backups & pre-production environment setup** | Cloud platforms like Microsoft Azure and Amazon Web Services offer backup capabilities to snapshot your entire environment which further improves application resiliency to outage.



For further research...

- Check out this blog comparing some of the major players in the IaaS space and how Atlassian Data Center fits in the mix
- Hear from Atlassian's Head of Server Products about the direction of Atlassian Enterprise and the cloud in his talk at last year's Summit - Atlassian Offerings for the Enterprise: Data Center and Cloud

Running your own PoC

Proof of concept deployment is a hands-on evaluation of promised features and scalability of a vendor's product, ensuring that you can natively utilize said benefits in your environment. In other words, it should prove the selected solution will do what it says in the context of your environment. This section focuses on setting up an evaluation for Jira Software Data Center to supply evidence that Data Center is the best way to solve current challenges and prepare to rapidly react for upcoming ones.

Define the scope of your PoC

To define the scope of your PoC Jira Software Data Center deployment, you will first need to:

- Identify your goal(s)
- Define questions you want to answer
- Evaluate the constraints of your current deployment

To begin, let's identify the subject matter that applies to you most directly and set goals for our PoC. When selecting your goals, keep any internal stakeholders in mind that need know of proposed changes to the system setup. What questions are they going to ask? What data will they need to sign-off on? This will help keep your PoC on a track that will be mutually beneficial for you and your stakeholders.

Which of the following factors are important to you and your stakeholders?

1. **Growing your infrastructure in a more scalable and sustainable way**
   You can only take a single server so far. Is it time for an upgrade?

2. **Managing data complexity**
   Is data on your instance growing exponentially? Do you need to clean up your existing system while laying the groundwork for future growth?

3. **Growing your capacity for concurrent users**
   Experiencing growing organic user adoption? Do you have periods of peak load? Combining instances and need to handle more users?

4. **Improving version upgrade procedures**
   Planned outages keeping you from your weekend plans?

5. **Better handling of integrations and indexing**
   Are API calls consuming vast amounts of resources? Does re-indexing need to be done outside of business hours because of the performance implications?

6. **Better leveraging cloud computing**
   Does your IT team want to benefit from Infrastructure as a Service?

How many of these apply to your situation? You may find that only one, or maybe all six, contribute to your situation.

Now, let's design a PoC that will investigate how Data Center helps you answer the questions you selected.

Design a proof of concept

The first step of designing a PoC configuration is to set up an environment identical to your production deployment. Ideally, you will be able to simulate real-life production user load to reproduce previously identified symptoms of scaling. One of the most reliable methods is to replay a realistic load pattern based on production user access logs. Atlassian customers often rely on widely used performance testing tools in conjunction with the Atlassian Performance Testing Framework for a bespoke testing platform ideal for testing their Atlassian tools.

Preparation

While ideally you would want to work in the environment where production will live with real-life data, this is often not possible due to data privacy issues. Luckily, Jira's REST API endpoints provide all the functionality required to populate an instance with dummy information. Also, the Jira Data Generator app can be used to rapidly produce data for your experiments. *Note:* the Jira Data Generator app is not officially supported by Atlassian and we specifically advise against using this app on a production environment.

Next, let's add in the factors you chose from on the previous page.

Growing your infrastructure in a more scalable and sustainable way

If the likely source of performance challenges is system resource exhaustion on your application nodes, you have two general solutions to consider. First, to **increase the capacity of your existing node(s)** and second, to **scale the application infrastructure horizontally**. There are only so many resources you can throw at a single piece of physical or virtual hardware, thus most administrators reach the point when it makes more sense to scale horizontally.

To incorporate this into your PoC

When it comes to evaluating the best solution to tackle resource bottlenecks - having reproduced the symptoms on the initial PoC environment with simulated user load - you will want to re-run those tests after either adding a new node to your application cluster or allocating more system resources to the existing configuration. You should use the same monitoring tools and techniques throughout testing to measure the effect of proposed changes in the architecture by comparing apples-to-apples. The ideal outcome of the analysis is a decrease in severity, or a complete elimination of symptoms.

THE GOAL

You will demonstrate that when your environment reaches hardware limitations on your current setup, you can dynamically scale it for sustained performance and response time.

Managing data complexity

To incorporate this into your PoC

Create a staging environment (optionally start by importing your production data) and perform a baseline load test, benchmarking your proof-of-concept architecture. Note the results and then generate an increasing number of configuration elements - projects, custom fields, permission schemes, security levels, workflows, statuses, etc - in scenarios that match your growth plans using the Jira Data Generator app. You

may want to define multiple stages/scenarios with varying number of configuration elements and issue counts. You can then run different load-test campaigns scaling the cluster up with additional cluster nodes to benchmark the performance implications. This way you can identify trends on the performance vs. growth of your deployment and the beneficial effects of scaling up to multiple application nodes.

> **A note on governance...**
> Even though growth is inevitable as usage increases, you need to make sure that the complexity of your deployment remains under control. While Data Center can reliably cater to increased end-user demand, it is not designed to alleviate problems that are a result of unmanaged data growth. In other words: improved hardware cannot make up for a poorly governed instance.

THE GOAL

You will demonstrate that the increasing complexity of your existing deployment will not have negative performance implications. Every transaction will take slightly more resources, which in combination with the increased user-load can take a toll on your hardware resources. This test will show the importance of governance policies and offer remediation options by allocating further resources to the cluster.

Growing your capacity for concurrent users

To incorporate this into your PoC

Clone your production environment to set up a proof-of-concept load-test instance. Run a baseline load-test with the average number of concurrent sessions in your production environment - the benchmark should be similar to what you see under real life user load. Upgrade your proof-of-concept environment to Data Center and provision an additional cluster node to perform the same test. If your hardware resources were maxed out when running the baseline test, you should see an improvement in response times and resource utilization. If the baseline test did not push your single-server deployment to the limits, you can start increasing the number of concurrent sessions executing the load-test (generally the increment should be a few hundred sessions each time). Once you see a degradation in performance, spin up an additional node and benchmark the results.

THE GOAL

You will identify the tipping point where the number of concurrent user sessions push your hardware resources to their limitations. With this baseline, you can then scale up your environment to multiple application nodes, demonstrating the options to proactively prepare for the increased demand of additional users.

Improving version upgrade procedures

To incorporate this into your PoC

For a realistic test scenario, set up a staging environment that's a clone of your production instance. Upgrade your single-server proof-of-concept environment to a new version and perform smoke-tests to ensure all critical functionality is working as expected. Once the test is done, restore the earlier snapshot of the PoC instance and follow the steps to migrate the instance to Data Center. After the migration, perform the upgrade process from the beginning to verify that no downtime is needed and that the application was upgraded successfully.

THE GOAL

You will demonstrate the value and feasibility of production Jira application upgrades during business hours. This should be accompanied by an operational cost saving estimation for the avoided overtime and after-hours standby of your application support teams.

Better handling of integrations and indexing

Even with the most accurate estimation of end-user load, you will need to prepare for problems caused by integrations and automated scripts. Some third-party systems may become faulty and trigger a large number of requests over a short period of time which can cause service degradation of specific application nodes, or the entire service. Dedicated application node(s) for integrations and external automated/REST API calls, as well as origin-based request throttling, on the load-balancer are a common best practice, along with allocating a specific service node for re-indexing.

To incorporate this into your PoC

Create a node in your Data Center cluster that is dedicated to API traffic. Once your load balancer has been configured to send external REST API calls to that node, your user nodes (the nodes not being used for API traffic) will see a reduction in load while the API node starts to receive traffic. From here, gauge the load on the API node to determine if API calls need to be better managed, or if another node needs to be added for API traffic. This is also a good time to monitor how the system operates without API traffic. Are the other nodes still overloaded? What effect was API traffic having on the system as a whole? This can help you come up with a better plan for handling API traffic in the future. You may want to simulate a flood of traffic, while running the baseline load-test on the rest of the nodes. This will demonstrate how rogue integration traffic leaves the response time and performance on the rest of the nodes unaffected.

THE GOAL

You will demonstrate the feasibility of dedicating specific application nodes to certain type(s) of traffic - such as REST API calls, integrations and reporting. This proof-of-concept scenario is meant to show how even DoS-level request flooding can be isolated to a specific node, not affecting the quality of service for other users.

Better leveraging cloud computing

The evaluation phase is also the perfect time to make any changes at the infrastructure level. Maybe your organization has a goal to take software to the cloud wherever possible but the Atlassian cloud offerings are not an option. Atlassian has made it easy for customers to leverage Infrastructure as a Service providers for their Data Center deployment.

To incorporate this into your PoC

Begin by identifying the IaaS provider you'd like to use for the evaluation and subsequent production instance. Currently AWS and Azure are both supported for Jira Software Data Center (other options like Google Cloud Platform can be used but are not currently supported by Atlassian). Atlassian has created a quickstart guide in AWS, and a plan in Azure, to allow Jira Software Data Center to be deployed quickly and easily.

The quickstart guide and the plans were created specifically for this purpose, to allow customers to create a one-size-fits-all deployment, and then adjust from there. When deploying to production we recommend creating your system based on your unique system requirements. The PoC is also the perfect time to test the IaaS providers' other services. For example, database services provided by AWS and Azure provide a solid database platform without the administrative overhead.

THE GOAL

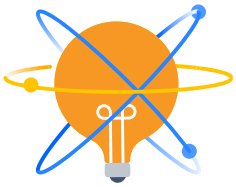
You will demonstrate the benefits of hosting your infrastructure on a public cloud provider, one-click deployments, infrastructure-level high-availability, and easy cloning/snapshot of your production environment.

Performing a proof of concept

If you are currently using a single-server installation and want to test the effects of Data Center, this is where you will install Data Center for the first time. Start by obtaining a free evaluation license from atlassian.com. This evaluation license is not limited in terms of users and can be used for 30 days. Next, we recommend creating a migration plan, followed by the technical steps of installing Data Center. Use a combination of the tools and methods outlined in this section alongside your favorite testing and monitoring tools to evaluate the effect Data Center has on the pain points you selected from the list above.

The documentation that was created during the design phase should be used as a guide to set up the proposed environment, and should be updated based on any new findings throughout the process. Break down the testing into small chunks and evaluate the results before moving on to the next portion. If tests are performed all-in-one (using a black-box approach) you risk wasting unnecessary time and resources, while making root-cause diagnosis extremely complicated and time consuming. Step-by-step execution and matching results to predefined metrics allow quick reaction to unforeseen obstacles - an agile approach testing your mission-critical Jira Software deployment with a PoC.

Throughout the testing, refer back to the initial goals and questions when documenting the metrics and results. Once complete, use the final version of the design documentation for implementing Data Center in production.



Ready for a production migration from Server to Data Center? References to check out...

- Start your production migration by reviewing the PoC documentation and results and create a migration plan with an agreed upon timeline
- Refer to our migration checklist when creating your own, to ensure a successful deployment
- Understand how Data Center works, align with the installation documentation and review how to migrate Jira to another server for performing a side-by-side upgrade from server to Data Center
- Familiarize yourself with the Jira Software Data Center FAQs

Need a hand?

Atlassian has a few Enterprise Services designed to get customers the help they need in making the most of their investment. Our network of third-party Solutions Partners provide professional services and hands-on support of your Atlassian applications. Need help performing a PoC or installing Data Center? Solutions Partners are there to help. If you are looking for more guidance toward best-practice and planning, Atlassian's Technical Account Managers provide weekly guidance so you can get your teams working in the right direction. For teams needing an elevated level of support, Premier Support offerings gives you access to a team of dedicated senior support engineers who provide increased levels of support and enhanced SLA's. The Premier Support team learns your environment to make case resolution more efficient and performs health-checks to hopefully get ahead of any issues.

In conclusion

The new journey to mission critical

Earlier, we covered how Jira Software becomes a critical application in an organization. Organic growth leads to a large and difficult to manage instance. Performance may be suffering from high load or configuration complexity and your admins are losing sleep at night (sometimes literally losing sleep to perform upgrades). When you introduce Data Center to an organization at the point when a tool is essential to success, things start to look a whole lot better.

We simulated scenarios that mimic this kind of usage at enterprise organizations to see how Data Center changes the outcome. Our stress tests pushed the architecture to the limit to emulate the challenges that customers may face when scaling. To summarize the findings that proved how Data Center addresses these challenges:

Reaching hardware limits

- Jira Software Data Center helps to improve and sustain performance with growing end-user load via horizontal scaling
- Service degradation can be avoided by transparently adding further application nodes

Concurrent Users

- Dynamic scaling of the application allows sustained performance during increased concurrent usage
- As instances grow in user count, either naturally or through consolidation, Data Center can increase throughput capacity accordingly

Instance Complexity

- Data Center is able to offset the negative impact of data growth
- As Jira Software grows to contain more issues, workflows, custom fields and other data elements, Data Center does not experience performance degradation in the way that a single server instance can

Additionally, there are other features and best practices that can make Data Center a game changer for Enterprise organizations:

Upgrades

- Planned downtime can be entirely avoided using Zero Downtime Upgrades

Integrations and Indexing

- Data Center can dedicate separate nodes for API calls, system integrations, and administrative tasks
- Having a node dedicated for re-indexing means users will not experience performance degradation during re-indexing
- Indexing can then be performed during business hours as it will not affect user experience

Going Cloud

- Get complete architectural redundancy and instant failover on the infrastructure level
- Rapid snapshots allow almost instant cloning of environments

THE BOTTOM LINE

Jira Software instances at Enterprise organizations are complex and growing more so every day. Earlier we mentioned that Atlassian's largest customers have an average of 1.4 million issues in their Jira instances. We cranked Data Center up to 10 Million issues and 5,000 concurrent users and saw a marked increase in performance over that of a single server instance. Data Center brings more to the table than just scalability, performance, and high availability. Features like Zero Downtime Upgrades and the flexibility to deploy in the cloud give admins control, while best practices like traffic shaping help them take your system to the next level.

Try Jira Software Data Center today

Evaluation licenses are available to you at no charge. This will allow you to run your own proof-of-concept to see how Data Center can address current problems, or help you avoid them in the future. Quickstart guides in AWS and Plans in Azure can help you get these evaluations up and running as fast as possible (if IaaS is an option for you) and remember chapter 9 "Running your Own POC" for our recommended approach. With a deeper understanding of the Data Center deployment option, the findings from our testing, and a plan to perform testing of your own go visit www.atlassian.com/enterprise to start an evaluation with Jira Software Data Center.

APPENDIX

Testing details

EC2 specifications

Instance Type

Operating System	Amazon Linux
Instance type	m4.4xlarge
Number of nodes	1, 2, 4, 6
Volume size	1000GB

EBS Volume Type

Root volume	/dev/xvda
Root volume size	1000GB
Block volume	/dev/xvdf
Block volume size	1000GB
Volume specs (both)	Solid state gp2 IOPS: 3600 Non-encrypted

NFS Shared Storage Drive

Instance Type

Operating System	Amazon Linux
Instance type	m4.large
Number of nodes	1 shared home volume host
Volume size	1000GB

EBS Volume Type

Root volume	/dev/xvda
Root volume size	1000GB
Block volume	/dev/xvdf
Block volume size	1000GB
Volume specs (both)	Solid state i01 IOPS: 3000 Non-encrypted

RDS Database

Database type	PostgreSQL
Database engine	PostgreSQL 9.4.15
Instance type	db.m4.10xlarge
Storage type	Provisioned IOPS (SSD)
IOPS	10000
Storage	500GB
Number of connections	100
Multi AZ	No
Publicly accessible	Yes

Notes on testing methodology

Research, test scripts, and a portion of test execution were provided in partnership with Atlassian Platinum Solution Partner, Valiantys.

- Ensure that the local home storage, local installation folder storage, and the shared storage size cater for the needs and size of the instance.
- When creating index snapshots, the Java temp directory will be used, which defaults to the Jira installation folder. Index snapshots are created on a regular basis and are not getting auto-archived, so the disk size for all three storages need to be able to hold multiple times the regular size.
- The database storage sizing will also have to account for not only the data stored in the Jira database, but also the database indexes created automatically based on those. We have installed the database client on each application node for troubleshooting purposes.
- Ensure that time synchronization across all architectural components is established (NTP or Chrony).
- In order to retroactively monitor system resource usage, the sys stat package has been installed an enabled. This was used to track CPU, RAM, disk I/O and network utilization.
- We have increased the default assigned heap memory allocation for each application node to 21GB. This value has been established via previous test and trial and is specific to the dataset(s) we worked with.

- We have increased the Tomcat default maximum thread count (in server.xml) to 300 per application node and made sure that the database connection pool and the load-balancer connection pool is aligned with this maximum setting. The database connection pool size has been increased to 100 per node and we've adjusted the database-side pool to cater for 6 nodes with this setting.

- For indexing a general Jira operation, the open file handler count default has to be increased. On most system this setting is 1024 handlers per user and we have adjusted this to a global 50000 setting. This is not only required for I/O heavy operations like reindexing, but often appear as a limit for heavily used instances.

- We tuned the number of indexing threads and various other suggested variables for indexing, described in the this documentation.

About the authors

Benjamin King
Enterprise Solution Engineer

As a member of the Enterprise Solution Engineering team at Atlassian, Benjamin spends his days advising Atlassian's largest customers on how to most effectively and efficiently deploy at massive scale. Often these conversations revolve around system architecture best practices that will result in the best possible installations. Outside of the office, Benjamin enjoys woodworking and spending time with his wife and dog in Austin, Texas.

Peter Koczan
Enterprise Solution Engineer

As a member of the Enterprise Solution Engineering team at Atlassian, Peter utilizes his six years providing Atlassian Support across every Atlassian product to help customers and Solution Partner network troubleshoot their way to success at scale. Outside of the office, Peter enjoys spending time with his growing family and discovering new technologies and new sites in Amsterdam.

About Valiantys

Valiantys is a top Atlassian Platinum Solution Partner with proven expertise in DevOps, Agile and ITSM. Our mission is to revolutionize the way teams collaborate and empower them to work smarter. We've rendered client-tailored services to over 4,000 companies, providing expert guidance on the deployment, adoption and support of Agile tools. We're a global company with Atlassian certified consultants in France, the United Kingdom, the Netherlands, Belgium, Switzerland, the United States, Canada and China.

To find out more, visit **valiantys.com.**