Demystifying Atlassian Data Center migration

Best practices for planning, preparing, and testing your way to Data Center migration success

Adaptavist

Contents

Chapter 1

Introduction

Rapid change is the new normal in today's world. Keeping pace and staying relevant is critical to the survival of every business, big or small.

But in order to behave with agility, organisations need the right systems and tools in place to help them adapt, scale, and grow with as little friction as possible.

What does all of this mean for your Atlassian software estate?

You likely rely on these tools to perform mission-critical work across your business, but how can you be sure they're enabling your ability to change and not hindering it? Are they ready to scale when you need them to?

Managing the growth of an Atlassian platform is challenging. The tools tend to first make their way into your business at an individual or team level, often under the radar of the wider organisation. As adoption grows, a patchwork of poorly-integrated tools running on different servers can emerge. Such environments create blind spots and instability and can hamper your business's ability to adapt, scale, and grow.

Sound familiar? If so, it might be time to consider a switch to the brave new world of **Atlassian Data Center**.

"

Businesses with the ability to meet a world that is changing around them are pushing well ahead of competitors.

Simon Haighton-Williams, CEO, Adaptavist

Top business drivers for migrating to Data Center

Improve integration capabilities

Your Atlassian stack is an essential enabler of your broader digital transformation goals. By moving to Data Center, you will significantly enhance your integration capabilities (e.g., single-sign-on, centralised administration). And with more consistent and well-supported tooling, the move will enable stronger collaboration between your teams — a crucial component of realising agility at scale.

Minimise operational risk

We all need downtime — except when it comes to business. System downtime can have a catastrophic effect on your service, reputation, and bottom-line.

As reliance on the Atlassian suite grows, stability can become critical to the success of your business. Yet increased risk of the financial loss incurred from outages is unavoidable when using software at scale.

Data Center mitigates the operational risks inherent in a single server platform, including performance, scalability, and availability constraints. It also addresses risks relating to non-certified add-on Apps making their way into your environment.

Reduce cost and complexity

Moving to Atlassian Data Center allows you to combine multiple server-based Atlassian instances onto a single platform, enabling ease of maintenance and central administration of all your tooling. It also presents valuable opportunities to consolidate and reduce licensing costs.

Accelerate business growth

Organisations planning for fluctuations in user numbers and data volumes need scalability that goes beyond Atlassian Server products. Atlassian Data Center is specifically designed and built to be highly scalable. This means that as your organisation grows, your Atlassian suite will scale with you.

Ready to discover more?

Migrating your Atlassian software from Server to Data Center can be a daunting prospect. With so many moving parts and things to consider, it can be challenging to know where to start.

Drawing on our experience helping large, complex customer organisations successfully transition to Atlassian Data Center, we've created this eBook to demystify the journey. Dive into the chapters to discover expert tips on planning, preparing, and testing your way to Data Center migration success.

Overview of the migration process

1
Lay out the path

Analyse and review relevant systems

2
Plan for success

Confirm scope and get your house in order, testing every step of the way!

3
Trial run

Resolve issues and produce a runbook with tasks and timings

4
UAT

Conduct User Acceptance Testing, performed by real-life users

5
Migration

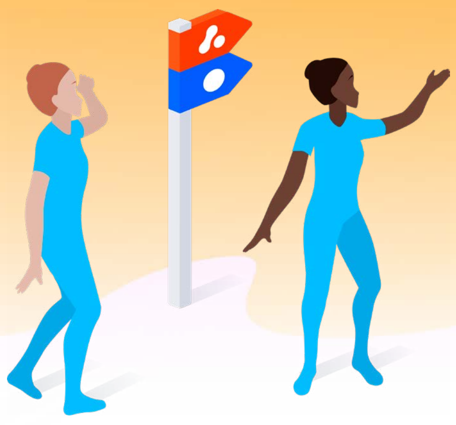
Move your data across to your new platform

6
Post-migration support

Execute final tweaks and optimise your platform

Chapter 2

Laying out the path to migration



Note that If you're migrating a single instance of an Atlassian product from Server to Data Center, the process is fairly straightforward and is outlined **here**. This eBook is focused on more complex, multiple-instance migrations.

Like any big project, the best way to start a migration of Atlassian software from Server to Data Center is to break things up into small, manageable pieces. It's important to have a realistic mindset at the outset, as this will help you craft a sensible plan of attack.

Know what migration can (and can't) do

There's a crucial caveat to observe in all migrations that's worth surfacing from the get-go: migrating a poor performing Jira Server product to Data Center alone will not enhance the performance of your instance. While a migration can bring a host of other benefits, it is not inherently a performance-tuning operation.

Proper governance, along with ongoing maintenance of configurations like custom fields, apps, and notification schemes are the essential factors in supporting the performance of your instance — regardless of whether the deployment is on Server or Data Center. For this reason, we strongly recommend cleaning up your existing data and configuration before migrating — only this will ensure the best results and performance.

Plan, plan, plan

At Adaptavist, we follow a well-proven blueprint when approaching migrations and upgrades for our customers.

It begins with a discovery phase, where we identify the key stakeholders in a project and then research and evaluate what's already in place. This is followed by the creation of a detailed plan and timeline. Then comes execution, testing, revision, and finally acceptance of the new state.

The two factors below are key in helping you to follow such a blueprint.

1. Phased approach

Before jumping into the migration of projects and data from one Jira instance to another, it's essential that you identify the people in your organisation who will be a part of the process. Stakeholders, teams, and the various different types of users across the organisation should all be defined, identified, and kept in the loop as the migration process unfolds.

We recommend starting by involving your platform owners (those responsible for the servers that the software is installed on), service owners, and system administrators for a review of the architecture. Engaging with their resources will give you a thorough picture of what the state of the system is, and will also help you understand processes around change, Jira configuration, and any non-Jira aspects such as add-on apps and other third-party integrations.

Next, identify the key Atlassian-specific stakeholders in your organisation, along with project leads, and your super-users and champions. These are the people with whom you'll define the future state of your toolset, along with what the projects will look like, governance around how the system is maintained, how it should perform, and more.

What follows is the development of your schedule. Take particular care to work with all impacted parties in planning when and how to execute any necessary downtime.

2. Big bang versus phased approach

The biggest determining factor process-wise is whether your migration should be a big-bang event, or if you'll split the migration up by moving a certain number of projects at a time. While the phased approach may seem less efficient, we recommend it for larger organisations due to a number of inherent advantages.

First, you'll have a chance to focus more closely on narrower sets of data and configuration. Being thorough in the sanitisation of these items will ensure a more stable experience as more projects are added, since you'll be able improve your process incrementally based on your learnings.

Secondly, by working with fewer teams at once you'll be able to spend more time on User Acceptance Testing (UAT), which is as an essential component of any successful migration.

Lastly, the phased approach allows you to migrate without significant downtime to all users at once — essential if you're looking to minimise business disruption.

There are some potential drawbacks to the phased approach to migration to be aware of, however. It requires greater levels of communication in order to keep teams apprised of timelines, testing, and go-live, placing a greater burden on internal teams. There's also a greater chance of uncovering tricky configuration issues when moving smaller chunks of data from instance to instance. This in itself isn't necessarily a bad thing, however, as the decisions made will inform your next migration piece.

Timelines and the D.I.Y. approach

While we're discussing planning and timelines, Atlassian states **here** that the process of migrating to a test environment, testing, and refining the process before migrating to production "typically takes about 3–6 months to execute."

This timeframe is based on reviews of customers of various shapes and sizes taking the approach of dedicating internal resources to plan and execute the migration. Internal teams frequently have competing responsibilities, and are likely to have gaps in skillset in terms of Atlassian migration expertise — hence the rather large time window that many companies report.

One key advantage of enlisting the help of an experienced Atlassian Solution Partner is that Data Center migrations, even if complex, can be executed in a matter of weeks instead of months. Platinum Enterprise partners in particular will have encountered encountered common issues many times over, enabling them to resolve them effectively or simply avoid them altogether, allowing them to be overcome in far less time.

Chapter 3

Planning for success

The planning phase of your migration presents the perfect opportunity to undertake a spring clean of your instance. This will help ensure that your migration is as pain-free as possible, mitigating the likelihood of unexpected surprises down the line.

What's in scope for migration?

A great starting point for your migration planning is gaining consensus on what you want to migrate — and perhaps more importantly, what you don't. Taking stock of the state of your system and laying out migration scope can often highlight things that don't need to be migrated, streamlining the process and eliminating unnecessary work.

This information should be collected as part of an audit where you assess the current state of your platform. Other key points to identify include:

- The infrastructure on which the existing software is hosted
- The server management policies it is governed by
- The scale and complexity of the data stored within the software
- Any configuration objects such as custom fields and workflow statuses; and
- The add-on apps and integrations that sit on top of the tools

Getting your house in order: clean-up checklist

Preparing the data you'll be migrating is a good first step, but it's important not to overlook the destination, too. This can prove more challenging than expected, given that Data Center requires a much more complicated infrastructure than Server deployments do.

At Adaptavist we're often faced with customer instances that have grown organically into complex, disordered, and often disparate platforms. To address these problems in a systematic way, we've developed the following clean-up checklist.

☑ Databases

Data format varies by database type. If we compare a Jira database running on MySQL with another running on Microsoft SQL Server with identical content, for example, the data itself appears differently.

While these data formats aren't completely alien to one another, there are some caveats and differences in how they work (e.g. incrementing numbers), and in the order in which they present data.

Database query handling is a prime example of something that may present the same result to a user, but which may be generated in very different ways from one database to another. Simply exporting data from MySQL to Microsoft SQL Server without performing some significant transformations won't work, as any database format changes must include data preparation plans in applicable migrations.

☑ User IDs

Imagine a scenario where you want to merge two Jira instances: one in the UK and the other in the US. A user has been set up on both instances because they need to work on projects in both locations. However, the login ID is different on each system to meet the naming convention of each region. In this example, a plan is needed to ensure that users like this don't end up with two accounts on the new system, and that the actions taken on both systems remain accessible.

Keep in mind that a username can be duplicated on different instances while belonging to two different users. Let's say in the US, we have an employee named Angie User whose username is "auser" and in the UK, we have an Andrew User, whose username is "auser". We don't want to merge these two — they need to maintain their distinct identities and data. This sort of data clash issue must be clarified before proceeding with your migration efforts.

☑ Custom fields

Custom fields with duplicate names will also need to be managed, as these will cause an error on merge.

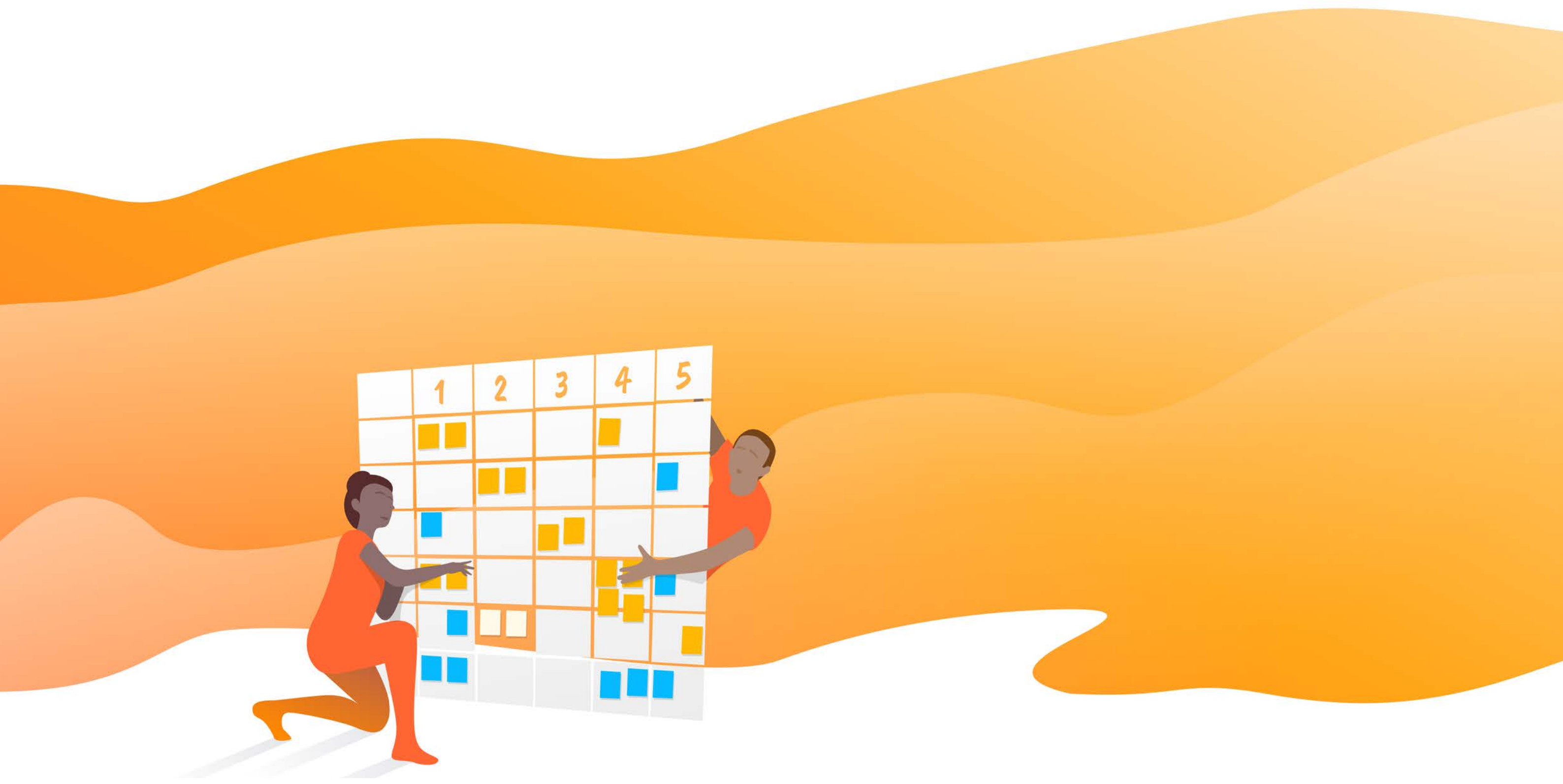
Let's say that you have a "Customer" drop-down custom field in both your UK and US Jira instances, and the options on the drop-down are different on each system. Do you rename one of the custom fields before the merge, or edit the field context? What about fields that have names spelled according to regional rules? Fields could have the same name but have different types (for example, an open text field in one system but a drop-down in another). This can cause data integrity problems when merging.

To solve these conflicts, you can rename one or both of the fields, or merge them together beforehand to keep things simple if you're able. Don't forget to communicate the change in the custom fields to your users.

There can potentially be hundreds of these kinds of custom field conflicts to work through before merging, so it's important to do your due diligence and find them programmatically here in advance.

☑ Projects and spaces

Its important to consider whether all projects or spaces are required in the new system. Do duplicate names or IDs exist for projects and spaces? If so, then you may need to rename one or more of them first. And naturally any project or space that is not actively being used should be archived or deleted.



☑ Workflows

Unpublished workflows in a source instance can sometimes cause merges to fail. As a rule of thumb, any changes to workflows should be routinely published, while any outstanding changes should be deleted or published prior to migration.

Are all the copies of workflows required? If not they should be deleted, as they may contain incorrect JQL code that will need to be fixed before the merge can continue.

☑ Filters, dashboards, and agile boards

These can cause a large number of small errors as a result of:

- Ownership by inactive or missing users
- Incorrect JQL in personal or shared filters
- Incorrect JQL in quick filters or swimlanes
- Swimlanes and quick filters referring to objects that no longer exist (such as issues, custom fields or users)

As with the previous checklist areas, now is the right time to document these issues and make a plan to address them before migrating.

☑ Default schemes

It is only possible to have one of each default scheme, and the source will override the target for merges and migrations. Therefore, any amendments that have been made to the default scheme on the target will effectively be lost once the source is merged. Proceed with caution by uncovering any potential amended default schemes to be merged and decide how to manage them.

☑ Instance health

It's vital that the source instances are healthy before migrating data from them. For this definition of instance health, we're referring to re-indexing having been completed after the most recent changes, and having the **integrity checker** and the **instance health checks** all showing as passed.

While these factors are often overlooked, it's important that they all run clear prior to migration. Remember, now is the best time to get your house in order!

System requirements for successful Data Center deployments

While you're determining the architecture of your new Data Center configuration, take this opportunity to think about the future. Consider the sort of performance you'd like to see from your system, and then give thought to what it would take to maintain that performance if a node went down.

Also consider headcount for managing the infrastructure. Are there trends in your usage patterns that will need to be addressed, such as an intensively-used apps being rolled out to more users? These factors should influence decisions around resourcing your deployment.

Start where you are

Your current applications' resource consumption is a good place to begin when considering speccing out your Data Center nodes.

Atlassian recommends that for a two-node Data Center implementation, each node should be sized identically to your current Jira instance to ensure high availability.

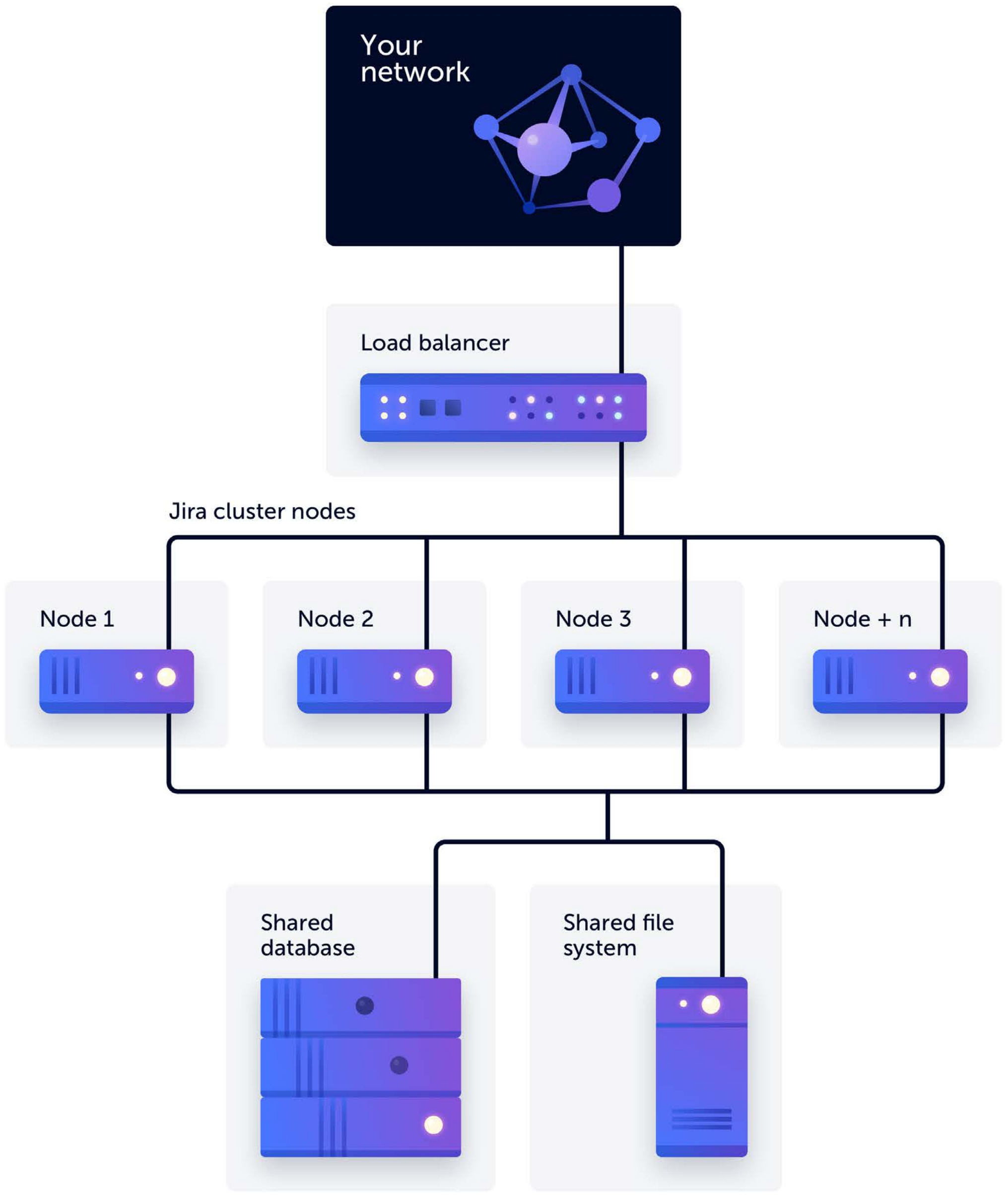
The more you scale and add more nodes, the more powerful your Data Center deployment will be. The more resources you're able to support, the more resilient the system will be. But setting a baseline that meets or exceeds your current system is important as a starting point.

Infrastructure

Atlassian is agnostic to the infrastructure that you run Data Center on, so we advise that you stick to what you know when it comes to server selection.

We recommend running Data Center applications on Linux and with a PostgreSQL database — but if your organisation already uses Microsoft Windows servers and databases and there is support infrastructure around them, don't be tempted to move unless there is already momentum in your organisation to make that change.

Data Center infrastructure model



The above diagram illustrates the optimal model of deployment for Atlassian Data Center products. Note the load balancer in front of the nodes — this is a vital component to distribute requests and prevent errors if a node fails or is taken out of service for maintenance. All nodes require access to a shared database and file system as well, and these will need to be configured to support predicted traffic.

Again we recommend sticking to what you know when it comes to load balancers. If you already use F5s, for example, then stick with those. Other good choices depending on your environment might be Apache, nginx or an AWS Elastic Load Balancer. For a file system, we recommend NFS or CIFS running on a dedicated server.

If you have multi-node master/slave databases and resilient filesystems already available in your infrastructure, you should make the most of these according to your own existing procedures (with the acknowledgement that this can be an expensive proposition). Public cloud services such as the **AWS Relational Database Service** and **Elastic File System** can give you this resilience if you don't have it already and don't want to build your own.

The Adaptavist Operate team maintains a broad portfolio of Data Center instances for our customers, and we find that in situations where there is a demand for the highest levels of performance, the benefit from investment in resilient filesystems and servers will definitely be worthwhile.

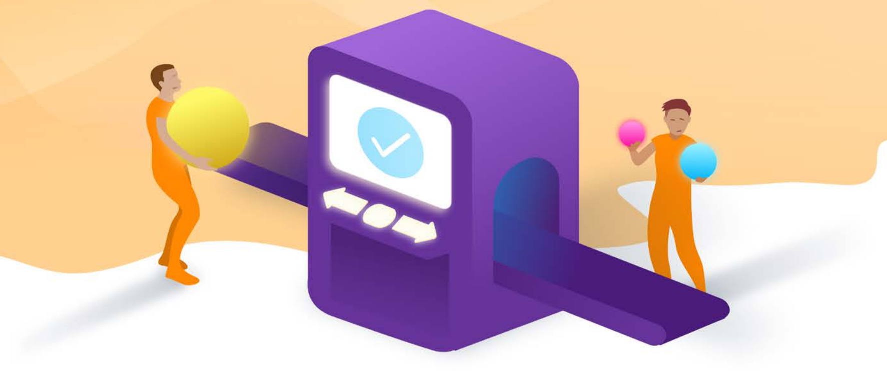
Test at every step of the way

One thing we cannot stress enough is that an abundance of caution in your process will always lead to the best results. This means test, test, test! We encourage that with every step along your road to migration, you test that the results of the action have been successful. If there are any discrepancies, address them immediately before moving forward.

Although there are many facets of migration to examine and prepare, careful attention to these items will ensure a smooth migration.

Chapter 4

User Acceptance Testing: Ignore it at your peril



User Acceptance Testing (UAT) is a crucial step in ensuring a successful bug-free migration.

Despite its significance, however, UAT is often stuffed onto the final phase of migration as an afterthought.

But, deprioritising UAT is a high-risk strategy when migrating to Atlassian Data Center. The likelihood of issues popping up during this phase of testing is high, but that's okay — finding and resolving bugs early is precisely why UAT is a step you can't afford to miss.

Why UAT is essential

Let's start by exploring what UAT is.

UAT is a set of tests, performed by a select group of users, designed to verify that everything on your new target platform works as expected. It forms the final stage of testing before you have the green light to begin your migration.

UAT differs from other forms of migration testing in that it focuses on the behaviour and perspective of the people who will actually use your platform. It helps to ensure their needs and requirements are met in real life. Inadequate or poorly executed UAT can lead to migration failure — and in the worst cases, a permanent loss of data, reputation, and revenue.

In short, UAT is essential to achieve a successful migration.

Getting it right: tips for nailing UAT

Start early

Start UAT planning as early in the migration process as possible — we'd recommend as soon as you've decided the platform you're migrating to. From here, assign a UAT lead that's accountable for the user experience end-to-end, and involve them in all aspects of migration planning.

The majority of UAT work will kick in as close to the end of the migration phase as possible, but it will ramp up quickly. This is why when it comes to UAT, preparation is critical. If you know a change will affect users, involve them early. Doing so will help identify issues fast and ensure your migration meets their needs, requirements, and expectations.

Agree your scope

While in an ideal world you could test every piece of data and functionality during UAT, in reality this is rarely possible. Instead, focus UAT on the functionality that is most critical for your users.

In the case of migration from Jira Server to Data Center, you could consider the following:

- How critical is Jira to your organisation?
- What would be the impact if it became partially or entirely unusable during or after the migration?
- Are there any specific security or regulatory requirements to consider?
- If yes, your UAT should test out compliance with these.

Get to know your users

UAT leads should know their users better than anyone else. To get there, it's helpful to think about a typical user's work routine. Some questions you might want to ask include:

- What common tasks do they perform daily?
- What are their favourite apps, and what do they use them for?
- Are they using a customised set-up? If so, why?
- What expectations do they have post-migration, and are they realistic?

The best way to get to know your users is to get out there and talk to them. Take the time to listen to their needs and priorities and watch how they interact with tools and apps daily. Understanding user behaviours and motivations will help ensure the change has a positive impact on their experience.

Involve a broad range of users

To maximise the power of UAT, involve a broad cross-section of your user base.

While in an ideal world all of your users would be engaged, this isn't realistic. Instead, involve users across a broad range of teams and roles. Give each of them access to the migration staging environment and perform regular tasks like opening agile boards, running sprints, or moving data around. Testing out everyday tasks should cover about 90% of potential use cases — and quickly highlight areas where things aren't quite right.

Make UAT part of every phase

It's important to remember that UAT is not a one-off event. It's a continuous process — particularly if you take a phased approach to migration over a more extended period of time.

Better performance is usually the key driver for moving from Server to Data Center. If early adopters experience 50% faster performance to begin with, before seeing it move to 5% once all remaining users migrate over, they won't be happy! Keeping users informed and making sure their expectations are realistic is vital.

Nobody knows your data as your users do. They'll be the first to notice if something isn't quite right

Trefor James, Head of Managed Services, Adaptavist

Chapter 5

Tooling for success

When it comes to migration it makes sense to use existing tools you are comfortable with, and that your teams can support. However, there are add-on apps specifically designed to help facilitate your migration efforts which are worth consideration if you're not already familiar with them.

Migrating data between instances

There are several apps available on the Atlassian Marketplace that help you to migrate data between instances of Jira. Adaptavist's **Project Configurator for Jira** is a leader in this space, and enables you to export data from your Jira server instances, transform this data into a portable format, and then import it to your new Jira Data Center instance.

As you might imagine, the potential for data clashes and errors is very high with any merging process. Project Configurator gives administrators a valuable toolkit for working with Jira data to reduce this risk, including the ability to export configuration as code, export only specific parts of an instance chosen by you, generate reports on potential migration errors, and ultimately lift and shift items reliably from one Jira instance to another.

To try Project Configurator for Jira for free, head to the **Atlassian Marketplace**

Project Configurator
for Jira

Add-on app data, versioning, and other challenges

Moving configurations and data from Jira add-on apps can be challenging during migration. Depending on the add-on app, it's possible that its data may be stored alongside Jira's data, either in the database or elsewhere. This means unexpected database tables created by add-on apps should be uncovered and managed as part of your approach to apps and migration.

Add-on app versioning is another factor to be aware of. We recommend sticking with the same version of the add-on app when you migrate, as this means there are fewer variables being introduced at one time. Note that you may need to relax this stance if you are upgrading to a newer Jira version and the older add-on app versions are not compatible with your target version. Differing versions may also provide significantly different features — which may be an unexpected surprise to your users. It's important to ensure this is accounted for in your migration plan.

Another important consideration is add-on configuration. Users going about their work may not actually be aware that they are using a Jira add-on for a given process or task, which can result in crucial settings for app functionality being tucked inside workflow objects or custom fields. To mitigate any issues, thoroughly check relevant documentation and settings in your add-on app portfolio.

To ensure all add-on app functionality has successfully migrated over to your destination instance, some of your users can create a dashboard or filter that uses each and every type of custom field they rely on, and use that for testing. Some common inclusions we see are calculated fields and automated options for example.

A final note on add-on apps

If you are also planning a Jira version upgrade alongside your migration (for example from Jira 7 to Jira 8), the version of Jira that you're migrating to might have added functionality that makes some of your add-on apps redundant. What's more, some apps duplicate functionality which may introduce conflicts during an upgrade.

Migration presents a great opportunity to audit add-on app usage and utilisation, and change your licensing strategy appropriately. This is important because costs can vary greatly depending on your user tier and deployment model. We often find through audits that we can consolidate and streamline licensing for our migration customers, and reduce their apps costs significantly as a result.

Did you know...

When we tested ScriptRunner and other Adaptavist add-on apps to ensure Data Center readiness, we overhauled our entire testing framework?

Read the blog

Chapter 6

Communication: Make it personal

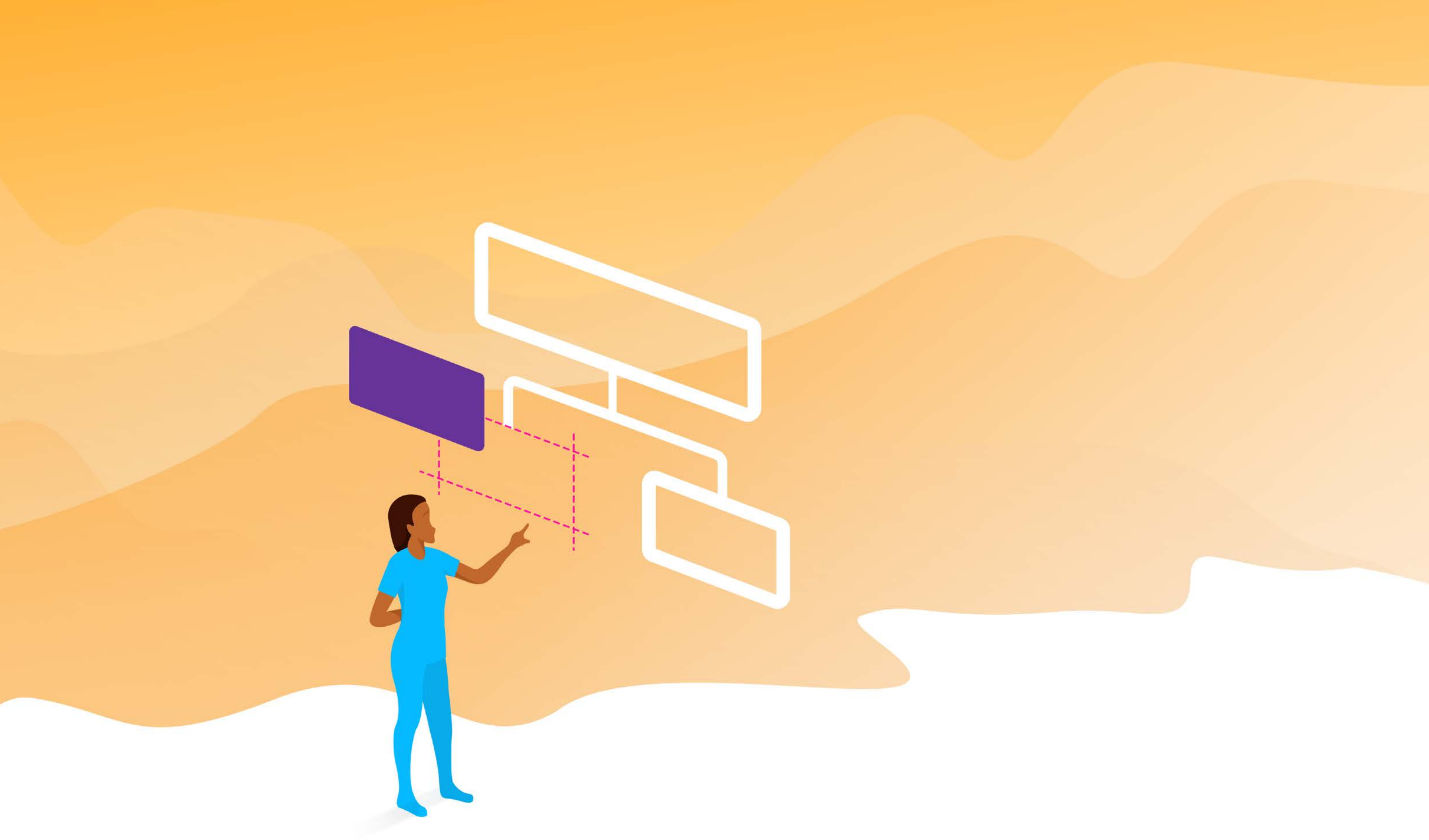
When executing your migration communication plan, follow one simple rule: tell everyone.

Don’t just update your stakeholders, testers, and project team. Tell everyone — whether it’s the CEO or the office manager. Even if some groups are less engaged, it’s important to keep them informed. If you limit communication to a few people or teams, what happens if something goes wrong on migration day? People will have no idea why they’re experiencing an issue or who to contact to get it fixed, leaving your support team overwhelmed with complaints.

And yes, tell the cleaners too. As the majority of migration work takes place in the evening or over a weekend, it’s important for them to know you will be in the office. And if they know you will be working over the weekend, they may even top up the coffee machine.

Executing your migration communication plan is all about sharing the right level of information with the right people, at the right time. The how, what, and when you communicate will depend on your organisation’s communication protocols.

To get you started, here are a few tips for communicating effectively during migration.



Be clear, consistent, and timely

All communications should help your users understand how the migration will affect them, and address their concerns to ensure a positive end-to-end experience. If you take a phased approach to migration (over several months or years), maintaining a clear and consistent flow of information is essential.

When planning your communications, think about the following questions:

- Is it clear what you are doing, and why?
- Are you explaining what's happening in a way that people understand?

Timing is also important when it comes to communicating about the migration. Start by mapping out the different migration phases, project milestones, and key dates. Next, work out and schedule the messages you'll need to share during each phase, including the most appropriate channels and audience for each message.

The goal of every migration communication is for the audience to read it and take the action you want, so in order to build trust and credibility, consistency is key.

One size doesn't fit all

When it comes to migration communications, 'one size fits all' does not work. Instead, we recommend segmenting your audience — identifying the different groups of people you need to communicate with, and deciding what they need to know, and when. Tailoring your communications in this way will help avoid overwhelming people with irrelevant information.

When planning your communications, think about the following questions:

1 Who are the critical groups of people who need to know about the migration?

2 What level of knowledge do they already have about the migration (i.e. what information can you leave out for each group?)

3 How can you make the message relevant to each group?

4 What action do you want them to take as a result of the communication?

5 How can you convince them to take action? (time to get creative and experiment with different communication techniques!)

You don't have to tell everybody, everything. Different audiences will need to know different things. While key stakeholders will need to know milestones, project plans, and preliminary migration dates, this level of detail should taper off for other people in your organisation.

Here are some examples:

Line managers: *"We wanted to let you know that Tony, Lucy, and Sam have been chosen to take part in UAT testing over the next X weeks..."*

Users: *"You're moving to a new platform and your user experience will improve as a result. Here's all the information you need, if you have any other questions let us know..."*

Cleaners and security: *'We're going to make a change to our IT systems this weekend. We would appreciate if you could ensure we have building access and fill the coffee machines for us".*

It's important to make everyone aware a change is coming, while tailoring the message to meet their needs.

Show and tell

It can be helpful to use visual cues on the new platform to notify users of the change — adding banner notifications on your Jira instance, for example, will help remind users. In the event of downtime during migration, make sure users are directed to a landing page with all the details.

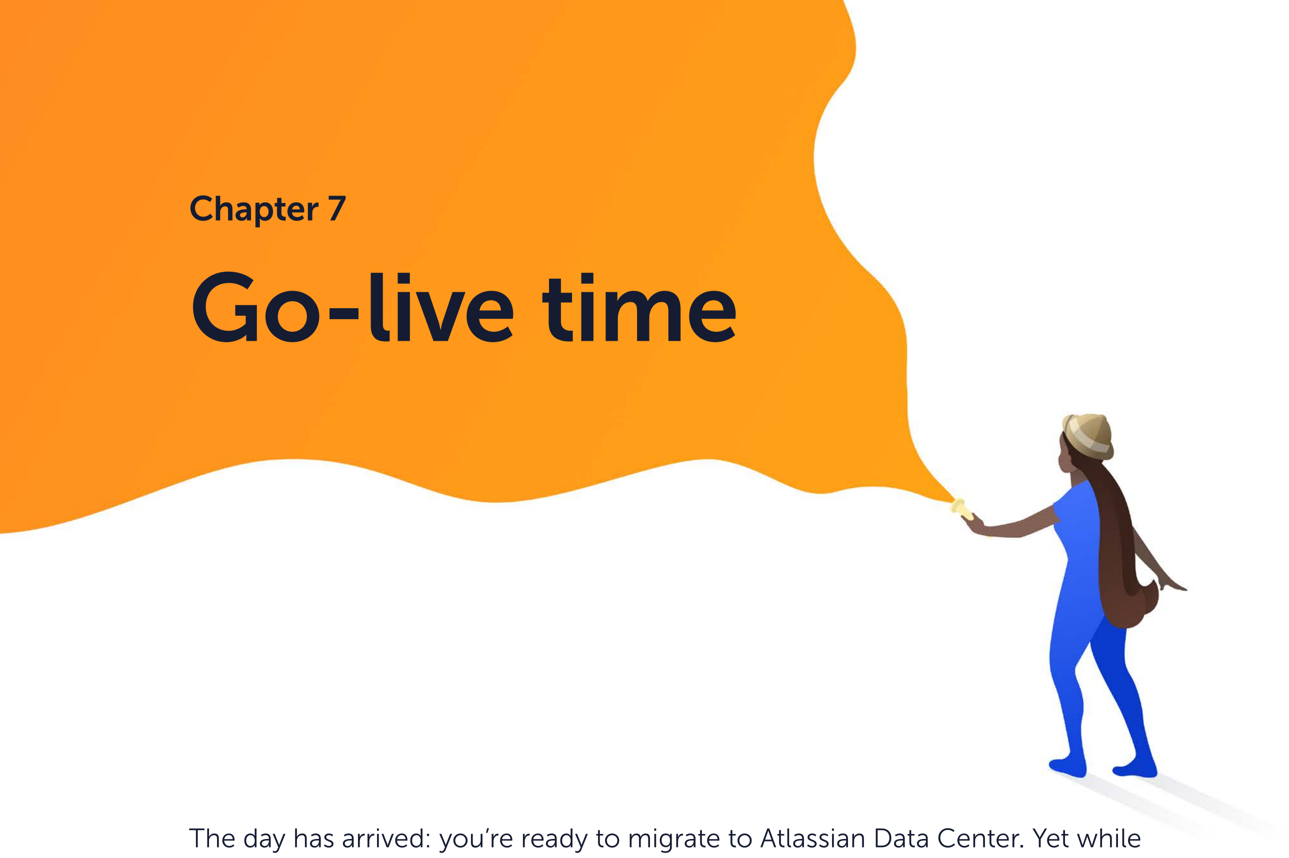
With merges, one instance will be permanently unavailable, so make sure you have a permanent holding page to guide users to the new instance.

Plan for contingency

When it comes to migrations from Jira Server to Data Center, unexpected events can, and do, happen. While unlikely, in the worst cases you may even need to execute a roll back of the entire migration. For this reason it's important to consider any contingency dilemmas you may face and prepare any draft communications in advance to keep one step ahead. This will help ensure you are ready to send the right message to the right audience at short notice. And if all goes to plan, you'll never need to use them!

Chapter 7

Go-live time



The day has arrived: you're ready to migrate to Atlassian Data Center. Yet while going-live can be nerve-wracking, rest assured this is where all your planning, preparation, and testing will pay off.

To ensure your go-live event is as seamless as possible, here are a few considerations to keep top of mind:

Timing is everything

Remember: migration should only take place once UAT is complete. Beyond that, choosing the right time to migrate will depend on your individual business needs — and whether you can afford to experience any service downtime.

Finding the best window of opportunity

It's important to ensure nothing else is running in the background during migration. If you have other programs or systems running during migration and you experience an issue, it will make it more difficult to determine what has caused it. So before you begin, make sure you freeze any batch processing or data extraction jobs that may be in the works.

Once you're ready to migrate your data and make the final move to the new platform, inform everyone that needs to know. A Confluence or Jira instance banner can be helpful to notify users when migration will start (e.g. 6pm on Friday), and to inform users what they need to do (save their work and log off by 5:30pm).

Runbook at the ready

The go-live phase of migration is where your runbook (covered in chapter 3) comes into its own. With so many moving parts to consider, following the process carefully is key, and your runbook will help you navigate through the final migration steps with ease.

Often a page in Confluence or your chosen collaboration tool, your runbook ensures that everyone involved knows exactly what will happen and when. It should also offer an outline, owner, and status for each action, along with go-live milestones to maintain momentum and keep your team on track.

What's more, if you experience any unforeseen delays, your runbook will help you assess the impact.

To roll back, or not to roll back

Before going live, ensure you have a fully-tested rollback plan in place. It will serve as a critical safety net should anything go wrong during the final stages of migration. All being well, you won't ever need to use it — but it should be there as a contingency measure to help you troubleshoot serious issues and avoid costly downtime.

Of course, if you follow a thorough UAT process, a rollback will not be needed. But, in reality, you can't always predict what will happen during migration, so it's essential to have the ability to roll back to your pre-migration environment if needed.

Post-migration support

Once migration has taken place and all of your users have migrated over, a few tweaks will usually be needed. But if test merges and UAT have been rigorous to this point, any changes should be minor.

Business as usual

If everything works as planned when you migrate, your users shouldn't notice any change — they will still be able to perform everyday tasks as they did pre-migration. The only noticeable difference should be that everything works better and faster than before!

In summary, migration and go-live success hinges on careful planning and preparation, so it's worth investing the time upfront.

Chapter 8

Conclusion: Welcome to the new world of Data Center

You’ve completed your journey to the brave new world of Atlassian Data Center and are ready to unlock new possibilities for your business

So what can you expect?

Atlassian Data Center is designed to be highly scalable. Taking you beyond a single server, it enables powerful collaboration capabilities for your teams and delivers unrivalled reliability to meet the increasing demands of your business.

Here’s what else your teams can look forward to:

Enhanced performance

With multiple nodes in place, your users should notice a boost in performance. By provisioning new nodes as you need them, you now have the ability to adapt at speed to any planned or sudden surges in demand.

Increased resilience

In reality, it’s all about what you don’t notice here. With multiple nodes in place, you’ve improved your Atlassian application’s performance. And in the unlikely event of an outage, your users can be swiftly redirected to a working node, avoiding the dreaded system error page.

Improved business continuity

After migration, you will no longer need to shut down your application to perform upgrades. Instead, you'll have the flexibility to perform 'silent' upgrades at a time that suits your users or business, avoiding the inevitable complexity involved in managing upgrades outside normal working hours.

High scalability

As part of your migration, you may have added an extra node or two to help with everyday tasks like re-indexing, or included an extra node for high-pressure Marketplace add-on apps like **EazyBi**. Having these additional nodes in place will minimise strain on your production systems, empowering your team to get the most out of the tools they use.

Centralised functionality and management

For many organisations, moving to Atlassian Data Center usually involves migrating and merging multiple systems into a single one, leading to an easier platform to manage with all of your Jira or Confluence data in one place. This takes collaboration and team-working to a new level, as previously isolated teams are now seamlessly using the same system.

Top tip: If your remote development teams are struggling with latency whilst checking out repositories from Bitbucket, adding a Smart Mirror near to where they are based will help ease their pain. The mirror will enable teams to check out code quickly and easily regardless of where your core architecture is located — yet more good news for global enterprise teams.

So there you have it — expert tips to help you stay one step ahead and demystify your journey to Atlassian Data Center. Now that you know what to expect and how to prepare, we hope it feels a lot less daunting.

New to Data Center and unsure how to get the most out of it? Or hoping to migrate but facing a few hurdles?

We’ve been supporting customers with Atlassian Data Center since the very beginning, so there’s no-one better placed to help you make the move. Whether you’re looking for simplified Data Center licensing, an expert migration partner, managed services to reduce tool burden or certified Data Center apps to use, you’re in trusted hands with us.

Tell us about your needs and find out how Adaptavist can assist in your migration

Glossary

Here are the definitions of the many terms that fall under the scope of a 'migration'.

Agile board
A representation of a set of issues visualised in columns mirroring the process flow. People can move issues through a process and instantly see where all the current work is in the process. Usually intended for supporting Agile methodologies such as Scrum and Kanban

Configuration
Settings (or groups of settings) that define the behaviour of an application or add-on app

Custom field context
Where a custom field is used in Jira, by project and/or issue type

Custom fields
Fields in Jira for differing types of data required by each specific customer, for storing their data in a way that is best for them (dates, text, select-list, user account lists, etc.)

Dashboard
A place where a person can build reporting that helps them visualise what they need, especially in Jira

Default schemes
Jira ships with a set of configuration schemes built into it, which determine configuration for notifications (emails to tell people when things change), permissions, workflows, fields, screens, and more for any given project

Destination
The instance of Atlassian software that data is being moved to

End user
A human sitting behind a customer computer, using the system remotely. Usually assumed to exclude administrators, the end user is a person who uses the system to do their job (an admin looks after it, configures it, and is often an end user themselves)

Continues

Export
Moving or cloning of data out of an Atlassian product into a data format suitable for a non-Atlassian application

Filters
A saved search in Jira. End users save a search as a “filter” so that it can be reused in it in numerous places without having to reconstruct the search every time

Import
Moving or cloning of data from a non-Atlassian application in to an Atlassian application

Instance (multiple)
Multiple systems running multiple Atlassian applications which may each be made up of multiple nodes. End users see different data in different instances

Instance (single)
A system running an Atlassian application which may be made up of multiple nodes, while appearing to be one system from an end user’s perspective

Instance health
A set of standard self-checks that a system conducts on itself to see if there are any settings that may be incorrect

JQL
Jira Query Language. This is not SQL, it is a language built for finding Jira issues. It does not report on them, just provides lists of issues for other things to report on

Latency
A measure of the time delay for data to move from one place to another. Not to be confused with bandwidth, this is mostly used when talking about networking. High latency will mean delays in data being available on a target system, so you will always be looking for low latency in your systems

Merge (aka. consolidation)
Combining configuration and data from more than one Atlassian instance in to a single Atlassian instance, resulting in the functionality and data from those instances moving to a single Atlassian instance

Migration
Any action that involves transferring all or a subset of data from one place to another. This includes moving, cloning, or transforming data from one or more Atlassian instance to one or more other Atlassian instances

Continues

Move (aka. re-platform)
Moving all or some components of configuration and data without modification from one Atlassian instance to another within the same or onto different infrastructure

Node
An individual server (either physical or virtual) running an Atlassian application in combination with other nodes to form a Data Center instance

Runbook
A set of routine procedures and tasks carried out during migration. Format can be either electronic (for example, in Confluence) or on paper, depending on preference

Source
The original instance of Atlassian software, or where data originates

Space
A collection of pages in Confluence, usually set up to gather together people and pages for a specific theme or project to enable collaborative input

Split
Separating the configuration and data in an Atlassian instance by pre-defined criteria, resulting in two or more instances each containing a subset of the configuration and data

Swimlanes
Selectable lines that go across a board, grouping cards into rows by some chosen parameter, eg. current state or assignee

Upgrades
A change of version of a system taking it to a higher version ("higher" meaning "released more recently, and intended to be better")

User ID
The unique identifier for a person's account

Workflows
A process map in Jira, containing a list of status showing where an issue is in a process, and ways to move between status

Adaptavist

Adaptavist is one of the world's leading Atlassian Platinum Solution Partners, supporting more than three quarters of the Fortune 500. We're more than 250 thinkers, makers, and doers who help organisations transform to a state where continuous change is their business as usual.

Adaptavist powers transformation by supplying technology, providing advice, and delivering change through modern, iterative approaches to development, deployment, and application lifecycle management. We're one of Atlassian's largest partners, and winner of the inaugural President's Award for Technical Excellence. Our team includes Atlassian Community champions, authors of Atlassian's official training programs, and strategic influencers in all segments of the ecosystem.

Offering a complete solution from strategic review, managed services, and a leading array of products all focused on delivering the most value from your Atlassian platform, we make application lifecycle management work.

Whether you want training for your team, to build a platform for your organisation, or to automate your existing tooling, Adaptavist is the perfect partner to guide your journey of transformation.

e hello@adaptavist.com

w www.adaptavist.com

Follow us on Twitter

Follow us on LinkedIn

ATLASSIAN | Platinum Solution Partner ENTERPRISE

ATLASSIAN | Platinum Top Vendor

ATLASSIAN | Training Partner