ATLASSIAN University

Jira Administration Part 2

Data Center and Server

1 Course Overview

22 January 2021 – Data Center and Server 8.5

What will you learn?

- Map your organization's business requirements to Jira configurations
- Create issue types, fields, and screens to make it easy for your users to get their work done
- Customize your workflows to match how your teams work
- Configure sprint and board permissions so users have appropriate access
- Create a set of standard schemes and configurations that you can use to stamp out new projects and apply to already existing projects
- Be a Jira Administrator Superhero!

To succeed here, you need to have

- Completed the following courses:
  - Jira Essentials *or* Jira Essentials with Agile Mindset
  - Jira Administration Part 1
- Downloaded the lab courseware
- Tested the lab environment

To succeed here, you should have completed the Atlassian University courses listed above or have equivalent knowledge. Also ensure you download the lab courseware and test your access to the lab environment before the course begins.

Course Overview

Mapping Your Business into Jira

Configuring Issue Types, Fields & Screens

Customizing Workflows

Configuring Board & Sprint Permissions

Applying New Configurations to Projects

Live Teach Schedule

1	Course Overview	15 minutes
2	Mapping Your Business into Jira	1 hour 15 minutes
	Break	10 minutes
3	Configuring Issue Types, Fields & Screens	2 hours 10 minutes
	Lunch	40 minutes
4	Customizing Workflows	1 hour 25 minutes
	Break	10 minutes
5	Configuring Board & Sprint Permissions	1 hour
6	Applying New Configurations to Projects	55 minutes
	Total	8 hours

Schedule for this course.

2 Mapping Your Business into Jira

Course Overview

Mapping Your Business into Jira

Configuring Issue Types, Fields & Screens

Customizing Workflows

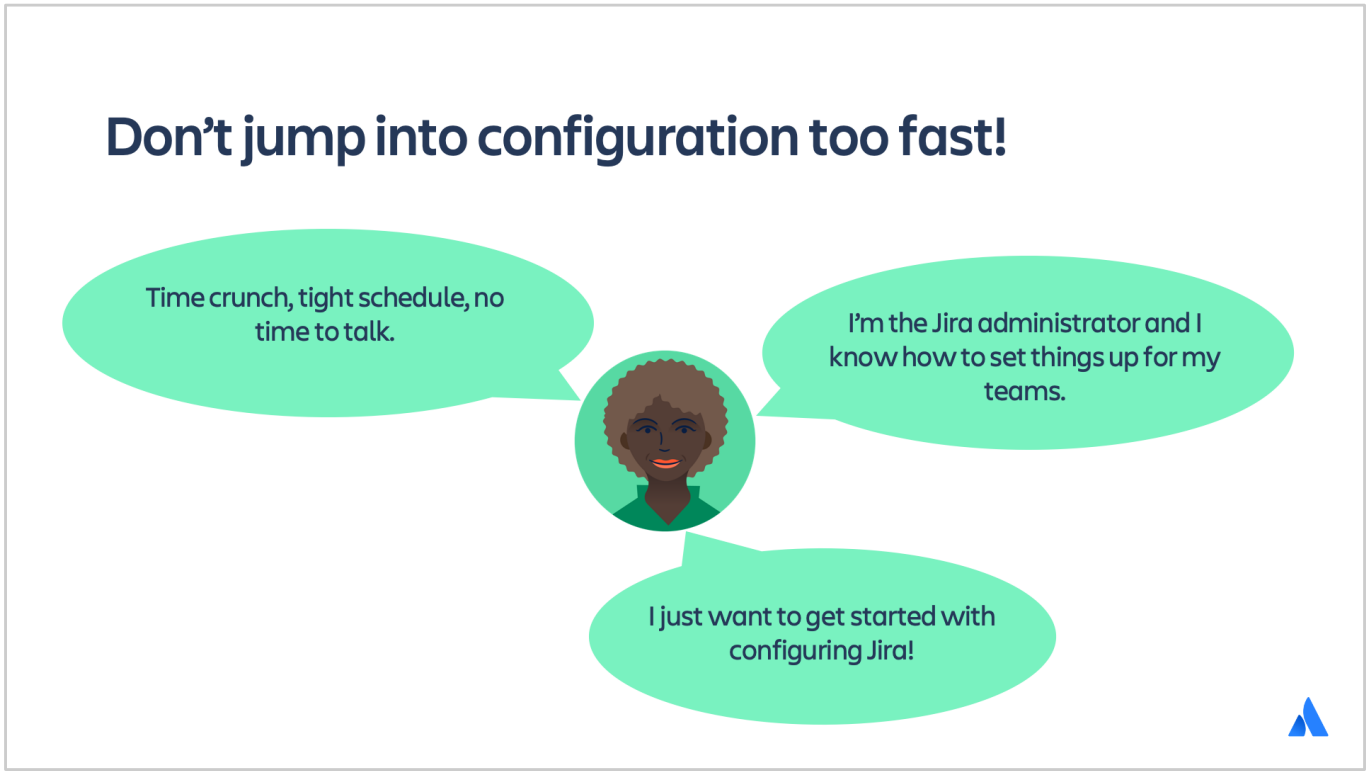
Configuring Board & Sprint Permissions

Applying New Configurations to Projects

What will you learn?

- Discover the business requirements in your organization
- Ask stakeholders the right questions
- Map business requirements to Jira configurations
- List the benefits of sharing schemes among projects
- Create a standard project to serve as a basis for new projects going forward
- Assign project roles to match how your teams work

Do you want to be your organization's dream Jira administrator?
Your organization's business can thrive, flourish, and conquer new worlds with a properly administered Jira instance. It can make the difference between super high performing teams and teams that are constantly frustrated and feel like Jira is an overhead and not a streamlining function. You need to make it as easy as possible for them to use Jira. You need to build something that reduces barriers to entry and make work easier. Jira shouldn't be the work. And you need to ensure your company gets value from Jira.
The goal of this course is to make you a Jira administrator superstar who can roll out a Jira instance that improves everyone's performance and happiness! You have the ability to shape and mold Jira to fit the way your teams work, to craft Jira to make your teams hum!
Every organization works differently, and Jira can fit the needs of any organization and any team. There are often no right or wrong ways to set things up. What we provide you with in this course are guidelines that you should work within. Beyond that, it's up to you to shape and mold the instance to your heart's content. It's a lot of responsibility, but also an extremely cool opportunity.

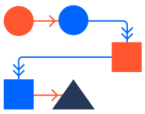


Is this Jira administrator on the right track? No.
One of the biggest mistakes you can make as a Jira administrator is jumping into Jira configuration without understanding the business requirements and how these business requirements map to Jira configuration. Often administrators are under tight schedules and it’s tempting to avoid sitting down for meetings with stakeholders. But your job will be so much harder if you don't have a clear idea of what your target configuration is before you start working in Jira. You will waste time and effort and likely have to do things over later. Your users will not be happy with Jira as it doesn’t meet their needs and your organization won’t be getting much value from Jira. Any you won’t enjoy your job!
Discovering and understanding the business requirements and then mapping these requirements to Jira configurations before you even start implementing the changes in Jira takes time. To do this successfully you will need buy-in from the whole company, especially from your management. So ensure your management is aware of the importance of this step and how it will benefit them and the whole company in the long run.

These are the goals of business analysis.
If you build it right the first time it will reduce future overhead. Plan in advance (measure twice, cut once). Understand requirements before you start building so you don't have to backtrack or undo changes. For example, If you add in the wrong custom field types and have to migrate data later, it is a nuisance and wastes time.
Design it well so people want to use it and help people get their work done more quickly and efficiently. Build a solution that doesn’t become work itself, but instead gets out of the way and let’s people focus on their actual job tasks. If we configure the tool and its projects well, people will want to use it, they will have fewer change requests, and they will need less training.

Approach

1. Discover and understand the business requirements



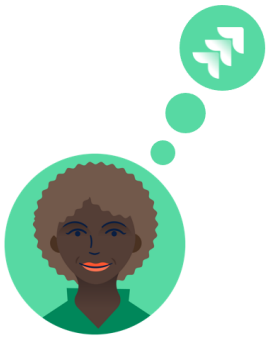
2. Map requirements to Jira configurations

3. Implement in Jira

To create a successful Jira configuration, it needs to relate to the business where it is being used. This can be an ongoing process, for example, when you need a new scheme or project, etc. Or you've had Jira for a while but up until now you've just used it for basic business processes/workflow and now you need something more complex. Actual implementation in Jira can be fast. 90% of the effort is discussing and mapping out the business processes, etc.

What you should have

- Jira technical knowledge
  - Jira's technical capabilities
  - Limitations of Jira
  - Alternative ways of implementing requirements
- Knowledge of your organization's Jira instance



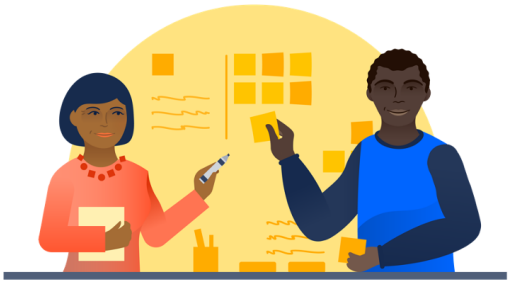
To have a successful Jira implementation you need Jira technical knowledge (which you'll get in this course) and knowledge of your Jira instance. You are the trusted advisor. People are going to ask you questions and ask for things that could be very difficult e.g. lots of customers want to rename system fields. You need to know the limitations of Jira.

- You need to come into this process knowing what all the different field types are, knowing what all the different permissions are, etc.
- You need to be able to push back on your users and suggest alternatives, which requires knowledge.
- Make sure they understand the technical capabilities of Jira and the alternatives e.g. if they say they need a list of xyz, you can make them aware that there are cascading select lists. You don't want to find out about capabilities later then and then have to migrate data.
- By knowing all of the different capabilities, you are able to draw out requirements and then map them to Jira implementations, so they don't have to be rebuilt in the future.

Someone with a very good knowledge of Jira Administration and all the configuration options needs to be involved in this process. If you, the Jira administrator, don't have that technical knowledge, get someone in who does and who can also help with business analysis.

Purpose of stakeholder meetings

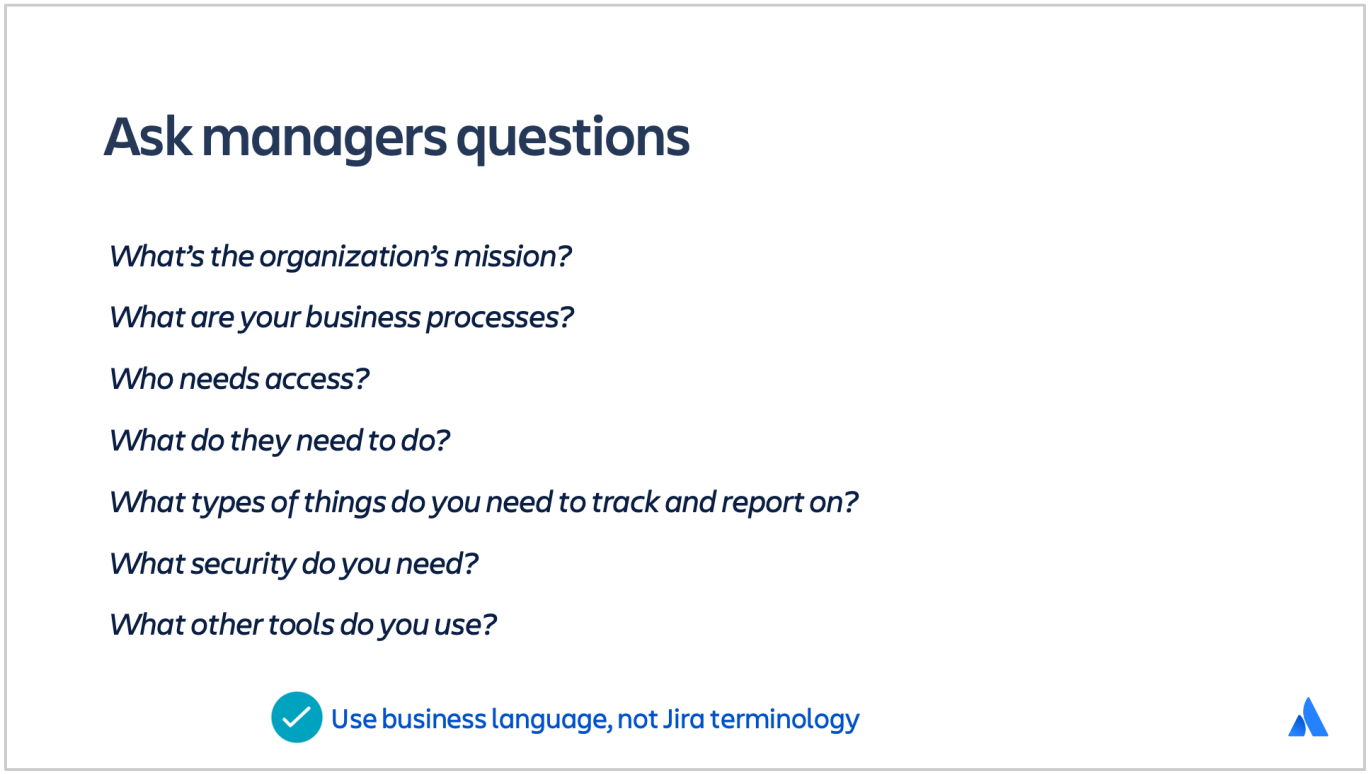
- Draw out what they need - the business requirements
- Teach them the capabilities of Jira



Spend a lot of time talking with stakeholders - it's a combination of drawing out what they really need (the business requirements) and teaching them the capabilities of Jira. (For example, they may not be aware of all the security measures that are available like issue level security.) There's no set template or list of interview questions because everyone is at a different level in terms of what they know.

Business analysis involves drawing out the needs, and prioritizing those needs, from a group of people who have disparate interests, sometimes competing interests, and competing needs. Nailing it down the first time so you've got a really good picture that everyone can agree on. You understand what the long term goals are so this thing you build is going to be good for 6 to 18 months, and not just for a few weeks.

Some things can be fluid e.g. data could be mapped to a status, or it could be a custom field, or it could be a component. There could be different ways to approach something depending on what the needs of the stakeholders are. Which is why it's so important to understand the why - why do they need it, what is the long term goal, what decisions need to be made because that's going to define how you need to build it.



Talk to stakeholders using business terminology to extract the business requirements. Stakeholders are anyone who uses Jira including business managers, team leads, project owners, project managers, scrum masters, iteration managers, developers, etc.
Questions to ask:

- First ask what the organization's vision is, what's the mission, what is it they really want to accomplish and why. Don't just focus on the mechanics as it may not be ultimately satisfying. Figure out the what and the why and deal with how later.
- Who needs access? Talk to everybody who's going to need access to the project and find out what they need to interact with. Are you sure those are all the people that need access? What exactly do they need to do? Go through project permissions, global permissions, issue level security, comments security, etc.
- What permissions do people need? What security?
- What types of things do you need to track? Do you really need to track that? What decisions are going to be driven by that data?
- What other tools do you use? For example, other Atlassian tools such as Confluence, Bitbucket, Bamboo, etc. Or third party tools such as GitHub, Perforce, etc.
- If they know something of Jira, ask: Who needs access? What's the workflow going to look like? What fields do you need? What field types? What values?
- What data do you need to collect and why do you need it (custom fields)? Then decide what type of field will best match that data need.

Ask users questions

What are the different tasks you perform in a typical day?

What is your process for those tasks?

Is your work part of a larger process within the company?

Where do you start and where do you hand work off to other people?

How long do these tasks normally take?

What sort of data do you need to collect to perform these tasks?

How can you tell when something is done?

Do you need to provide reports on your work? What, to whom, and how often?

Users, such as developers, are very important sources of knowledge about how Jira is actually used to get their work done.
Ask them questions such as:

- What does your typical day look like? What are the different tasks you perform?
- What is your process for those tasks?
- Is your work part of a larger process within the company?
- Where do you start and where do you hand work off to other people?
- How long do these tasks normally take? How long do their steps take?
- What sort of data do you need to collect to perform these tasks?
- How can you tell when something is done? Are there times when you think a task is done, but there's more work remaining that needs to be completed?
- Do you need to provide reports on the work that you do? What data do you need to report, and to whom? How frequently do you report to them on this work?

Techniques to use in stakeholder meetings

- White boarding
- Post It notes
- Flow charts
- MoSCoW prioritization technique
  - **M**ust have, **S**hould have, **C**ould have, **W**on't have
  - Discuss and decide together before implementation

During meetings in addition to white boarding and Post It notes, you can use flow charts, drawings/diagrams of fields and screens, etc.
To meet the goals, use MoSCoW approach/prioritization technique (Must have, Should have, Could have, Won't have). This gives a sense of priority and designing for future needs.

- What do they absolutely have to have to get the job done.
- What should they have - not 100% necessary but ought to be there.
- What's the nice to have (could).
- What are we going to agree together will not be implemented.

Map these out in advance on a Confluence page so you can look at them together and discuss and decide before start the implementation.

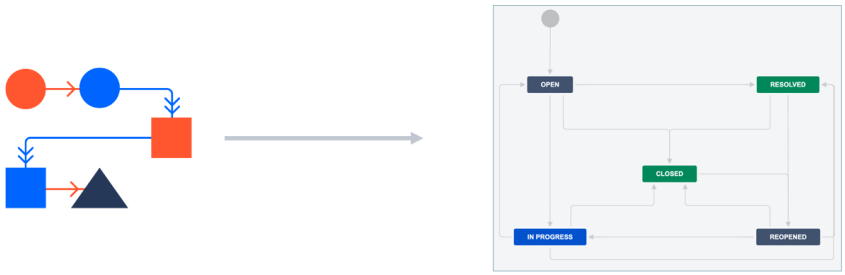
Mapping requirements to Jira configuration

Once you understand the business requirements, map these to Jira configurations

Once you understand the business requirements, map these to Jira configurations. Plan out what changes you will make in Jira before you actually implement them.
For example, from your stakeholder meetings you now have notes, Post Its and diagrams that lay out the business processes that development teams use to take their issues from creation through to completion. Now you formalize all this information into a UML or process diagram. This gives you a clear picture of what you need to implement in Jira and includes all the workflow statuses, transitions, conditions, validators, post functions, etc. for each workflow. You also know at this point which workflow needs to apply to which development issue types – one workflow for bugs, stories and feature request, and another workflow for all the other issue types.

Implementing in Jira

Once you have established how business requirements map to Jira configurations, implement them in Jira



Finally, take your business requirements that you've mapped to Jira configurations and actually implement in Jira. For example, using your UML or process diagram as a guide, you create your workflows in Jira creating all the workflow statuses, transitions, conditions, validators, post functions, etc. for each workflow. You also associate workflows with development issue types – one workflow for bugs, stories and feature request, and another workflow for all the other issue types – by creating workflow schemes.

Here we see the benefits and drawbacks of leaving Jira at the defaults (new schemes for each project) and keeping permissions very open vs. implementing control with standardized configurations and schemes and tightening up permissions.
More control will initially be more administrative work up front. But it will make it easier to maintain as you grow as there will be fewer schemes and configurations. Jira is a dynamic workspace so things will change over time and you'll need to customize which adds to administrative work. So set expectations.
On the other hand, you don't want to lock down things too tightly as users will get frustrated. If you don't allow others to have the Jira Administrators permission, make them aware of what what can be done with Jira and what tasks they need to request you to perform e.g. change workflow (if not using Simplified Workflow). If every change request has to go through a single individual, or if only one person can create a project, or a board, or a component, or a version, there may be bottlenecks and users will get frustrated.
However when you have a very large enterprise instance you need to safeguard performance so doing things like sharing schemes will help. Also security is more important in large enterprises.
You need to find the balance to suit your organization.

Jira implementation best practices

- Don't over-engineer
- Build it to meet your business requirements
- Build it so your users want to use it

We have two goals for our Jira implementation:

- Help people get their work done more quickly and efficiently
- Build a solution that doesn't become work itself, but instead gets out of the way and let's people focus on their actual job tasks

If we configure Jira and its projects well and don't over-engineer it, people will want to use it, they will have fewer change requests, and they will need less training. And you'll be their hero!

7 key steps to becoming a Jira superhero

1. Configure Jira to meet your business requirements and suit your users
2. Make fields, transitions, statuses, and screens intuitive and easy to understand
3. Keep your workflows simple
4. Set permissions that balance how your teams work and your security requirements
5. Use project roles in schemes
6. Use few schemes and share them
7. Automate as much as possible

Here are 7 steps, in no particular order, to becoming your organization's Jira superstar.
Fields, transitions, screens, and dashboards need to be intuitive and easy to understand. Don't create a lot of fields on a screen, keep it simple. Name or label global elements (custom fields, statuses, resolutions) with generic or reusable names so they can be shared with other projects. E.g. "Risk level based on finance department standards" vs. just "Risk level".
Keep your workflows simple. Too many statuses and thus too many options of available workflow transitions will confuse the user.
You want to give your teams as much permission as possible to let them get their jobs done. But you need to balance that with the organization's need for security and their risk tolerance.
Use project roles in schemes (not users or groups) so you can share schemes and have different users in each role for different projects. Ensure project administrators add their team members to these roles.
Aim to share global elements. Minimize the number to just what's needed exactly by sharing these global elements.
The typical enterprise is where the Admin has a small number of schemes that they know well. Plus some amount of fragmentation and customization for special cases.
Automate as much as is possible - let Jira handle calculations and manual functions whenever possible. All of these steps will be covered throughout this course.

Discussion

Where is your organization on the control scale?

- Tightly controlled Jira instance
- Some control
- Free for all

How large is your organization?

- < 50 users
- 50 – 1000 users
- 1000+ users

Are you getting it?

Why is it important to go through the business analysis step before implementing your changes in Jira?

The answer is on the next slide.

Did you get it?

It's important to go through the business analysis step before implementing your changes in Jira. If you dive into implementation before getting input and thinking things through, you will likely end up with a system that doesn't meet users' needs. That means you'll have frustrated users who will not use Jira properly, if at all. Your company will get little business value from Jira. Also, you'll have to redo configurations again to get them right, which will mean a lot of wasted time and effort for you. It will make your job harder in the long run.

Are you getting it?

Which of the following is most important?

a. Having an open Jira instance where users have full access to do what they want to do.

b. Having a fairly closed Jira instance where many functions can only be performed by a small number of Jira administrators.

c. Having a Jira instance that balances the needs of your users with the organization's business requirements.

The answer is on the next slide.

Did you get it?

Which of the following is most important?

a. Having an open Jira instance where users have full access to do what they want to do.

b. Having a fairly closed Jira instance where many functions can only be performed by a small number of Jira administrators.

✓ c. Having a Jira instance that balances the needs of your users with the organization's business requirements.

Answer:

C - Every organization is different and will have different requirements for control and security. It's up to you to find the right balance of freedom and control for your organization so users are happy and the organization's business requirements are met.

Takeaways

- Ask lots of questions in business language to draw out of stakeholders what they need
- Build it right the first time to avoid future work
- Build it to meet your business requirements and suit your users

Spend a lot of time talking to stakeholders (in business language) to gather the business requirements.
Go through the process of business analysis to help you build it right the first time and avoid reworking things in the future.
Design it well so people want to use it because it helps them get their work done quickly and efficiently.

Try it

Lab 2 – Starting Your Lab Environment, Creating a Standard Project & Assigning Project Roles

Exercise 1: Starting Your Lab Environment & Viewing Business Requirements

- Use the link provided to access your lab VM
- Start your Jira instance
- View Teams In Space business requirements

You won't be doing anything in Jira yet but you need to start up your lab VM so it's available for the next lab.

Teams In Space is a fictional company that specializes in space travel for teams. This is the case study we will use throughout the course. The company is distributed across the universe.

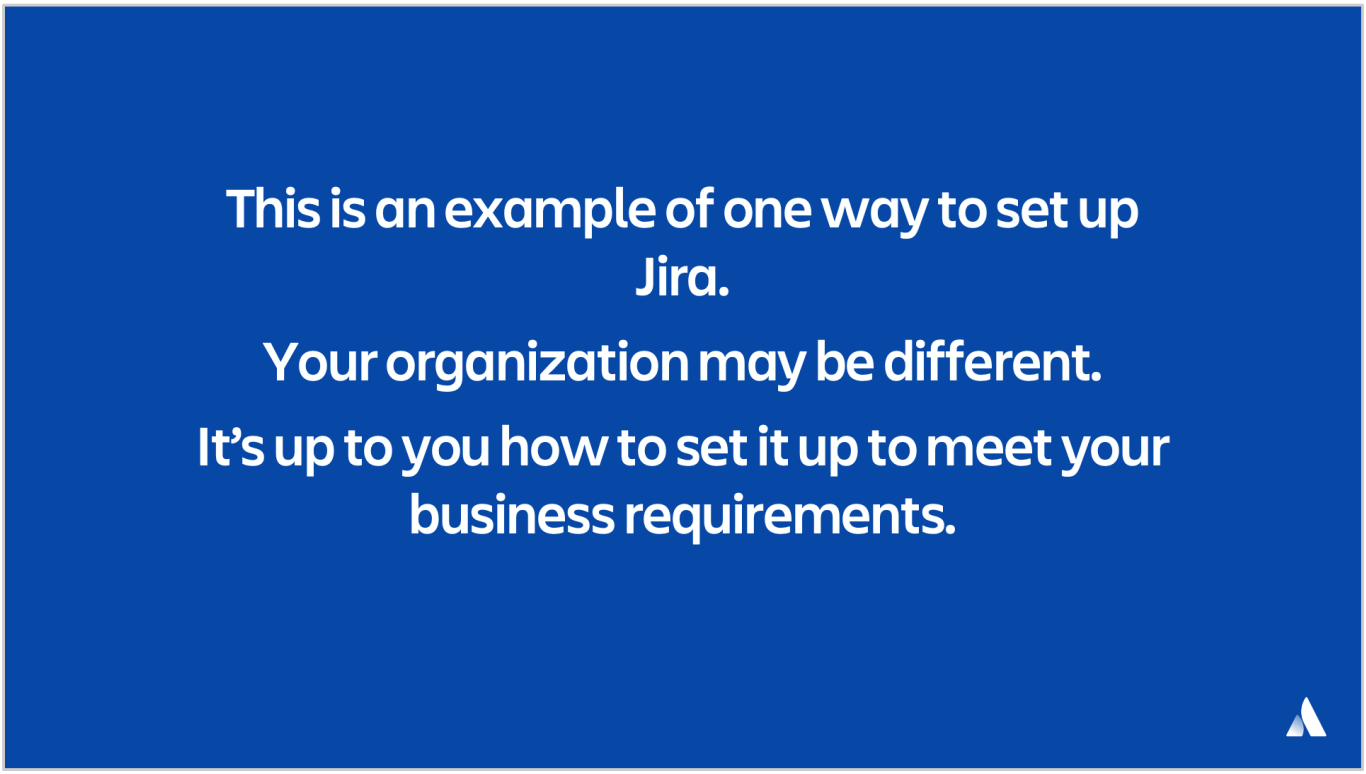
Teams In Space and their Jira instance has grown fast and now they're having problems. Up until now each development team has managed themselves - everyone has had the Jira Administrators permission and so have created their own projects which has resulted in too many schemes, workflows, statuses, etc. and no oversight. There is also no project role management. It's a bit of a mess. You are the newly appointed Jira administrator and have to clean things up and enforce some control to manage the instance as you scale even larger. Teams In Space wants more control over projects and what's created. And only you (and your team) will have the Jira Administrators permission.

There may still be a few projects that will have unique requirements and for those you will need to create some custom configurations and schemes. But the majority of Software projects have the same requirements and will use these standard configurations and schemes.

Your Mission Today

- Map the business requirements for development teams into Jira configurations
- Set group and project role membership to match how the teams work
- Create a standard set of configurations and schemes in the context of creating a new development project which will be used for creating development projects going forward
- Fix the old projects by associating the new schemes, etc., then delete old, unused schemes

These are the tasks you will be performing throughout the rest of the course.



Every organization works differently and Jira has the ability to fit the needs of any organization and any team, it doesn't make organizations and teams adapt to fit the tool.
There are often no right or wrong ways to set things up. What we provide you with in this course are guidelines that you should work within. Beyond that, it's up to you to shape and mold the instance to your heart's content. It's a lot of responsibility, but also an extremely cool opportunity.

Who does what in Scrum teams

Product Owner

- Creates issues, prioritizes the backlog, runs reports...

Scrum Master

- Creates and configures boards, creates sprints, runs reports...

Developers

- Work on issues, move issues on the board, create boards...

This is just an example of what a Software Scrum team might do in Jira. The scrum teams in your organization might be organized differently.
The Product Owner may also be the Product Manager.
The Scrum Master may also be the Software Manager, Team Lead, Iteration Manager, etc.
Find out who does what so you can set roles/permissions so the right people can do what they need.
We are focusing on Scrum in this course. For more details on Kanban, see the Getting Started with Jira Software course.

Who does what in business teams

Project Managers

- Plans and leads project, runs reports, creates issues...

Team members

- Create issues, work on issues, transition/resolve issues, etc.

This is just an example of what a business team might do in Jira. The business teams in your organization might be organized differently.
Find out who does what so you can set roles/permissions so the right people can do what they need.

Jira automatic project roles

Project role	What they can do	Who can be assigned
Project Administrator	• Assign project members to project roles • Edit project details • Edit workflow and screens	• One or more users and/or groups • Default: jira-administrators
Project Lead	• Manage the project • Is used in schemes and more	• One user • Default: Jira admin user who created the project
Default Assignee	Issue Assignee is set to this if no one is assigned when issue is created	• Project Lead or Unassigned • Default: Unassigned
Board Administrator	Configure boards e.g. change columns, configure swimlanes, etc.	• One or more users and/or groups • Default: user who created the board*

Project Lead, Default Assignee, and Project administrator are project roles that are automatically created for all projects. Board administrator is also created for all projects that contain boards. These project roles could be different people or the same person.

- Project administrator - can edit the project details i.e. name, description, etc., the project type, and the project lead. An important responsibility is to assign users to both automatic and custom project roles. If Extended Project Administration is enabled, they can also do some limited editing of project workflow and screens.
- Project Lead - person who manages the project. Can be used as the 'Default Assignee', and potentially elsewhere in Jira e.g. in schemes and workflow conditions.
- Default Assignee - when a new issue is created, if an assignee is not specified, the project's Default Assignee is used. This can be either the Project Lead, or Unassigned (the default). In Jira's general configuration, the setting ‘Allow unassigned issues’ is set to ON by default. You can turn this off if you do not wish to allow unassigned issues.
- Board administrator - when you create Scrum and Kanban projects or Business Project management in Jira Cloud, a board is created automatically. * The Board administrator for that board is, by default, the Jira administrator who created the project. Also if the Jira administrator specified a Project Lead in the process of creating the project, that user will also be the board’s administrator. Any logged in user with application access can create a new board by default and if they create a board, they will be the board administrator. The board administrator configures boards e.g. changing columns etc.

Who can configure boards?

- **Jira administrators** and **Board administrators** can configure boards
- Users who create a board are automatically the board administrator
- Project administrators cannot configure a board unless they're also a board administrator

The only users who can configure a board are the users who are specified as board administrators and Jira administrators.
When the Jira administrator creates a Scrum or Kanban project, a board is automatically created. If the Jira administrator specified a Project Lead when creating the project, that user will also be assigned the board administrator role for that default board.
If any user in the project creates their own board, they automatically become the board administrator. They would have to add the project administrator to the board administrator role for the project administrator to configure the board.

Which project templates have boards?

Project Template	Board creation
Scrum & Kanban software development	A board is created automatically
Basic software development	No automatic board but you can create one
Business projects	No boards

A board displays issues from one or more projects, giving you a flexible way of viewing, managing, and reporting on work in progress. When you create some projects, you get a board automatically, depending on the project template used.

- For Scrum and Kanban projects, you get a board automatically when the project is created.
- For a Basic Software Development project, you don't get a default board but you can create a board manually.
- Boards are not available for Business (Jira Server) projects.

For more information on using boards, see the Getting Started with Jira Software course. We'll cover board permissions later in the course.

Mapping real world team roles to Jira project roles

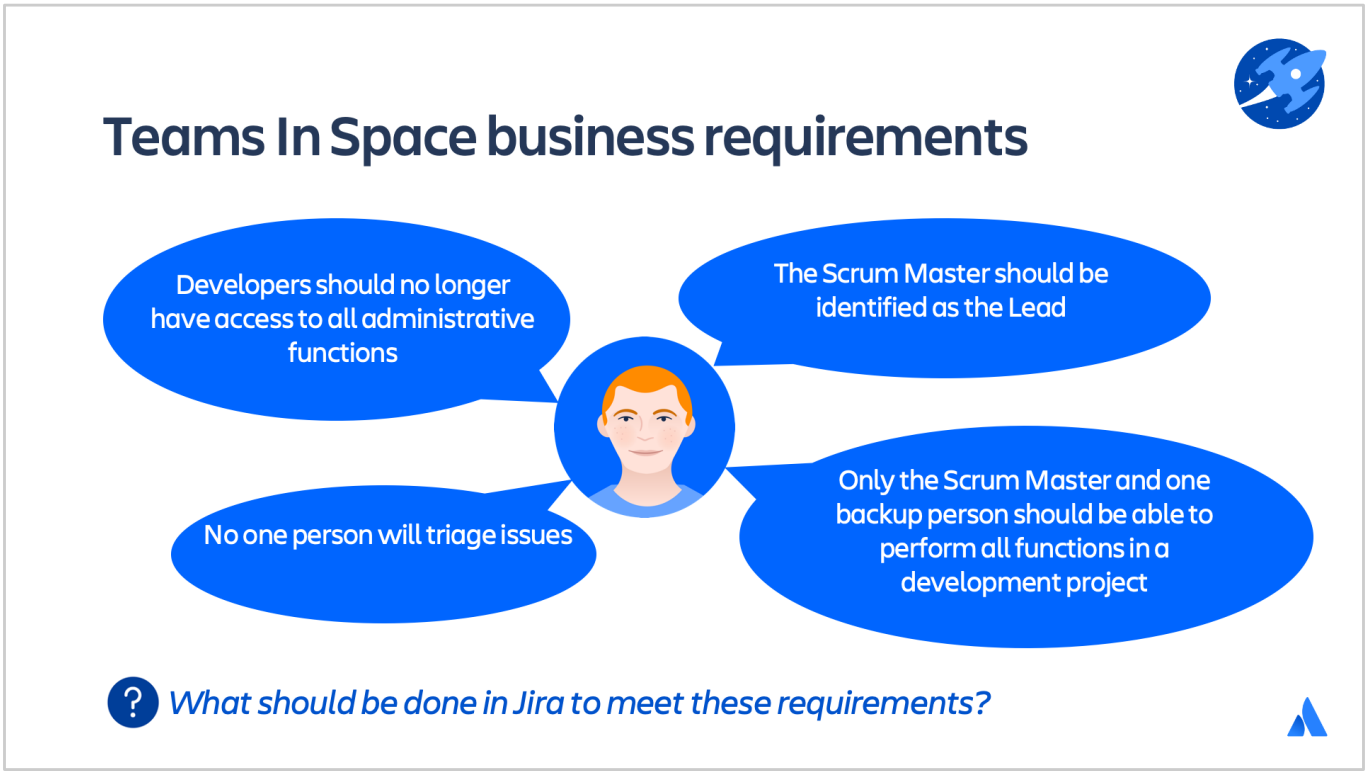
Set your team roles for how they do their work

Marketing

Project Manager is Project Lead, Project Administrator & Board Administrator

It's up to you to find out from stakeholders how their teams work and what each of them needs to do to do their work in Jira. Then you can set their roles and permissions in their projects appropriately. For example:

- Some teams may be more hierarchical, for example this Marketing team. They have one person, the Project Manager, who's in charge and who will have the most control so that person will be the Project Lead, the Project administrator and the board administrator (assuming they are using a Project Management project in Jira Cloud). Other team members will be users with the ability to manage their own issues.
- Other teams, for example agile Software teams want to manage their own projects and have everyone in the team have all the project permissions.



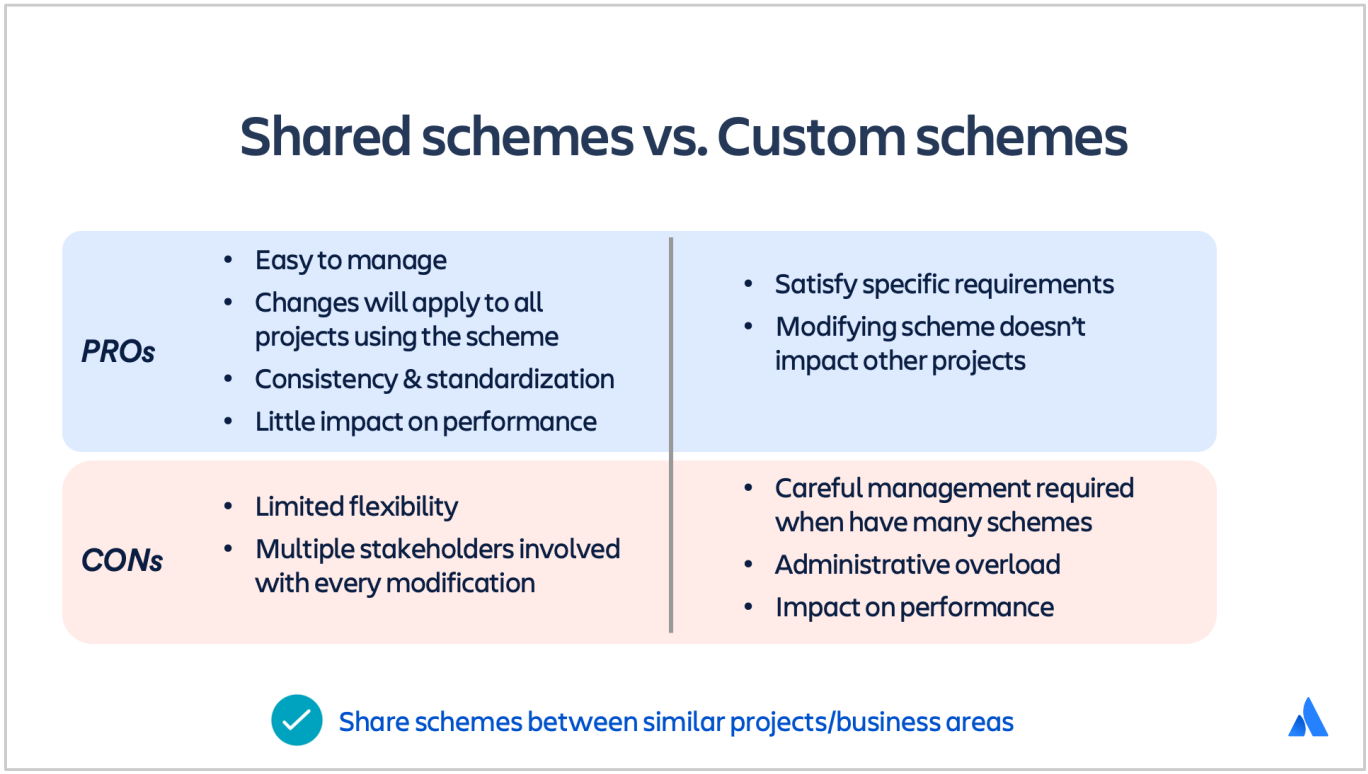
At Teams In Space the development teams were the first ones to adopt Jira. They have been self-managing and self-organizing. They create their own projects and build everything themselves in their projects. There's not a Jira administrator there's just a bunch of people with Jira Administrators permission and they build what they need and use the tool how they need it. They use an agile approach where whole team is creating issues and assigning them to themselves. There's no separate Project Manager who assigns issues. But there is a Product Owner who contributes to the backlog. It's a very flat organization. This becomes problematic as the organization's use of Jira grows larger and larger. Letting many non-administrative users have administrator permission could be dangerous.

Teams In Space wants to remove global administrator access from all developers. And for each development project they want the Scrum Master to be identified as the project lead, no one person to triage issues, and the Scrum Master and one backup person to perform all functions in their project.

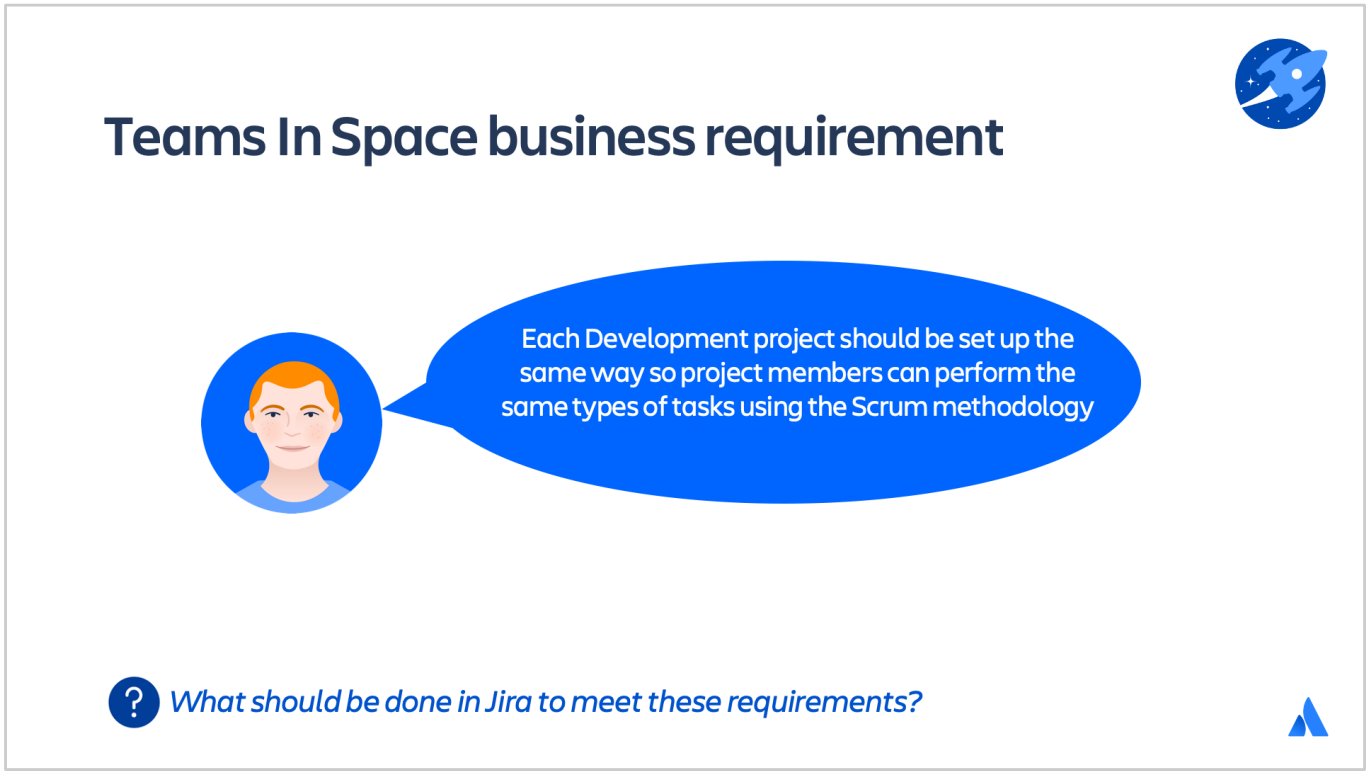
Teams In Space roles implementation

Role	Who
Jira administrators	• No developers
Project Administrators	• Scrum master • One backup person
Project Lead	• Scrum master
Default Assignee	• Unassigned
Board Administrators	• Scrum master • One backup person

At Teams in Space they decided to restrict Jira administrative functions to just a small group of Jira administrators, so developers are removed from the jira-administrators group. They also decided to restrict all project administration and default board administration functions to just the Scrum master and one backup person (in case he's on vacation, out sick, etc.). The backup person is someone who has good Jira knowledge.
The scrum master should be identified in the project as the Lead so he/she can be the point of contact for non-project members.
The Default Assignee is set to Unassigned.



Sharing Schemes reduces administrative overhead. Share schemes as much as possible where you believe that the shared configuration will stay in sync. Even if things go out of sync, most schemes can be easily copied and reverted if necessary. Do what you need to do to meet your business needs but try to re-use schemes and workflows. The more that are standardized across the whole system, the fewer features there are and fewer screens and fields to be loaded every time an issue is created, edited or closed. This will mean better performance. A large enterprise installation with hundreds of projects and with schemes for every project will definitely see a performance hit. Reducing the number of schemes will improve performance. Having good governance over the creation of schemes becomes more important as your Jira instance grows. Too many schemes means too much administrative overhead, too much complication and poor performance.
Another advantage to sharing schemes is consistency. For example, developers switching to a different development project will have the same permissions, will get the same notifications, will see the same issue types, etc. This makes it easier for them.
The disadvantages with shared schemes are that you have some limited flexibility and as changes have an impact on all projects that use shared schemes a lot of stakeholders (Product Managers, etc.) will need to be involved with every change.
Having as few schemes as possible and share them is the key to scaling Jira. Share schemes between similar projects types or departments or align them with project categories.

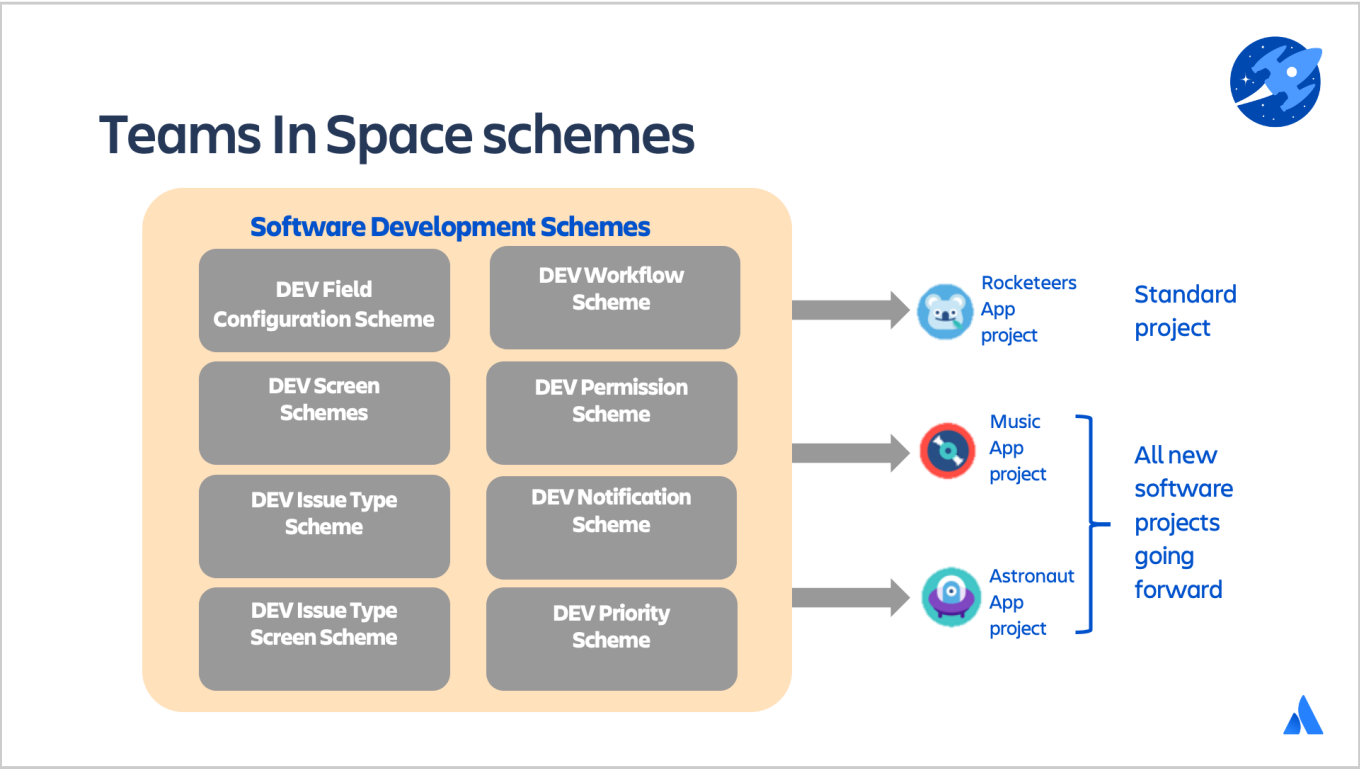


For the Teams in Space implementation, we are looking to standardize the development-type projects. We want to set them up in the same way so that project members can perform the same types of tasks using the Scrum methodology.

Teams In Space Jira implementation

1. Create a new standard Scrum software development project using **DEV** key
2. Configure the project to create a standard set of schemes and configurations
3. Share the standard project's configuration when creating Scrum projects
4. Apply the new schemes and configurations to in progress Scrum projects

You've gathered the business requirements for the Software teams at Teams In Space and you've mapped them into Jira configurations.
Now you'll create configurations and schemes in the context of creating a new standard Software project which will be used to stamp out new Software projects going forward. There may still be a few projects that will have unique requirements and for those you will need to create some custom configurations and schemes. But the majority of Software projects have the same requirements and will use these standard configurations and schemes.
For this project we'll choose the template type that matches the business requirements, in this case Scrum software development. We'll also give the project a standard project key, DEV, so the schemes are easily recognizable as being the development standard schemes. Recall that the project key is prepended to all project scheme names.
Later in the course, you'll apply the new schemes and configurations to in progress Scrum projects where appropriate to standardize work and to reduce the number of schemes and configurations in your Jira instance. By sharing schemes among all the Scrum projects you can make the instance easier to administer and easier to use.
You'll also do some tidy up work and delete unused schemes, etc.



At Teams In Space, they want a set of schemes to be shared for all development projects. They can use shared schemes since the general requirements (process, data etc.) are basically the same. This makes things easier to manage in a growing instance, and especially in a large organisation. You can ensure that each development project has the same schemes and so each development project works in an expected way, which follow your organisational processes. This means that everyone will know how to use Jira to support the expected processes for that project type and helps to get the same project outcomes.

Are you getting it?

What are the advantages to using shared schemes for many projects?

a. Less administrative overhead
b. Satisfy project specific requirements
c. Possible faster performance
d. Modifying schemes doesn't impact other projects
e. Consistency and standardization
f. All of the above

Answers are on next slide.

Did you get it?

What are the advantages to using shared schemes for many projects?

- ✓ a. Less administrative overhead
- b. Satisfy project specific requirements
- ✓ c. Possible faster performance
- d. Modifying schemes doesn't impact other projects
- ✓ e. Consistency and standardization
- f. All of the above

Answers:

1. a, c, and e

See how it's done

Take a tour of the Teams In Space Jira instance to see some of the challenges they are facing as they grow

In this demo we'll see what can happen if you have a poorly administered Jira instance that's grown over time.

Takeaways

- Assign project roles to match how your teams work
- Use few schemes and share them
- Share schemes between similar projects/business areas

Teams In Space is a fictional company that specializes in space travel for teams. This is the case study we will use throughout the course. The company is distributed across the universe.
Teams In Space and their Jira instance has grown fast and now they're having problems. Up until now each Development team has managed themselves - everyone has had the Jira Administrators permission and so have created their own projects which has resulted in too many schemes, workflows, statuses, etc. and no oversight. There is also no project role management. It's a bit of a mess. You are the newly appointed Jira administrator and have to clean things up and enforce some control to manage the instance as you scale even larger. Teams In Space wants more control over projects and what's created. And only you (and your team) will have the Jira Administrators global permission.

Try it

Lab 2 – Starting Your Lab Environment, Creating a Standard Project & Assigning Project Roles

- Exercise 2 – Creating a standard project
- Exercise 3 – Assigning project roles

In your Lab Workbook follow either the High Level Instructions or the Detailed Instructions for each exercise.

3

Configuring Issue Types, Fields & Screens

Course Overview

Mapping Your Business into Jira

Configuring Issue Types, Fields & Screens

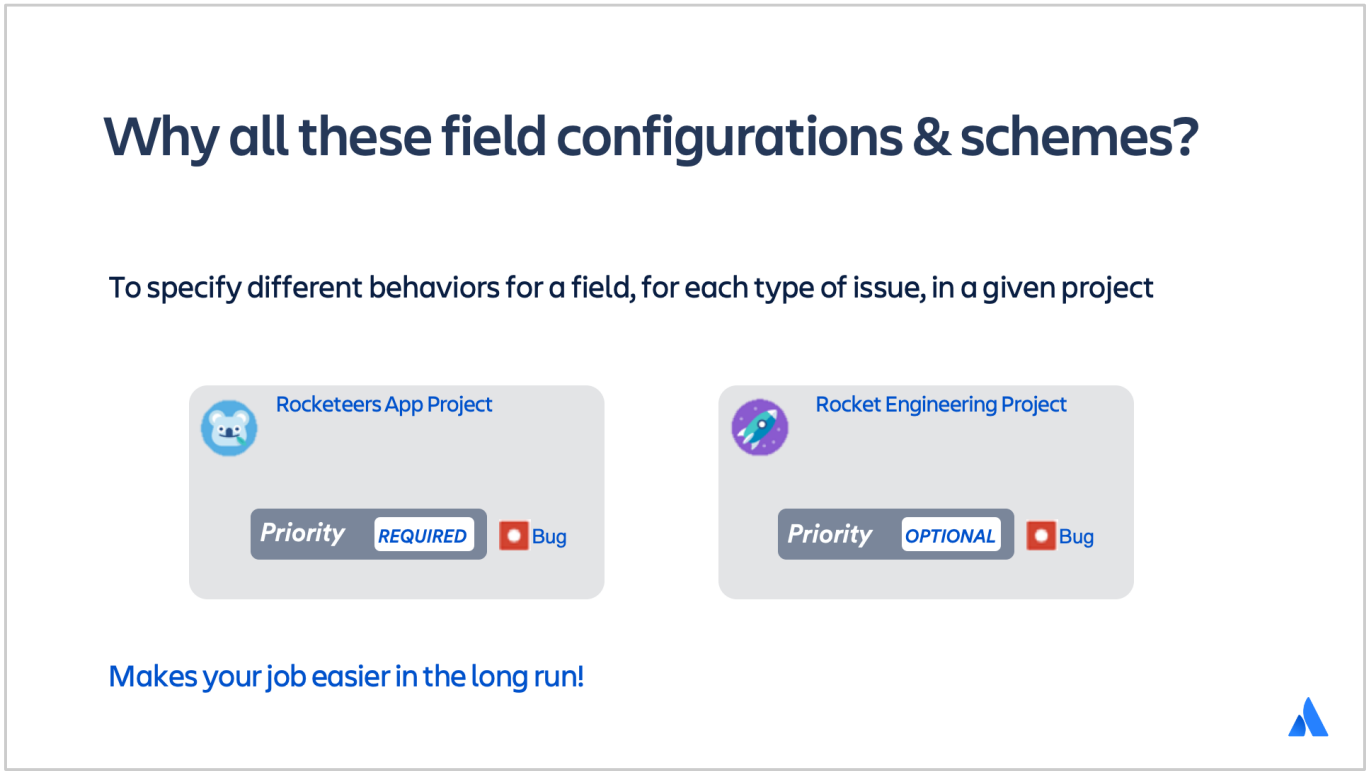
Customizing Workflows

Configuring Board & Sprint Permissions

Applying New Configurations to Projects

What will you learn?

- Create new issue types and associate them with a project
- Create a new field configuration and set field behavior
- Associate a field configuration with an issue type and a project
- Create custom fields and associate them with screens, issue types and projects
- Create a new screen and associate it with an issue operation
- Configure Priority and Resolution built-in fields
- State best practices for issue types, fields, and screens



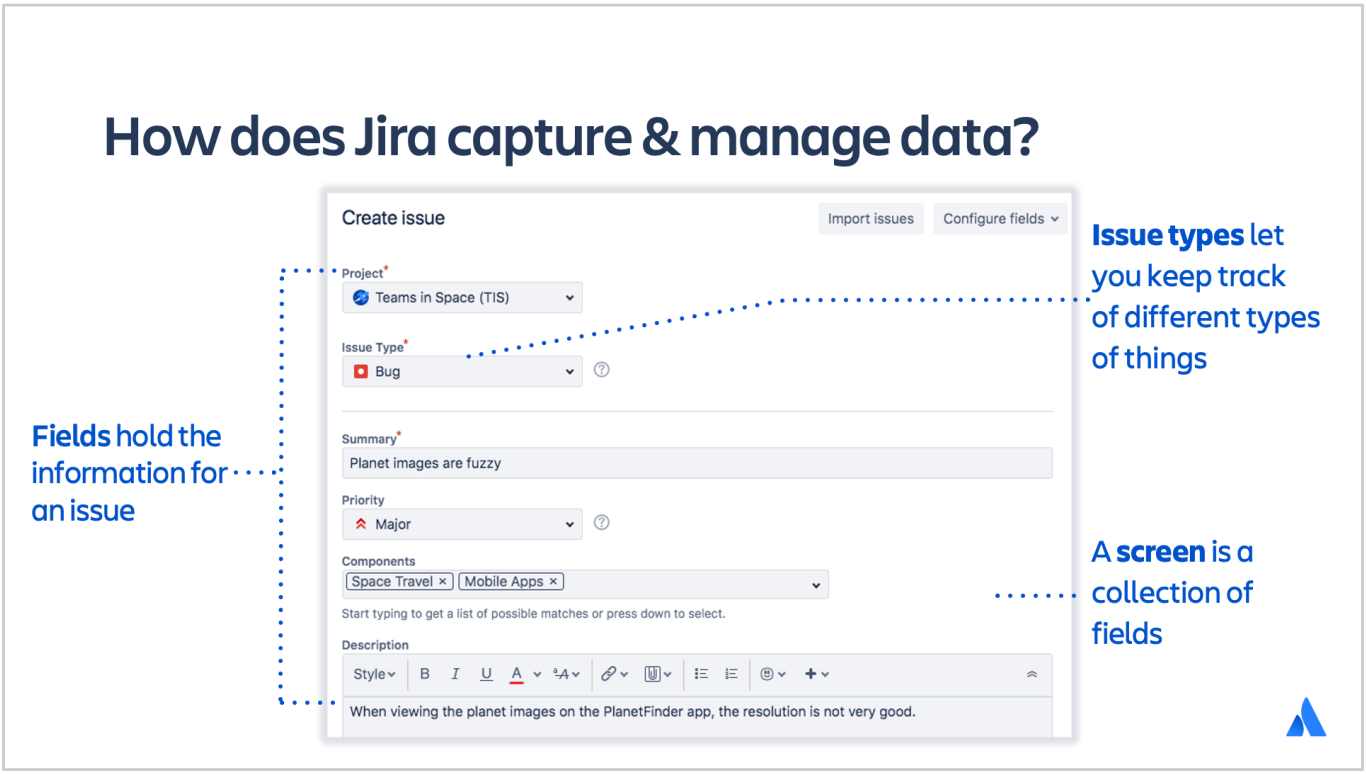
In this section we'll be covering a lot of different configurations and schemes; issue types, issue type schemes, fields, field configurations, and field configuration schemes. These are all inter-related and it can get quite complex.

Why does it need to be so complex?

The whole point of having all these different configurations and schemes is so you can have different behaviors for a field, for each type of issue in a given project or projects.

In this example, the Priority field is required for bugs for the Rocketeers App project but not required for bugs for the Rocket Engineering project

If you didn't have all these configurations and schemes, you would have to do everything manually and it would be a huge amount of administrative work. Because a field configuration scheme can be associated with more than one project (and associations between field configurations and issue types in a field configuration scheme are flexible), you can minimize your administrative workload as you can reuse the same field configuration for the same (or different) issue types across multiple projects.



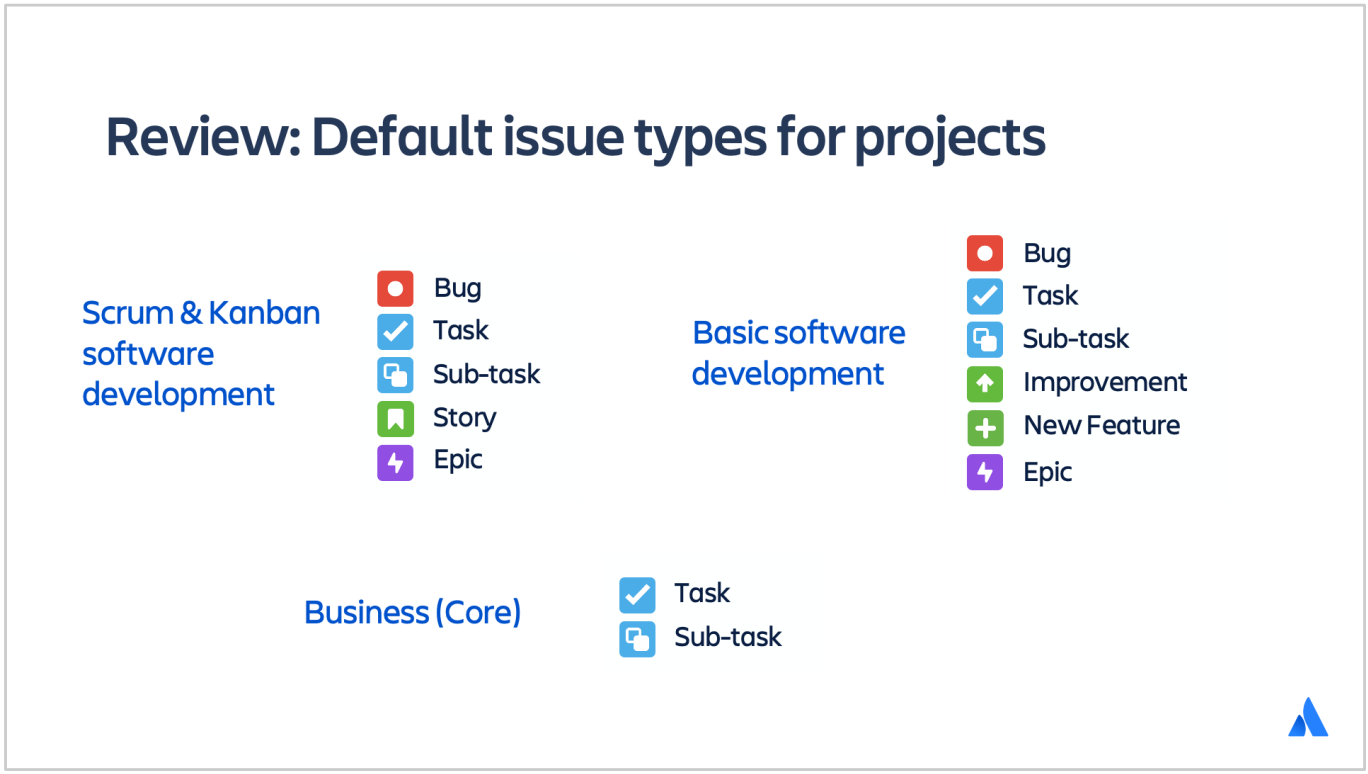
There are three elements involved in gathering and managing data; fields, screens, and issue types. Each of these three elements is created and configured separately, and then once created and configured, they need to be bound together.

Issue types are business objects that mirror real-world objects e.g., work task, bug, person, etc.

Fields hold the data associated with each issue type. Fields allow teams to track data individually so that it's easy to search and track later.

A screen is the user's view of an issue and is simply a collection of fields. Screens are the UI component that collects and displays the data fields. In Jira, a screen defines which fields are present on a screen, and their order. Fields and screens are how Jira captures the data you need to manage your projects and business processes. They are the data entry/view/edit mechanism for issues. By defining screens, you can display particular pieces of issue information at particular times. You can choose which screen to display when an issue is being created, viewed, edited, or transitioned through a particular step in a workflow.

Here we see a Create Issue screen. The fields have been tailored to capture the data required for the process of fixing bugs.



Jira enables you to keep track of different types of things – bugs, tasks, new features, etc. – by using different issue types. Issue types define the configurations that support the process of that issue (such as Bug, Task, Epic, or Story). When creating a project, project-specific issue types are established by default, and the Jira administrator can add/edit Issue types as necessary to support the business process. You can also configure each issue type to act differently (e.g., to follow a different process flow or track different pieces of information). An Issue Type is an issue with fields and workflows that are used to progress an issue to completion.

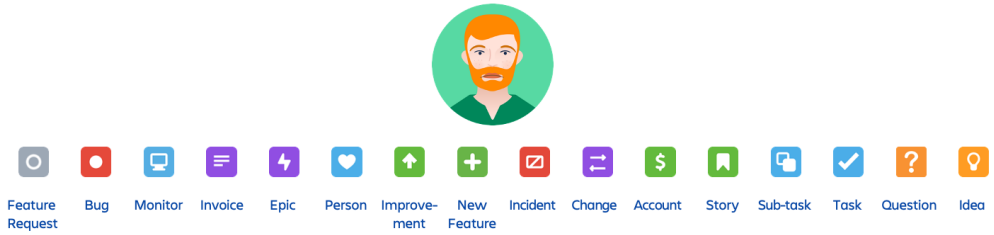
Here you see the default issue types for Software and Core projects. After installation, you only see a few issue types – more are created as you create projects of different types.

Default issue types for Software and Core (for Core it's just Task and Sub-task):

- **Task** – Work that needs to be done.
- **Sub-task** – A piece of work that is required for a task. Can be disabled on the Sub-tasks administration page.
- **Bug** – A problem that impairs or prevents the functions of a product.
- **Story** – The work to develop some deliverable feature.
- **Epic** – A big user story that needs to be broken down.
- **Improvement** – An enhancement to an existing feature.
- **New Feature** – A new feature of the product.

Creating issue types

- Create your own issue types to meet your business requirements
- But remember, too many irrelevant issue types confuse users



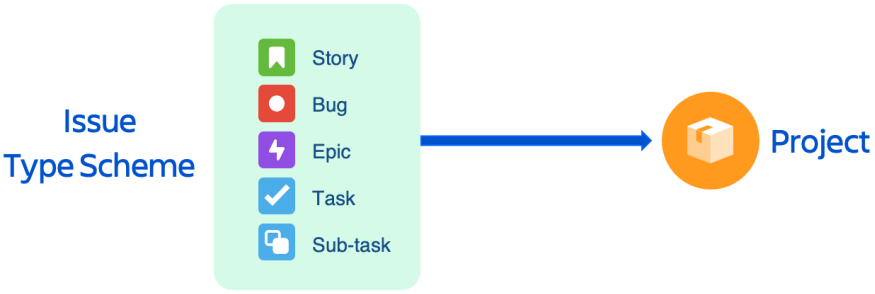
Restrict the issue types for a project to just what users need

In addition to the Jira default issue types, you can add, edit and delete your own custom issue types to suit the needs of your team. Don't create too many irrelevant issue types and ensure you give your issue types intuitive names. Otherwise you will confuse and frustrate users.

Associating issue types with projects

Issue Type Schemes

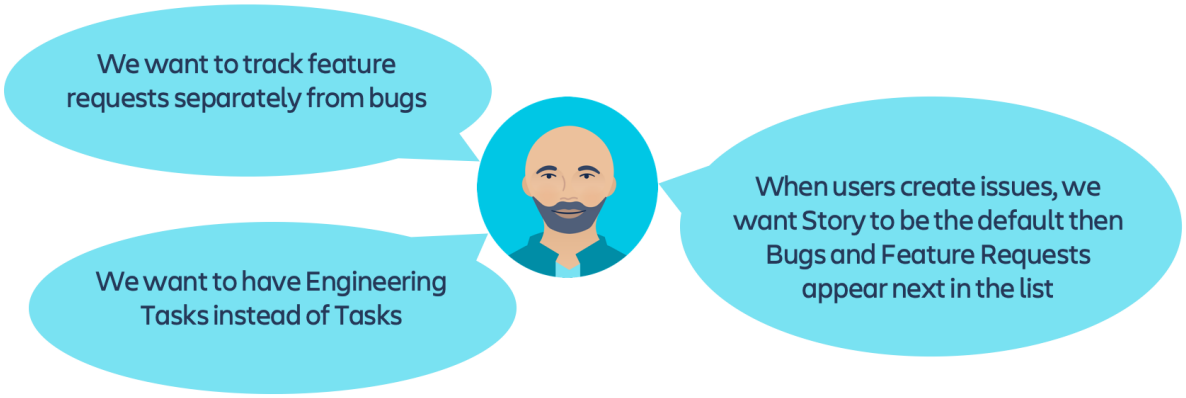
- Determine which issue types are available in which projects
- Specify the default issue type and the order in which the issue types are presented in the UI



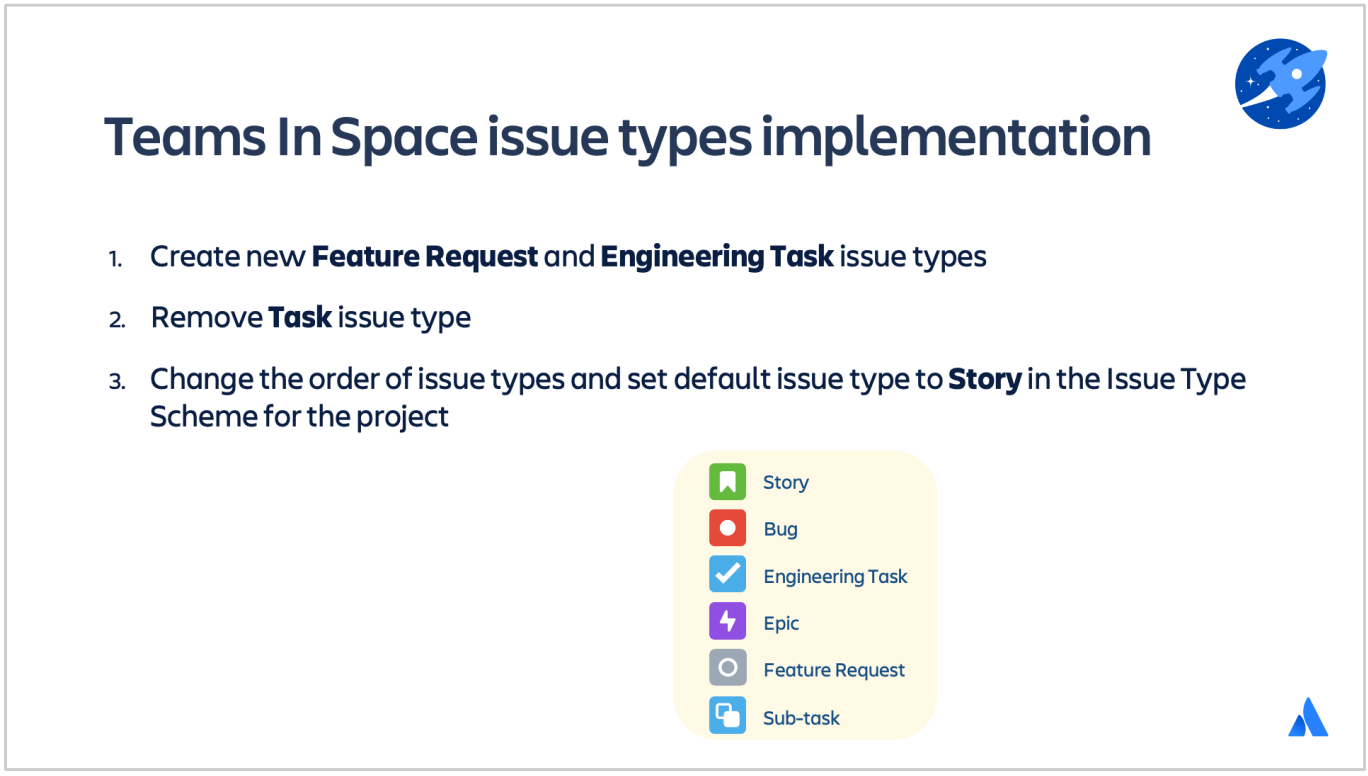
Issue type schemes are applied to projects and define (or restrict) which issue types are available to those projects. Issue type schemes also specify the order in which the issue types are presented in the UI.
By default when you create a project you get an issue type scheme for that project with the default issue types for that project template already configured. You can edit that scheme for the project and/or associate it with different projects.

Supplemental information:
For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Associating+issue+types+with+projects.

Teams In Space issue types business requirements

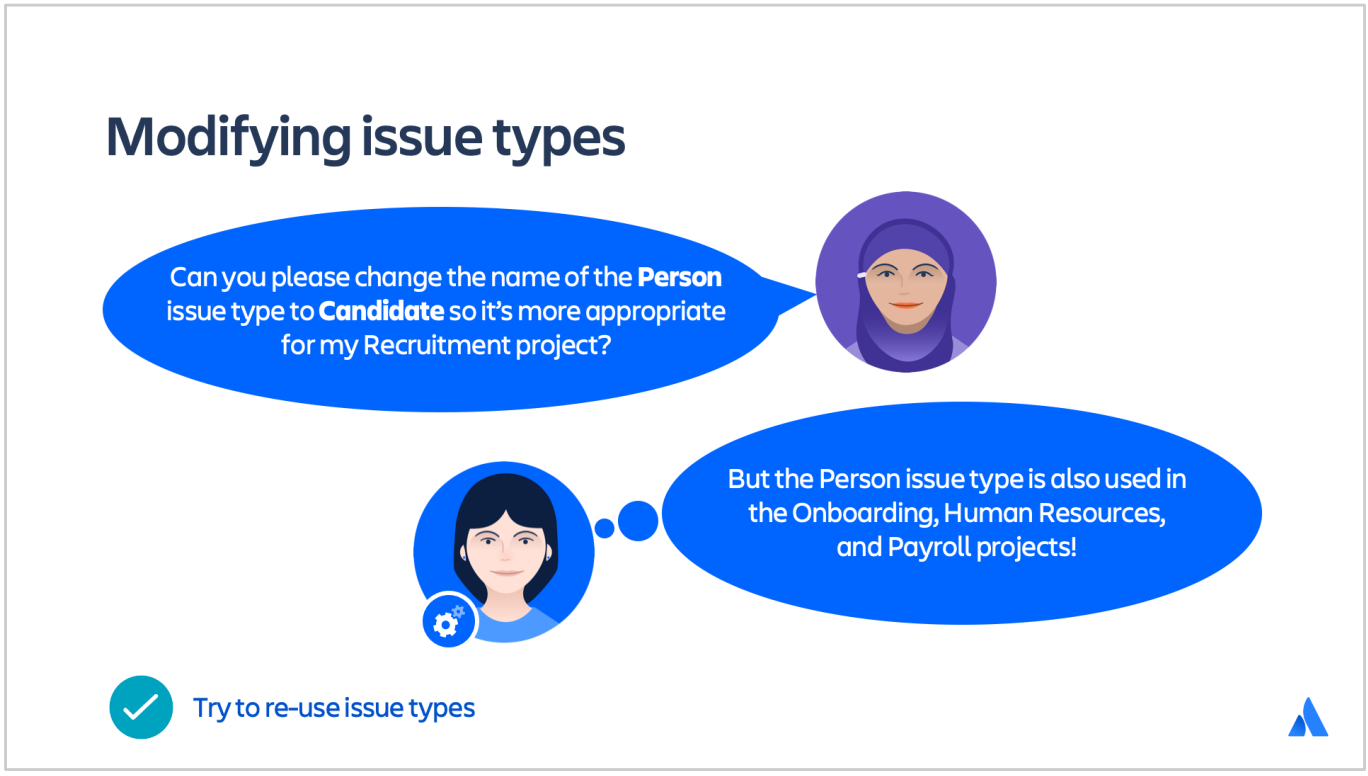


What should be done in Jira to meet these requirements?



Here's how Teams In Space will implement the issue types business requirements:

- Create a new Feature Request and Engineering Task issue types in the new standard project.
- Remove the Task issue type
- Change the order of issue types and set the default issue type to Story in the DEV: Scrum Issue Type Scheme (the issue type scheme for the new standard project).



What happens if you modify an issue type e.g. change it's name? It will change the name in any project that it's used in. Be careful of modifying issue types as those changes will affect every project that's using them. In the example, if the Jira administrator renamed the Person issue type to Candidate, this is what users in the Onboarding, Human Resources and Payroll projects would now see. That would not be appropriate.
You need to weigh the impact of creating new issue types or re-using issue types across projects. Re-using them across projects means you can share issue type schemes across projects which lessens the administrative load.

Deleting issue types

If you want to delete an issue type:

1. Search for all issues currently using that issue type
2. Bulk change issues to a new issue type
3. Delete the issue type

If you need to delete an issue type you need to deal with the issues that are currently associated with that issue type. One option is that you can simply search for all issues that currently use the issue type which you are about to delete and perform a bulk move to change those issues to a different issue type. Then you can delete the issue type itself. When you delete an issue type, Jira only prompts you to choose a new issue type if the issue type uses:

- The same workflow in all workflow schemes associated with projects
- The same field configuration in all field configuration schemes associated with projects
- The same screen schemes in all issue type screen schemes associated with projects

Setting up fields

1. What fields are needed to satisfy business requirements:
   - System fields
   - Custom fields
2. How should those fields behave:
   - Visible/hidden
   - Required/optional

Your first step in defining your data requirements is identifying the fields you need. This occurs during the business analysis phase. This should be vetted with the stakeholders and should start with the default system fields to identify which ones will support the process before reconstructing any screens. This will help procure reporting and other features that might already be leveraging these fields as well as reduce the amount of customization required.
Jira comes with a set of system fields per screen to allow quick start of your install and start collecting data to work issues and build metrics. For example, Description, Environment, Priority, Summary, and so on. You should start with these system fields to identify which ones will support the process before reconstructing the screen. This will help reduce the amount of customization required.

Questions to Ask When There's a New Field Request

What is the business value of this field?

What is required of the end user to submit this information?

How long does it take a user to reliably collect and submit the information correctly?

Is there any way to automate capturing that information?

When in the issue lifecycle do you need this information?

When someone asks you to add a field into Jira Software, always ask the following questions:

- What is the business value of this field? In other words, how does knowing this information meaningfully impact how you do your job?
- What is required of the end user to submit this information? How long does it take a user to reliably collect and submit the information correctly? Is there any way to automate capturing that information?
- When in the issue lifecycle do you need this information?

Every field has a cost. Users have to:

- Understand what the field is
- Enter data
- Update it as its value changes over time

For fields to be useful, they need to be filled out accurately and contain current information. When the system becomes cumbersome, users no longer enter quality data. Field rot then sets in, as the data is no longer dependable. Ensure each and every field is legitimately needed!

Review: Field configurations

- A field configuration:
  - Lists all the system and custom fields that are available in your Jira instance
  - Defines how fields behave – required/optional, hidden/visible

- The default field configuration applies to the whole of Jira by default

A field configuration is a set of definitions for all fields. For each field, a field configuration specifies:

- the description that appears under the field when an issue is edited
- whether the field is hidden or visible
- whether the field is required (i.e. the field will be validated to ensure it has been given a value) or optional
- (for text fields only) which renderer to use

In this example you see a portion of a field configuration. The Reporter field is a required field and has no description. It shows what screens it appears on. You can click Hide to hide the field or click Optional to make it optional.

The Default Field Configuration applies to the whole of Jira by default and shows all the fields configured in Jira, their properties and where they are used.

You can create and customize field configurations for different issue types. We'll cover that soon.

Supplemental information:

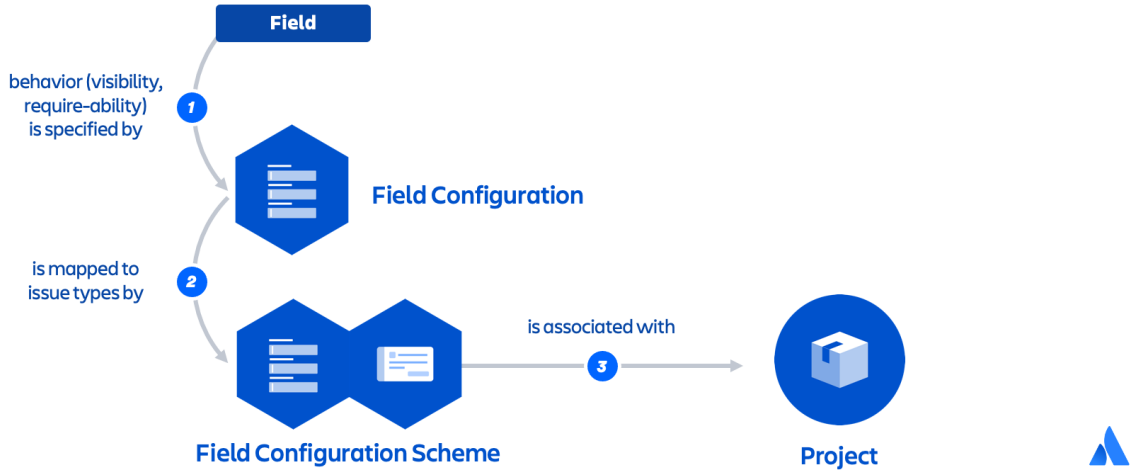
For more information, see https://confluence.atlassian.com/display/ADMINJIRASERVER085/Specifying+field+behavior.

Users are not entering data in the Environment field. But it is important for Teams In Space to gather data on what platform the issue will be fixed on, so later they will create a custom Platform field with specific options for users to choose which will make it easier for them to enter the data.

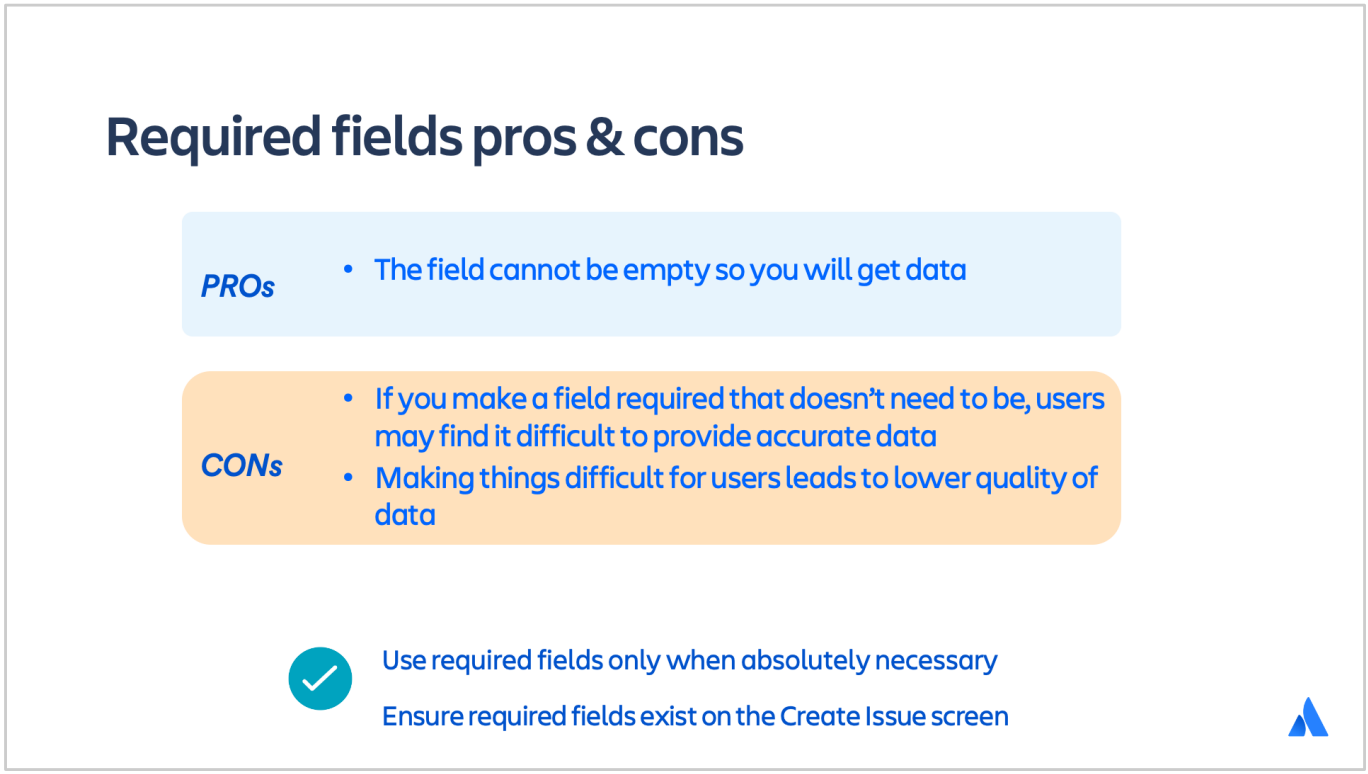
Teams In Space Fields Implementation

1. Create a new field configuration for bugs, **DEV Bugs Field Configuration**, by copying the Default Field Configuration
2. Set **Description** and **Priority** as required fields
   - Summary and Reporter are required by default
3. Hide the **Environment** field

1. Setting Field Behavior



This graphic shows all the steps that are required to set field behavior for a specific issue type or types in a specific project or projects. This graphic will be repeated throughout this module as we cover each step.
Define your field behavior in your field configuration before you add them to screens. We'll cover screens later in this module.

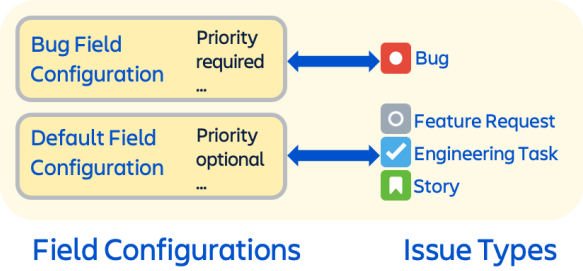


Pros of required fields: the field cannot be empty, so there will be some data entered.
Cons of required fields: If that field doesn't need to be required and you require it, then you may have made it impossible for the user to provide accurate data for that field. You are likely making it harder for the user to complete the form. A required field must have data in it or the issue cannot be worked. Make sure your users have access to the required data or else they won't be able to enter issues successfully and will become frustrated. Making things difficult for users leads to a lower quality of data, and fewer new issues. The best practice is to require fields only when absolutely necessary.
There's sometimes a strong temptation to make fields required. You need the data, so why not ensure the users enter it? But if your users don't know what to enter, they may give you bad data – and bad data is worse than no data. It's hard to get rid of bad data in searches and that makes reporting a nightmare.
If you must require a field, have a way to say "I don't know" and give a field description. Also, if you make a field required, it's mandatory from the start so ensure that the field is present on your create issue screen(s).

Mapping field behavior to issue types

Field configuration schemes

- Associate field configurations with issue types
- Allows you to specify different behaviors for fields for each type of issue



A field configuration scheme associates (or "maps") field configurations with issue types. You apply a field configuration scheme to a project so you can define different field configurations for each issue type that is available in a given project.
For example, it is possible to have separate field configurations for the Bug and the Feature Request issue types (whose associations are defined in a field configuration scheme) for a project. This allows you to specify different behaviors for a field, for each type of issue in a given project. So for bugs, the Priority field is required, whereas for Feature Requests it is optional.
It could be that there is already a field configuration scheme set up for a project. In this case you'd update the field configuration scheme and associate an issue type with it. Otherwise you can create a new field configuration scheme.
If you don't explicitly associate an issue type with a field configuration, the default field configuration will be used.
Supplemental information:
For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Associating+field+behavior+with+issue+types.

Default field configuration scheme

By default, when a project is created all the issue types for that project are mapped to the default field configuration

If you create a project the default way (not using the Create with shared configuration option), you see the default field configuration scheme in the project which maps the project's default issue types to the default field configuration.

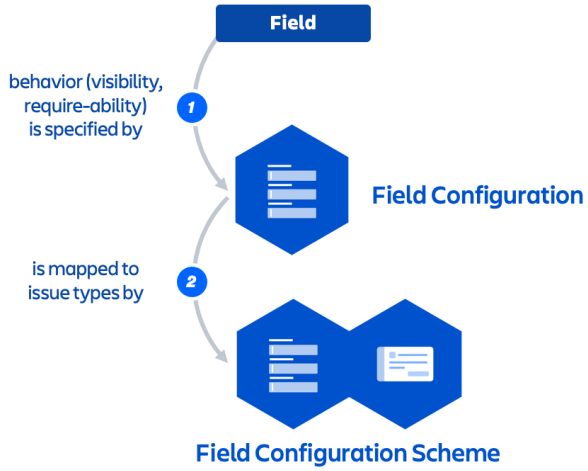
Teams In Space Fields Implementation

To have the new field changes apply to just bugs:

1. Create a new **DEV Field Configuration Scheme**
2. Associate the new **DEV Bugs Field Configuration** with the **Bug** issue type

The Teams In Space Jira administrator created a new field configuration for bugs by copying the Default Field Configuration. Then they set Description and Priority as required fields and hid the Environment field in the new bugs field configuration. Now to have these field changes apply just to bugs, they need to associate the new bugs field configuration with the Bug issue type.

2. Mapping field behavior to issue types

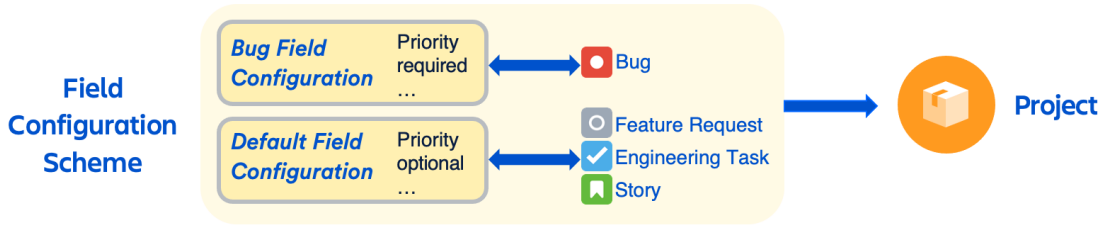


The field configuration scheme associates the field configuration(s) with issue types.

Activating new field behavior

A newly created field configuration won't take effect until you:

- Associate it with one or more issue types in a field configuration scheme
- Associate the field configuration scheme with one or more projects



A newly created field configuration will not take effect until you associate it to one or more issue types and associate it with one or more projects. This means that you can define different field configurations for each issue type that is available in a given project. Because a field configuration scheme can be associated with more than one project (and associations between field configurations and issue types in a field configuration scheme are flexible), you can minimize your administrative workload as you can reuse the same field configuration for the same (or different) issue types across multiple projects.

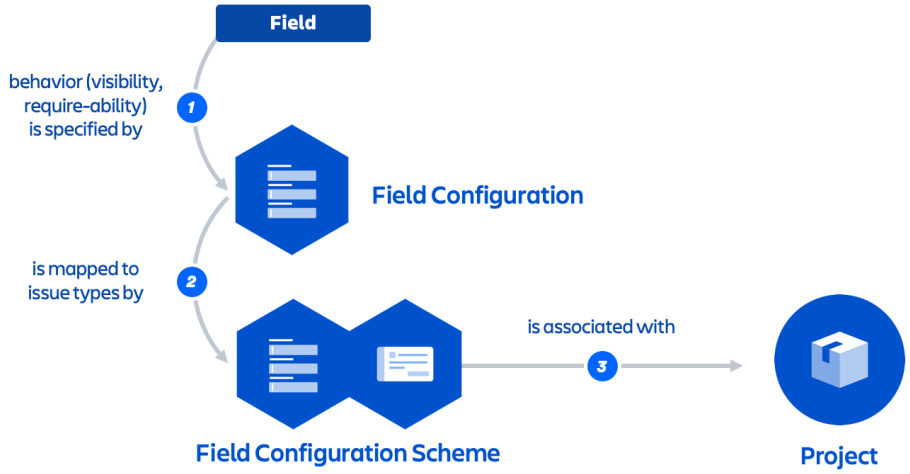
Teams In Space fields implementation

To have the bug field changes apply to the standard **DEV** project:

- Associate the new DEV Field Configuration Scheme with the project

The Teams In Space Jira administrator created a new field configuration for bugs by copying the Default Field Configuration. Then they set Description and Priority as required fields and hid the Environment field in the new bugs field configuration. Then they associated the new bugs field configuration with the Bug issue type in the bugs field configuration scheme. Now they need to associate the new bugs field configuration scheme with the project.
Later they will stamp out new development projects using these schemes.

3. Associating issue type field behavior to a project



By applying the field configuration scheme to a project or projects, you can specify different behaviors for a field, for each type of issue in a given project or projects.

Are you getting it?

A field configuration scheme associates...

a. Field behavior with fields
b. Field configurations with issue types
c. Issue types with projects

The answer is on the next slide.

Did you get it?

A field configuration scheme associates...

a. Field behavior with fields

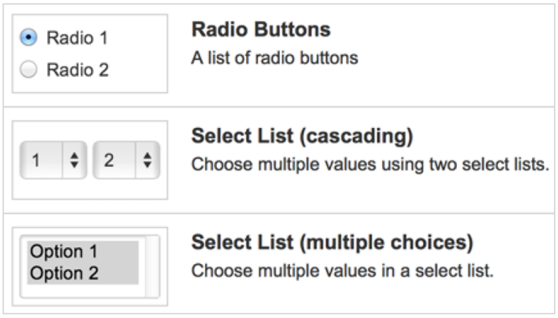
✓ b. Field configurations with issue types

c. Issue types with projects

Answer:
b – A field configuration scheme associate field configurations with issue types.
A field configuration associates field behavior with fields.
Issue type schemes are associated with projects.

Creating custom fields

- When you create a custom field, you specify:
  - Field type, for example
  - A default value, if any
  - The field's context
- Custom fields are added to the default field configuration



You can add custom fields in addition to Jira's built-in system fields. The standard field types include checkboxes, date picker, number field, radio buttons, etc. The advanced field types include single group picker, multiple group picker, read only text field, etc. Custom fields are always created as optional fields so you can create a new custom field without requiring existing issues to be changed. The existing issues will contain no value for the new custom field, even if a default value is defined.

The field's context is the issue types and projects to which they apply. This can be global and apply to all issue types and/or all projects or just to a specific issue type(s) and/or project(s). By default, new fields are globally available to all issue types and projects. Custom fields are added to the default field configuration so you can set behavior.

Supplemental information:

You may want to limit a field to particular project(s) and issue type(s), for example, when:

- A field is only needed for one project or issue type.
- The select list options can be modified according to project requirements (e.g. different payment terms for different projects).
- To limit the number of fields returned via simple search.

If you limit a field to an issue type and project, be careful when moving issues to a different project. Make sure you update the field context for the new project!

For more information on custom fields, https://confluence.atlassian.com/display/AdminJIRAServer085/Configuring+a+custom+field.

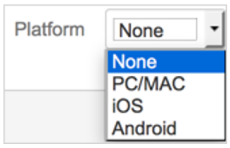
Teams In Space Custom Fields Business Requirements

What should be done in Jira to meet these requirements?

Teams In Space Custom Fields Implementation

Create a **Platform** custom field:

- Select List (single choice) type, with PC/MAC, iOS, or Android options
- For Bug issue type



Create a **Developer** custom field:

- User Picker (single user) type
- For Feature Request, Story, and Bug issue types

The new Platform field replaces the hidden Environment field.

Custom fields best practices

- Limit the number of custom fields
- Choose field types that aid reporting
- Don’t duplicate names
- Make names as generic as possible to aid re-use
- Scope each field
- Fill out the field description

- Limit the number of custom fields – Be careful how many custom fields you define in Jira. More than a thousand is a large number and may affect Jira's performance.
- Use fields types that aid reporting – Consider what reports are needed from Jira and only create fields that support those fields. Custom field types, such as select and multi-select are good for reporting. On the other hand, text fields are not as useful, since people don't always enter data as expected by the report's query.
- Don't duplicate names – Don't create new custom fields with the same name as existing custom or system fields. If you do, then choosing the correct field in JQL searches can become confusing for users.
- Give custom fields non-specific names that can be reused in other places. For example, instead of naming a field "Marketing Objective", use "Objective", and give a description in the field configuration that states the Jira projects where that field is used.
- Scope each field - When administrators create custom fields, they have the option to limit them to certain projects or make them available to all projects. Using project contexts to limit where custom fields show up makes it easier for everyone. Forms that only have what’s necessary, makes everyone more efficient.
- When creating a custom field, enter a description of the new field. Make use of that description so users can learn what information is needed in the field. I’d keep the description short so that the overall screen remains compact. You can use HTML in field descriptions, too, so that it’s easy to link off to more extensive documentation.

General fields best practices

- Capture just the data that is needed to facilitate data driven decision making
- Don't create unnecessary fields
  - Don't create **End Date** if you can use system **Due Date**

Don't over-engineer your Jira fields. Your goal is to facilitate data driven decision making. Capture the data that is needed, and don't capture data that is not needed. Capturing unneeded data results in two problems:

- Wasted time gathering/entering the unneeded data
- Wasted time sorting through or looking at the unneeded data

Don't create unnecessary fields and try to not copy fields. Sometimes people will ask for fields with the same name but different configurations, and they want these on different screens within the same project. In this case, custom field context configuration isn't an option and you just have to do it, but try to avoid this whenever possible

Don't create a custom field when a system field will do. Don't create "End Date" if you can use the system field "Due Date."

See how it's done

- View issue type schemes
- View a project's Summary page
- View a project's field configuration and the Default Field Configuration Scheme

Instructor demo

Are you getting it?

Your Sales team is requesting four new fields to run reports against:

- Q1 Sales Goal
- Q2 Sales Goal
- Q3 Sales Goal
- Q4 Sales Goal

Is it a good idea to use these names?

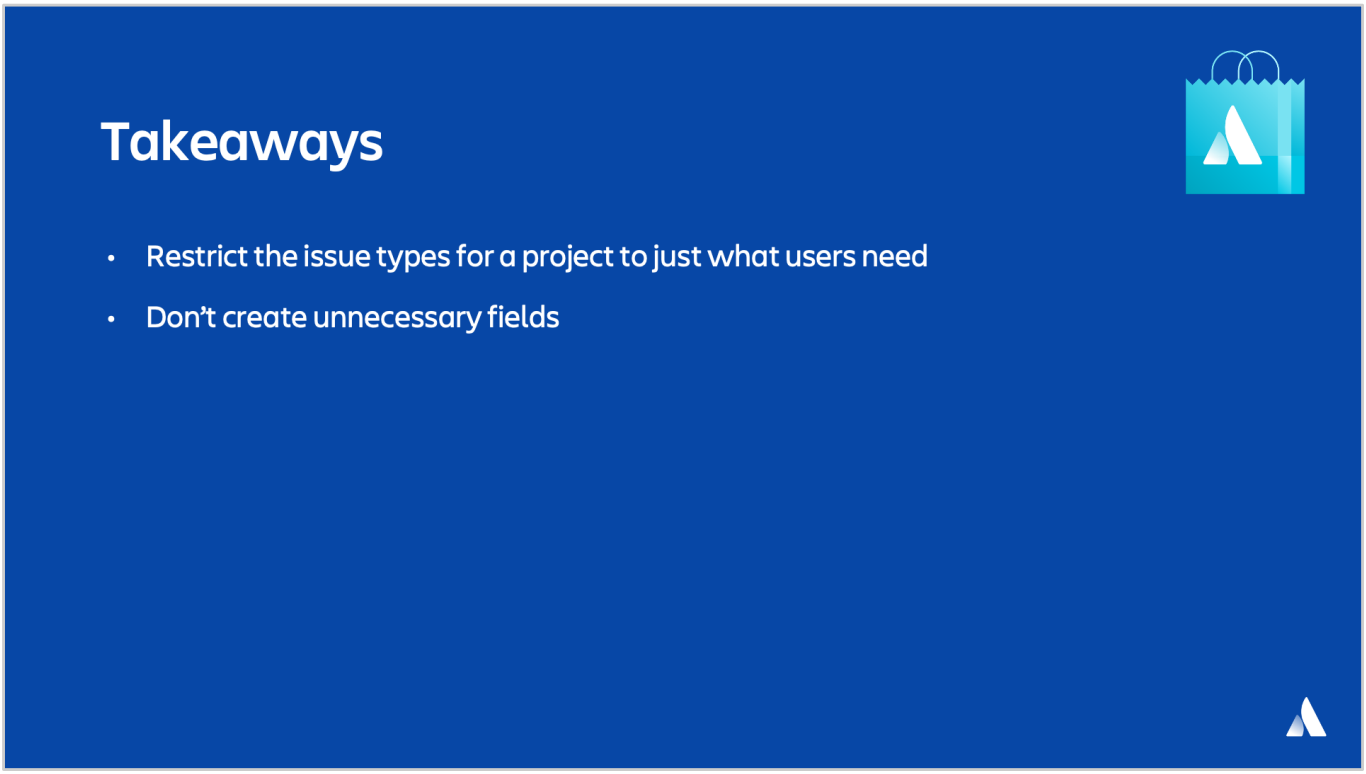
The answer is on the next slide.

Did you get it?

Using very specific field names like Q1 Sales Goal, Q2 Sales Goal, Q3 Sales Goal, and Q4 Sales Goal is not a good idea.

First you want to avoid creating a lot of custom fields (less than a thousand) so you don't impact performance. The Marketing team as well as others might want to use the same types of fields for their quarterly goals.

So a better solution would be to create generic fields – Q1 Goal, Q2 Goal, Q3 Goal, and Q4 Goal. Then you can re-use these fields for any team projects that have quarterly goals.



Too many irrelevant issue types confuse users so restrict the issue types for a project to just what users need.
Don’t create unnecessary fields, and try to not copy fields.

- Sometimes people will ask for fields with the same name but different configurations, and they want these on different screens within the same project. In this case, custom field context configuration isn’t an option and you just have to do it, but try to avoid this whenever possible.
- Don’t create a custom field when a system field will do. Don’t create “End Date” if you can use “Due Date.”

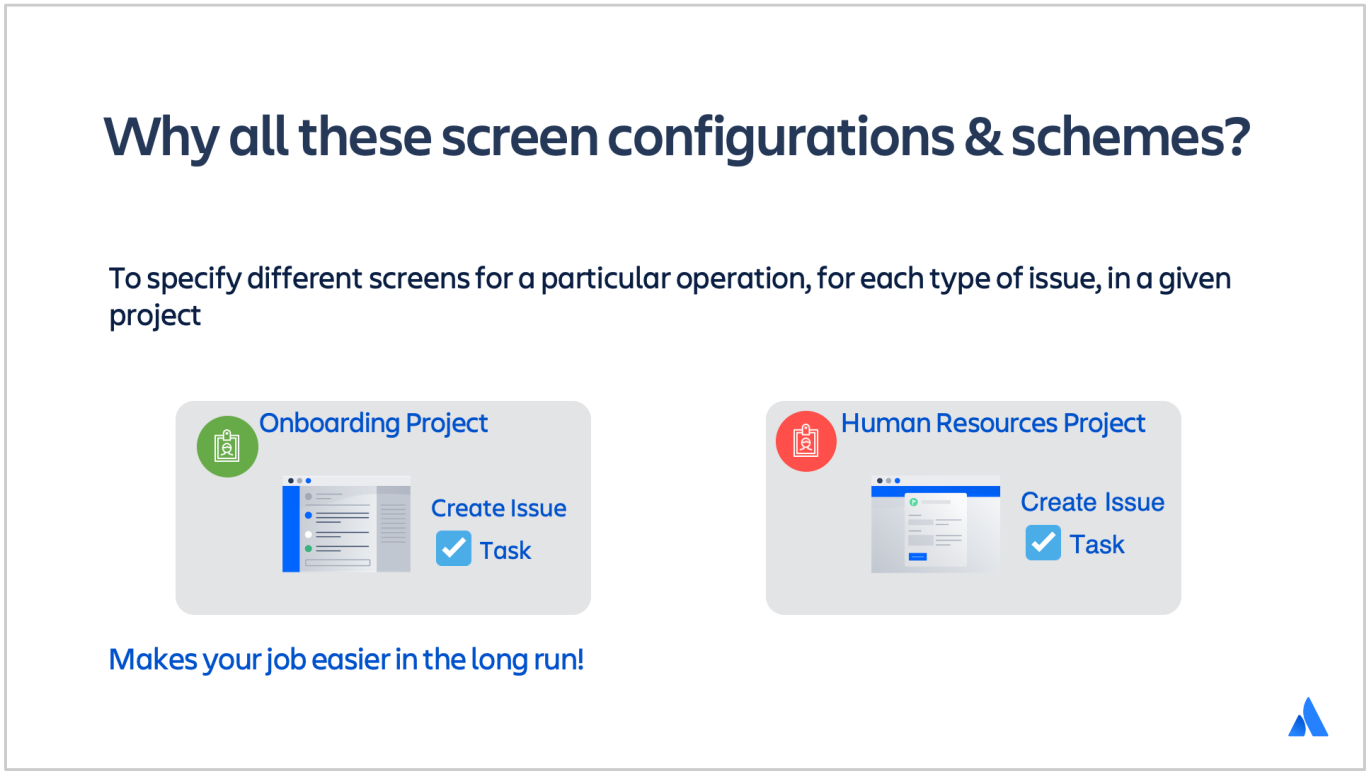
Limit the number of custom fields – Be careful how many custom fields you define in Jira. More than a thousand is a large number and may affect Jira's performance.

Try it

Lab 3 – Configuring Issue Types, Fields, Screens, & Schemes

- Exercise 1 – Configuring Issue Types
- Exercise 2 – Configuring Fields

Perform exercise 1 and exercise 2 in Lab 3. You will perform the remaining exercises at the end of this module.



We've covered issue types and fields. Now we'll cover screens and how they all tie together.

In this section we'll be covering a lot of different configurations and schemes –screens, screen schemes, and issue type screen schemes. These are all inter-related and it can get quite complex. Why does it need to be so complex?

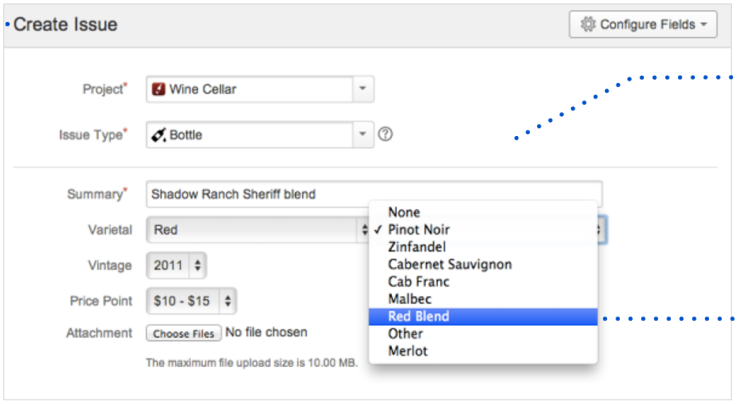
The whole point of having all these different configurations and schemes is so you can specify different screens for a particular issue operation, for each type of issue in a given project or projects. For example users see the Candidate Creation screen when creating a task for the Onboarding project but see the Task Creation screen when creating a task for the Human Resources project.

Another example, creating a task in the Onboarding project could display a different screen with different fields, and some of the same fields but with different behavior, than the screen you see when creating a task in all the other five Human Resources projects.

If you didn't have all these configurations and schemes, you would have to do everything manually and it would be a huge amount of administrative work. Because an issue type screen scheme can be associated with more than one project (and associations between screens and issue types in a screen scheme are flexible), you can minimize your administrative workload as you can reuse the same screen schemes for the same (or different) issue types across multiple projects.

Review – screens

A screen is mapped to a specific issue operation



A screen is a collection of fields

You can add and position fields on screens

Users can view, enter and edit issues on screens

A screen is the user's view of an issue and is simply a collection of fields. In Jira, a screen defines which fields are present on a screen, and their order. Note that a hidden field can be present on a screen but will still be invisible. Fields and screens are how Jira captures the data you need to manage your projects and business processes. They are the data entry/view/edit mechanism for issues.

Jira allows you to display particular pieces of issue information at particular times, by defining screens. You can choose which screen to display when an issue is being created, viewed, edited, or transitioned through a particular step in a workflow.

Information for each issue is held in the fields that are associated with that issue and you can tailor these fields to suit your organization's needs.

Here we see a Create Issue screen. This is and example of using Jira to manage a wine cellar. The fields have been tailored to capture the data required for the process of collecting wine.

Supplemental information:

See http://blogs.atlassian.com/2014/05/life-runs-jira-sommelier-edition/.

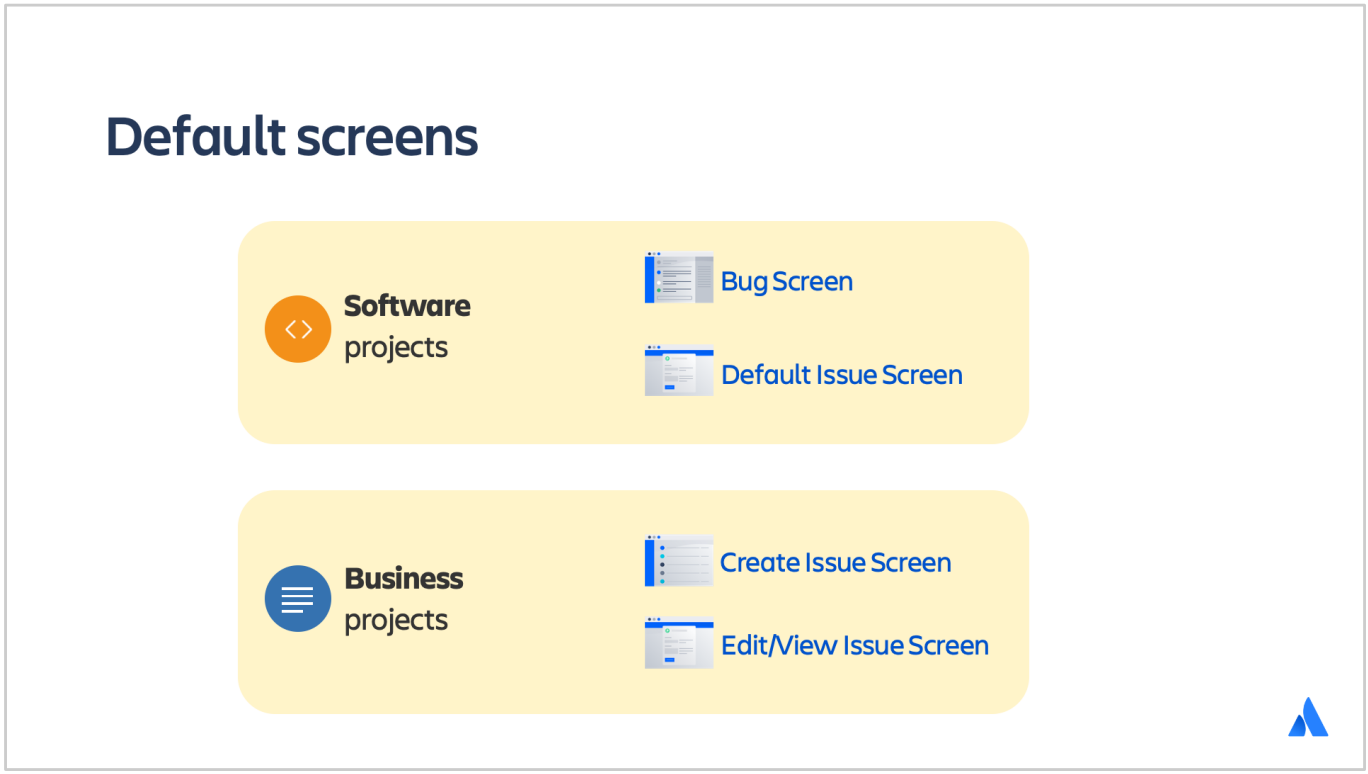
Screens where users interact with issues

- Users interact with issues, when they:
  - **Create** an issue
  - **Edit** an issue
  - **View** an issue
  - **Transition** an issue through workflow and information is required

You can choose a screen for each of these issue operations!

There are 4 places where the user interacts with Issues to input data: Create, Edit, View and Transition. These are known as “Issue Operations”:

- The create screen appears when a user creates an issue
- The edit screen appears when a user edits an issue
- The view screen appears when a user views an issue
- The transition screen is used when information is required as an issue moves through a transition in workflow. To make a transition screen available to users you associate it with a workflow transition. For example, when an issue is resolved, we can indicate how the issue was resolved via the Resolution Field. We’ll discuss workflow transitions in the next module.



There are different default screens for Software and Business type projects (all project templates).
When you create a Software project, you get two default screens for the project – a Bug Screen and a Default Issue Screen.
When you create a Business project, you get two default screens for the project – a Create Issue screen and an Edit/View Issue Screen.
We'll cover what issue operations they are used for next.

Creating & editing screens best practices

- Copy and customize an existing screen rather than creating from scratch to save time
- Order fields on screens from general to specific and most important first
- If you need a lot of fields, group them in tabs
- Keep your screens simple

Too many fields causes
user frustration and bad data

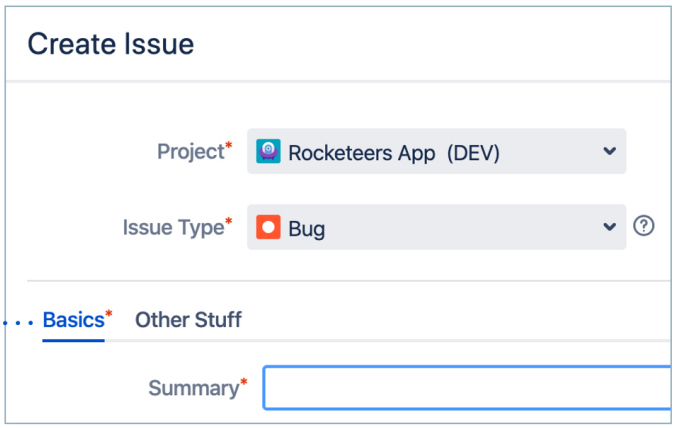
You can create a screen from scratch but copying an already existing screen and customizing it saves you a lot of time, especially if the changes are not great.
Order fields on screens from general to specific and most important first. By default, Jira Software adds new fields to the bottom of the screen. When you initially build a screen, all the fields should be in a logical order for that application. As the team needs extra fields, take the time to add them to the right place on the right screen. Doing so will optimize the data entry experience for your end users. Each new field should bring a user closer to completion. After you've created the screen, ask a few people who use each screen on a regular basis if the flow is right. Fine tuning the flow will help users enter better data, making for a more optimized workflow for the entire team.
If you have a lot of fields that you need, group related fields into tabs (more next slide.)
Simplify screens for users to make them easier to use. Remove unused fields as too many fields means frustrated users, inefficient use of time, difficulty managing a lot of custom fields, and bad data.
You know you have too many fields when:

- Users are not filling out all the required information when creating issues (field-itis).
- People don't submit issues because it's too much work.
- Certain fields have bogus information because end users don't understand value.

Less is more and will ensure the integrity of the data and your reporting.
Supplemental information: See http://go.atlassian.com/jirafieldgreatness/.

Creating tabs on screens

Group related fields in tabs so they're easier to read



Simplify your screens

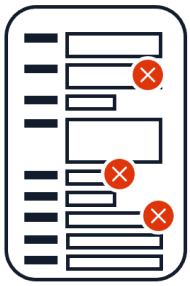
Tabs can be used to help group related fields. Tabs are useful for organizing complex screens, as you can place less used fields onto separate tabs. You want to avoid overly complex screens. You can also add, remove and reorder tabs, as well as rename them.
In this example you see a Create Issue screen with two tabs – Basics and Other stuff. The Basics tab contains fields such as Summary, Description, Assignee, etc. The Other stuff tab contains fields such as Reporter, Publish Date/Time, Original Estimate, Remaining Estimate, etc.
Supplemental information:
For more information see https://confluence.atlassian.com/display/AdminJiraServer085/Defining+a+screen.

Hiding, removing & deleting fields

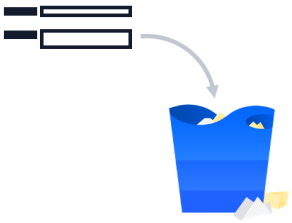
There are times when you may need to:

Hide fields in a field configuration

Remove fields from a screen



Delete custom fields



What fields do you want on your screens to meet your business requirements?
You can hide fields in field configurations, remove fields from a screen and delete custom fields. Note, you cannot delete system fields. Each of these are explained in the next two slides.

Removing vs. Hiding fields

Removing a field from a screen	Hiding a field in a Field Configuration
• Won't delete existing data • Won't affect search or reporting	• Won't delete existing data • Field will not be viewable, searchable, or usable in any way

Hidden fields are no longer available for use in the context they're configured, they are not removed, but hidden (archived away). The context is set by what issue type or types the field configuration is associated with and what projects it's used in. You can hide a field in a field configuration without impacting other field configurations. So, for example, you could hide a field just for the bug issue type, just for one project. A hidden field will not be available to interact with on any screen, it will not be viewable, searchable, or usable in any way. Issues that had field data will no longer display it or return search results for it. That data however does not go away. If you show the field in the field configuration in the future, any pre-existing data associated with that field will become available at that time.

Hiding a field in the field configuration is distinct from not adding a field to a screen. Fields hidden through the field configuration will be hidden in all applicable screens, regardless of whether or not they have been added to the screen.

If a hidden field has a default value, then it will not receive a value when an issue is created, regardless of whether the field is present on the create issue screen(s).

The fields Summary and Issue Type cannot be hidden and as such there is no Hide option available for these fields.

If you hide the Fix Version/s field, the Change Log and Road Map reports won't work.

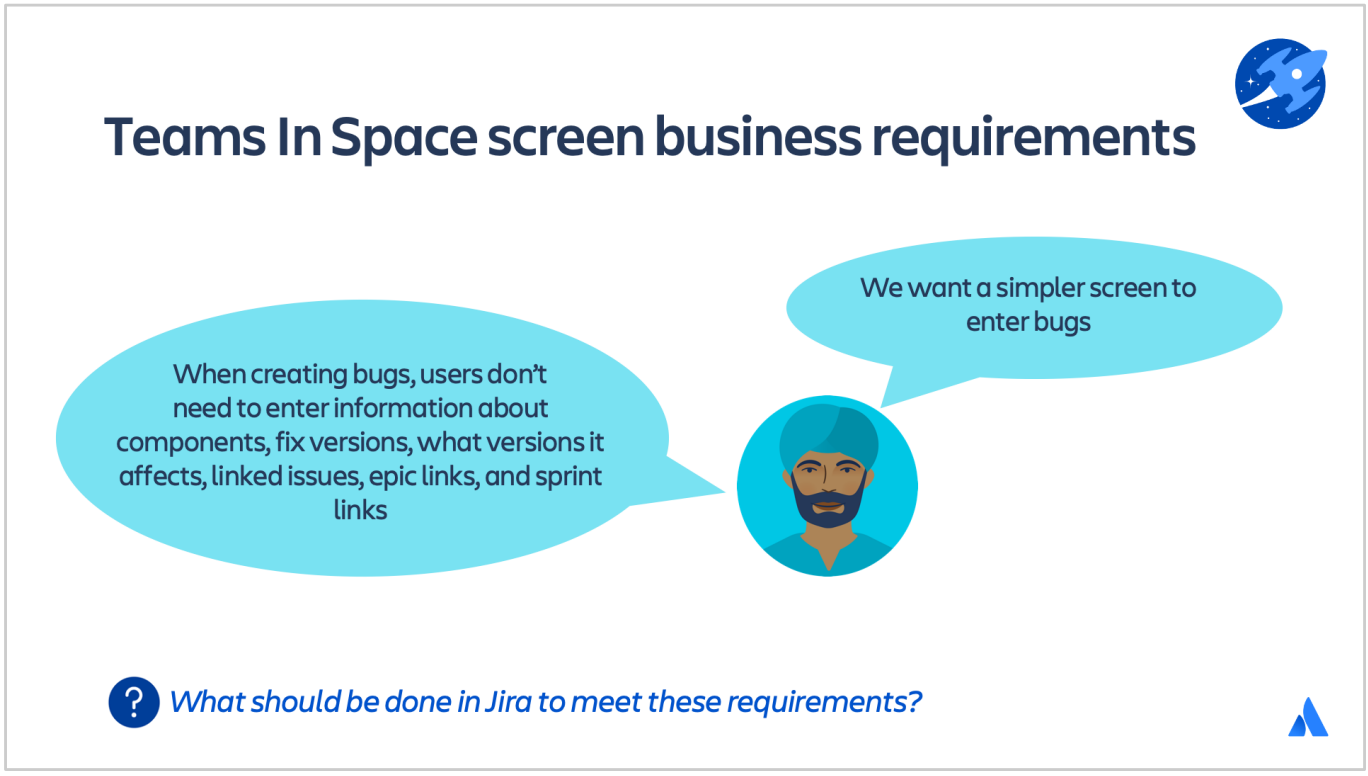
Removing a field from a screen will not delete the existing data, nor will it affect search or reporting.

Removing vs. Deleting custom fields

Removing a custom field from a screen	Deleting a custom field
• Won't delete existing data • Won't affect search or reporting	Will destroy the data in the field

⚠ Think carefully before deleting custom fields!

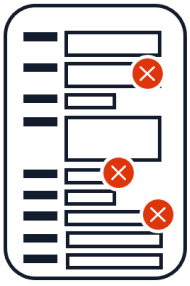
Removing a field from a screen will not delete the existing data, nor will it affect search or reporting. However, if you delete a field, it will destroy the data in the field.
You cannot delete system fields.



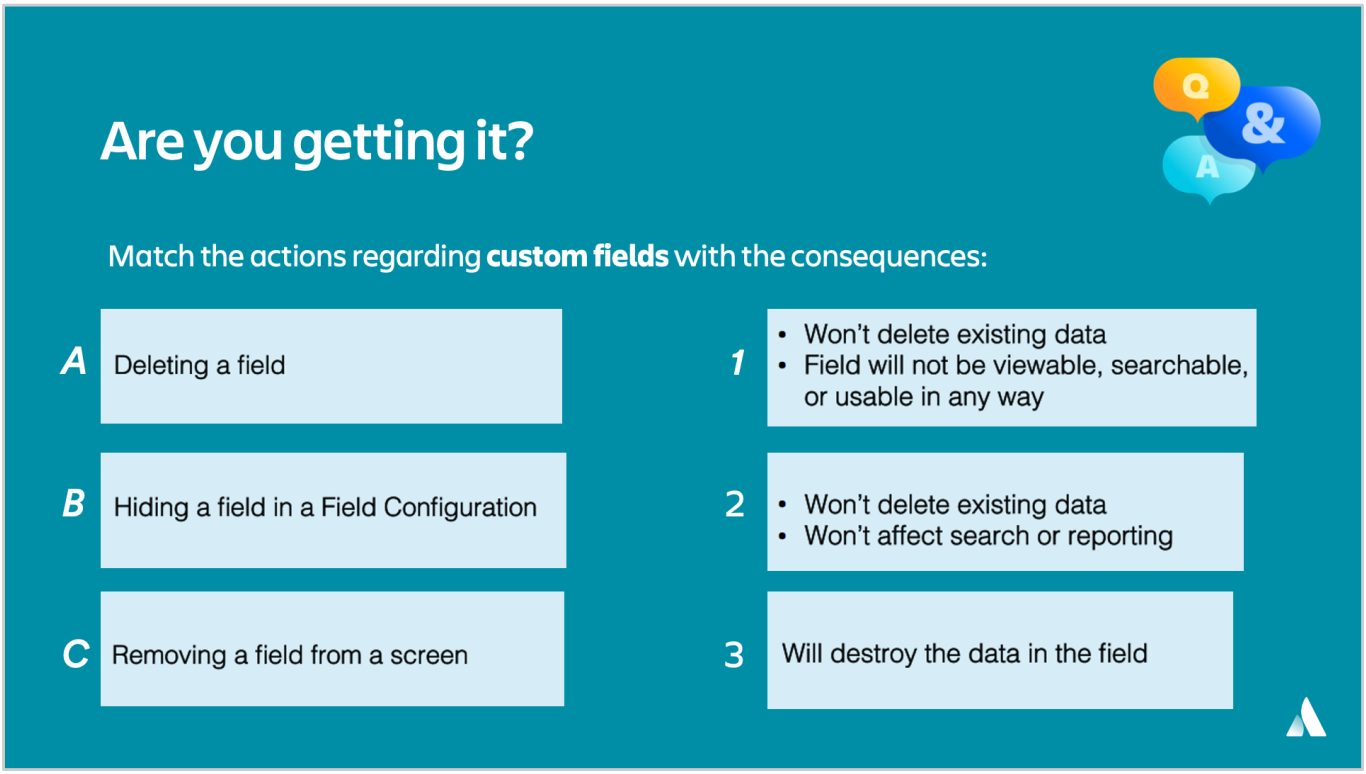
Here are the screen business requirements for Teams in Space. They want a simpler screen to enter bugs. When creating bugs, users shouldn't need to enter information about components, fix versions, what versions it affects, linked issues, epic links, and sprint links.

Teams In Space screen implementation

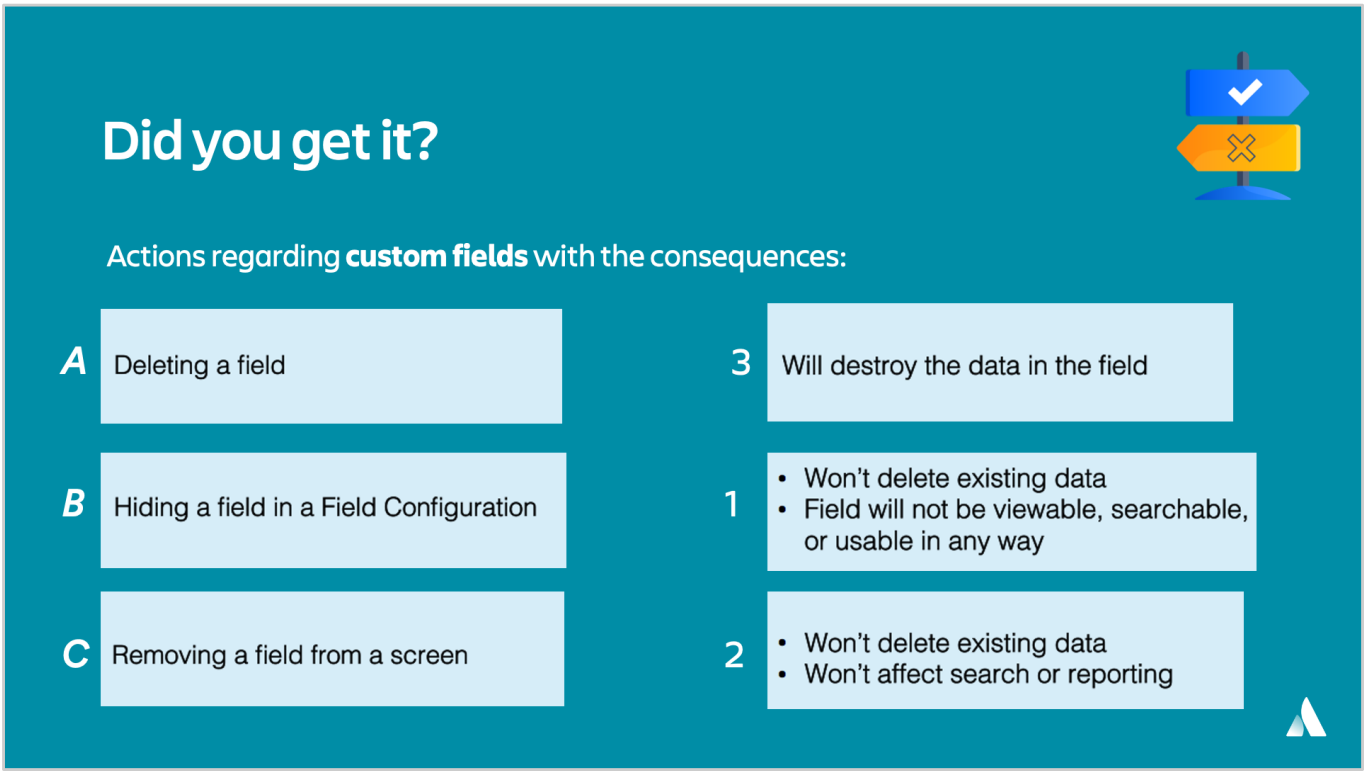
1. Create a new **DEV: Scrum Create Bug Screen** by copying the project's default DEV: Scrum Bug Screen
2. Remove Components, Fix versions, Affects versions, Linked Issues, Epic Link, and Sprint fields from the new bug creation screen



To implement their business requirements, Teams in Space will create a new Create bug screen by copying the projects' default bug screen. Then they'll edit that screen and remove the unwanted fields.



The answer is on the next slide.

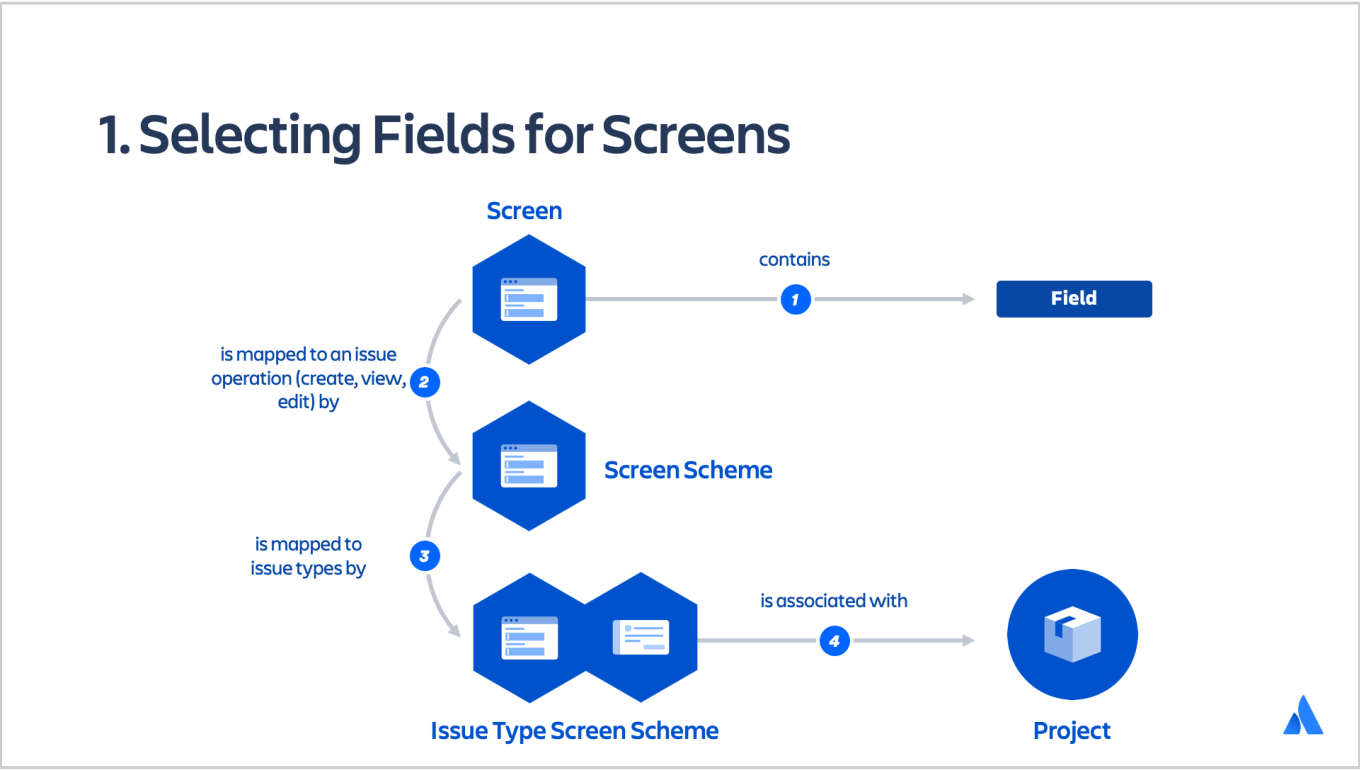


Answers: A 3, B 1, C 2

Deleting a custom field will destroy data in the field.

Hiding a custom field in a field configuration won't delete existing data and the field will not be viewable, searchable, or usable in any way.

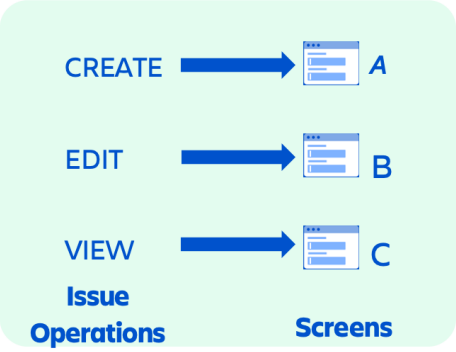
Removing a custom field from a screen won't delete existing data and won't affect search or reporting.



This graphic shows all the steps that are required to set the screens that are available for specific issue type or types in a specific project or projects. This will be repeated throughout the rest of this module as we cover each step.

Associating screens with issue operations

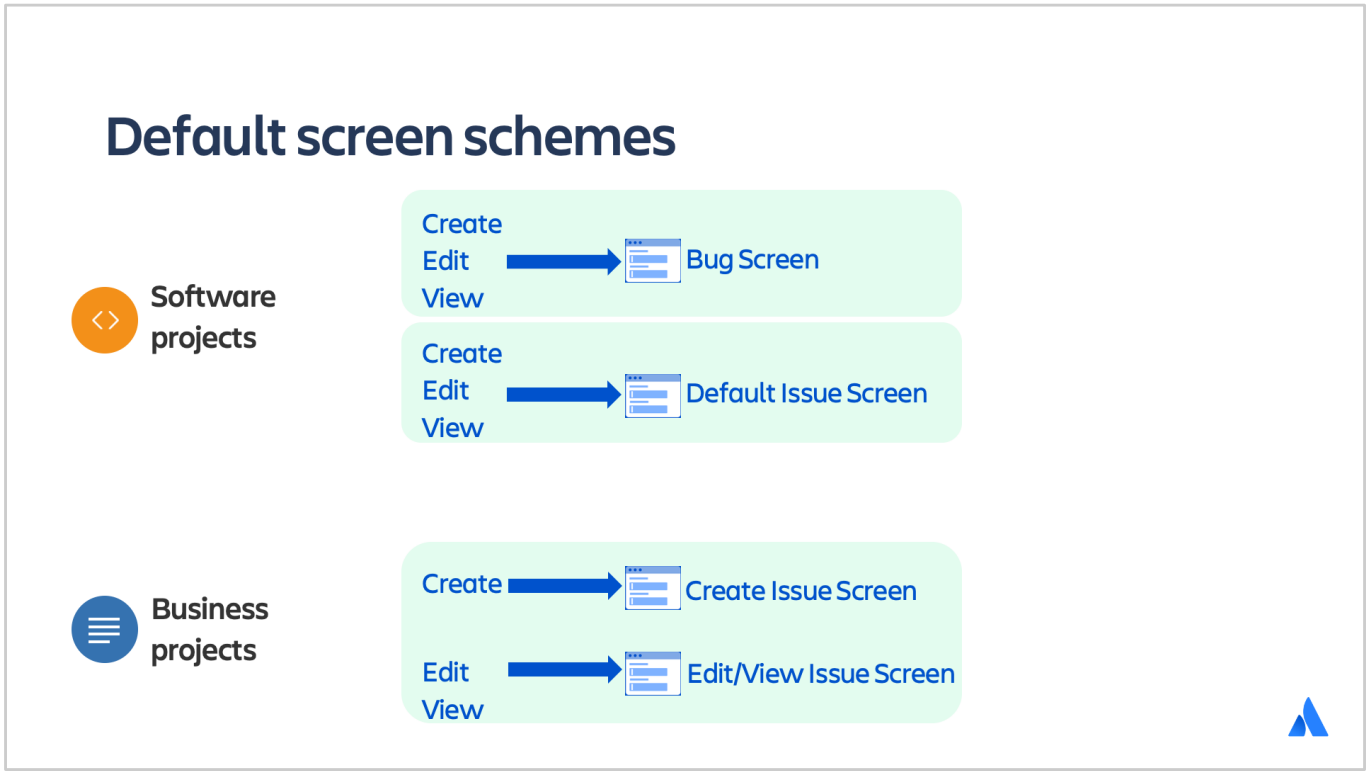
Screen Schemes determine which screens are shown for each issue operation



To choose screens that are displayed when issues are created or edited, map the screens to issue operations using screen schemes.
A screen scheme allows you to choose which screen will be shown to a user when they perform a particular issue operation. There are three issue operations for which you can choose a screen:

- Create Issue – the screen that is shown when an issue is being created.
- Edit Issue – the screen that is shown when an issue is edited.
- View Issue – the screen that is shown when a user views an issue.

In a screen scheme, you can specify the same screen (or choose different screens) for these issue operations. If there are no mapped operations for a screen it will show as default.
In this example, if a user creates an issue, they will see screen A. If they edit an issue, they will see screen B. And if they view an issue, they will see screen C.
You can only delete a screen if it's not part of a screen scheme and is not used in any workflows. And you can only delete a screen scheme if it is not used in an issue type screen scheme. Note: Although it is possible to associate any screen defined in your Jira installation with either a screen scheme or a workflow transition view, screen schemes and workflow transition views are distinct and unrelated.
For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Associating+a+screen+with+an+issue+operation.



When you create a project the default way (not by selecting Create with shared configuration), new screen schemes are created just for that project.
For Software projects, by default, you get two new screen schemes. When you create, edit or view a bug you see the default bug screen. Whereas, by default, when you create, edit or view all other issue types you see the default screen. This applies to all the Software project templates – Scrum, Kanban and Basic software development.
For Business projects, by default, you get one new screen scheme. When you create an issue of any type, you see the default create issue screen. Whereas, by default, when you edit or view an issue of any type you see the default edit/view screen. This applies to all the Business project templates – Project Management, Process Management, and Task.

Deleting screens & screen schemes

You can only delete a...	if...
Screen	• It's not part of a screen scheme *and* • It's not used in any workflows
Screen scheme	• It's not used in an issue type screen scheme

You can only delete a screen if it’s not part of a screen scheme and is not used in any workflows. And you can only delete a screen scheme if it is not used in an issue type screen scheme. We'll cover issue type screen schemes soon.
Note: Although it is possible to associate any screen defined in your Jira installation with either a screen scheme or a workflow transition view, screen schemes and workflow transition views are distinct and unrelated.

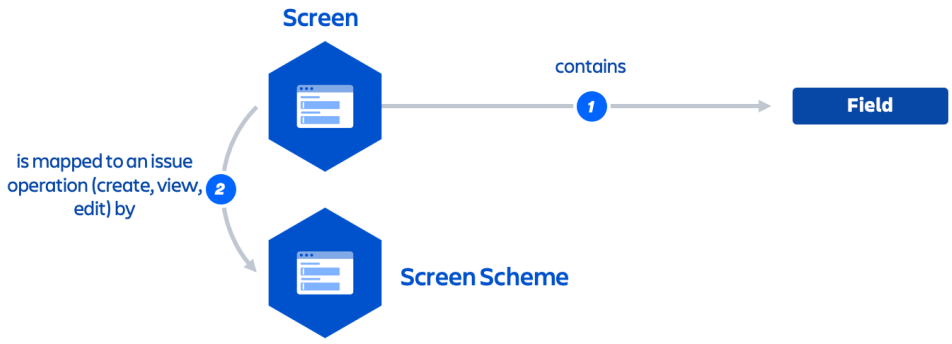
Teams In Space issue operation screen implementation

To make the DEV: Scrum Create Bug Screen only appear when users **create** issues:

1. Update the DEV project's default **DEV: Scrum Bug Screen Scheme**
2. Associate the new DEV: Scrum Create Bug Screen with the **Create Issue** operation

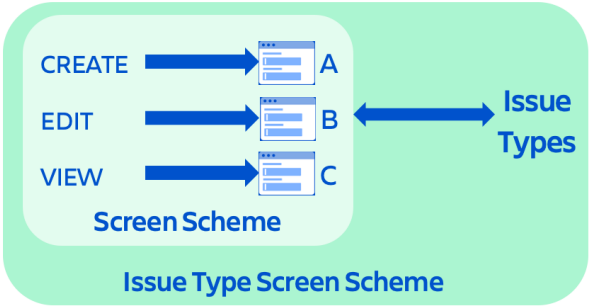
The Teams in Space Jira administrator has created a new DEV: Scrum Create Bug Screen by copying the project's default DEV: Scrum Bug Screen. They have then removed the Components, Fix Version/s, Affects Version/s, Linked Issues, Epic Link, and Sprint fields from the new bug creation screen. Now they update the DEV project's default screen scheme and associate the new bug creation screen with the create issue operation.

2. Mapping Screens to Issue Operations



Mapping screens for operations to issue types

Issue Type Screen Schemes associate screen schemes with issue types

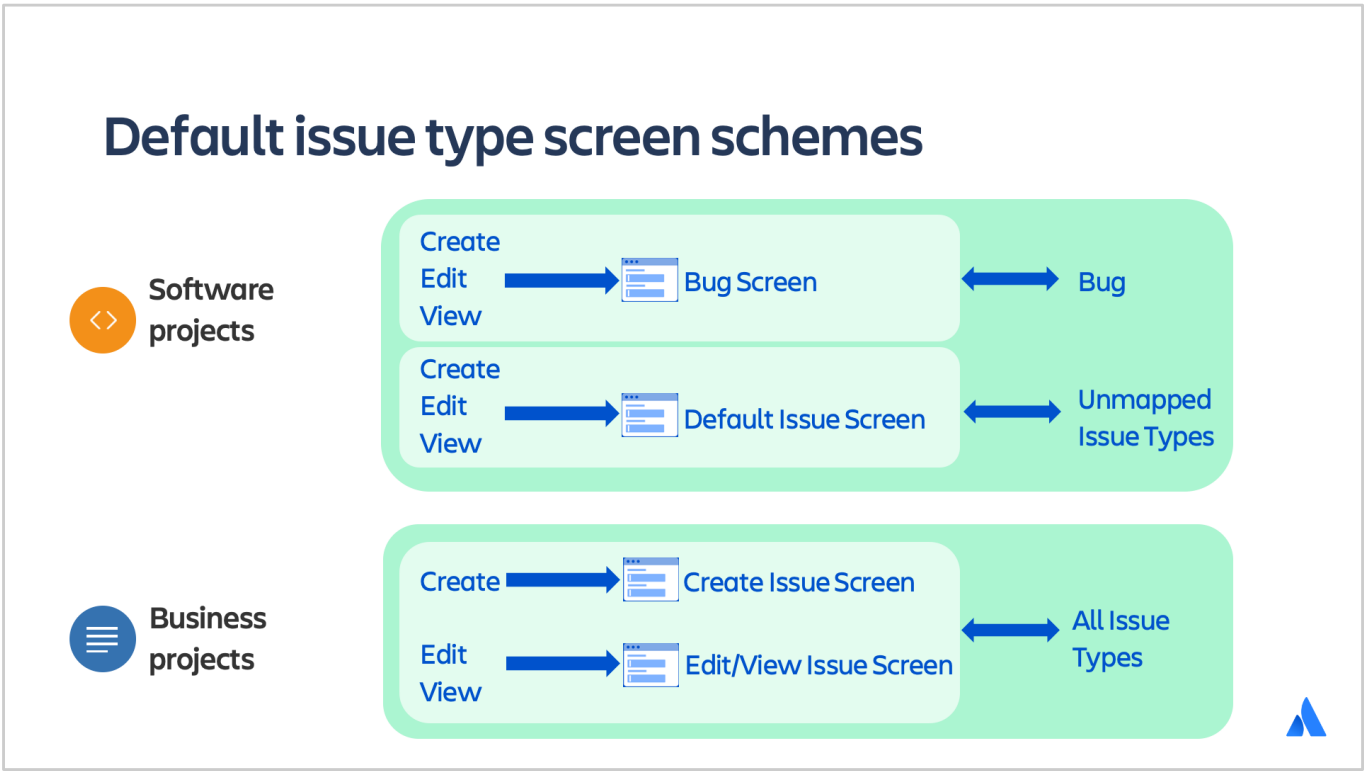


Once you have created your screen scheme, you will need to associate the screen scheme with an issue type or types via an issue type screen scheme. This allows you to specify different screens for particular operations, for each issue type. For example, you could use one screen when creating an issue of type 'Bug', and a different screen when creating an issue of type 'Task'.

If the field configuration scheme, issue type screen scheme, and workflow scheme associated with a given project contain associations with other issue types that are not specified in the project's issue type scheme, then those other issue types will be ignored by the project since the project's issue type scheme restricts what issue types the project can use.

Supplemental information:

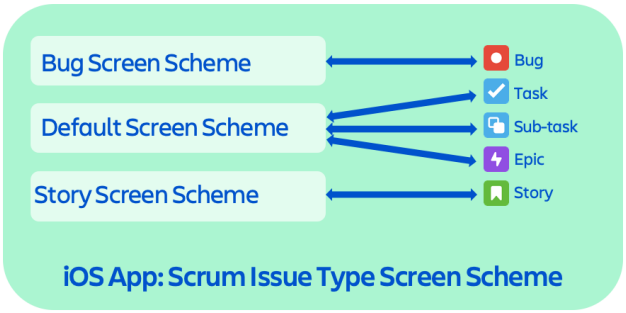
For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Associating+screen+and+issue+operation+mappings+with+an+issue+type.



When you create a project the default way (not by selecting Create with shared configuration), a new issue type screen scheme is created just for that project.
When you create a Software project, you get one issue type screen scheme that maps the Bug Screen Scheme to the Bug issue type and the Default Screen Scheme to all other unmapped issue types. This applies to all the Software project templates – Scrum, Kanban and Basic software development.
When you create a Business project, you get one issue type screen scheme that maps the Default Screen Scheme to all issue types. This applies to all the Business project templates – Project Management, Process Management, and Task.

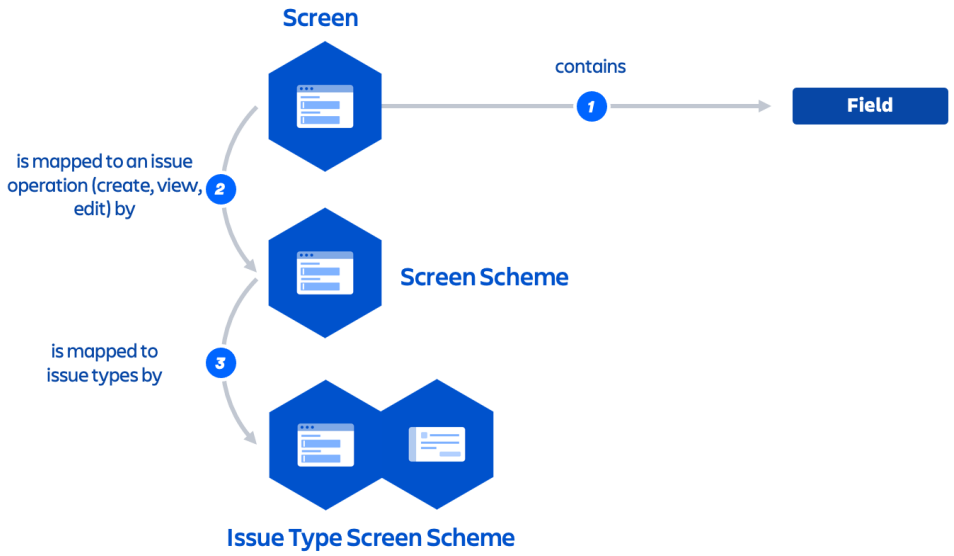
When to update a project's issue type screen scheme

- Update the Issue Type Screen Scheme for a project when you create a new screen scheme for a previously unmapped issue type
- For example, in a Software project when you create a new screen for the Story issue type



You would need to update the issue type screen scheme for a software project if you had created a new screen scheme for an issue other than Bug. For example, if you created a new screen for creating Stories. You would first associate the new screen with the create issue operation in a new create story screen scheme. Then you would associate the create story screen scheme with the Story issue type in the issue type screen scheme for the project.

Mapping screens for operations to issue types



Let's review...
You place fields on screens.
Then you map screens to issue operations - create, view, edit - in screen schemes.
Then you map screen schemes to issue types in issue type screen schemes.

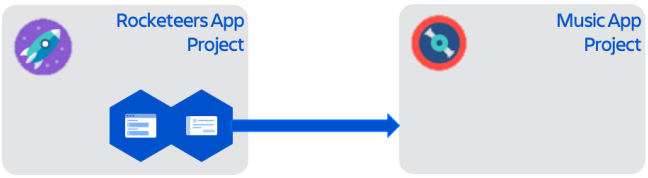
Teams In Space screen issue type implementation

To make the DEV: Scrum Create Bug Screen only appear when users create **bugs**, nothing needs to be done

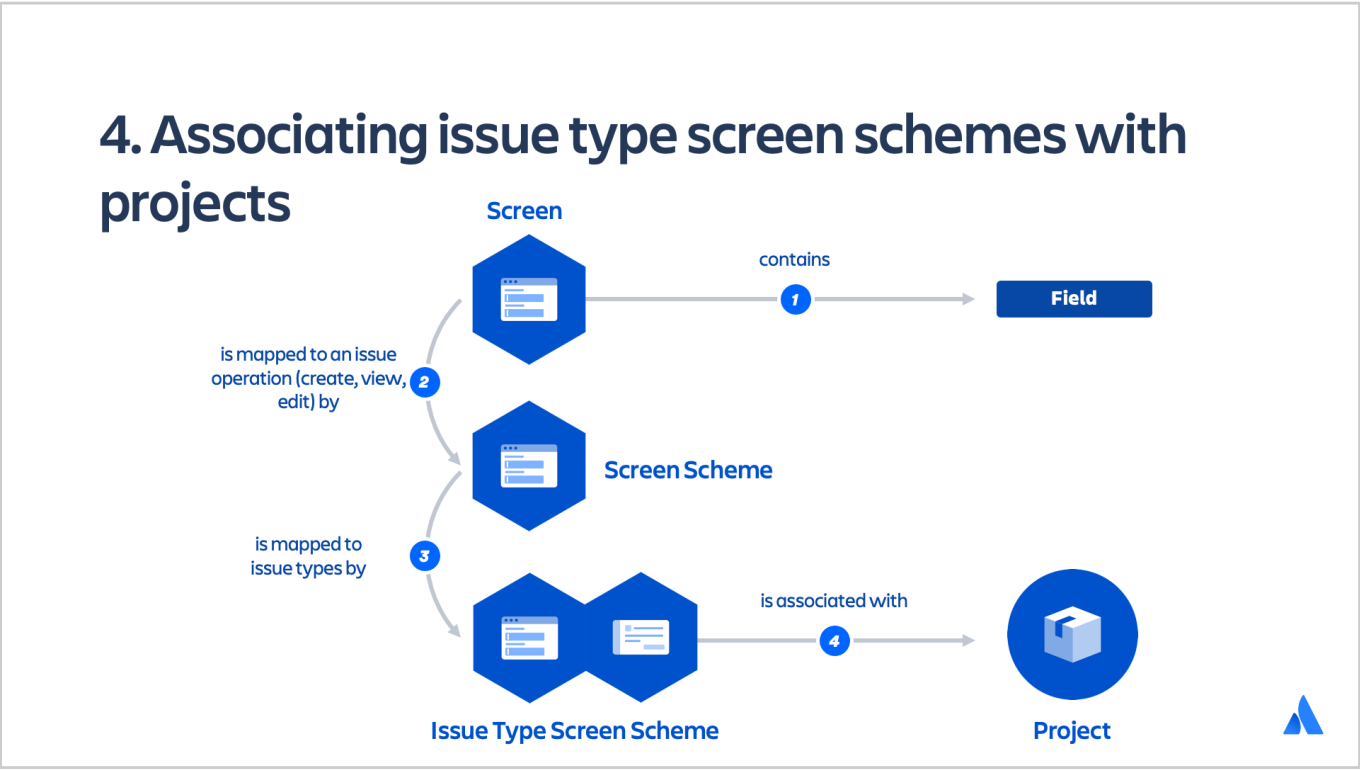
- By default, the DEV: Scrum Bug Screen Scheme is already associated with the Bug issue type in the project's **DEV: Scrum Issue Type Screen Scheme**

Associating issue type screen schemes with projects

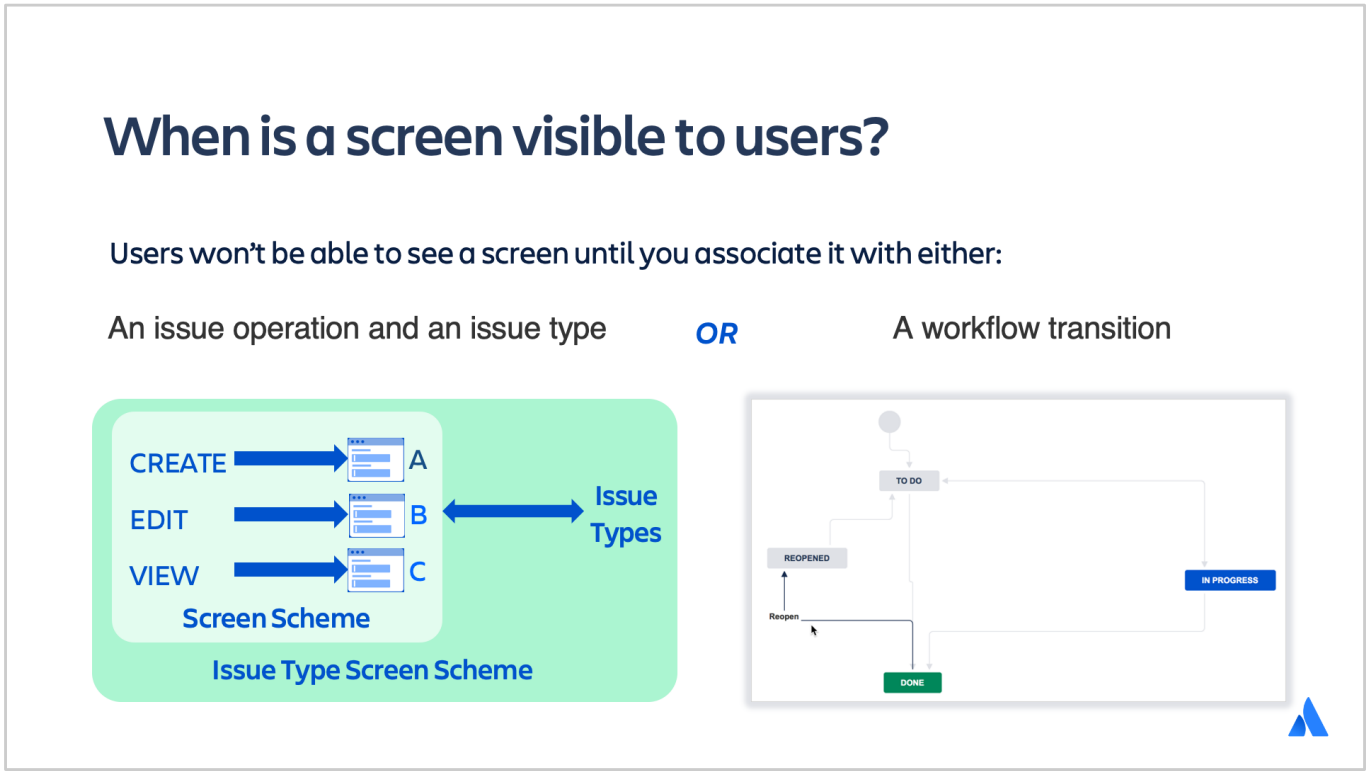
- When you create a project, you get an issue type screen scheme just for that project
- If you want the issue type screen scheme to apply to another project, you can associate it with the project



Associating issue type screen schemes with a project allows you to specify different screens for a particular operation (e.g. 'Create Issue'), for each type of issue in a given project. For example, you could use one screen when creating an issue of type 'Bug', and a different screen when creating an issue of type 'Task'.
When you create a project the default way i.e. not sharing project configuration, you get an issue type screen scheme created just for that project. You don't need to manually associate it with the project.
You only need to associate an issue type screen scheme with a project if you want it to apply to an already existing project that's different from the one you're editing.
Note: If the issue type screen scheme associated with a given project contain associations with other issue types that are not specified in the project's issue type scheme, then those other issue types will be ignored by the project since the project's issue type scheme restricts what issue types the project can use.



And this is the final piece in the issue type screen scheme diagram, showing how screens, screen schemes and issue type screen schemes fit together.



Any newly created screen in Jira is not usable by a Jira project, and so visible to a user, until it has been associated with either:

- An issue operation, for example create issue and an issue type via a screen scheme and then issue type screen scheme.

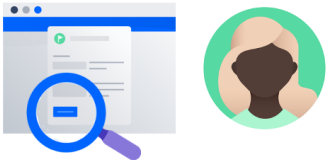
or

- A workflow transition, for example Resolve Issue. We'll discuss workflow transitions and screens in the next module.

When is a field visible to users?

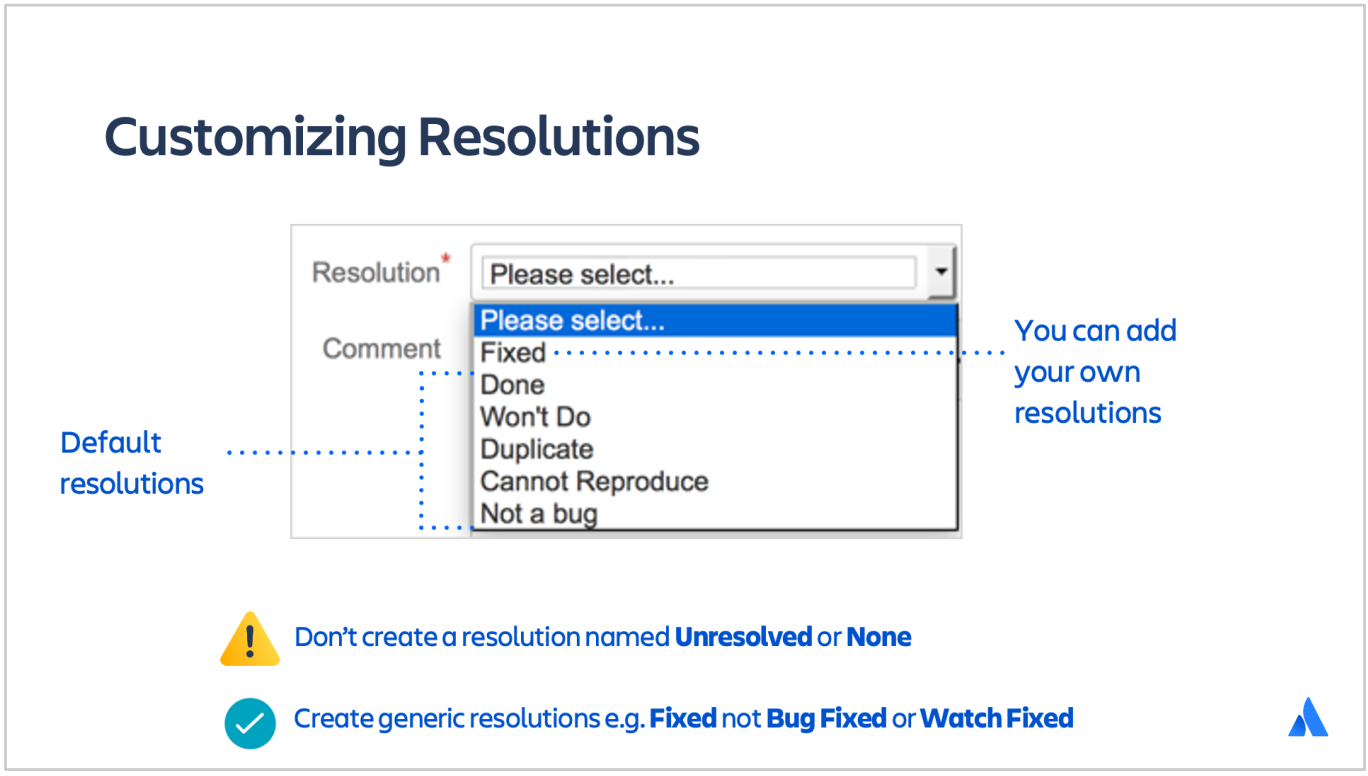
Users will only see fields that:

1. Are present on the screen associated with the issue operation
2. Are not hidden in the field configuration applicable to the issue
3. The user has permission to edit the field



Users will only see a field if certain conditions are met.

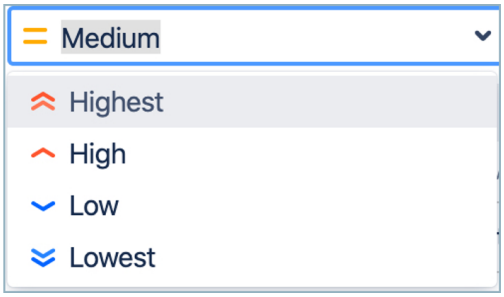
1. First the field needs to be present on the screen associated with the issue operation.
2. Also the field cannot be hidden in the field configuration for the issue (which is defined by the project's field configuration scheme).
3. Finally, the user must have permission to edit the field. There are no field level permissions. But some permissions affect fields that can be edited e.g. the Due Date field can only be edited by users with the Schedule Issues project permission.



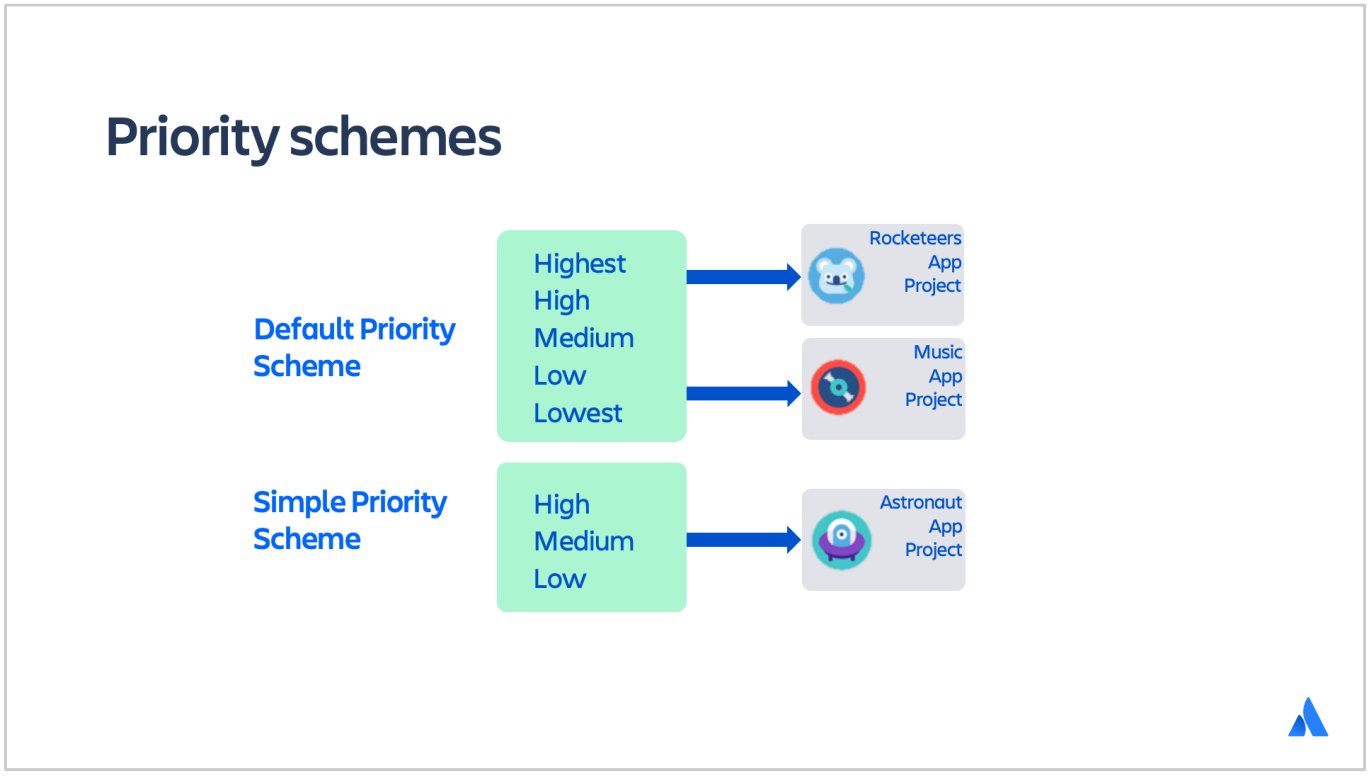
Resolution is a built-in field and it can be customized. Resolutions are the ways in which an issue can be closed. Jira applications ship with a set of default resolutions (Done, Won't Do, Duplicate, Cannot Reproduce, Not a bug), but you can add your own.
Don't create a Resolution named "Unresolved" or "None". Any issue that has the Resolution field set is treated by Jira applications as "resolved". The Issue Navigator displays Unresolved when no resolution is set for an issue. So adding a resolution named Unresolved/None and setting it in an issue will mean that the issue is seen as resolved. This will lead to confusion and is not recommended. Setting resolutions incorrectly is one of the biggest mistakes new Jira administrators make.
Create generic resolutions that are not specific to projects so they can be used by many different projects.
Supplemental information:
For more information on defining resolution field values, see https://confluence.atlassian.com/display/AdminJiraServer085/Defining+resolution+field+values.
For information on translating resolutions (as well as priorities, statuses, and issue types) see https://confluence.atlassian.com/display/AdminJiraServer085/Translating+resolutions,+priorities,+statuses,+and+issue+types.

Customizing priorities

- Create new priorities
- Edit or delete a priority
- Re-order priorities
- Change the icon and color
- Set the default priority



An issue's priority defines its importance in relation to other issues. Jira applications come with a set of default priorities, which are Highest, High, Medium, Low, and Lowest. You can create new priorities, edit a priority, re-order priorities, delete priorities, change the priority's icon and color, and set the default priority.
Supplemental information:
For more information on defining priorities, see https://confluence.atlassian.com/display/AdminJiraServer085/Defining+priority+field+values.
For information on translating priorities (as well as resolutions, statuses, and issue types) see https://confluence.atlassian.com/display/AdminJiraServer085/Translating+resolutions,+priorities,+statuses,+and+issue+types.



A priority scheme works like a mapping that allows you to associate a subset of priorities with particular projects. You can use it to achieve the following goals:

- Restrict the set of available priorities for a project.
- Control the order in which priorities are displayed.
- Select a default priority that is assigned to all newly created issues in a project.

In the example shown there the Rocketeers App and Music App projects are using the default priority scheme with 5 priorities (Highest, High, Medium, Low, and Lowest). Where as the Astronaut App project is using the Simple Priority Scheme with only 3 priorities (High, Medium, and Low).

Supplemental information:

For more information, see https://confluence.atlassian.com/display/adminjiraserver085/associating+priorities+with+projects.

See how it's done

View screen schemes and issue type screen schemes for Software and Business projects

Instructor demo.

Takeaways

- Think carefully before deleting custom fields
- Don’t create a lot of fields on screens – keep them simple

If you delete a custom field it will destroy any data contained in the field.

Don’t create a lot of fields on a screen. Too many fields means users will leave half of them empty and you can't drive business on bad data. Less is more and will ensure the integrity of the data and your reporting. Every data point should be an input to some process, not a nice to have, but used in the process lifecycle.

Simplify screens for users to make them easier to use – less is more. Remove unused fields as too many fields means:

- Unhappy users
- Inefficient use of time
- Custom field management difficulty

Try it

Lab 3 – Configuring Issue Types, Fields, Screens, & Schemes

- Exercise 3 – Configuring Screens
- Optional Exercise 4 – Configuring Priorities & Resolutions

4 Customizing Workflows

Course Overview

Mapping Your Business into Jira

Configuring Issue Types, Fields & Screens

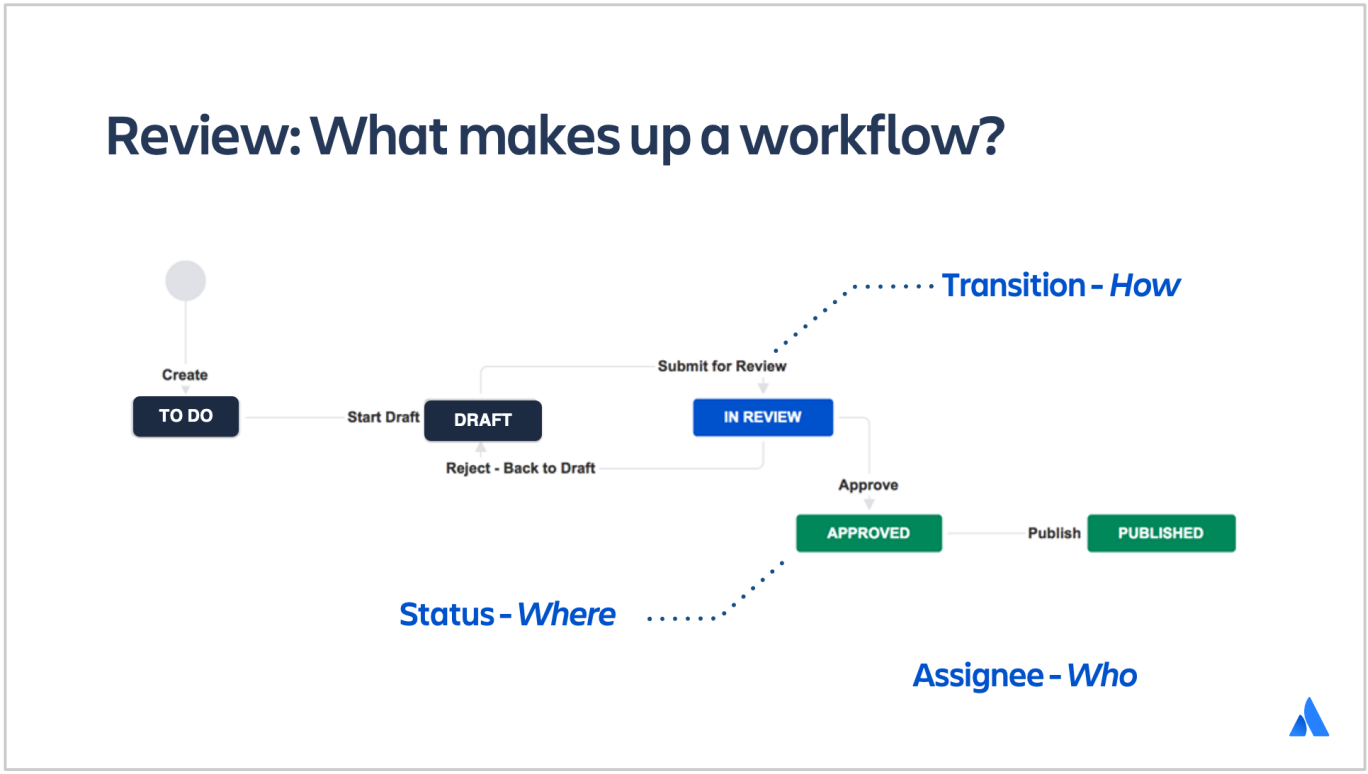
Customizing Workflows

Configuring Board & Sprint Permissions

Applying New Configurations to Projects

What will you learn?

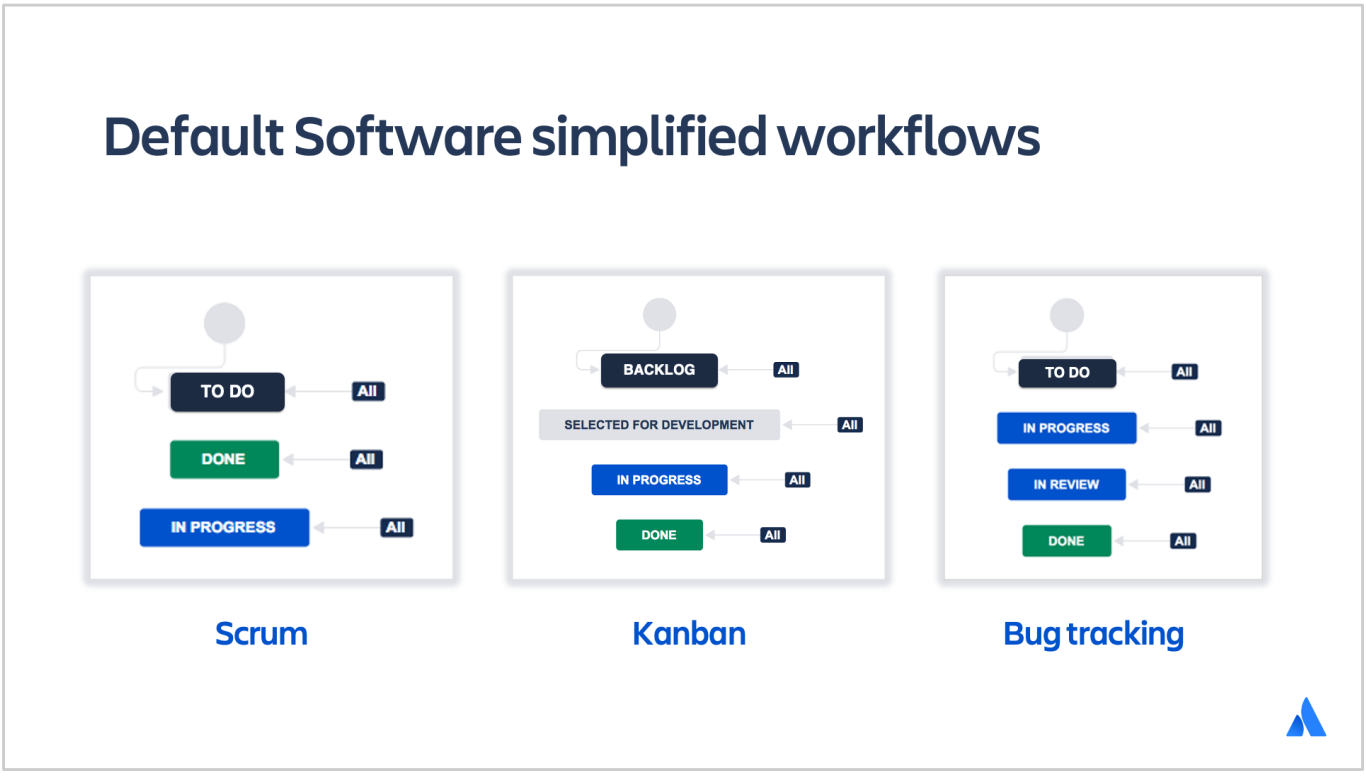
- Customize a workflow transition using conditions, validators, and post functions
- Create a workflow scheme and associate it with a project
- Configure a board to match a new underlying workflow
- Export and import a workflow
- List workflow best practices



Here is an example content management workflow (one of the Business templates). Each content item (e.g. article, course, etc.) could be stored in Jira as an issue and administered in the workflow shown here.
A workflow is made up of statuses and transitions.

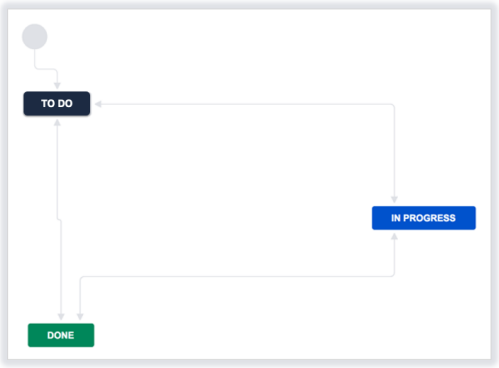
- **Statuses** represent the position of the issue in a workflow. In this workflow, an article can be identified as to do, in draft form, in review, approved, and published. Every status should be a unique state in your workflow and describe what's already happened.
- **Transitions** are the way a particular issue moves from status to status. Articles are submitted to review, rejected and sent back to be written again, approved, and published.

Assignees are also important in workflow - workflows guide how people work together. In Jira, the assignee dictates who's responsible for an issue. When books are in draft form, the assignee is set to the writer. When the book is reviewed the assignee may be a senior writer, and so on.

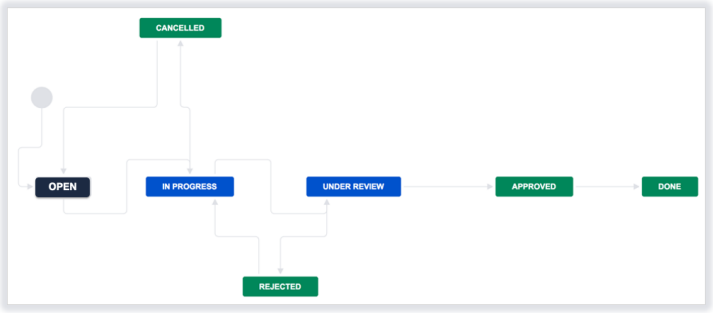


Each template for each project type (Software, Service Management, and Business) has a different default workflow.
Here you see the default workflows for each of the templates for the Software project templates. By default, all Software projects use the simplified workflow. This means issues can transition from any status to any other status (and developers can freely drag issues between columns on a board). Note the 'All' label beside each status which indicates this. The simplified workflow also means there are no screens displayed on transitions and the workflow can be edited via board settings.

Default business workflows



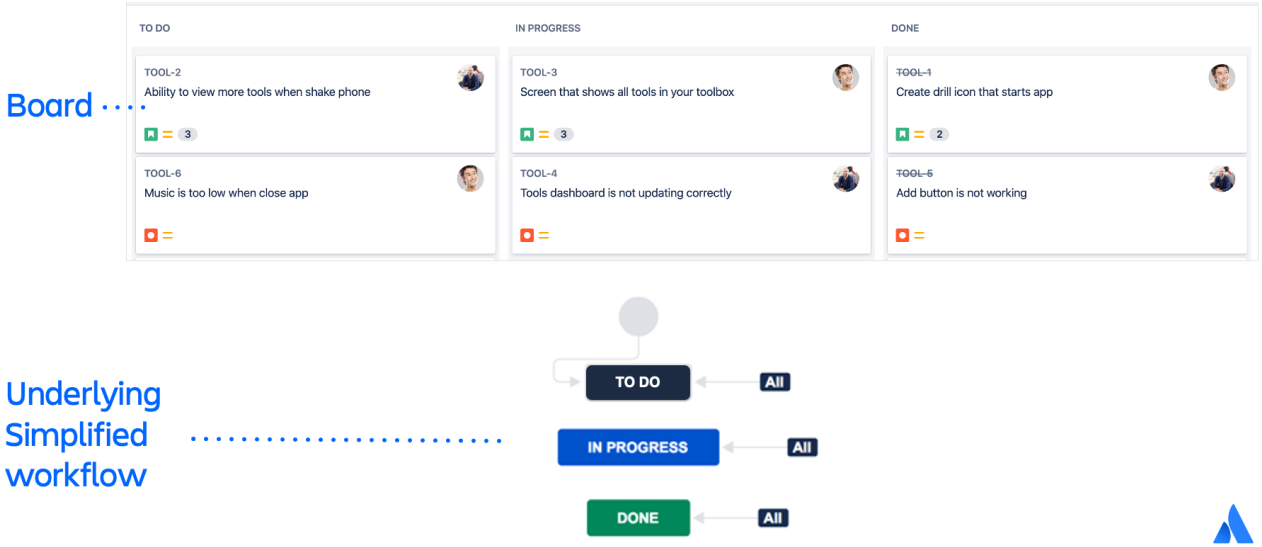
Project management



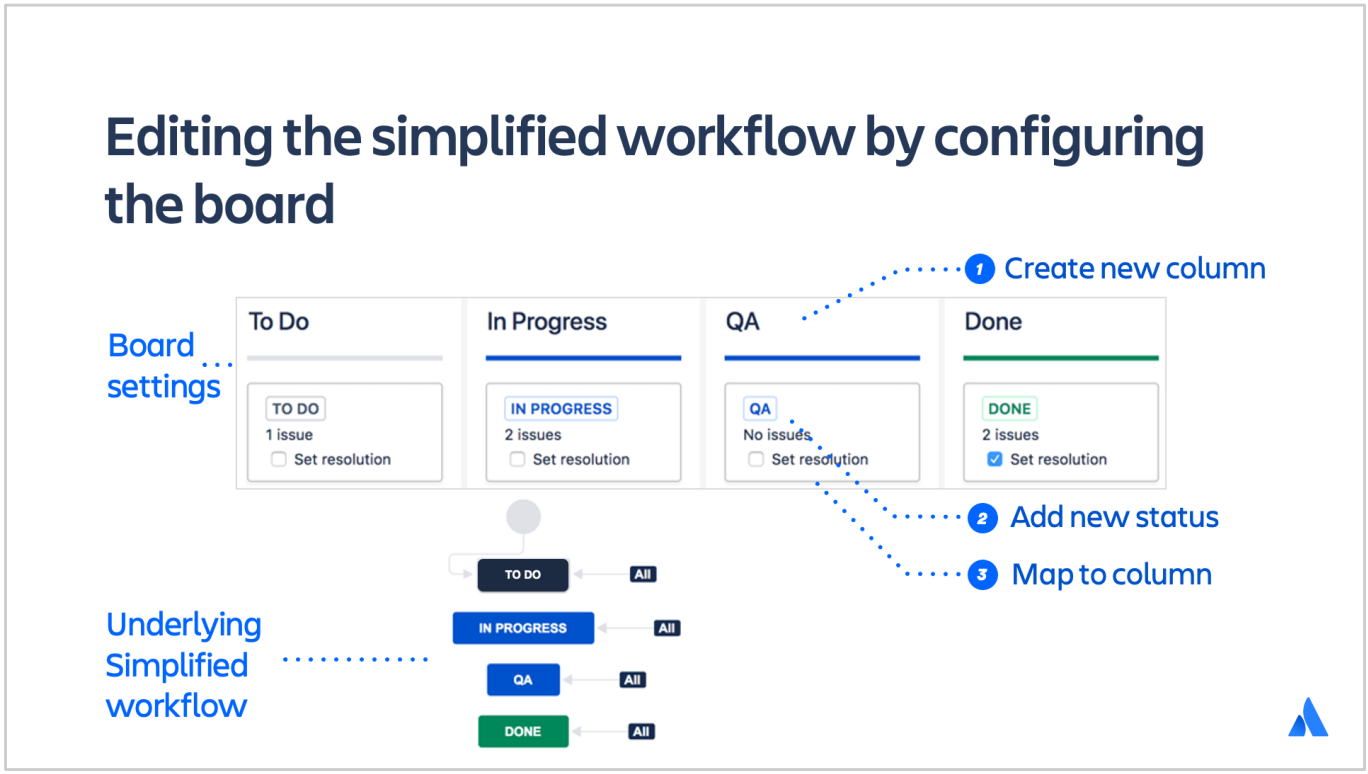
Process control

Here you see the default workflows for two of the Business project templates.
On the left you see the default workflow for the Project management template. It's a simple workflow with issues typically flowing from TO DO to IN PROGRESS to DONE. But note the double headed arrows meaning issues can also flow from DONE back to TO DO or IN PROGRESS, and from IN PROGRESS back to TO DO.
On the right you see the default workflow for the Process control template. It's a more complex workflow with a number of statuses and different ways issues can progress through the workflow.

Review: Boards and their underlying workflow



Boards are tied to their underlying workflow. For Software projects, the simplified workflow is used for the board that's created when the project is created.
Here's an example of the simplified workflow for a Software project using the Scrum template. The top screenshot shows the sprint board in the project. The diagram at the bottom shows the underlying workflow. Each column represents a status in the workflow. In this example the workflow has three statuses - TO DO, IN PROGRESS, and DONE.
Moving an issue on the board changes the value of its status and its position in the workflow. And because it's the simplified workflow, you can move an issue from any column on the board to any other column.



You can edit the simplified workflow by configuring the board. In this example of a Software Scrum project, we've first created a new column for the board called QA. Then we've added a new status also called QA and finally we've mapped it to the QA column. The underlying simplified workflow now has a new QA status. Users can still move issues freely on the board from any status to any other status.

Who can edit the simplified workflow for a board?

Configure the board	Jira admin	Project admin	Board admin only
Add statuses	✓	✓	
Add columns	✓	✓	✓

⚠ Be watchful of project admins creating a lot of statuses

To add a status to a board the project must be using the simplified workflow. You need to be a Jira administrator, a project administrator, or a board administrator to view board configuration. Only Jira administrators and project administrators can add statuses and columns when configuring the board. If you are only the board administrator (and not a Jira administrator or the project administrator) you can configure the board, but you can only add columns to the board but not statuses to the simplified workflow.
So project administrators can work with their project's simplified workflow and not get a Jira administrator involved. But be watchful, as you can end up with a lot of statuses when project administrators can freely add their own statuses to their boards in their projects!

Simplified workflow vs. Custom workflow

- A simple workflow
- Easy to use – no screens for transitions
- Users can drag issues freely between columns on a board
- Project administrators can edit the workflow via the board

- A more complex workflow
- Has transition screens, conditions, input validation, etc.
- A specific sequence of steps
- Project administrators can edit the workflow in a limited way
- Only Jira administrators can fully edit the workflow

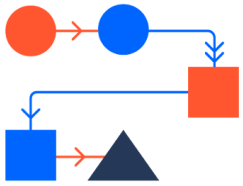
The simplified workflow, as its name suggests, is a simple workflow. We saw some examples earlier for Scrum and Kanban projects. It's easy to use the simplified workflow. There are no screens for transitions and users can transition an issue from any status to any other status (they can drag issues freely between columns on the board). Also, project administrators can edit the workflow via the board configuration, adding statuses and columns.

You use a custom workflow when you want a more complex workflow to manage a specific process. You may want screens to pop up during transitions that allow users to enter data, for example, a resolution. Also, you can add conditions, user input validation, and automated functions to transitions in the workflow. With a custom workflow, you can build in a specific sequence of steps rather than allowing every status to transition into any status.

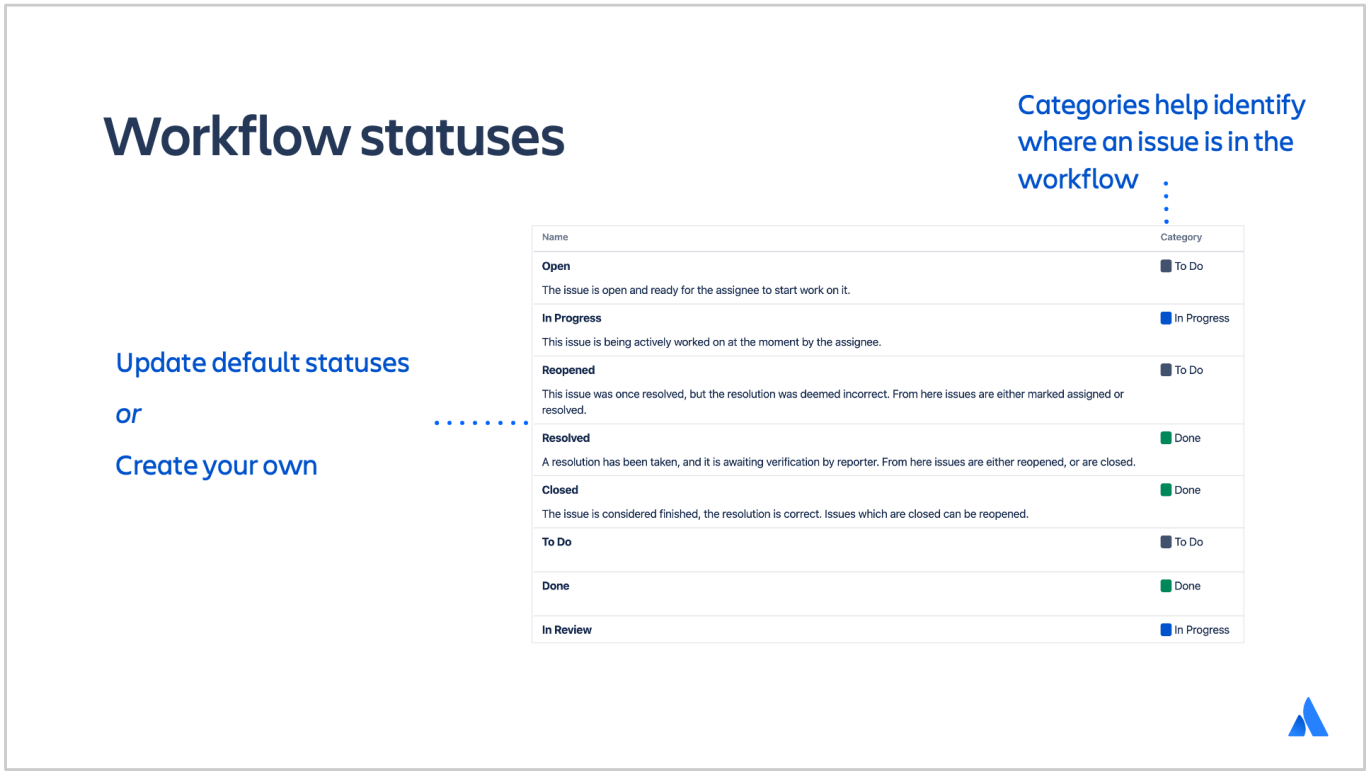
Finally, only Jira administrators can edit all aspects of a custom workflow using the workflow editor, so you have more control. Project administrators can add statuses and transitions if they are using a project with extended project permission. However, they are only able to perform a limited amount of configuration under specific circumstances. They cannot, for example, edit a workflow if it is shared with another project.

Creating new workflows

1. Gather your stakeholders & design the workflow
2. Create new statuses
3. Use the workflow editor to build the workflow
4. Configure the board to match the workflow
5. Test the workflow



If the default workflows don't meet your needs, you can create your own. But before you create a new workflow, gather all your stakeholders. When you build a process around a set of people, identify all the stakeholders in that workflow. For example, if you're building a workflow between product management, software development, and support, ensure you have one representative from each team in the meeting. Talk with each stakeholder about what's important to them. Before the meeting starts, draw a draft workflow on the whiteboard, then walk through each case in the meeting and gather feedback. Then design the workflow to meet everyone's needs.
Then you may need to create new statuses. You use the workflow editor to build the workflow. Once you've done so, you configure the board to match the workflow. Then, you'll test the workflow before making it live.
Let's look at this in more detail.



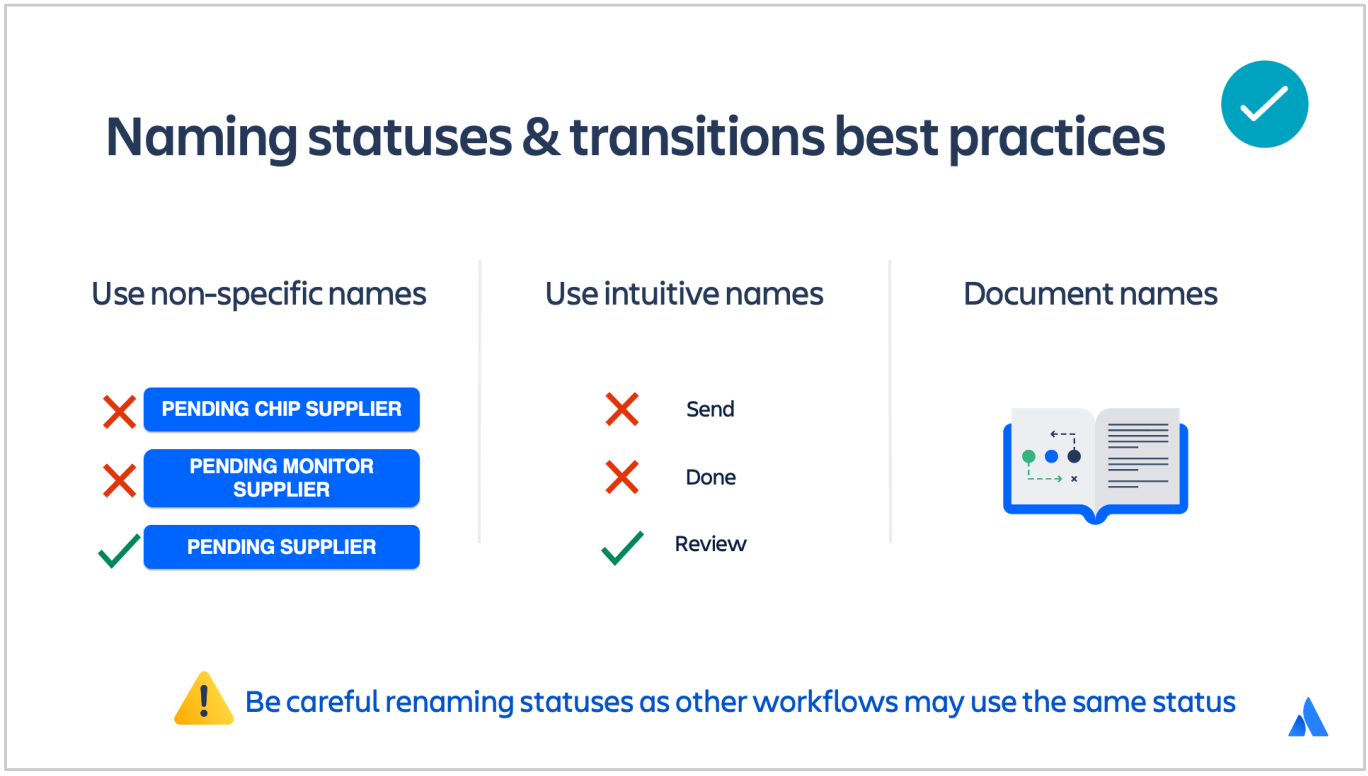
Jira ships with a set of default statuses that are used by the default workflows. A portion of the statuses Jira administration page is shown in the screenshot. Jira administrators can update the default statuses here. If the default statuses don't meet your needs, you can create new statuses.

When creating a status, you specify a category. Choose a category that this status will be grouped into: 'To Do' (grey), 'In Progress' (blue) or 'Done' (green). Categories help you identify where issues are in their lifecycle, particularly in places where many issues are rolled up, for example, the Version Details page. The category is also used to map statuses to columns in Jira Software when creating a new board for an existing project.

Creating new statuses

- Create a status whenever:
  - There's a handoff from one person to another
  - A piece of work is going to be in that same status for a long time
- Question requests for new statuses
- Don’t create a lot of statuses

Create statuses that reflect how people do their work. Create a new status when work needs to be re-assigned or handed off to another person. Another time to create a new status is when a piece of work is going to be in the same status for a significant period of time. This period of time will differ depending on the workplace and the overall period needed for the workflow to be completed. In most cases, it will be over a day, but it might be a few hours, depending on the turnaround time for issues in that environment.
Question requests for new statuses, don't just create them automatically. For example, Bill, the QA manager, wants to add a new status called 'Failed Verification' for all issues that don't pass review by his team. I'd advise against doing this, as the test engineers can send any issue that fails review back to a previous status, such as 'In Progress' or 'In Development'.
Don't create too many statuses. You want to get fine-grained visibility into the status of work but building a workflow with 20 statuses results in a workflow nobody wants to use. Start with a simple workflow and add statuses when you need them. Make workflows that people like to use. Don't exceed around seven statuses unless it's complex change management or a shared workflow where different phases will be used by different teams.



It's important to give statuses and transitions names that are easily understood by all. Give statuses generic (non-specific) names so they can be shared. For example, rather than 'Pending Chip Supplier' and 'Pending Monitor Supplier,' use 'Pending Supplier.' Transition and status names should be intuitive. Users should never have to wonder which workflow transition to use next. For example, in a content management workflow, if the writer has finished their draft and needs to have it reviewed, use' Review' for the transition name rather than 'Send' or 'Done,' which could be confusing. 'Send' could be interpreted as 'Send for Review' or 'Send for Publication' or some other send. 'Done' is confusing because this implies everything is complete with the issue, which it is not. If someone looks at a transition or a status and is confused about what work is being performed at that point, review the status names, and make sure that they are intuitive. It's also good to document transition and status names, ideally using Confluence. Include documentation of transition behaviors, transition properties, and status properties. Documentation facilitates both training and collaboration without having to provide direct access to the Jira workflow.

Be careful when renaming existing statuses as other workflows may use that status. If you change a status name, workflows that also use that workflow will also update. Without any notification, users will be confused. Filters based on that status name must be updated, or they will produce unexpected results.

Creating custom workflows

- Create a new workflow by:
  - Copying and customizing an existing workflow
  - Creating a new workflow from scratch
  - Importing a workflow

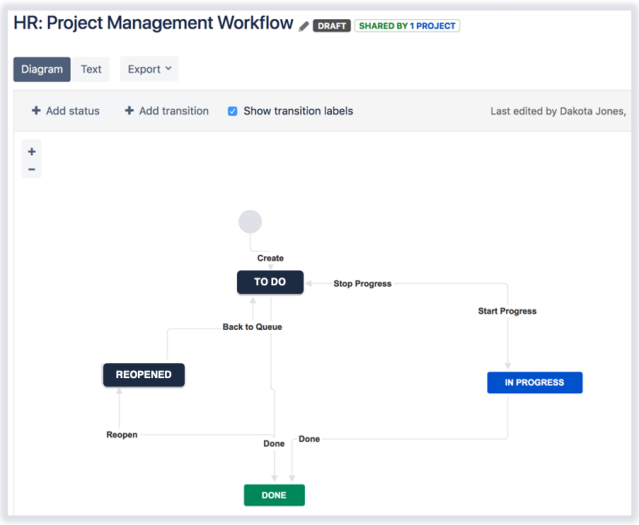
Only Jira administrators can create new workflows

Copy and customize an existing workflow rather than creating one from scratch

Give statuses generic (non-specific) names so they can be shared. For example, rather than 'Pending Chip Supplier' and 'Pending Monitor Supplier,' use 'Pending Supplier.'
You can create a new workflow using one of these methods:
•You can copy and customize an already existing workflow.
•Or you can create a new workflow from scratch.
•Or you can import a workflow from the Atlassian Marketplace (we'll go into this later in the module.)
Copying and customizing an existing workflow is usually the easiest way to create a new workflow as creating one from scratch can be a lot of work. Look for an existing workflow that's close to the workflow you want to create.
Only a Jira administrator can copy, customize, and create workflows.
Supplemental information:
See http://blogs.atlassian.com/2013/10/building-workflow-awesome/.

Using the workflow editor

- Create new, move, re-use, and delete statuses and transitions
- Add advanced features
- You can't modify a 'live' workflow – edit a draft then publish
- Save a backup copy when publishing edited workflows



When you create a project, the project gets a copy of the workflow for the project type and template. You can change this workflow, and the changes will only apply to that project.

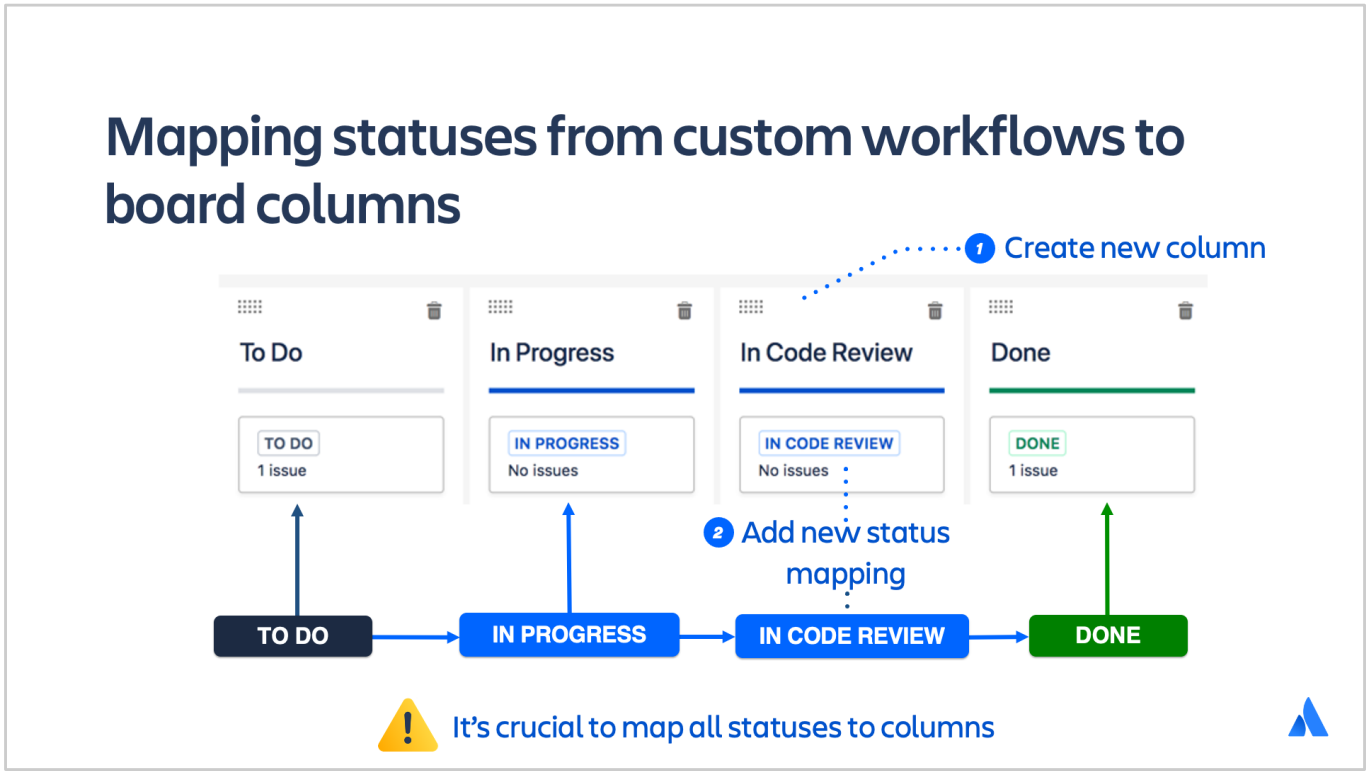
Using the workflow editor, you can create new, move, re-use, and delete statuses and transitions.

You can also create a more advanced workflow by adding conditions, validators, post functions, and transition screens; however, we won't be going into these in-depth in this course. See the **Getting More from Jira Workflows** course.

You can't modify a 'live' workflow; you can only edit a draft copy. Then you publish the draft, which replaces the live workflow. You also get the option to save the original workflow as a backup copy. It's always a good idea to do this in case you ever want to revert to the original workflow or use it somewhere else.

Supplemental information:

See https://confluence.atlassian.com/display/AdminJIRAServer085/Working+with+workflows.



If you've created a custom workflow for a project and added a status, for example, In Code Review, you then need to configure the board, so it matches your new workflow. You first create a new column on the board, which can be the same name as the status or a different name. Then you map the new status to that column.

You can also map a status to an existing column. You can map more than one status to one column. For example, an issue with the workflow steps "draft", "review", "approve" might all appear in the "In progress" column.

You must map all statuses to columns. Otherwise, issues in "unmapped" statuses will be hidden from the board.

Are you getting it?

You've created a new workflow for your project and added a new status. Some issues aren't showing on the board.

When you configured the board, what could explain this? (pick one)

a. You mapped the new status to a column that already contained a status.

b. You didn't map the new status to a column.

c. You mapped the status to a column with a different name than the status.

The answer is on the next slide.

Did you get it?

You've created a new workflow for your project and added a new status. Some issues aren't showing on the board.

When you configured the board, what could explain this? (pick one)

a. You mapped the new status to a column that already contained a status.

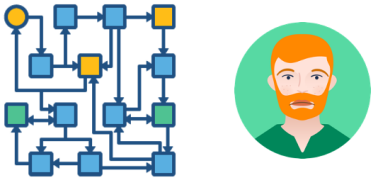
✓ b. You didn't map the new status to a column.

c. You mapped the status to a column with a different name than the status.

Answer: b. For issues that are in your new status to appear on your board, the new status needs to be mapped to a column on the board.
You can map a new status to a column that already contains a status. Columns can contain multiple statuses. And the names of the status and the column it's mapped to don't need to be the same.

Challenges when customizing workflows

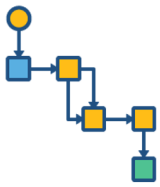
- Overly complex workflows that nobody wants to use
- Users stuck on statuses that don't transition anywhere
- Too many statuses



Do you have workflows that are so complex nobody wants to use them?
Do you have people stuck on statuses that don't transition anywhere?
Do you have far too many statuses in your workflow?
In Jira, there is some workflow validation that the system does to identify statuses that are missing transitions, and also validate transition permission conditions when importing workflows.

Workflow design best practices

- Keep your workflows simple!
- Keep the number of statuses and transitions to a minimum
- Make sure users don't get marooned on a status

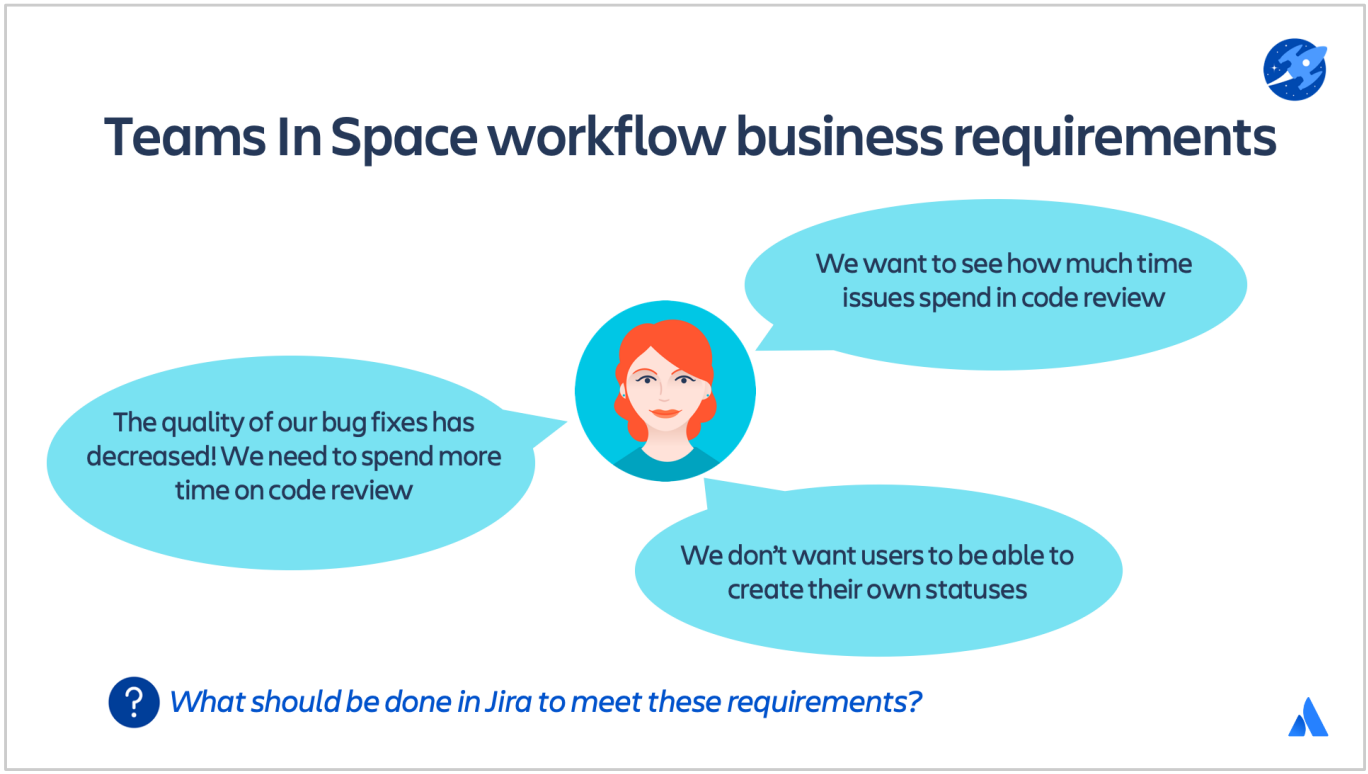


Keep your workflow simple! If they're too complex nobody will want to use them. Too many statuses and thus too many options of available workflow transitions will just confuse the user. Stakeholders often want to have statuses for each part of the workflow. That's generally a good thing but remember that each status adds more transitions and complexity. Aim for simple and scalable instead. Whenever adding a new status to a workflow, make sure you have no other option.
Ensure users don't get stuck on statuses that don't transition anywhere.
Supplemental information:
See http://blogs.atlassian.com/2013/10/building-workflow-awesome/.

Testing your workflows

- Go through each status & transition
- Build and test on a test instance
- Use a test project on a production instance

Once you've published your new workflow, test out an issue, make sure that it works as expected. Go through every status and transition and possible path through your workflow.
An alternative to working live, and preferred when working in a staging/production environment, is to build the workflow on your test instance and test it there. If you don't have a staging/production environment, then use a test project on your production instance and test your new workflow there.
Always test your workflows thoroughly before rolling out to your production projects. When you change a workflow, existing issues in a project are migrated, and changes you make may not be able to be undone easily.



At Teams In Space, all current development projects use the Default Simplified Workflow (TO DO - IN PROGRESS – DONE). Most teams have done the code review informally but now management wants to see where issues are so they can use this in reporting, etc. Also, they don't want to use the simplified workflow any more as users create too many statuses/columns which results in a lot of statuses, unique workflows, and so on.

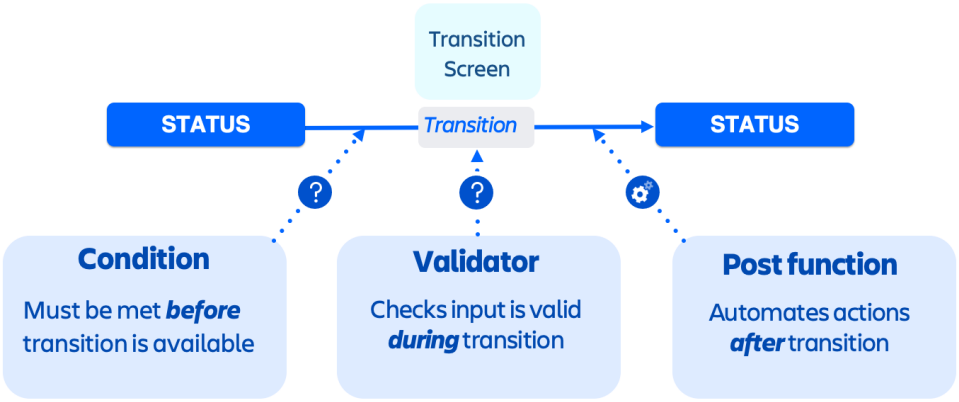
Teams In Space workflow implementation

Edit the Code Review transition and create:

1. A condition that only users in **Development** can transition issues to code review
2. A validator that the **Time Spent** field must have a value
3. A post function that copies the name from the **Assignee** field to the **Developer** field

To implement the validator and post function, you need to install a workflow app such as Suite Utilities for Jira. You can download apps from the Atlassian Marketplace at marketplace.atlassian.com. Note that add-ons are now called apps on the Atlassian Marketplace.

Configuring transitions



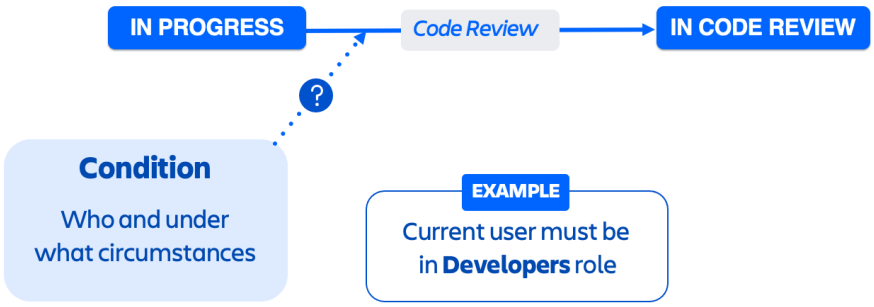
You can add conditions, validators, and post functions to transitions in your workflows.

A condition specifies a situation that must exist BEFORE something else is possible or permitted (pre-requisite). Conditions control if a transition is available (before it can be triggered).

A validator checks that any input made DURING the transition is valid before transition is performed.

A post function is an automated action AFTER a transition has been triggered.

Teams In Space condition



Teams In Space want just developers to be able to send an issue for code review (resolve an issue). This can be achieved by associating a condition to the "Code Review" transition.

Conditions control who performs a transition and under what circumstances. As examples, conditions can be used to allow:

•Only the reporter to execute a transition.

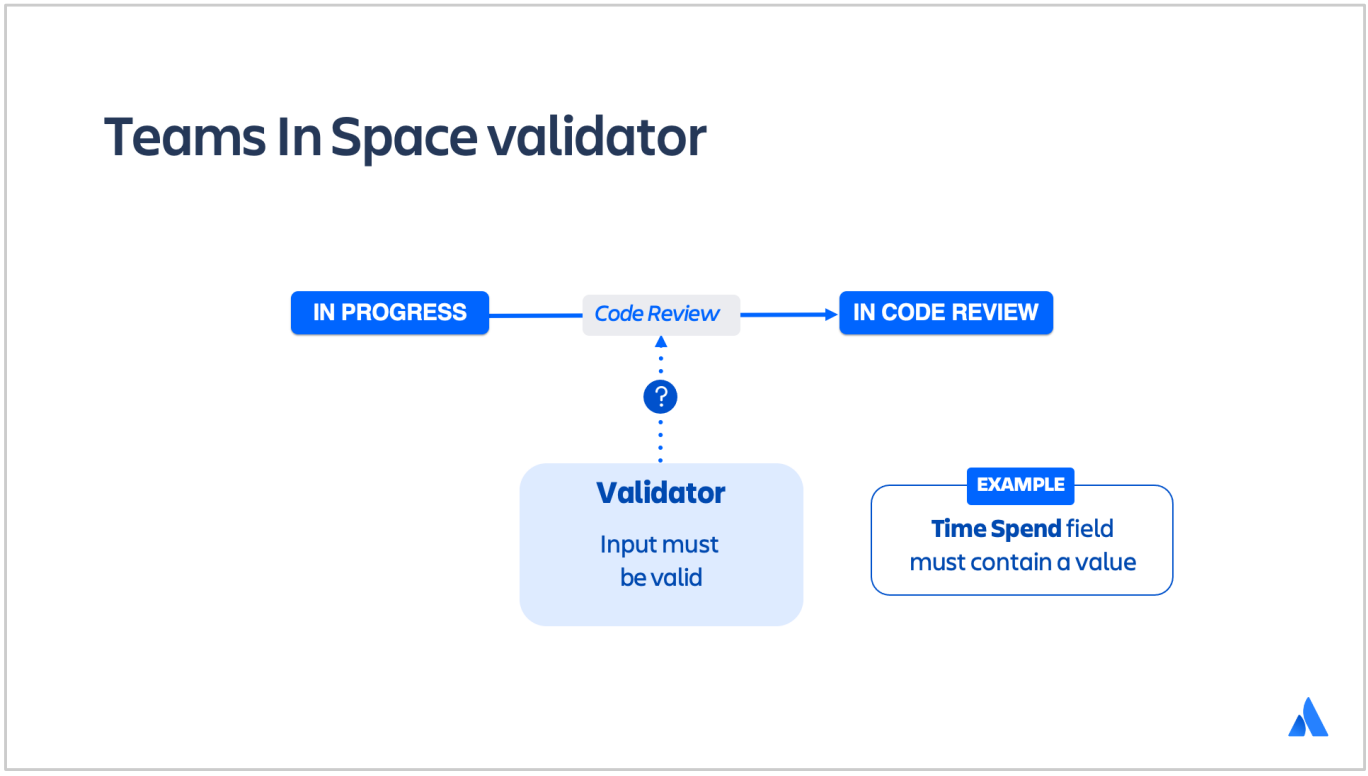
•Only users with a certain permission to execute a transition.

•Execution only if code has, or has not, been committed against this issue.

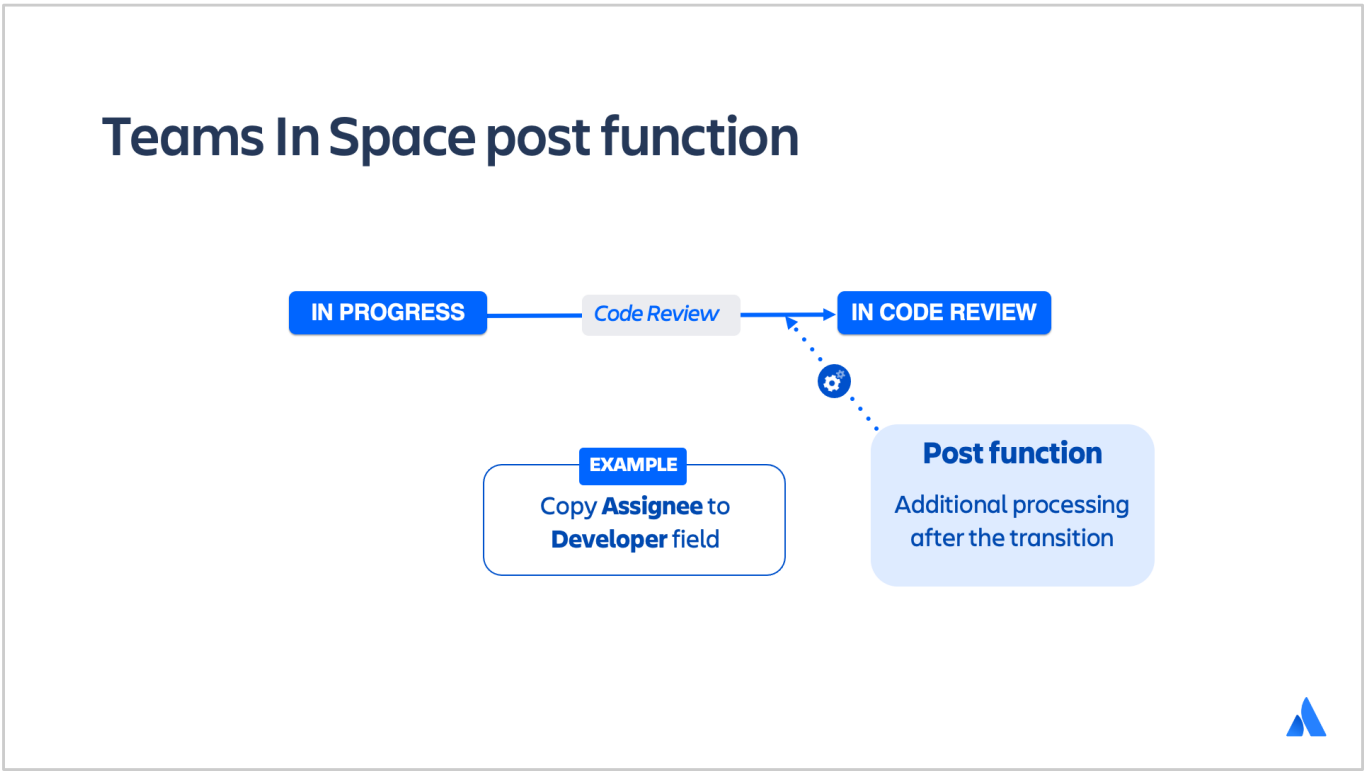
If a condition fails, the user will not see the transition button on the 'View issue' page, and so will not be able to execute the transition. Conditions cannot validate input parameters gathered from the user on the transition's screen – you need a validator to do this. You can also construct complex conditions by grouping and nesting conditions. Change any condition into a group by clicking the 'Add grouped condition' icon for the condition. Now you can add further conditions to this new group.

Conditions control if a transition is available (before it can be triggered).

Most common use cases involve either the status of the current user (permissions, roles, groups, etc.) = WHO or the current state of the issue (fields have certain values, code has been committed, etc.) = WHAT.



At Teams In Space, as soon as one of the developers fixes the bug (resolves the issue), we to ensure the Time Spent field has a value for reporting purposes. Validators check that any input made to the transition is valid before the transition is performed. If a validator fails, the issue does not progress to the destination status of the transition, and the transition's post functions are not executed.



Teams In Space wants to see the developer who did the coding shown in the issue. But often, the assignee is changed once the issue is in code review. This is the perfect opportunity for an automated post function to copy the value of the Assignee field to the Developer field. Post functions carry out any additional processing required after a transition is executed. A pre-requisite is that the conditions and validators have been passed. The most common post function is to clear the resolution when transitioning an issue back to an unresolved state. Other post functions are:

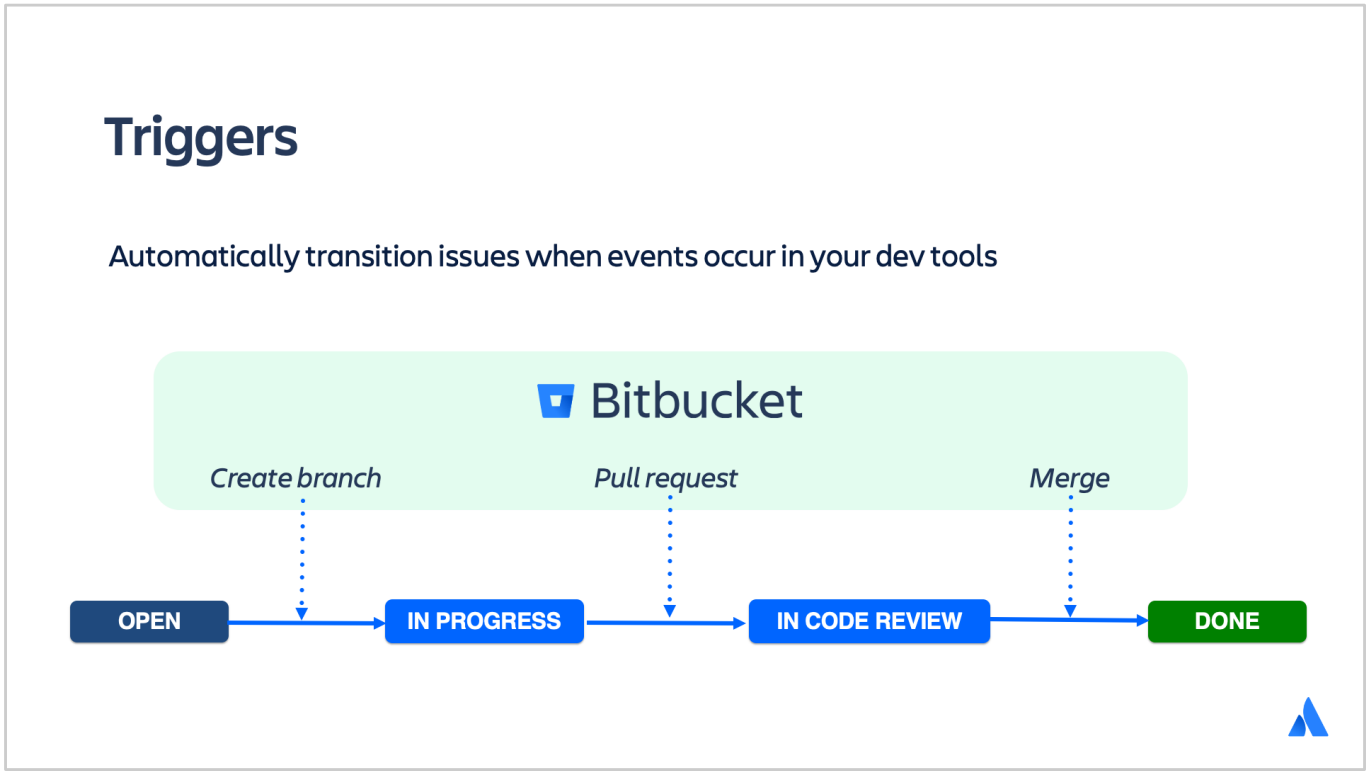
•updating an issue's fields e.g., change the assignee automatically or show a transition screen to choose who to reassign it to.

•generating change history for an issue

•adding a comment to an issue

•generating an event to trigger email notifications. For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Adding+a+custom+event

Every transition has a number of automatic essential post functions that cannot be deleted from a transition or reordered. However, you can insert other (optional) post functions between them. For the full list of Jira post functions, see https://confluence.atlassian.com/display/AdminJiraServer085/Advanced+workflow+configuration. Additional post functions may be available from installed plugins, or you can create your own post functions; see the Workflow Plugin Modules for details at https://developer.atlassian.com/display/JiraDEV/Workflow+Plugin+Modules.

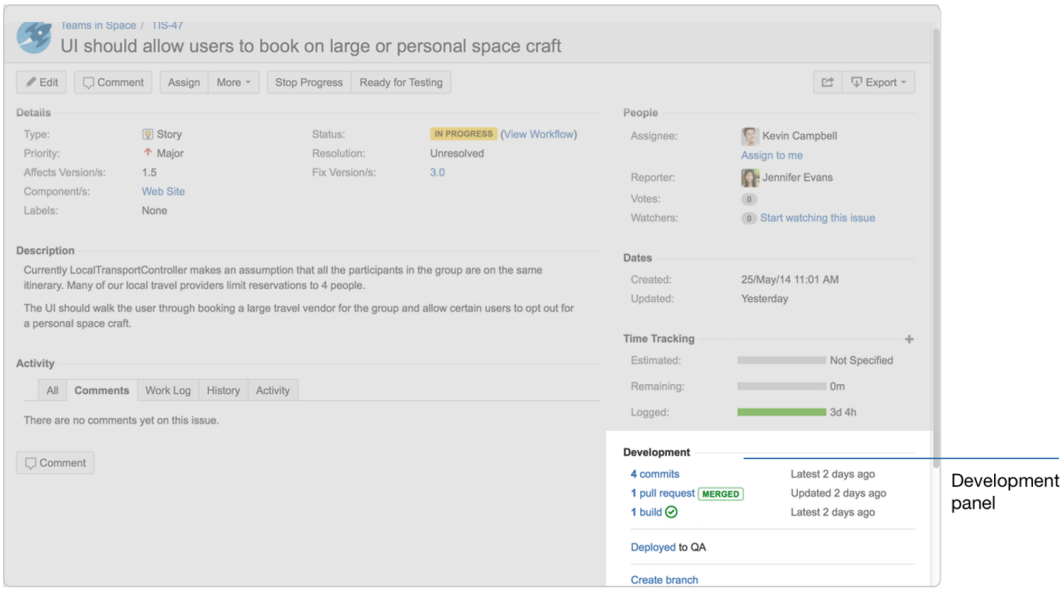


Jira administrators can configure triggers in Jira workflows that respond to events in your linked development tools. This allows you to set up your development tools and Jira workflows so that, for example, when a developer creates a branch to start work on an issue in Atlassian's Bitbucket, the issue will automatically be transitioned from 'Open' to 'In progress.' Then when the developer creates a pull request for other developers to do code review, the issue will automatically be transitioned to 'Code Review.' And finally, when the issue is merged in Bitbucket, the issue is considered Done.

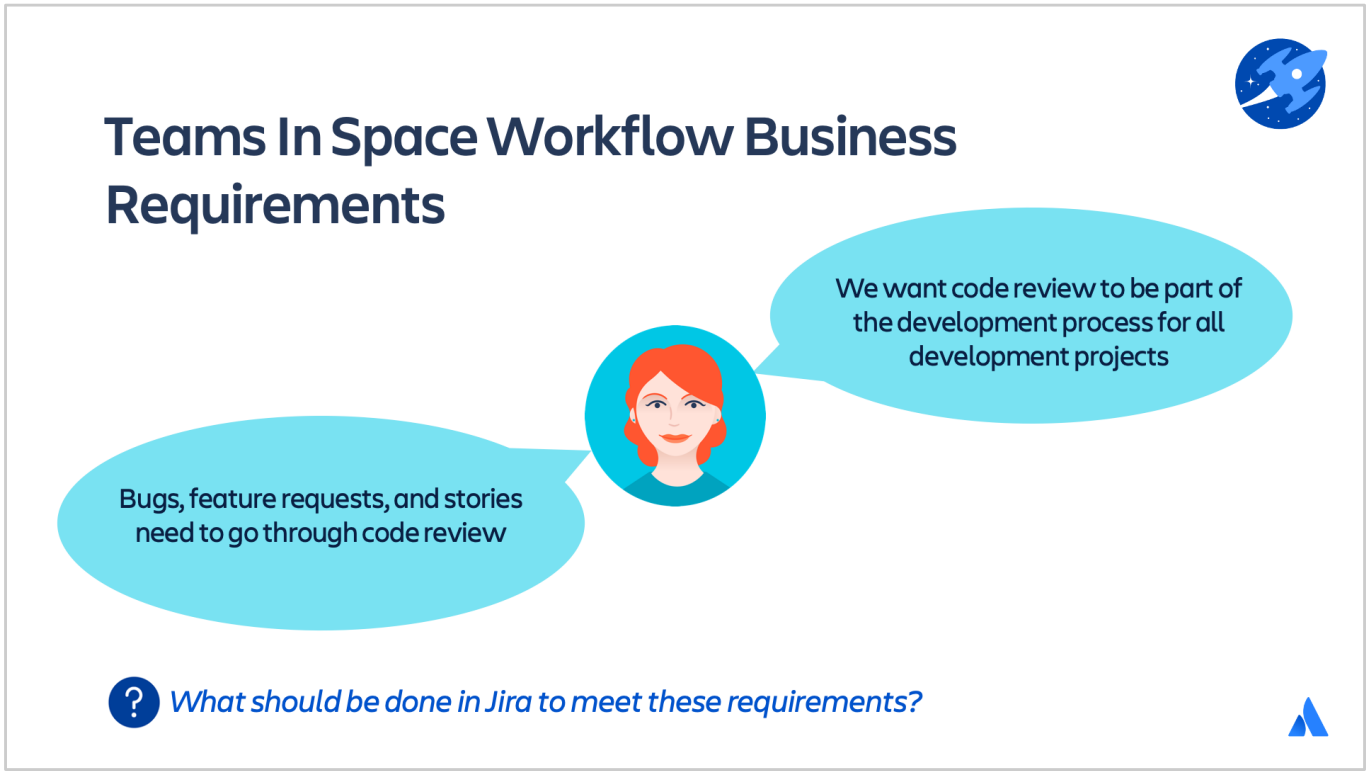
If you haven't set up a trigger before or you want to learn about triggers in more detail, see our guide on triggers at https://confluence.atlassian.com/display/AdminJiraServer085/Configuring+workflow+triggers.

The guide also shows you how to configure a workflow with triggers, similar to this sample development workflow: Development Workflow with Triggers (from Atlassian Marketplace) at https://marketplace.atlassian.com/plugins/com.atlassian.jira.workflow.sdwftriggers/server/overview.

Trigger example



Here we see an issue (story). The Development panel appears in the bottom right, and this is where developers can create branches, etc. The Development panel allows users in a software project to view development-related information on the issue, such as commits, reviews, and build information.
For users to see this Development panel, they need the 'View Development Tools' permission. This is set in the Default software scheme and is for Software users only. By default, any logged-in Software user has this permission and can see this panel.
For more information, see For more information on the development panel, see https://confluence.atlassian.com/display/JiraSoftwareServer/Viewing+the+development+information+for+an+issue.



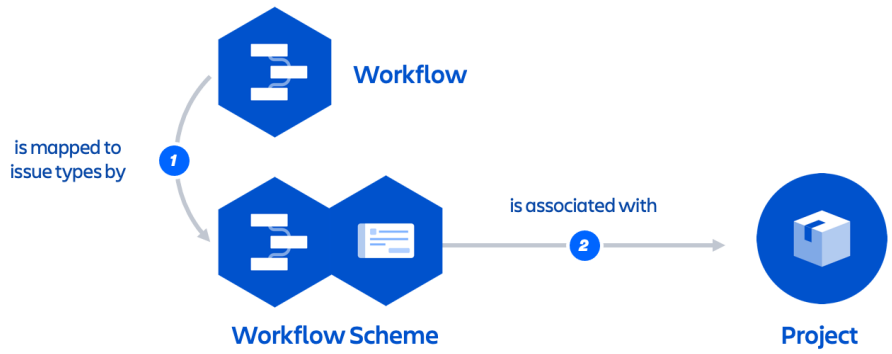
Teams In Space want to formalize their development process and add code review as a separate step (so they can generate reports, assign developers, etc.). However this will not apply to all issue types, only bugs, feature requests, and stories.

Teams In Space Workflow Implementation

1. Create the **DEV Workflow Scheme** and associate the:
   - DEV Code Workflow with the Bug, Feature Request, and Story issue types
   - Default Jira workflow with all unmapped issue types
2. Associate the DEV Workflow Scheme with the standard **DEV** project

Create the DEV Workflow Scheme and associate the new DEV Code Workflow with the Bug, Feature Request, and Story issue types. Associate the default Jira workflow with all unmapped issue types.
Replace the simplified workflow for the standard DEV project with the DEV Code Workflow (which only admins can configure) by associating the DEV Workflow Scheme with the standard DEV (Rocketeers App) project.

Workflow schemes & projects



Once you've completed your workflow, you associate with issue types in a workflow scheme. This is a defined set of associations – or mappings – between workflows and issue types.
Then you associate your workflow scheme with a project (or projects). This makes it possible to use a different workflow for every combination of project and issue type.

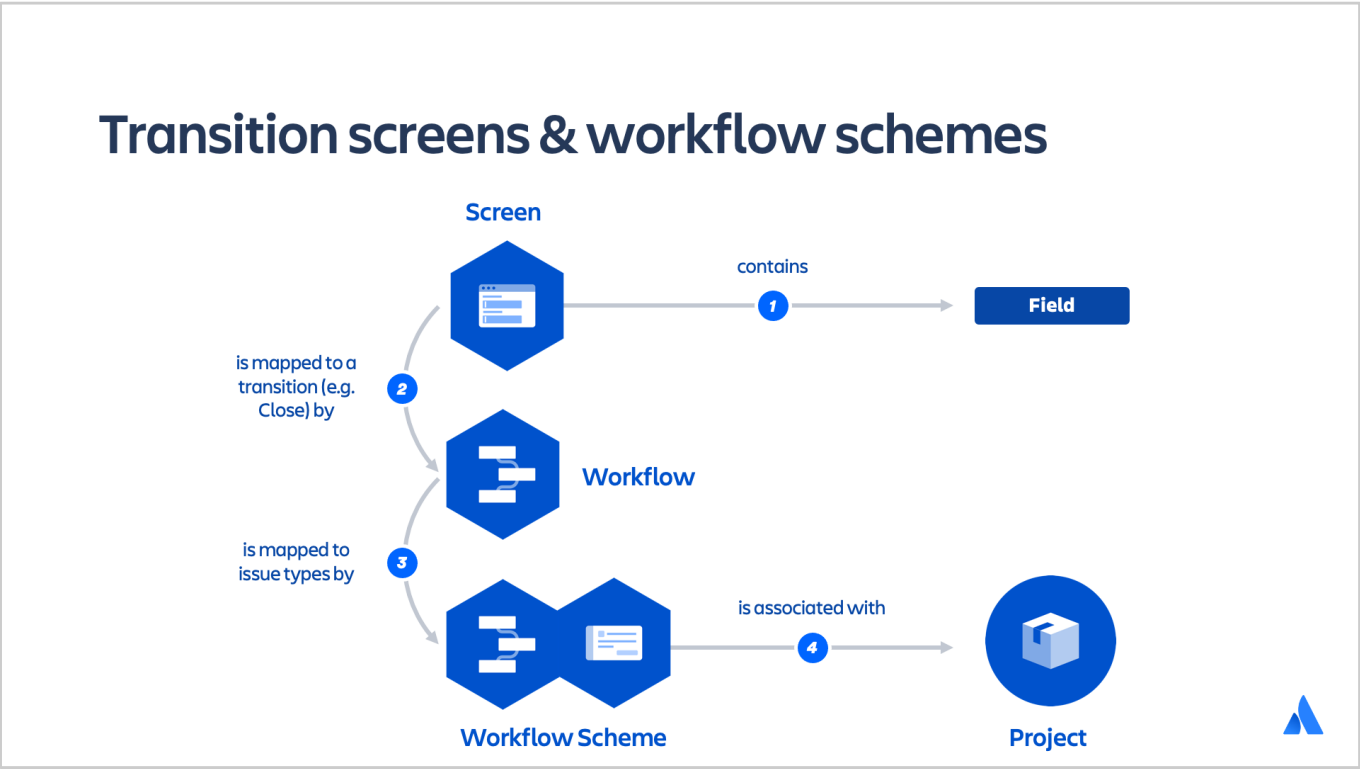
Transition Screens

- Transition screens are used when information is required as an issue moves through a transition
- You can create custom transition screens

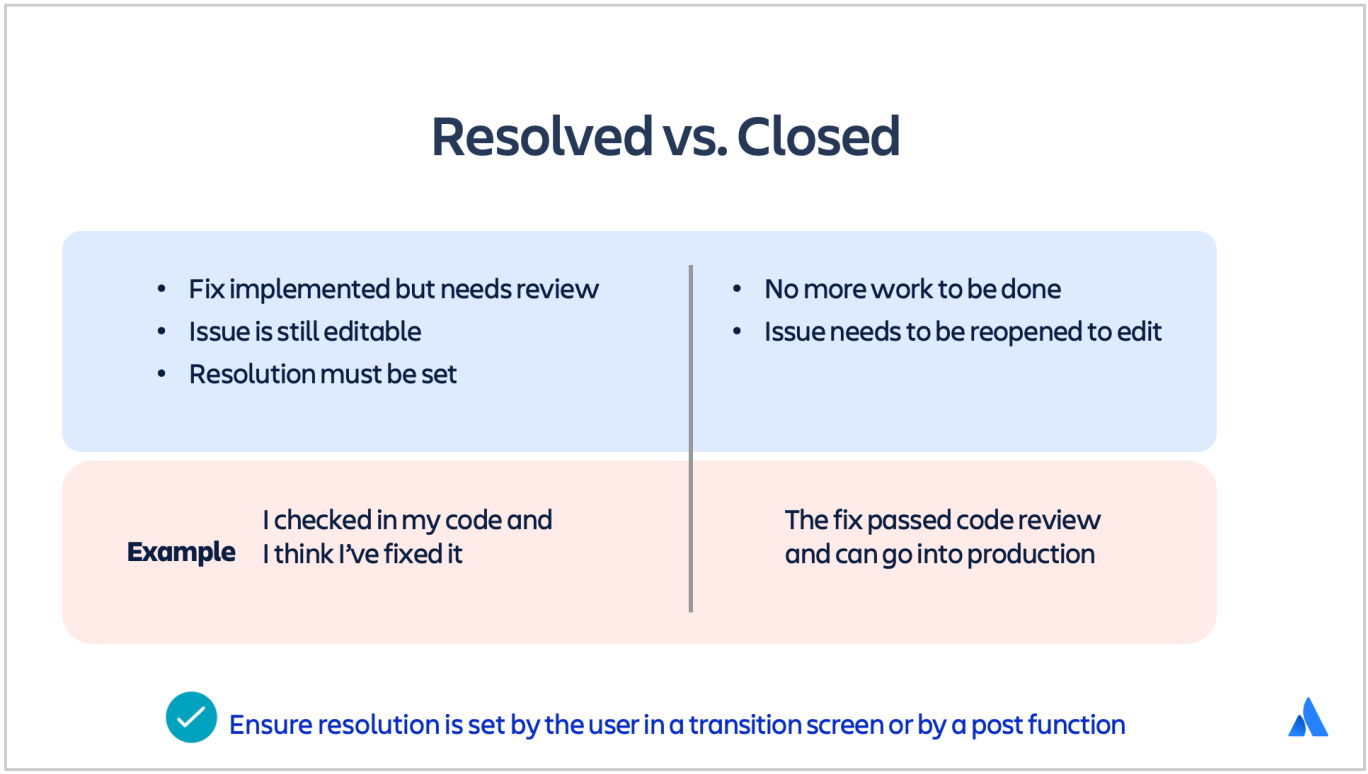
Users interact with issues to input data when they create, edit, view, and transition an issue. To choose the screens that are displayed when issues are created, edited, or viewed, map the screens to issue operations using Screen Schemes. We covered these in the last module.

The Transition Screen is used when information is required as an issue moves through a transition. For example, when an issue is resolved, we can indicate how the issue was resolved via the Resolution Field.

Jira supplies transition screens that are built into the default workflows, for example, Create Issue screens, Resolve Issue screens, Default Issue screens, Bug screens, etc. You can create your own custom screens and associate them with workflow transitions. To select which screen is displayed for a particular workflow transition, select the workflow the transition belongs to and edit it.



Screens can be customized to add or remove fields.
To make a new transition screen available to users, you associate the screen with a workflow transition, e.g., 'Close Issue.'
To make a new workflow available for certain issue types, you associate the workflow and the issue type(s) in the workflow scheme.
To use a different workflow for every combination of project and issue type, you associate the workflow scheme with a particular project or projects.
This allows you to specify different screens for a particular workflow transition, e.g., 'Close Issue' for each type of issue in a given project. For example, you could use one screen when closing an issue of type Bug, and a different screen when closing an issue of type Task in the Design project. Then you could use a completely different screen to close both bugs and tasks in the Performance project.



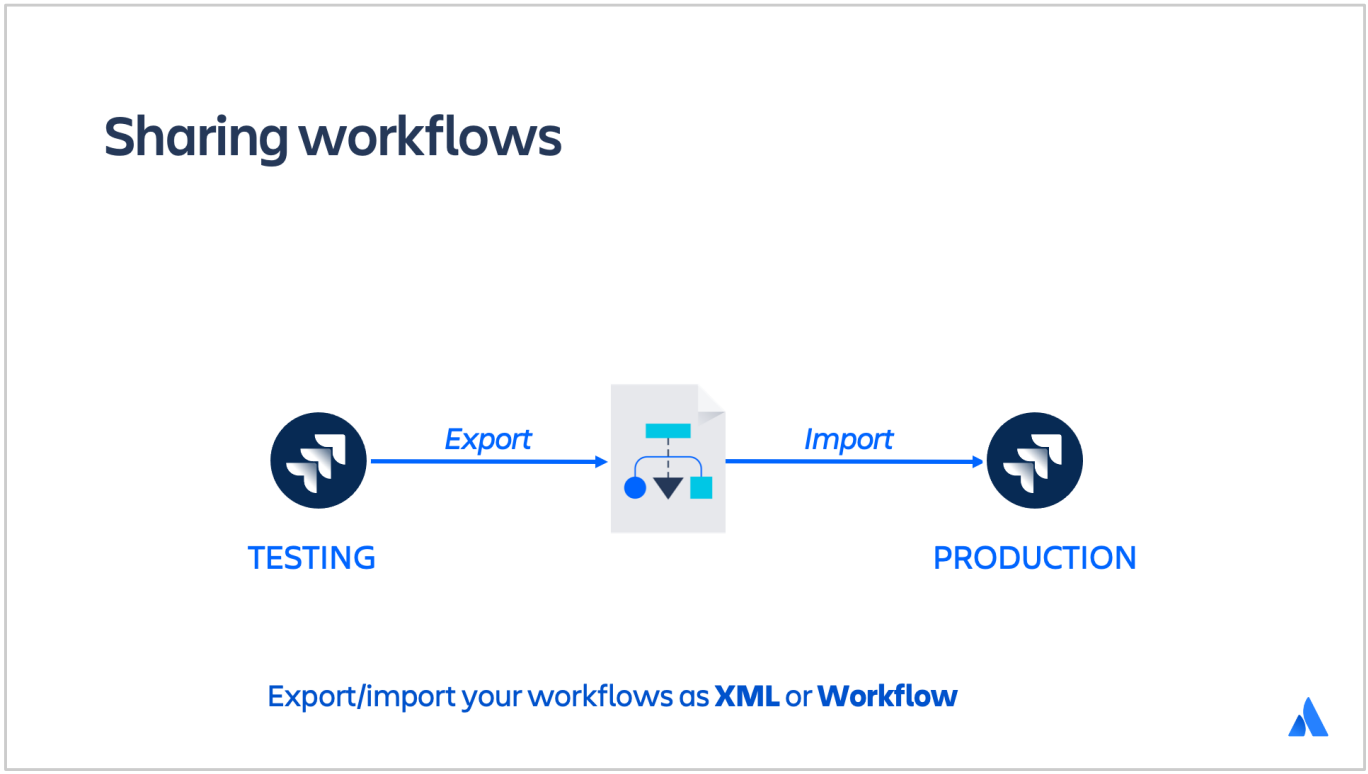
The default Jira workflow comes with two final statuses – Resolved and Closed. This can be a source of confusion. It's up to you whether you use these statuses in your workflows. In a typical use case is a software development workflow. When a developer checks in a change, the issue begins the verification phase of its lifecycle. But just because an issue has a resolution doesn't mean it's done. For software teams, once a developer checks in a change, the resolution on that issue should be set to fixed. The issue then transitions to a code review state for someone else to review. The issue is still open, but it has been "resolved." Once the issue's resolution has been verified, then it can be closed.

You must set a resolution when an issue moves from an unresolved state to a resolved state. So for any transition going to a status that you think of as "closed", make sure that you either use a workflow post function to set a resolution automatically or put the resolution on the workflow transition screen, so the user has to fill it in.

Likewise, the resolution needs to be removed if an issue flows back into an unresolved state. For any transition going into a status that you think of as "open" (e.g., when reopened), use a post-function to clear the resolution.

Jira sees the Resolution field as a flag. If it has a value, then the issue is resolved. If it's empty, then the issue needs some form of action.

Searching for everything that's resolved (using the Resolved status), as well as why it's been resolved (value of Resolution field), provides granular reporting options. Many teams review resolved issues to ensure they are resolved for the right reasons.



Using workflow export and import, you can share your team's workflow with other teams in your organization on different Jira instances. This feature allows you to share and use workflows that other people have published easily, or to move a workflow from testing (or staging) to production in your own organization. Note, Jira administrators can export workflows, but only Jira System administrators can import a workflow from a local instance.

Supplemental information:

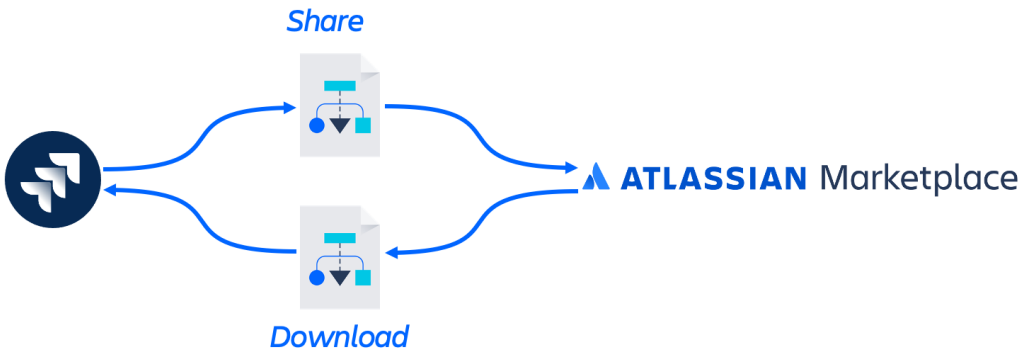
For more information, see Sharing Your Workflow https://confluence.atlassian.com/display/AdminJiraServer085/Sharing+your+workflow and https://confluence.atlassian.com/display/AdminJiraServer085/Using+XML+to+create+a+workflow.

Importing & exporting gotchas

- Workflow app functionality may not be exported/imported
- Enable any disabled custom fields in the imported workflow before importing
- Importing XML workflows is supported for Server only

If you use apps (add-ons) that add functionality in workflows, aspects of the workflow might not be exported/imported. This is especially true when exporting ‘as workflow’. If you export/import as XML the app functionality generally works as long as you have the app installed on the target Jira instance.
If the workflow that you are importing contains custom fields that are disabled, the workflow importer will not create these fields unless they are enabled before importing. You will receive a warning about this. To fix this, you need to enable the missing custom fields before proceeding with the import. For more information on custom fields in workflow imports, Sharing Your Workflow https://confluence.atlassian.com/display/AdminJiraServer085/Sharing+your+workflow.
Importing XML workflows into Jira is supported for Jira Server only (not Jira Cloud).

Sharing/downloading workflows on the Atlassian Marketplace



Using workflow export and import, you can share your team's workflow with external parties in other organizations via the Atlassian Marketplace. You can also import workflows that others have created from the Atlassian Marketplace.
Only Jira administrators can export and import workflows from the Atlassian Marketplace.

See how it's done

- Explore the simplified workflow and how it relates to a project's board
- View conditions, validators, and post functions in one of the default Jira workflow's transitions

Instructor demo.

Are you getting it?

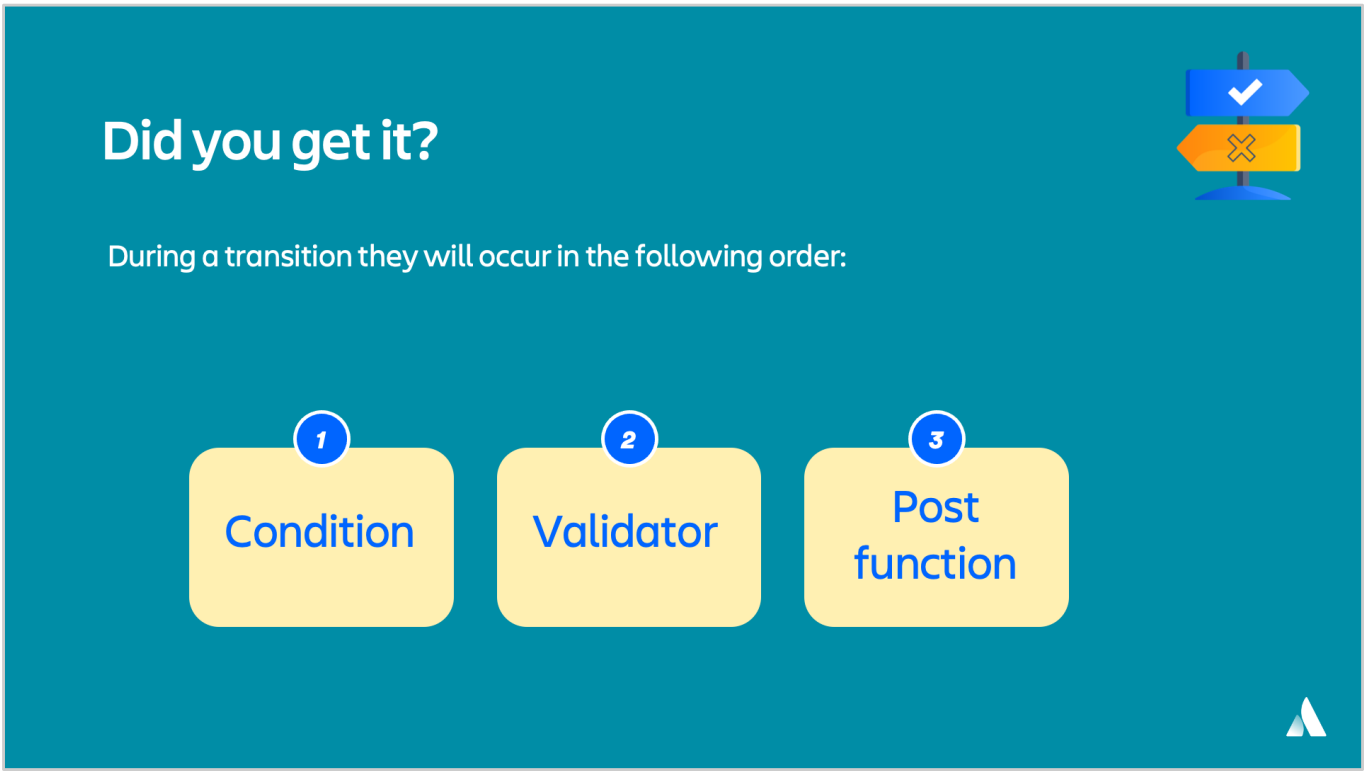
In what order will post functions, conditions and validators occur in a transition?

Validator

Post function

Condition

The answer is on the next slide.



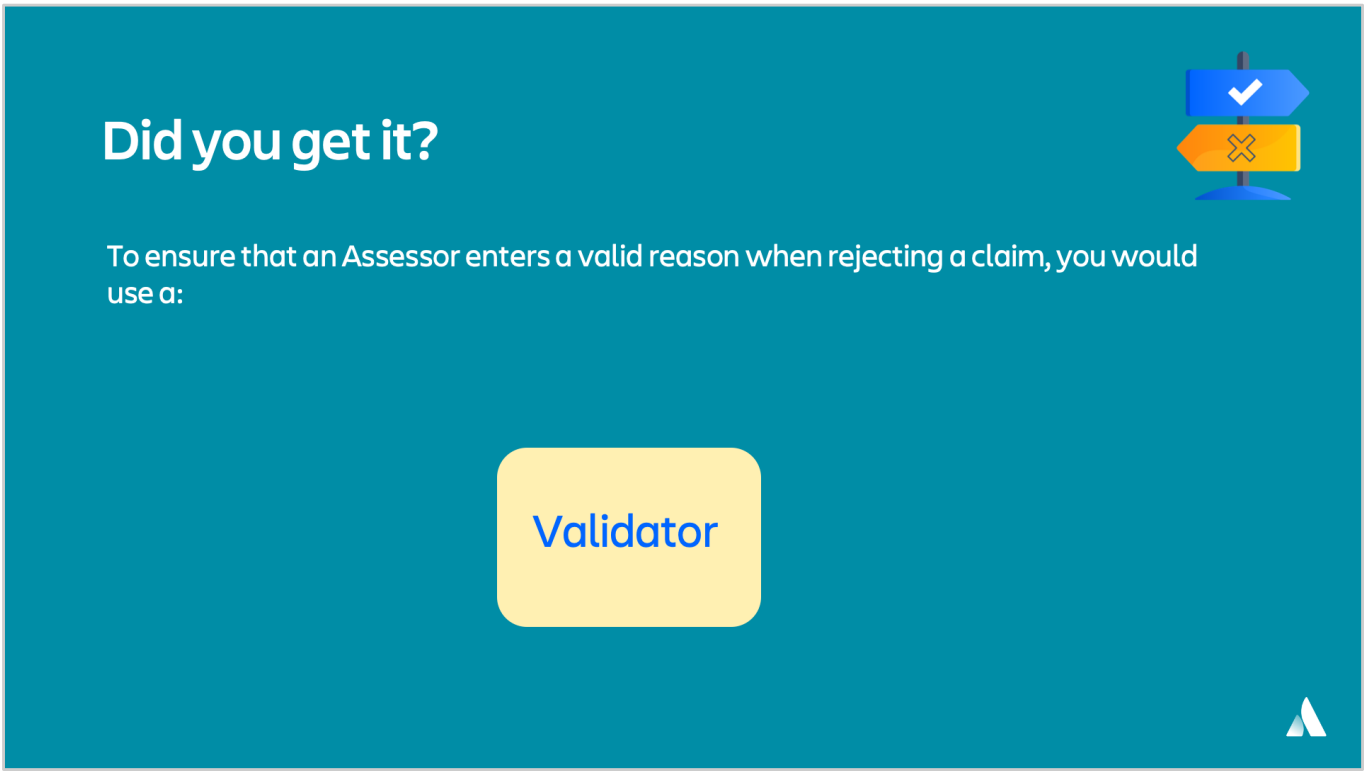
Answer: During a transition conditions must exist before the transition is permitted, input made during the transition must be valid (validator), and automated actions (post functions) occur after the transition.

Are you getting it?

To ensure that an Assessor enters a valid reason when rejecting a claim, you would use a:

- Condition
- Validator
- Post function

The answer is on the next slide.



Answer: To ensure an Assessor enters a valid reason when rejecting a claim, you'd use a Validator. Recall that validators validate input made during the transition.

Are you getting it?

To ensure only users in the Managers project role can submit a pay change request, you would use a:

- Condition
- Validator
- Post function

The answer is on the next slide.

Did you get it?

To ensure only users in the Managers project role can submit a pay change request, you would use a condition.

Condition

Answer: To ensure only users in the Managers project role can submit a pay change, request you'd use a condition in the transition for the pay change request. Recall that a condition must exist before the transition is permitted.

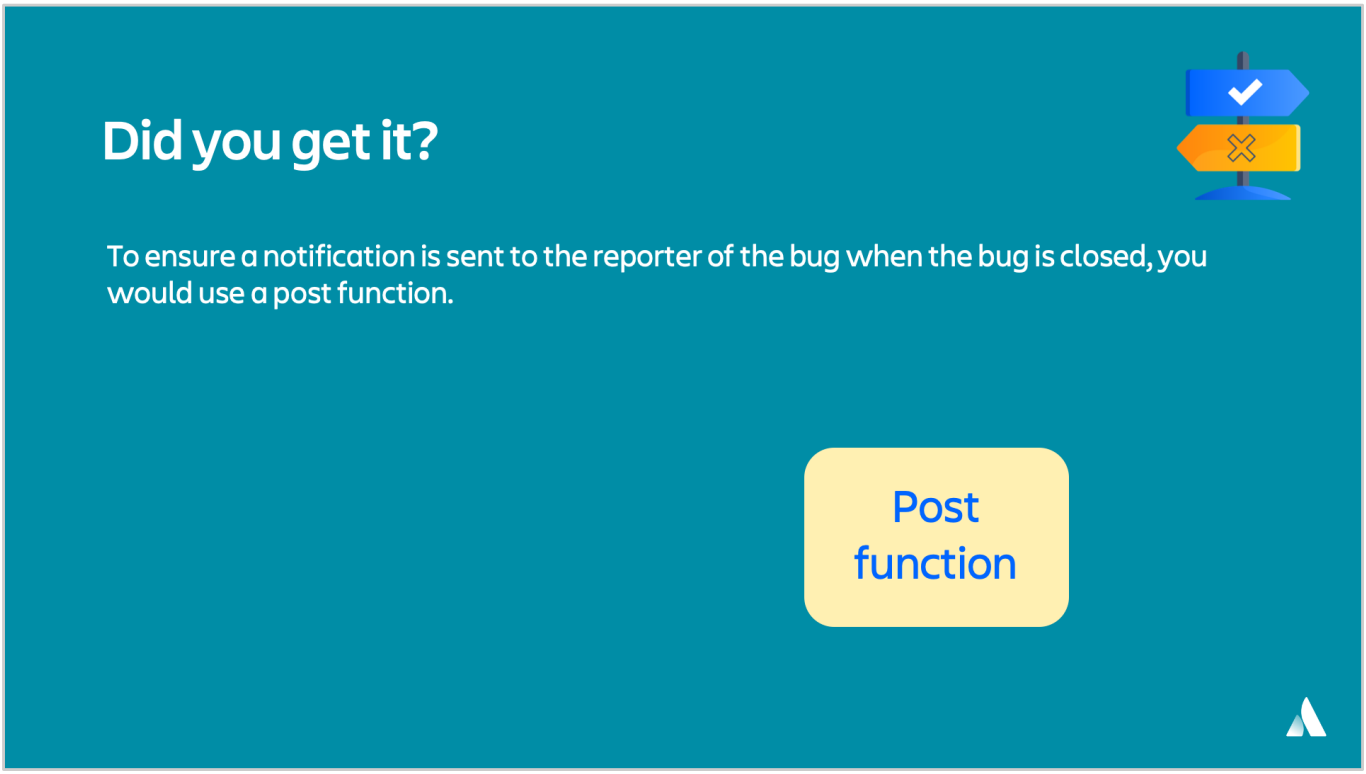
Are you getting it?

To ensure a notification is sent to the reporter of the bug when the bug is closed, you would use a:

Trigger

Post function

The answer is on the next slide.



Answer: To ensure a notification is sent to the reporter of a bug when the bug is closed, you'd use a post function. Recall that post functions are automated actions that occur after the transition has occurred. Whereas triggers automatically transition Jira issues when certain events occur in a connected developer tool such as Bitbucket.

Takeaways

- Copy and customize an existing workflow rather than creating one from scratch
- Keep your workflows simple
- Don't create too many statuses
- Transition and status names should be intuitive

Keep your workflow simple! Too many statuses and thus too many options of available workflow transitions will confuse the user.
Stakeholders often want to have statuses for each part of the workflow. That's generally a good thing, but remember, each status adds more transitions and complexity. Aim for simple and scalable instead. Whenever adding a new status to a workflow, make sure you have no other option.
Don't exceed around seven statuses unless it's complex change management or a shared workflow where different phases will be used by different teams.
Transition and status names should be intuitive. Your users should not be confused about which status to use.

Try it

Lab 4 – Customizing & Sharing Workflows

- Exercise 1 – Customizing a Workflow Transition
- Optional Exercise 2 – Sharing a Workflow

5

Configuring Board & Sprint Permissions

Course Overview

Mapping Your Business into Jira

Configuring Issue Types, Fields & Screens

Customizing Workflows

Configuring Board & Sprint Permissions

Applying New Configurations to Projects

What will you learn?

- List the permissions and access required for sprint and board tasks
- Customize sprint permissions
- Create a new permission scheme and associate it with a project
- List the advantages of using project roles in schemes
- Outline the use case for issue-level security
- Create a new notification scheme and associate it with a project

Who can administer boards?

Jira admin	Project admin + Board admin	Board admin only	Project admin only	Task
✓	✓	✓		Add columns to a board and map statuses
✓	✓			Add new statuses to a board
✓	✓	✓		Switch to Simplified Workflow

You can add columns to the board and map statuses using any type of workflow if you're a Jira administrator or a board administrator. By default, the person who created the board becomes the board administrator.

To add a status to a board, the project must be using the Simplified Workflow. To add a status to a board, you need to be EITHER a Jira administrator OR a project administrator and a board administrator. If you are only the board administrator (and not a Jira administrator or the project administrator) you can configure the board, but you can only add columns to the board but not statuses to the simplified workflow.

Project administrators and board administrators can work with their project's simplified workflow and not get a Jira administrator involved. Jira administrators and board administrators can switch the workflow of a board to Simplified Workflow.

You can only do this if:

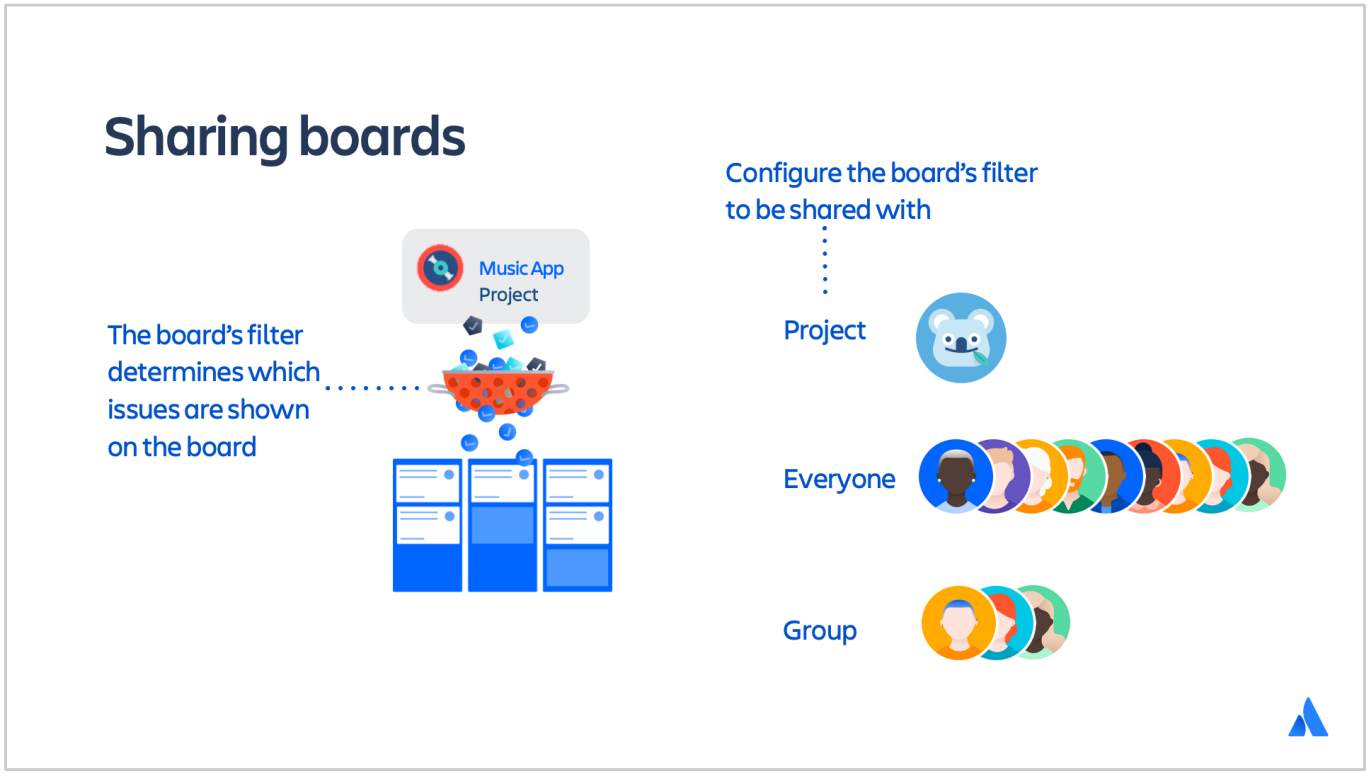
- There is only one project being viewed by your board
- That project uses a Jira workflow scheme, which only has one workflow for all issue types
- Your workflow only uses post functions, validators, and conditions that are provided by Atlassian – and not any of which provided by apps
- The existing workflow has at least one outgoing transition for each status

If you are only a project administrator and not a board administrator, you cannot configure the board.

Creating boards

- For a user to create a board, the user needs:
  - The **Create Shared Objects** global permission
  - Application access for Jira Software
- Exception:
  - Users without Create Shared Objects global permission can still copy a board to create a new board.

Only users who have the Create Shared Objects global permission and who belong to a group assigned to the Jira Software application can create boards. By default, application access for Jira Software is provided through the default group jira-software-users. All logged-in users for all applications have Create Shared Objects global permission by default.
One exception to this is that users without Create Shared Objects global permission can still copy a board to create a new board. However, they will not be able to change the share settings for the underlying filter.
There are no boards available for Business projects in Jira Server.



Each board has a filter that determines what issues are shown on the board. By default, the filter for a project's board is shared with everyone who has permission to browse the project.

If you wish to share a board with different people, you will need to either edit the saved filter or choose a different filter.

You can edit the board's filter sharing by configuring the board then clicking **Edit Filter Shares** and specifying who to share it with. You can share a board's filter with:

- Everyone – be careful with this as this will make this pubic and visible to users who are not logged in (see more conditions to see a board coming up).
- Group – select a group to share it with, but you can only share with groups you are a member of.
- Project – you can share it with everyone who has permission to browse a particular project or has a particular project role in a particular project.

Permission for sharing boards

'Create Shared Objects' Global Permission	
Required when you	**Not required when you**
Create a board via the Boards page	Create a project - board is created by default
	Copy a board

Depending on how the board was created you may or may not need the ‘Create Shared Objects’ global permission to edit the filter shares.
If you created a board via the Boards page (by selecting Boards in header > View All Boards then Create board), you will not be able to share it, unless you have the 'Create Shared Objects' global permission. All logged in users for all applications have Create Shared Objects permission by default.
If you created the board via the methods below, you do not need the 'Create Shared Objects' global permission to share the board:

- Creating a project (where a board is created for the project by default). You need the Jira Administrators global permission to create a project. This also applies if you create a project when setting up Jira Software for the first time (where you're prompted to create a project, which also creates a board for the project).
- Copying a board (the copied board will be shared with the same users as the original board)

Since only administrators can create projects, this permission is needed for all non-administrator users who create a board by any means other than copying it.
For information on sharing filters, see https://confluence.atlassian.com/display/JiraSoftwareServer/Saving+your+search+as+a+filter#Savingyoursearchasafilter-sharing_filtersSharingafilter.

Board permissions

	Browse Projects permission	Application access	In board's filter share
Create a board	✓	✓	
See a board	✓	✓	✓

By default, all logged in Software users have this permission

The default board for a project is created when the project is created. Only Jira administrators can create projects.

For a non-administrator user to create a board in a project they need:

- *The 'Browse Projects' permission for the project they want to create the board in. By default, all logged in users with Application Access have this permission.*
- Have the correct application access i.e. belonging to a group that has Jira Software application access, for example, the default jira-software-users group. Application access for Jira Software is required to see Software boards. Users with Software application access are the only ones that can access Software only functionality such as boards.

A board is available to all users who:

- *Have 'Browse Projects' permission for the project(s) whose issues are shown on the board. The 'Browse Projects' permission is a project permission.*
- Have Jira Software application access.
- *Can view the saved filter on which the board is based. By default, the filter for a project's board is shared with everyone who has permission to browse the project. We'll look at this more on the next slide.*

Who can see boards?

For a user to see a board, they need:

- The **Browse Projects** permission
- The board's filter to be shared with them
- Application access

A board is available to all users who:

- Have 'Browse Projects' permission for the project(s) whose issues are shown on the board. The 'Browse Projects' permission is a project permission in the permission scheme. By default, all logged-in users with application access have the Browse Projects permission.
- Can view the saved filter on which the board is based. By default, the filter for a project's board is shared with everyone who has permission to browse the project.
- Have the correct application access i.e., belonging to a group that has Jira Software application access, for example, the default jira-software-users group. Application access for Jira Software is required to see Software boards. Users with Software application access are the only ones that can access Software only functionality such as boards. For example, if a user only had application access for Jira Core or Jira Service Management and not Jira Software, they would not be able to see Scrum or Kanban boards. As long as a user has any application access, they can see Business boards.

Who can move issues on boards?

For a user to move issues on a board, they need:

- **Transition Issues** permission
- **Resolve Issues** and **Close Issues** permissions for non-simplified workflows
  - For example, only developers resolve and only QA members close issues
- Not to be excluded by a workflow transition condition

For a user to move issues on a board, they need:

- The 'Transition Issues' permission whether the board is using the simplified or a non-simplified workflow.
- Both the 'Resolve Issues' and the 'Close Issues' permissions for a non-simplified workflow. Note, the 'Resolve Issues' permission applies to both resolving and reopening an issue.
- Not to be excluded by a workflow transition condition. Recall from the last module on customizing workflow that you can create conditions on transitions that specify something that must exist before the transition is permitted. For example, only specific people e.g., Reporter, or only users with a specific permission or in a particular group (Development), etc. can execute a transition.

Are you getting it?

For a non-administrative user to see all the issues on the board they need:

a. **Browse Projects** permission in the project
b. **Transition Issues** permission in the project
c. Jira Software application access
d. To be in the board's filter shares

The answer is on the next slide.

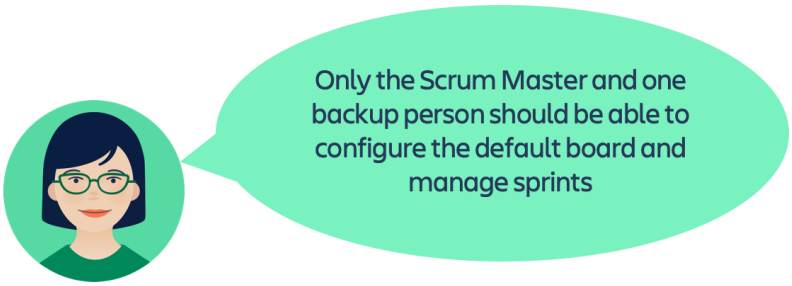
Did you get it?

For a non-administrative user to see all the issues on the board they need:

- ✓ a. **Browse Projects** permission in the project
- b. **Transition Issues** permission in the project
- ✓ c. Jira Software application access
- ✓ d. To be in the board's filter shares

Answer: a, c, and d.
To see issues on a board, a non-administrative user needs the Browse Projects permission in the project, Jira Software application access, and to be in the board's filter share.
The Transition Issues permission in the project is not required to see issues on the board but it is required to move issues on the board. By default logged in Jira Software users have these permissions.

Teams In Space Business Requirements



What should be done in Jira to meet these requirements?

By default, board and sprint permissions are fairly open. Here's a use case for restricting these permissions. Teams In Space wants to restrict board and sprint permissions to just the Scrum Master and one other designated backup person from the team. The backup person for the Scrum Master should have very good knowledge of Jira.

Teams In Space Jira Implementation

- Add the Scrum Master and one backup to the default board's **Administrators**
- Create **DEV Permission Scheme**
- Restrict the **Manage Sprints** permission to just the Scrum Master and one backup by using the **Administrators** project role
- Associate the DEV Permission Scheme with the standard DEV project

At Teams In Space the backup person should be able to perform all tasks in the project, not just board and sprint related tasks so they are both being assigned to the Administrators project role. By putting them in a project role, we can use this project role in the permission scheme which will later be shared with other development projects going forward.
Back in Lab 2, you added the Scrum Master and one backup to the default board's administrators.

Sprint permissions for software projects

The **Manage Sprints** project permission is required to:

- Create, start, complete, and reopen sprints
- Reorder and delete future sprints
- Edit sprint information and move the sprint footer

By default, all logged in Software users have this permission

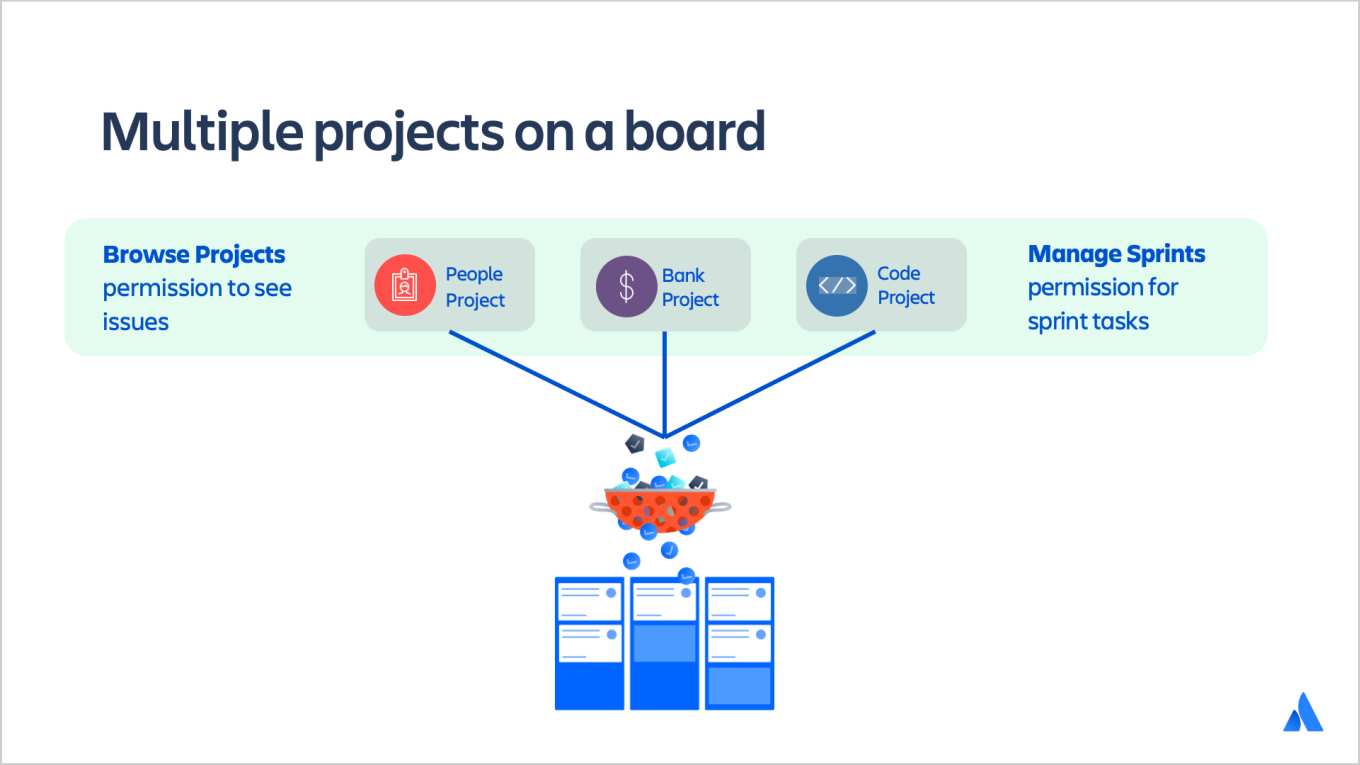
Sprints are not available with Business (Core) projects, so sprint permissions to do not apply to Business projects.
The Manage Sprints project permission is used in versions of Jira after version 7.1 to perform these sprint-related actions that are to create, start, complete, and reopen sprints. It is also required to reorder and delete future sprints, edit sprint information (name and dates) and move the sprint footer. If you create a new Jira instance, the manage sprint permissions, by default, will be enabled for all Software projects, and every user will have the ability to complete each sprint responsibility.
For more information, see https://confluence.atlassian.com/display/AdminJiraServer/Managing+project+permissions
Note that if you upgrade from a previous version of Jira that does not have the Manage Sprints permission (prior to version 7.1) the Manage Sprints permission will be set to the same setting that is in effect for Administer projects. So, for example, if only the Administrators project role can administer projects they will be the only ones who can Manage Sprints.

Sprint tasks requiring project permissions

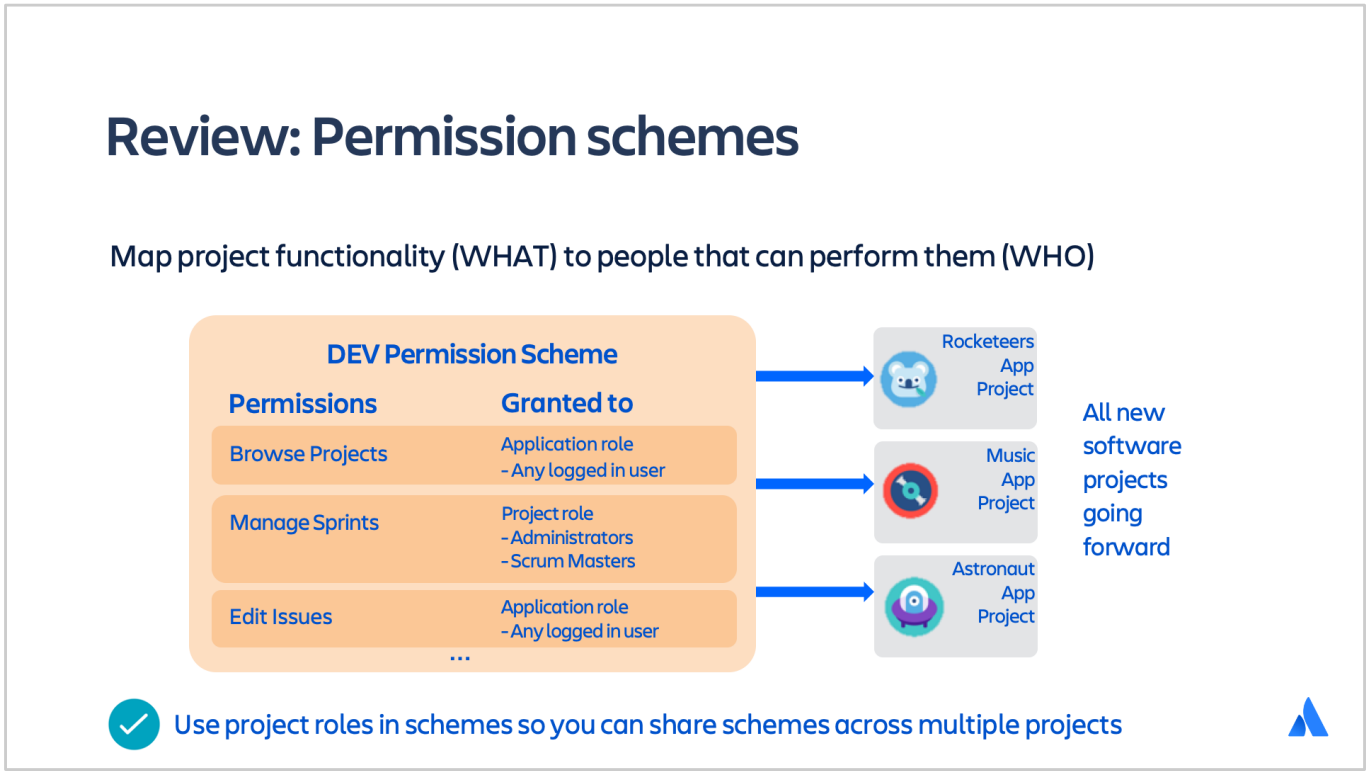
Manage Sprints	Schedule Issues	Edit Issues	Task
✓	✓	✓	Move sprint footer
	✓	✓	Move issue (reorder/rank)
	✓	✓	Add/remove issue to/from sprint

By default, all logged in Software users have these permissions

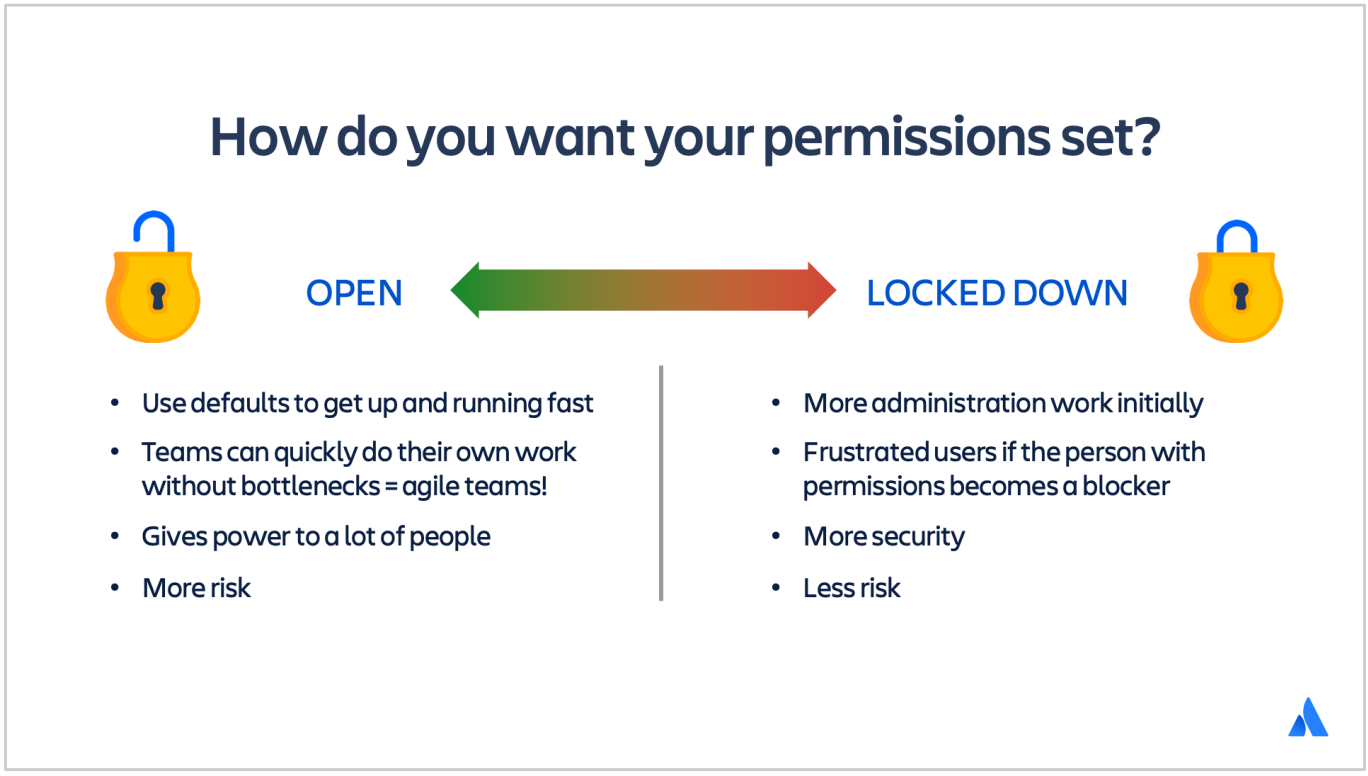
The Schedule Issues and Edit Issues permission is required, in addition to the Manage Sprints permission, to move the sprint footer.
The Schedule Issues and Edit Issues permission is required to move, reorder, and rank issues as well as to add and remove issues to/from the sprint. However, the Schedule Issues and Edit Issues permissions are not required to reorder/rank issues if you only move issues across the sprint footer without changing the order of the issues.
For permissions required for Epics and Versions, see https://confluence.atlassian.com/display/JiraSoftwareServer/Permissions+overview.



For a user to see all the issues from multiple projects on a board, the user will need to have the 'Browse Projects' project permission for all projects on the board. The default application groups have the Browse Projects permission by default.
Also, to perform sprint related tasks, the 'Manage Sprints' permission is required for all projects on a board. The sprint related tasks are – create, start, or complete a sprint, reorder or delete future sprints, and edit sprint information. You still need the Manage Sprints permission for all projects on the board even if the sprint (that is to be started) doesn't include issues from all projects queried by the board.
With the Manage Sprints permission, the board's filter query determines the projects that users need to have permission on.
Also, permissions are now checked against the filter query of the board from which the sprint originates, not just against the issues within the sprint. For information about boards with complex filter queries and this permission, see https://confluence.atlassian.com/display/AdminJiraServer/Using+Manage+Sprints+permission+for+advanced+cases.



Permissions are defined in Permission Schemes that are then associated with projects by the Jira administrator. The project's permission scheme determines who has permission to perform various tasks in this project. Permission schemes allow you to create a set of permissions and apply this set of permissions to any project. All permissions within a scheme will apply to all projects that are associated with that scheme.
In this example, the permission requirements in all development projects are the same. Any logged in user with application access can browse projects and edit issues but a user requires the Administrators project role to Manage Sprints.
Project permissions can be granted to individual users, groups, project roles, and issue roles such as 'Reporter', 'Project Lead', and 'Current Assignee', and 'Anyone' (e.g. to allow anonymous access).
Note that some permissions are dependent upon others to ensure that users can perform the actions needed. For example, in order for a user to be able to resolve an issue, that user must be granted both the 'Transition Issue' permission and the 'Resolve Issue' permission.



Jira is a tool for helping people do work. Sometimes you need more security to protect restricted data, but otherwise, you should err on the side of giving people access, as long as this meets your business requirements. Every time the tool is a bottleneck and slows or prevents work, it's a problem. We have to think through permissions in advance, including edge cases, so we configure Jira to help workflow instead of causing it to fail.
One good example of this is restricting the manage sprints permission. If we restrict it to one person, and that person is out sick or on holiday, then work will halt.
When the whole team has the Manage Sprints permission, this also helps the whole team take more ownership of the sprint that they are working on day in and day out, while giving benefits to project and global admins, too. This fine-grained control removes a lot of internal support requests. Restricting this permission minimizes risk, but escalating problems relating to sprint responsibilities can be a huge blocker for teams. If you create a new instance, then the Manage Sprints permission, by default, will be enabled for all projects, and every user will have the ability to complete each sprint responsibility listed above. This is great for new teams and/or new team members that don't need to lock anything down.
Using the defaults, let's your teams get up and running fast. If you lock permissions down, it will be more administrative work initially to set up, but if you use project roles in the permission scheme, it should ease work going forward.
See http://blogs.atlassian.com/2016/02/jira-software-sprint-permissions/.

Are you getting it?

Which of the following is required for a user to see a board?

a. 'Create Shared Objects' global permission

b. 'Browse Projects' project permission

c. Jira Software application access

d. Be a board administrator

e. Be in the board filter shares

The answer is on the next slide.

Did you get it?

For a user to see a board, they must have:

a. 'Create Shared Objects' global permission

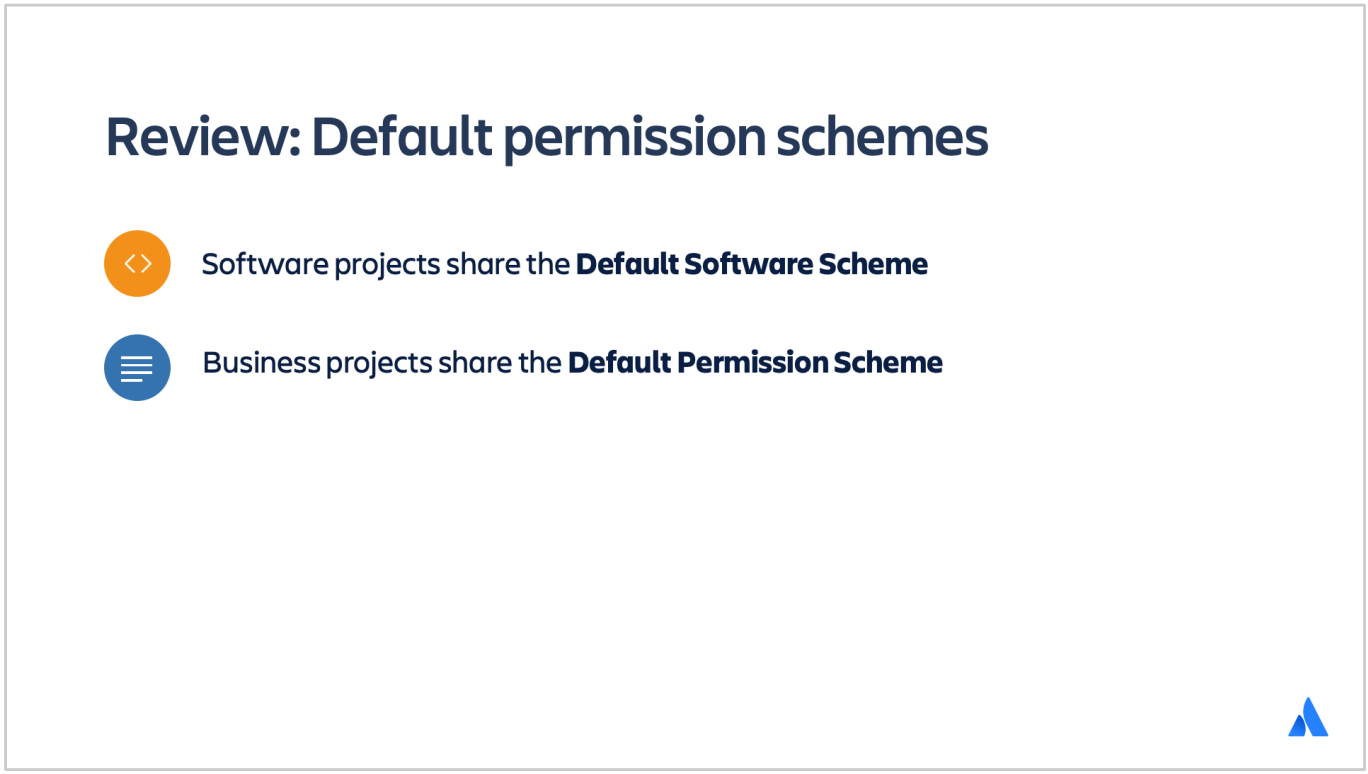
✓ b. 'Browse Projects' project permission

✓ c. Jira Software application access

d. Be a board administrator

✓ e. Be in the board filter shares

Answer: For a user to see a board, they must have the ‘Browse Projects’ project permission for the project the board is in, Jira Software application access, and be in the board’s filter shares.



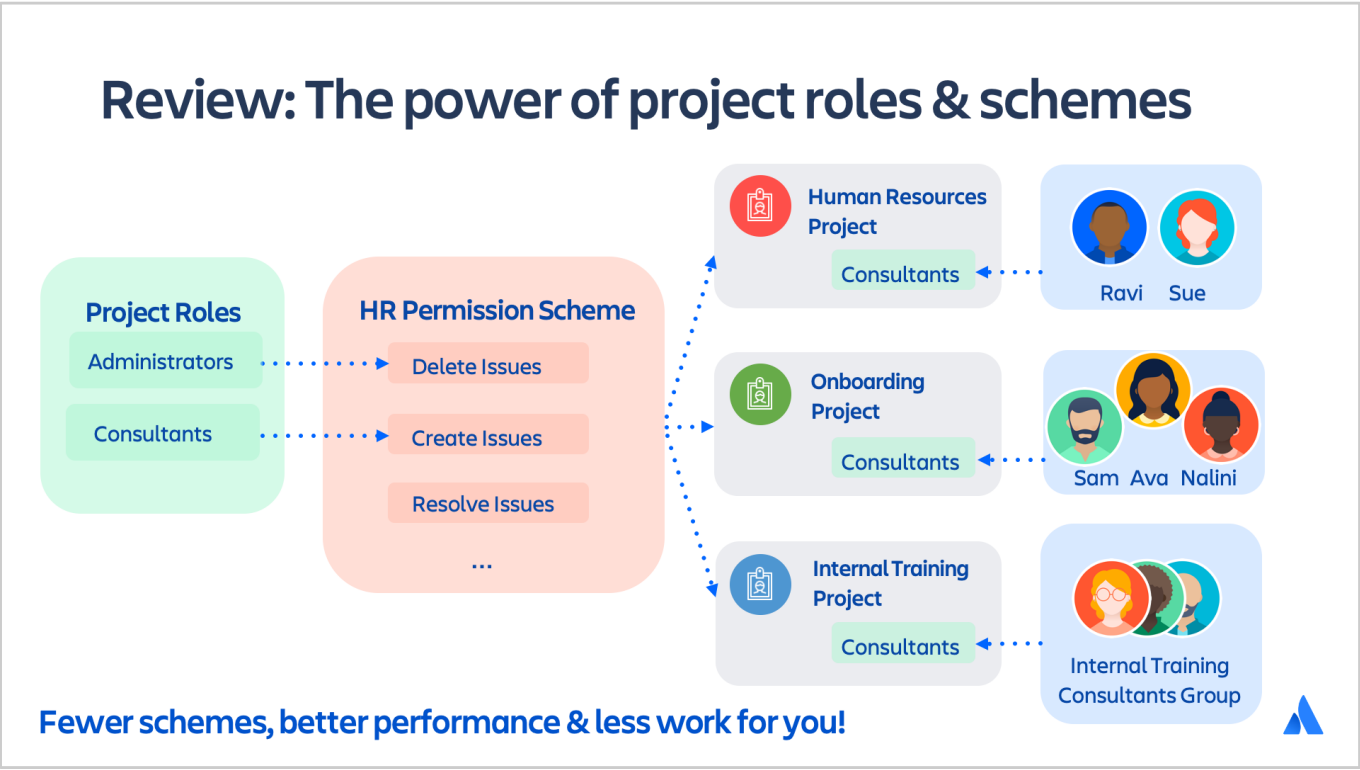
When you install Jira it automatically creates the **Default permission scheme**. This is used for, and shared by, any new Core (Business) projects by default.
If you have Jira Software installed and create a Software project, Jira will create the **Default software scheme** which will be used for, and shared by, any new Software projects by default.

Review: Project roles vs. Groups

- Membership is for one project
- Only Jira administrator can create
- Project administrators can alter membership

- Membership is global
- Only Jira administrators can create
- Only Jira administrators can alter membership

Project roles are a flexible way to associate users and/or groups with particular projects. A project Role is kind of a bucket that holds individual users or groups. The members of project roles are users/groups who fulfill particular functions for a project.
Project roles can be used in many places, including permission and notification schemes, issue security levels, workflow conditions, and comment visibility. They can also be given access to issue filters and dashboards. But the core usage is for permission and notification schemes. While you could assign permissions and notifications to users and groups directly, roles are more flexible and sustainable.
Project roles are similar to groups, the main difference being that group membership is global, whereas project role membership is project-specific. Additionally, group membership can only be altered by Jira administrators, whereas project role membership can be altered by project administrators. Using groups in schemes mean a lot more work for the Jira administrator.
Supplemental information:
See
https://confluence.atlassian.com/display/AdminJiraServer085/Managing+project+roles.



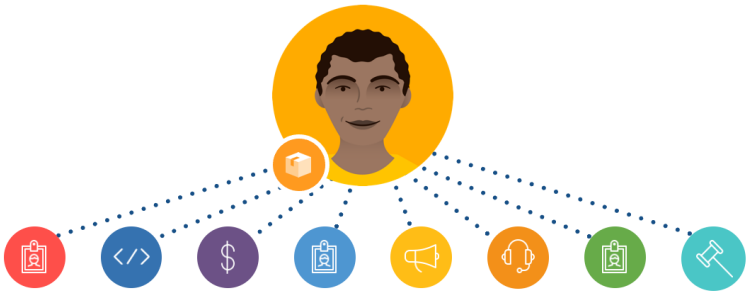
Here we see the power of roles and schemes. You use roles in schemes. Project roles are global – they can be assigned to any scheme. Then you share the schemes among projects.

In this example, the Consultants project role is granted the 'Create Issues' permission in the HR Permission Scheme. Then that permission scheme is used in multiple HR projects – Human Resources, Onboarding, and Internal Training. In each of those projects the project administrator has assigned different users or groups to the role. So the two consulting users, Ravi and Sue, can create issues in the Human Resources project, the three consulting users, Sam, Ava, and Nalini, can create issues in the Onboarding project and the Internal Training Consultants group can create issues in the Internal Training project. (You can also have users *and* groups in a project role.)

By using project roles you can use fewer permission schemes. Fewer schemes means better performance. It also means you can share schemes and save a lot of administration work. It takes requests that you would normally handle and puts them in the hands of project administrators (assigning the particular users/groups to the role). That means you can focus on the tasks that can't be delegated to project administrators. So using project roles in schemes rather than users or groups means individual projects can use the schemes and assign their own members to roles without the Jira Admin's involvement. That makes it easy to share a few schemes (which is good for performance) and lessens Jira administrator's load.

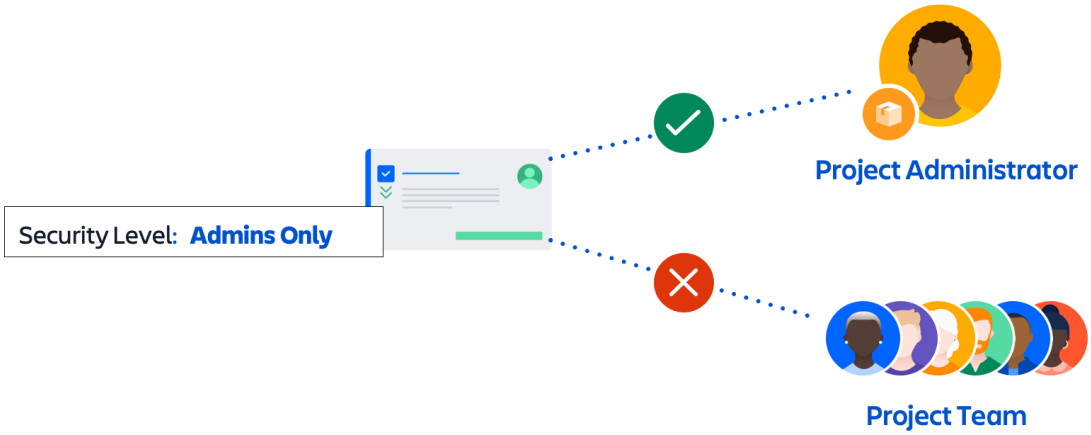
Setting default members for a project role

- Default members for a project role are assigned to that project role in all newly created projects
- For example, Malik is a non-administrator user who needs project administrators role in every project

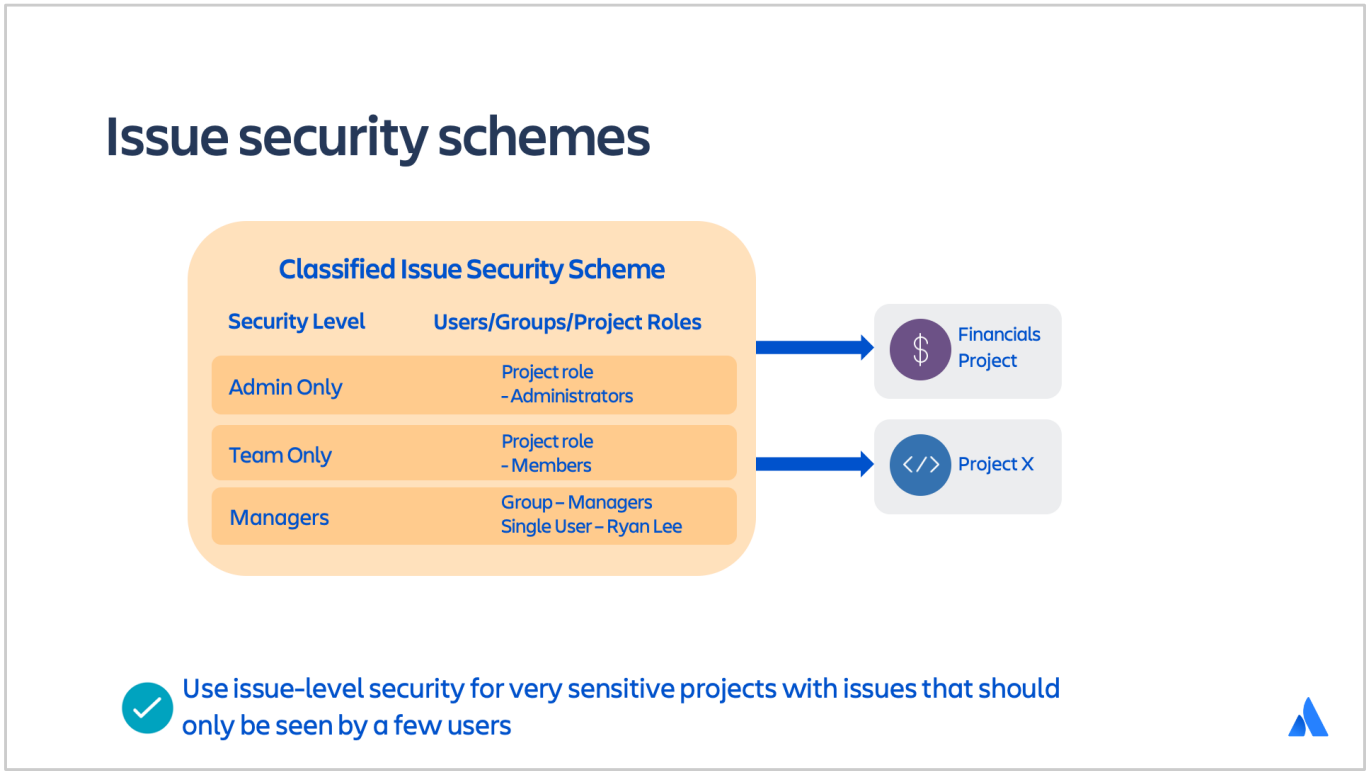


You can specify default members for a project role. If you do, these default members will be assigned to the project role for all newly created projects. The membership for any particular project can then be modified by the project administrator.
You (the Jira Administrator) can grant users and groups default membership of roles.
For example, Malik is a non-Jira administrator user, but he has been tasked with being the Jira project expert who can troubleshoot and help other project administrators. To do so, he needs Administrators role in every project. To achieve this, he is granted default membership of the project's Administrators role.

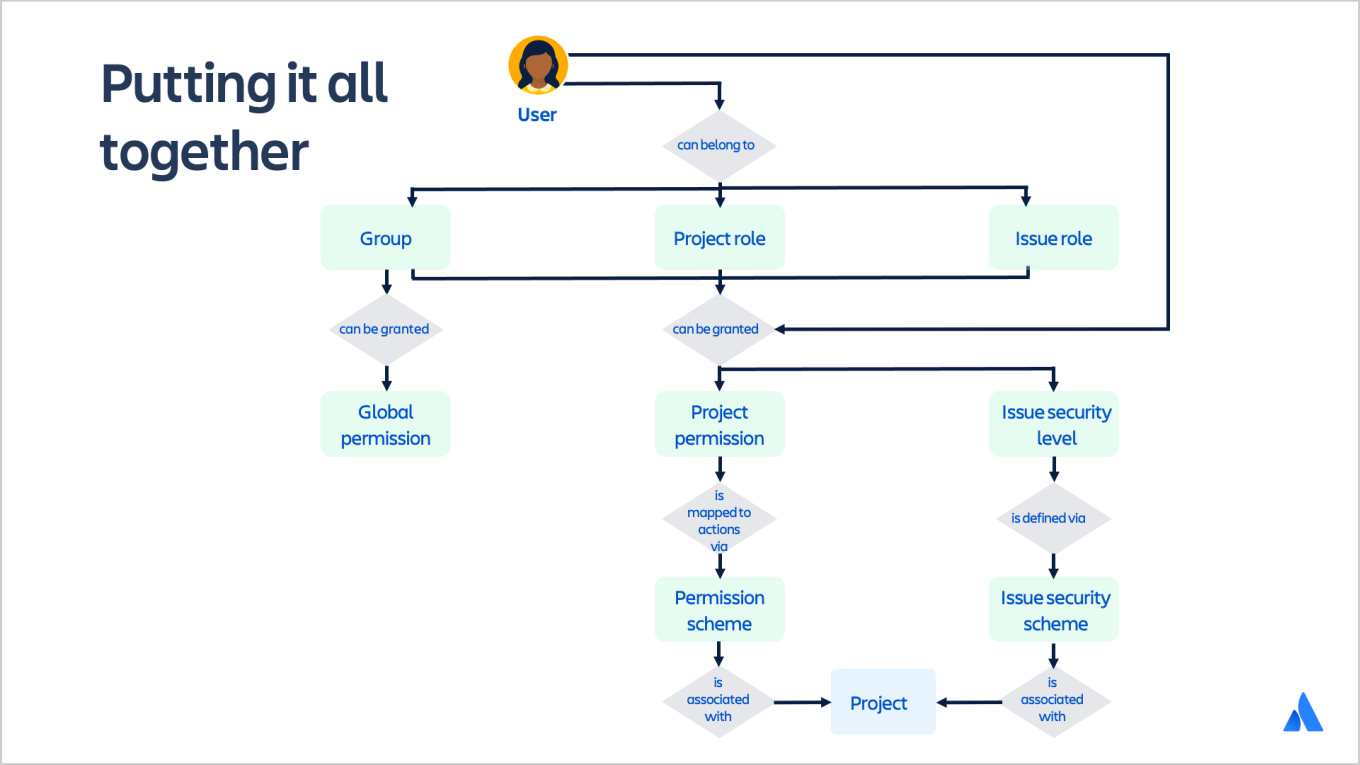
Introduction to issue-level security



As a Jira administrator, you can control who sees issues by using issue-level security.
When creating or editing an issue, a user can select a Security Level (this is a field in the default field configuration). This ensures only users who are assigned to this security level can view the issue.
You can create security levels, and specify the users, groups, and/or project roles that can see issues for each security level. For example, you could create an Admins Only security level that is restricted to project administrators. If the Security Level field for an issue is set to Admins Only, then only project administrators can see that issue; other project team members cannot.
Supplemental information:
For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Configuring+issue-level+security.



You set up the security levels and who can view each level in issue security schemes. Each scheme consists of a number of security levels that have users/groups/project roles assigned to them. The scheme maps security levels to the people (users/groups/project roles) that can access issues of those levels.
Then you can associate different issue security schemes to different projects. Issue security levels allow the visibility of individual issues to be adjusted within the bounds of the project's permissions. By default, there are no issue security schemes created on a Jira instance. This example shows the Classified Issue Security Scheme where three security levels are set. Only project administrators can see issues set as Admin Only. Only users in the Members project role can see issues set as Team Only and only members of the Managers group or the user Ryan Lee can see issues set as Managers. This scheme is applied to two very sensitive projects – Financials (which contains sensitive company financial projection figures) and Project X (top-secret new app under development).
You need to have the Jira Administrators global permission to configure issue-level security.



This graphic puts it all together – global permissions, project permissions, and issue level security, along with their schemes.
Examples of an issue role is ‘Assignee’ or ‘Reporter’.
An example of a project role is 'Administrators'
An example of a global permission is ‘Share dashboards and filters’.
An example of a project permission is ‘Create Issues’.
An example of an Issue Security Level is ‘Restricted’.

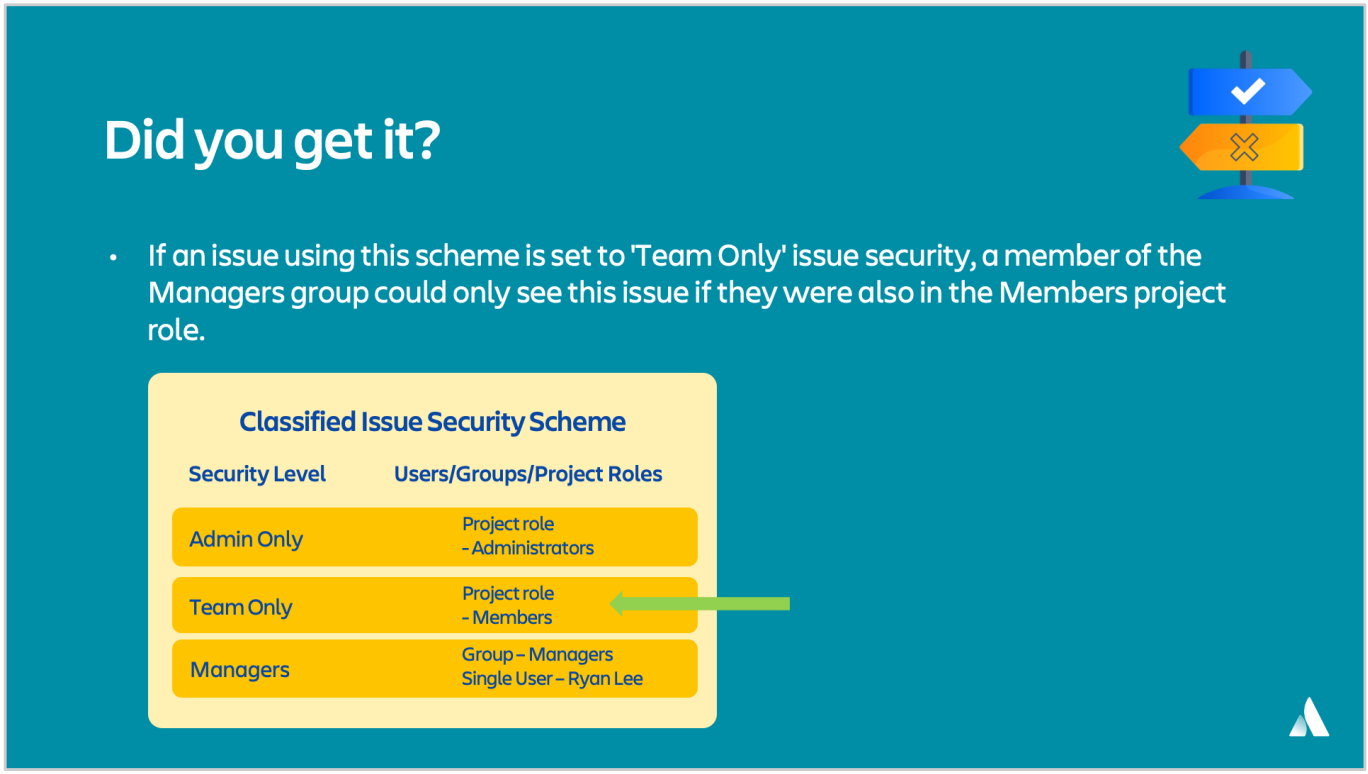
Are you getting it?

- If an issue using this scheme is set to 'Team Only' issue security, can a member of the Managers group see the issue?

Classified Issue Security Scheme

Security Level	Users/Groups/Project Roles
Admin Only	Project role - Administrators
Team Only	Project role - Members
Managers	Group – Managers Single User – Ryan Lee

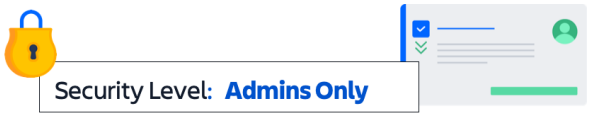
See the answer on the next slide.



If an issue using this scheme is set to 'Team Only' issue security, a member of the Managers group could only see this issue if they were also in the Members project role. If they were a member of the Managers group but not in the Members project role, they could not see the issue.

Who can set issue security?

Only users with the **Set Issue Security** permission can set the security level on an issue to control who can access the issue



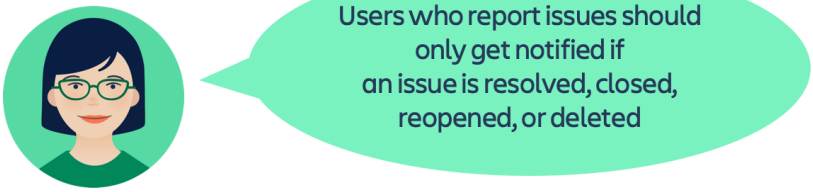
'Set Issue Security' is a permission in all the default permission schemes. You can grant this permission to users so they can set the level of security on an issue so that only people in that security level can see the issue. This is only relevant, of course, if issue security has been enabled.

Review: Notifications best practices

- Don’t send too many notifications or people will start ignoring them
- Explore the use of custom events on workflow transitions to really target notifications at critical times

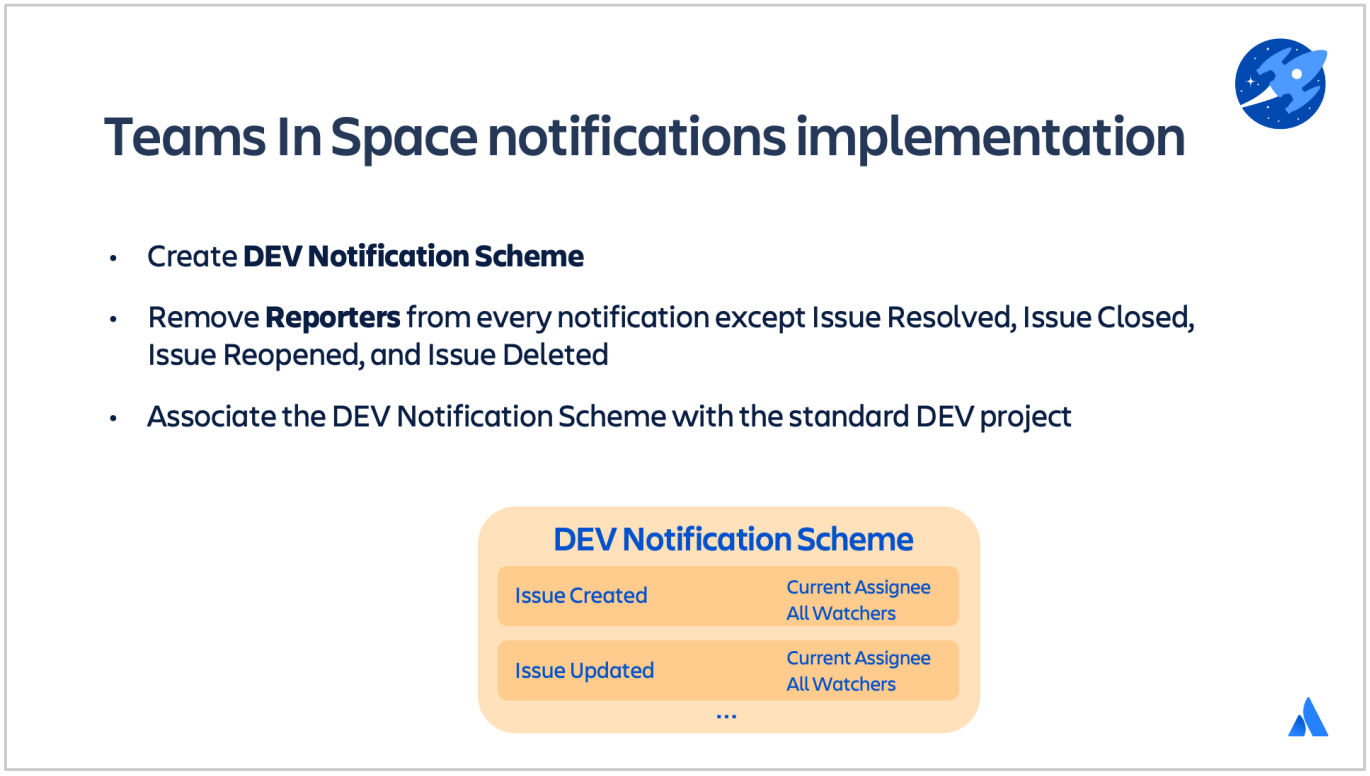
Notifications let people know that work needs to be done or that something significant has changed about which they need to be aware.
You don't want people to have to go to their email all the time for notifications. Instead, use notifications for important updates and typically have folks getting updates via their dashboards/filters.
Jira can notify the appropriate people when a particular event occurs in your project (e.g., "Issue Created," "Issue Resolved"). You can choose specific people, or groups, or roles to receive email notifications when different events occur. As a best practice, roles are the easiest way to manage notifications.
Notification Scheme – the project's notification scheme determines who receives email notifications of changes to issues in this project.
Email – specifies the 'From' address for emails sent from this project. Only available if an SMTP email server has been configured in Jira.
Please note, the default notification scheme for each project type is associated with all new projects of the corresponding type by default. This means that if you have an outgoing (SMTP) mail server set up, that email notifications will be sent as soon as there is any activity (e.g., issues created) in the new project.

Teams In Space notifications business requirements



What should be done in Jira to meet these requirements?

Teams In Space's requirements for notifications are that users who report issues should only get notified at a few points in an issue's lifecycle; when the issue is resolved, closed, re-opened, or deleted.



Developers have requested they get fewer notifications when issues are assigned and updated. To accomplish this the Jira administrator has:

1. Created a new notification scheme, DEV Notification Scheme, by copying the Default notification scheme.
2. Updated the new DEV Notification scheme, removing Reporters from every notification except Issue Resolved, Issue Closed, Issue Reopened, and Issue Deleted.
3. Associated the new scheme with the standard DEV project.

Are you getting it?

How does it help your time management as a Jira administrator to use custom project roles?

The answer is on the next slide.

Did you get it?

How does it affect your time management as a Jira administrator to use custom project roles?

You can create one custom role, use it in many schemes, then share those schemes among many projects. Then it's up to the project administrator to assign users/groups to that role; your job is done in record time! Without project roles, you would have to create separate schemes and groups for each project. And, for each project, you would need to assign those groups (and/or users) to their schemes. Also you would need to update the schemes and groups every time someone left or joined the company. It would be a time sink!

Takeaways

- Set permissions to match how your teams work and also to meet the organization's business requirements
- Use roles in schemes, not users or groups
- Use issue-level security for very sensitive projects with issues that should only be seen by a few users
- Use notifications to alert users about important information

You want to give your teams as much permission as possible to let them get their jobs done. But you need to balance that with the organization's need for security and their risk tolerance.
Use roles in schemes so you can share schemes and have different users in each role for different projects.
Don't send too many notifications or people will just ignore them.

See how it's done

Explore sprint and board permissions

Instructor demo.

Try it

Lab 5 – Configuring Board & Sprint Permissions

- Exercise 1 – Configuring Sprint Permissions
- Exercise 2 – Troubleshooting Board & Sprint Permission Problems
- Optional Exercise 3: Configuring Notifications

6 Applying New Configurations to Projects

Course Overview

Mapping Your Business into Jira

Configuring Issue Types, Fields & Screens

Customizing Workflows

Configuring Board & Sprint Permissions

Applying New Configurations to Projects

What will you learn?

- List problems that can occur as a Jira instance grows
- Apply a standard set of configurations to new and existing projects
- Migrate issues to a new issue type
- Migrate issues to a new workflow
- List best practices for changing schemes in an existing project
- Reduce the number of unused schemes

You saw this back in Module Two, and we've used this case study throughout the course. Now that we've created a set of standard schemes, we're ready to apply them to new and existing projects. But before we do that, let's revisit why they need to make these changes.

Teams In Space and their Jira instance has grown fast, and now they're having problems. Up until now, each Development team has managed themselves - everyone has had the Jira Administrators permission and so have created their own projects, which has resulted in too many schemes, workflows, statuses, etc. There is also no project role management. It's a bit of a mess. You are the newly appointed Jira Admin and have to clean things up and enforce some control to manage the instance as you scale. Teams In Space wants more control over projects and what's created in Jira. And only you (and your team) will have the Jira Administrators permission. See the next slides for the consequences of this scenario.

Too many schemes

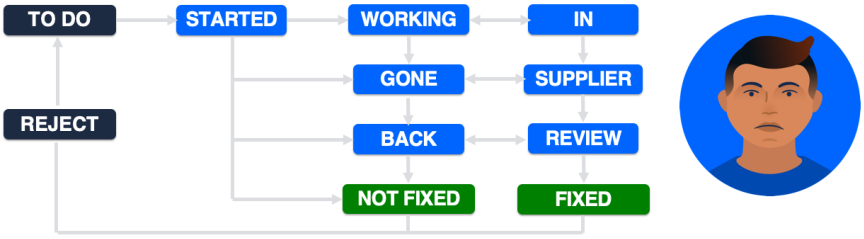
- Projects created the default way have their own set of schemes
- Large enterprises could end up with a huge number of in-use schemes
- Too many schemes can affect performance and it's a huge administration overhead

When you create a project the default way, that is, without selecting shared configuration, that project ends up with its own set of schemes. In a large enterprise, this could mean a huge number of in-use schemes.
This can adversely affect performance and requires a lot of maintenance. This means too much administrative overhead and work for you, the Jira administrator.

Hard to use workflows

Users who are board and project administrators may create many statuses

- No standard process for similar projects
- Confusing status names
- Complex workflows

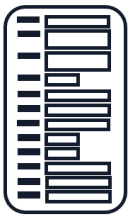


Users who are both board administrators and project administrators can create new columns/statuses for their boards. Creating a new status updates the current workflow. By default, that's the simplified workflow used for the project. If many projects allow this, then you will end up with many different project-specific workflows, possibly for similar types of projects. That means there's no standard process for similar projects. This may work in a small organization, but as your organization and your Jira instance grow, it's best to implement standard business processes across similar projects by using a shared standard workflow that users cannot edit.
Also, if users are creating statuses, they may create statuses with confusing names and so other users may have difficulty figuring out what the statuses mean and will get frustrated. Or they may use status names that should be resolutions, for example, FIXED.
A further impact can be that workflows become overly complex.

Hard to use screens

Users with the Jira Administrators global permission may create screens with:

- Too many fields
- Confusing field names



If users have the Jira Administrators global permissions they may create screens with too many fields. They may also create fields with confusing names. This will cause user frustration and users may end up not filing out the screens correctly or not at all!

Are you getting it?

Which of the following will likely cause users to become frustrated:

a. Confusing status names in workflows
b. Screens with too many fields
c. Screens that they use regularly to do their work suddenly change
d. Suddenly they cannot do things they used to be able to do in Jira
e. They start receiving a lot more or a lot less notifications

The answer is on the next slide.

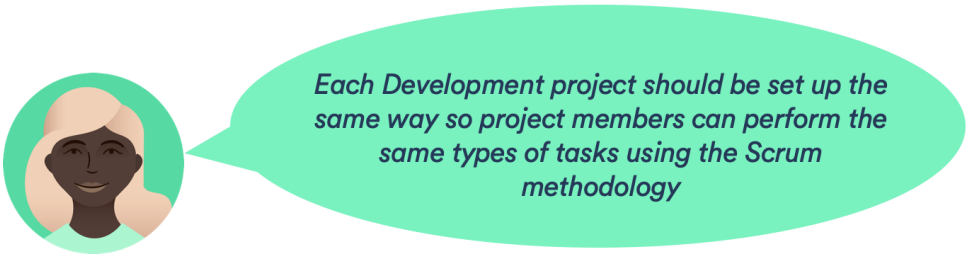
Did you get it?

Which of the following will likely cause users to become frustrated:

- ✓ a. Confusing status names in workflows
- ✓ b. Screens with too many fields
- ✓ c. Screens that they use regularly to do their work suddenly change
- ✓ d. Suddenly they cannot do things they used to be able to do in Jira
- ✓ e. They start receiving a lot more or a lot less notifications

Answer:
All of these things can cause users to become frustrated.
Use intuitive names and avoid confusing names for statuses in your workflows.
Create simple, easy to use screens.
If you change screens, permissions or notifications, let your users know before the changes are implemented.

Teams In Space business requirement



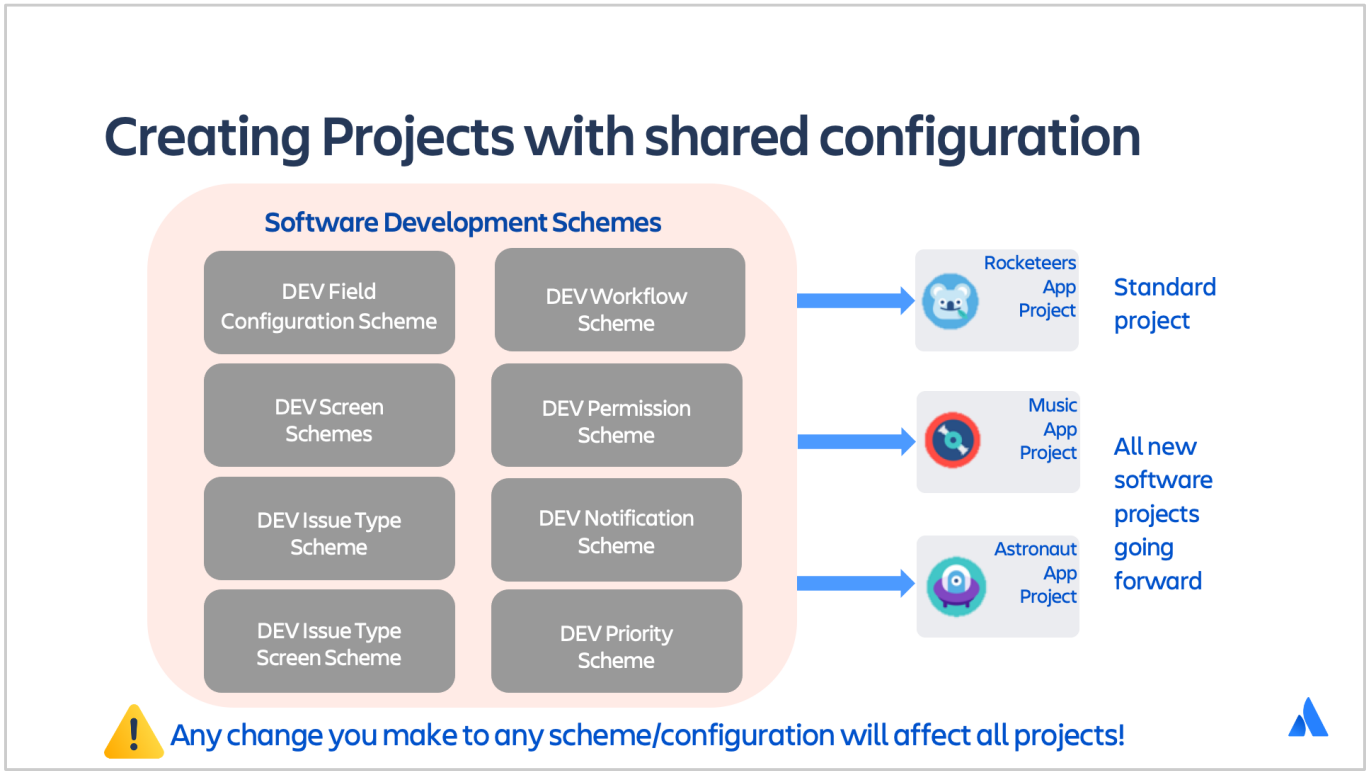
What should be done in Jira to meet these requirements?

Recall this business requirement from the beginning of the course and the reason why Teams In Space needs this.
You are the newly appointed Jira administrator and have to clean things up and enforce some control to manage the instance as you scale. Only you (and your team) will have the Jira Administrators permission.
There may still be a few projects that will have unique requirements, and for those projects, you will need to create some custom configurations and schemes. However, the majority of Software projects have the same requirements and will use these standard configurations and schemes.

Teams In Space Jira implementation

- Create a new standard Scrum software development project
- Configure a standard set of schemes and configurations associated with the standard project
- Share the standard project's configuration when creating Scrum projects
- Apply the new schemes and configurations to in progress Scrum projects

You've nearly completed your mission!
Based on the Teams In Space business requirements, you've created configurations and schemes in the context of creating a new standard DEV Scrum software project.
Now you'll use this standard DEV project to stamp out new Scrum projects going forward.
You'll also apply the new schemes and configurations to in progress Scrum projects where appropriate to standardize work and to reduce the number of schemes and configurations in your Jira instance. By sharing schemes among all the Scrum projects, you can make your Jira instance easier to administer and easier to use.
You'll also do some tidy up work and delete unused schemes.

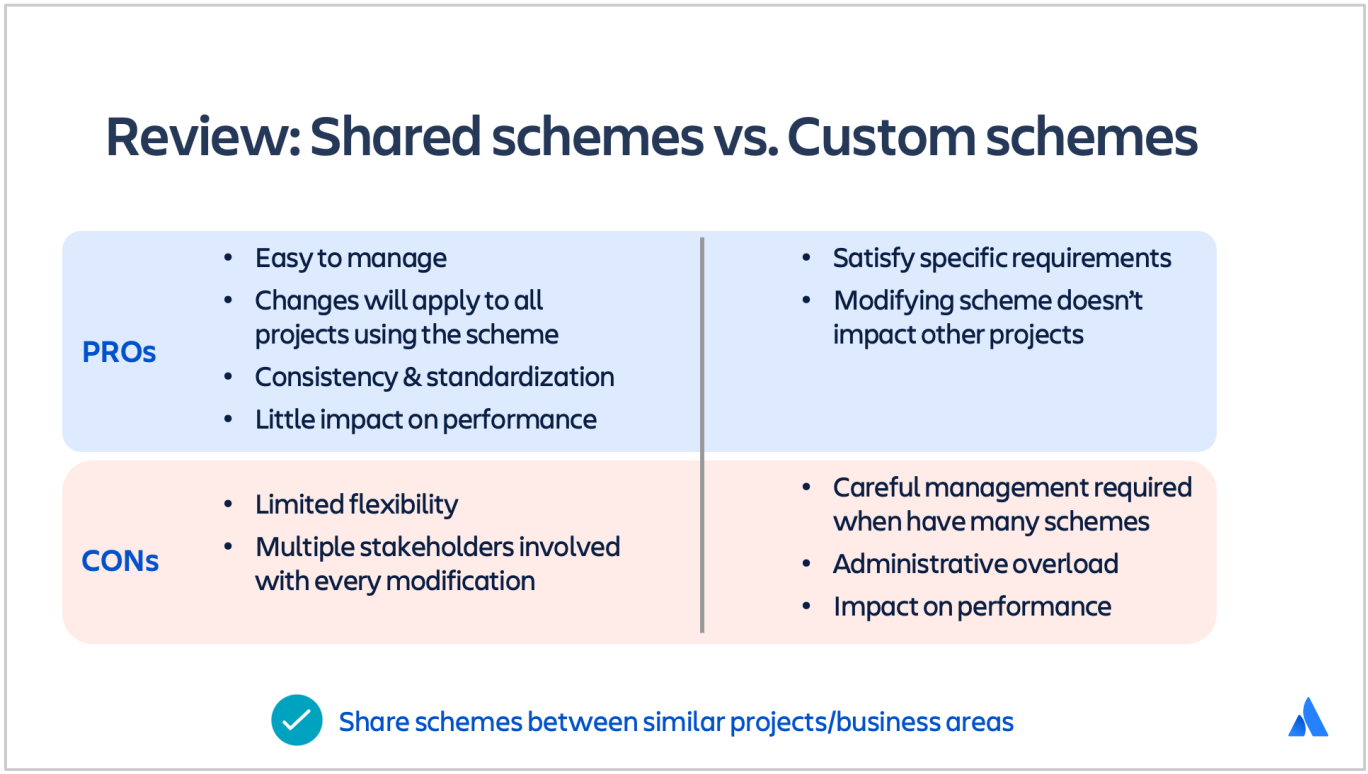


Now Teams In Space can create new Scrum projects using 'Create with shared configuration.' This will create new projects that share all the same schemes and configurations with the standard DEV project. Note that this includes ALL schemes, even the Permission and Notification schemes.

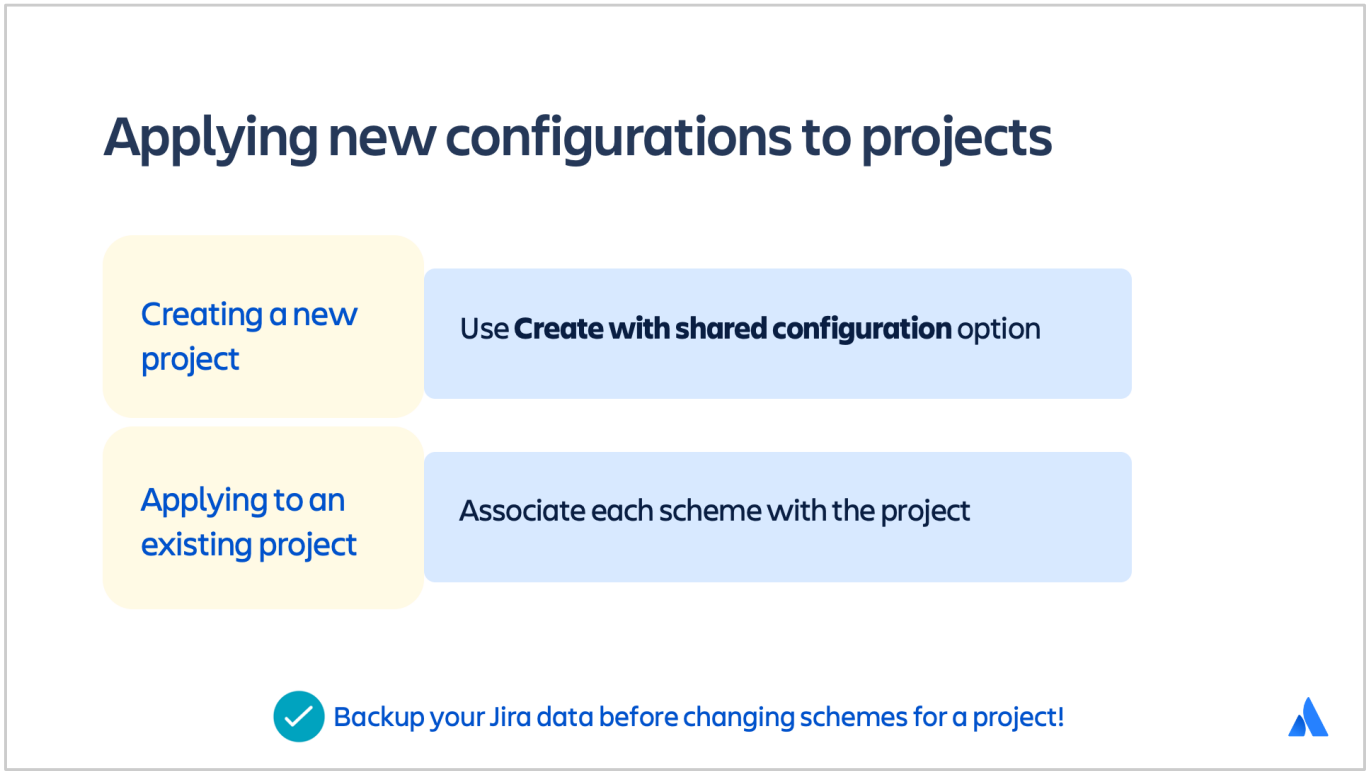
Recall that any change you make to any scheme or configuration will affect all projects that share these schemes and configurations.

They can use shared schemes since the general requirements (process, data, etc.) are the same. This makes things easier to manage in a growing instance, and especially in a large organization. You can ensure that each development project has the same schemes, and so each development project works in an expected way, which follows your organizational processes. This means that everyone will know how to use Jira to support the expected processes for that project type and helps to get the same project outcomes.

Note that when you create a project using the 'Create with shared configuration' option, you still need to assign group/users to project and board administrator roles for the new projects.



Sharing Schemes reduces Administrative Overhead. Generally, you want to avoid having too many schemes as that requires a lot of work to maintain. You want to share schemes as much as possible, where you have a reasonable belief that the shared configuration will stay in sync. Even if things go out of sync, most schemes can be easily copied and reverted if necessary. Do what you need to do to meet your business needs but try to re-use schemes and workflows. The more that are standardized across the whole system, the fewer features there are, and fewer screens and fields to be loaded every time an issue is created, edited, or closed. This will mean better performance. Reducing the number of schemes will improve performance. Having good governance over the creation of schemes becomes more important as your Jira instance grows. Too many schemes mean too much administrative overhead, too much complication, and poor performance.
Another advantage of sharing schemes is consistency. For example, developers switching to a different development project will have the same permissions, will get the same notifications, will see the same issue types, etc. This can make their use of Jira easier.
The disadvantages with shared schemes are that you have some limited flexibility, and as changes have an impact on all projects that use shared schemes, a lot of stakeholders (Product Managers, etc.) will need to be involved with every change.
Have as few schemes as possible and share them is the key to scaling Jira. Share schemes between similar projects types or departments, or align them with project categories.



To apply a project's configuration to a new project, when creating the new project select 'Create with shared configuration', then you can select which project's configuration to use.
To apply a set of configurations to an existing project, associate each scheme one by one to the existing project.
Backing up Jira data will be covered later in this module.

Changing schemes in existing projects

To apply configurations to an existing project you:

1. Perform a system backup
2. Manually change the schemes in the project, which may require you to:
   - Migrate issues to new issue types
   - Migrate issues to new workflow statuses
   - Configure the board for the new workflow

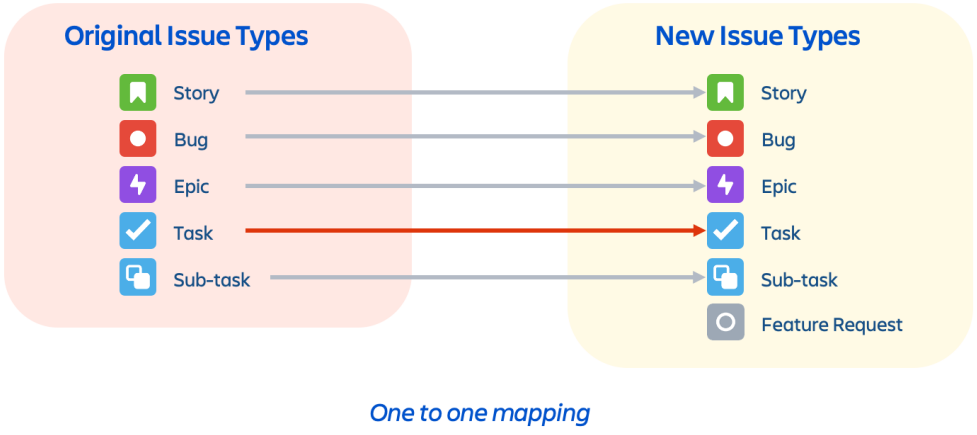
Before you change schemes in an existing in progress project, you should always perform a system backup.
To apply configurations to an existing project, you manually change the schemes in the project.
In doing so, you may need to migrate the issues to new issue types if they're different between the configurations.
Also, you may need to migrate issues to the new workflow statuses if they're different between the configurations.
And finally, you may need to configure the board for the new workflow if it's different.
We'll discuss these in the next slides.

Changing issue type schemes

Original vs. new schemes	Issue type migration
Same	None
Old scheme is a *subset* of the new scheme	None
Old scheme has *unused* issue types not in new scheme	None
Old scheme has *used* issue types not in new scheme	Choose a new issue type for old issue types not in new scheme

There are four scenarios for changing issue type schemes in an in progress project, and only one of them has a significant data impact.
In the first scenario, the issue types in both schemes are the same.
In the second scenario, the old scheme is a subset of the new scheme, so all issue types in the old scheme are present in the new scheme.
In the third scenario, the old scheme has issue types, not in the new scheme, but there are no issues in the project that use those issue types.
In these three scenarios, issue type migration is not required.
In the fourth scenario, the old scheme has issue types that are not in the new scheme, and there are issues in the project that use those issue types. In this scenario, the issues that use issue types that are not in the new scheme have to be migrated. Jira automatically starts the migration wizard, and you need to choose a new issue type for these issues. You have to tell Jira what to move the old issue types to because you can't have issues of an invalid type in a project. The wizard is similar to Bulk Move, except that you can't change the project for the issue. For more information, see https://confluence.atlassian.com/display/AdminJiraServer085/Associating+issue+types+with+projects.

Issue type migration example



One to one mapping

In this example, the original issue type scheme is a subset of the new issue type scheme. There is no Feature Request issue type in the original issue type scheme. Also, there is a Task issue type in the original issue type scheme that is not in the new issue type scheme. And the new issue type scheme has an Engineering Task issue type that is not in the original issue type scheme. During migration, issues of the Task issue type can be migrated to the Engineering Task issue type.

You should ensure that the new issue type is the right thing for your business data. Issue migration cannot handle splitting out issue types; it's a one to one mapping. If you needed to do this, you would need to find out if there is some way the feature requests are identified as being different from bugs e.g., for a specific field used or some unique text in the Summary or Description. If so, after you change the scheme, you could search for just these issues and then use Bulk Change – Edit Issues to update the issue type to Feature Request.

Changing workflow schemes

Original vs. new statuses	Status migration
Same	None
Old statuses are a *subset* of new statuses	None
Old statuses not in new workflow - *no issues in old statuses*	None
Old statuses not in new workflow - *issues in old statuses*	Choose a new status for old statuses not in new workflow

Each issue has to be in a valid workflow status. The valid statuses for an issue are defined by its workflow. This means that when changing a workflow, you may need to tell Jira the status for specific issues after the change.

There are four scenarios for changing workflow schemes in an in progress project and only one of them has a significant data impact.

In the first scenario, the statuses in both schemes are the same.

In the second scenario, the old statuses are a subset of the new statuses, so all statuses in the old scheme are present in the new scheme.

In the third scenario, the old scheme has statuses not in the new scheme, but there are no issues currently in the old workflow that use those statuses.

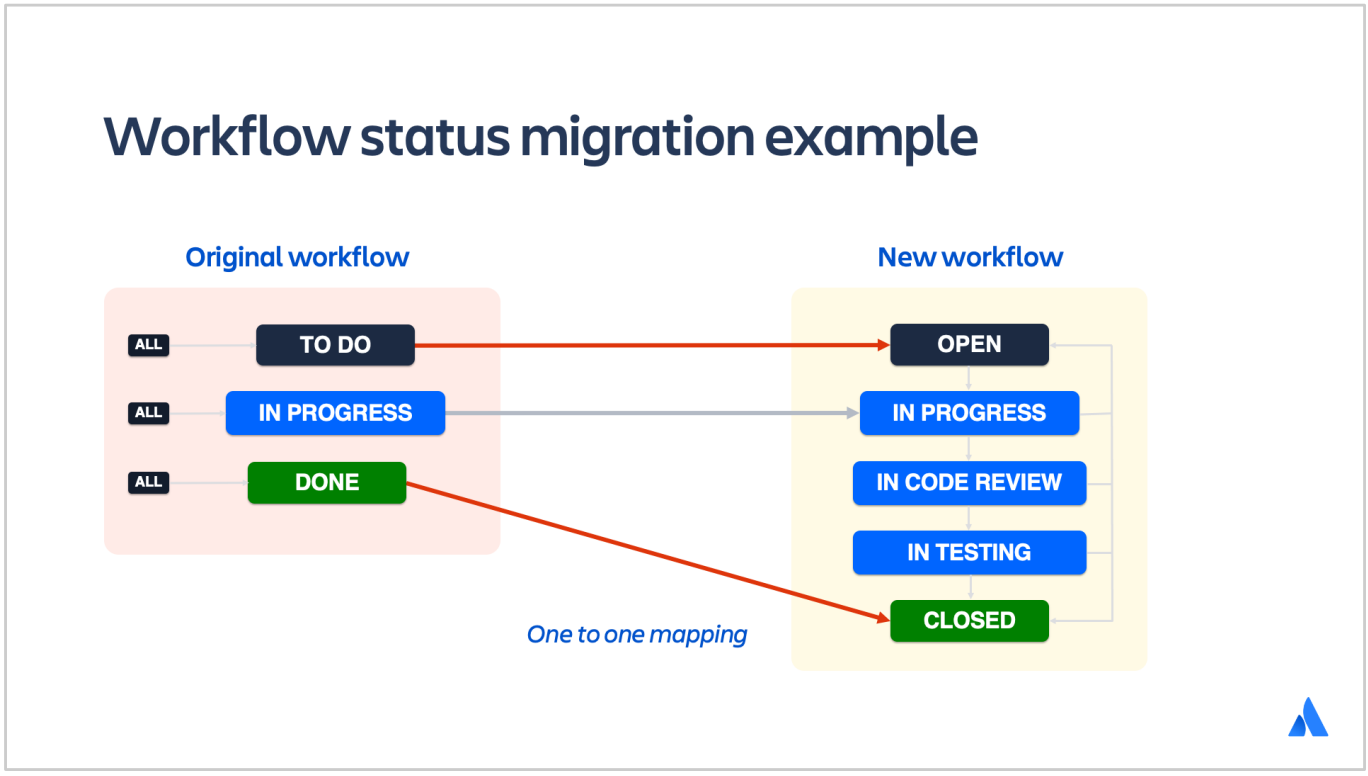
In these three scenarios, no issue workflow status migration is required.

In the fourth scenario, the old scheme has statuses that are not in the new scheme, and there are issues currently in the old workflow that use those statuses. In this scenario, these issues have to be migrated. Jira automatically starts the migration wizard, and you need to choose a new status for these issues.

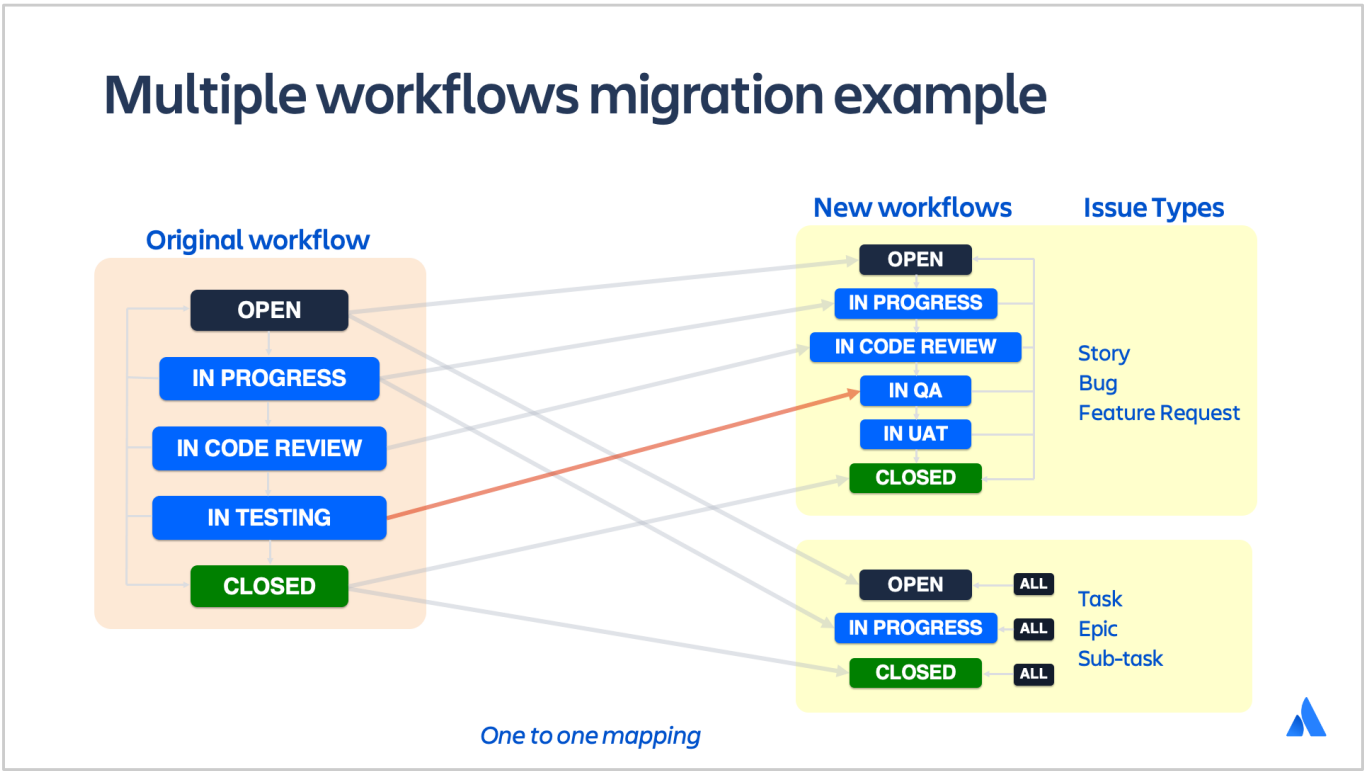
Jira will only ask you about issues where you are removing an existing status from a workflow. If you do that, it needs to know what status in the new workflow to flip the issues over to, but it won't ask you about anything else.

Changing workflows won't lose any data.

If an issue type is not defined in the project's issue type scheme, its workflow is not used.



In this example, the original workflow has only three statuses – TO DO, IN PROGRESS, and DONE, and it's using the simplified workflow where all statuses can transition into all other statuses. In this case, all issue types are using one workflow.
The new workflow has five statuses – OPEN, IN PROGRESS, IN CODE REVIEW, IN TESTING, and CLOSED. It's not using the simplified workflow.
During workflow migration, you choose how to map the statuses that are different. In this example, issues in the TO DO status are migrated to the OPEN status in the new workflow and issues that are in the DONE status are migrated to the CLOSED status in the new workflow.
No migration is needed for IN PROGRESS as that status exists in both workflows.
Workflow migration cannot handle splitting out statuses; it's a one to one mapping. For example, you could not migrate issues that were in the old status IN PROGRESS and split these out into two statuses, IN PROGRESS and IN CODE REVIEW.



This example shows how statuses are mapped when you change the workflow scheme for a project that had just one workflow to a workflow scheme that has multiple workflows. In a workflow scheme, workflows are mapped to issue types. So when the migration happens, and you've told Jira which statuses to map, it will map each issue to the appropriate status in the appropriate workflow depending on the issue type of the issue.

In this example, the new workflow scheme has one workflow for stories, bugs, and feature requests and a simple workflow for tasks, epics, and sub-tasks.

For all issue types, OPEN is mapped to OPEN, IN PROGRESS to IN PROGRESS, and CLOSED to CLOSED.

But stories, bugs, and feature requests are being mapped to a more complex workflow. IN CODE REVIEW is mapped to IN CODE REVIEW, and IN TESTING is mapped to IN QA. This last one is the only one that's different, so you would need to specify this during migration.

If there are any Tasks or Sub-tasks in either IN CODE REVIEW or IN TESTING, you can only choose to migrate them to one of those statuses that are in their new workflow i.e., OPEN, IN PROGRESS, or CLOSED.

Because the mapping is one to one during migration, we cannot split issues in IN TESTING between the new statuses IN QA and IN UAT.

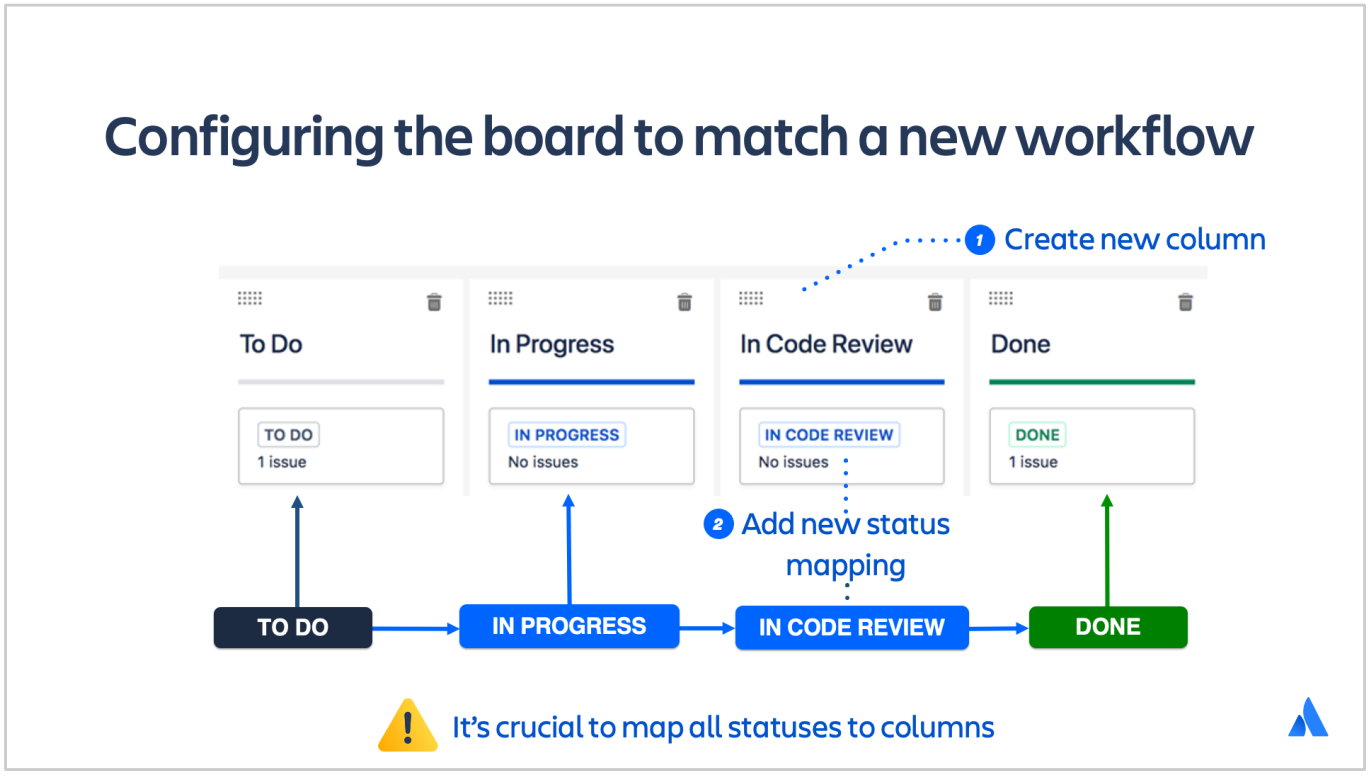
What to watch out for when changing workflows

- Changing the workflow for an issue type, affects **all issues of that type, open or closed**
- By default, all statuses will be migrated to the first status in the workflow

Ensure you select the appropriate statuses to migrate to

A change in workflow scheme affects the whole project. There is no selective migration. If you change the workflow for an issue type in the project, then all issues of that type are changed, irrespective of their current state in the workflow. For example, new or closed issues will be changed. If you want to retain the statuses of closed issues, you could keep old statuses in your new workflow. Ensure there are no incoming transitions for these. This would not work if you are trying to provide a single standard workflow to share among many projects, and there were many different statuses in the old workflows. Also, your workflow could end up very cluttered, depending on how many statuses you need to keep.

When you migrate workflow schemes, by default, all the old statuses will be mapped to the first status in the workflow, for example, OPEN. So if you just click through the migration wizard without reading it, you may accidentally map many statuses to OPEN that you didn't intend to. Ensure you select the appropriate statuses to migrate to.



Once you've changed the workflow scheme for a project, you need to configure the board in the project to match the new workflow (or workflows). The new statuses will automatically be added to the configuration page, but you need to create new columns if necessary, then drag the new statuses into the columns and drag the old statuses off the board. For example, here, the board administrator added a new column, In Code Review, as well as mapping the IN CODE REVIEW status to the column.
It's crucial that you map all statuses to columns. Otherwise, issues in "unmapped" statuses will be hidden from the board.

Changing field configuration & screen schemes best practices

Action	Best Practice	If you don't...
Apply a new screen scheme	Ensure new screens have all the fields users typically see	Frustrated users
Apply a field configuration with required fields	Ensure required fields are present on Create Issue screens	Issues can't be created
Add a new issue type	Also apply issue type screen scheme	Users won't see the familiar screen

If you use new screens (in the screen schemes) that don't have some fields that were in the old screens, they will no longer be visible. The data will still be there, but because you've switched to a screen without the fields, you won't see them. So if you apply a new screen scheme to a project, ensure the new screens have all the fields that users typically see. If you don't, users will be frustrated.

If you apply a field configuration that has required fields to a project, ensure those fields are present on the create issue screens in the project. If you don't, issues cannot be created.

If you add a new issue type to a project, ensure you also map the issue type screen scheme. If you don't, users won't see the familiar screen for that issue type.

Changing permission & notification schemes best practices

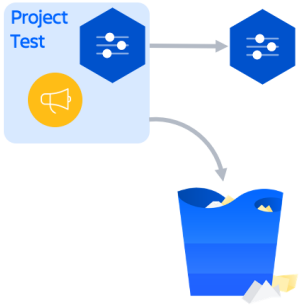
- Alert users of upcoming changes, especially if:
  - The new permission scheme is more restrictive
  - They'll be getting a lot more/less notifications
- Ensure project administrators add their team members to project roles

Ensure users are aware of the changes you make to permissions and notifications that will affect them. Especially if the new permission scheme is more restrictive, and users cannot do things they could do before. Also, alert them if they'll be getting more or fewer notifications.
If you use project roles in the permission and notification schemes (which we recommend), ensure project administrators add their team members to the roles.

What happens to schemes when you delete a project?

The schemes remain!

- May be shared with other projects
- You can reuse them in other projects
- You can delete them



If you delete a project, all the unique schemes created for that project remain, they are not automatically deleted. Also, if you change the schemes for a project, the original schemes remain.
This is because you may want to share the schemes with other projects. Also, you may want to reuse them in another project. Or you can delete them. (see next slide)

Delete unused schemes

- Occasionally delete unused schemes
- Unused schemes clutter up admin pages and make administration more difficult

Don't delete schemes you may want to use again!

It's a good idea to clean up unused schemes. These can clutter up Jira administration pages and make administration more difficult. These could be schemes left over when projects are deleted or when project schemes are changed. Ensure you only delete unused schemes or schemes that are not in use by projects. Only delete schemes when you are sure they will not want to be used again.
For example, sometimes, you need to test something by creating a test project. Then later, you delete that project. Don't forget also to delete the schemes that were created as part of that project. Ensure that those schemes are not shared by any other project.

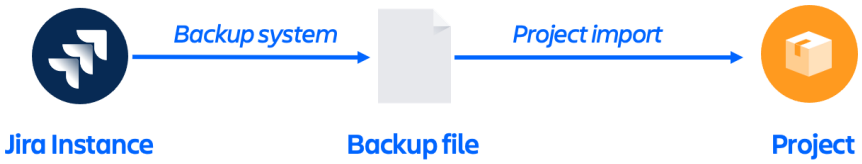
When you can delete schemes

Scheme	Can only delete if it's...	If it's associated with a project...
Workflow	inactive	
Screen	not used in an issue type screen scheme	
Issue type screen Field configuration Notification	not associated with any projects	
Issue type Permission		The project will be associated with the default issue type/permission scheme

This table shows you the circumstances under which you can delete schemes.
You can only delete inactive workflow schemes.
You can only delete a screen scheme if it’s not used in an issue type screen scheme.
You can only delete issue type screen schemes, field configuration schemes, and notification schemes that are not associated with any projects.
You can delete issue type schemes and permission schemes that are associated with a project. If you do, the project will revert to using the Default Issue Type Scheme or the Default Permission Scheme respectively.

Backing up your system & restoring a project

- Before you change a project's schemes, perform a system backup
- If something goes wrong, restore just that project



When you change a project's schemes, you may need to migrate issue types and workflow statuses, which will change issue data. In case something goes wrong or you make the incorrect selections, you should back up your Jira data. To backup your data, you can use Jira's XML backup utility in Jira administration. The backup file is stored in the export folder under the Jira application home directory. This backs up all your database data, but it doesn't backup attachments (and other data such as user's avatars). These are stored in the data folder under the Jira application home directory. It's a good idea to backup this folder on a regular basis.

To restore just one project, you can use Jira's project import tool in Jira administration. It uses the system backup file to import a single project.

Other reasons to backup your data are before an upgrade and when splitting your instance across multiple servers. There are two ways to backup the database contents:

•Using Jira's XML backup utility.

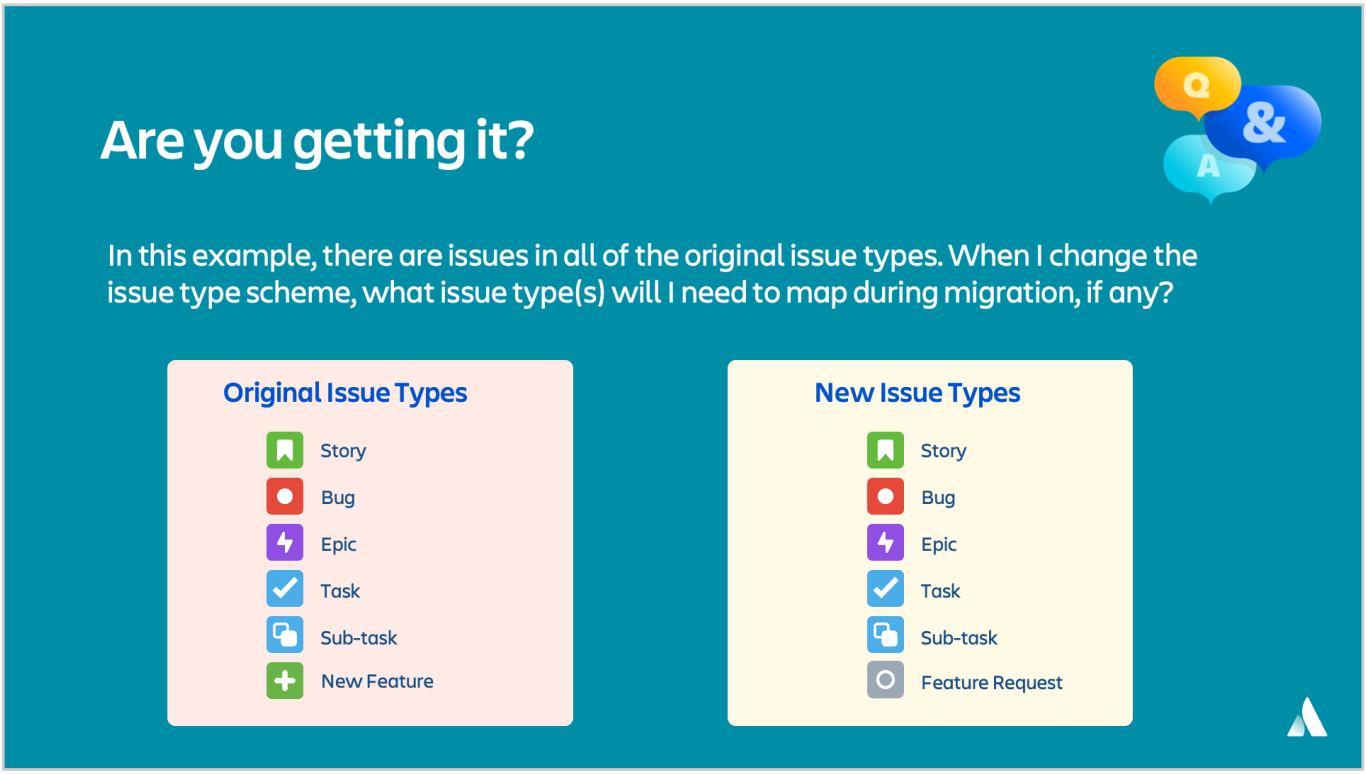
•Using native database backup tools. For production use, it is strongly recommended that for regular backups, you use native database backup tools instead of Jira's XML backup.

For more information on backing up data, see https://confluence.atlassian.com/display/AdminJiraServer085/Backing+up+data. For more information on restoring a project from backup, see https://confluence.atlassian.com/display/AdminJiraServer085/Restoring+a+project+from+backup.

See how it's done

Go through issue type migration

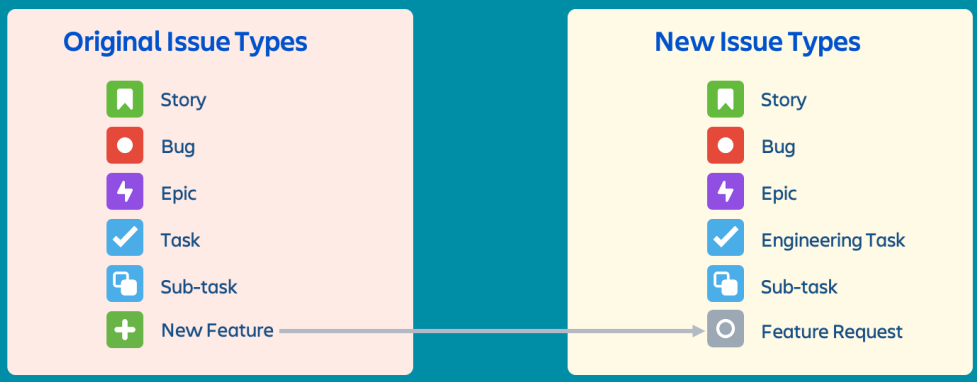
Instructor demo.



The answer is on the next slide.

Did you get it?

In this example, there are issues in all of the original issue types. When I change the issue type scheme, I'll need to map New Feature to Feature Request during issue migration.



Takeaways

- Use few schemes and share them
- Use project roles in schemes and ensure project administrators add their team members to these roles
- Alert users of upcoming changes before you apply new schemes to existing projects
- Backup your Jira data before changing schemes for a project

Try it

Lab 6 – Applying New Configurations to Projects

- Exercise 1 – Stamping out a new Scrum Project
- Exercise 2 – Applying Configuration to Existing Projects
- Optional Exercise 3 – Exploring More Consequences

Atlassian skills are in high demand in the job marketplace. The path to Atlassian certification is designed to help you deepen and expand your Atlassian skills, and prepare you to take on your next challenge.

Follow the path to Certification – you'll propel your career, boost your skills and learn new ways to help your teams do their best work.

To find out more, visit **atlassian.com/certification**.

Training for Jira is a new Marketplace app containing a growing collection of short, interactive tutorials, designed to get teams over the hump and using Jira quickly.

Why Training for Jira?

- Help teams help themselves to training and reduce the number of Jira questions.
- Training for Jira scales – it's built right into Jira and is available to everyone.
- The tutorials are short and topic-focused; users can choose what to learn and when.
- The app also keeps track of users' progress so they can see where they left off, and so managers can track the progress of their teams.

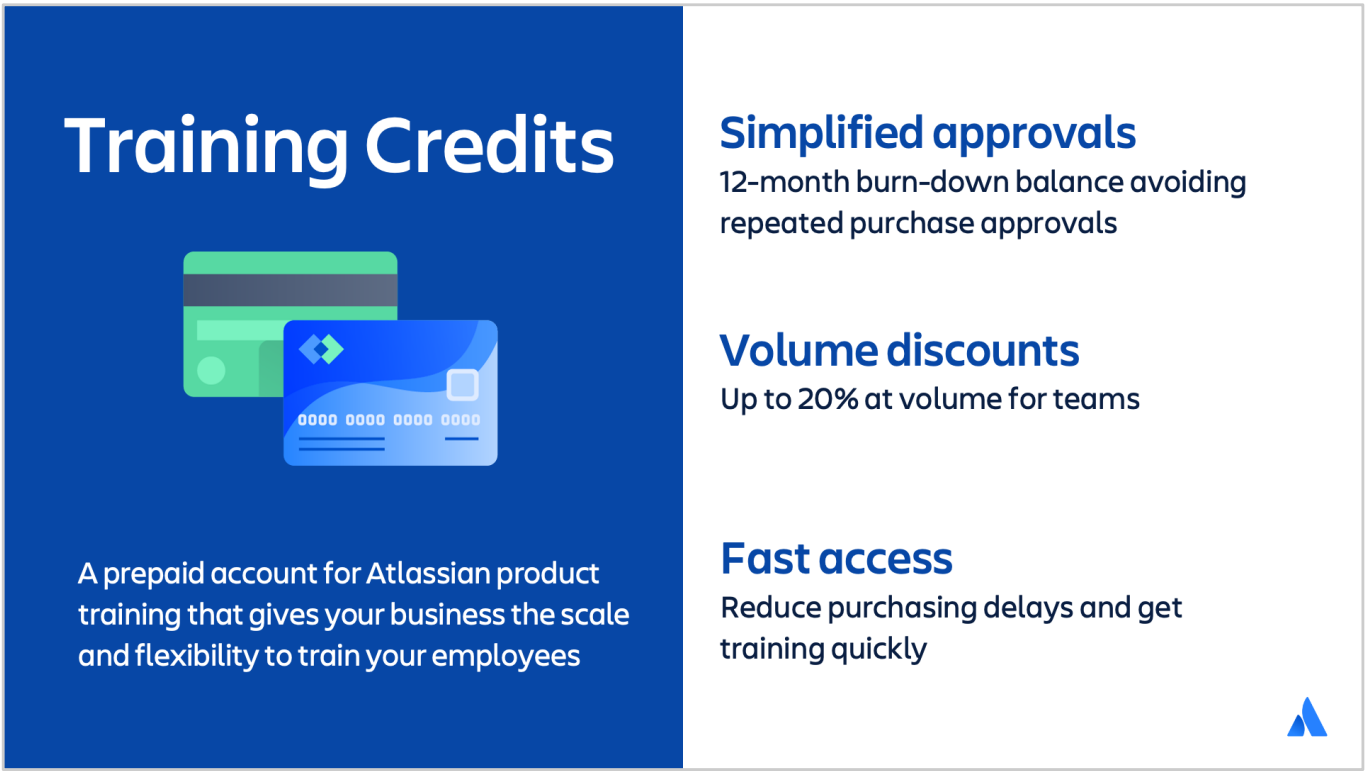
The Tutorials...

- Cover popular topics and tasks
- Introduce essential concepts
- Showcase in-product demos
- Include non-graded quizzes so you can see if you're getting it
- Have optional voice-over narration and closed captions

Available for Cloud, Server, and Data Center

Learn more

- Search for "Training for Jira" from marketplace.atlassian.com to download a free trial.
- Requires internet connectivity to access content
- Note: If you're using a firewall or have network restrictions where Jira is installed, you may need to whitelist IP addresses to access this app.



Simplified Approvals

Training credits allow your procurement team to establish a 12-month burn-down balance for Atlassian training. With this simple arrangement, you'll no longer need to process repeated purchase approvals for small transactions.

Volume Discounts

By pre-paying for training as an annual credits program, you can take advantage of volume training discounts to get a more effective return on your training budget.

Fast Access

With training credits, you reduce the purchasing delays for your business users and make it faster to access the training they need.

Want to learn more?

Training & Certification	atlassian.com/university
Training & Certification Online Community	go.atlassian.com/traincertcommunity
General Online Community	community.atlassian.com
Documentation	confluence.atlassian.com
Support	support.atlassian.com
Enterprise Services	atlassian.com/enterprise/services
More Resources	atlassian.com/resources

Join your local Atlassian Community Events (ACE) group

What happens at an ACE?

A diverse group of Atlassian users, both technical and non-technical, come together to share best practices and enjoy food and drink.

How do I join?

ACEs are in over 30 countries around the world. Go to aug.atlassian.com to join a group near you.

No ACE in your city?

You can start one! Go to aug.atlassian.com/leaders to sign up. You can also connect with fellow users on community.atlassian.com

Please take the survey and give us your feedback

Congratulations on completing the course!