JIRA ESSENTIALS

1 Course Overview

What will you learn?

- Use lean and agile principles
- Differentiate kanban and scrum
- Configure Jira to match your team's current process

To succeed here, you need to have

- No knowledge of agile or Jira is assumed

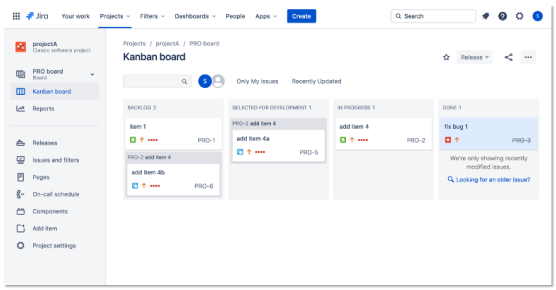
Schedule

1	Course Overview
2	Agile and Jira Overview
3	Visualize Work Using Project Boards
4	Enrich Issues
5	Kanban Method
6	Lean and Agile Principles
7	Scrum Overview I- Artifacts
8	Scrum Overview II- Roles and Events

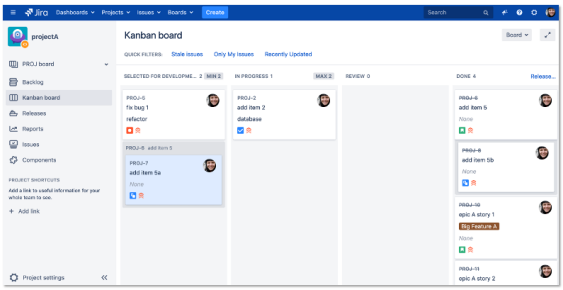
Schedule

9	Quick Search and Basic Search
10	JQL
11	Filters
12	Epics
13	Dashboards
14	Putting it all Together

Jira Cloud vs Jira Server/Data Center



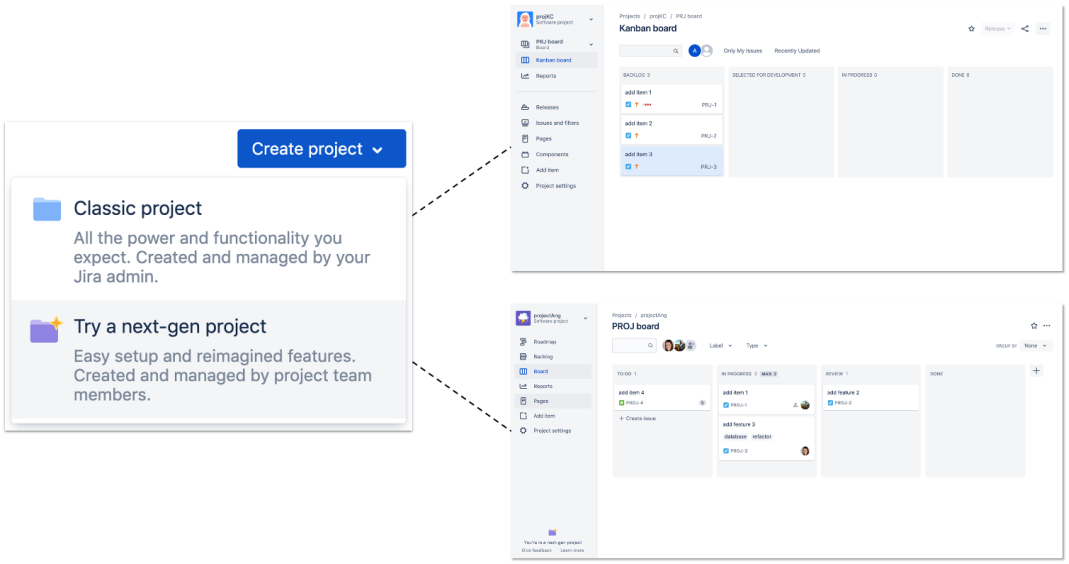
Cloud



Server /
Data Center

There are two main ways that a company can choose to run their Jira instance. Using Cloud, Atlassian does the work of hosting and maintaining the Jira software, so that the company can focus on their value-added activities. This is much like a company that chooses to host their email using an external service such as Google Gmail.
If a company wants to host their own Jira instance, they could choose the Server or Data Center version. They can host their Jira instance in the cloud, for example using Amazon Web Services (AWS), or they can host their Jira instance in their own data center. The Server version of Jira is hosted on a single node. The Data Center version is hosted on multiple nodes, leveraging the reliability benefits of a distributed architecture.

Jira Cloud- classic vs. next-gen projects



In the Cloud version of Jira, you are given a choice of creating a classic project or a next-gen project.

Classic projects are traditional Jira projects closely resembling Server/Data Center projects. In general, for now they are more mature and have more functionality than next-gen projects. They have more configuration capabilities, and configurations can be shared among multiple projects. They are mostly created and managed by Jira administrators.

Next-gen projects are newer to Jira, and capabilities are being added by Atlassian engineers in an agile way. In general, for now you will see more changes to next-gen projects than to classic projects. While there are many similarities, Next-gen projects have a different perspective on managing the projects. Instead of Jira administrators mainly creating and configuring projects, next-gen projects can be easily created and configured by the members of the project team. The configuration applies only to that project, so team members don't have to worry that they are changing the configuration of other projects.

The type of project that you create is up to you and your organizations. Some organizations may choose to only use classic projects and keep better control over the project configuration. Some organizations may prefer the ease and flexibility of next-gen projects. Some organizations may allow the project teams to choose.

Tasks

Log in to Jira

- ❑ Decide if you want to do the Cloud or Server version of the labs
- ❑ Log in to Jira

2 Agile and Jira Overview

What will you learn?

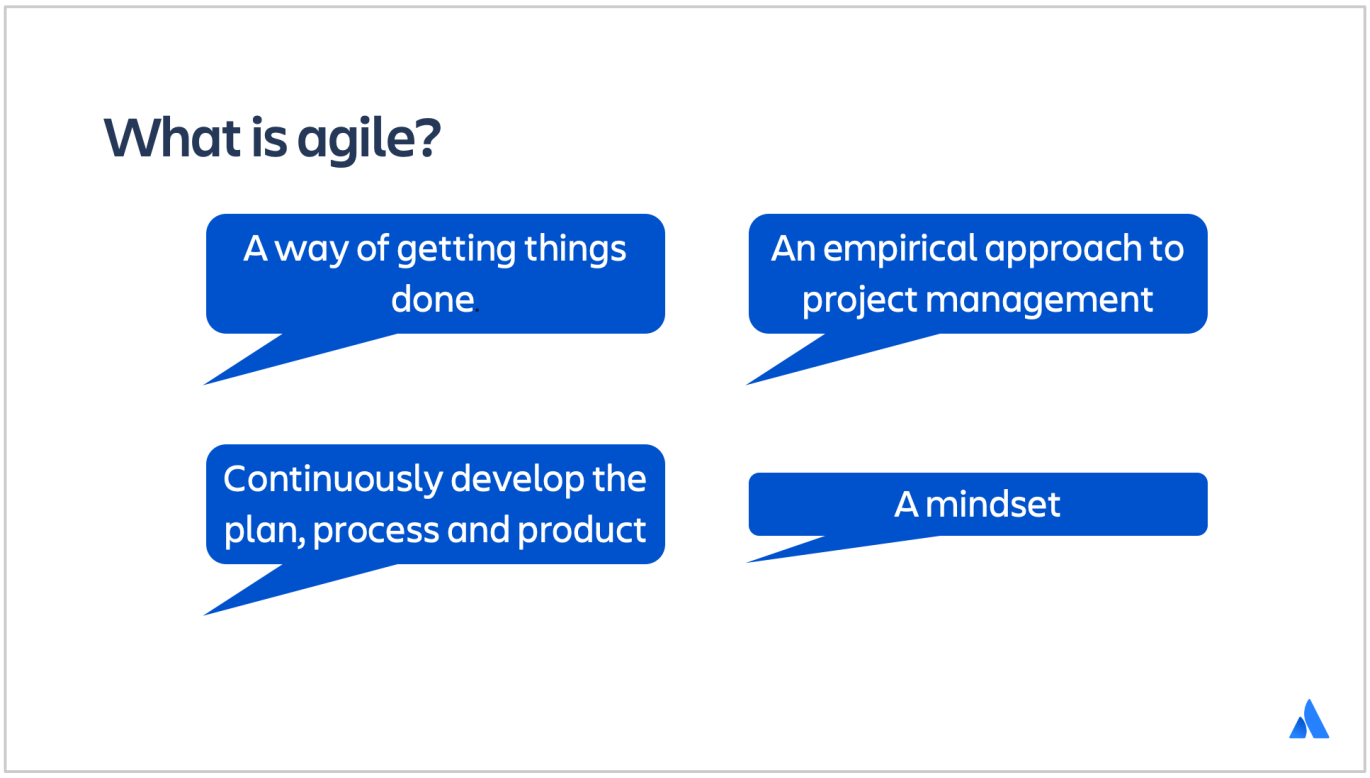
- Describe agile
- Describe Jira
- Identify how Jira relates to an agile mindset
- Create a Jira project
- Create a Jira issue
- Use a project board
- Identify Jira user types

Topics

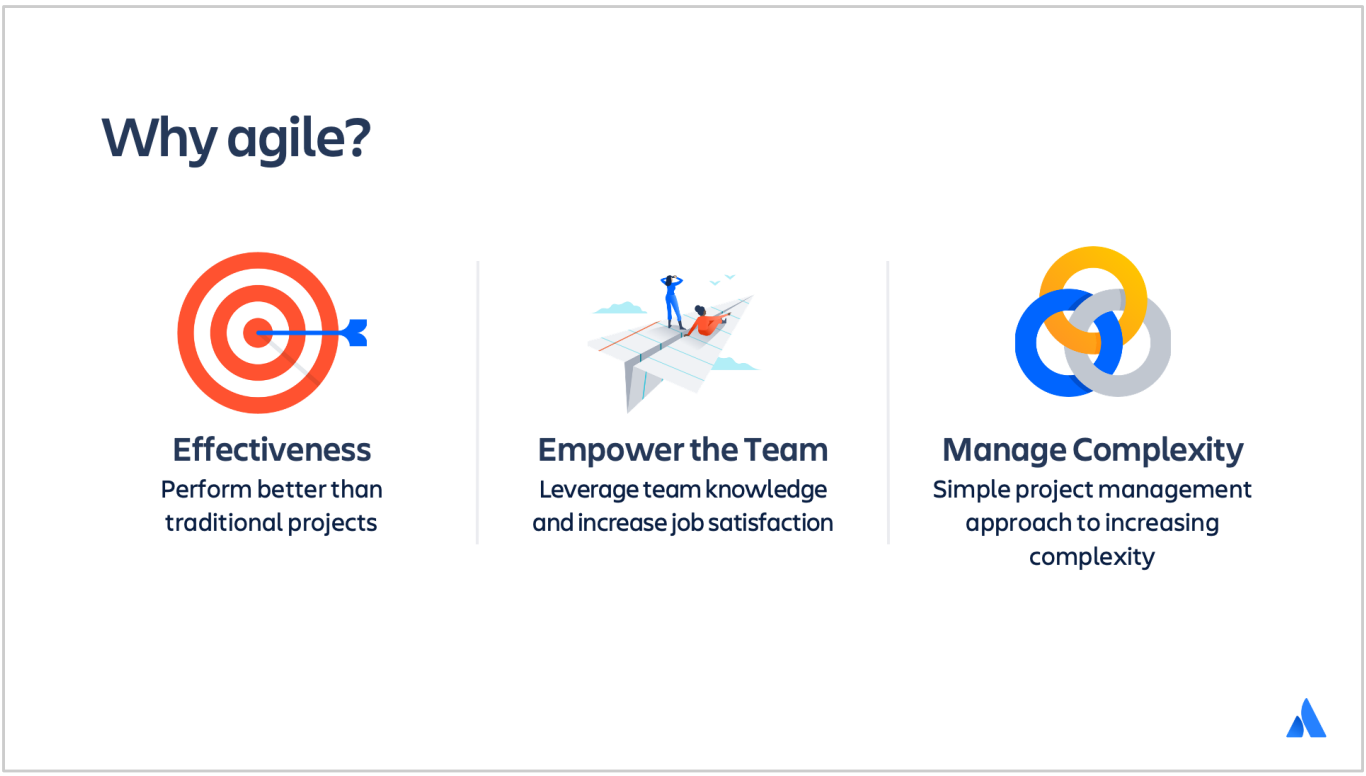
Agile overview

Jira overview

Projects, issues and boards



If you ask ten people to define agile, you would probably get ten different answers. Agile is a way to get things done.
It is an empirical approach to project management, meaning that decisions are driven by the results of experiments.
With agile projects, the team is continuously developing and improving not only the product, but the plan to build the product and the process used to create the product. This is different from traditional project management, where a big upfront plan is created, the process to create the product is well-defined, and the product is built in stages.
Agile is not a methodology or framework, it is a mindset. That is, it is a way of thinking about or approaching project management.



In many contexts, teams that work in an agile way tend to be more effective than teams that work in traditional ways. An agile team can often build a product with more speed, reliability and at a lower cost than traditional teams.
Agile companies empower their agile team members. Team members are often in the best position to make certain decisions, such as how to get the work done. An empowered person in generally happier than an un-empowered person, which leads to higher overall job satisfaction.
As the complexity of projects grow, using a complex project management approach adds to the complexity and decreases the chance of project success. Agile approaches to project management are relatively simple, allowing the overall project complexity to be much more manageable.

What is an agile mindset?

- A growth/continuous improvement way of working
- Allowing the data to change your approach
- Uses agile techniques to accomplish work

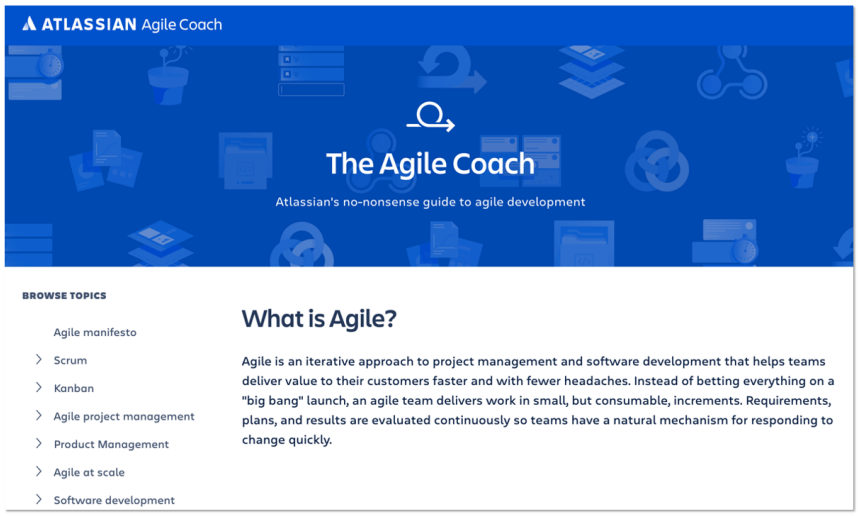
A mindset is a way of thinking and working. An agile mindset embodies the principles of agile, which we will discuss throughout the course. A main principle is to continuously grow and improve as you learn.
Agile is an empirical approach to project management, meaning that it is scientific in nature, allowing the data to change your approach.
An agile mindset uses agile values, principles and techniques to accomplish work. We will discuss these throughout the course.

Why an agile mindset?

For an agile team to perform its best, all team members must have an agile mindset

A team can not just be told to “be agile”. For a team to achieve optimal effectiveness, every member of the organization, including leaders and team members, must have an agile mindset.

The agile coach (atlassian.com/agile)



The agile coach site (atlassian.com/agile) contains great resources related to many aspects of agile. You are encouraged to take a look!

Topics

Agile overview

Jira overview

Projects, issues and boards

What is Jira?

- A tool used to help teams perform, visualize and manage work
- Models the team's current processes/workflows

Jira is a tool that teams can use to help facilitate agile work. It helps the team perform their work, visualize work and manage the work using things like boards, reports and dashboards. A team can use Jira to help make better decisions throughout the life of the project.
Jira is very configurable, allowing the team's current desired processes or workflows to be modeled and used to accomplish the team's work. You want the tool to conform to the processes that your team desires, not be constricted by the capabilities of the tool. Because agile process are always open to improvement, you want your tool to be adaptable as your team learns. A solid understanding of Jira's capabilities helps the team work with confidence as they continue to improve.

Why Jira?

- Leverage project management technology, allowing teams to focus on their work
- Facilitates planning, prioritizing, organizing and completing work
- Visualizes work using project boards, reports and dashboards
- Facilitates team communication

Jira provides administrative project management capabilities out of the box, so teams can focus on the work that differentiates them. For example, Jira reports are automatically created and updated as the team performs its work.
Jira helps the team focus on its work, facilitating planning, prioritizing, organizing and completing the work of the team.
A key agile/lean principle is to visualize work. Jira provides work visualization out of the box using things like project boards, reports and dashboards.
Jira facilitates team communication in ways such as visualizing the work of the team, informing the team of the planned work, allowing teams to discuss work items, and by notifying team members when the work items change.

How does Jira relate to an agile mindset?

Jira is a tool that teams can use to model and execute their agile processes

Agile is a mindset. Jira is a tool that enables a team to perform their work in an agile way.

Topics

Agile overview

Jira overview

Projects, issues, boards and user types

What is a Jira issue?

- An item of work (work item) identified by the team
- An issue has an associated type (for example, story, task, bug)
- The details of the issues are known as *fields*

Issues

A Jira issue is the name of an item of work that has been identified by the team. The term “issue” comes from Jira’s historic roots as bug, or issue tracking software.
Every issue has an associated issue type. You will work with these throughout the course, but issue types include stories, tasks, bugs, epics and custom issue types. Each issue type can have unique screens and workflows associated with them.
An issue can contain a lot of information. This information is broken down into fields. Fields include the name of the issue, its unique identifier, its description, comments, date of creation, current assignee and many others. Custom fields can also be created to match your team’s desired business processes.

What is a Jira project?

- A collection of related issues
- A team “to do” list
- Can have a fixed end date or be an ongoing project
- A project has an associated type (for example, kanban, scrum)

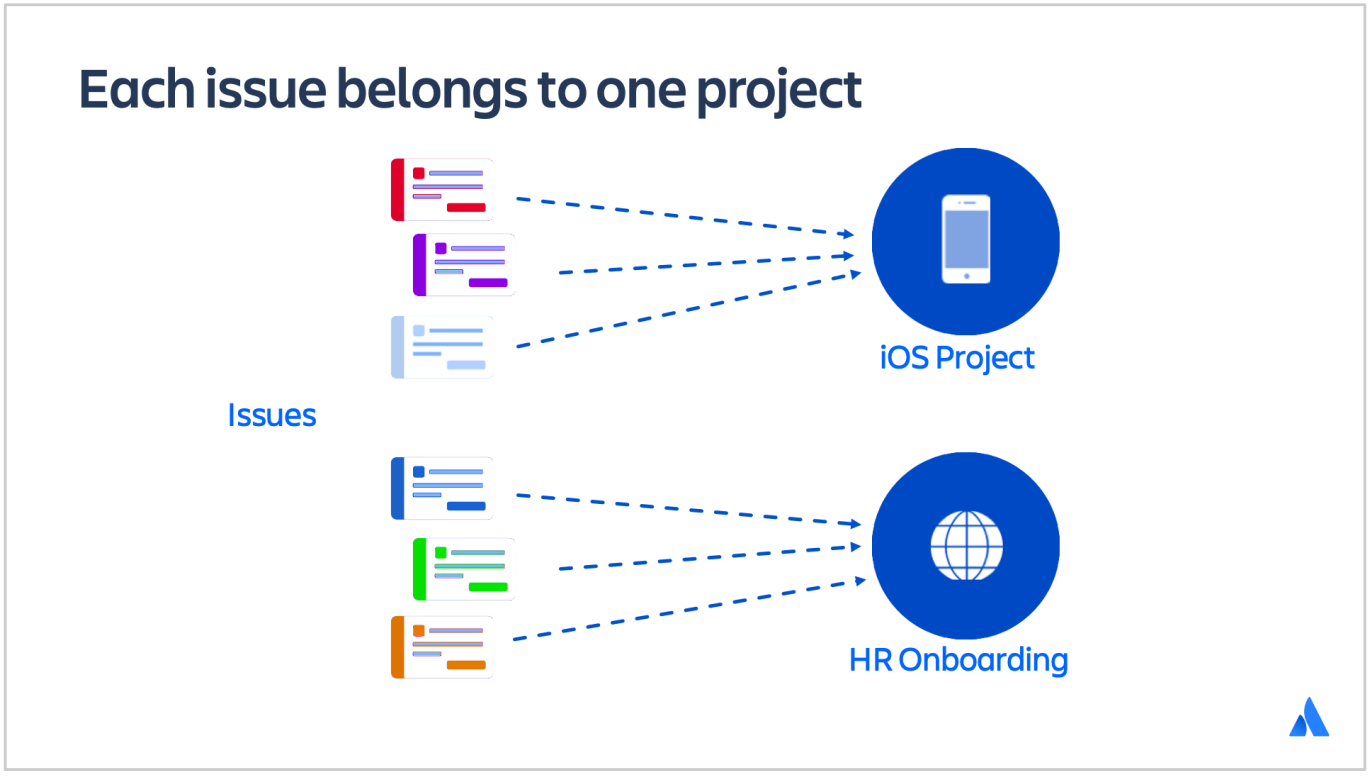
A Jira project is a collection of related issues. A project is a way to organize work. The issues can be related in any way that the team desires.
You can think of the issues of a project as the team’s “to do” list. The issues can be in different states, such as not ready for development, ready for development, in progress and done. Jira therefore contains a record of the team’s work, allowing the team to do things like report on their work.
A Jira project is not necessarily a project in the traditional sense, which usually has a start and end date. The term “project” is used loosely to include work that has no planned end date. For example, a product that is planned to continuously be improved, with no plan to ever stop work on it.
Projects have an associated type, depending on how the team wants to accomplish their work. When you create a project, you select its type, such as kanban or scrum. We will discuss these later in the course. Over time, the team can configure their project to use a custom agile methodology or framework.

Issue key

Jira automatically assigns a unique *issue key* to created issues

<issue_key> = <project_key>-<issue_number>

Jira automatically assigns a unique issue key to each issue. The letters before the dash represent a unique identifier for the project. This is called the *project key*. The issue number in the project follows the dash. Issue key values are unique to the Jira account. To ensure this, you can not have two projects with the same project key.

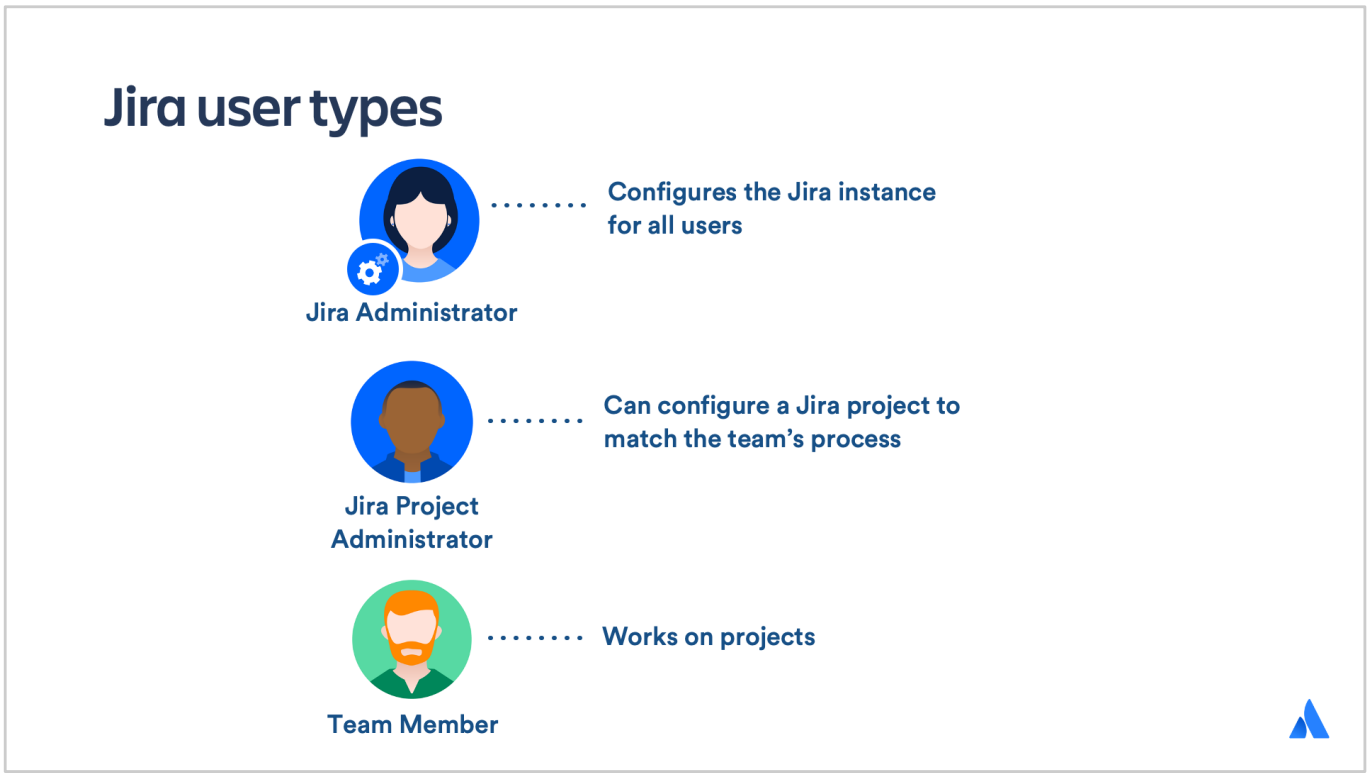


Every issue in Jira is unique and belongs to just one project. The issues are the project's work items. For instance, the HR Onboarding project may have issues such as create email account, supply laptop and assign desk. The iOS project may have issues such as code login button, create help screen and configure iCloud.
No issue can belong to multiple projects.

What is a project board?

- A two-dimensional “to do” list
- A way to visualize issues
- A visualization of the team’s process/workflow
- Displays issues as *cards*

A project board is a two-dimensional view of the work to be done by the team. The two dimensions become more important than with a personal to do list because the work items can be in multiple states and be worked on by multiple team members. Project boards visualize the work process of the team, and can be physical boards (such as sticky notes in columns) or digital boards (such as Jira boards).
On a Jira board, each issue is shown as a card, which displays a convenient subset of the issues’ fields for easy visualization. The fields displayed on a card can be configured to match the team’s desires.



There are three main types of Jira users. A *Jira administrator* configures the Jira instance for all users. They in general know the most about the technical capabilities of Jira and can set policies for the entire company to use with their Jira projects. The changes that they can make can affect multiple projects, so they must be very knowledgeable of Jira.
A *Jira project administrator* can configure a specific project to match the team's desired process. Jira project administrators work closely with the agile team to understand the desired process, and works with the Jira administrator when they do not have permissions to perform some of their desired Jira configuration.
A team member uses Jira to work on projects.
In general, a company has a few Jira administrators, more Jira project administrators, and even more team members.

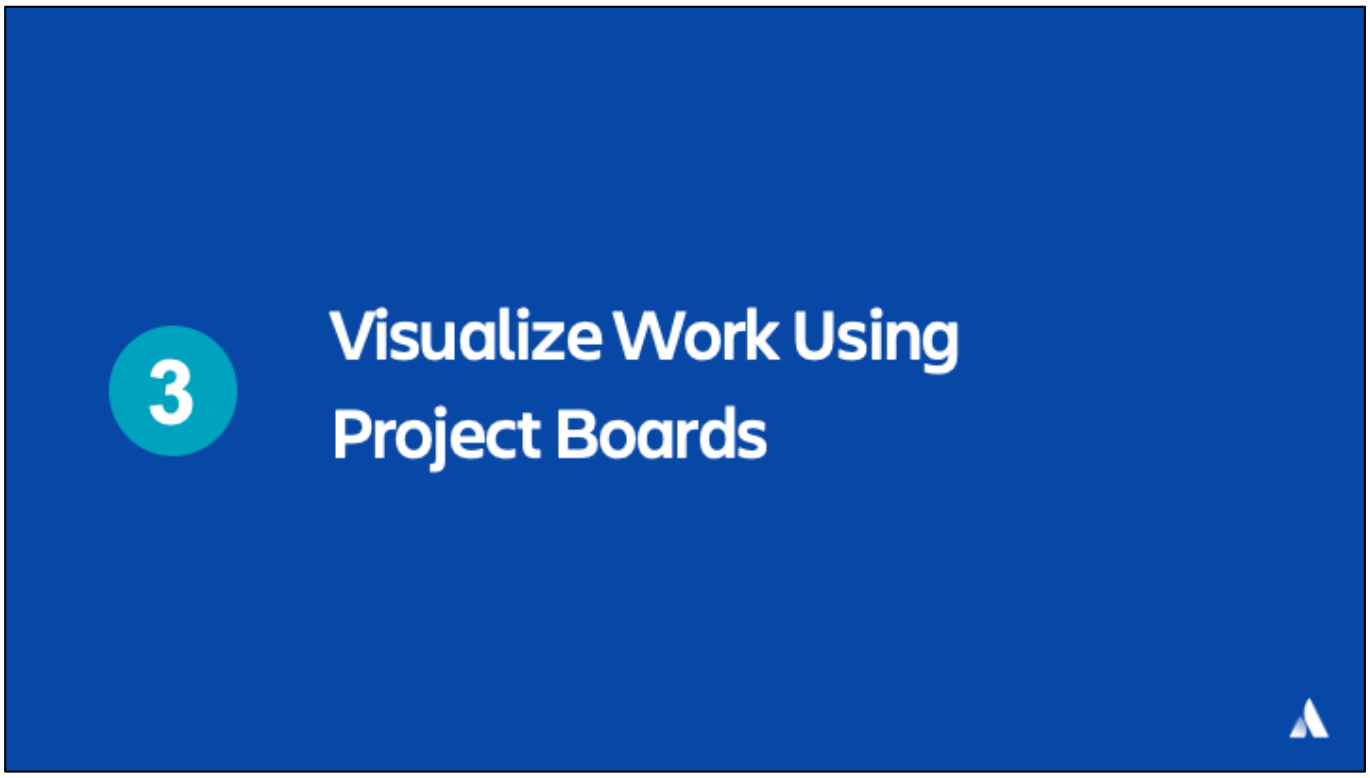
Takeaways

- Agile is a way of working on projects
- Jira is a tool teams use to manage and visualize the work of a project
- Jira can be configured to match a team's continuously improving processes
- A Jira issue is an item of work identified by the team
- Project boards visualize a team's work
- The main types of Jira users are Jira administrators, Jira project administrators and team members

Tasks

Create a project

- Create a project
- Create issues


3
Visualize Work Using
Project Boards

What will you learn?

- Describe the importance of visualizing work
- Describe common workflows
- Differentiate Jira boards and workflows
- Describe the purpose of an issue's *status* field
- Configure board columns

Topics

Visualizing work

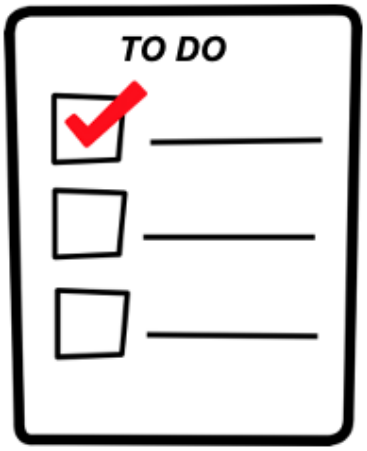
Workflows

Jira boards and workflows

Configuring board columns

Visualizing work: a "to do" list

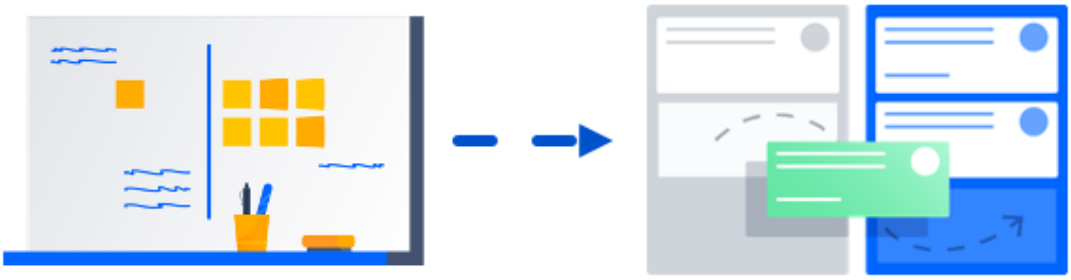
- Reminds you
- Focuses you
- Sets priorities
- Tracks progress



A to do list is a classic example of a tool used to visualize your work. Even though a to do list is very simple to use, it has some valuable characteristics.
One is that it visually reminds you of the work that you need to do. We all have a tendency to forget things if we don't write them down, especially if there are a lot of items. A to do list helps prevent you from forgetting things.
A to do list focuses you, because you can concentrate on doing only the things on the list.
If you want, you can prioritize the work items on your to do list, simply by changing their order.
As you check off your work items, you are tracking your progress. This is usually quite rewarding.
We will see that this simple example of a to do list contains a lot of the building blocks related to visualizing the work of agile teams.

Visualizing work: a board

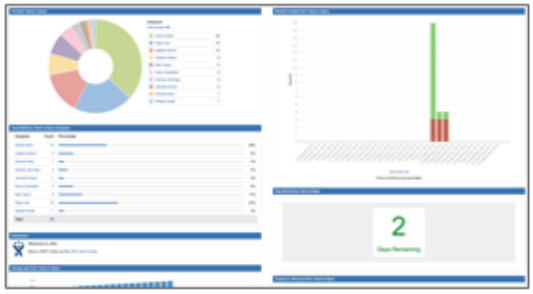
- A principle of agile projects is to "visualize work"
- A board is an agile tool used to visualize and manage work



A principle of agile projects is to visualize work. A board is an agile tool used to help visualize and manage the work of the team. Depending on the circumstances, a board may also be referred to as a task board, a project board, a kanban board or a scrum board.
A board can be a physical board like a whiteboard with sticky notes, or can be based in software. The board shown here is in Jira.
Depending on which template that you choose when creating the project, a board can be automatically created for you.
You can see that the board contains columns. Each column can contain issues, which are the work items for the project.
You can see that a board is like a two-dimensional to do list. The extra dimension is used to allow a work item to go through multiple steps by multiple team members before it is finished. Agile teams commonly break up the project into manageable issues and work on them in steps like this.

Visualizing work: reports and dashboards

Reports



Dashboards

We saw that project boards are used to visualize the work of the team. In Jira, reports and dashboards are automatically created and updated for you. This is another great way to visualize the team's work, especially for identifying problems.

Why visualize work?

- To easily **see** the work of the project
  - Allows anyone to see the (true) current state of the project
  - Organizes and focuses the team
  - Only work on tasks on the board
- To **manage** things
  - Easy to add and prioritize the work of the project
  - Easy to update work items
- To **improve** the team's way of working
  - Can visually identify problems

Why is visualizing work an important principle of agile? Like a to do list, visualizing work with tools such as a board makes things easier, better and more rewarding.
Visualizing work allows everyone to see the current state of the work of the project. With a board, you can easily see which items are done, which items are in progress, and which items have not been started. Boards are very transparent in that they show the true state of the project, not only to the project team, but to any stakeholders who have access to the board.
Visualizing work also organizes and focuses the team. The team should only be working on the issues on the board, and the next issue to be worked on is very obvious.
Visualizing work allows you to manage things. For example, the team can easily add items to the board, modify existing items or change their priority. As the team works on a work item, the item can be updated to show the progress.
Visualizing work can also improve the team's way of working. A key principle of agile projects is to continuously improve, not only the product, but the way that the team works on the product. A board can help the team visually identify problems or bottlenecks with the process. This highlights areas for the team to focus on improving.

Topics

Visualizing work

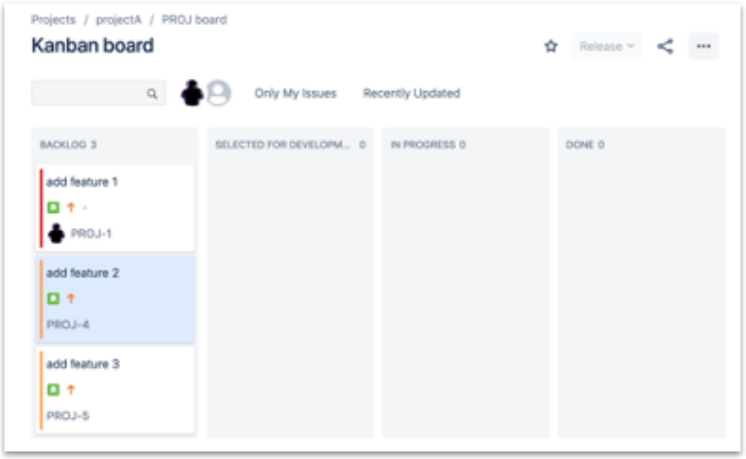
Workflows

Jira boards and workflows

Configuring board columns

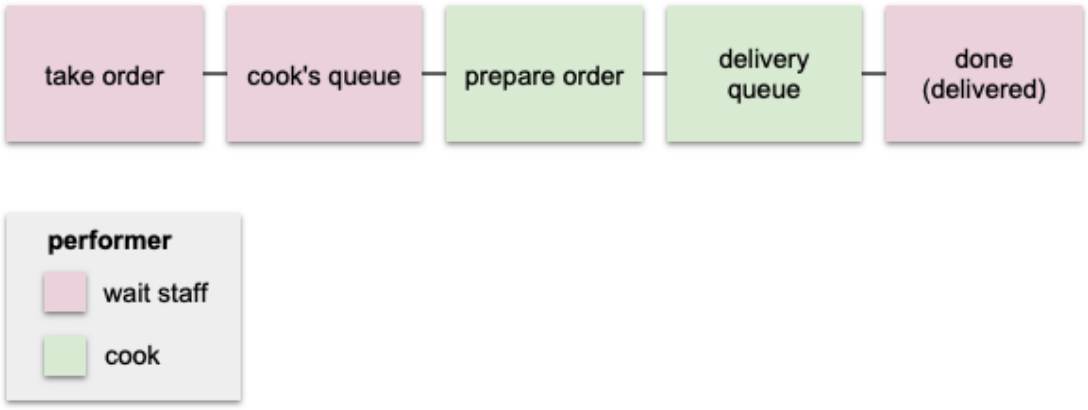
Workflows

- The set of columns of a board represent a *workflow* (or *process*) for completing the work of an issue
- Workflows are broken down into *statuses* (or *steps*)



The set of columns of a board represent a workflow for completing an issue.
Workflows are used to model the processes involved in the project. Even though a workflow technically can be considered a model of a process, you often will find the terms workflow, process, business process and value stream used interchangeably. They all represent breaking down the work into a series of steps.
Each column in a board usually represents a single step in the workflow. These steps may also be called statuses, states or stages.
You can see that workflows and boards are closely related. The board visualizes the workflow.
The board shown here has four statuses in the workflow. These statuses happen to have the same name as the columns in the board. The statuses are backlog, selected for development, in progress and done.

Example workflow: restaurant order and delivery



Let's take a simple process and model it as a workflow. We will model the process of ordering and delivering a customer's food at a restaurant.
The first step of the workflow is for the wait staff to take the order from the customer.
In the second step, the wait staff adds the order to the cook's queue. It's placed in a queue because the cook only can start the order when they have bandwidth.
In the third step, the cook takes the order from the incoming queue and begins to prepare the order.
In the fourth step, the cook has finished the preparation and adds the order to the delivery queue. It's placed in a queue because the cook usually doesn't directly deliver the order to the customer and the wait staff might not be immediately available.
In the final step of the workflow, the wait staff delivers the order to the customer.
Even though this is a simplified example of a workflow, the same basic idea of breaking down the work of the project into steps applies. One of the benefits of breaking down the workflow into steps is that the work of the steps can be done by different performers, allowing the process to scale to large teams.
You can model any process that you want using this simple block diagram approach. All it takes is a pencil and paper.

Boards vs. workflows

- A team works using a board
- The board's structure is defined by an underlying workflow

We can see that boards and workflows are closely related. A team works using a board. The board's structure is defined by an underlying workflow.
This example shows the restaurant workflow that we just defined. Below that, we have created a Jira project with columns of the board matching the workflow. We will discuss configuring boards later.

Topics

Visualizing work

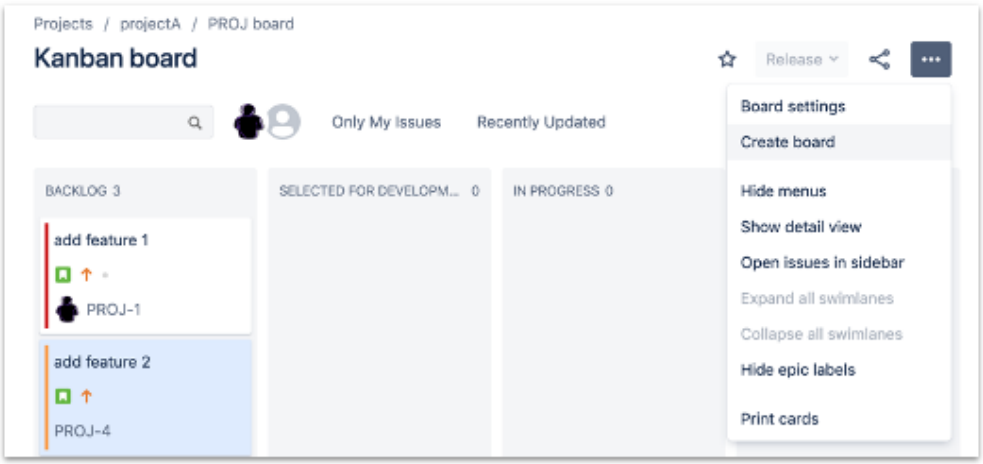
Workflows

Jira boards and workflows

Configuring board columns

How are boards created?

- Automatically
- Create additional boards at any time



Boards are automatically created when you create a project using kanban or scrum templates. In this example, we have created a kanban project.
You can create additional boards at any time. A single project can have multiple boards, and a single board can contain the issues of multiple projects. (Currently, you can not create extra boards with Cloud next-gen projects.)

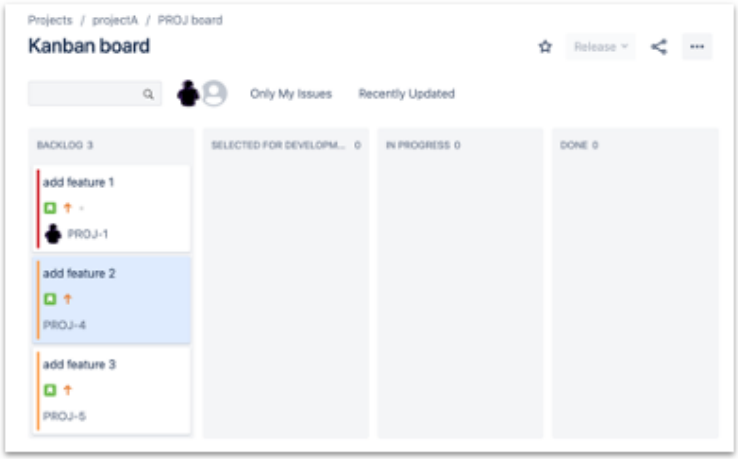
An issue's status field

- Every project automatically has an associated workflow
- The *status* field for each issue must be set to one of the workflow's statuses

In Jira, every project that is created automatically has an associated workflow.
The status field for each issue must be set to one of the workflow's statuses.
In this example, we are viewing the details of an issue with an issue key of PROJ-1.
You can see that there is a Status field. This issue's status is currently set to Backlog, and the dropdown box shows that there are three other statuses that we can set for this issue.

Boards and status

- Boards are a view of issues arranged by *status*
- Moving an issue changes the value of its *status* field



We are now in a better position to understand boards. Boards are a view of issues arranged by status. When you move an issue to a different column in the board, you are changing the value of its status field.

Changing an issue's status: status field value

Another way to change an issue's status is in the issue details. You can change the status using the Status dropdown box. This will move the item to a new location on the board.

Topics

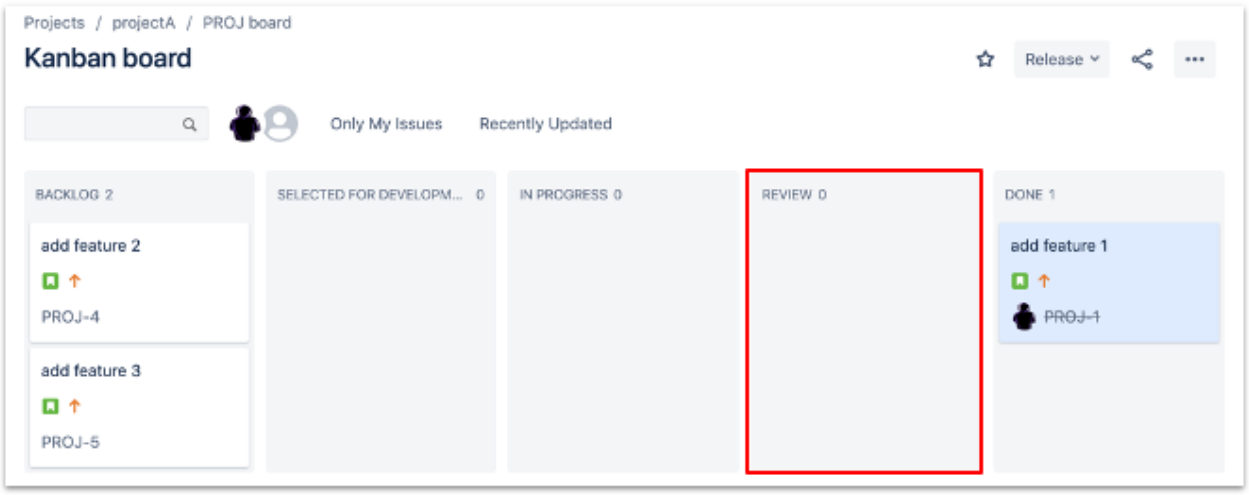
Visualizing work

Workflows

Jira boards and workflows

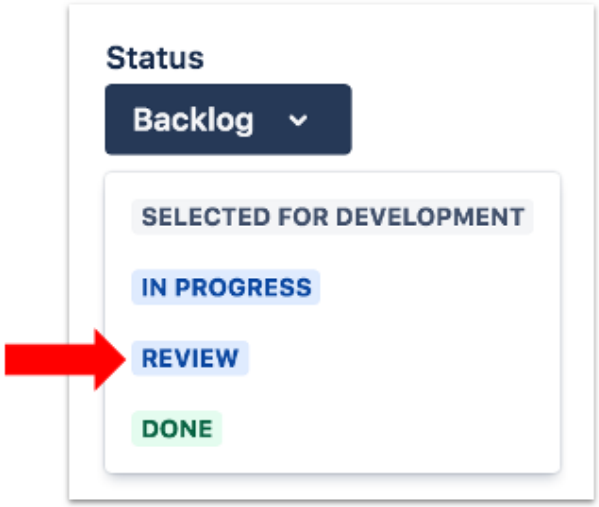
Configuring board columns

Adding a board column



Here we have added a Review column (and its related status) to the board. Issues can now be moved to this column, exactly like moving them to the other columns.

Viewing the new status



If we view an issue's details, we can select that the Review status is now selectable.

Takeaways

- A project board is a two-dimensional way to visualize the work of a team
- In Jira, a workflow is often represented using a project board
- Project board columns usually map to the *status* field of issues
- Board columns can be added or removed to match the team's desired process

Tasks

Visualizing Work Using Project Boards

- Move issues through a workflow
- Assign an issue
- Add a Review column to the board
- Explore the difference between Jira project administrators and standard users

4 Enrich Issues

What will you learn?

- Identify ways that issues can be enriched with information
- Describe the benefits of using issue types
- Describe subtasks
- Use labels to organize issues
- Introduce integration with version control and build systems

Topics

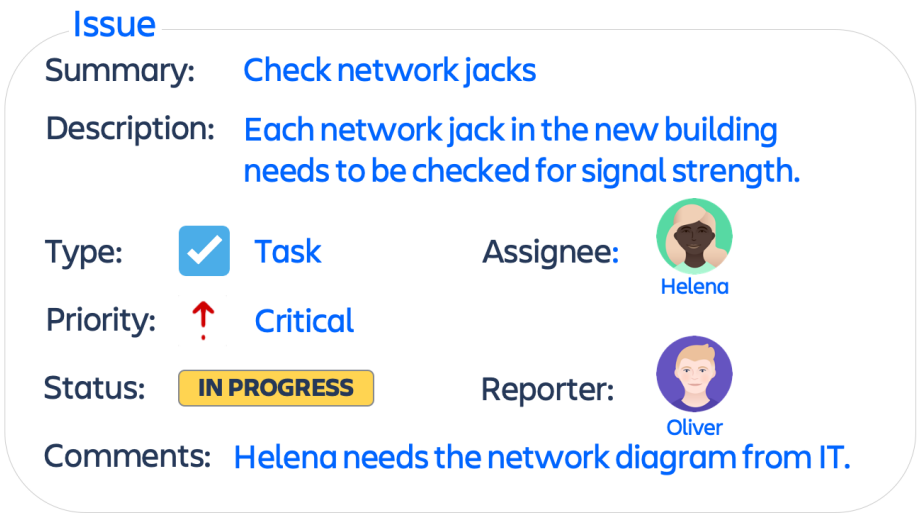
Enriching issues

Issue types

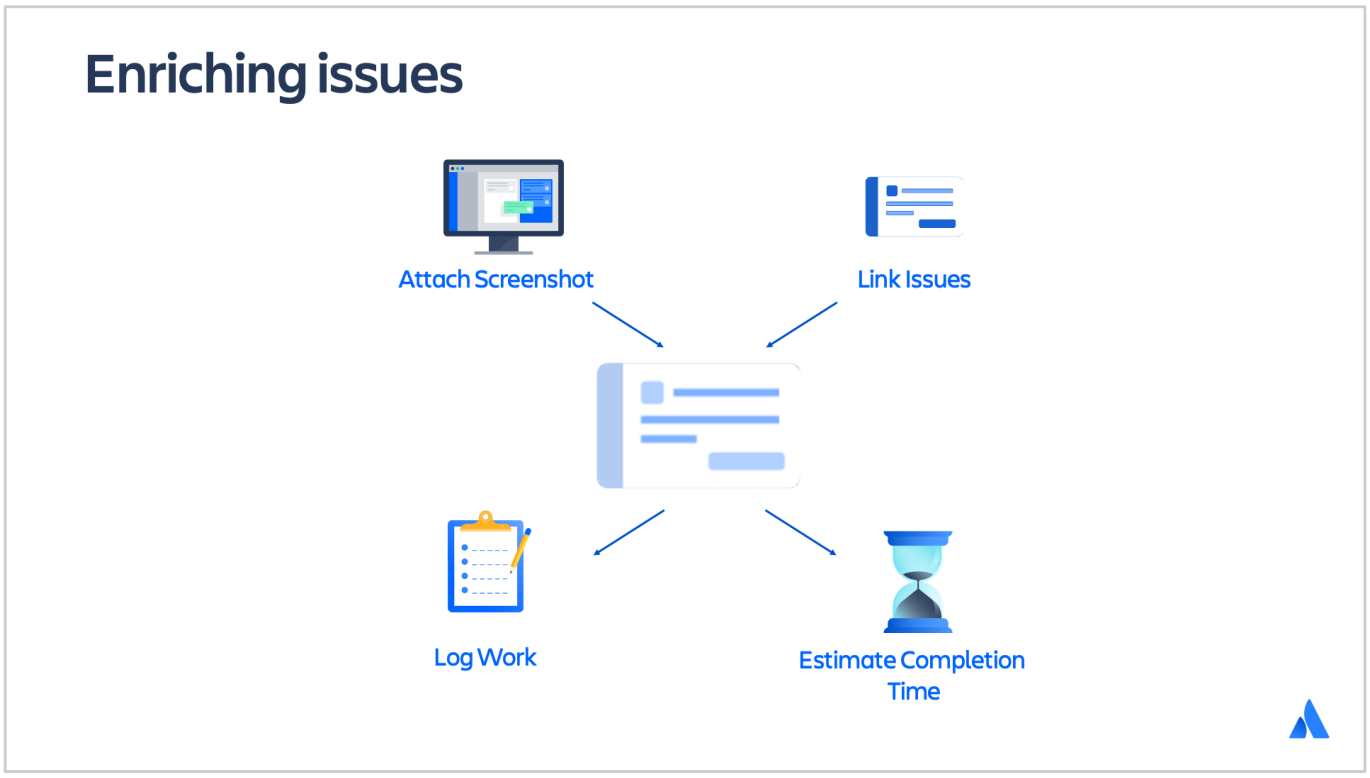
Labels

Developer integration overview

Issues contain work-related information



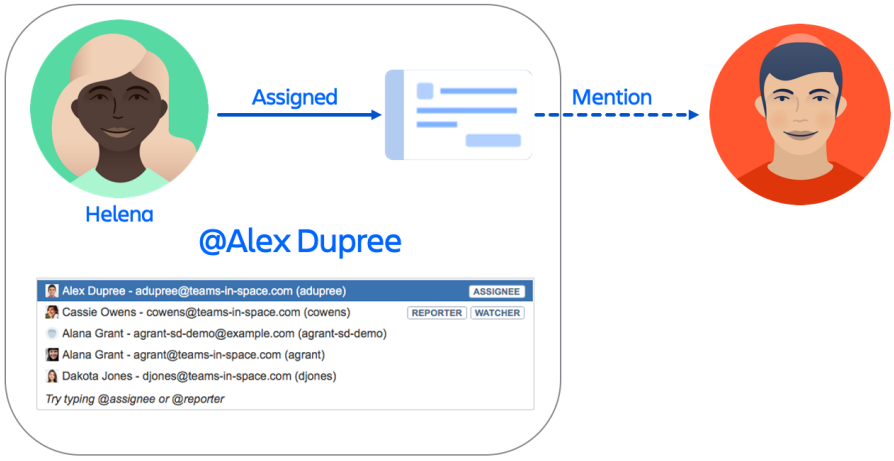
An issue in Jira can contain all the information about that particular work item in one place. This information is stored in fields.
The assignee is the person currently assigned to work on the issue. This often changes throughout the life of the issue.
The reporter is usually the person who created the issue. An issue will only have one reporter and this rarely changes.
What fields you see in an issue depends on the type of issue and how your project is configured. You can also create custom fields to match your team's desired process.



Here are some other common actions you can perform to enrich issues:

- You can add **attachments** to issues, such as a **screenshot** showing an error or an **invoice** file. This keeps all the important information easily accessible for the issue.
- You can **link** an issue to another one. For example, a problem request in an IT Service Desk can be linked to the bug that caused it in Jira Software. By linking these two issues, the support engineer can easily navigate to the bug issue to see what's being done. Also, the problem request will automatically be updated when the bug is fixed.
- When you create an issue, you can estimate the **time** needed to complete it.
- As you work on an issue, you can **log** the **time** and the actual **work** performed on the issue. This is important for project planning, tracking and reporting.

Mention team members



You can “at mention” (@mention) team members in an issue, and they will be notified that they have been at mentioned.
By simply entering the @ character, a list of available team members will appear for you to select. Jira will then notify that team member that they have been mentioned in a particular issue.
Helena is assigned to add some new artwork to the company website. She needs to get the files from Alex. Helena enters @Al and Alex’s name appears in the list for Helena to select. Helena has now tagged Alex to the issue by using an at mention.

Topics

Enriching issues

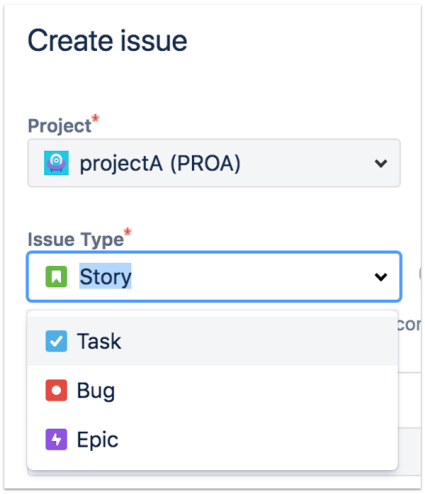
Issue types

Labels

Developer integration overview

The issue type field

- **Epic**- a big issue that can contain issues
- **Story**- requirement from the user's perspective
- **Task**- team work item
- **Bug**- a flaw that needs to be fixed
- **Subtask**- a child of another issue



A project can also use custom issue types

In Jira, an issue is a generic name for a unit of work. On your projects, there usually are different types of units of work. The issue type field is used to differentiate these different types of units of work. When you create an issue, you can choose the issue type. You can also change the issue type after creating the issue. Notice that each type has an associated icon to help easily identify the issue type.

A story is a requirement from the user's perspective.

A task is a work item that needs to be done by the team, but is not directly tied to a user requirement. An example might be upgrading the version of a product used by the team.

A bug is flaw that needs to be fixed in the product. It can be tracked with its own issue type to differentiate this work from other types of work.

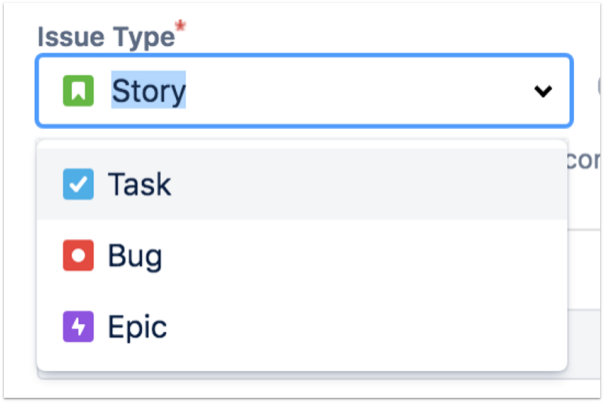
An epic is a big issue that can contain other issues.

A subtask is a child of another issue. It is used to break an issue down into specific pieces of work. When creating an issue, subtask is not shown in the issue type dropdown because subtasks must have a parent issue- they can not be created independently.

In addition to these out of the box issue types, you can create custom issue types. This provides your team the flexibility to work the way that they want to work. How the team actually defines what each issue type means is entirely up to them. Also, different projects can use different issue types.

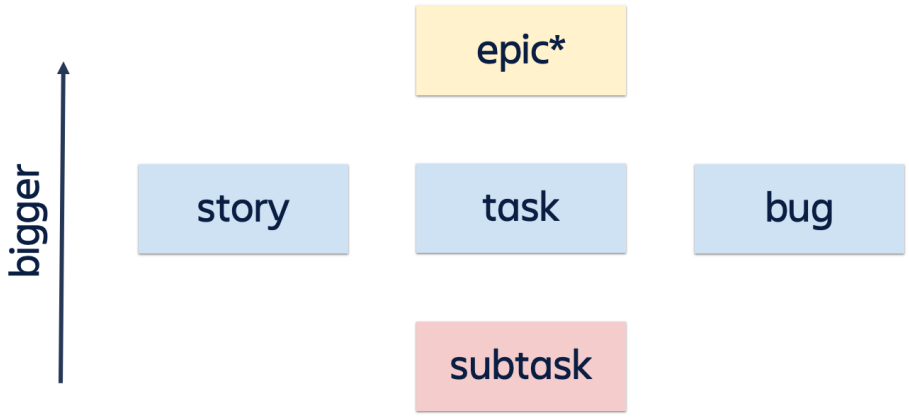
Why issue types?

- Supports different types of work
- Each type can have different fields, screens and workflows
- Can report on types separately



There are several reasons to use issue types with your projects.
Issue types support different types of work items. A team usually doesn't have only one type of work, and issue types allow the team to differentiate those types of work.
Each type can have different fields, screens and workflows. For example, you might want bugs to appear at the top of the project's board.
You can also report on types separately. For example, because the issues of the project have been categorized by issue type, you can easily create a report with the number of bugs fixed in the previous week.

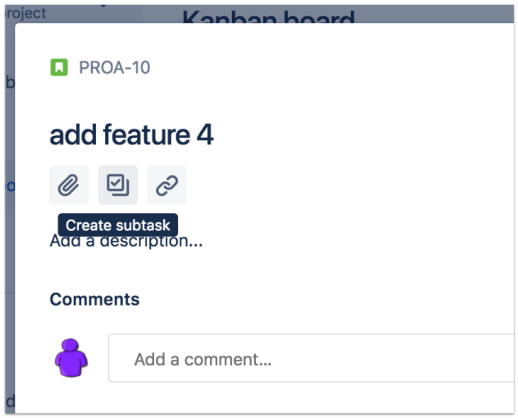
Jira's issue type hierarchy



** Epics are discussed later*

Subtasks

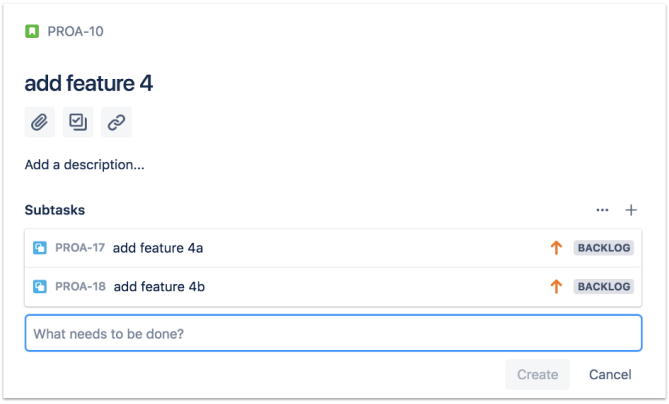
- An issue type that must have a parent issue
- Allow and issue to be broken down into individually manageable tasks
- Can be more technical than the parent issue



Subtasks are an issue type that must have a parent type.
Subtasks allow an issue to be broken down into individually manageable tasks. They can be assigned to different team members.
Subtasks can be more technical than the parent issue. For example, if the parent issue is a story, the story will be written in non-technical language that all team members and stakeholders understand, but the subtasks can be written for the technical person implementing the subtask.

Subtask characteristics

- Have their own issue key and field values
- Have independent workflow status



Subtasks have their own issue keys and fields. Here is a story with two subtasks. Each subtask has its own issue key and summary field value.
The subtasks have independent workflow statuses and move through boards independently.

Topics

Enriching issues

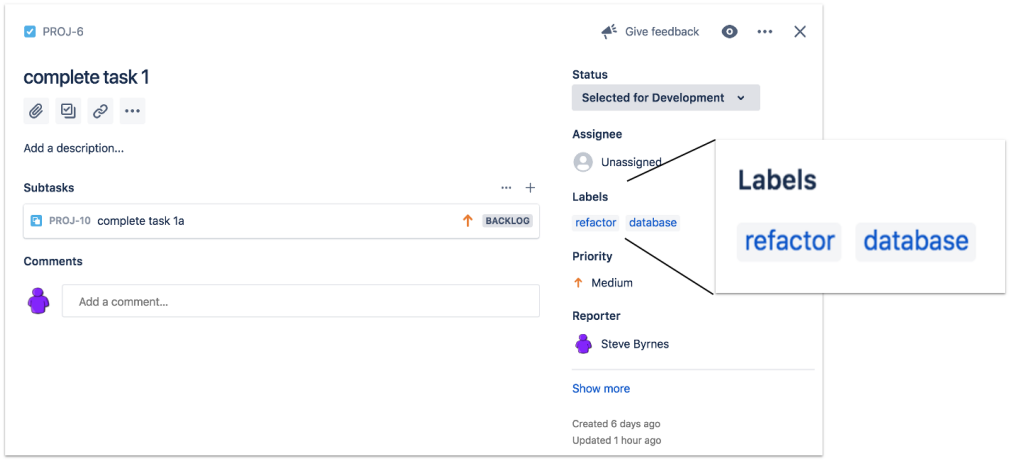
Issue types

Labels

Developer integration overview

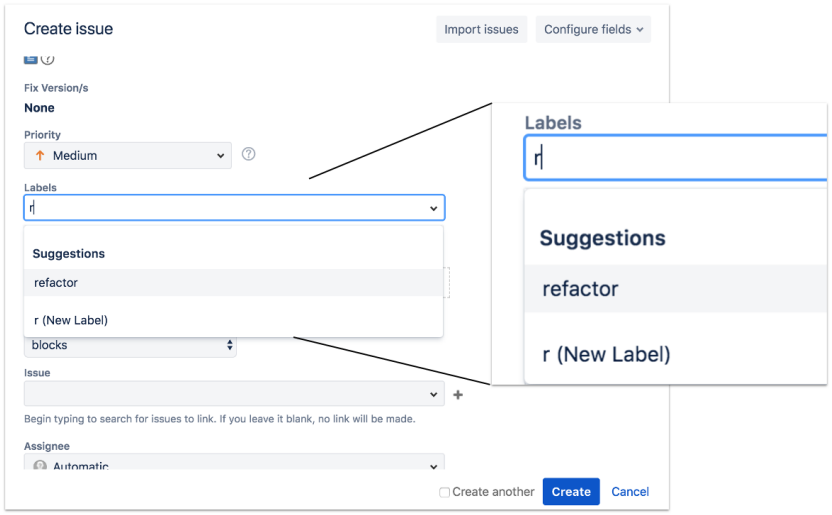
Labels

A field used to categorize and search for issues



Labels are a field used to help categorize and search for issues in any way that makes sense to the team. You can have multiple labels per issue. This issue has two labels- refactor and database.

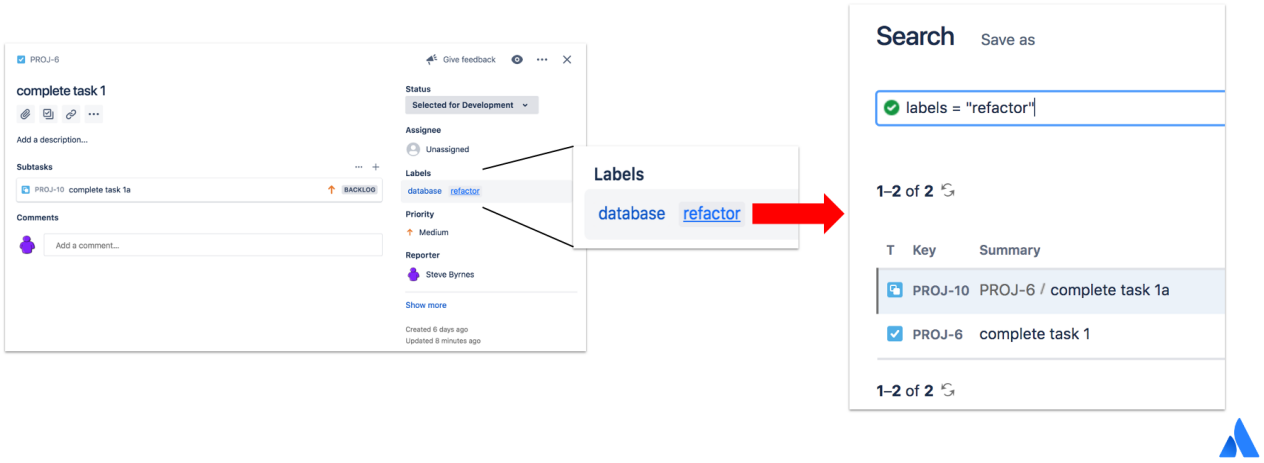
Adding or creating a label



Labels can be added as you create an issue or can be added to existing issues. Jira will suggest existing labels as you type. In this example, in the create issue screen, we typed an "r", and we see that we can select the refactor label. We could also type a new name and the label will be created.

Searching for labeled issues

Click on a label to search for all issues with this label



You can click on a label in an issue, and Jira will take you to a search page containing all issues with this label. In this case, two issues use the “refactor” label. You can also search for issues with certain labels using basic search or JQL, which we will discuss later.

Topics

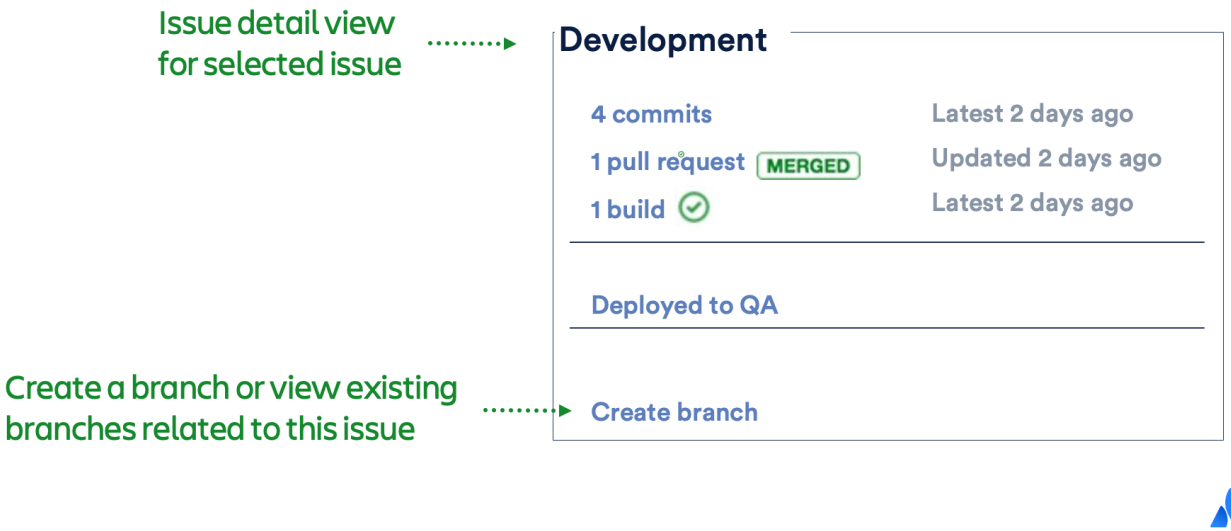
Enriching issues

Issue types

Labels

Developer integration overview

The issue detail development panel



When Jira Software is integrated with development tools, the development panel is displayed with the issue details.
In the development panel, we see that there are four commits related to this issue, one pull request that's been successfully merged, and one successful build. We also see that this issue was included in a deployment to QA.

Integration works through the issue key

Using Commit Message
Include issue key in commit message
“Initial commit – TIS-498”

Using Branch Name
Include issue key in branch name
“Feature branch TIS-498”

For Pull Requests
Include issue key in pull request title, or Jira can use issue key from associated commit or branch

For Builds and Deployments
Jira uses the issue key associated with a commit in the build

The integration is enabled through a reference to the issue key. This reference can be in a commit message or a branch name.
For pull requests, the issue key can be used in the pull request title, or the pull request can just get the reference from a commit or branch associated with the request.
Builds and deployments use the issue key reference from an associated commit.

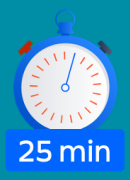
Takeaways

- An issue contains a diverse set of fields that are used to add information to the issue
- Issues can facilitate team communication with comments and @mentions
- Issue types can have unique fields, screens and workflows
- Subtasks are children of another issue type
- Subtasks have their own issue key and field values
- Labels can be used to categorize and search for issues
- Jira can be integrated with version control and/or build systems to improve developer-related communication

Tasks

Enrich Issues

- Add information to an issue
- Use team-related issue features
- Create issue of different types
- Create subtasks
- Add labels to issues




5
Kanban Method

What will you learn?

- Describe the kanban method
- Describe the importance of flow
- Identify the purpose of work-in-progress limits
- Differentiate pull vs. push processes
- Identify reasons to separate the backlog from the board

Topics

Kanban method overview

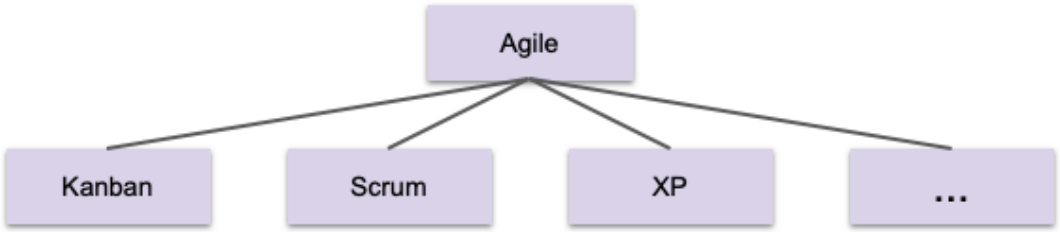
Improving flow

Pull vs. push

Separate backlog

Agile methods

- Agile is a way of thinking (mindset) and working
- An agile method (or framework) is an approach to implementing agile
- Common agile methods include kanban and scrum
  - Each embody core principles of agile
  - These are often combined



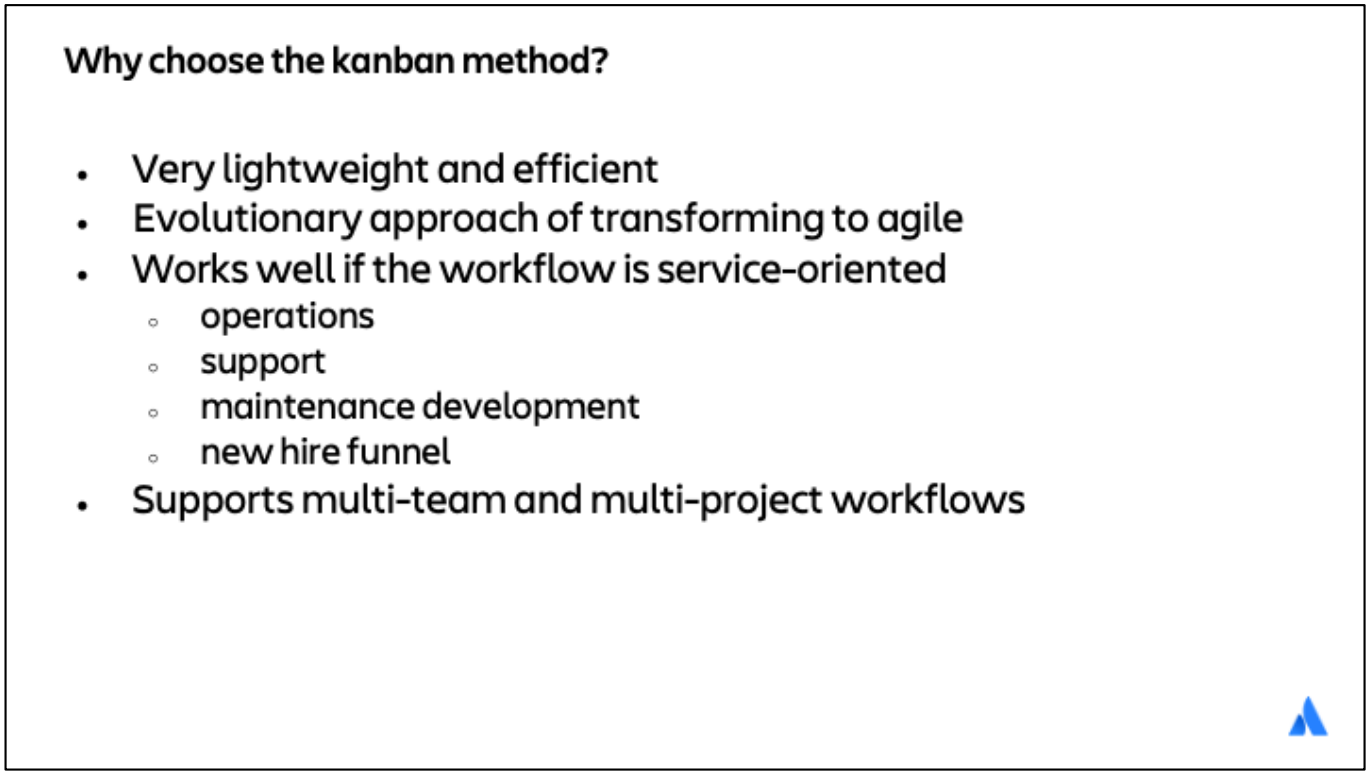
Agile methods are approaches to achieving agility. The methods are also sometimes called frameworks or methodologies. Agile on its own is more of a mindset than an actual method for managing projects. Agile methods add structure to agile ideas. Common agile methods are kanban and scrum. There are others as well, such as XP, or extreme programming.
Even though the methods are different, they each embody the core principles of agile. Example core principles of agile are empower the team, focus on continuous improvement, work in small batches and deliver increments of value.
The ideas from different methods are often combined by teams to create a custom method for each team.

What is the kanban method?

- An agile method used to manage a continuous queue of work items
- Commonly used ideas:
  - Visualize work
  - Remove process bottlenecks to improve "flow" of value
  - Limit work in progress / small batch size
  - Pull work rather than push (where it makes sense)
  - Continuously prioritize work items

https://en.wikipedia.org/wiki/Kanban_(development)

The kanban method is an agile method commonly used to manage a continuous queue of work items. The kanban method leverages many of the ideas of the Toyota Production System, which we will discuss later in the course.
Some of the ideas used in the kanban method include:
Limit work in progress. At any given time, the team should only be working on the amount of work that it can handle sustainably. Limiting work in progress results in small batch sizes.
Remove bottlenecks to improve flow. Ideally you would like to have the team outputting a steady stream, or flow of work. The reality is that there will be bottlenecks in the process. Issues will become stuck in certain places and may pile up in other places. These can be because of a problem with the process itself, or just the reality of the complexity of the issues. Either way, the team works together to remove these bottlenecks once they are identified. Attempting to remove bottlenecks forces the team to "see the whole" process rather than just a portion of it. When removing the bottleneck, it is best to find the root cause and fix the process there.
Pull work rather than push, where it makes sense. For many steps of a process, it is better for someone working on the next step to pull the work from the previous step rather than having it pushed on them when the previous step is finished. We will discuss this idea more a little later.



Why would a team choose the kanban method?
The kanban method is a very lightweight and efficient agile method. All agile methods by definition are lightweight, but kanban is lighter compared to the others like scrum. You can think of the kanban method as a bare-bones way to achieve agility. This means that it is quite easy to understand and begin to use. Some teams find that they are more efficient using kanban than other agile methods.
The kanban method can also be used as an evolutionary approach of transforming to agile. You can use your existing team in their existing roles and begin to use the kanban method. Kanban doesn't need a reorganization, new types of meetings or new roles. You can start implementing it now. You can then continuously improve to become more agile over time.
The kanban method tends to work well if the workflow is service-oriented. Examples include some of the work of the operations team, support requests, maintenance development and a human resource department's new hire funnel. Anywhere where the is a continuous flow of issues can be a good candidate for the kanban method. This does not mean that kanban should not be used to develop products, it is used to develop products as well.
The kanban method supports multi-team and multi-project workflows. An issue can be moved among teams using a single board, or multiple boards that are specific to each team. Each team can accomplish the work of the issue in any way that they choose.

Topics

Kanban method overview

Improving flow

Pull vs. push

Separate backlog

Continuous flow of work items

Kanban boards focus on improving the flow of issues through the workflow. New issues are continuously added to the backlog and are always being prioritized, usually by the business team. Once the issues are ready to be worked on, they are pushed to the selected for development column and then flow through the statuses of the workflow. Work is continuously being finished before starting new work. When a team member is ready to work on a new issue, they simply choose the top issue under selected for development and drag, or pull, the issue to the in progress column. This ensures that the team is always working on the most important issue as defined by the team.

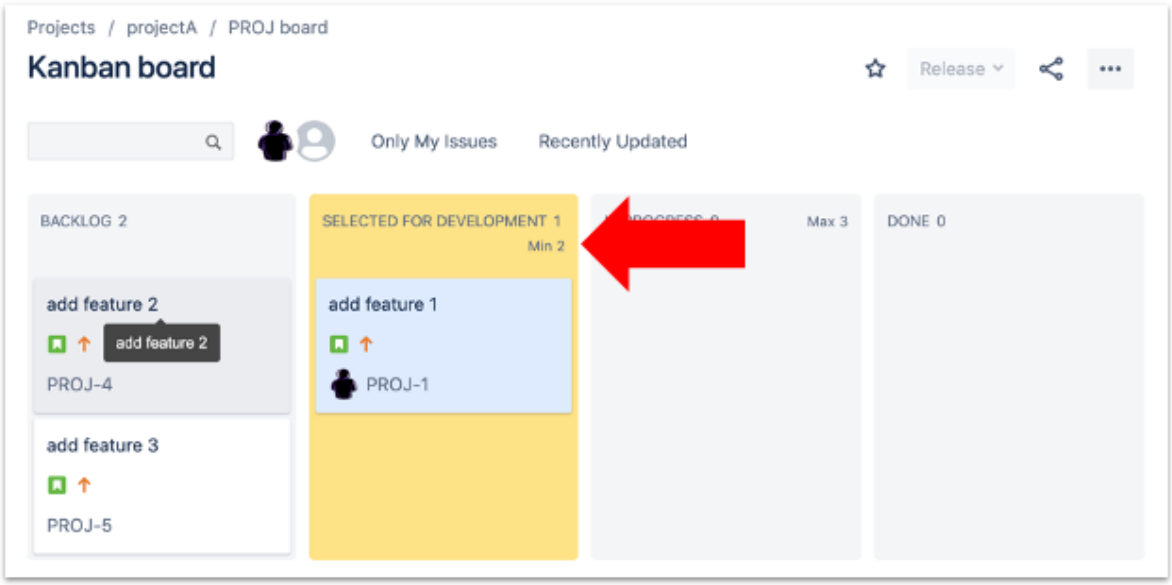
Improving flow- limit work in progress (WIP)

- How?
  - Specify the minimum and/or maximum number of issues allowed in certain project board columns
- Why?
  - Better flow
  - Limits waste
  - Promotes teamwork

https://www.atlassian.com/agile/kanban/wip-limits

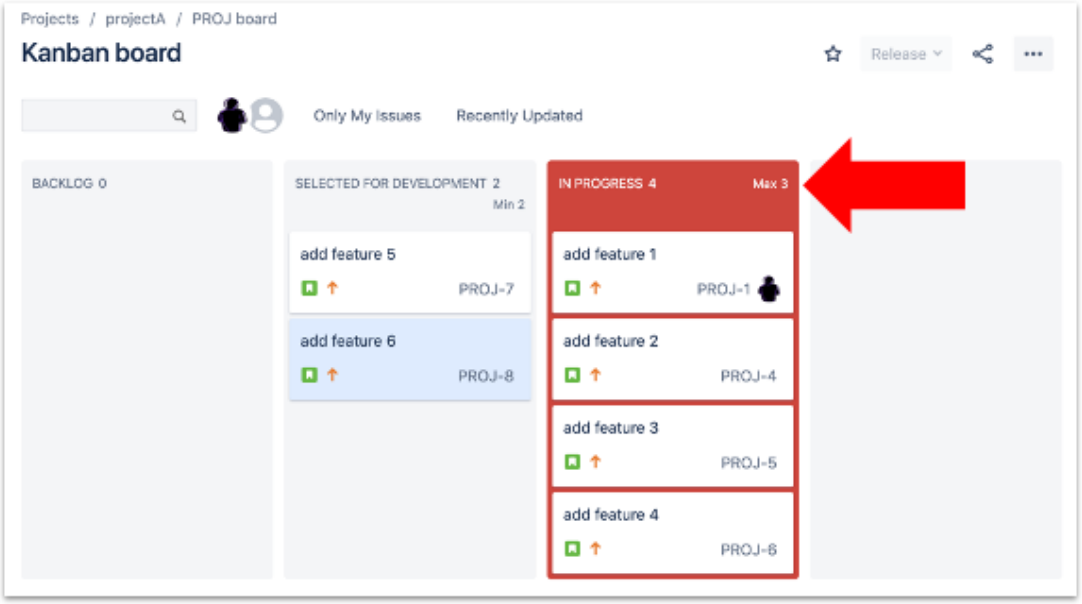
Limiting work in progress is done by specifying the minimum and/or maximum number of issues allowed in certain board columns. Limiting work in progress has many benefits.
If you limit work in progress, there are less issues being worked on at one time, which leads to better flow of work getting finished. Because of the work in progress limits, the team focuses on finishing the work in progress before starting new issues. This focus results in less multitasking, which is good because multitasking decreases productivity. The delivery of issues is faster with less work in progress because issues are not piling up in certain columns waiting for work to be restarted. Limiting the amount of work in progress means that issues are sure to achieve the "done" status. This means that the work is really done and can be provided to the customer. Moving issues into the "done" status is the primary measure of progress of the team. By limiting work in progress, any bottlenecks in the process are quickly identified, because there are relatively few issues in progress at any time and problems tend to be easily visible. Identifying and fixing bottlenecks is a way to continuously improve the work of the team.
Limiting work in progress also limits the amount of waste in the process. If there is an inventory buildup of issues in a status, there is waste because work on the issues is delayed. Limiting work in progress also limits the amount of rework that may need to be done if there is a problem caused in an early process step but identified in a later step. There are simply less issues to fix.
Limiting work in progress has the effect of promoting teamwork. Because the number of issues in a status is limited by the WIP limit, the team tends to work together to clear up any blockages.
The link at the bottom of the slide is to a good article on work in progress limits in case you are interested in more discussion on the topic.

Column under minimum limit



Here we have configured a minimum WIP limit. You can see that the selected for development column has a “Min 2” indication next to the column name. Since we only have one issue in this column, it is highlighted in yellow. This is a signal that notifies the team that issues should be added to the column. (Note that minimum WIP limits are currently not supported on Cloud next-gen boards.)

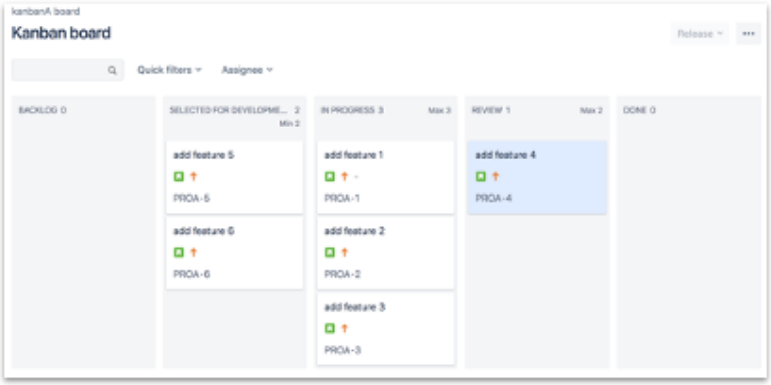
Column over maximum limit



Here we have configured the In Progress column with a maximum WIP limit. You can see the “Max 3” indication next to the column name. We have four issues with a status of In Progress, so the column has turned red. This is a signal to the team to finish the work in progress before starting more issues.
Notice that we don't need to put limits on the backlog or done columns, because those statuses do not contain the team's work in progress.

What should WIP limits be set to?

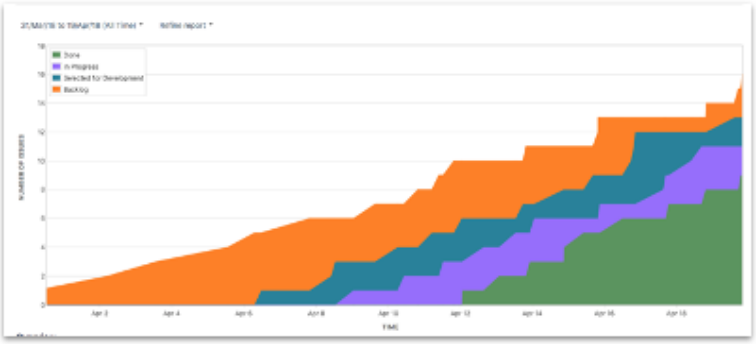
- Could start with no WIP limits
- Add WIP limits as the process shows problems
- Could set WIP limits to discourage multitasking
- Could set WIP limits on steps that the team neglects



What should work in progress limits be set to? The short answer is that this depends on the specific project and the team. Here are some ideas.
You could start with no work in progress limits and see if the issues are flowing nicely.
You could add limits as the process starts to show problems. For instance, if sometimes there are no items in the selected for development column, the flow is negatively affected because the team can finish an issue and have no issues to work on. You might want to set a minimum limit on the selected for development column to ensure that this doesn't happen.
You could also set work in progress limits to discourage multitasking. For example, if the team has seven members, you might want to set limits so that each member is working on only one issue at a time.
You could also set work in progress limits on steps that the team tends to neglect. For example, this workflow has a review status after the in progress status. This step ensures that other team members have looked at the work before it is considered done. If you find that issues are piling up in the review column, you could set a maximum limit on that column to ensure that the issues flow all of the way to the done column.

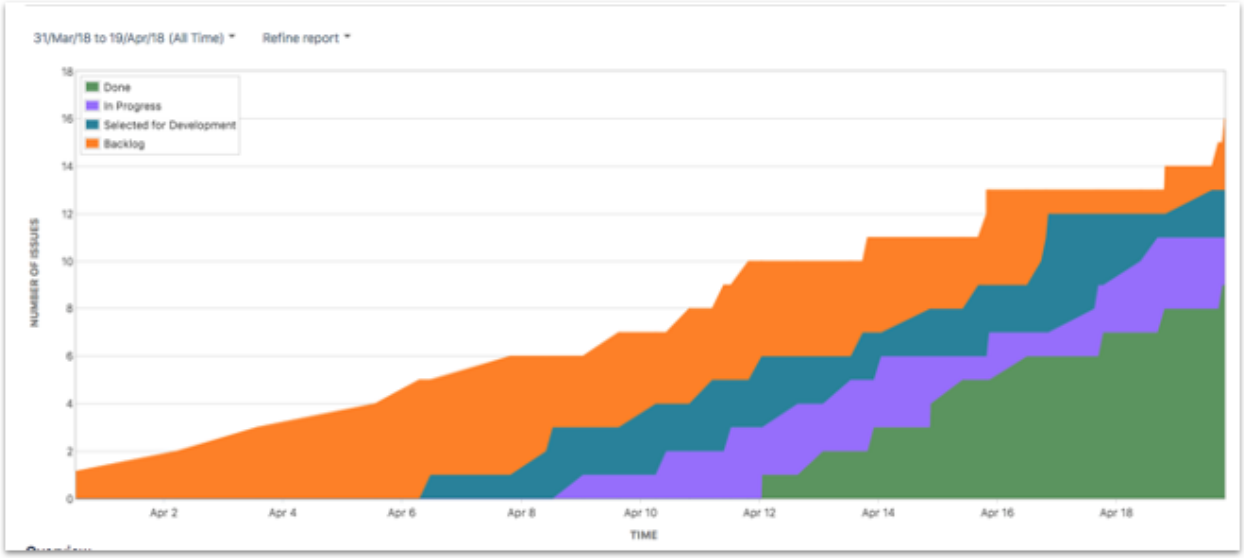
Why agile reports?

- Visualize the work
- Promote transparency
- Aid troubleshooting and continuous improvement
- Aid planning and estimating



Reports are a key tool related to being agile. Like with boards, agile reports help to visualize the work of the team. This promotes transparency of the project, allowing anyone with access to see and have a common understanding of the current state of the project. The transparency means that there should be no surprises about the delivery of the project.
Reports help with troubleshooting and continuously improving the project. For example, certain reports can clearly show process bottlenecks.
Reports also help with planning the work of the team. They are commonly used in planning meetings to help see the historic progress of the team and provide more accurate planning and estimating.

Cumulative flow diagram



A cumulative flow diagram is a popular report for kanban projects. This is because maintaining and improving the flow of the team is a high priority. Jira provides automatic, real-time reporting. As the team updates issues, the reports are automatically updated. All you have to do is view them.

A cumulative flow diagram shows the number of issues per status over time. You can see from this diagram that for the first few days, all of the issues were in the backlog status, indicated by the orange band. The number of issues in the backlog status increased from 1 to 5 before the first issue was moved to selected for development, which is the second band here. The third band represents issues with the in progress status, so you can see that actual work on an issue started here. The final band indicates issues with a status of done. You can see that the done band increases nicely over time, indicating pretty good flow. You can see that the backlog has almost been depleted, indicated by the shrinking orange band.

You can see many important kanban-related things from a cumulative flow diagram, including bottlenecks in the process, overall flow improvement and how well the team is keeping up with the backlog. You can't really see individual issue problems with this diagram- you can use other reports or views to focus on specific issues.

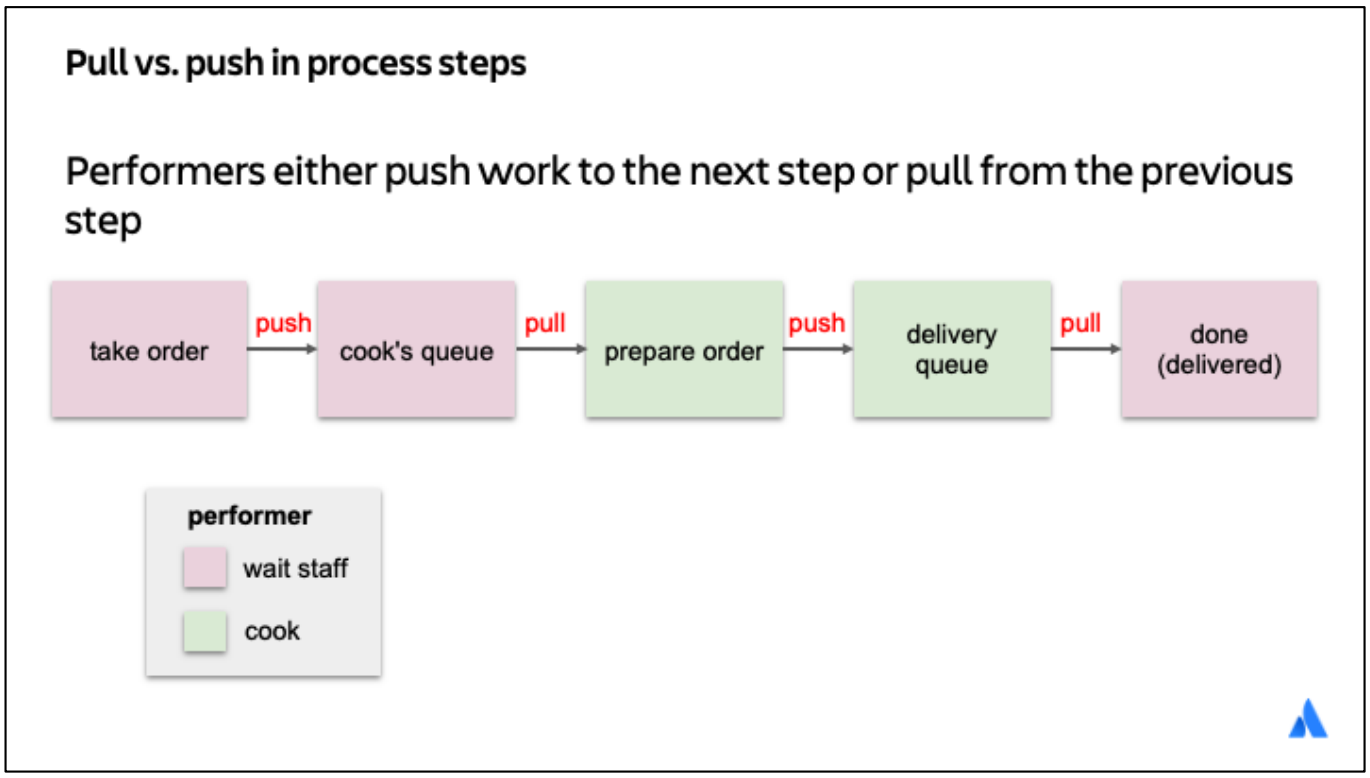
Topics

Kanban method overview

Improving flow

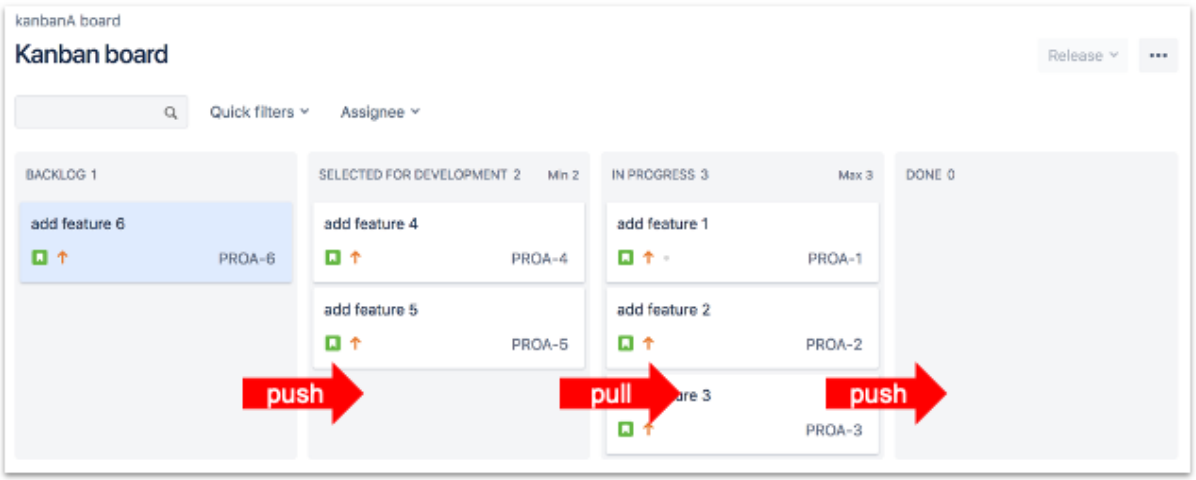
Pull vs. push

Separate backlog



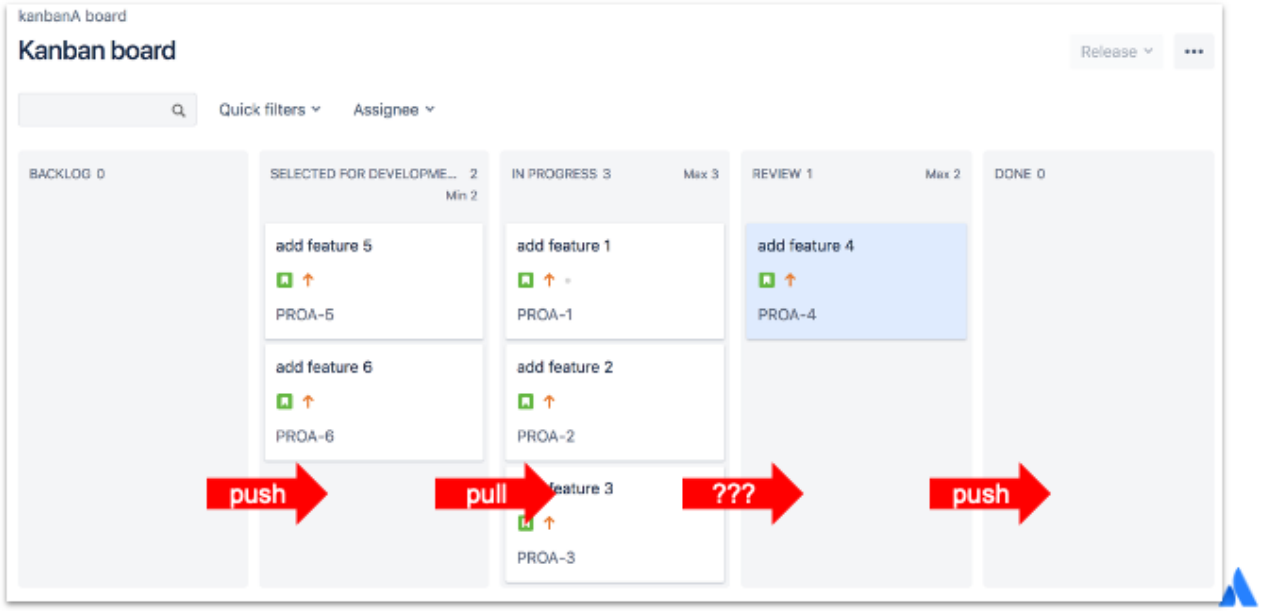
The performer of a step in a workflow either pulls the work from the previous step or has the work pushed to them. If we go back to our example workflow where a restaurant takes and delivers orders, we can see that sometimes the work is pushed and sometimes it is pulled.
After the wait staff takes the order, they usually push it to a queue for the cook to start when they get a chance.
When the cook has the availability to start preparing the order, the cook pulls the order from the queue and places it in the work in progress location.
After the food is prepared, the cook pushes the order to the wait staff's delivery queue.
When the wait staff has availability, they pull the order from the delivery queue and deliver it to the customer.
So, you can see that in this workflow, sometimes the performer of the work pulls the work from the previous step and other times the work is pushed to them from the previous step. Overall, this process is a pull process, because the cook only prepares the orders after they are received. If no customers come in, the cook never prepares any orders. If this was a push process, the cook would prepare the food ahead of time and wait for customers to come in and take the food. If no customers come in, the food could be wasted.

Pull vs. push



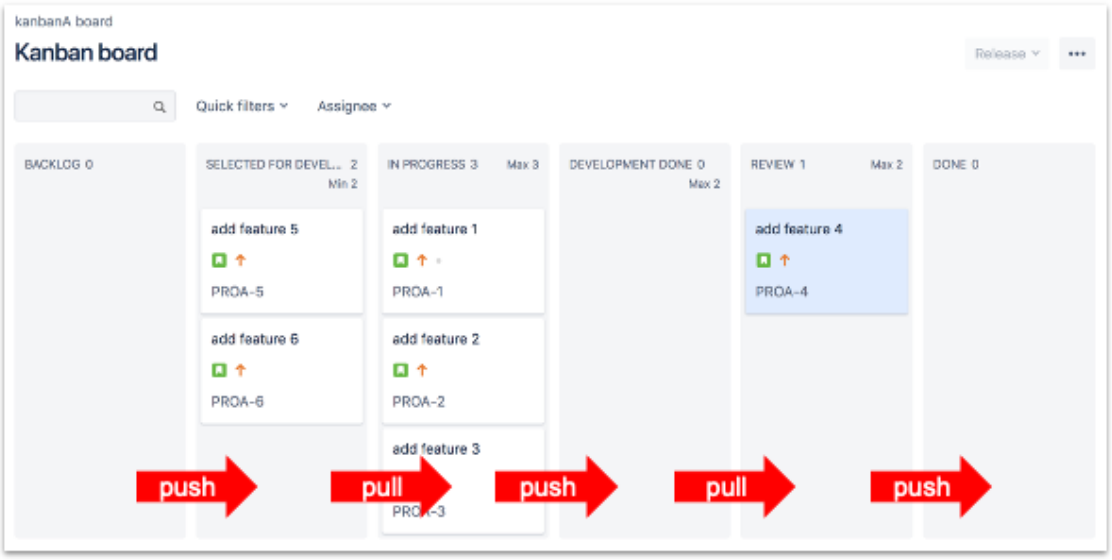
Pushing and pulling of issues can happen on kanban boards.
Typically an issue is pushed onto the selected for development column when it is ready to be worked on.
An issue in the selected for development column is only pulled into the in progress column when a person is ready to work on the issue. This is the main reason why the overall process is a pull system. The people doing the work pull issues only when they have availability.
When the work of the issue is complete, the person that did the work pushes the issue onto the done column.

Adding queues to enable pull (1 of 2)



Sometimes you might want to add queue columns to your workflow. For example, in this workflow we have a review column before the done column. An issue in the review column usually means that the work of reviewing is under way. However, when work in the In Progress column is complete, if you push it into the review column, a reviewer may not be available to work on it. If a reviewer pulls the work from the In progress column, the work of the issue might not be done. This is a good case for inserting a queue before the review column.

Adding queues to enable pull (2 of 2)



Here we have added a queue called development done to the workflow. When the work of an issue in the in progress status is done, that performer can push the issue to the development done queue. When a reviewer is available, they can pull the issue into the review column. Adding this queue makes the exact status of the work item easier to see.

Why pull?

- Empowers the team- team members select work, they are not assigned work
- Maintains a sustainable pace

We have seen that pulling work is common in the kanban method. We will also see later that most agile methods pull work, because the development team decides how many issues can be worked on and pulls an issue to start the work.
Pulling work is common on agile teams because it enables team members to select their work. This is what happens in self-organizing, empowered team. They are not assigned work.
Pulling also maintains a sustainable pace. A performer only pulls more work when they are ready, instead of work being pushed to them that they may or may not be able to keep up with.

Topics

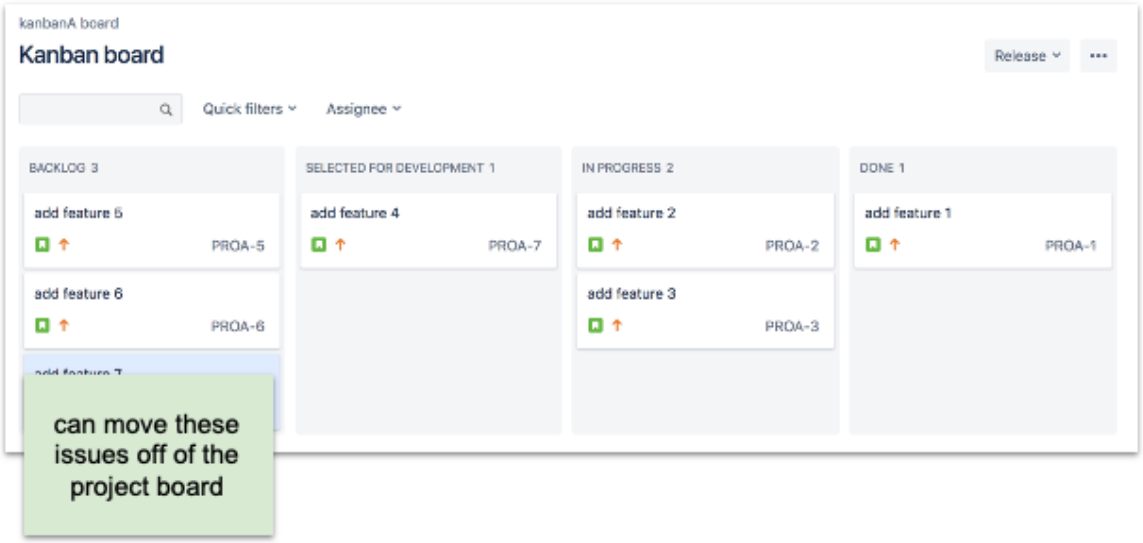
Kanban method overview

Improving flow

Pull vs. push

Separate backlog

Separate backlog



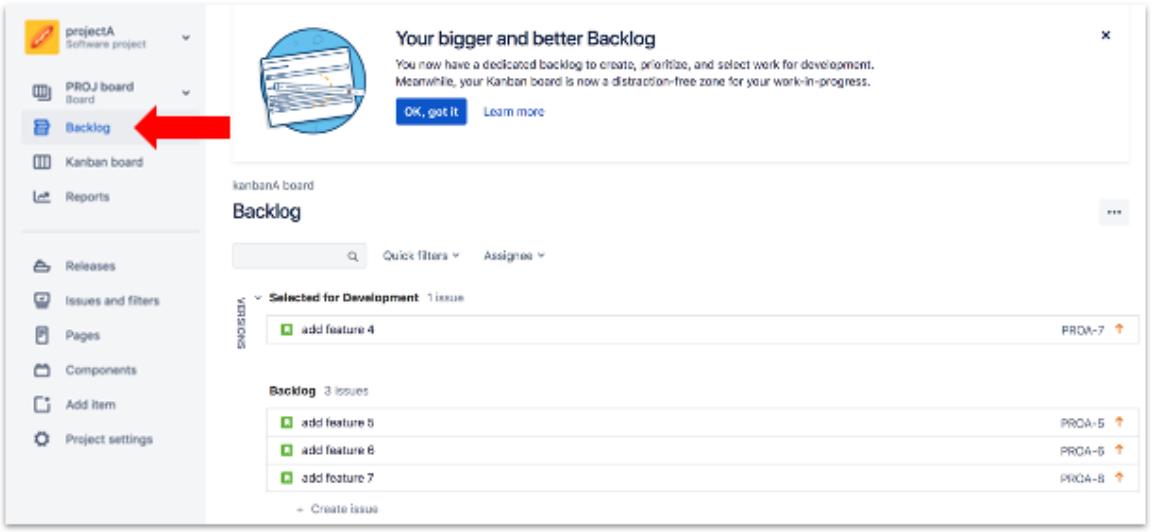
In Jira, the backlog column on a kanban board can be separated from the rest of the board. This has a few advantages. It allows the development team to only see and focus on issues that they can work on, because issues in the backlog column are not ready to be worked on. The backlog column also can become quite long and hard to manage in this format. A separate backlog column is easier to manage, and any work there is not visible to the rest of the team.

Why a separate backlog?

- Simplicity- separates the planning of issues from the project board
- The team can focus on work items that are ready to be worked on

Separating the backlog from the project board separates the planning of the backlog from the work of the team on the project board. Since the team only can work on issues once they are in the selected for development column, the backlog can be removed so that the board only shows issues that can be worked on.

Managing the separate backlog



When you click on the Backlog tab, you see the backlog. These are all of the issues with the status that was dragged to the kanban backlog when configuring the board. In this case, these are the issues with a status of backlog. Notice that the items in the backlog are in a list and you have a lot of room to edit the backlog.
Above the backlog is the first column of the board. This is there so that you can easily move issues from the backlog onto the board, allowing the team to see the issues. Also, if you would edit an issue here, changing the status to any column on the kanban board, the issue will move out of the backlog and onto the kanban board.

Takeaways

- Kanban is a lightweight agile method
- A project board should have a continuous flow of issues moving from backlog to done columns
- Work in progress limits can improve the flow of value by focusing the team
- In Jira, the backlog can be separated from the project board, simplifying the board and allowing separate backlog work

Tasks

Kanban Method

- ❑ Configure WIP limits
- ❑ View a cumulative flow diagram
- ❑ Configure a separate backlog


6
Lean and Agile Principles

What will you learn?

- Identify reasons the Toyota Production System is studied today
- Identify kanban objects
- Describe benefits of using kanban objects
- Identify kanban systems
- Describe lean principles
- Describe agile principles
- Compare lean and agile principles

Topics

Toyota Production System

Toyota kanban

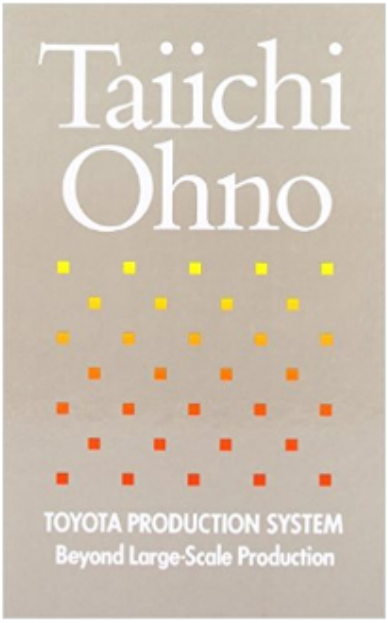
Lean principles

Agile Manifesto

Lean vs. agile

Toyota Production System

- Written in 1978, English translation in 1988
- Describes what is now sometimes called "lean thinking" or "lean management"
- Many agile principles are similar



In 1978, Taiichi Ohno, considered to be the father of the Toyota production system, wrote a book about the thinking behind the system.
In this book, Taiichi Ohno describes what is now sometimes called "lean thinking" or "lean management".
We will see that this book contains many of the principles that are used in agile projects.

Toyota simplified history

- "Catch up with America in three years."
- Focus was to eliminate waste and increase productivity
- Embraced ideas from Ford, but used a more "agile" approach

> "I would like to emphasize that (the Toyota Production System) was realized because there were always clear purposes and needs."
>
> Taiichi Ohno
> Toyota Production System: Beyond Large-Scale Production

After World War II, Toyota had a goal to "catch up with America in three years" in the automobile industry. Japan's automobile industry was very small at the time compared to American companies like Ford and General Motors.
The main focus of the Toyota production system is to eliminate waste and increase productivity. Taiichi Ohno had heard that American workers were something like nine times more productive than Japanese workers and decided that there must be a lot of wasted effort. The result was a decades long focus on eliminating waste and continuously improving. This eventually resulted in Toyota becoming the largest automobile company in the world. Whether or not the productivity numbers were true, that's what Toyota believed and helps explain their focus on eliminating waste.
Toyota studied and embraced many of the production ideas from Ford, but used a more "agile" approach to producing cars. After World War II, resources were very scarce in Japan, and Toyota simply couldn't afford to hold a lot of inventory as was done in traditional mass production. Out of need, Toyota developed a leaner, more flexible approach to mass producing cars. We will see that this lean approach uses many of the same principles that are used today in agile projects.
In his book, Taiichi Ohno emphasized the need for the team to have a clear purpose, as the quote above demonstrates.

Lean principles apply broadly

"I am confident (the Toyota Production System) will reveal its strength as a management system..."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production
preface to the English edition (1988)

Lean principles of the Toyota production system can be applied in many contexts. They do not apply only to lean manufacturing or production. In fact, Taiichi Ohno wrote about this in the preface to the English edition of his book, which came out in 1988.
He said "The Toyota production system is not just a production system. I am confident it will reveal its strength as a management system adapted to today's era of global markets and high-level computerized information systems."
Our discussion in this course focuses on lean principles that apply broadly.

Topics

Toyota Production System

Toyota kanban

Lean principles

Agile Manifesto

Lean vs. agile

What is a kanban?

- Kanban- an object that controls the flow of work
- The idea came to Toyota from supermarkets
  - Instead of push, order when inventory is low (pull)
  - This matches the supply and demand
  - An empty box is a "kanban"- a signal to order more

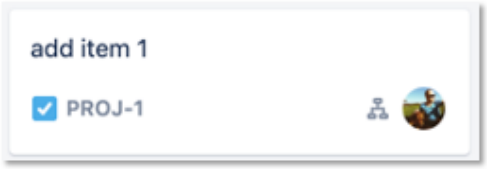
> "From the supermarket, we got the idea of viewing the earlier process in a production line as a kind of store."
>
> Taiichi Ohno
> Toyota Production System: Beyond Large-Scale Production

Kanban is a Japanese word used at Toyota to generically describe an object that controls the flow of work. The actual object can vary depending on the situation, but it usually holds some information related to the work. By naming this type of object a kanban, Toyota basically was adding a new definition to the word.
The idea came to Toyota from the way that supermarkets manage their inventory. Taiichi Ohno mentions in his book that he once came to the U.S. to visit car companies, but he was more excited about visiting and learning from the supermarkets.
In supermarkets, for some products, instead of forecasting and having a product delivered at regular intervals, which is a push system, more product is only ordered when the current inventory reaches a reorder point. This is a pull system.
This matches the supply and demand. For example, the supermarket doesn't need to guess how many cans of tomatoes to order and risk having too little or too much inventory. They can wait for one of the boxes to empty, then order another box.
In this example, an empty box is acting as a "kanban". It signals employees to order more product.
Taiichi Ohno said in his book, "From the supermarket, we got the idea of viewing the earlier process in a production line as a kind of store." When a later process used a component, such as an engine, it would provide a kanban to the earlier process. That kanban could be a piece of paper with information on it, or something like an empty cart. This would signal the earlier process to build another component. The earlier process only builds a component when it has been given a kanban. This is sometimes called a just-in-time system.

Other examples of kanbans

guest check

coffee cup



Jira issue

Even though the word kanban is used to represent work to do at Toyota, the concept has been around for a very long time.
A guest check at a restaurant is a kanban. The work preparing an order does not start until the guest check is filled out. If no guests come in, no orders are prepared. The information related to the order is on the guest check.
An empty coffee cup when ordering coffee can also act as a kanban. The size of the cup, your name and the desired ingredients provide the information that is needed to start and fill your order. This kanban also happens to be a convenient delivery vehicle for your order.
Another example of a kanban is a Jira issue. It provides information related to the work of the issue. Without the Jira issue in the correct column, the work shouldn't be started.
You can see that all of these kanbans represent and hold information related to work to be done.

Kanban systems

"The Toyota Production System is the production method and the kanban system is the way it is managed."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

Kanbans are not used in isolation. There is an associated system to them, which sets the rules to how the kanban are used.
Taiichi Ohno said that "The Toyota production system is the production method and the kanban system is the way that it is managed." He is pointing out that the kanbans are part of a kanban system.
Similarly, every restaurant uses guest check kanbans in slightly different ways, depending on the kanban system.
Also, agile projects that use issues as kanbans use them in unique ways.

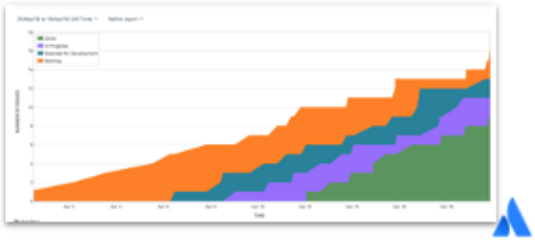
Benefits of kanban systems

- Visualizes work
- Simple
- Reliable
- Efficient
- Eliminates waste
- Identifies bottlenecks/easy to improve

There are many benefits to using kanban systems to manage work. Some of the benefits include:
They visualize the work. Kanbans are visual, allowing you to see very clearly what work needs to be done.
They are simple. Forecasting demand and maintaining inventories is usually a much more complicated approach to managing the work than using kanbans.
They are reliable. This reliability comes from the visual aspect and simplicity of the system. A simple, visual system is less likely to fail.
They are efficient. An effective kanban system uses resources only to do work that is of value to the customer.
They eliminate waste. Instead of holding and managing inventory, the supply is matched to the demand.
And finally, it is easy to identify bottlenecks or problems in the system and make adjustments to improve the workflow. This is mainly because the simplicity and just-in-time nature of the system make it easy to see where problems occur.

Kanban definitions

- Kanban token- An object that controls the flow of work
- Kanban system- A system that controls the flow of work using kanbans
- Kanban method- A lightweight agile method



The term kanban has a few meanings, depending on the context.
Kanban may refer to an object that controls the flow of work. We have seen that restaurant guest checks and Jira issues can be considered kanbans. We could call these kanban tokens to help clarify the meaning.
Kanban may also refer to the system that controls the flow of work using kanbans. We have seen that Toyota has a system for how it uses kanbans, just as restaurants have a system for how they use guest checks and how a team uses Jira issues. We could call this a kanban system to help clarify the meaning.
We have also seen that there is a lightweight agile method called kanban. It takes its name from the kanban system at Toyota, but applies it to agile projects. This is a bit confusing, because kanban tokens, kanban systems and even things like kanban boards are used in most agile methods, not only in the kanban method.

Topics

Toyota Production System

Toyota kanban

Lean principles

Agile Manifesto

Lean vs. agile

Why principles?

"With a better tool, we can get wonderful results. But if we use it incorrectly, the tool can make things worse."

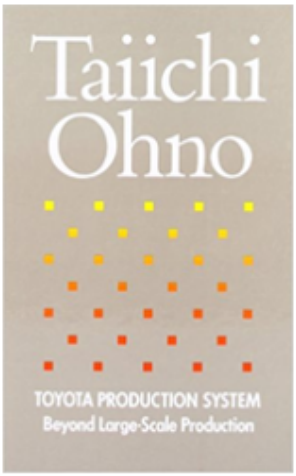
"We should not forget to always use the principles…"

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

We are going to take a look at lean principles because, in many ways, the principles are more important than the practices or tools. Principles tend to be foundational and long-lived, even as the practices and tools change.
Taiichi Ohno discusses the importance of understanding the principles in his Toyota Production System book, as the quotes above demonstrate.

Lean principles

1. Empower the team
2. Visualize work
3. Embrace the scientific method
4. Improve the "flow" of value
5. Build quality in



Here are some of the lean principles discussed in Taiichi Ohno's book. We will go through these one-at-a-time, and you will see that we have already discussed many of these earlier in the course.
This is a subjective list. You might read his book and come up with a different list. Toyota didn't invent many of these principles, but the combination of them can and have been used very effectively for many types of projects and businesses.

Empower the team

"…Operators acquire a **broad spectrum** of production skills… and **participate in** building up a total system in the production plant. In this way, the individual can **find value in working**."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

One of the key principles of lean is to empower and leverage the creativity and talent of the team. This is not only smart business, but improves overall team happiness.
These two sentences contain some key concepts related to the people on the team. First, they have a diverse set of skills, including an understanding of lean principles. This is different from a traditional mass production line, or a command-and-control led team, where team members tend to have specialized jobs. Acquiring this diverse set of skills means that the team members are always learning.
Taiichi Ohno also mentions that team members participate in building the system. This means that the team is trusted to make decisions, and even have the ability to stop the production line if necessary.
And finally, not only is empowering the team good for the success of the company, it improves team satisfaction. Many people want to feel like they are contributing to something of value, and empowering individuals provides that even more than monetary or recognition rewards.

Teamwork is a big part of empowering the team.
Taiichi Ohno thought that we could learn a lot from team sports, and he mentions in his book that "A championship team combines good teamwork with individual skill."

Visualize work

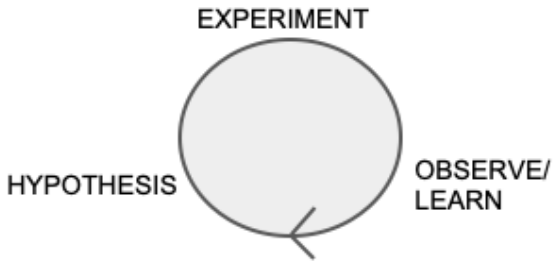
- Visual control- Toyota uses kanbans to signal and control the work
- Andon board- An information board that shows any existing problems

> "When one looks up, the andon (the line stop indication board) comes into view, showing the location and nature of trouble situations at a glance."
>
> Taiichi Ohno
> Toyota Production System: Beyond Large-Scale Production

A principle that we have already discussed is to visualize work. We have seen that the work of agile projects is usually visualized with boards, and the progress of the team is visualized with charts.
Toyota uses kanbans to signal and control the work. We have discussed kanbans and their benefits earlier.
Toyota also has something called an andon board. This is an information board that shows any existing problems.

Embrace the scientific method

1. Create a hypothesis
2. Build an experiment
3. Observe/learn from the results
4. Repeat/iterate

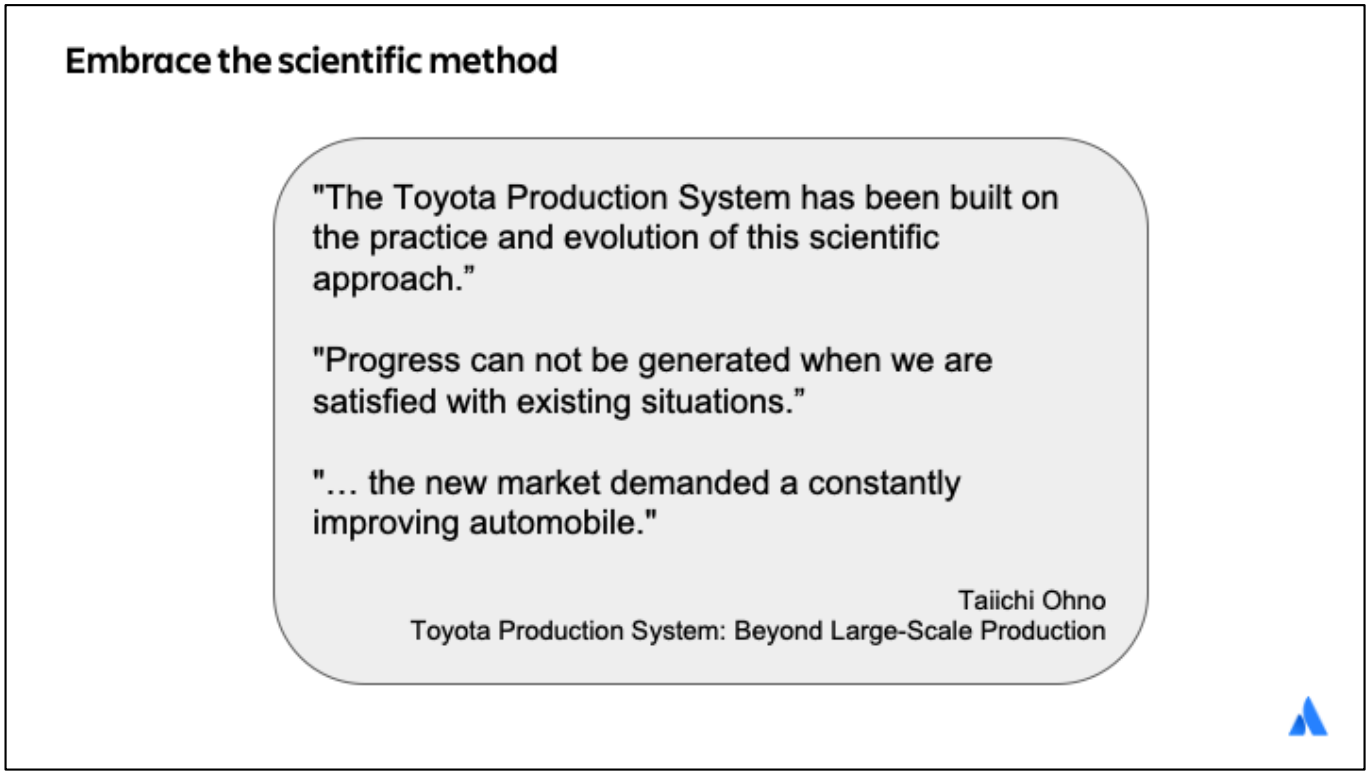


We have seen that managing and working on agile projects includes many small iterative learning loops. This resembles the scientific method, and in fact the scientific method can be thought of as the foundation of agile projects. Since most of us are at least somewhat familiar with the scientific method, we can use that knowledge to help understand agile projects.

The basic idea of the scientific method is to start with an idea or hypothesis, then build an experiment to test the idea, then observe the results of the experiment and learn from it. You can then repeat the process, using what you have learned to improve during the next iteration. It is a series of continuously learning and improving iterations, which could also be called loops or cycles. This is called an empirical approach to problem solving, where you are learning from the results of experiments.

In agile projects, we shift our thinking a little bit so that instead of using the scientific method only for discovery, we also use it to accomplish the work of a project. In addition to creating a hypothesis, we plan some work. We then do the work. We then learn from the feedback on both our work and the process that was used to do the work. We repeat the cycle in order to do more work, leveraging the knowledge that we have gained in the previous iterations.

The scientific method can be thought of as a formalized description of what the brain is doing when it is problem solving. Accomplishing the work of projects involves a lot of problem solving, so it is not surprising that agile approaches are based on the scientific method.



The scientific method, or similar experimental, evidence-based methods such as plan-do-check-adjust cycles, is used to continuously improve. We have already seen that the scientific method is the foundation of agile methods, and is also true with lean thinking.
Every organization should be a learning organization. Every team should be a learning team. Everyone should realize that, even if they are successful now, they have a lot to learn.
A big focus at Toyota has been to continuously improve. Here are is a quote from Taiichi Ohno's book that help demonstrate that.
"...the new market demanded a constantly improving automobile." If you think about it, many markets, such as markets for cell phones and software, assume constantly improving products.

Embrace the scientific method- embrace change

"As long as we can not accurately predict the future, our actions should change to suit changing situations. In industry, it is important to enable people to cope with change and think flexibly. "

"Build a fine-tuning mechanism into the business so that change will not be felt as change."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

A lean principle is to embrace change rather than create a fixed process and stick to it. Here are some quotes from Taiichi Ohno's book:
This second quote points out that in a lean or agile environment, change feels normal.

Improve the flow- limit work in progress / small batch size

"Reducing the number of kanban increases their sensitivity."

"People prefer working with large quantities. It is easier than having to work hard and learn from producing small quantities."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

Earlier, we have discussed work in progress limits and described how limiting the amount of current work can increase the flow of value and increase overall productivity.
Here are some quotes from Taiichi Ohno related to limiting work in progress.
If you limit the amount of work in progress, you are getting constant feedback, and you tend to see problems easier because they don't get lost in a sea of other work.
Some people like the comfort of not getting consistent feedback, because it is easier to just continue doing what you are doing.

Improve the flow- map the value stream

1. Draw the current state
2. Draw the desired future state
3. Iterate toward the future state

https://en.wikipedia.org/wiki/Value_stream_mapping

Value stream mapping is a tool used for continuous improvement. Toyota calls this "material and information flow mapping".
You draw a diagram with the steps involved in the process that you want to improve. These often are the steps in bringing a product to the customer. This is called the current state.
You then identify the desired future state. This is done by finding ways to increase value and reduce waste in the process. This often includes finding better ways to do things. The organization's vision and overall direction help to identify the desired future state.
You then iterate toward the future state. You may never fully reach the future state, but iterating towards it is a way to continuously improve. You can see that this is a learning loop that is similar to the scientific method. It is also similar to the agile approach of building the product incrementally. This simple exercise can be very good for planning and getting the team working in the same direction.
The Wikipedia link provides more information on value stream matching.

Improve the flow- pull work

"The conventional way was to supply materials from an earlier process to a later process. So, I tried thinking about the transfer of materials in the reverse direction. "

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

We have discussed the concept of pulling work from the previous step rather than having work pushed on you. This allows you to limit inventory and manage the flow of value.
Taiichi Ohno discusses this in the quote above. He was a big proponent of using this kind of unconventional thinking to help solve problems.

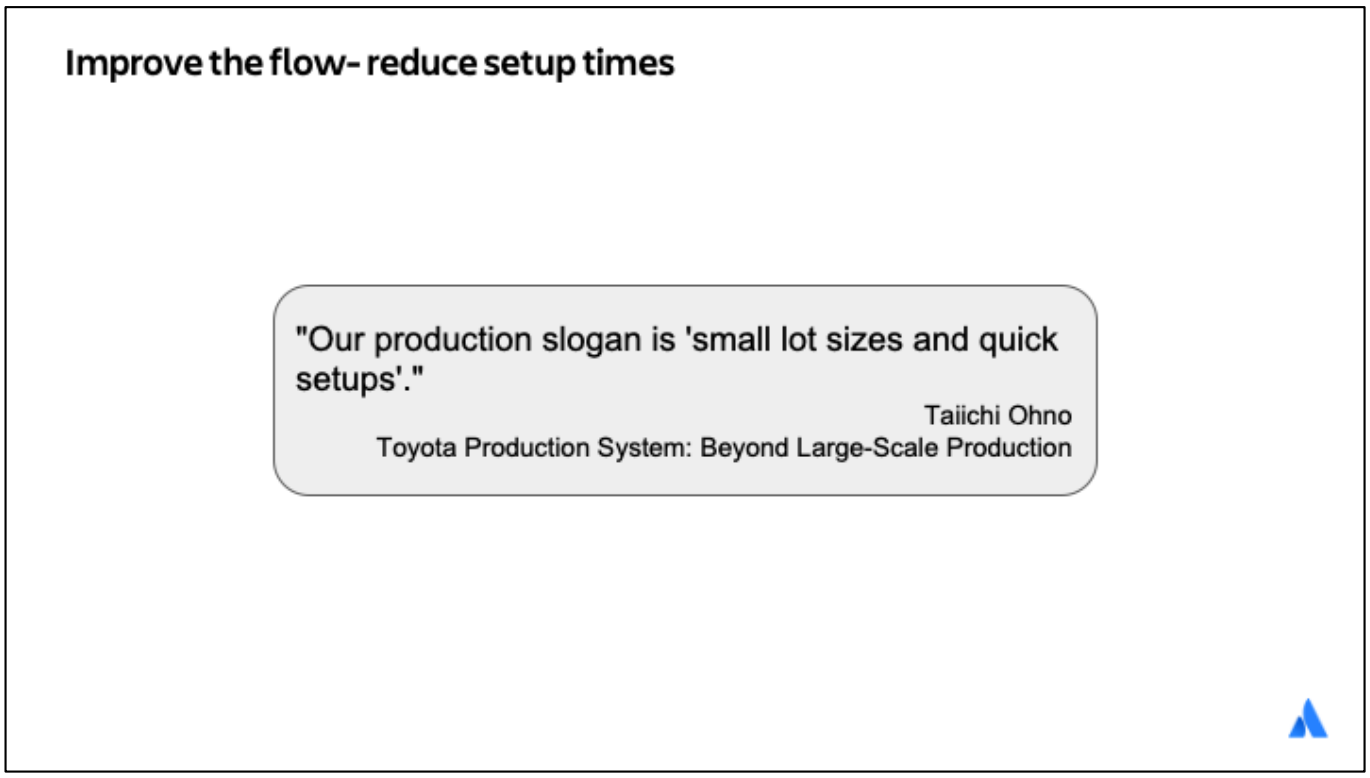
Improve the flow- eliminate waste

"The basis of the Toyota Production System is the absolute elimination of waste."

"The vicious cycle of waste generating waste hides everywhere…"

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

One of the most important principle in lean is to eliminate waste. Here are some quotes related to eliminating waste.
Taichi Ohno gives some examples of waste generating waste. For example, having too much inventory is a waste. It could cause the company to need a storage facility. This requires a system to manage the inventory. It also requires methods to make sure that the inventory does not degrade before it is used. While this is a physical example, this concept of waste generating waste is true with other types of projects. For example, having too many features planned in detail is a waste because some features may never be built.



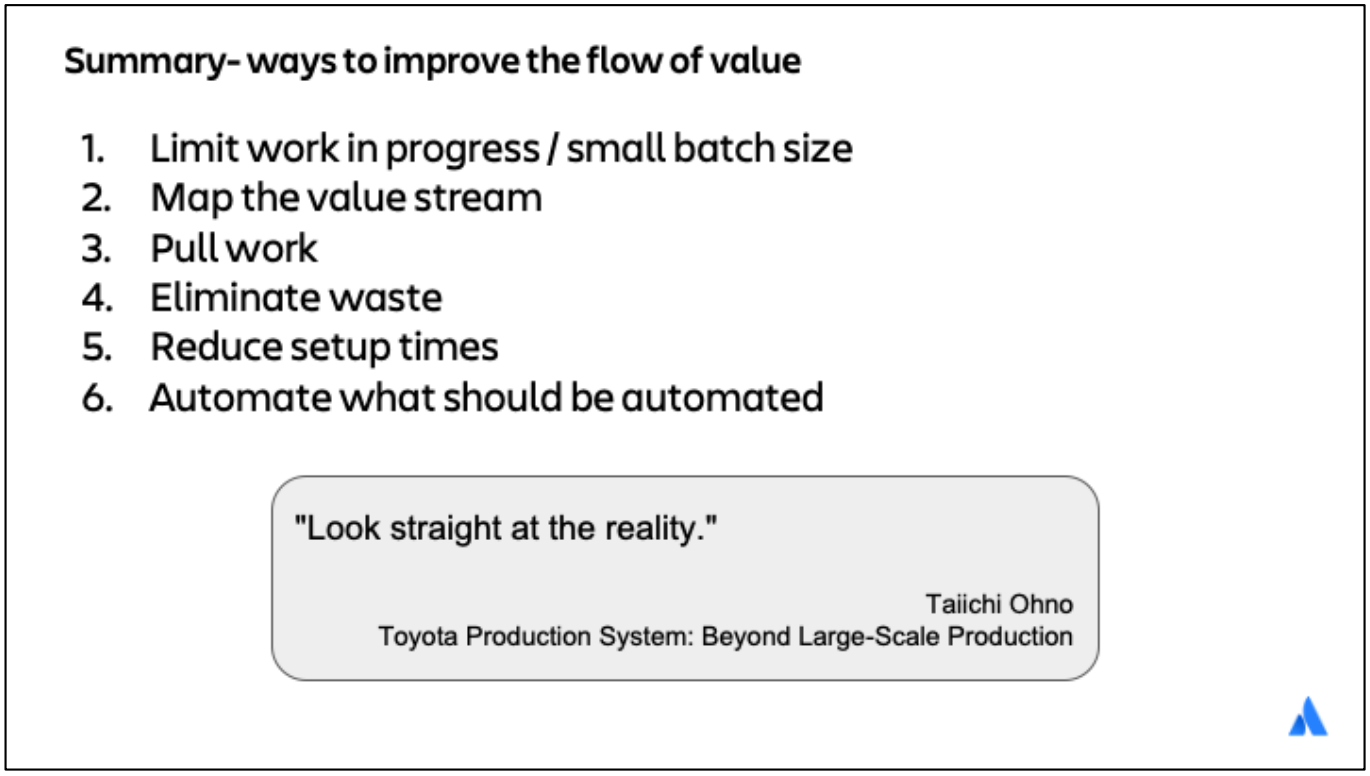
One way to reduce waste is to reduce setup times. Toyota has flexible factories, allowing small batches of a variety of products to be manufactured every day. For small batches to be practical, the tooling has to be fast.
That's why Taiichi Ohno said "Our production slogan is 'small lot sizes and quick setups'.
Reducing setup times is important in all types of work, so that you can focus on doing the actual work. Reducing setup times often adds flexibility that can change the way that the team works.

Improve the flow- automate what should be automated

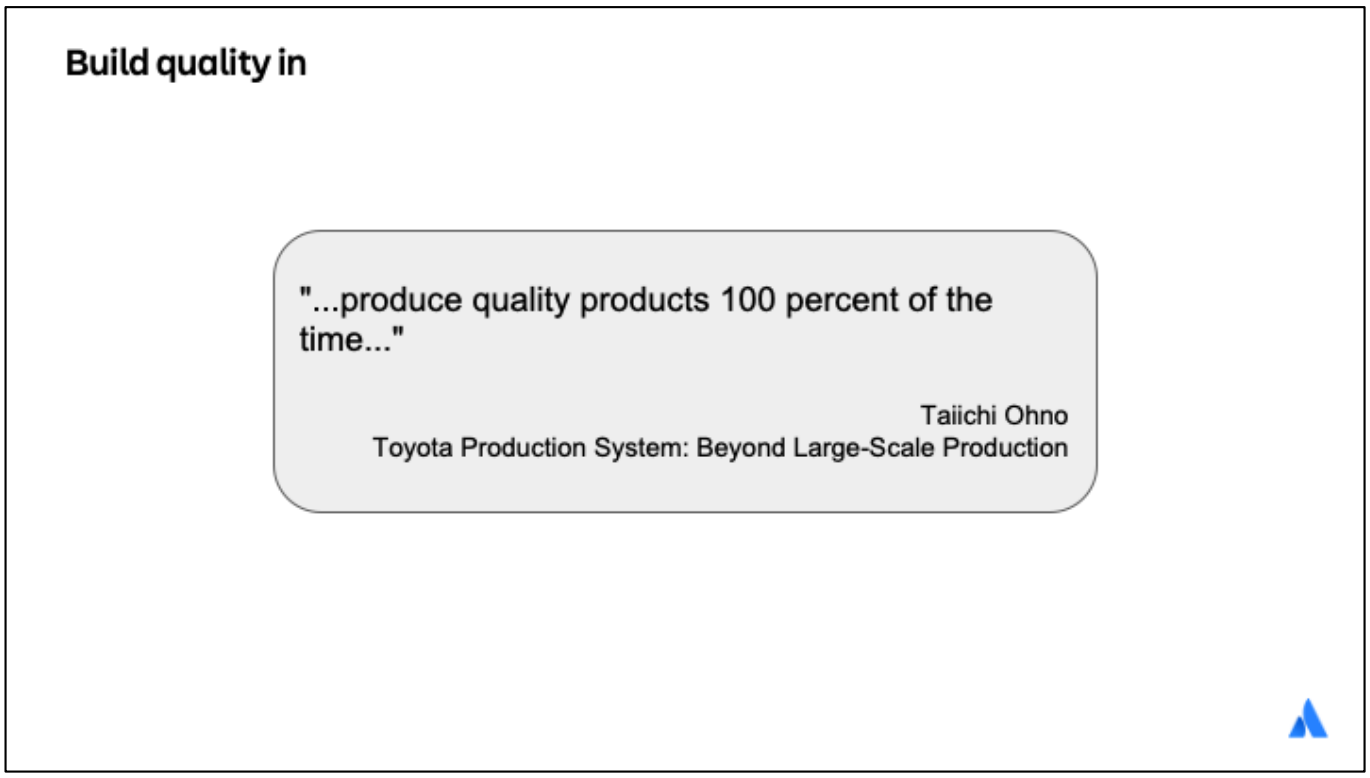
"With computers available, it is a waste to perform calculations by hand."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

Another type of waste is not automating what should be automated. Toyota makes sure to automate where it makes sense. The same is true for any knowledge work tasks that can and should be automated.



Here is a summary of the ways to improve flow on any team.
Taichi Ohno said "Look straight at the reality." It's important for teams to visualize their work and look into the the ways that the flow of value can be improved.

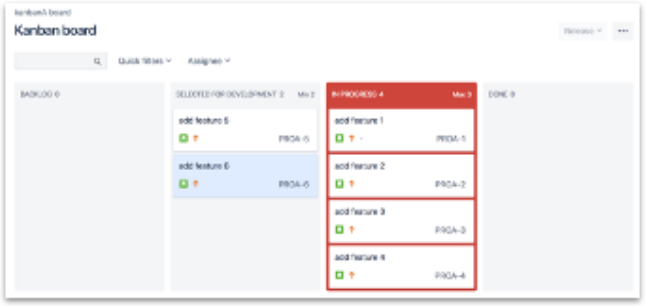


A very important lean principle is to build quality into the product as you are building it. You don't want to start with poor quality and then have the mindset that the quality will improve over time.
As Taiichi Ohno says, "produce quality products 100 percent of the time". It's import to start with a strong technical foundation of what you are doing, and to never compromise on quality to save time.

Build quality in- the process should identify problems

"...distinctions between normal and abnormal operations must be clear and countermeasures (solutions) always taken to prevent recurrence."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production



Related to building quality in is the principle that the process should identify problems. You do not want problems to go unnoticed.
This is one of the reasons for having the andon board at Toyota. If there is a problem, it is displayed for all to see and take appropriate action.
We have also seen an example of the process identifying a problem by adding work in progress limits to columns on the board. When the team goes over the work in progress limit, the column is turned red to signal the team that there is a problem, and to take appropriate action. Your team will have custom ways of identifying problems.

Build quality in- fix problems when they are discovered

"Correct a mistake immediately- to rush and not to take time to correct a problem causes work loss later."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

It is important to fix problems when they are discovered. You don't want to build a list of problems to fix. This takes more time and discipline, but the result is much better quality.

Example- fix problems when they are discovered

"Because a device that could distinguish between normal and abnormal conditions was built into the machine, defective products were not produced."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

http://www.tcmit.org/english/exhibition/textile/fiber03.html

In his book, Taiichi Ohno describes the origin of the idea to fix problems when they are discovered.
Sakichi Toyoda was the founder of the Toyota Motor Company. Prior to that, he owned a company that weaved fabrics. He invented an auto-activated weaving machine. The machine would stop if any of the threads broke. An operator could then replace the thread. Prior to this invention, if a thread broke, it was only discovered after the defective products were already made. In other words, the quality control was moved from after the product was built to during the building of the product.
If you are familiar with the concept of test-driven software development, you will notice that this is the same principle. Any problems with the software are discovered and fixed as the software is being developed.

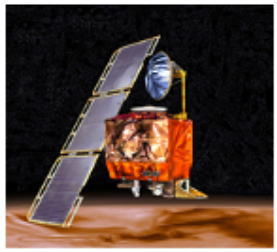
Build quality in- identify and fix the root cause

"By asking why five times and answering it each time, we can get to the real cause of the problem, which is often hidden behind more obvious solutions."

Taiichi Ohno
Toyota Production System: Beyond Large-Scale Production

When problem solving, it is important to identify and fix the root cause of the problem. Taiichi Ohno talks about Toyota's "five whys" approach to problem solving.
The idea is to continue asking why questions until you find the root cause of the problem, and then fix the problem there. The number five is just a typical number of times the why question needs to be asked. You want to keep asking why until you aren't talking about a symptom, but the root cause of the problem. Stopping early and fixing a symptom will usually mean that the root cause will create a problem somewhere else.

Example: asking "why" to find the root cause and solution



Problem: The orbiter crashed into the planet.

Question: Why did the orbiter crash into the planet?
Answer: Because it didn't have the proper trajectory on approach.

Question: Why didn't it have the proper trajectory on approach?
Answer: Because the thrusters did not work properly.

Question: Why didn't the thrusters work properly?
Answer: Because the acceleration data in the software was inaccurate.

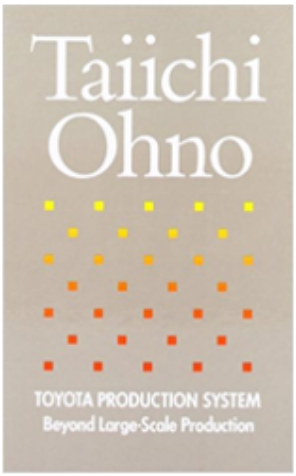
Question: Why was the acceleration data inaccurate?
Answer: Because one team used metric units and the other used English units.

Solution: Standardize on a single system of measurement. Ensure pre-launch tests fail in similar circumstances.

This example is based loosely on the 1998/1999 Mars Climate Orbiter.
https://en.wikipedia.org/wiki/Mars_Climate_Orbiter

Lean principles

1. Empower the team
2. Visualize work
3. Embrace the scientific method
    a. Continuously learn and improve
    b. Embrace change
4. Improve the "flow" of value
    a. Limit work in progress / small batch size
    b. Map the value stream
    c. Pull work
    d. Eliminate waste
    e. Reduce setup times
    f. Automate what should be automated
5. Build quality in
    a. The process should identify problems
    b. Fix problems when they are discovered
    c. Identify and fix the root cause



That is an overview of some of the lean principles. Taken together, they provide a way of working that is very effective in many contexts.
If you look at the individual principles, you can see that they are timeless. The word "lean" is currently used to describe them, but that word is not important and may not be universally used. We will see that agile principles are very similar to lean principles.

Topics

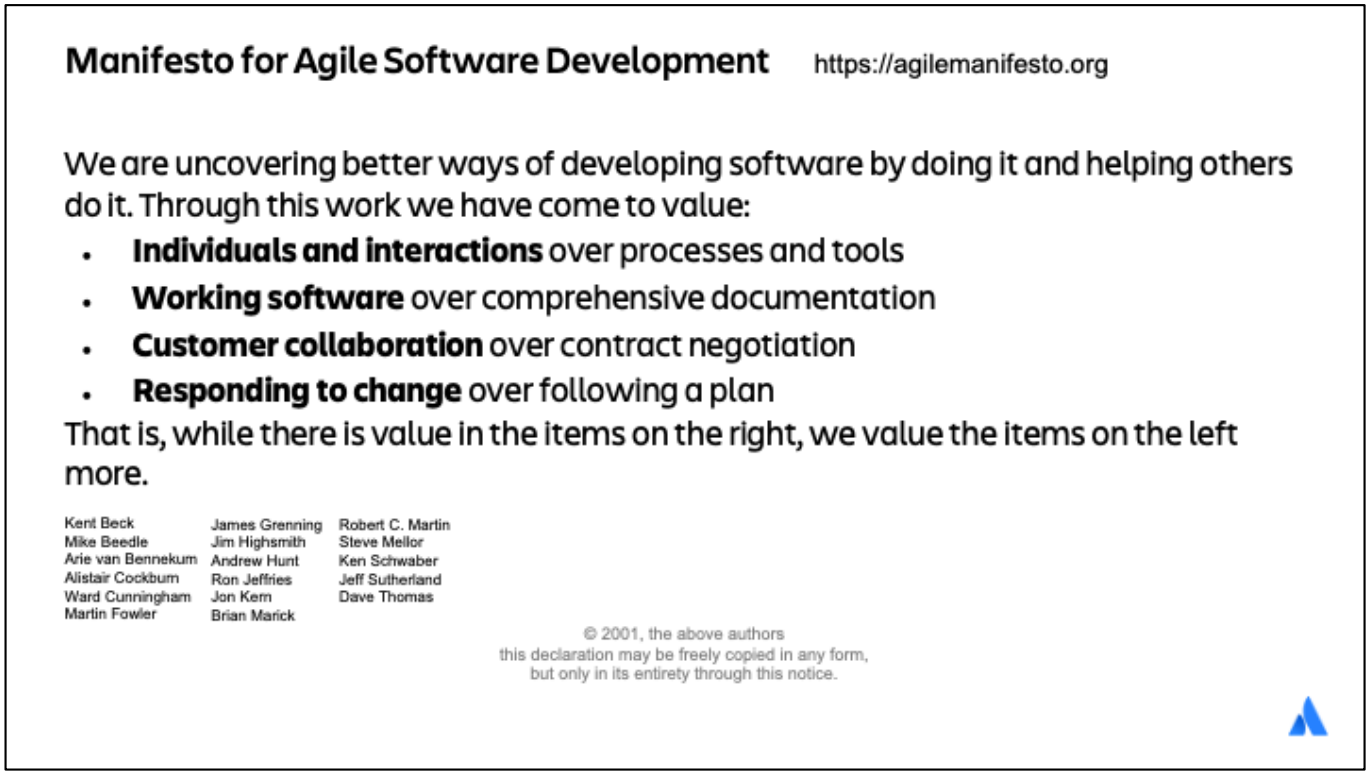
Toyota Production System

Toyota kanban

Lean principles

Agile Manifesto

Lean vs. agile



Here is the manifesto for agile software development, commonly called the Agile Manifesto. We will go through it piece by piece.

First of all, notice that the copyright statement is from 2001. The authors agreed at the time not to change it moving forward, so it is a static, historic document. Having a static document related to agile software development is a bit ironic, because continuous improvement is fundamental to agility. The document is static not because the writers thought it was perfect, but because this is what they agreed to at that time. The agile manifesto represents the commonality of some of the different agile approaches at the time.

If we step back and look at the title of the agile manifesto, you can see that it was intended as an approach to software development. Agility is broader than that, so the scope of this document is limited. However, we will see that many of the ideas apply to agile projects in general.

The agile manifesto starts with a bit of context setting. "We are uncovering better ways of developing software by doing it and helping others do it."

The "we" in that sentence refers to the people who authored the agile manifesto. They had a lot of experience, and they described in general what they found to work well in developing software.

By "uncovering better ways of developing software", they are basically saying that in their experience, the plan-based or waterfall approach does not work well for software development compared to an agile approach. At the time they wrote this, agile approaches were not as common as they are today, and this was an attempt to help transform the thinking of the software development industry.

Next the manifesto contains four value statements. The first one is "through this work we have come to value individuals and interactions over processes and tools." Using the word "over" in the value statements helps clarify what they are talking about. In the context of trying to convince an industry to change from waterfall to agile, this approach makes more sense. They are placing high value on individuals and their interactions. This is related to empowering the team and encouraging collaboration. They are placing lower value on processes and tools. An empowered team can choose the processes and tools that work best for them.

After the four value statements, they further explain the structure of value statements. "That is, while there is value in the items on the right, we value the items on the left more." Again, in the context of trying to convince an industry to change, this makes sense. In an agile approach, the team is empowered to make decisions. The processes and tools that this team uses is secondary to that.

Then next statement is "... we have come to value working software over comprehensive documentation". This places the value on continuously building things that work over heavy upfront planning. This is similar to the lean principle of focussing on value and eliminating waste. A large up-front planning document is likely to be quite wrong and/or misunderstood, and therefore is a source of waste.

The next statement is "... we have come to value customer collaboration over contract negotiation". This is basically saying that the customer should be a member of the team, realizing that building the project is a continuous learning process on all sides. You don't want to have an arms length relationship with your customer that is bound by an inflexible agreement, as is common with waterfall projects. When an upfront agreement is made, the true nature of the project is not completely known, and the team and its customers should realize that and work together.

The last of the four statements is "...we have come to value responding to change over following a plan". This is basically acknowledging that, especially in complex one-off projects, you can not predict the future and instead of heavy up front planning , a better strategy is to embrace change.

So that is a look at the value statements in the agile manifesto. They were basically saying that, for software development, they have experienced that agile processes work better than traditional waterfall processes. These ideas seem straightforward and almost common sense now, but many projects that should have been managed this way were not, and in some cases, that is true for current day projects.

You can see that most of these value statements have nothing to do with software, and can apply to many types of projects. For example, if you change the words "working software" to something like "incremental planning and development", the ideas are more generally applicable.

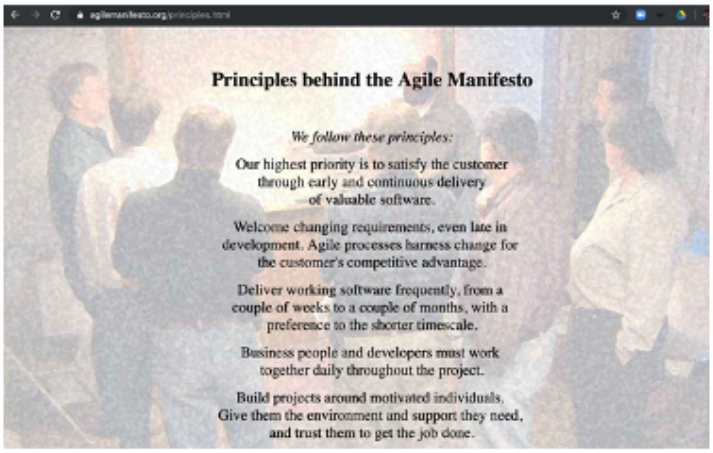
Agile Manifesto value statements- takeaway principles

1. Empower the team
   - "**Individuals and interactions** over processes and tools"
2. Embrace change
   - "**Responding to change** over following a plan"
3. Partner with the customer
   - "**Customer collaboration** over contract negotiation"
4. Plan, develop and deliver incrementally
   - "**Working software** over comprehensive documentation"

To simplify things, we will convert the agile manifesto value statements into principles. We will then build an organized list of principles as we continue.
The first principle is to empower the team. This comes from the "individuals and interactions over processes and tools" statement.
The second principle is to embrace change. This comes from the "responding to change over following a plan" statement.
The third principle is to partner with the customer. This comes from the "customer collaboration over contract negotiation" statement.
The fourth principle is to plan and develop incrementally. This comes from the "working software over comprehensive documentation" statement.

Principles behind the Agile Manifesto

https://agilemanifesto.org/principles.html



Besides the four value statements, there are 12 principles behind the agile Manifesto. The link above contains the principles.

Principles and ideas from the Agile Manifesto

1. Empower the team
    a. Select motivated individuals
    b. Teams should self-organize
    c. Collaborate to create shared understanding
2. Embrace change
    a. Partner with the customer
    b. Obtain fast feedback
    c. Continuously inspect and adapt
3. Plan, develop and deliver incrementally
    a. Prefer conversations for conveying information
    b. Continuously refactor to maintain agility
    c. Maintain a sustainable pace
    d. Completed work items are the primary measure of progress
4. Focus on value
    a. Eliminate waste
    b. Continuously strive for simplicity
    c. Don't compromise on quality

This is a combined list of principles from the agile manifesto value statement and 12 principles, rewritten for general project use.

Topics

Toyota Production System

Toyota kanban

Lean principles

Agile Manifesto

Lean vs. agile

Lean vs. agile

- Lean
  - Used at MIT by John Krafcik (1988)
  - Described the ideas of the Toyota Production System
  - Applies to any type of project
- Agile
  - Used by the participants who created the Agile Manifesto (2001)
  - Described a lightweight alternative to waterfall software development
  - Applies to any type of project
- The terms are often used interchangeably

We have discussed lean and agile principles.
Lean is a term used at MIT by John Krafcik.
It was used to describe the ideas of the Toyota Production System.
The ideas apply to any type of project, even personal projects.
Agile is a term used by participants who created the Agile Manifesto in 2001.
Agile originally described a lightweight alternative to plan-driven, or waterfall, software development.
Over time, agile has taken on a broader meaning that can be applied to any type of project.
The Toyota Production System could have been named agile, it's just that lean was chosen. Similarly, agile could have been named lean.
The terms lean and agile are often used interchangeably to describe an adaptive approach that focuses on value. Many of the common principles are found in Taiichi Ohno's 1978 book called "Toyota Production System". Agile is probably a better term because lean sometimes has the mistaken connotation that it is just about removing waste, because the opposite of lean is fat.

Lean and agile principles

Lean

1. Empower the team
2. Visualize work
3. Experiment using the scientific method
   a. Continuously learn and improve
   b. Embrace change
4. Improve the "flow" of value
   a. Limit work in progress / small batch size
   b. Map the value stream
   c. Pull work
   d. Eliminate waste
   e. Reduce setup times
   f. Automate what should be automated
5. Build quality in
   a. The process should identify problems
   b. Fix problems when they are discovered
   c. Identify and fix the root cause

Agile

1. Empower the team
   a. Select motivated individuals
   b. Teams should self-organize
   c. Collaborate to create shared understanding
2. Embrace change
   a. Partner with the customer
   b. Obtain fast feedback
   c. Continuously inspect and adapt
3. Plan, develop and deliver incrementally
   a. Prefer conversations for conveying information
   b. Continuously refactor to maintain agility
   c. Maintain a sustainable pace
   d. Completed work items are the primary measure of progress
4. Focus on value
   a. Eliminate waste
   b. Continuously strive for simplicity
   c. Don't compromise on quality

Here are the lean and agile principles that we have discussed. There is a lot of overlap, but also some differences. It's probably not necessary to differentiate lean and agile principles, so next we will simplify this by combining the lists.

Combined lean and agile principles

1. Empower the team
   a. Select motivated individuals
   b. Teams should self-organize
   c. Collaborate to create shared understanding
2. Visualize work
3. Experiment using the scientific method
   a. Continuously learn and improve
   b. Embrace change
   c. Partner with the customer
   d. Continuously inspect and adapt
4. Plan, develop and deliver incrementally
   a. Prefer conversations for conveying information
   b. Continuously refactor to maintain agility
   c. Maintain a sustainable pace
   d. Completed work items are the primary measure of progress
   e. Obtain fast feedback
5. Improve the "flow" of value
   a. Limit work in progress / small batch size
   b. Map the value stream
   c. Pull work
   d. Eliminate waste
   e. Reduce setup times
   f. Automate what should be automated
   g. Continuously strive for simplicity
6. Build quality in
   a. Don't compromise on quality
   b. The process should identify problems
   c. Fix problems when they are discovered
   d. Identify and fix the root cause

Here is the combined list of lean and agile principles. These are all good principles to understand well and to use where appropriate.

7 Scrum Overview I- Artifacts

What will you learn?

- Define scrum
- Describe an increment
- Identify scrum artifacts
- Define velocity

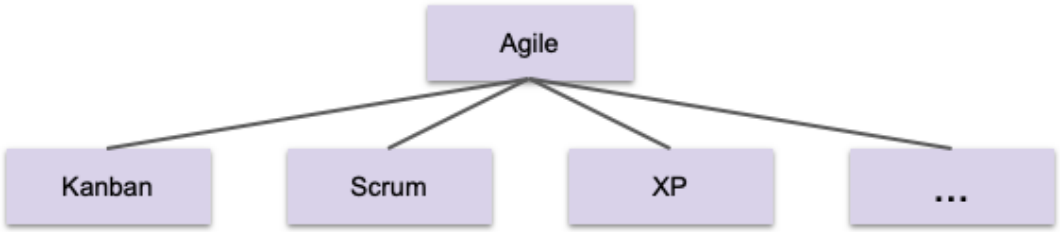
Topics

What is scrum?

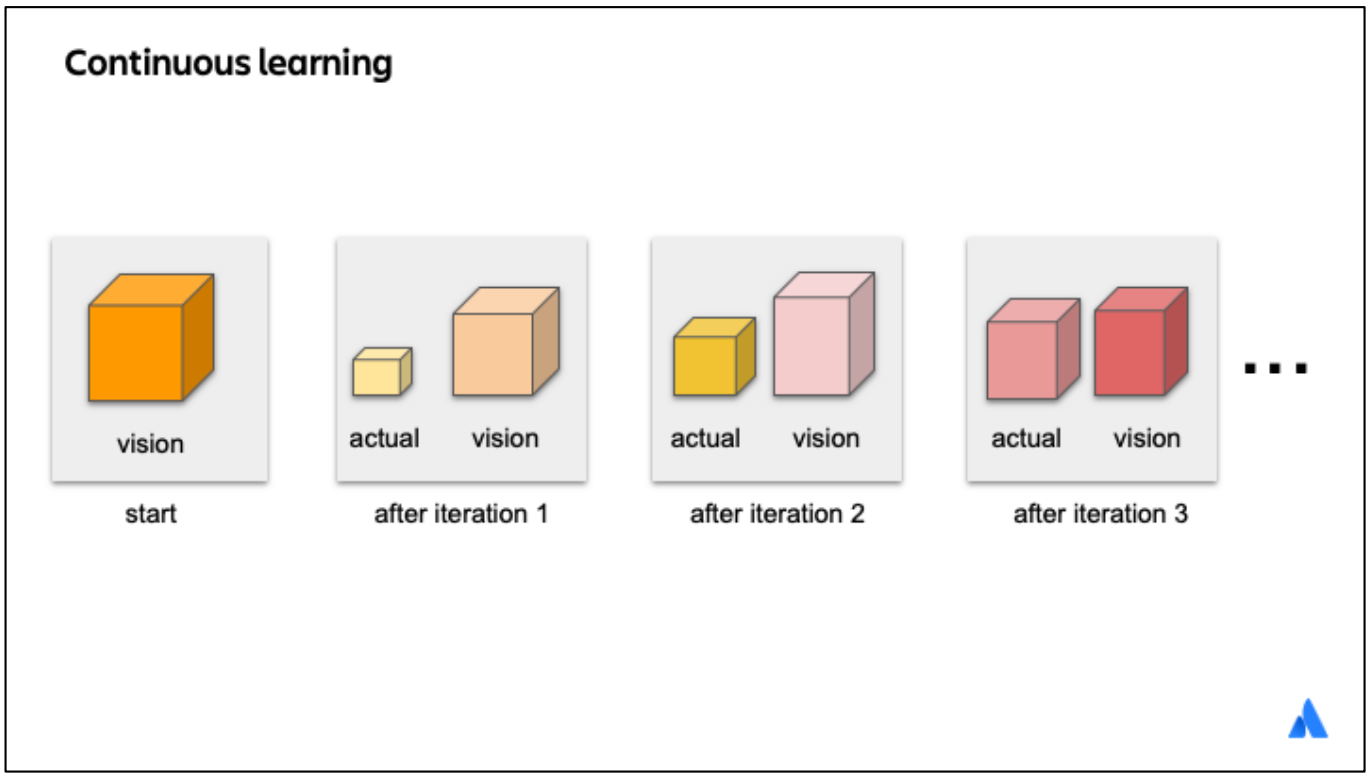
Scrum artifacts

What is scrum?

- "Scrum is a framework for developing, delivering, and sustaining complex products." Scrum Guide 2017
  - https://www.scrum.org/resources/scrum-guide
- A way of achieving agility



According to the Scrum Guide, which is written by the creators of scrum, scrum is a framework for developing, delivering, and sustaining complex products. It is a relatively simple framework for dealing with complex, unpredictable projects.
By framework, we mean that scrum contains basic structure and ideas for completing a project. The Scrum Guide refers to scrum as a process framework for your project management and work techniques, rather than a standalone process or definitive method. Every team and every project is different, so the basic ideas of scrum can be customized to suit your specific project. This is contrasted to a rigid methodology in which every project is executed the same way.
Much of what we will discuss here is discussed in the The Scrum Guide. It is free, relatively short and very well written. It is highly recommended that you read it.
Scrum is a way of achieving the idea of agility.
Previously, we have seen that kanban is another option. You can think of agile as a mindset, and the methods such as kanban, scrum and XP as ways of achieving agility.



A key component of scrum and agile in general is continuous learning. Scrum projects start with a vision. This is the initial desired end result of the project. The stronger the vision at the start of the project, the better. The product is then worked on an iteration at a time.

After the first iteration, we have some of the features of the product. Even though that product only has a few features, it can be considered usable with respect to those features. That product should have value and potentially can be given to the customer. After the first iteration, notice that our vision has slightly changed. This is because we have learned from implementing the first product features and adapted our vision accordingly.

After our second iteration, the product contains the work of the first iteration as well as the work of the second iteration. It now has more features that the customer will value. Because of continuous feedback, we may have improved some of the features from the first iteration. Notice that our vision has again been adapted.

After the third iteration, our product is closer to our vision, but our vision has again changed slightly. You can see that we are building the project incrementally, because we are building it piece by piece, and we are building it iteratively, because the product gets improved as we are learning along the way. That is why scrum is both an incremental and iterative approach to building project. This process of building and improving parts of the product continues indefinitely, allowing the product to stay relevant in a changing marketplace.

If the product that is being built has a physical aspect, such as a rocket, the result of an iteration may be a prototype rather than a part of the completed product.

Increment

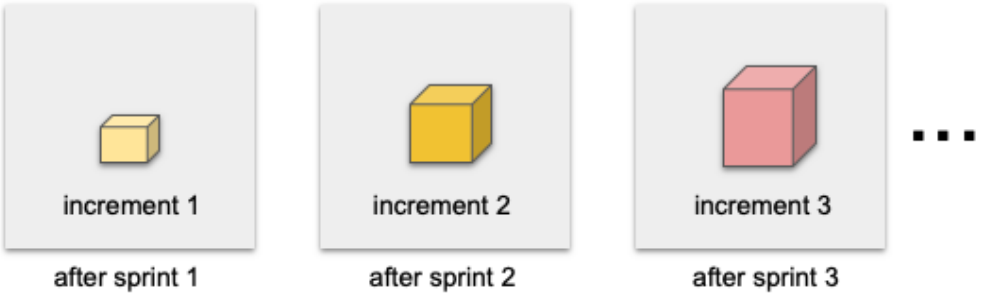
- A usable product that may be given to the customer
- Meets the organization's "definition of done"
- Contains the work of the current iteration, as well as all prior iterations

At the end of every iteration, a product called the increment is ready. Here we see that three iterations have created three increments of the product.
An increment is a usable product that may be given to the customer. The organization always has the option to release the increment.
Each increment must meet the organization's agreed upon definition of done. This includes quality and security standards, as well as other organizational requirements, such as requiring documentation for each feature.
Each increment contains the work of the current iteration, as well as the work of all prior iterations. In other words, an increment is the state of the product after an iteration.

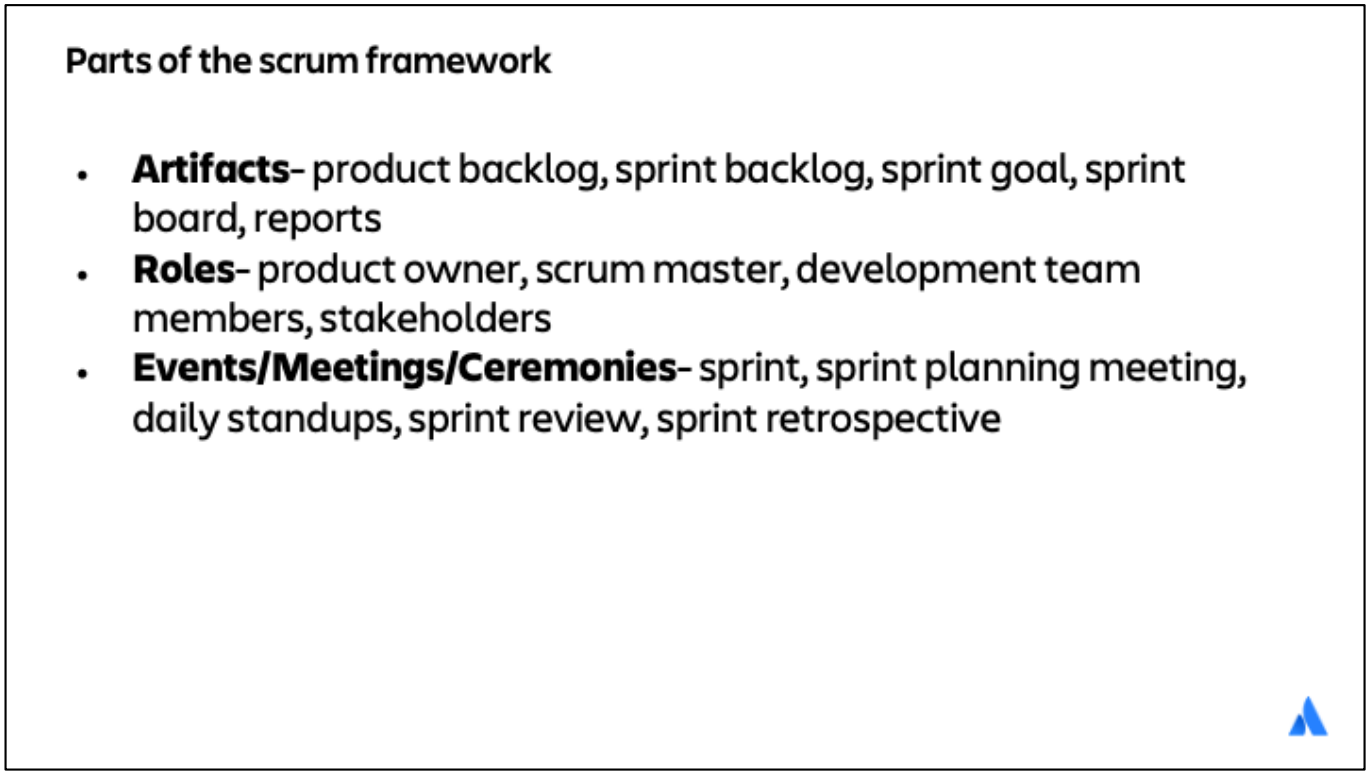
Sprint

A time-boxed period used to work on an increment of the product

- Usually 1-4 weeks (typically 2 weeks)



In scrum, the iterations are called sprints. A sprint is a time-boxed period used to work on an increment. The time period of a sprint is fixed. In general, you do not shorten or lengthen the duration of the current sprint. Sprints usually have a duration of one to four weeks, with two week sprints being typical. Shorter sprints create an opportunity for more adaptation. Longer sprints allow for more work to be done in a single increment. It's up to each team to decide on the appropriate sprint length.



There are three main parts of the scrum framework.
Artifacts are tools that allow for transparency of the project. They allow anyone with access to them to see the current state of the project. The artifacts that we will talk about in this module are the product backlog, the sprint backlog, the sprint goal, the sprint board and sprint reports.
The second part of the scrum framework is the roles related to scrum. In the next module, we will discuss the roles of product owner, scrum master, development team members and stakeholders.
The third main part of the scrum framework is the events related to scrum. These are also called ceremonies or meetings. The sprint guide considers the sprint as a container event for the other events. In the next module, we will discuss the sprint planning meeting, daily standups, the sprint review and the sprint retrospective.

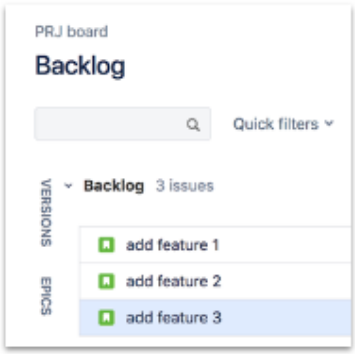
Topics

What is scrum?

Scrum artifacts

Scrum artifacts

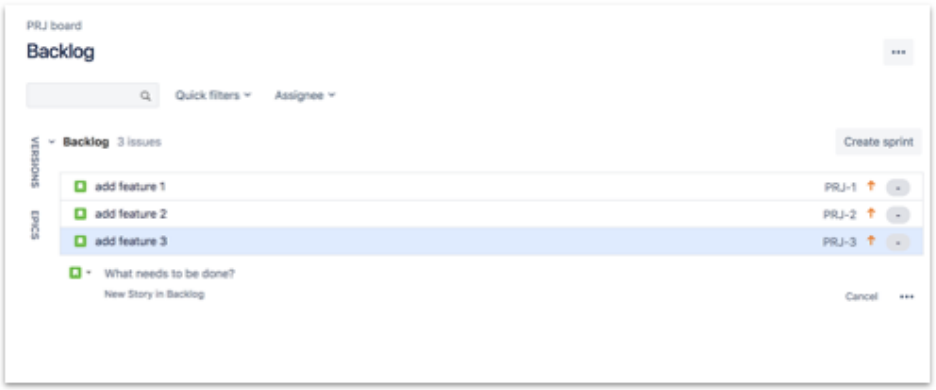
- Artifacts:
  - Product backlog
  - Sprint backlog
  - Sprint goal
  - Sprint board
  - Reports
- Provide project transparency
- Enable shared understanding
- Enable inspection and adaptation



A main purpose of the artifacts is to provide project transparency. Anyone with proper access can use the artifacts to see the current state of the project, including the project's history and future plans. This enables the team to have a shared understanding of the project, so that everybody is on the same page. The scrum artifacts are used to enable inspection and adaptation, both inside and outside of scrum meetings.

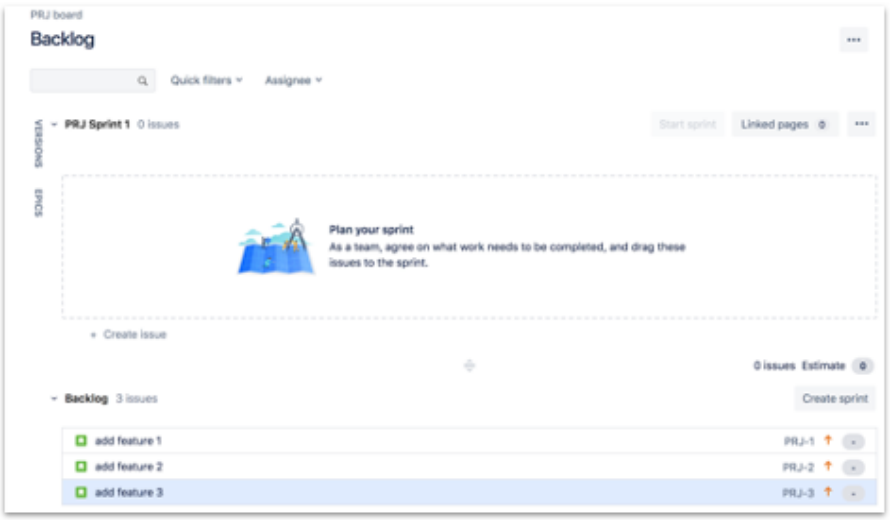
Product backlog

- An ordered, ever-changing to do list for the project
- Can include features, improvements, bug fixes, etc.
- Issues near the top should include more detail
- Modifying the product backlog is called product backlog refinement



A product backlog is an ordered, ever-changing to do list for the project. It contains issues that are not yet part of any sprint. The Scrum Guide refers to issues as items. You might also hear them referred to as stories. Constant feedback means that the product backlog is always changing.
Here you can see the product backlog in Jira. It is located under the Backlog tab for scrum projects. This product backlog contains three issues. When you first add issues to a scrum project, they are automatically placed in the product backlog.
The product backlog can include issues that represent features, improvements, bug fixes and any other type of issue that you would like.
The product backlog is ordered. Issues near the top of the backlog are the closest to being worked on, so they usually have more details than the lower items.
Modifying the product backlog is called product backlog refinement. You might also hear this referred to as backlog grooming. According to the Scrum Guide, each scrum team decides how to do product refinement, but it should consume no more than 10 percent of the development team's time.

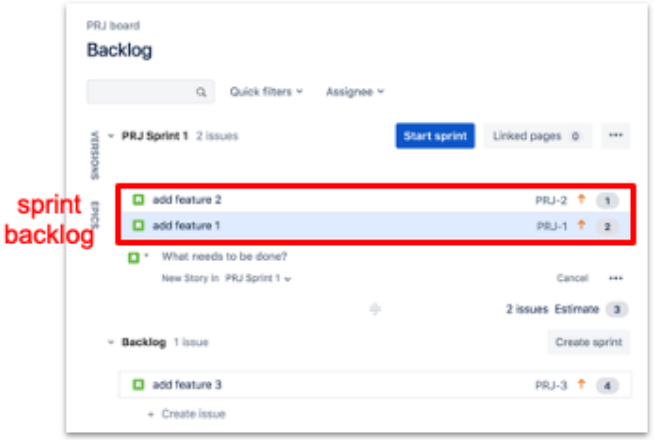
Creating a sprint with Jira



To create a sprint in Jira, you click the Create sprint button in the product backlog. After clicking the Create sprint button, Jira creates an empty sprint. You can see here that Jira named the sprint using the project key, which is PRJ in this example, and the sprint number, which is one in our case. You can see that Jira invites you to drag issues from your product backlog into the sprint.

Sprint backlog

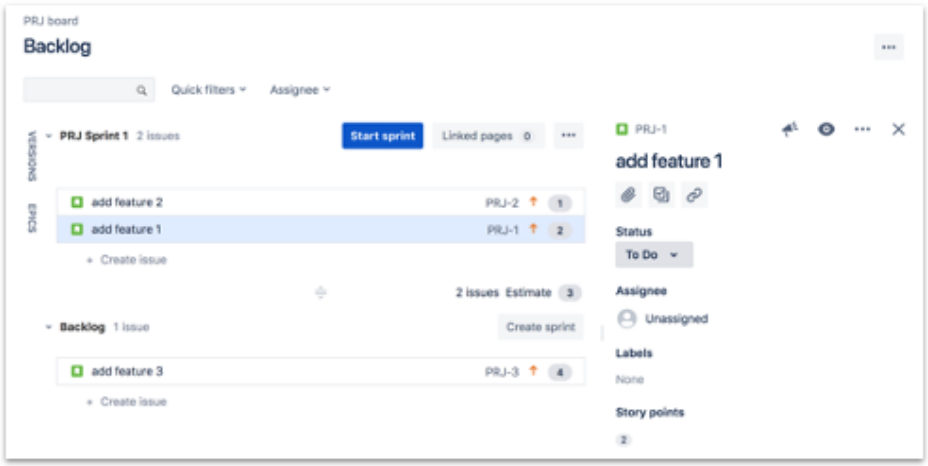
- A subset of the product backlog
- The list of issues to be completed in the sprint
- Includes the plan on how to accomplish the work of the issues



Here, we have dragged two of the issues from the product backlog to the sprint. The list of issues to be completed during the sprint is called the sprint backlog.
The sprint backlog includes a plan on how to accomplish the work of the issues. In Jira, this means that before starting the actual sprint, more details are added to the issues in the sprint backlog. Those details describe how the work of the issues will be done. We will discuss more about adding these details later in the course

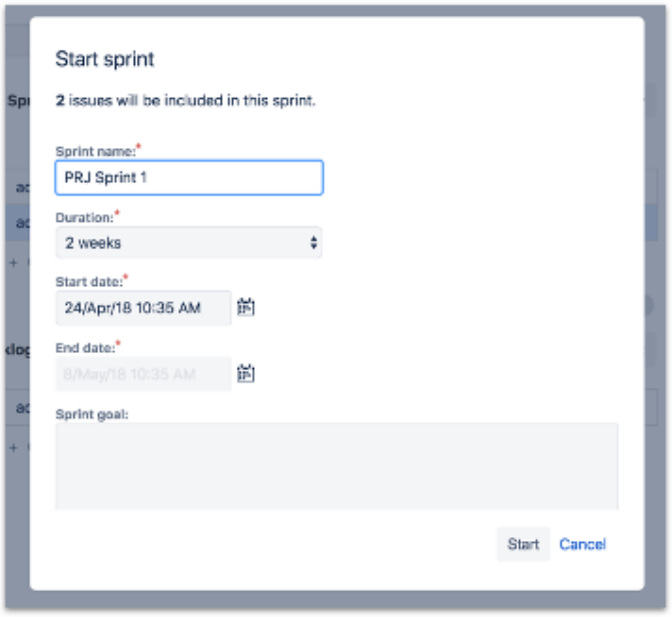
Estimation- story points

- Story points are a relative measure of the amount of work required to complete the story
- Used to help decide how many stories can be completed in the sprint



As part of planning for sprints, it is common to estimate how much work an issue will take. Story points are the most common estimation statistic. In Jira, you can use story points, hours, issue count or create your own estimation statistic.
Story points are a relative measure of the amount of work required to complete an issue. For example, an issue that is assigned two story points is assumed to take about twice as long to complete as an issue that is assigned one story point.
You can see that in Jira, there is a field on each issue named story points. Here we have assigned 2 story points to this issue. In the sprint backlog, you can see that the story points are show in the gray boxes, along with a total estimate of 3 points for the sprint backlog. You can see that the issue remaining in the product backlog has been assigned four story points.
Story points are used to help the team decide how many issues can be completed in a sprint.

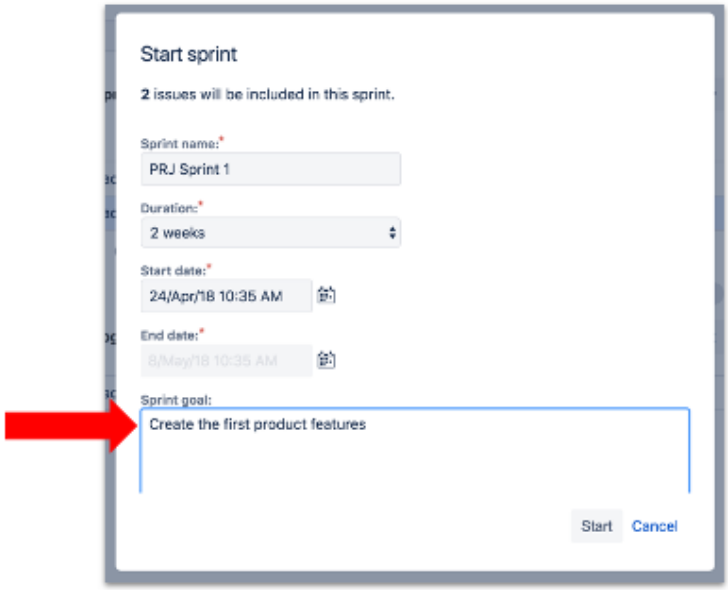
Sprint details



The start sprint screen appears. Jira starts by reminding you that you have added two issues to the sprint backlog. You can modify the sprint name, specify the sprint's duration and specify the start date for the sprint. You can see that you can have the sprint started at a later date, so you don't have to actually click the start sprint button on the first morning of the sprint. You could also set up multiple sprints at one time.

Sprint goal

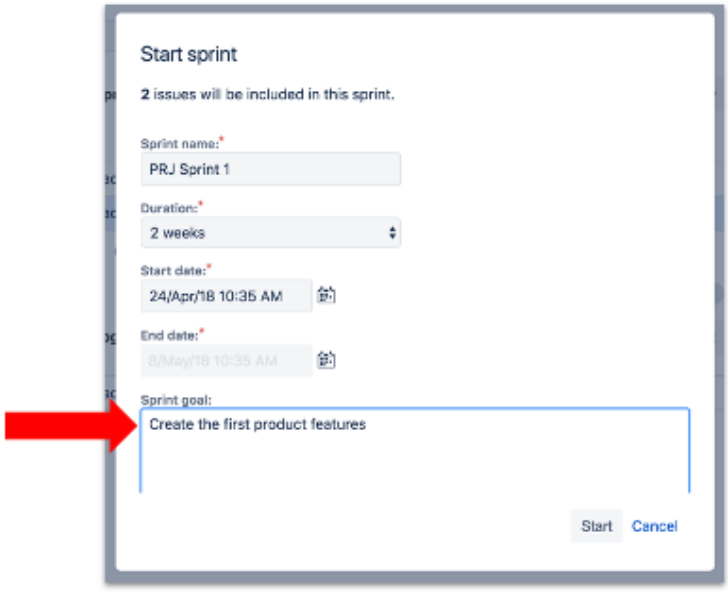
- Represents the objective of the sprint's increment
- Is reached by completing the sprint backlog
- Does not change during the sprint
- The sprint is a success if the sprint goal is reached



The start sprint window also contains a place to enter what is called the sprint goal.
The sprint goal represents the objective of the sprint's increment. It is agreed to by the team.
The sprint goal is reached by completing the issues in the sprint backlog.
A scrum rule is that the sprint goal does not change during the sprint.
The sprint is considered a success if the sprint goal is reached.

Why have a sprint goal?

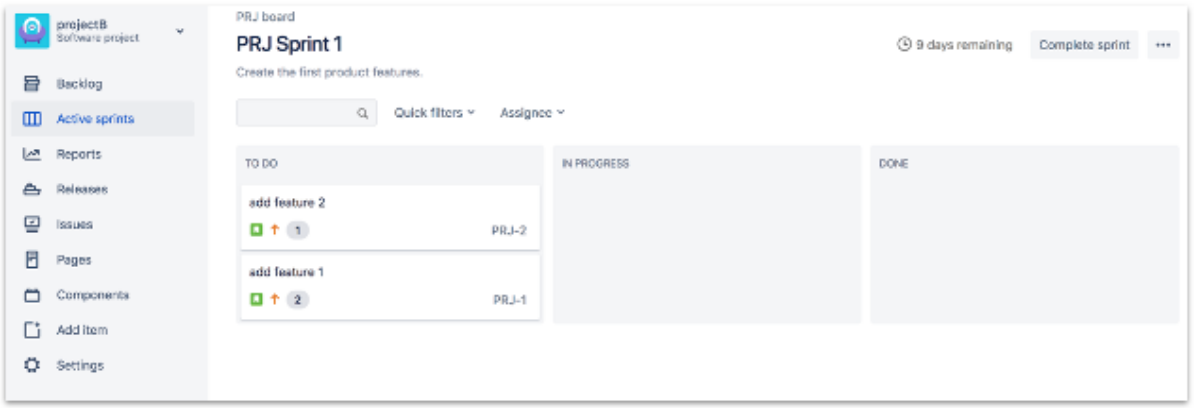
1. Provides coherence to the product increment
2. Enables flexibility with the sprint backlog



There are two major reasons to have a sprint goal.
The first is that it provides a coherence to the product increment. This means that the features are related, so that the product increment is valuable rather than building a collection of unrelated features. This also results in the scrum team working together to achieve the sprint goal.
The second reason is that it enables flexibility with the sprint backlog. Projects are complex, and even though the sprint duration is relatively short, the team can not predict the future and will learn and adjust during the sprint. There has to be flexibility somewhere. The sprint goal remains fixed during the sprint, but the issues that achieve the sprint goal can be modified as long as quality is not decreased. This means that there is flexibility in the makeup of the sprint backlog as the sprint is worked on. The sprint goal provides guidance for decisions as the team makes adjustments.

Sprint board

Only contains issues from the sprint backlog



A sprint has a sprint board. Notice that it only contains the issues in the sprint backlog. Issues in the product backlog, or issues that are assigned to other sprints are not shown on the sprint board. Even in sprint projects, boards are often called kanban boards, so don't be confused if you hear that term related to a sprint.

Scrum reports- burndown chart

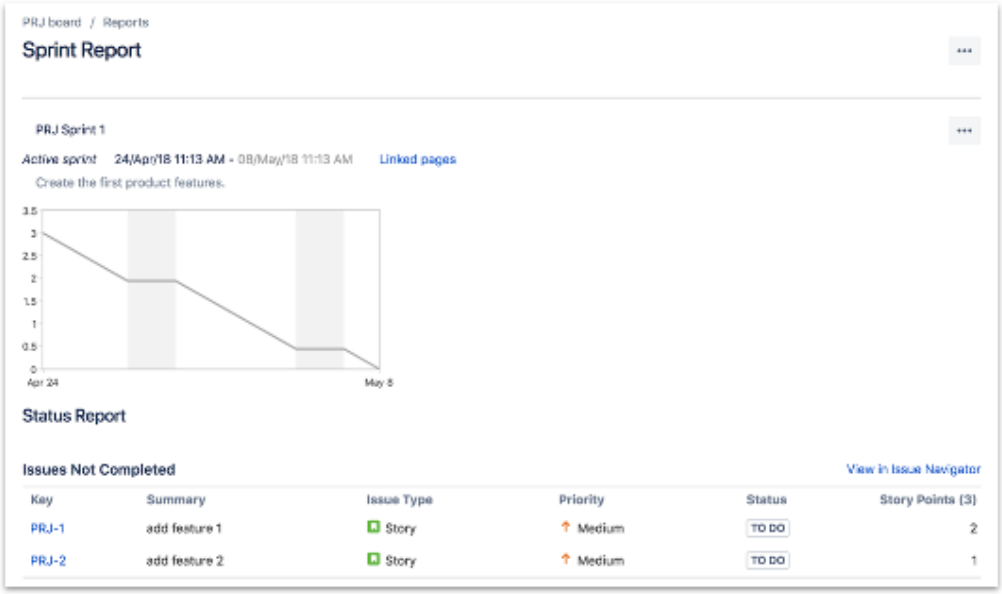
https://www.atlassian.com/agile/tutorials/burndown-charts

A burndown chart shows the progress that the team makes during a sprint. The sprint backlog starts with a certain number issues, each with an associated number of story points or other estimation statistic. The total number of starting story points is shown on the left of the chart. This is the number of story points that the development team estimated that it would complete in the sprint. In our case, this sprint has 3 story points. The gray guideline shown is used to show the number of story points that should remain on a given day, assuming a linear burndown of story points. On the last day of the sprint, the guideline reaches zero story points. This means that the work of the sprint should be finished on the last day. Notice that the non-working days are shown in the chart, and the guidelines assume no progress will be made on those days.

As the sprint is underway, Jira will automatically update the burndown chart as the status of the issues is updated by the team. The red line shows the actual number of remaining story points over time. You can see that about two days into this sprint, one story point was completed. On the last day of the sprint, the remaining two story points were completed. Consulting this chart is an easy way to see if the team is on track for the current sprint. If the red line is below the gray line, your team is on track to complete all of the story points and reach the sprint goal. If not, the team may need to make some adjustments to still reach the sprint goal.

The link shown contains more information on burndown charts.

Scrum reports- sprint report



The sprint report contains a nice summary of the sprint. It shows the burndown chart, as well as the current status of all of the issues in the sprint. This is an easy way to see how the sprint is progressing.

Velocity

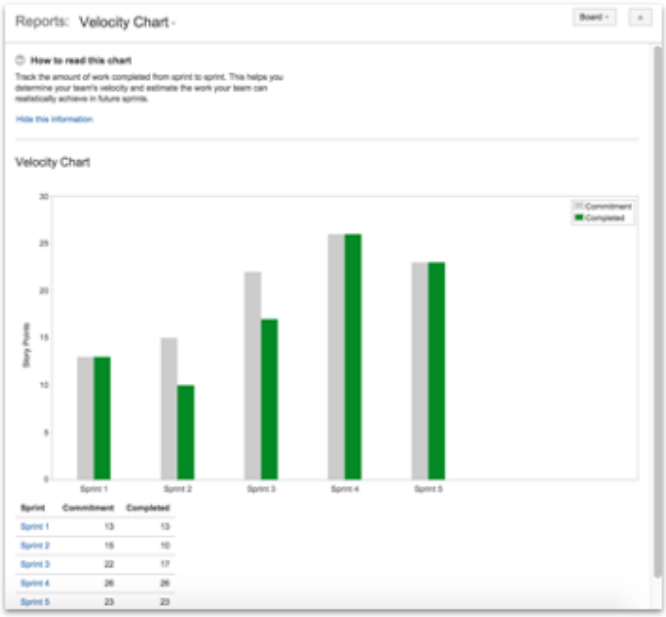
Represents the rate at which the team accomplishes work

- Usually it is the number of story points completed per sprint
- In this example, the velocity is 3 story points per sprint

Velocity represents the rate at which the team accomplishes work.
Usually it is the number of story points completed per sprint. Some teams use an estimation statistic other than story points, so in that case velocity measures some other units completed per sprint.
You can see the team's velocity of a single sprint by looking at a burndown chart. In this sprint, the team completed three story points in the sprint, so its velocity is 3.

Scrum reports- velocity chart

Shows the estimated and actual velocity of the team over time



The velocity chart shows the estimated and actual velocity of the team over time. In this example, prior to the first sprint, the team estimated that it would complete 13 story points, shown with the gray bar. It actually completed those 13 story points, as shown by the green bar. The velocity of the team for sprint one was 13 story points. In sprint 2, the team optimistically estimated that it could complete 15 story points. It actually completed 10.

The velocity chart shows a quick snapshot of the change in the team's productivity over time. As the team improves, its velocity generally improves, so velocity charts usually trend upward.

Takeaways

- Scrum is an agile framework
- An increment is a potentially shippable portion of the project that meets the "definition of done"
- A sprint is a time-boxed period in which an increment is created
- Scrum artifacts provide project transparency, enable shared understanding, and enable inspection and adaptation
- Artifacts include the product backlog, the sprint backlog, the sprint goal, sprint boards and reports
- Velocity is the rate at which the team accomplishes work, usually in story points per sprint

8

Scrum Overview II- Roles and Events

What will you learn?

- Describe scrum roles
- Differentiate the product owner and scrum master
- Identify common characteristics of scrum events
- Identify the purpose of the sprint planning meeting, daily standup, sprint review and sprint retrospective

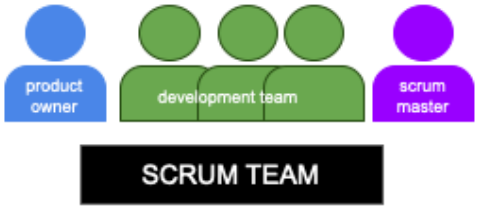
Topics

Scrum roles

Scrum events

Scrum team

- Made up three roles: product owner, scrum master, development team
- Cross-functional
- Flexible/adaptable
- Self-organizing



A scrum team is made up of three roles, product owner, scrum master and development team.
The scrum team is made up of cross-functional team members, allowing it to complete stories within the team.
It is flexible and adaptable, with members willing to help out where needed and continuously learning new things.
It is self-organizing. The team is responsible for deciding how to organize and do its work.

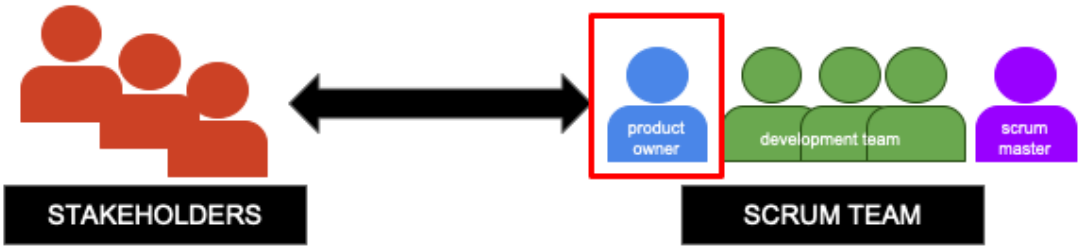
Stakeholders

- Others interested in the success of the project
- Internal- company managers, executives, other scrum teams
- External- customers, partners, investors

Stakeholders are not members of the scrum team, but are interested in the success of the project.
There are internal stakeholders, such as company managers, executives or other scrum teams that rely on the work of the scrum team.
There are also stakeholders that are external to the organization, such as customers, partners and investors.

Product owner

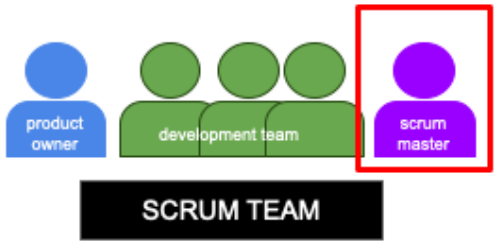
- Responsible for:
  - communicating the product vision
  - maximizing the value of each increment
  - the product backlog
- Interacts with, represents and is accountable to stakeholders



The product owner is the member of the scrum team who is responsible for communicating the product vision. The stakeholders and the scrum team need to have an understanding of the product vision in order to work effectively.
The product owner is also responsible for maximizing the value of each increment. Each feature that the development team works on should be of high value. <click>The product owner is responsible for the product backlog. Others may help with the product backlog, but the product owner is responsible for it.
Stakeholders primarily interact with the product owner. The product owner represents the stakeholders when they are not part of the discussion, such as during scrum team meetings. The product owner is accountable to the stakeholders for the success of the project.

Scrum master

- Responsibilities include:
  - promoting and supporting scrum
  - improving the day-to-day effectiveness of the team
  - protecting the focus of the team
  - increasing the transparency of the project
- Typical tasks:
  - coaching the scrum team and stakeholders on scrum
  - removing blocking issues
  - facilitating scrum events
  - configuring scrum artifacts
  - monitoring sprint progress



The scrum master is the member of the scrum team who is primarily responsible for promoting and supporting scrum, for scrum team members as well as stakeholders. It is up to the scrum master to ensure that everyone understands how and why things are done a certain way.
The scrum master is also responsible for the day-to-day effectiveness of the scrum team. This includes ensuring that the team is reaching its goals and continuously improving.
The scrum master is responsible for protecting the focus of the team. This may mean helping to remove bottlenecks or ensuring that those outside are interacting with the team in helpful ways. In general, scrum masters do what they need to do to allow team members to focus on their work.
The scrum master is also responsible for increasing the transparency of the project. There should be no surprises about the current status of the project.
The scrum master's tasks vary by project and from day to day. Typical tasks include:

- coaching the scrum team and stakeholders on scrum and agile.
- helping to remove blocking issues.
- facilitating scrum events.
- configuring scrum artifacts.
- monitoring sprint progress.

Product owner vs. scrum master

product value

team effectiveness

Why separate roles?

- divide and conquer
- checks and balances

The distinction of the product owner and scrum master roles should now be pretty clear. The product owner is responsible for the value of the product. The scrum master is primarily responsible for the effectiveness of the team.
You can think of this as both a divide and conquer approach, because the work of the roles is so different and as a checks and balances approach, because combining the roles could put too much weight on one of the responsibilities. The separate roles lead to greater team success and sustainability. Combining these roles into one person is discouraged because the roles have different, sometime competing, primary responsibilities. Combining the roles defeats the checks and balances.
Some companies choose to use a single scrum master for several teams. Some teams have a scrum master who is also a part time development team member.

Development team

- Cross-functional, adaptive team that does the work of the project
- Responsibilities include:
  - estimating issues
  - deciding how much work can be done in a sprint
  - deciding how to organize to do the work of the sprint
  - creating the increment of each sprint
  - ability to modify the sprint backlog during the sprint
- The Scrum Guide recommends three to nine members

The development team is a cross-functional, adaptive team that does the work of the project.
Responsibilities of the development team include

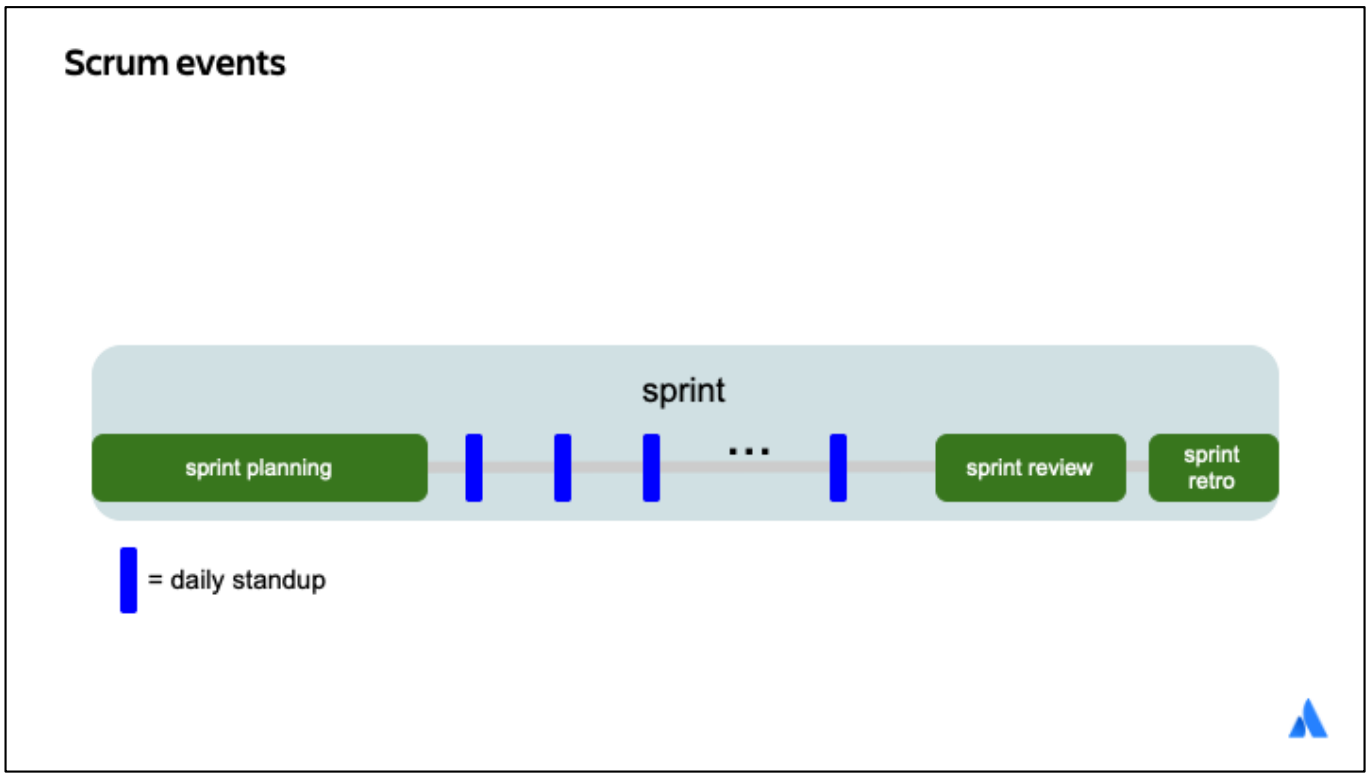
- estimating issues. This is usually done using story points, but other estimation methods can be used.
- deciding how much work can be done in the sprint. Only the development team can decide how much work to take on. It is assumed that the development team is in the best position to forecast how much work the issues take.
- deciding how to organize to do the work of the team. This is because the team is an empowered, self-organizing team.
- creating the increment of each sprint.
- the development team are the only members allowed to modify the sprint backlog during the sprint. This is sometimes necessary as the team learns and adjusts as it is building.

The scrum guide recommends having from three to nine members of the development team. Fewer than three members decreases the productivity and quality created by a cross functional group of people working together. More than nine members tends to increase the amount of coordination required. Jeff Bezos at Amazon refers to two pizza teams, meaning two pizzas should be able to feed the entire team. To scale scrum to more than nine people, it is usually better to create multiple teams.

Topics

Scrum roles

Scrum events

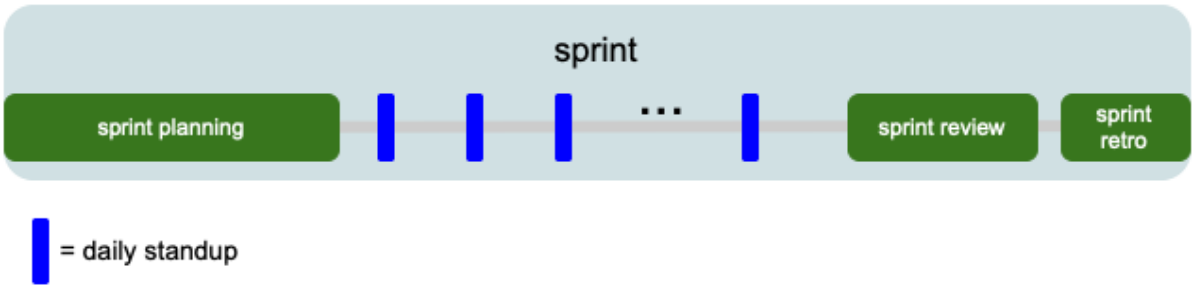


A sprint contains four types of events. You may also hear events referred to as ceremonies or simply meetings. The events are the sprint planning meeting, the daily standups, the sprint review and the sprint retrospective. Scrum events occur at regular intervals and minimize the need for other meetings. These events are designed to maximize the opportunities for feedback and continuous learning, and are a key part of achieving agility. In between these meetings is where the work of the issues of the sprint is completed.
We will discuss each of these meetings separately.

https://www.atlassian.com/agile/scrum/ceremonies

Characteristics of all scrum meetings

- Fixed maximum time limit, no minimum time limit
- Meetings are primarily to plan, inspect and adapt
- Primarily about collaborating, not updating status
- Primarily spend time on things of value to all participants



All of the scrum meetings have some common characteristics.
The meetings have a fixed maximum time limit and no minimum time limit. Meetings can never go over their allotted time, but they can be ended early if the purpose of the meeting is achieved.
The meetings are primarily to plan, inspect and adapt. In agile projects, the planning is distributed throughout the project rather than the mostly upfront planning of waterfall projects. These meetings are an important part of that distributed planning.
The meetings are used to inspect the project and adapt what the team is doing based on that inspection. This is a key part of increasing transparency and continuous improvement.
The meeting are primarily about the team collaborating, not about updating status. The work is visualized, so you don't need a meeting to see the status.
It is the responsibility of every team member to ensure that the meetings have value, and to help modify them to increase their value if necessary. If the discussion moves in a direction that becomes of value to only a portion of the participants, it should be moved to a later time outside of the meeting.

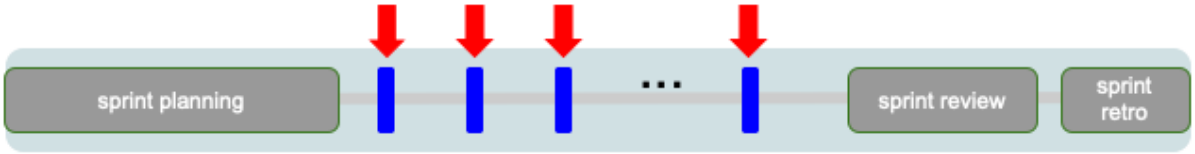
Sprint planning meeting

- **Attendees**: entire scrum team
- **Duration**: typically four hours for a two week sprint
- **Purpose**: plan the work of the sprint
- **Output**: sprint goal, sprint backlog

The sprint planning meeting is held at the start of the sprint.
The entire scrum team attends the meeting.
The duration of the meeting is typically four hours for a two week sprint. If your sprints are 4 weeks long, you could expect this meeting to be twice as long.
The purpose of the meeting is to plan the work of the sprint.
The output of the meeting is an agreed upon sprint goal and a sprint backlog.
Before the meeting, the product owner usually has a proposed sprint goal and the minimum set of issues that accomplish the goal. These preliminary items often come from the product backlog that has been updated during the previous sprint's sprint review and sprint retrospective meetings. We will discuss those meeting a bit later.
During the meeting, the team usually discusses the sprint goal, modifies the sprint backlog, places story point estimates on issues and add details to the issues to better describe the specific work to be done. The development team is responsible for estimating the story points for the work and deciding on how much work can be done during the sprint. They need to do enough planning to have an accurate forecast for the amount of work that they will agree to. The development team also creates subtasks for the first few days of the sprint. The subtasks are often a day or less of work. This is an example of an empowered team rather than a command and control-based team.
The purpose of the meeting has been met when there is an agreed upon sprint goal and a sprint backlog. The sprint backlog contains the issues that the development team has agreed to complete during the sprint as well as a plan for completing that work.

Daily standup

- **Attendees**: development team and scrum master (primarily)
- **Duration**: 15 minutes
- **Purpose**:
  - Inspect recent progress toward the sprint goal
  - Plan the day's work
  - Identify any impediments, and plans to resolve them
- **Output**: plan for the day



https://www.atlassian.com/team-playbook/plays/standups

The daily standup meeting is also called the daily scrum. It is a planning meeting that occurs every day, and the participants usually stand as a reminder that it is a short meeting. The meeting usually takes place in the same location at the same time every day.
The development team and scrum master are the primary attendee for the daily standup. Others may attend the meetings, but are usually asked to listen only.
The daily standup is often facilitated by the scrum master. Some teams choose to rotate the facilitator among the team members.
The purpose of the meeting is to inspect recent progress toward the sprint goal, plan the day's work and identify any impediments to the team. The team usually makes plans to resolve the impediments, but the discussion often moves to a discussion that follows the meeting.
The output of the meeting is the plan for the day.
It's important that the daily standup is a collaborative meeting and not simply a status update. The team collaborates to make the best decisions for the day given the latest information. They usually will decide on who will work on specific issues. The plan slightly changes after every daily standup. This is continuous improvement.
The link at the bottom is a link from the Atlassian team playbook on standups. You can find more information there.

Sprint review

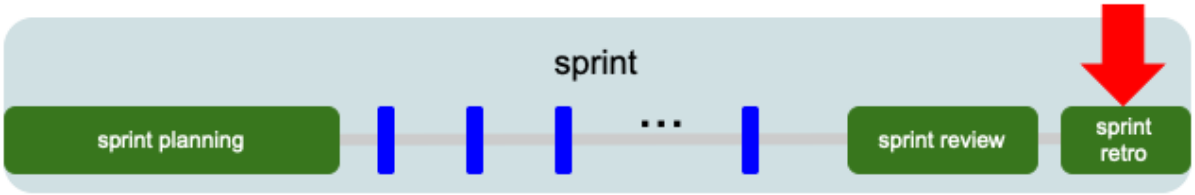
- **Attendees**: scrum team and stakeholders
- **Duration**: typically 2 hours for a 2 week sprint
- **Purpose**: Inspect the increment and collaboratively update the product backlog
- **Output**: first-pass next sprint backlog

https://www.atlassian.com/agile/scrum/sprint-reviews

The sprint review meeting occurs near the end of the sprint.
It is an informal meeting that includes the scrum team and interested stakeholders.
It is typically a two hour meeting for two week sprints.
The purpose of the meeting is to inspect the increment that was just created in the sprint and to collaboratively update the product backlog. This is a meeting with a lot of feedback on the project and includes a brainstorming session to help decide what to do next.
The output of the meeting is a first-pass at the next sprint's backlog. By the time the sprint planning meeting happens for the following sprint, the team already has a good idea of what they will be working on.

Sprint retrospective

- **Attendees**: scrum team
- **Duration**: typically 90 minutes for a 2 week sprint
- **Purpose**: the team inspects itself, including its processes, tools and team interaction
- **Output**: Improvement issue(s) added to the next sprint's backlog



https://www.atlassian.com/team-playbook/plays/retrospective

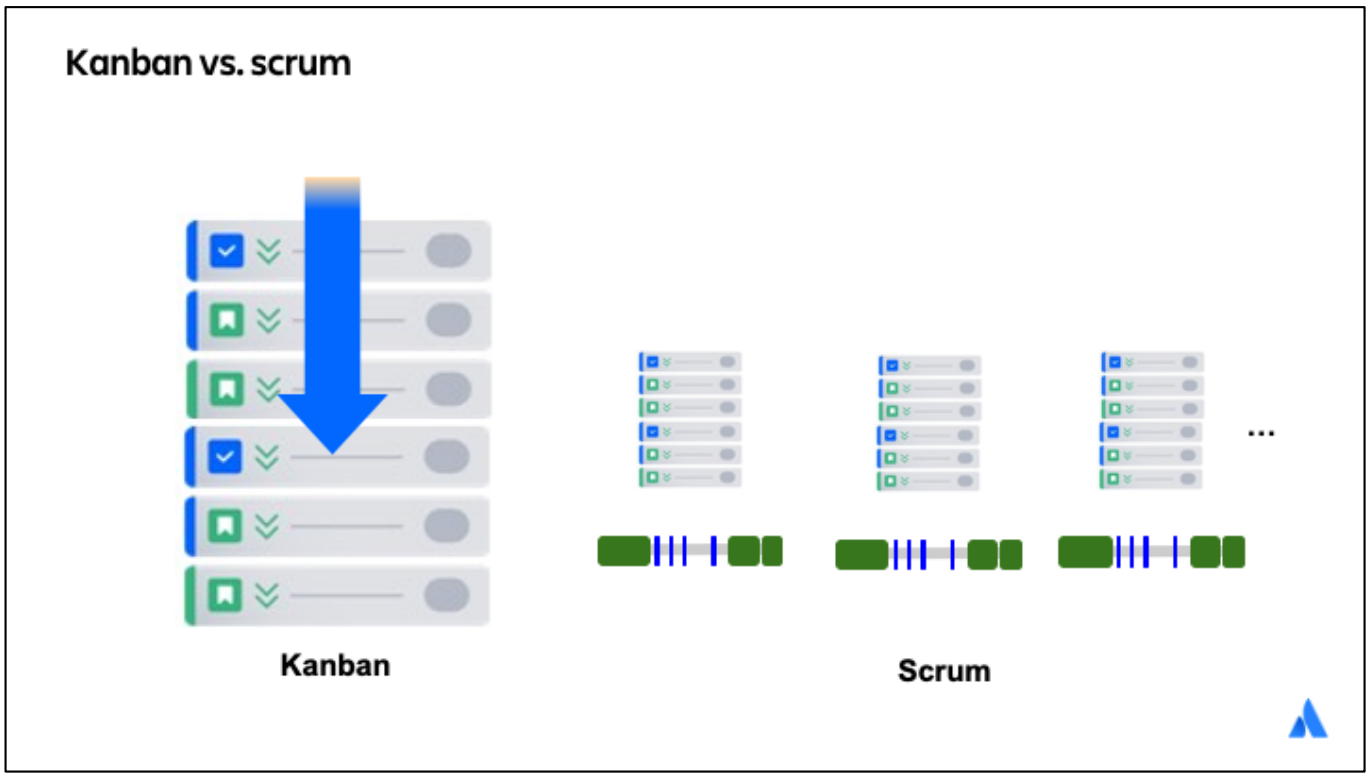
The sprint retrospective is the last event of the sprint.
The scrum team attends the retrospective. This meeting is often facilitated by the scrum master.
A retrospective typically takes 90 minutes for a two week sprint.
The purpose of the meeting is for the team to inspect itself, including its processes, tools and team interaction. The retrospective is a positive meeting containing constructive feedback. Everyone should always remember that they are a part of a team. The scrum master usually helps make sure that this is a positive meeting. A retrospective can have a celebratory feel, because the team should feel good about having delivered the increment. In a retrospective, the team usually discusses what they should keep doing, what they should stop doing and what they should start doing. The meeting is about continuously improving the team.
The output of the meeting is to add one or more improvement related issues to the next sprint's backlog. It's important that the team spends some of its time on these type of issues, rather than exclusively building the product.
The link at the bottom is from the Atlassian team playbook on retrospectives. You can find more information there.

Scrum meetings- summary

	Sprint Planning	Daily Standup	Sprint Review	Retrospective
Attendees	Scrum team	Development team (primarily)	Scrum team and stakeholders	Scrum team
Duration*	4 hours	15 minutes	2 hours	90 minutes
Purpose	Plan the work of the sprint	Inspect recent work, plan today	Inspect increment, brainstorm next sprint	Inspect team
Output	Sprint goal, Sprint backlog	Today's plan	Proposed next sprint backlog	Amended next sprint backlog

* typical max duration assuming a two week sprint

Here is a summary of the four meetings related to a sprint. It's important to always make sure that the meetings have high value, and the team should focus on continuously improving their meetings.
Notice that the next sprint goal and sprint backlog start to form in the sprint review. The retrospective usually adds one or more issues. This forms the starting sprint backlog for the sprint planning meeting.



Kanban is a smaller framework than scrum. Kanban does not define as much of the agile process as scrum does. For example, it is up to the kanban team to decide which ceremonies/events to use (if any). With kanban, a continuously prioritized backlog of work is presented to the team. When a team member is ready for more work, they pull a work item. A kanban project board contains an ongoing queue of items.

With scrum, the work of the team is divided into batches of work items that are completed during sprints. This is kind of a compromise between working with large batches and working with a continuous queue. A sprint contains four types of meetings. It also contains artifacts such as a sprint backlog, sprint goal and sprint board. A sprint board only contains the work items to completed during the sprint. Like with a kanban project, when a team member is ready for more work, they pull a work item.

It is up to the team to decide whether to use kanban or scrum, some combination of them, or another agile framework/method. If the nature of the work is more of a queue, like with support requests, kanban might be a better choice. For teams that are building a product, scrum might be a better choice, though some teams could be more effective using kanban. With kanban, the team will have to decide on things like when and why to meet.

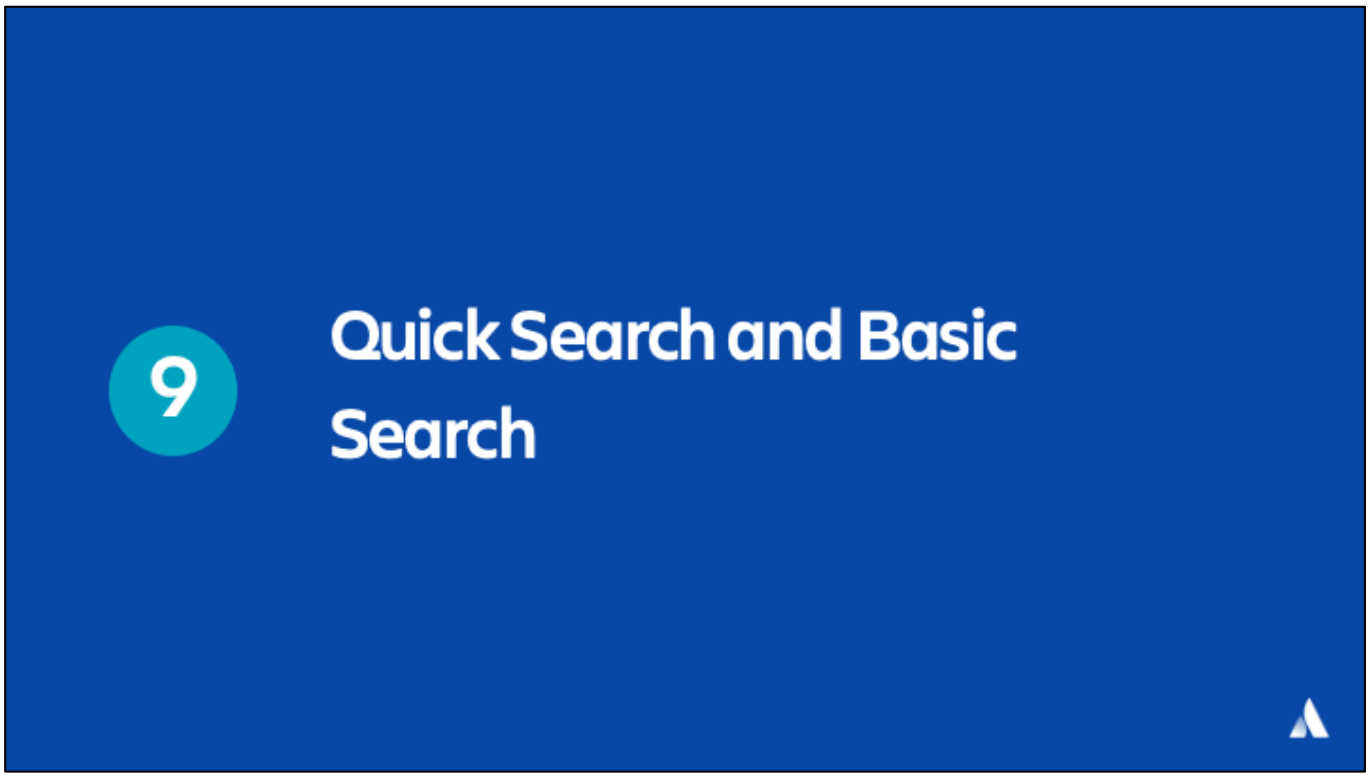
Takeaways

- Scrum roles include:
  - product owner
  - scrum master
  - development team members
  - stakeholders
- Scrum meetings include:
  - sprint planning meeting
  - daily standups
  - sprint review
  - sprint retrospective

Tasks

Scrum

- ❑ Create a scrum project
- ❑ Create issues in the product backlog
- ❑ Create and plan a sprint
- ❑ Execute a sprint
- ❑ Complete a sprint


9
Quick Search and Basic Search

What will you learn?

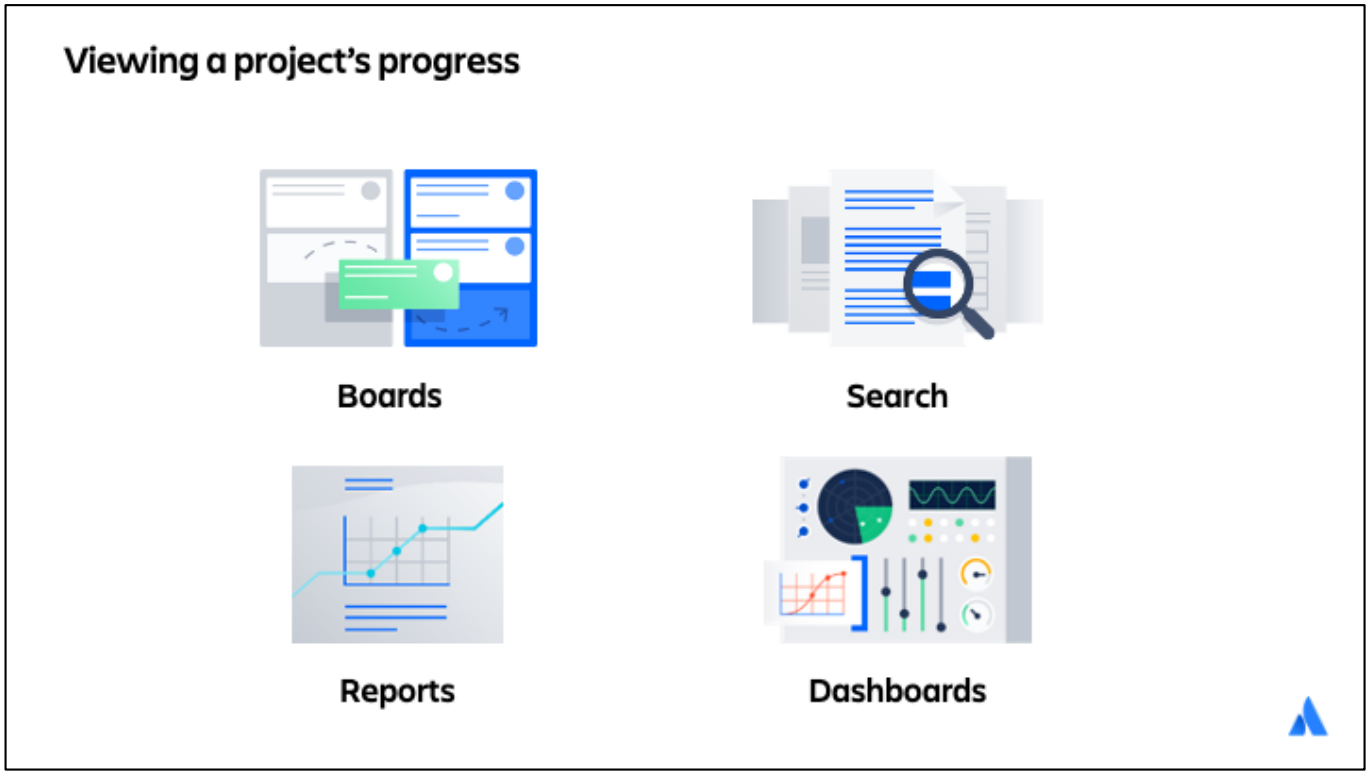
- Identify the ways to search in Jira
- Use quick search
- Use basic search

Topics

Searching overview

Quick search

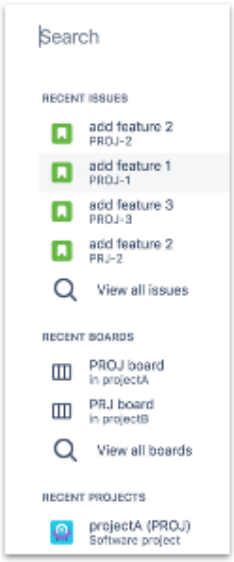
Basic search



Project boards, searching, reports and dashboards help your team view the current and historic status of the project, as well as help plan the project. Using these tools, you can answer most questions that are relevant to the team, either in an ongoing manner or with specific just-in-time team questions. We will cover searching now, but the same techniques are used behind the scenes for reports and dashboards.

Searching

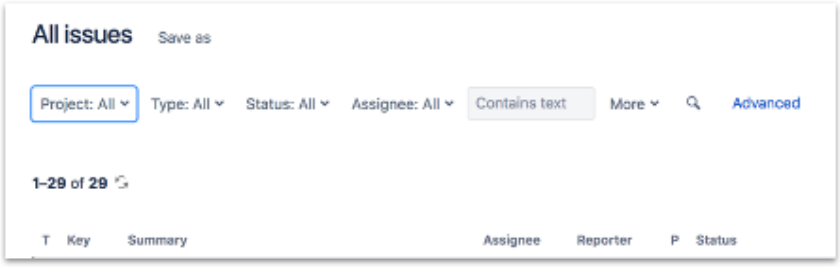
1. Quick search



There are several topics related to searching in Jira. We will discuss them quickly now, and then discuss them in more detail in this and the next module. The first is called a quick search. This allows you to quickly search for text in issues, board names, project names and filter names using a text-based search. You can also quickly access recently used items, as shown here.

Searching

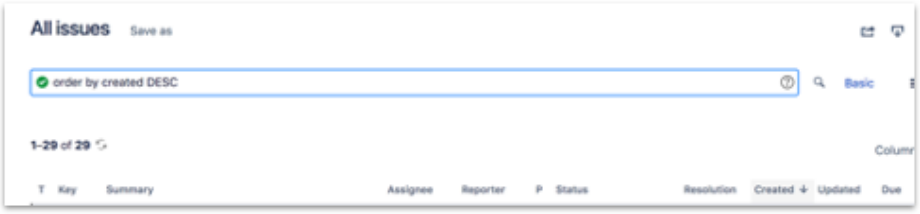
1. Quick search
2. Basic search (user interface elements)



Another type of search is called basic search. In a basic search, issues are searched for using a row of user-friendly interface elements, as shown here. Changes to these elements will change the resulting list of issues.

Searching

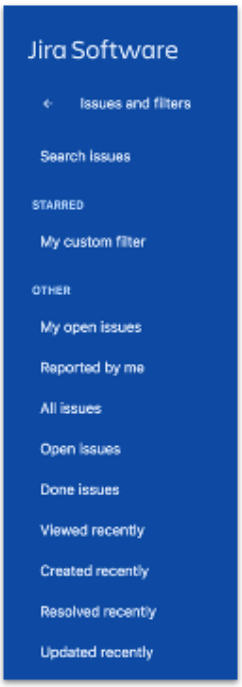
1. Quick search
2. Basic search (user interface elements)
3. Advanced search (JQL)



Advanced search is an alternative to basic search, where you can specify your search use Jira Query Language, or JQL. This is a way to execute more complicated searches than with basic search.

Searching

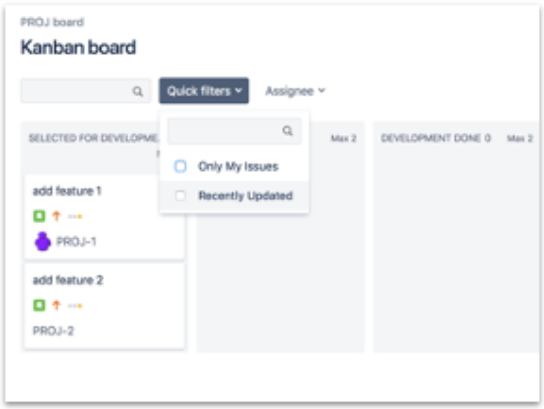
1. Quick search
2. Basic search (user interface elements)
3. Advanced search (JQL)
4. Filters



Filters are used to limit the issues that are displayed in search results. These are the default filters provided by Jira. For example, if you click on the Open issues filter, the results will show all issues in all projects that have not been closed. You can also add custom filters, such as the filter called My custom filter here. Even though filters are usually presented as user interface elements, they have an underlying JQL statement associated with them. That JQL statement defines which issues to show.

Types of searching

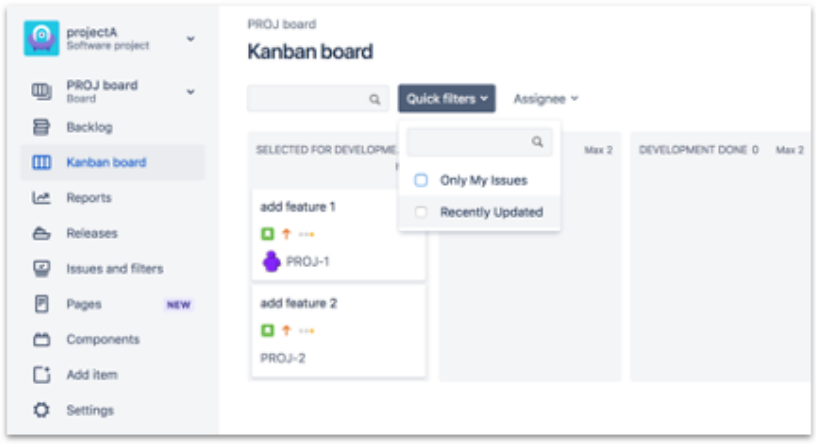
1. Quick search
2. Basic search (user interface elements)
3. Advanced search (JQL)
4. Filters
5. Quick filters



The final search topic that we will discuss is quick filters. Quick filters are used to limit which issues are shown on a board. Here are the two out of the box quick filters, used to show only your issues or recently updated issues. You can also add custom quick filters to the board. Like standard filters, quick filters have an underlying JQL statement associated with them.

Why is searching important?

Adapt your Jira experience to your team's processes



We are discussing searching, JQL and filters because they are used to help adapt your Jira experience to your team's processes. We have seen that each agile team is unique and constantly improving, so the tools that are used need to be flexible enough to adapt to changing circumstances.

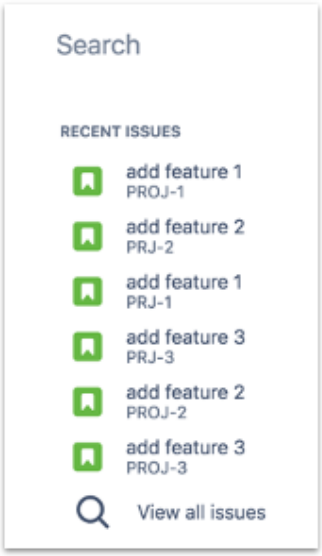
Topics

Searching overview

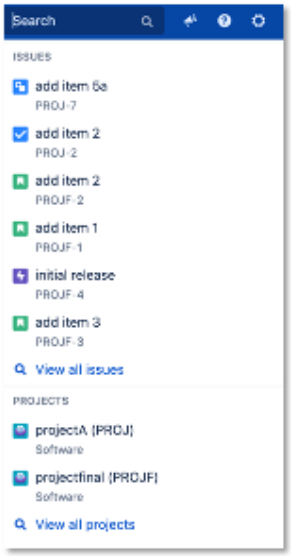
Quick search

Basic search

Quick search



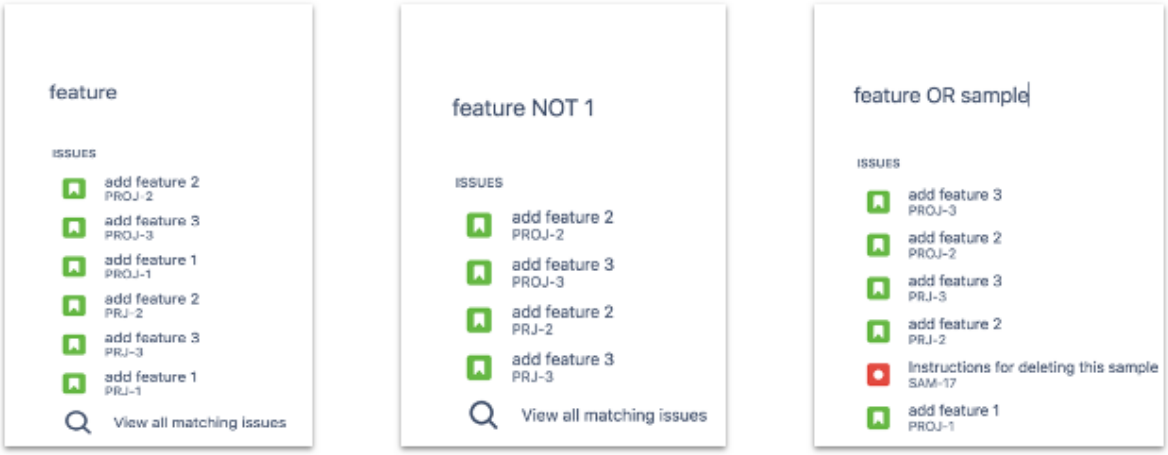
Cloud



Server

When you initially open the quick search, the results show recent items. You can click on an item to display it.
Quick search does not return just issues. When you open quick search, you can also access recent boards, projects and filters. When you perform a search, all of these items that match will be displayed.

Quick search- with search terms and keywords



https://confluence.atlassian.com/jiracorecloud/search-syntax-for-text-fields-765593720.html

Usually you will just be entering simple text when searching with quick search. You also have the option of adding slightly more sophisticated syntax. As an example, on the left, we are searching for the text "feature". You can see that this text is in the summary for six issues in our Jira account.
We can exclude feature one by adding a capitalized NOT followed by the number one. You can see that the two issue containing feature one are excluded from the results. In the search, NOT must be all caps. The search terms are not case sensitive, but text-field search keywords such as NOT, OR and AND must be capitalized.
In this example, we use the capital OR keyword to search for "feature" or "sample".
For more information on this type of searching, see the link shown here on search syntax for text fields.

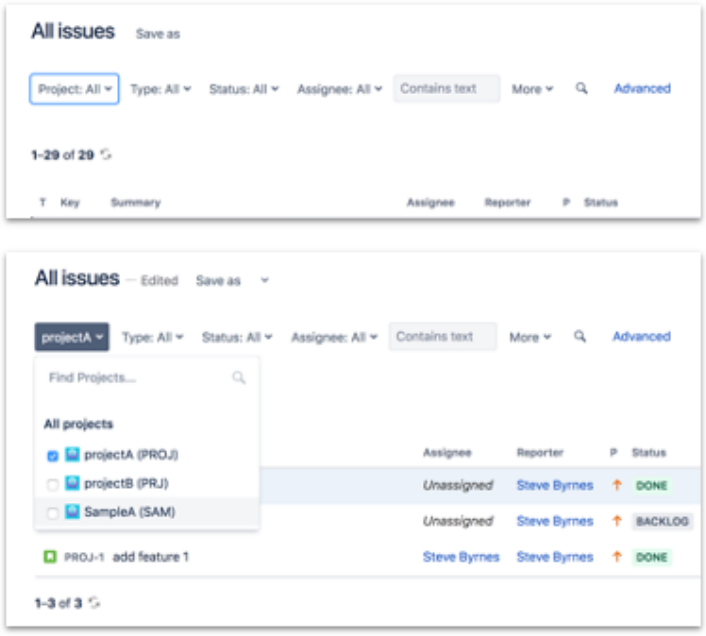
Topics

Searching overview

Quick search

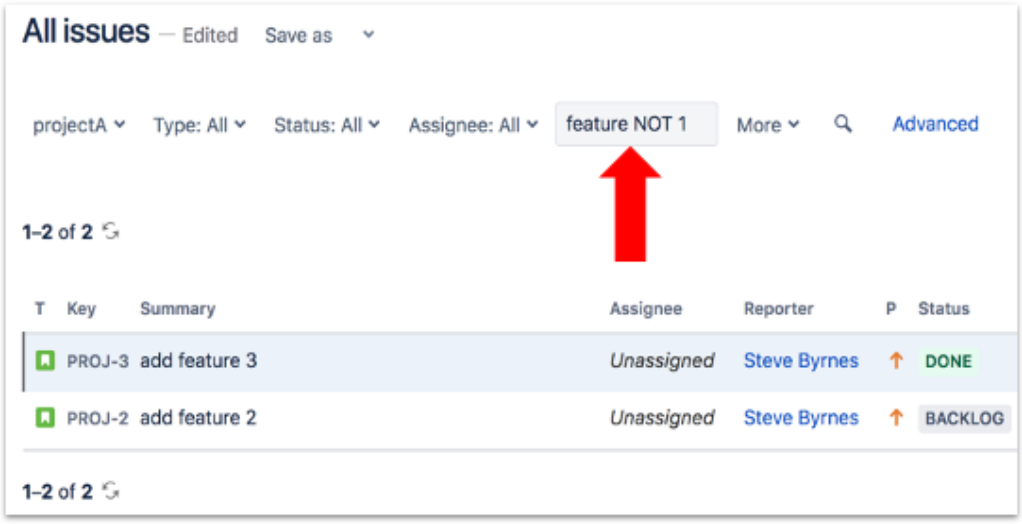
Basic search

Basic search



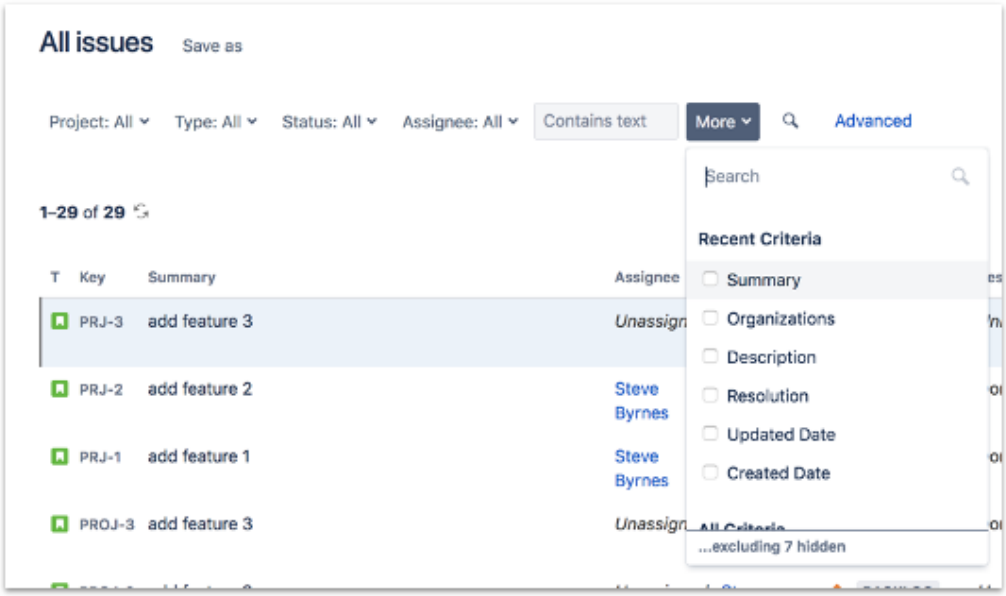
Basic search uses user-friendly interface elements in the issue navigator to search for issues. This row of elements shows that we are in the basic search. We could click on the Advanced link to switch to the advanced search. Likewise, if we are in the advanced search, we can click on the Basic link to switch to basic search.
You can see that this search searches all projects for issues of any type in any status and with any assignee. This is searching for all issues in the Jira account. You can see that there are currently 29 issues in this account, assuming that we have permission to see all issues.
If we click on the project dropdown and choose project A, you can see that three issues are returned.

Contains text



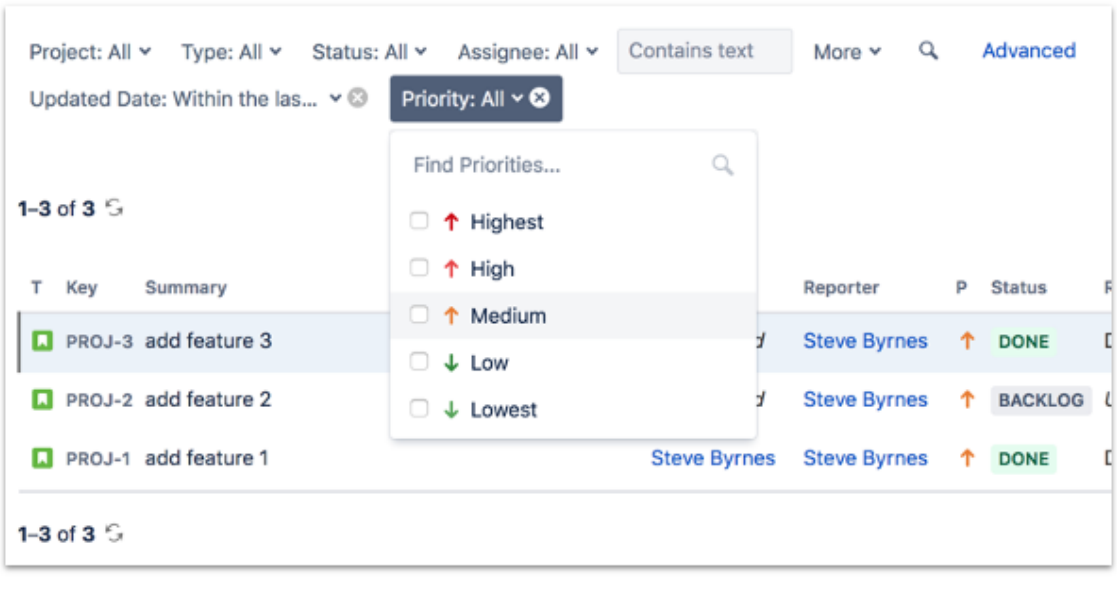
Notice that basic search includes a text box. This behaves much like the quick search, but is limited to searching issues. If you enter text in this textbox, Jira will search in fields that typically hold text, such as summary or description.
In this example, we have changed the project dropdown to project A and added "feature NOT 1" to the "contains text" box. This will filter the "add feature 1" issue from the project, and you can see that we are left with two issues. You can see that NOT in the quick search can also be used here.

Searching more fields



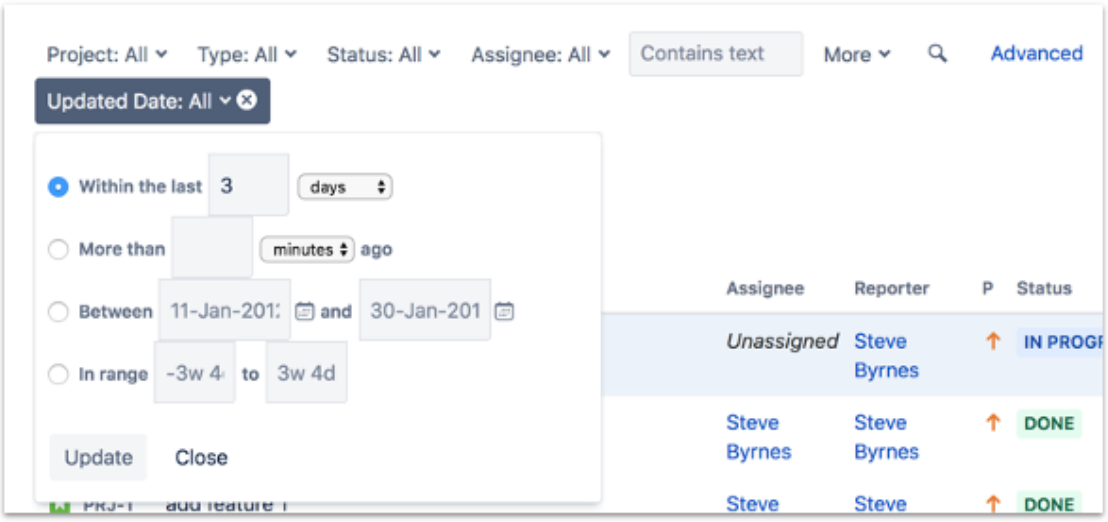
If you want to search for values of fields that are not listed in basic search, you can click on the More dropdown to add searches for other fields. We will click on the updated date checkbox to add a search related to the day that an issue was modified.

Specifying a priority



Here we have selected priority from the More dropdown menu. Notice that Jira will automatically provide the appropriate interface depending on the type of field chosen. We have just seen that when we selected the updated date field, we were provided an interface related to dates. In this case, we have selected the Priority field, so Jira provides us a set of priorities to choose from.

Specifying an updated date



Here we have clicked on Updated Date, and you can see that Jira adds a dialog allowing you to specify the date values to search for. We have chosen to search for any issues that have been updated within the last three days.

Takeaways

- Quick search can search the text of issues, board names, project names and filter names
- Basic search is a user-friendly way to search for issues

Tasks

Quick Search and Basic Search

- ❑ Perform quick searches
- ❑ Perform basic searches


10
JQL

What will you learn?

- Describe Jira Query Language (JQL)
- Write JQL using autocomplete
- Use functions in JQL queries

Topics

JQL overview

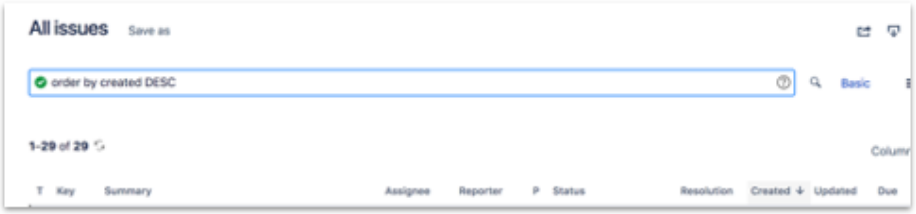
Autocomplete

Functions

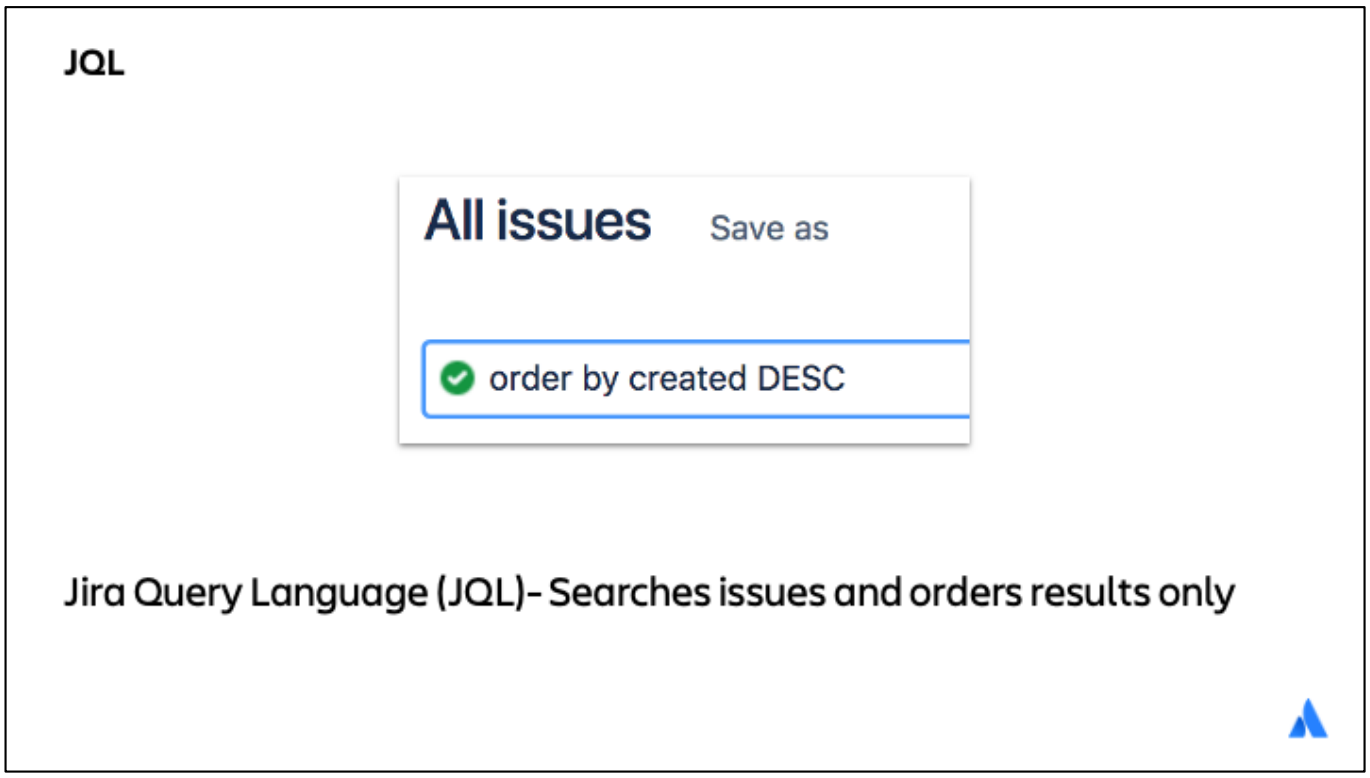
Basic and advanced/JQL search

basic

advanced/JQL

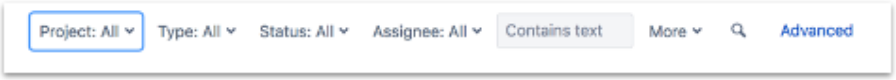


Previously, we explored basic search. In this example, we are searching for all issues that we have permission to view in the Jira account, because none of the search elements are being selective. On a basic search, there is an Advanced link, which allows you to switch to the advanced mode of searching for issues.
The bottom image shows advanced search. Jira will automatically populate the textbox with a text-based equivalent for the current basic search. This text says "order by created DESC". In other words, search for all issues in this Jira account and sort the resulting list by the value of the created field. The created field holds the time and date that the issue was created. Descending means to show the newest issue at the top of the resulting list and the oldest issue at the bottom. You can click the Basic link to switch back to the basic search.



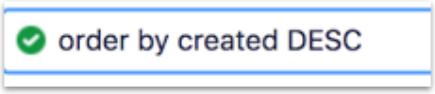
That text that we saw on the advanced issue search is called Jira Query Language, or JQL. JQL is used to search issues and order the results only.

Basic vs. advanced/JQL search



Basic search

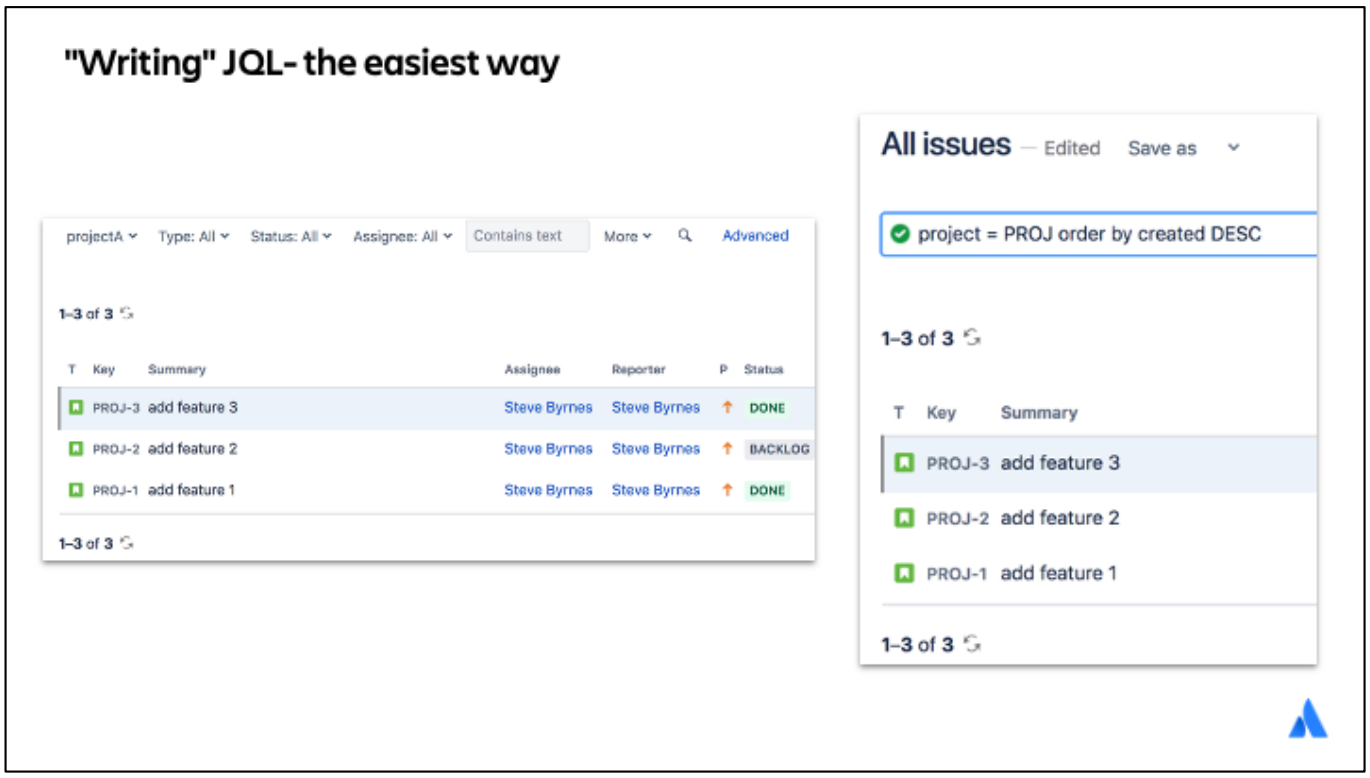
- User-friendly interface
- Queries can be complex, but there are limits



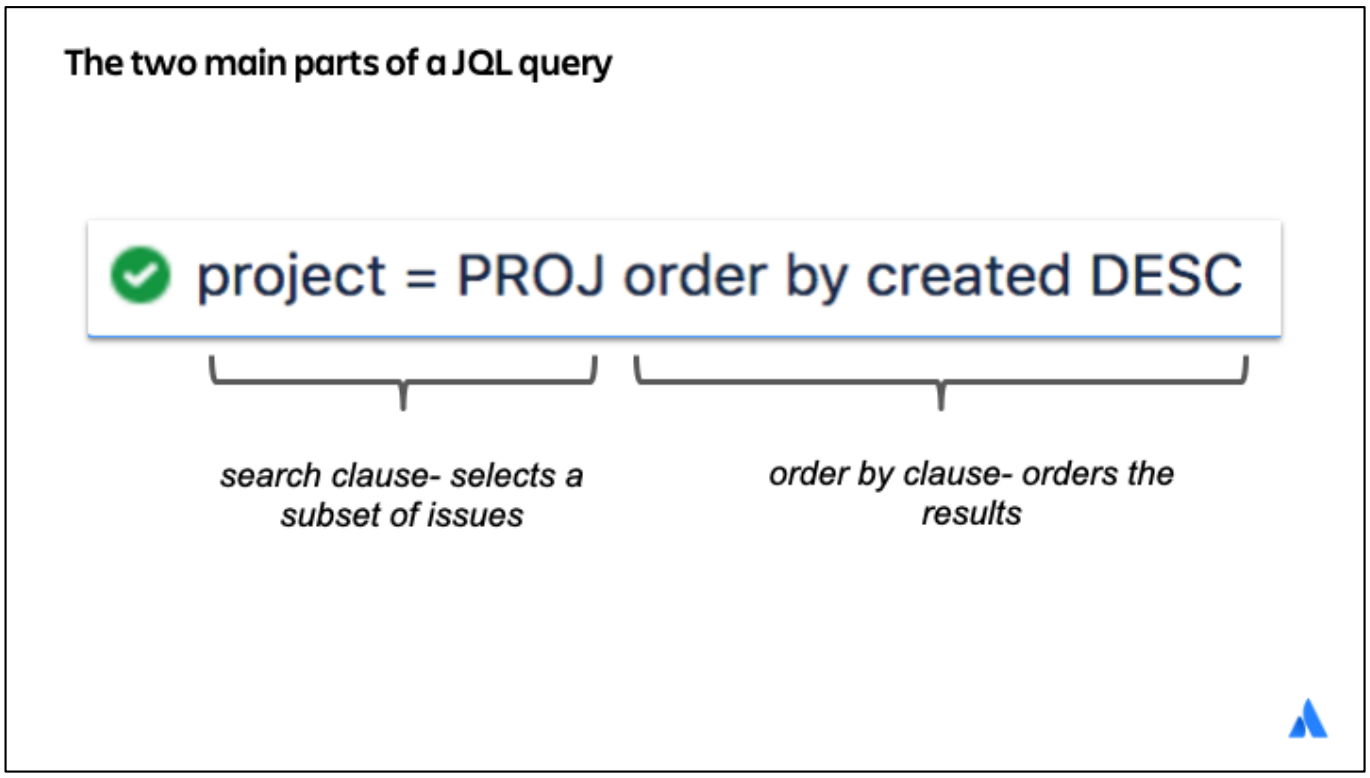
Advanced/JQL search

- Uses JQL
- Most powerful search method
- JQL can be used in automation scripts

We have seen that the basic search contains user-friendly interface elements to perform the search. We have also seen that the queries can be quite complex by selecting from multiple elements or adding text-based searching.
Advanced search uses JQL and allows for more powerful searches because it is not limited by the capabilities of the user interface elements of basic search. Also, if you are automating anything related to Jira, you would use JQL, because automation relies on text-based scripts.
Whether you are using basic or advanced search, the JQL related to the search is always there behind the scenes.

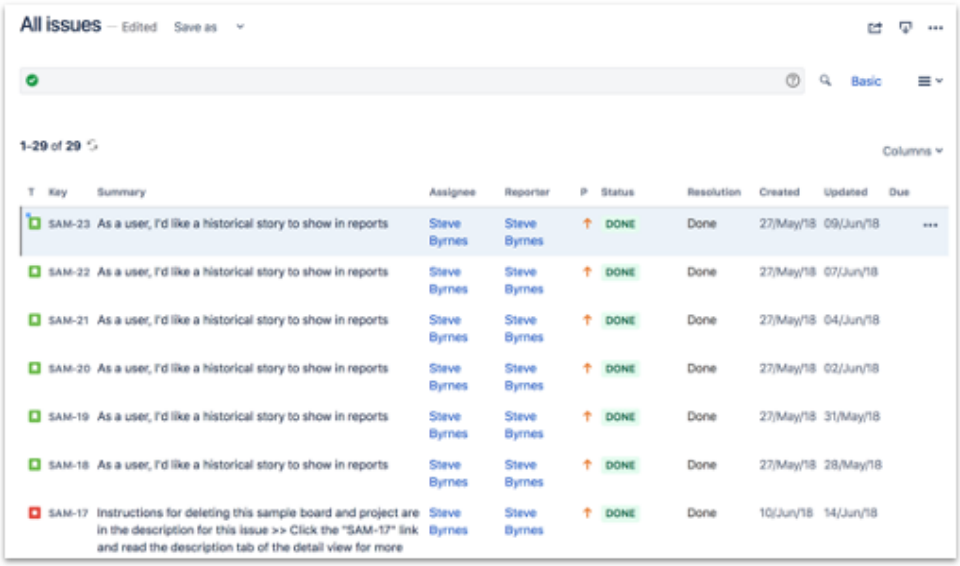


The easiest way to write a JQL query is to not write it at all. You can let Jira write it for you. Here we are in basic search and select project A from the dropdown. We can see that project A has 3 issues. We can then click on the Advanced link to enter advanced search.
The search results don't change at all, but a query is displayed instead of the basic search user interface. That query displays all of the issues with a project key of PROJ. The results are ordered by the date that the issue was created.



JQL queries can fundamentally do two things, and we see both of them in the query that was just created. The search clause can select a subset of issues, as we see in the first part of the query. We are selecting only issues in which the project field has a value of PROJ. And they can sort the results using an “order by” clause. We are ordering the results by the date that they were created.

The simplest JQL query



The simplest JQL query is an empty string, as shown here. This means that we are not being selective about which issues that we see, so we see them all. You can see that there are 29 issues in this Jira account, assuming that the current user has permission to see all of the issues. We don't specify how we would like the results to be ordered, so Jira picks the default ordering. It looks like the default order is by the issue key in a descending order.

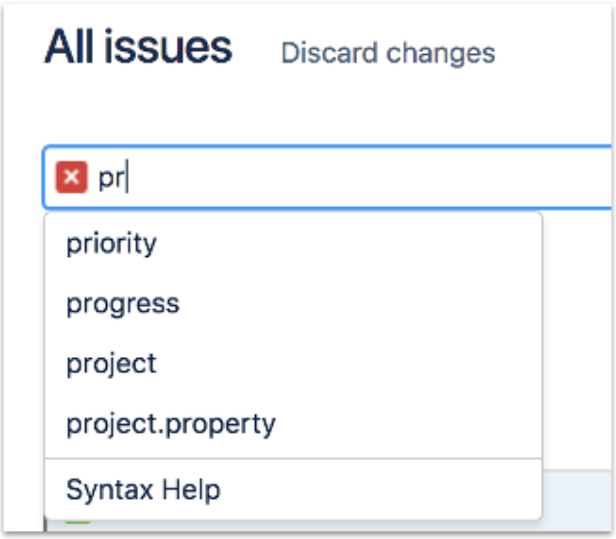
Topics

JQL overview

Autocomplete

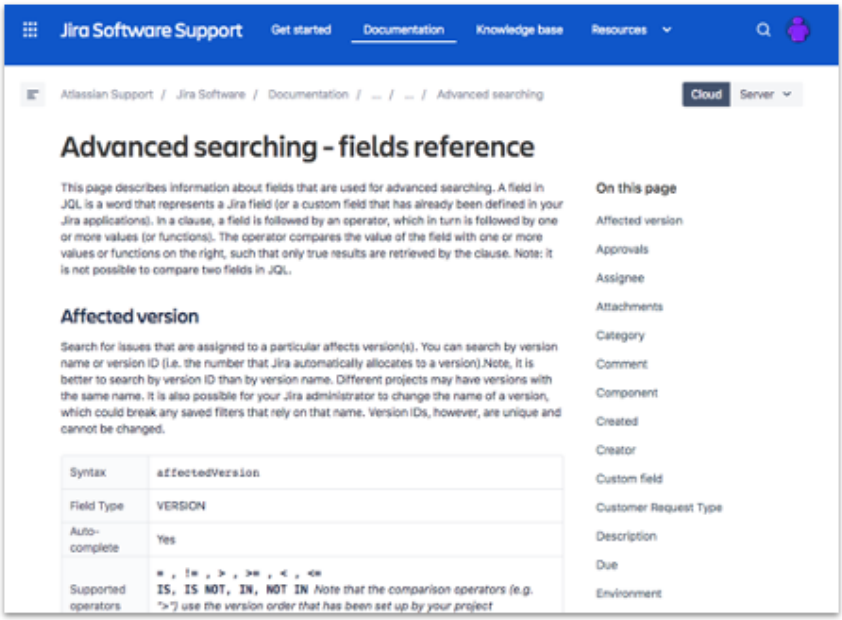
Functions

JQL with autocomplete- fields

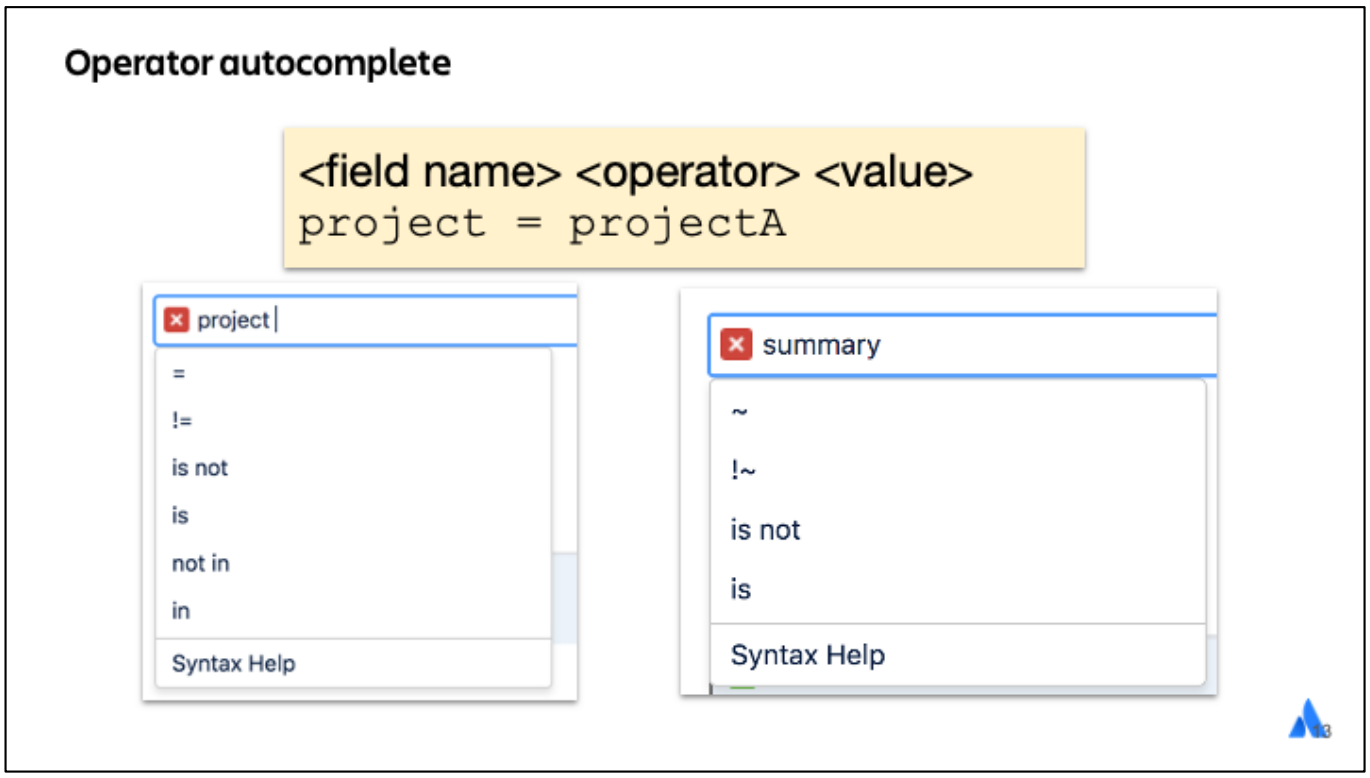


We have the option of creating a JQL query from scratch. Let's create a query that searches for the issues in a project. Project is a field name in a Jira issue and each issue is assigned to a single project. In the advanced search text box, we can begin typing "pr" and Jira will help us with autocomplete suggestions. Autocomplete will show up to 15 matches. This makes writing JQL much easier and helps avoid mistakes. In this case we select the "project" field as the start to the query.
Notice that there is a Syntax Help option at the bottom of the list. This will point you to the documentation for advanced search.

Advanced searching fields reference



In the previous example, we knew that we were searching for the project field. If you are unsure of which field to search for, or to get more information related to the field, such as supported operators and functions, search the web for the advanced searching fields reference documentation from Atlassian.



An operator is placed between the field name and the value. In this example, the operator is the equals sign.
You can use autocomplete to see which operators that you can use for a specific field name. In this example, we have entered the project field name and then pressed the space bar. Autocomplete provides a list of choices for the operators.
In this example, we have entered the summary field name and then pressed the space bar. Notice that Jira provides a different list of operators for the summary field. For example, the project field accepts the equals sign as an operator and the summary field accepts the contains operator, which is represented by a tilde (dil-duh). Jira's autocomplete provides the acceptable operators based on the expected type of field value. The project field expects a project name and the summary field expects a free form of text.

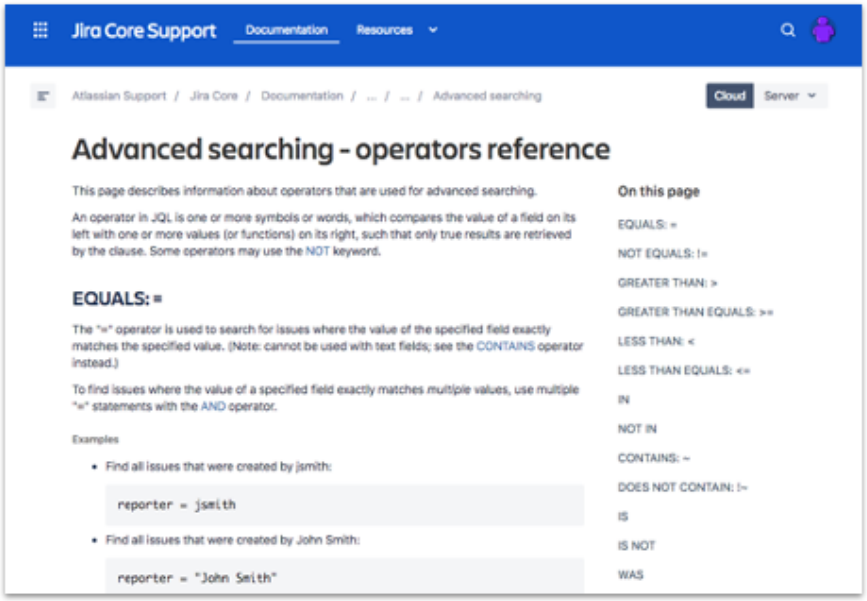
Boolean operators

- AND
- OR
- NOT

```
assignee = currentUser() AND status = "In Progress"

status = "Selected for Development" OR status = "In Progress"
status in ("Selected for Development", "In Progress")

NOT status = Backlog
status != Backlog

find unresolved issues in all projects except SampleA
resolution = Unresolved AND NOT project = SampleA
```

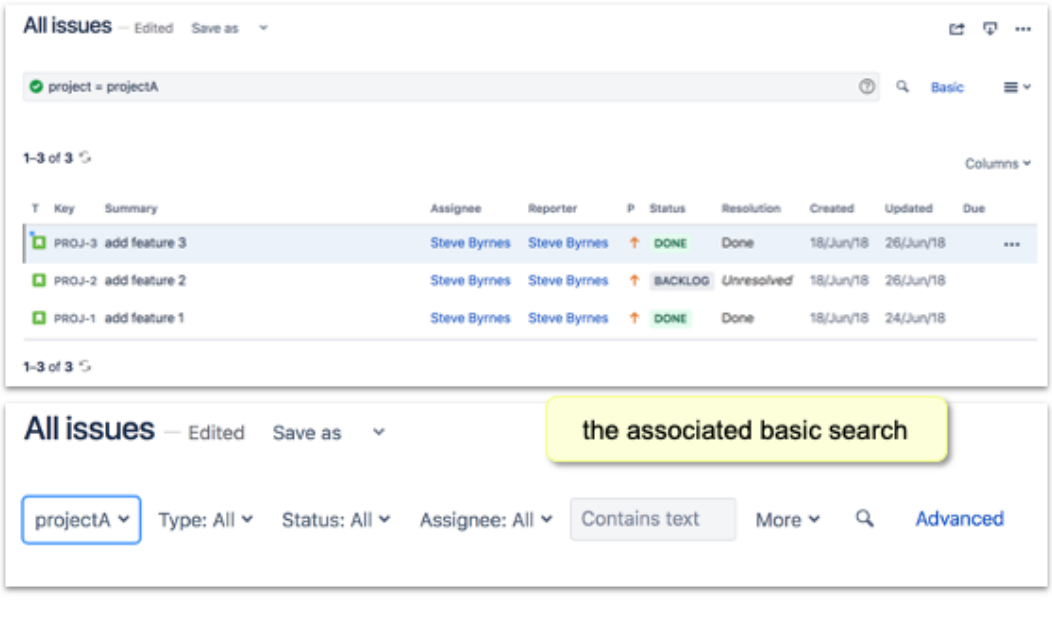
Boolean operators are used to either combine or negate clauses. The AND and OR boolean operators are used to combine multiple clauses, allowing you to refine your search.
For example, the first query finds all issues assigned to the current user with a status of "In Progress".
The second query finds all issues with a status of "Selected for Development" OR "In Progress". The third query is equivalent to using the in operator.
The NOT operator is used to negate one or more clauses.
For example, the fourth query finds issues that do not have a status of "Backlog". This is equivalent to using the not equals operator.
In the final example, we find unresolved issues in all projects except for the SampleA project. Notice that we are combining AND and NOT boolean operators. You could do the same this with OR and NOT boolean operators.

Advanced searching- operators reference



The advanced searching operators reference documents the details related to operators. We will briefly go over operators in this video, but this is mostly to provide you awareness of what is possible with operators in queries. The operators reference is very helpful as you are writing queries.

Executing the query



When we complete the query and press Enter, the search is performed. Here we can see that there are three issues in project A. At this point, if we click on the Basic link, we can see that project A is selected in the dropdown list. This is equivalent to the JQL that we just created.

Topics

JQL overview

Autocomplete

Functions

Search clauses

<field name> <operator> <**field value**>

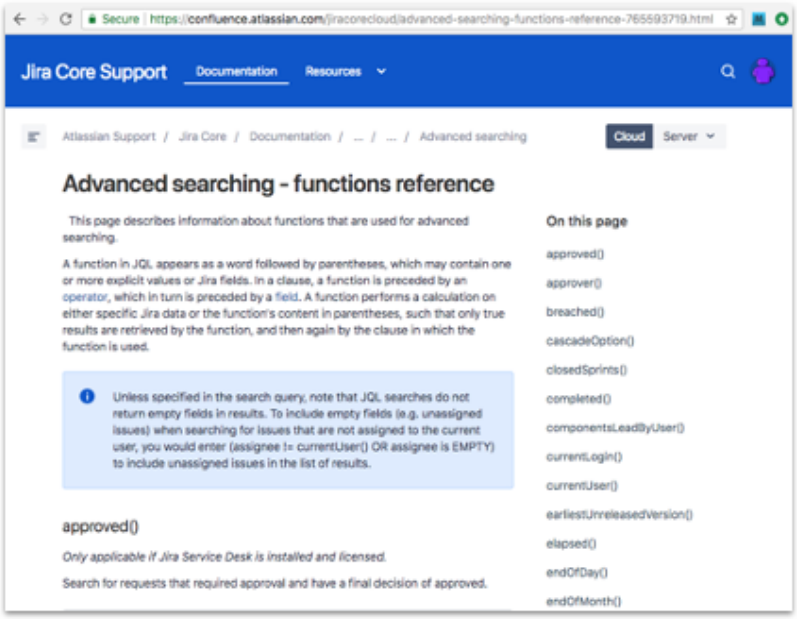
```
project = projectA
```

<field name> <operator> <**function**>

```
assignee = currentUser()
```

You usually only want to return a subset of issues from a search. A search clause is used to limit or filter the issues that are returned in the results. The basic structure of the clause is a field name followed by an operator followed by a field value. We have seen an example where we are selecting all issues that have a project field value of "projectA".
Instead of directly providing a field value in the search clause, you can provide a function. A function is a small program that Jira calls before the query is executed. The result of calling the function is then substituted in the query. As an example, this clause will search for issues that have been assigned to the currently logged in Jira user. A big advantage of using functions is that you are not hard-coding information. This same query can be used by all users to view their issues.

Advanced searching- functions reference



The Jira advanced searching functions are documented so that you can use them if and when you need them. Do a search for advanced searching- functions reference to find it on Atlassian's web site. You can then find the details related to any functions that interest you.

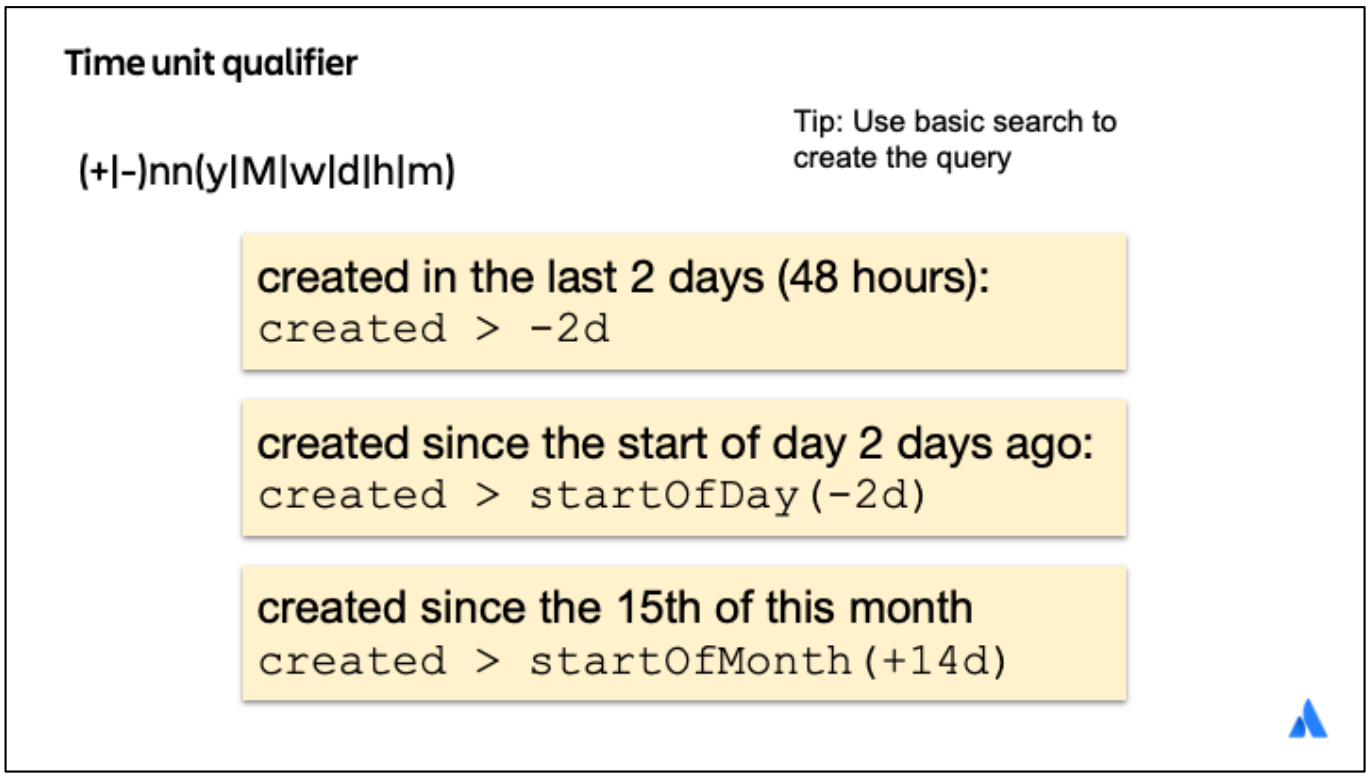
Time-based functions

- startOfDay()
- startOfWeek()
- startOfMonth()
- startOfYear()
- endOfDay()
- endOfWeek()
- endOfMonth()
- endOfYear()
- now()
- currentLogin()
- lastLogin()

Issues created since the start of today

```
created > startOfDay()
```

Here is a list of the advanced search functions that are related to time and date.
There are functions that specify the start of a time period and end of a time period.
The now function represents the current time. The currentLogin function results in the time that the current user logged into this session and the lastLogin function results in the time that the current user logged into the previous session.
As an example, this query will find issues created since the start of today.



If the field is related to dates, you can add what's called a time unit qualifier to specify relative dates as field values. This is a specially formatted string that Jira will replace with an actual date value before running the query. In this example, we are searching for issues created in the last two days.
The time unit qualifier is handy because it is simple, and the query doesn't have a hard-coded date.
At the tops, you see the syntax that can be used to specify time units. It starts with a plus sign, a vertical bar and a minus sign in parentheses. The parentheses means that what is inside is optional. The vertical bar means or. If you add a minus sign to the string, you are searching back in time. Using the plus sign, or leaving off the sign altogether, means that you are searching forward in time. You can see that this query uses the minus sign, so this query is searching for issues that were created 2 days ago or later.
The nn in the string represents numeric digits. In this string, the numeric digit is 2. Following the numeric digits is the optional time period unit. You can specify y for a year, capital M for a month, w for a week, d for a day, h for an hour or lowercase m for a minute. In this example, we specified d, so this query is searching for issues created within the last 2 days. We could change that d to a w, for example, to search for issues that were created in the past 2 weeks. If you leave off the units, Jira will assume a logical default unit, which depends on the circumstances.
You can also use time unit qualifiers as arguments to a function. An argument is a value that is passed to the function to change its behavior. If the trailing parentheses of a function call are empty, you are not passing any arguments and the function is

called with its default behavior. In this example, we add an argument to change the function so that we are searching for issues created since the start of day two days ago. If today is Wednesday, this would be searching for issues created Monday or later.
At the bottom is an example of finding issues created the since the 15th of this month, assuming we are currently in the second half of the month. Notice that the function is related to months, but we are passing an argument related to days. The time periods do not need to match.

Takeaways

- A JQL query is behind all basic and advanced searches
- Leverage basic queries and autocomplete to simplify creating JQL queries
- JQL queries may select subsets of issues and/or order query results
- Functions can be used to avoid hard-coding values in a search clause
- Time unit qualifiers (yMwdhm) can be used with date-related values

Tasks

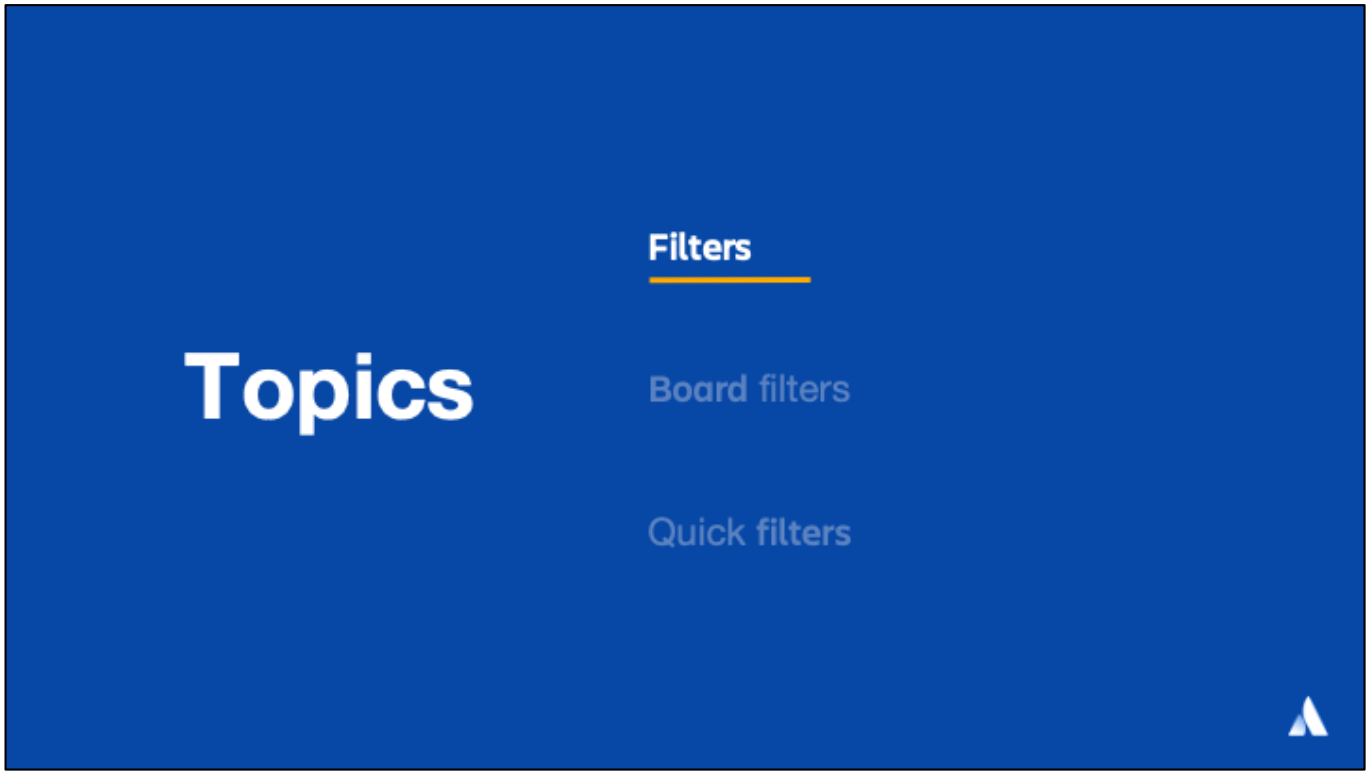
JQL

- ❑ Create a basic search and view the JQL query
- ❑ Create JQL queries using autocomplete
- ❑ Use functions as values

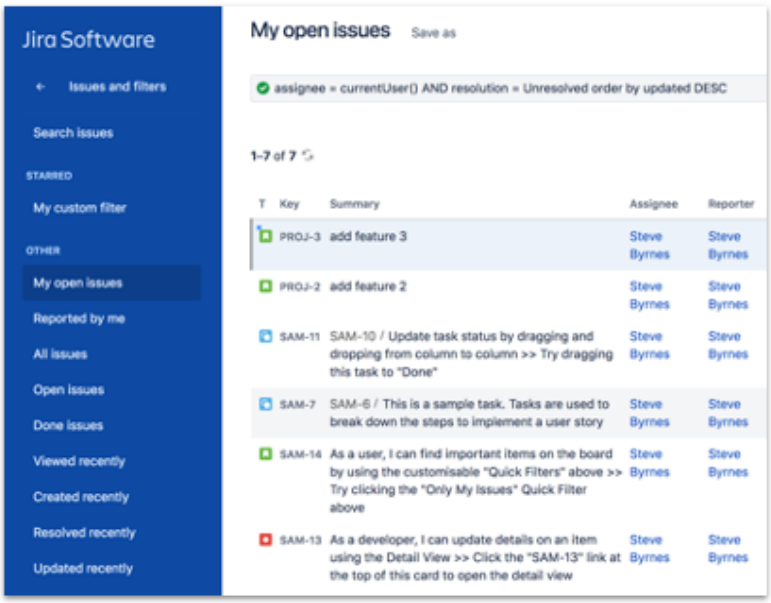
11 Filters

What will you learn?

- Create filters
- Describe board filters
- Use quick filters

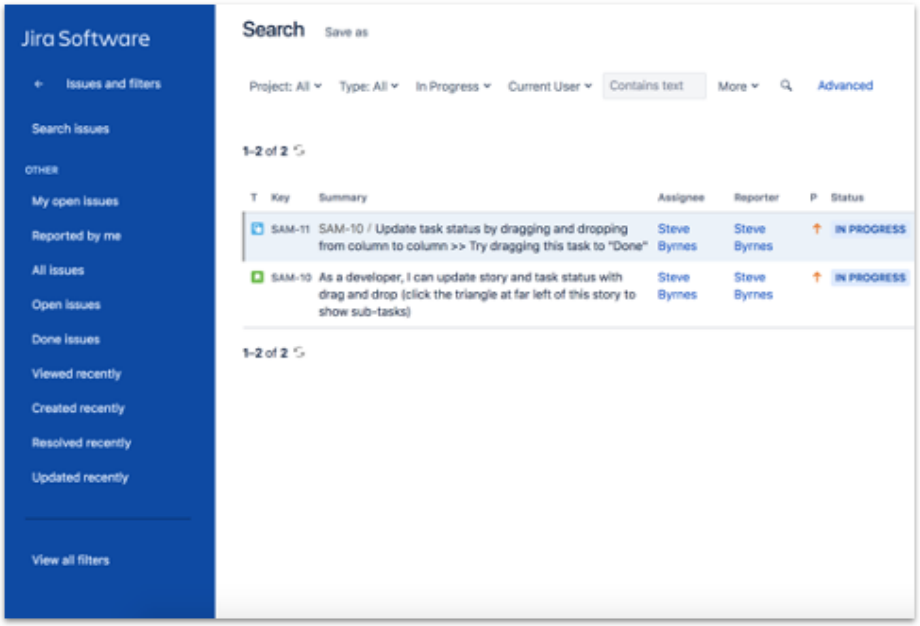

Topics
Filters
Board filters
Quick filters

Filters



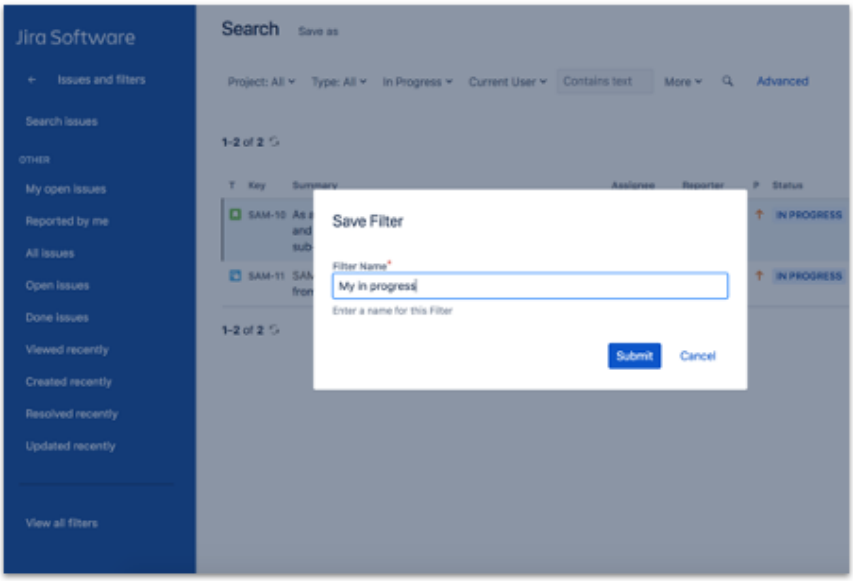
If you open the global issue navigator, you are presented with search-related tabs, as shown here. Each of these tabs is a filter. Filters are saved searches that can be exposed through user interface elements. Filters are a handy way to execute common searches. If you click on the tab, the search is executed, and you can see the query that was used for the search.
In addition to the default filters that Jira provides, you can create custom filters, such as the "My custom filter" that was created here.

Save a search



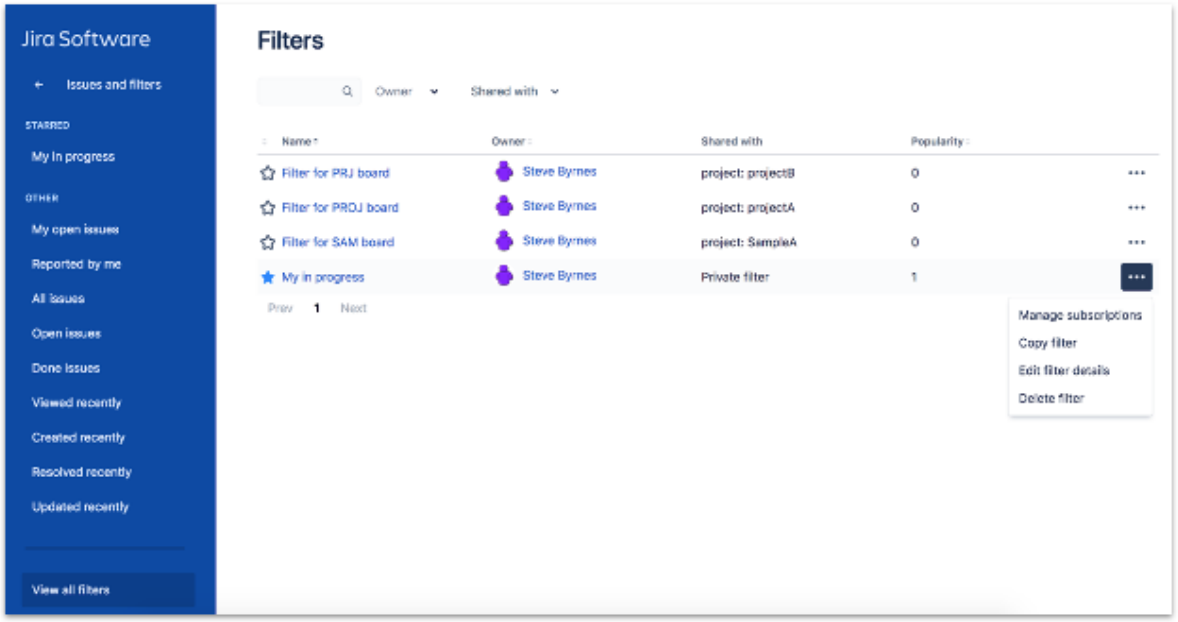
Saving a search creates a filter. Start by creating and executing a query in either basic or advanced search, then click on the Save as link to begin the process of saving the filter. In this example, we have created a search that selects issues that have a status of In Progress and that the current user owns. We then click on the "Save as" link to create the filter.

Save the filter



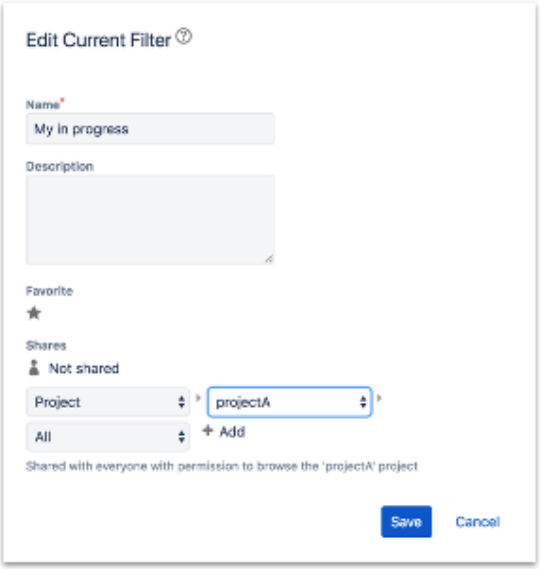
We will name this filter "My in progress".

View all filters



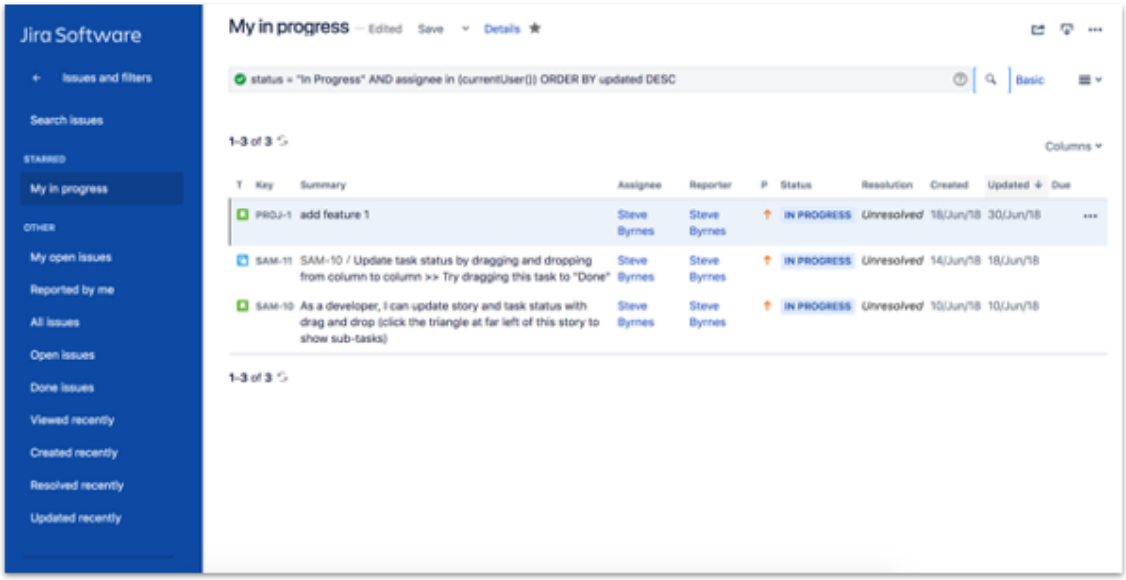
Once you have created the filter, it will show in global issue navigator under the "starred" category. Also, if you click on View all filters, you can see your newly created filter in the list. By default, the filter is a private filter, meaning that it is only accessible to you. You can click on the more icon to the right of the filter to change metadata related to the filter. Select Edit filter details to edit it.

Edit filter metadata



On the Edit Current Filter screen, you can change the name of the filter, add a description, add or remove it from your favorite filters, and share the filter with others. Here we are sharing the filter with the members of the project A team. They will see the filter in their sidebar. Filters are application-wide entities in Jira. That is why they are configured in the application-level sidebar. The extent to which a filter is visible to users depends on the extent that they are shared using this screen. Depending on the permissions of the user creating the filter, they can choose to keep the filter private, share it with only certain projects or groups of people, or expose the filter to all users of the account.

Edit filter query



To edit the query for a filter, execute the original filter so that you can view the query. Modify and execute the query. When the query is modified, you will see the Edited indication next to the filter name and can click the Save link to overwrite the existing query.

Topics

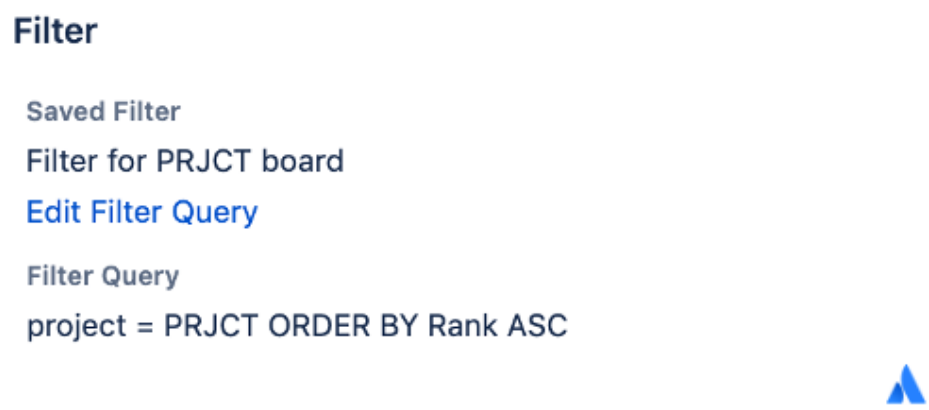
Filters

Board filters

Quick filters

Board filters

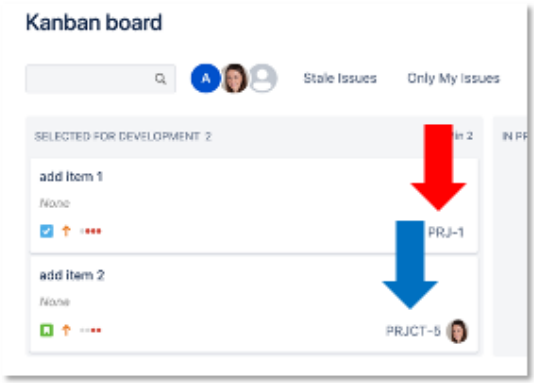
- Every board has a filter that defines the issues shown on the board
- You can edit the board's filter
- If you create a board, you must assign it a filter



A typical board for a project contains a filter that only shows issues for the project using a "project = xxx" clause. There may be other filters (such as sub-filters) associated with the board, further limiting the issues shown on the board.

Example board filter

A board filter can be used to show issues from multiple projects on a single board



Topics

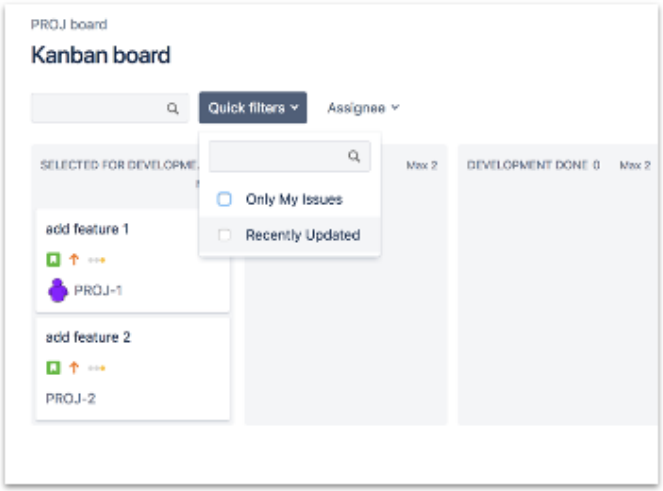
Filters

Board filters

Quick filters

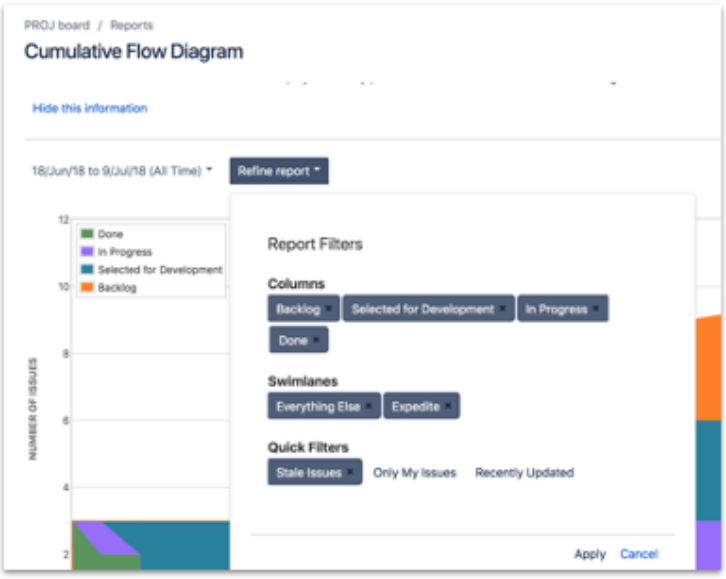
Filtering a board: quick filters

Further filter issues displayed on a board



Quick filters are a way to further filter issues displayed on a board. By default, all the issues are displayed. You can see that there are two default quick filters. Check the Only My Issues box to limit the board to displaying issues in which you are the assignee. Check the recently updated checkbox to show only issues that have been updated in the last day. If you have a lot of quick filters, you can use the search box to find your filter. Here we only have the default quick filters, so the search box is not too helpful.

Quick filters and reports



A board's quick filters can also be used to refine reports. Here we are viewing the cumulative flow diagram report. If we click on the "refine report" dropdown, we see that we can deselect columns or swimlanes to hide them from the report. We will discuss swimlanes a little later. We could further filter the issues that are being reported on by clicking on one or more quick filters to enable it. They are all not enabled by default. Be sure to click "Apply" to see the changes to the report.

Takeaways

- Filters are saved searches that can be exposed through user interface elements
- Every board has a filter that defines the issues shown on the board
- Quick filters are saved searches that are used to further limit the issues displayed on a board or in reports

Tasks

Filters

- ❑ Explore default filter queries
- ❑ Create a starred filter
- ❑ Explore and create quick filters

12 Epics

What will you learn?

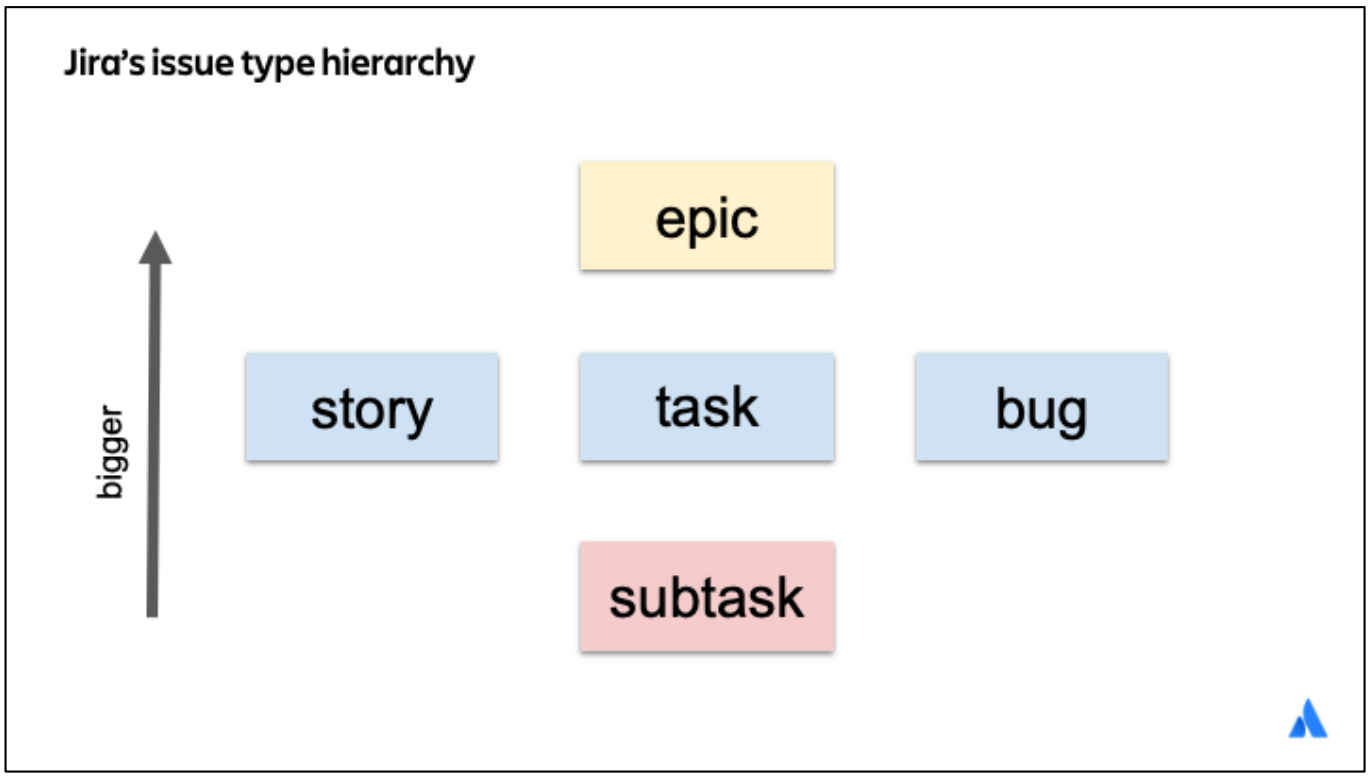
- Describe epics
- Work with epics
- Manage epics in the backlog

Topics

Epics overview

Working with epics

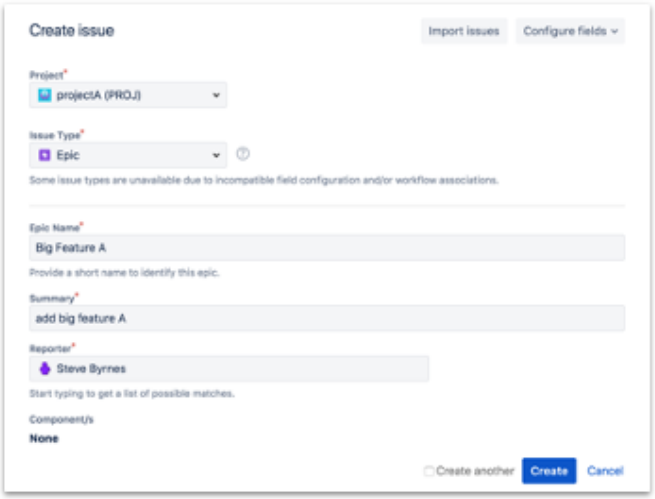
Epics in the backlog



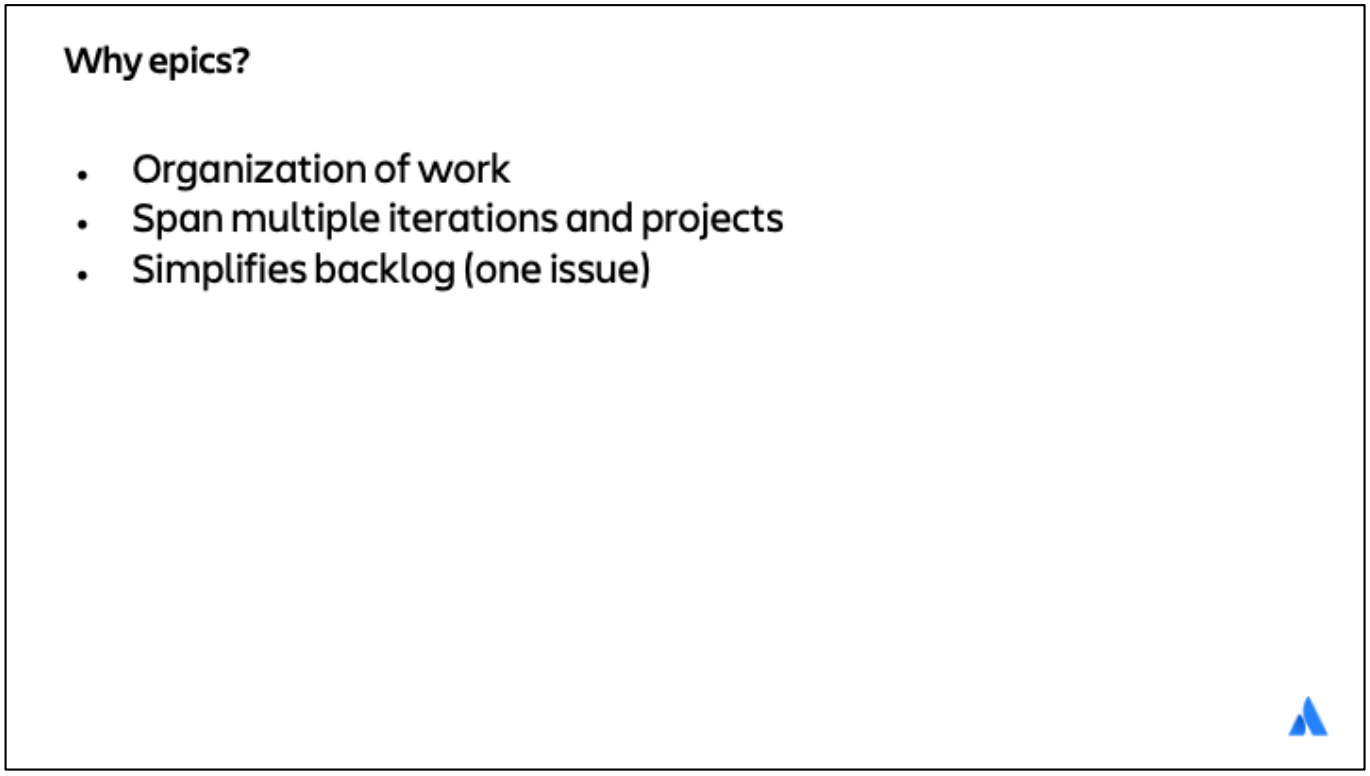
We have discussed stories, tasks, bugs, and subtasks. An epic is an issue type that has a bigger scope than all of these. How they are actually used is up to the team.

Epic

- A large issue
- Can contain other issues
- Child issues can span multiple iterations, projects, teams and boards
- Can be a placeholder for many stories



An epic is a large issue.
An epic can contain issues. They can contain stories, tasks, bugs and custom issue types.
Child issues of an epic can span multiples iterations, projects, teams and boards.
It can serve as a placeholder for many stories. For example, an epic might be to create an iPhone version of an app. You then would break this into stories when the team is available to actually work on it.



Epics are useful because they provide an organization of the team's work, rather than grouping unrelated stories.
They are also useful if the work spans multiple iterations.
Also, have one placeholder story to represent a lot of work can simplify the backlog.
You don't want to create detailed plans for an epic too early.

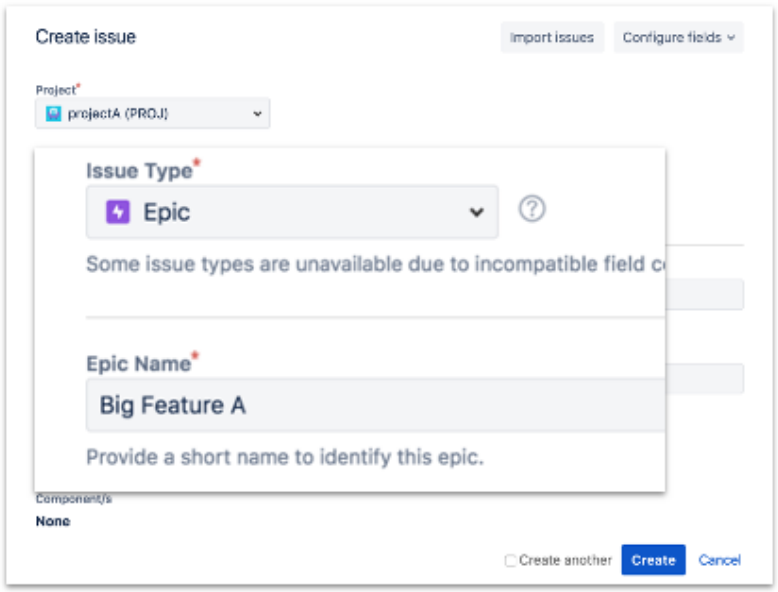
Topics

Epics overview

Working with epics

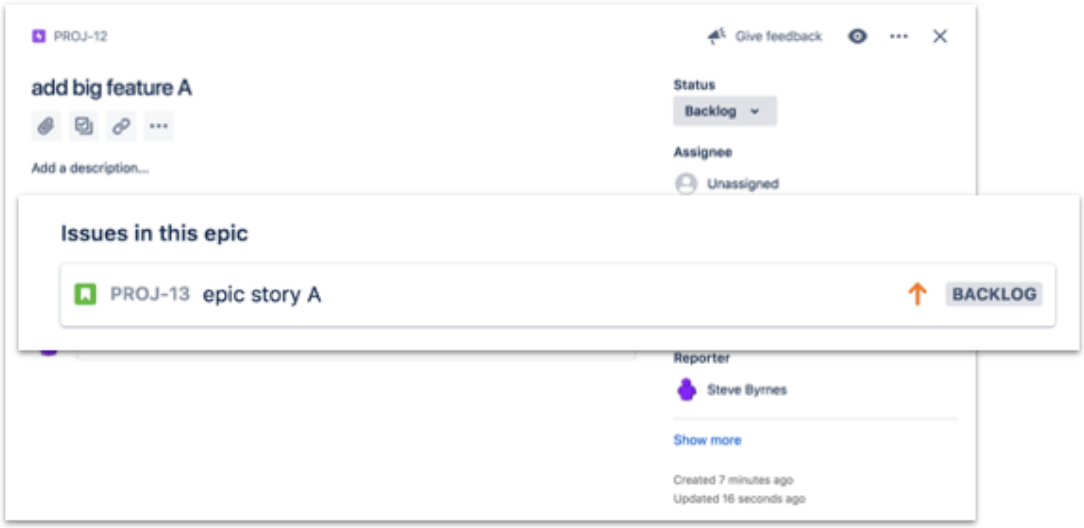
Epics in the backlog

Creating an epic



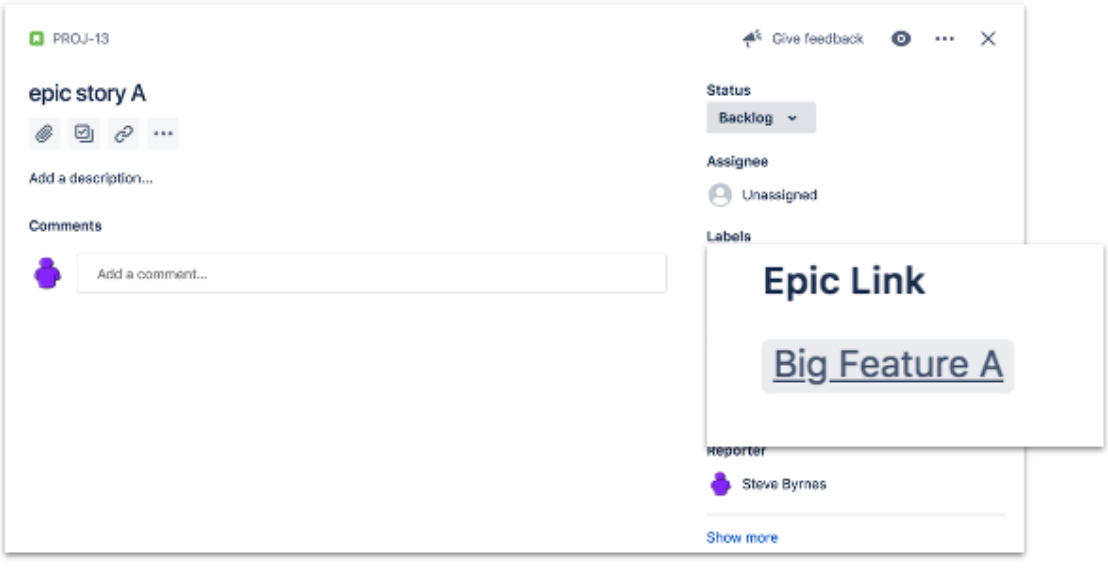
You can create an epic by selecting the Epic issue type when creating an issue. Here we have created an epic named Big Feature A.

Viewing the issues of an epic



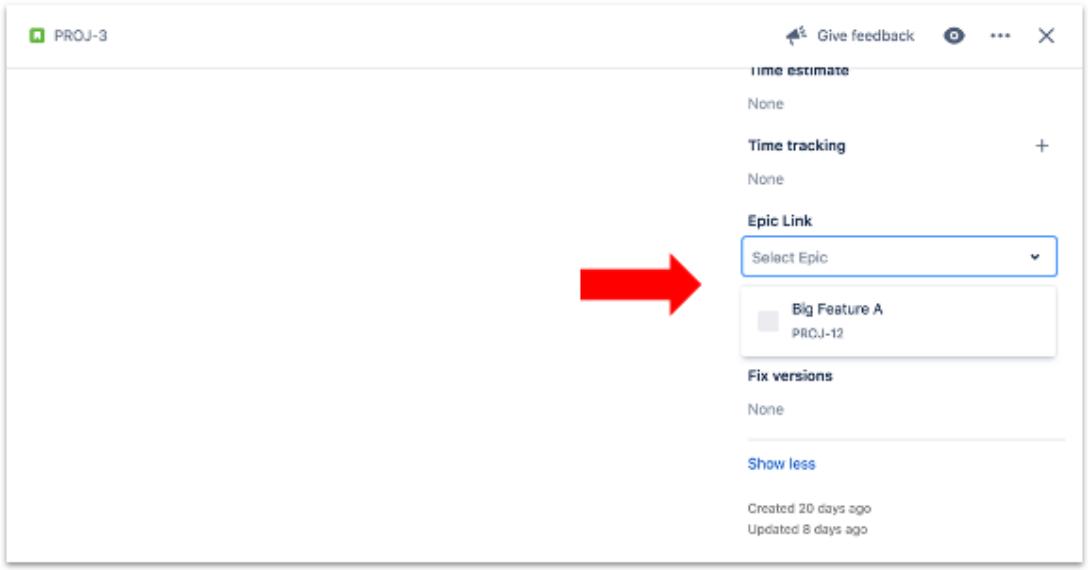
To view the issues of an epic, you can open the epic issue.
Issues that belong to this epic are shown.

Epic link



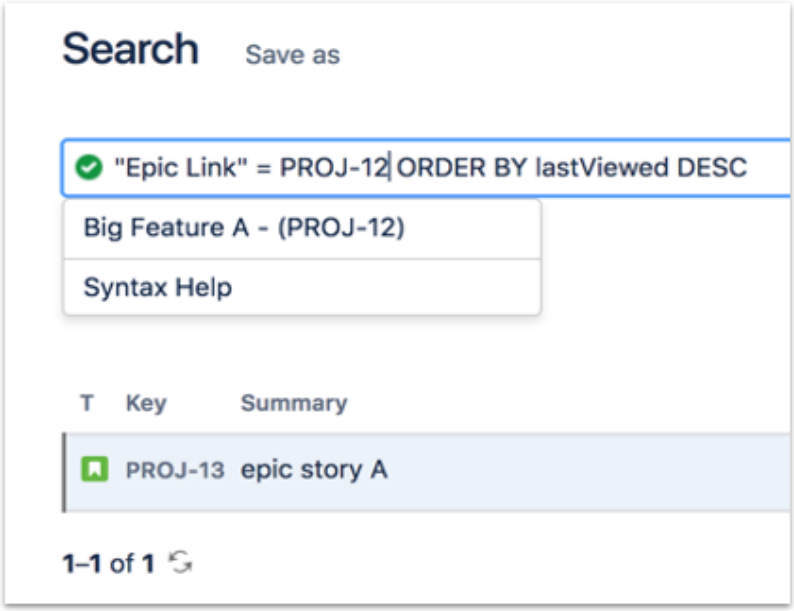
If you view an issue belonging to an epic, you will see the epic in the Epic Link field. You can click on the link to view the epic.

Adding an existing issue to an epic



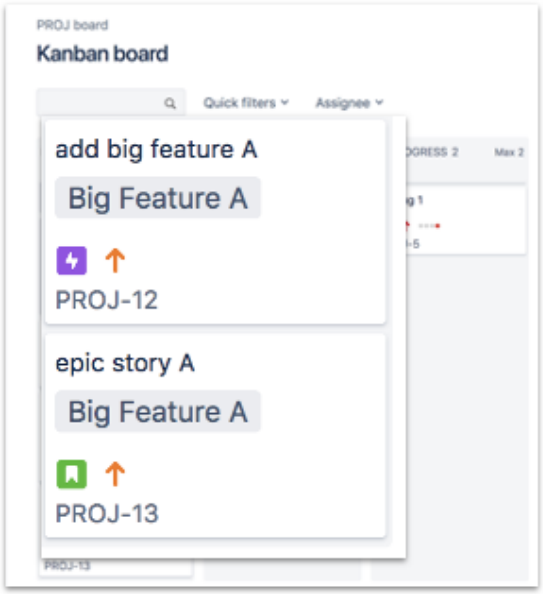
You also can add an existing issue to an epic at any time using the Epic Link dropdown.

Searching for issues of an epic



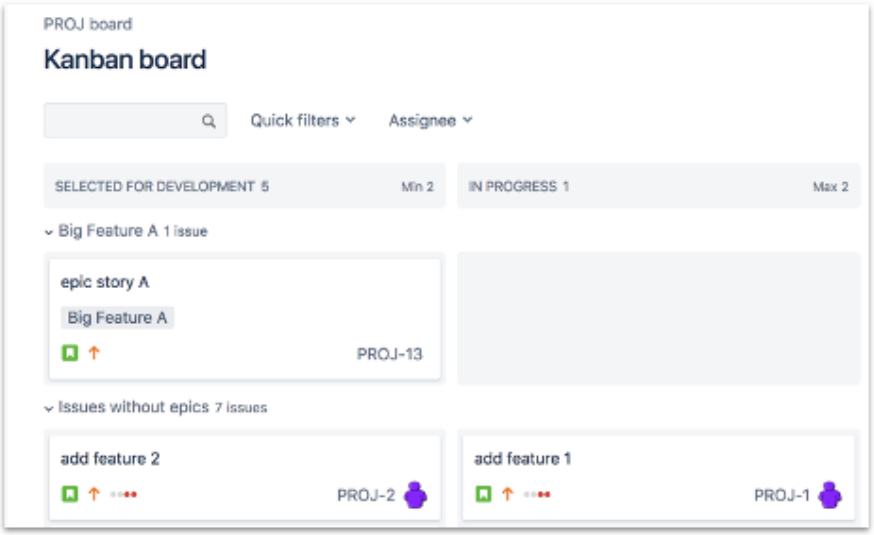
An epic is an issue type in Jira and, like all issue types, has an issue key. If an issue is part of an epic, its epic link field contains the issue key of the epic issue, which can be thought of as its parent. Here we are searching for all issues that are part of this epic. We can use Jira's autocomplete to help create this query.

Epic labels on a board



The kanban board can show the epic issue itself and any issues that belong to it are labeled.
Here you can see the epic issue, and a story that belongs to the epic.

Epic-based swimlanes



Here we see an epic-based swimlane. The issues of the epic are separated from the rest of the issues on the board.

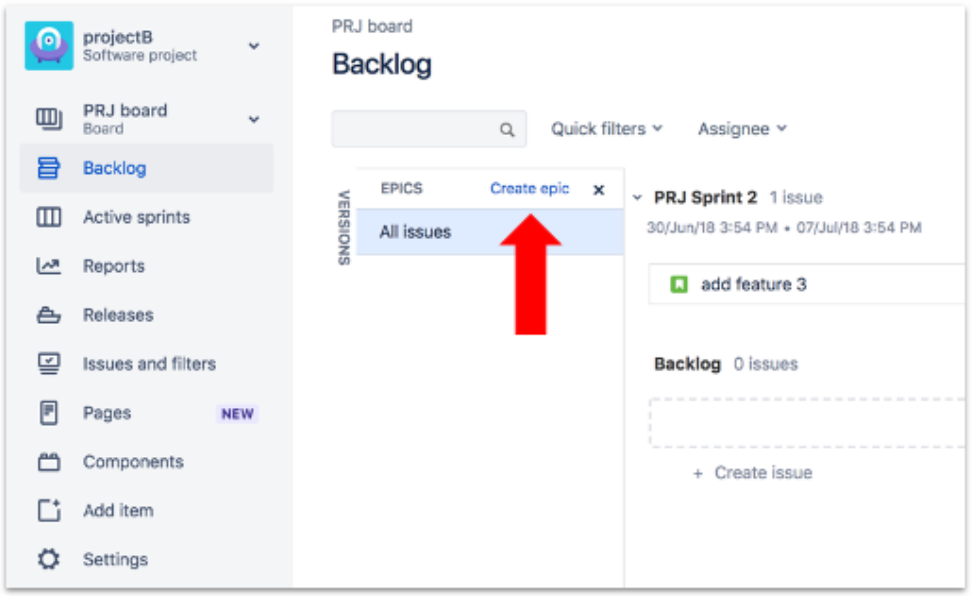
Topics

Epics overview

Working with epics

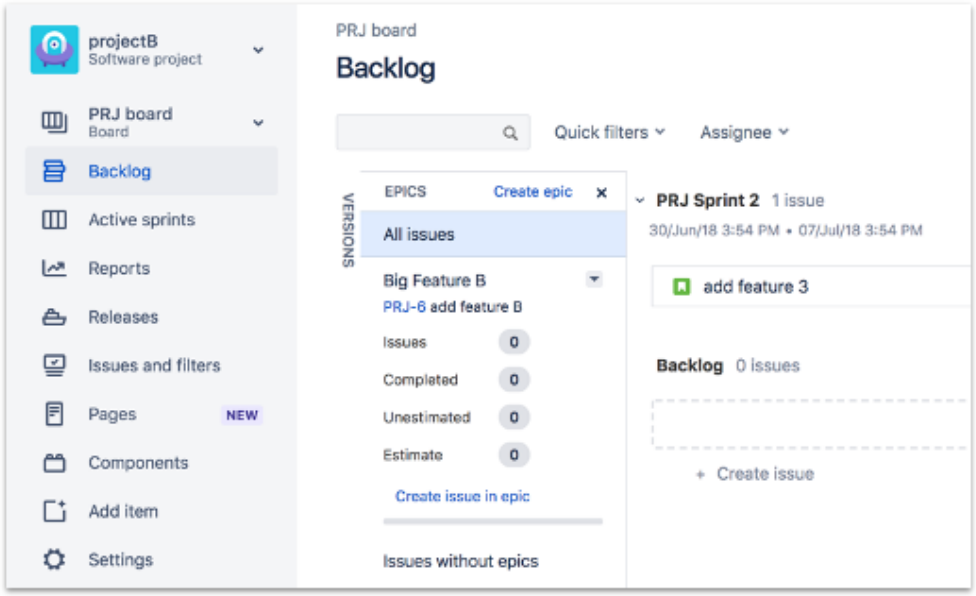
Epics in the backlog

Creating an epic from a backlog



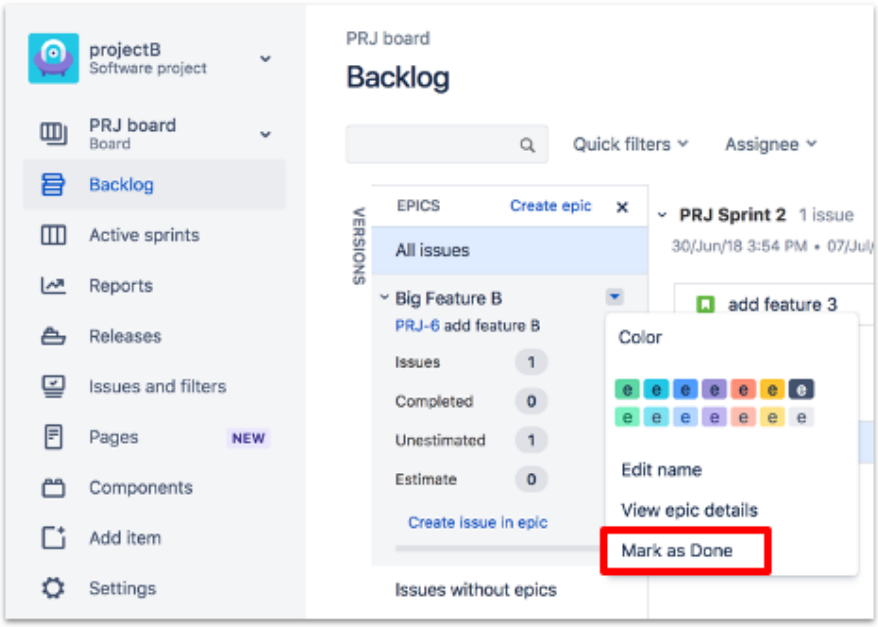
You can click "Create epic" under the Epics tab of the backlog.

Epics panel



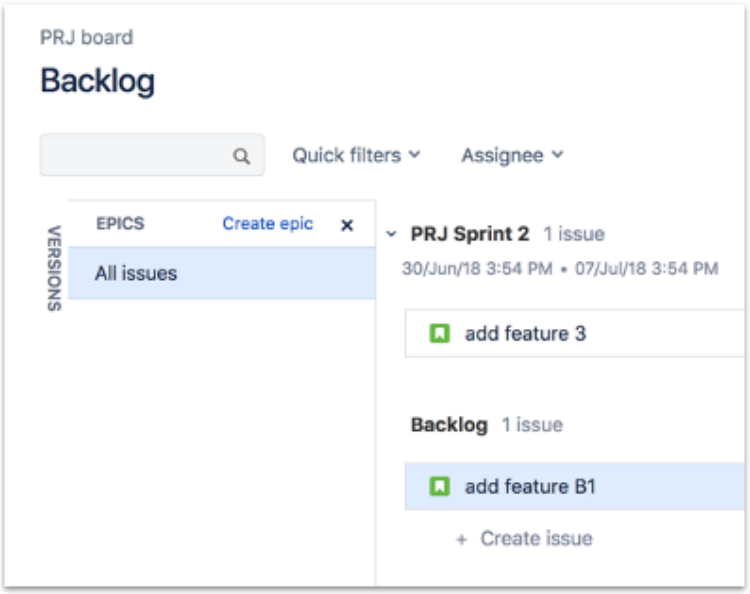
You can use the epics panel in the backlog to monitor the current status of the epic. Notice the create issue in epic link. You can start the process of creating an issue in the epic here.

Mark an epic as done



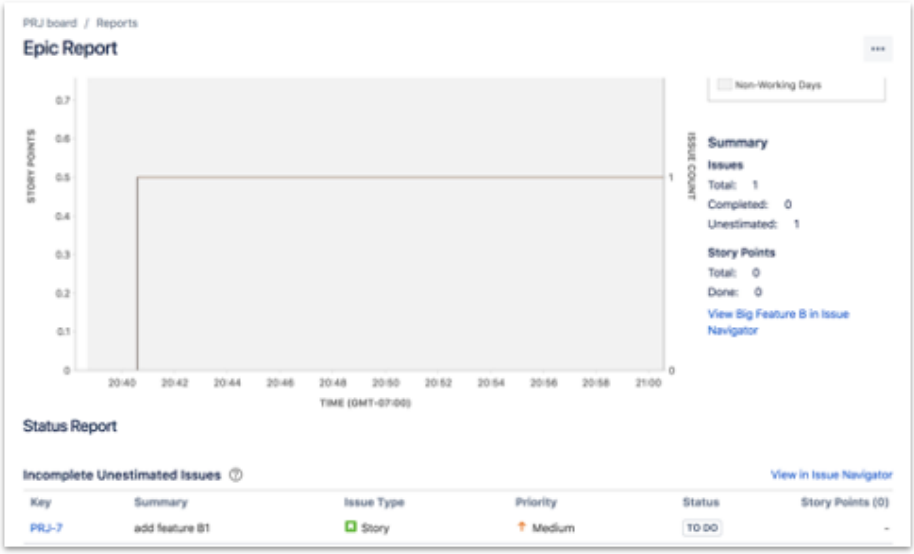
You can mark an epic as done in the backlo. Select mark as done from the dropdown next to the epic name.

An epic marked as done is removed from the epics panel



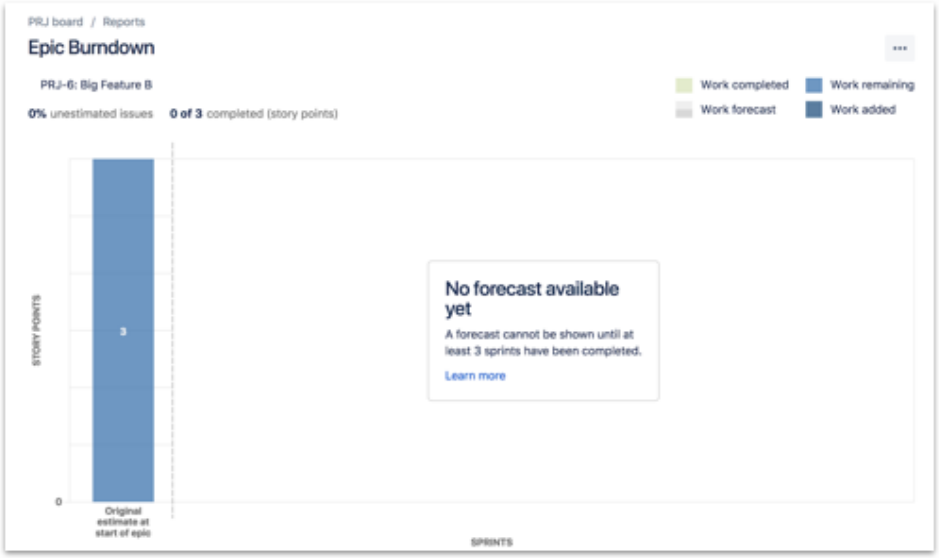
When you mark an epic as done, it no longer shows up in the epics panel of the backlog. The epic status field for the epic issue is set to a value of done.

Epic report



Under the reports tab, you can view the Epic Report. This shows the current status of the epic, as well as a list of the issues in the epic.

Scrum- epic burndown chart



You can view the epic burndown chart to see your progress on the epic as you complete sprints.

Takeaways

- An epic is a large issue of issue type "epic" that may contain other issues
- The "epic link" field is used to associate issues with an epic
- Epics can be shown on boards or in backlogs

Tasks

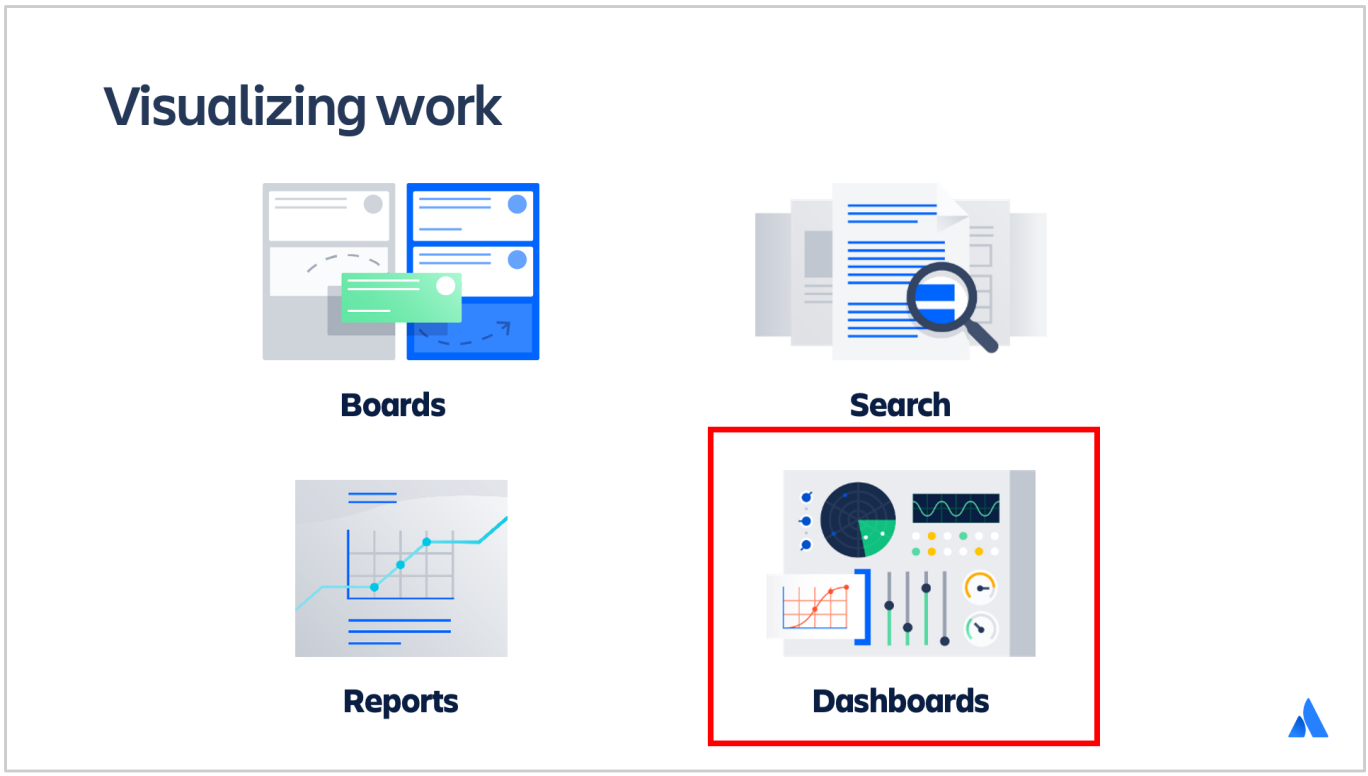
Epics

- ❑ Create an issue of type "epic"
- ❑ Add issues to the epic
- ❑ View swimlanes by epic
- ❑ View the epic in the kanban backlog
- ❑ Complete an epic

13 Dashboards

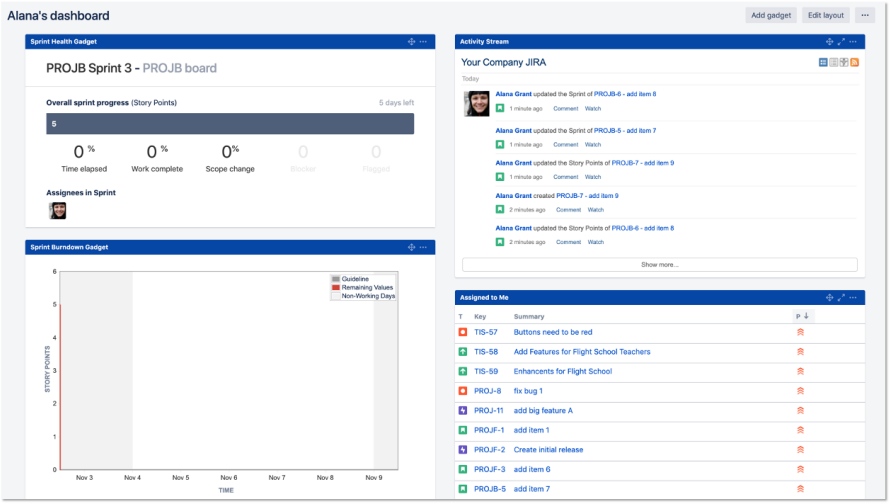
What will you learn?

- Describe dashboards
- Configure a dashboard
- Display a dashboard as a wallboard



There are the main ways to visualize work using Jira. Here we will discuss dashboards.

Dashboards

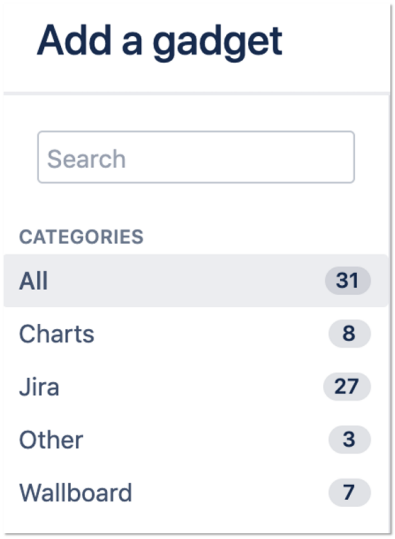


- Configurable view of the work of one or more projects
- Can be personal or shared
- Contains *gadgets*

Dashboards are a way to present a customized view of projects. They can be created for personal use or to share with others on the team.

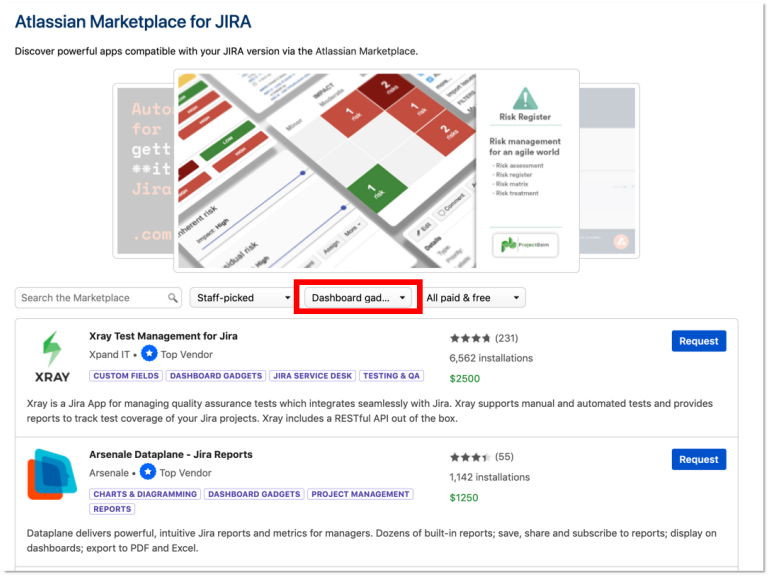
Dashboards are made up of one or more gadgets. A gadget displays some aspect of the work of the projects. In this example, we see four gadgets. Two are related to sprints, one for an activity stream and another showing issues assigned to the current user.

Types of gadgets



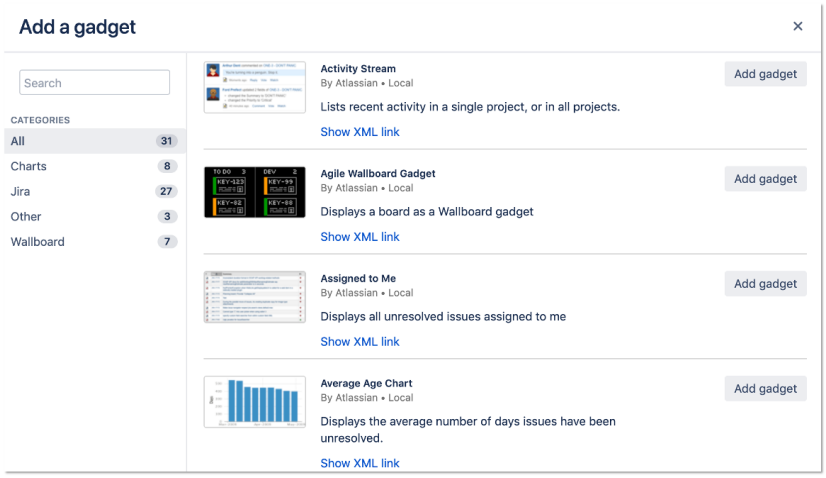
Gadgets can be divided into categories:
Charts – Displays information visually using pie charts, bar charts, etc.
Jira – Present information about issues. The issues could be filtered by project, team member, status, etc.
Wallboard – The information in these gadgets can be displayed in dashboards, but is optimized for display as a wallboard.

ATLASSIAN Marketplace



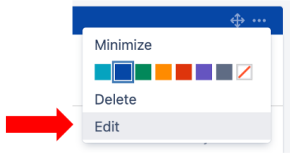
- Contains more dashboard gadgets
- Select “Dashboard gadgets”

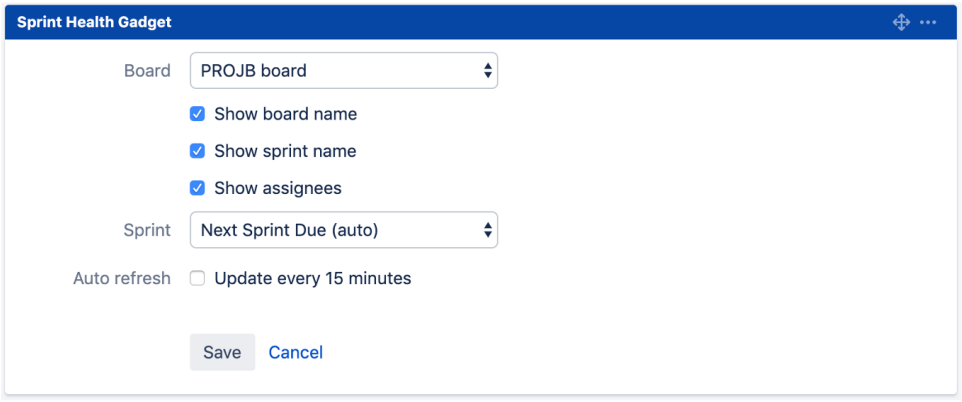
Adding gadgets



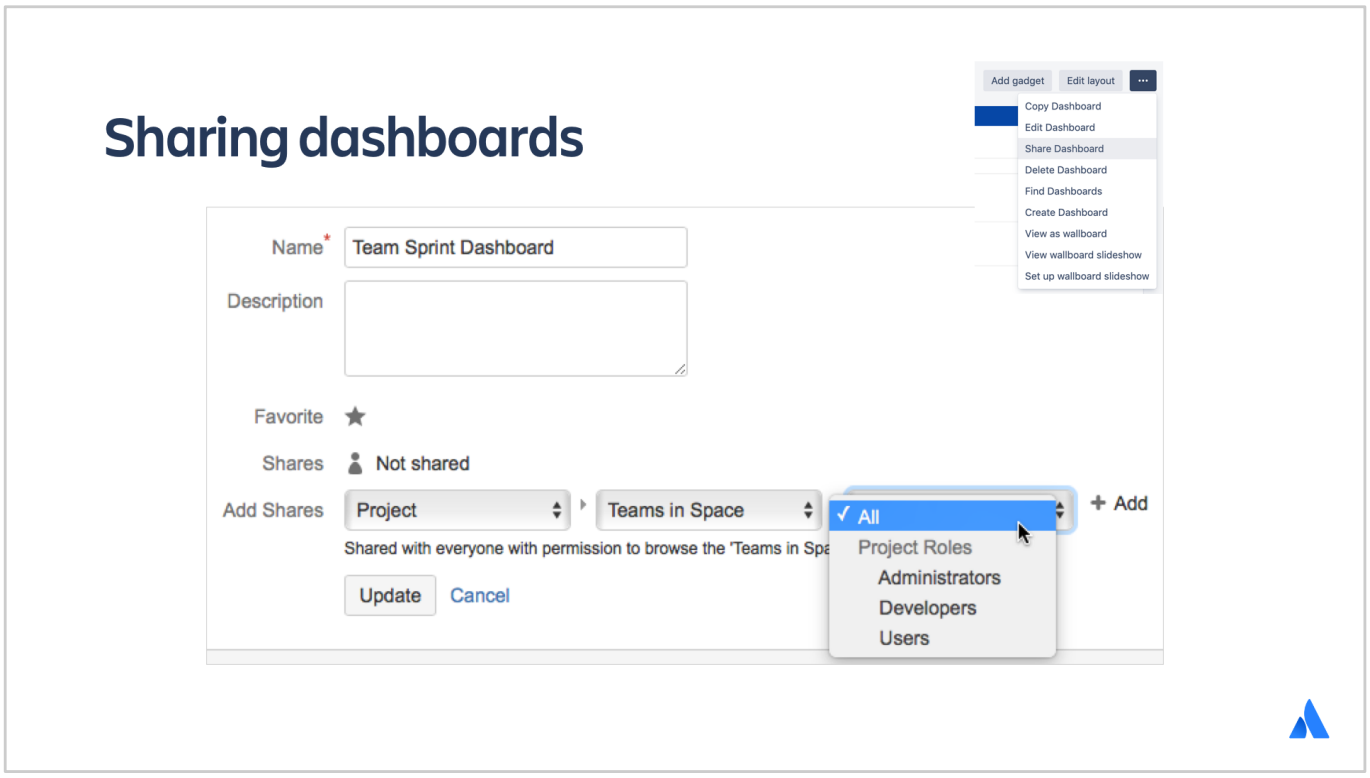
Click the Add gadget button to bring up a list of gadgets to add to your dashboard.

Configuring a gadget





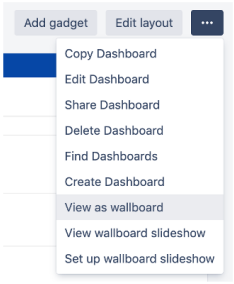
Most gadgets can be configured to show specific information. In this example, we are specifying the information related to the Sprint Health Gadget, such as which board to use as the source of data.



Dashboards can be configured and used privately. They also can be shared with all or part of the team.

Wallboards

- Turn any dashboard into a wallboard
- Acts as an information radiator



A wallboard is usually a television or monitor that is visible in a room. This radiates project status to the team, increasing the team's shared understanding of the projects.

Takeaways

- Dashboards display the work of projects
- Dashboards can be shared or used personally
- Gadgets display a portion of a dashboard
- Dashboards can be shown as a wallboard to radiate information

Tasks

Dashboards

- Create a dashboard
- Display a dashboard as a wallboard

14 Putting it all Together

Topics

Quick course review

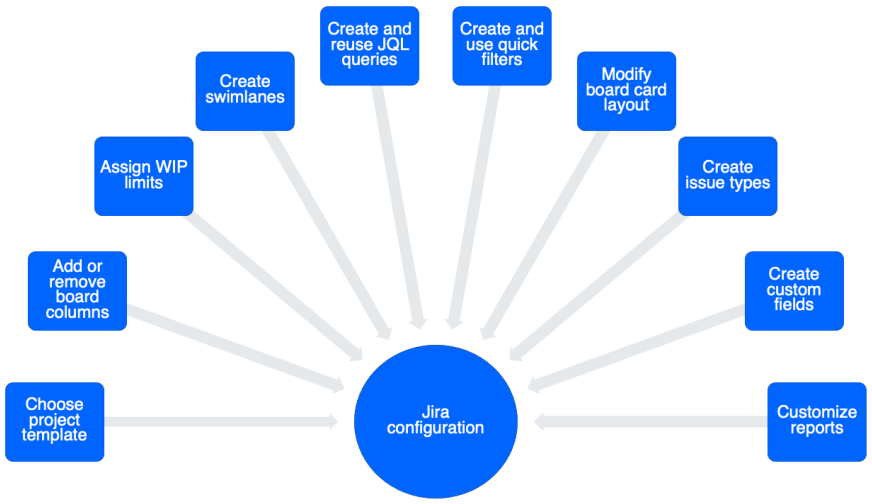
Jira family

Wrap up

Combined lean and agile principles

1. **Empower the team**
   - Select motivated individuals
   - Teams should self-organize
   - Collaborate to create shared understanding
2. **Visualize work**
3. **Experiment using the scientific method**
   - Continuously learn and improve
   - Embrace change
   - Partner with the customer
   - Continuously inspect and adapt
4. **Plan, develop and deliver incrementally**
   - Prefer conversations for conveying information
   - Continuously refactor to maintain agility
   - Maintain a sustainable pace
   - Completed work items are the primary measure of progress
   - Obtain fast feedback
5. **Improve the "flow" of value**
   - Limit work in progress
   - Map the value stream
   - Pull work
   - Eliminate waste
   - Reduce setup times
   - Automate what should be automated
   - Continuously strive for simplicity
6. **Build quality in**
   - Don't compromise on quality
   - The process should identify problems
   - Fix problems when they are discovered
   - Identify and fix the root cause

Ways to configure Jira to match your team's process



Each agile project has a different setup

- Different members
- Different processes
- Different personal preferences
- In total we have 9 areas that we can configure

- This is a continuous process. Agile is very responsive to change. Reconfiguring boards is just one way we can respond to changing needs.
- We'll be covering all areas in subsequent modules but let's start with the general configuration for now.

Topics

Quick course review

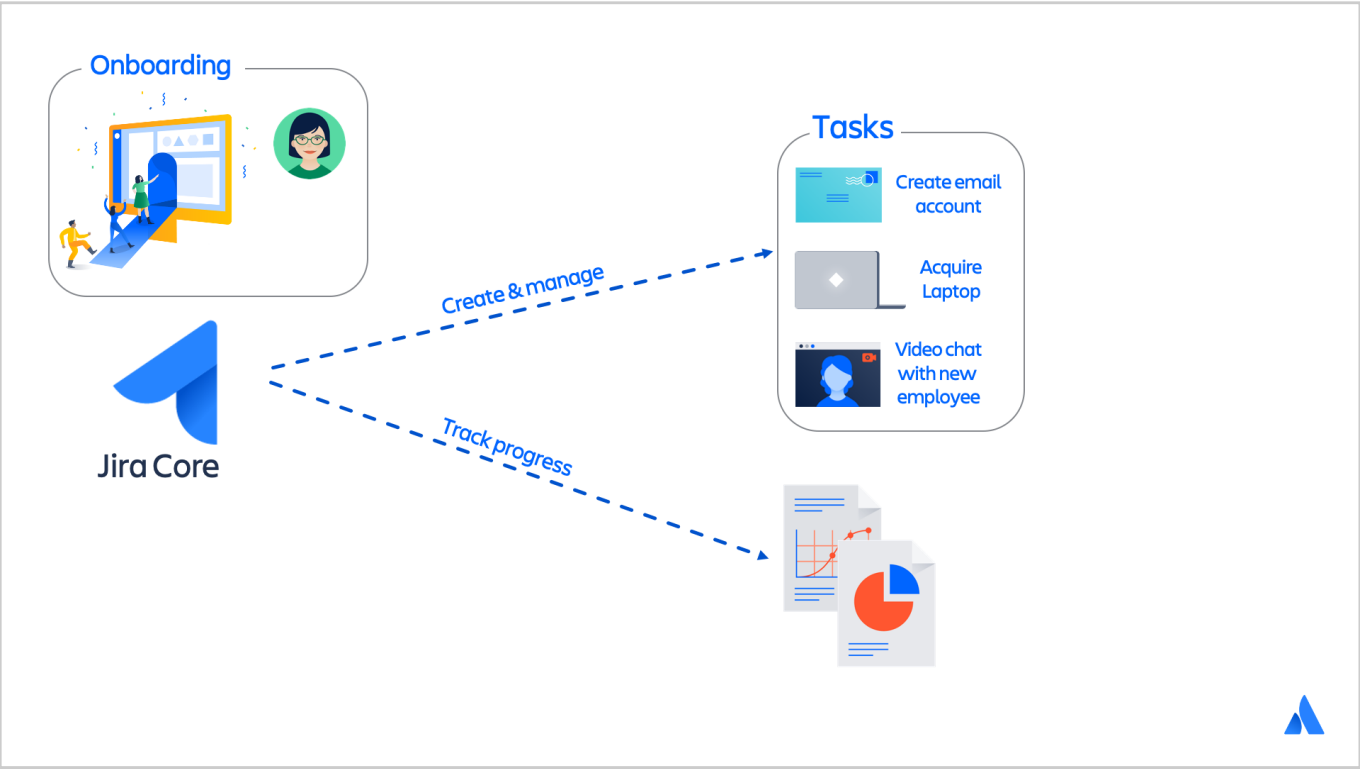
Jira family

Wrap up

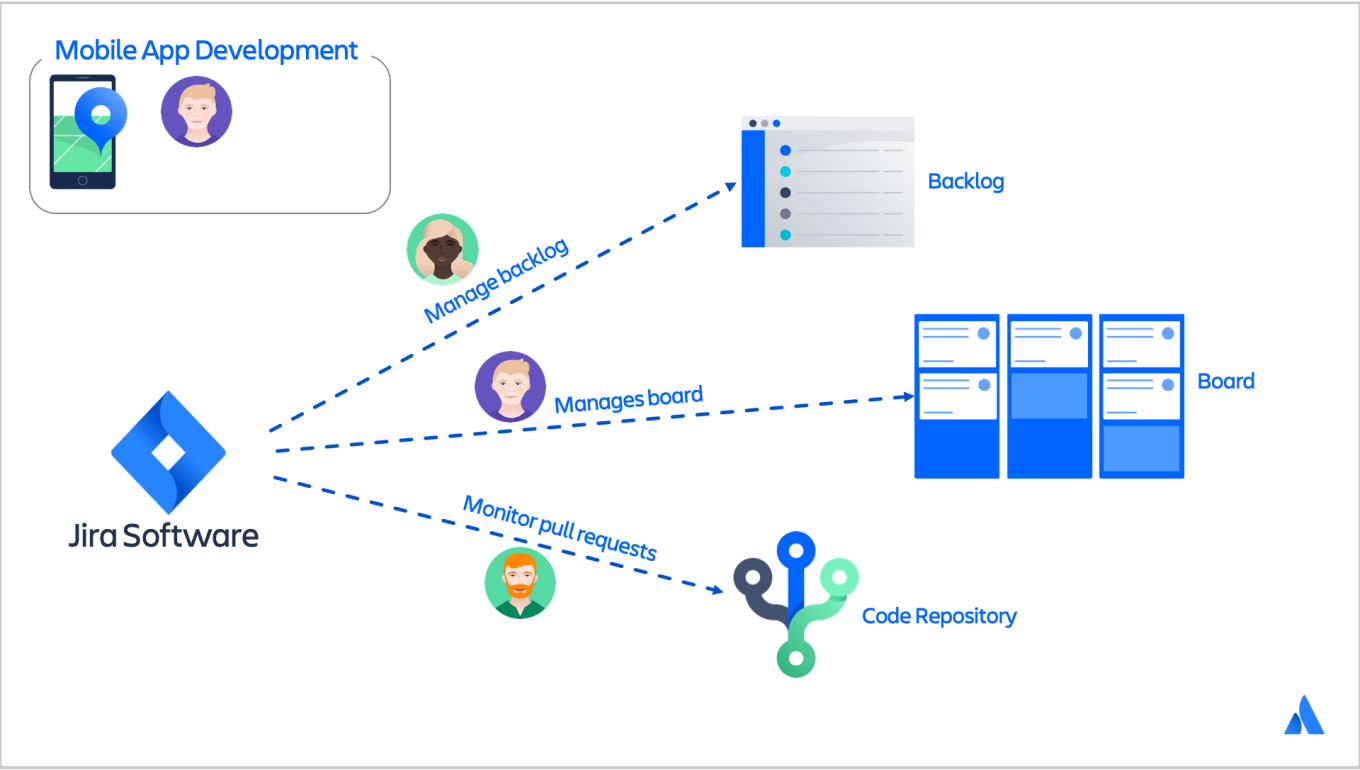
Jira is a family of three products.

- Jira Core - Used to manage non-software projects and help keep teams organized. Allows you to manage projects, monitor details, and measure performance.
- Jira Service Desk – Extends the core functionality of Jira Core. Used by IT and Customer Service teams to create and run their service desks. Has capabilities, which include self-service, automation, SLAs, and CSAT reporting.
- Jira Software – Extends the core functionality of Jira Core. A software development tool for agile teams that allows them to plan, track, release, and report their progress developing software. Whether your team uses Scrum, Kanban, Scrumban, or something else, Jira Software is designed with Agile teams in mind.

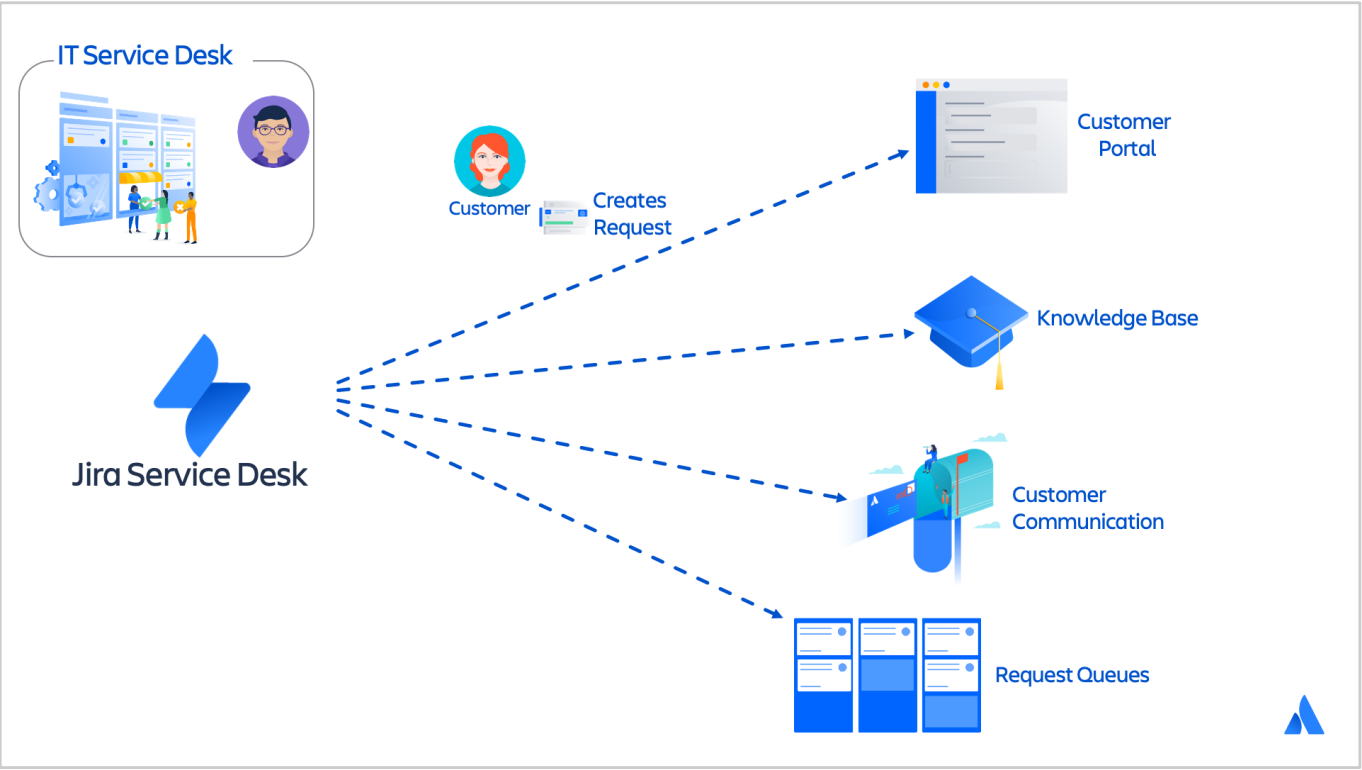
Both Jira Service Desk and Jira Software combine the core functionality provided by Jira Core such as issue tracking, workflows, and reporting, and add their unique functionality that allows you to manage your projects.
You can use Jira Core alone, a combination of Jira products, or all three Jira products.



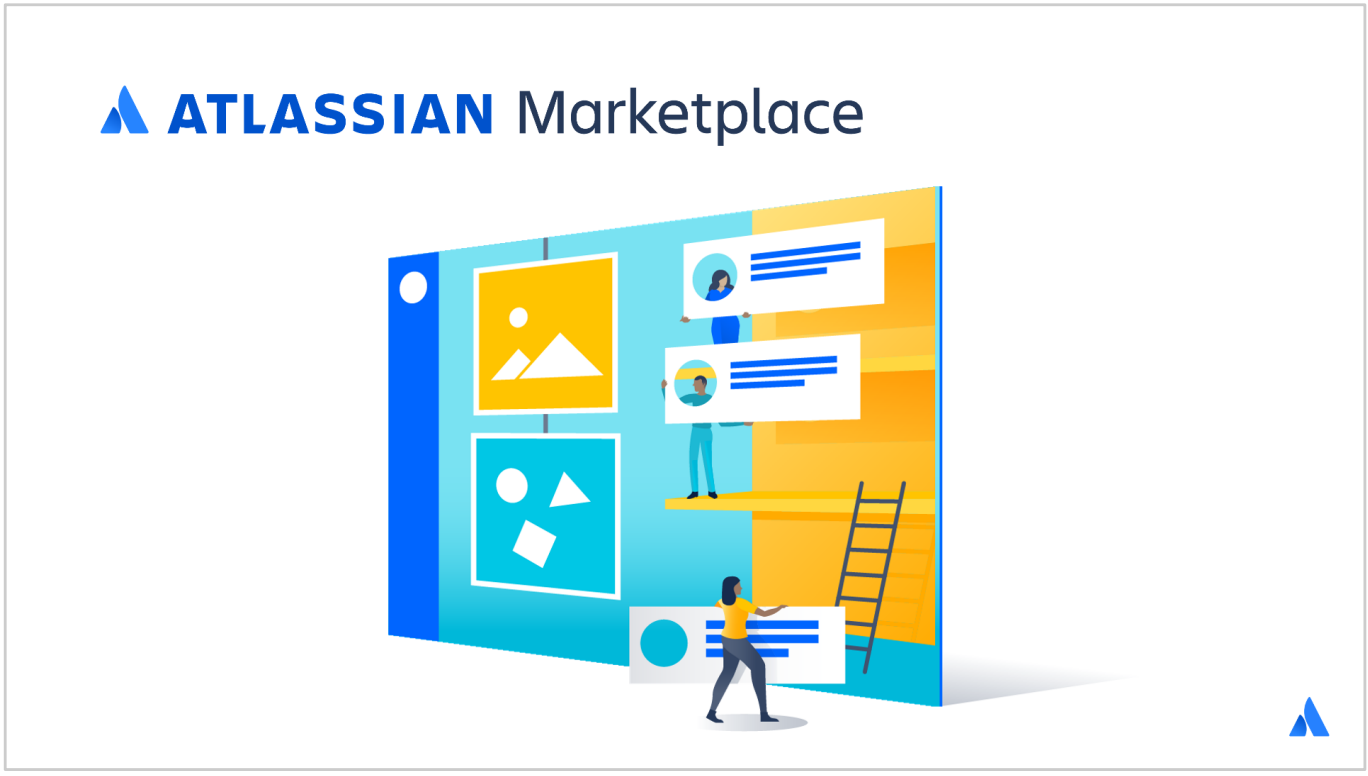
Jira Core helps you manage business projects. For example, Cassie manages an HR team, which uses Jira Core to manage their Onboarding project. Within Jira, they can create tasks and manage them. A task can be anything from creating an email address for the new employee, acquiring his/her new laptop, or keeping in touch before their start date. Cassie can get a quick look at the overall statistics to ensure the tasks aren't getting bottlenecked, and that helps keep the project on track



Jira Software is designed to help you manage software projects. For example, Will manages this agile software development team. They are currently adding location functionality to their mobile application. They manage all their work on a board and Will manages the board, which contains everything the team is committed to finishing. Emma, the Product Manager manages the backlog of items that still need to be started. Jira Software allows developers like Kevin to create branches and monitor pull requests and build statuses right from the Jira issues.



Jira Service desk helps you manage your IT service desk. For example, Ryan's team uses Jira Service Desk to manage their internal IT service desk. It allows the Service Desk team members to resolve problems and service requests from other employees throughout the company. Employees create requests using the Customer Portal. The requests immediately show up in the service desk queues. The service desk team members monitor these queues and grab the most urgent requests on which to work. Sometimes, they can resolve a problem immediately from the knowledge base suggestions that automatically appear in the service desk request. They can also communicate with customers and stakeholders.

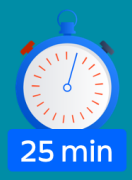


Can't find what you want for a gadget or a report out-of-the-box in Jira? Try the Atlassian Marketplace. There are over 1,000 apps available for just Jira. Someone may have already created what you need. Still can't find what you want? Jira has a rich UI in which a developer can create a custom gadget.

Tasks

Final project (optional)

- Modify a Jira project to match your team's ever-changing agile processes



Topics

Quick course review

Jira family

Wrap up

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**.