1.Introduction

Jira

Jira is a powerful project management and issue-tracking tool developed by Atlassian, widely used by software development teams to manage work. It helps teams plan, track, and release high-quality software by providing a flexible, customizable platform for organizing and prioritizing tasks

- **Improved Visibility and Transparency**
- **Customizable Workflows**
- **Efficient Issue and Bug Tracking**
- **Collaboration and Accountability**
- **Data-Driven Decision Making**
- **Integration with Other Tools**

Terminologies

Core concepts

- **Issue(Work Item)**: The fundamental unit of work (task, bug, story, etc.).
- **Project**: A collection/container for related issues.
- **Issue Type**: Classification for issues (e.g., Bug, Story, Epic, Task).
- **Board**: Visualizes issues (Kanban or Scrum board).
- **Workflow**: The defined path an issue follows (e.g., To Do -> In Progress -> Done).

Agile & Scrum Terms

- **Epic**: A large body of work, broken down into smaller stories.
- **Story**: A user-centric requirement (e.g., "As a user, I want...").
- **Task**: A generic piece of work.
- **Bug**: A defect or error.
- **Sprint**: A fixed time period (e.g., 2 weeks) to complete work.
- **Backlog**: The prioritized list of all pending work.
- **Burndown Chart**: Visualizes work remaining vs. time in a sprint.

Views & Tools

- **Dashboard**: Customizable overview with gadgets (charts, stats).
- **JQL (Jira Query Language)**: Powerful search syntax for issues.
- **Component**: A sub-section or feature within a project.

Statuses and states

- **To Do/Open**: Ready to start.
- **In Progress/Work in Progress**: Actively being worked on.

- **In Review/Under Review**: Awaiting peer check.
- **Done/Closed/Resolved**: Work completed.
- **Cancelled/Won't Do**: Work stopped.

Key Terminology Changes

- **Projects → Spaces**: Jira Cloud containers are now called "Spaces" to differentiate them from "Atlassian Projects" (high-level planning).
- **Issues → Work / Work Items**: "Issues" (bugs, tasks, stories) are now grouped under the umbrella term "Work," with individual items called "Work Items" or "Tasks".

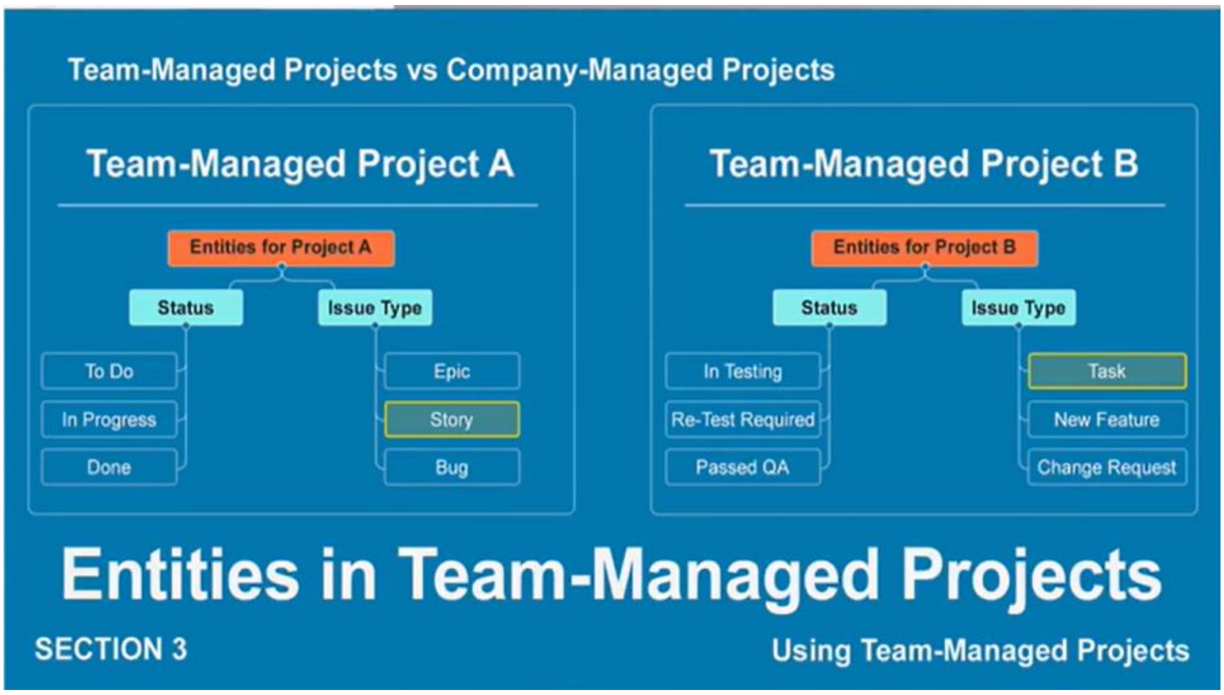
rollout began in **early 2025 (February/May)**

2. Jira Fundamentals

Project Types: Team Managed Vs Company Managed Projects

Team Managed Project

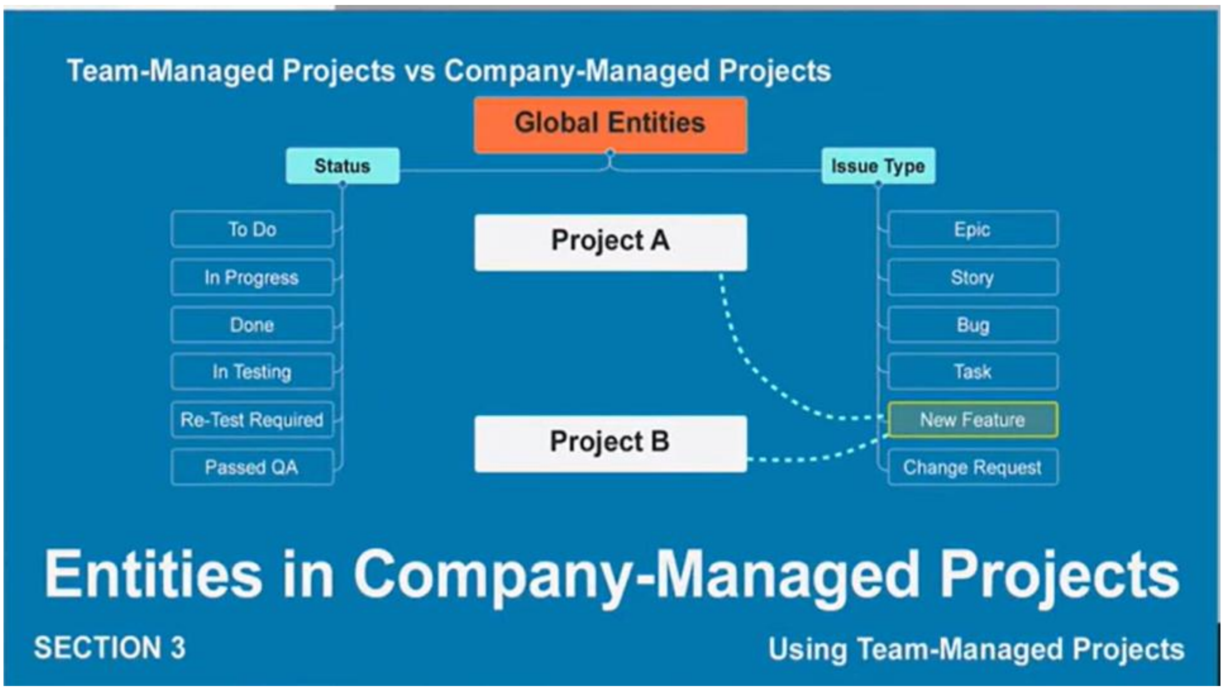
- Team Managed projects offer independence, quick setup, and isolated configurations for small teams, letting project admins easily control fields, workflows, and permissions without affecting others.
- Entities in team managed project are referred as project scope entities and they live only within the project where they are created



Company Managed Project

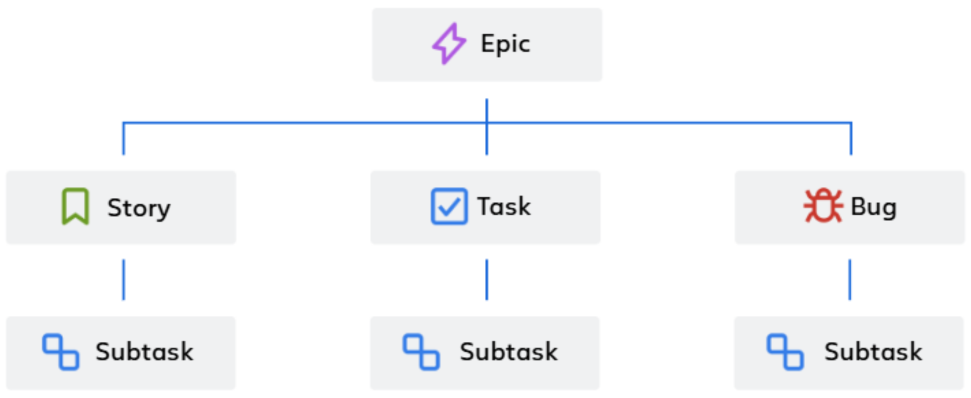
- Company managed projects provide powerful, standardized, and shareable configurations (schemes) for larger organizations, requiring Jira admins for setup, ideal for complex, consistent processes across many teams but with higher complexity and admin dependency

- Entities in company managed project are referred as global entities and they are independent of the project; change made to the entities is reflected across every project that the entities belong to.



Issue Types & Hierarchy

- **Epic**: A large body of work or a significant feature that can't be completed in a single sprint, spanning multiple stories/tasks.
- **Story**: A user-centric requirement or feature, often written from the user's perspective (e.g., "As a user, I want...").
- **Task**: A generic, actionable unit of work needed to complete a goal, but not necessarily user-facing like a Story.

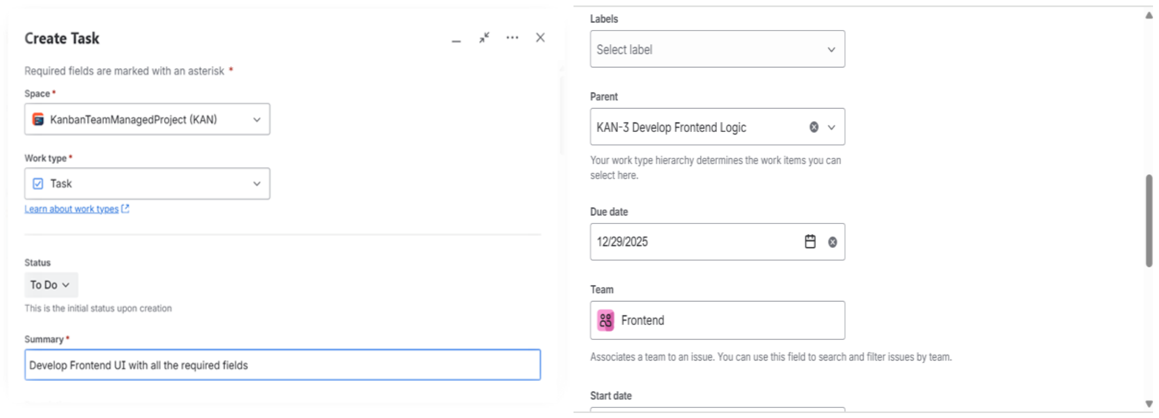


- **Bug**: A defect, error, or flaw in the software that needs fixing.

- **Sub-task**: A granular, manageable piece of work required to complete a parent Story, Task, or Bug; it's the lowest level of work.
- **Hierarchy Example**: An **Epic** (e.g., "User Authentication") might contain several **Stories** ("Login with Email", "Password Reset") and **Tasks** ("Set up Database"), with a **Bug** ("Login Button Broken") reported, and each Story/Task/Bug could have its own **Sub-tasks** ("Design UI", "Develop Backend", "Test")

Creating and Managing Issues

Creating a Task



Create Task

Required fields are marked with an asterisk *

Space * KanbanTeamManagedProject (KAN)

Work type * Task

Learn about work types

Status To Do

This is the initial status upon creation

Summary * Develop Frontend UI with all the required fields

Labels Select label

Parent KAN-3 Develop Frontend Logic

Your work type hierarchy determines the work items you can select here.

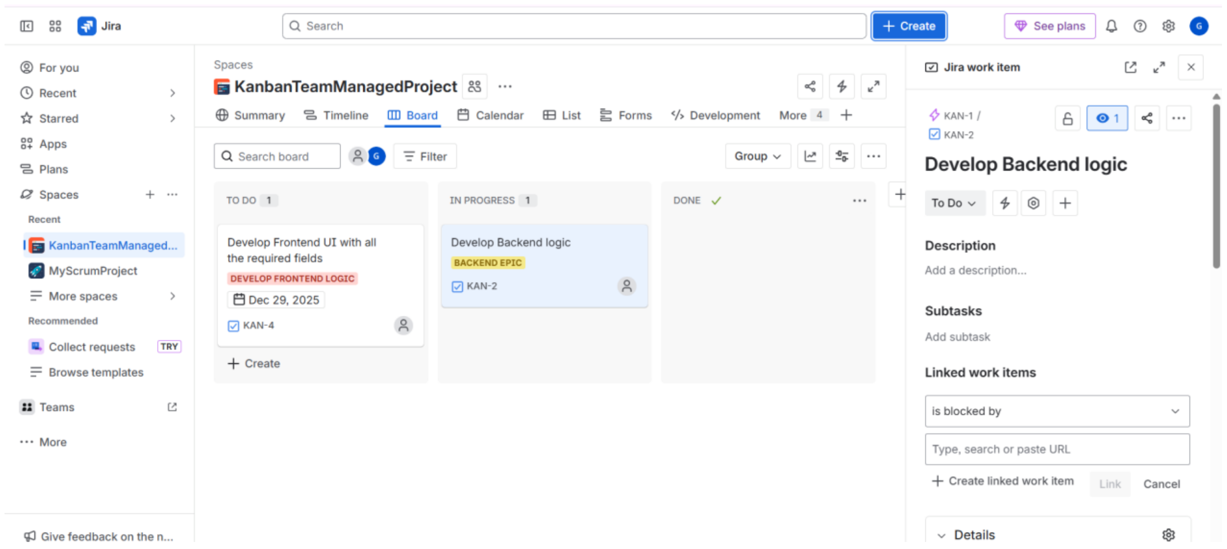
Due date 12/29/2025

Team Frontend

Associates a team to an issue. You can use this field to search and filter issues by team.

Start date

Kanban Board with some Issues/Work Items (Tasks)



KanbanTeamManagedProject

TO DO 1
Develop Frontend UI with all the required fields
DEVELOP FRONTEND LOGIC
Dec 29, 2025
KAN-4
+ Create

IN PROGRESS 1
Develop Backend logic
BACKEND EPIC
KAN-2

DONE ✓

Jira work item
KAN-1 / KAN-2
Develop Backend logic
To Do
Description
Add a description...
Subtasks
Add subtask
Linked work items
is blocked by
Type, search or paste URL
+ Create linked work item Link Cancel
Details

Kanban and Scrum Boards

Key terminologies

- **Board:** A visual representation of your workflow, using columns (statuses) and tasks (issues/cards).

- **Backlog:** A prioritized list of work (user stories, tasks, bugs) waiting to be done, managed in both.
- **Sprints (Scrum):** Fixed-length iterations where teams commit to a set amount of work.
- **Columns/Statuses:** Stages of work (e.g., To Do, In Progress, Review, Done).
- **Issues/Cards:** Visual representations of work items, moved across columns.

Working with a Scrum Board

1. **Create/Select Project:** Set up a Scrum project.
2. **Plan Sprints:** Use the backlog to drag and rank issues into planned sprints.
3. **Start Sprint:** Move committed issues from the backlog to the "Active Sprints" view.
4. **Track Progress:** Team members pull tasks into "In Progress," move them through columns (like Review, Testing), and to "Done".
5. **Review/Retrospect:** Use sprint reviews and retrospectives to inspect and adapt.

Working with a Kanban Board

1. **Create/Select Project:** Choose a Kanban project template.
2. **Manage Backlog:** Add and prioritize work in the backlog.
3. **Visualize Flow:** Tasks appear on the board as they are ready for development, moving from left to right.
4. **Limit Work-in-Progress (WIP):** Configure WIP limits (via Board Settings > Columns) to focus flow and prevent bottlenecks.
5. **Monitor & Improve:** Use board metrics (cycle time, lead time) to identify issues and improve the process.

3. Agile Workflows

Scrum Workflow

1. Backlog Management

- **Create Issues:** Populate the product backlog with user stories, bugs, and tasks representing project requirements.
- **Prioritize:** Drag and drop items in the backlog to rank them by importance (highest priority at the top).

2. Sprint Planning

- **Define Sprint:** Set a fixed duration (e.g., 1-4 weeks) for the sprint.
- **Select Work:** Pull prioritized items from the backlog into the sprint.
- **Set Goal:** Define a clear, achievable Sprint Goal.
- **Create Sprint:** In Jira, create the sprint and add details like dates and the goal.

3. Sprint Execution

- **Start Sprint**: Begin the timebox in Jira.
- **Jira Board**: Use the Scrum board to visualize workflow (To Do, In Progress, Done).
- **Daily Scrum:** Hold brief daily meetings to check progress and blockers.
- **Update Status:** Move issues across the board as work progresses.

4. Sprint Review & Retrospective

- **Review:** Demo completed work to stakeholders for feedback.
- **Retrospective:** Team discusses what went well, what didn't, and how to improve in the next sprint.
- **Complete Sprint:** Mark the sprint as done in Jira; move unfinished work to the backlog or next sprint.

Story Point

- Story points are units of measure for expressing an estimate of the overall effort required to fully implement a product backlog item or any other piece of work.
- Teams assign story points relative to work complexity, the amount of work, and risk or uncertainty. Values are assigned to more effectively break down work into smaller pieces, so they can address uncertainty.
- Using story points instead of hours encourages team collaboration, reduces emotional bias, and improves forecasting.

Daily Standups with Jira

To run effective daily standups with Jira, keep them short (15 mins), consistent, and focused on the three questions (yesterday, today, blockers) while **walking the Jira board** (Kanban/Scrum) to visualize progress, addressing blockers offline, and ensuring all team members participate actively, making it a collaborative sync, not a status report to management, using Jira's board for transparency

Kanban Workflow

1. **Create a Kanban Project:** Start a new Jira project and select the "Kanban" software development template.
2. **Customize Your Workflow:** Add or modify columns to match your team's process (e.g., Analysis, Development, Testing, Done).
3. **Add Work Items:** Create tasks, bugs, or user stories and add them to your backlog (the first column).
4. **Set WIP Limits:** Configure limits for your "In Progress" columns to control work flow.
5. **Pull Work:** Team members pull items from the backlog or previous column when they have capacity, moving them across the board.
6. **Monitor & Improve:** Use Jira's analytics (like cycle time, throughput) to spot bottlenecks and discuss improvements in regular reviews.

Bug Life Cycle and Status Updates

The typical, detailed bug life cycle in Jira involves the following statuses:

- **New**: A tester or user has identified and logged a new bug, but it has not yet been reviewed or assigned to a developer.
- **Assigned**: A project manager or team lead has reviewed the bug report and assigned it to a specific developer or development team.
- **Open / In Progress**: The assigned developer is actively working on analyzing the bug, identifying its root cause, and developing a fix.
- **Fixed / Resolved**: The developer has implemented the necessary code changes and believes the bug is resolved. They mark it as fixed and typically assign it back to the QA team for verification.
- **Pending Retest / Ready for QA**: The bug fix is awaiting verification by the testing team.
- **Retest**: The QA team is in the process of retesting the application to ensure the reported issue is successfully resolved and no new issues were introduced.
- **Verified**: The testing team has confirmed that the bug is successfully fixed and the software is working as expected.
- **Closed**: The bug has been fixed, verified, and the issue is considered complete. It is removed from the active workflow.
- **Reopened**: If the retest fails or the bug resurfaces, the QA team reopens the issue and sends it back to the developer for further attention, restarting part of the cycle.

Other Potential Bug Statuses in Jira

Bugs might also transition to other statuses depending on the scenario:

- **Duplicate**: The reported bug is the same as an existing one, so the new report is marked as a duplicate and closed, with focus remaining on the original issue.
- **Rejected**: The developer or team determines the report is not a genuine issue, is out of scope, or is due to user error, and the bug is rejected.
- **Deferred**: The bug is valid but not a high priority and is postponed to a future release or development cycle.
- **Not a Bug**: The reported behavior is actually an intended feature or design choice.

Labels

- Jira Labels are flexible, keyword-based tags you add to issues for easy categorization, tracking features (like mobile-feature or backend-bug), and creating custom filters, acting like dynamic hashtags that can span projects for quick grouping and searching without rigid component structures, often created by typing into the label field on an issue.
- They help manage cross-project themes, support queues, or specific development areas, and can be added, viewed on cards, or bulk-edited using JQL.

Uses of Labels

- **Flexible Categorization:** Tag issues for aspects that don't fit neatly into components (e.g., customer-X, performance, release-v2.1).
- **Quick Filtering & Searching:** Use JQL (Jira Query Language) like labels = "performance" to find related issues across your project.
- **Cross-Project Grouping:** Labels are shared across projects, allowing unified tracking.

In Jira, the **List view** (also known as the Issue Navigator) can be leveraged with **issue labels and priorities** to filter, sort, and group work effectively, providing a comprehensive overview of tasks and their urgency.

Using Issue Labels and Priorities in List Views

1. Adding Labels and Priorities to Issues

Before you can organize work, issues need the relevant fields populated.

- **Labels:** Add a label by selecting an issue and clicking the **Labels** field (or pressing the 'l' shortcut) in the details section. Type in existing labels or create new ones (use hyphens or underscores for multiple words, as spaces are not allowed).
- **Priorities:** In most projects, the **Priority** field is available when creating or editing an issue. Project administrators can manage and customize priority levels and their associated colors to align with team conventions.

2. Organizing Work in the List View

The List view displays issues in a table format where each row is an issue and each column is a field.

- **Filter Issues:** Use the filters sidebar or the search bar at the top to narrow down your list. You can filter by:
  - **Priority:** Select specific priority levels (e.g., "High", "Highest") to focus on critical tasks.
  - **Labels:** Choose one or more labels to view all related issues (e.g., frontend or customer_feedback).
  - **Combinations:** Combine filters (e.g., show all high-priority bugs due this week assigned to you) to create a highly specific, targeted view.
- **Sort Issues:** Click on the **Priority** column header to quickly sort all visible issues from highest to lowest priority (or vice versa). You can also sort by other fields like Due Date or Assignee.
- **Group Issues:** In the List view, you can group work items by specific attributes like Priority.
  - Select **Group** at the top-right of the list.

- Select **Priority** from the dropdown menu to organize issues into collapsible sections based on their urgency level. (Note: This grouping is personal and only visible to you).

3. Advanced Techniques and Best Practices

- **Save Filters:** Once you've created a useful combination of filters and sorting, save it for future use or add it to your dashboard as a gadget.
- **Use JQL for Complex Searches:** For more advanced organization and reporting, use Jira Query Language (JQL) in the advanced search mode (e.g., priority in (Highest, High) AND labels in ("backend", "UI") ORDER BY created DESC).
- **Maintain Consistency:** Establish clear labeling conventions with your team and regularly review and clean up the list of labels to avoid duplicates or misspellings.
- **Bulk Actions:** Use the list view to select multiple issues and perform bulk changes, such as applying or removing a specific label or updating the priority across several tasks at once.

JQL

Jira Query Language (JQL) is a powerful, structured query language for searching, filtering, and reporting on issues within Atlassian's project management tool, Jira [1]. It acts as Jira's "Google search" for data, providing a flexible way to locate specific issues that simple searches cannot find [1, 2].

What is Jira Query Language (JQL)?

JQL is used to write specific queries to find a precise set of issues [1]. Unlike a simple search, which might be limited to a few general fields, JQL allows you to specify detailed criteria across numerous standard and custom fields using a command-line interface [3]. This capability makes it a fundamental skill for anyone managing data in Jira, from project managers to developers [1].

Core Components of JQL

A JQL query is built from three basic parts:

1. **Fields**: These are the individual data points in Jira issues, such as project, priority, status, assignee, created date ,etc.
2. **Operators**: These are the commands that connect fields to values and determine the logic of the search. Common examples include:

1. = (equals)
2. != (does not equal)
3. IN (is one of)
4. ~ (contains text)
5. > (greater than) [1, 3]

3. **Values**: These are the actual data you are searching for, such as "Marketing", "High", "In Progress", or a specific user's name [1].

A simple JQL query looks like this:

project = "Website Redesign" AND status = "In Progress"

This query uses the project and status fields, the = operator, and combines the conditions with the AND keyword [1].

Basic Date & Time Searches

- **Specific Date/Time:** created >= "2024-01-01 09:00" (Issues created on or after Jan 1, 2024, at 9 AM).
- **Date Range:** resolved >= 2024-01-01 AND resolved <= 2024-01-31 (Issues resolved in January 2024).
- **Relative Dates:** updated >= -1d (Issues updated in the last day).
- **Using now():** dueDate < now() (Issues due before the current moment).

Advanced Date Functions

- startOfDay() / endOfDay(): created >= startOfDay() (Issues created today).
- startOfWeek() / endOfWeek(): dueDate < endOfWeek() (Issues due by the end of this week).
- startOfMonth() / endOfMonth(): worklogdate >= startOfMonth() (Work logged this month).
- startOfYear() / endOfYear(): created >= startOfYear() (Issues created this year).

Link-Based JQL Queries

Link-based queries filter issues based on their relationship (link type) to other issues.

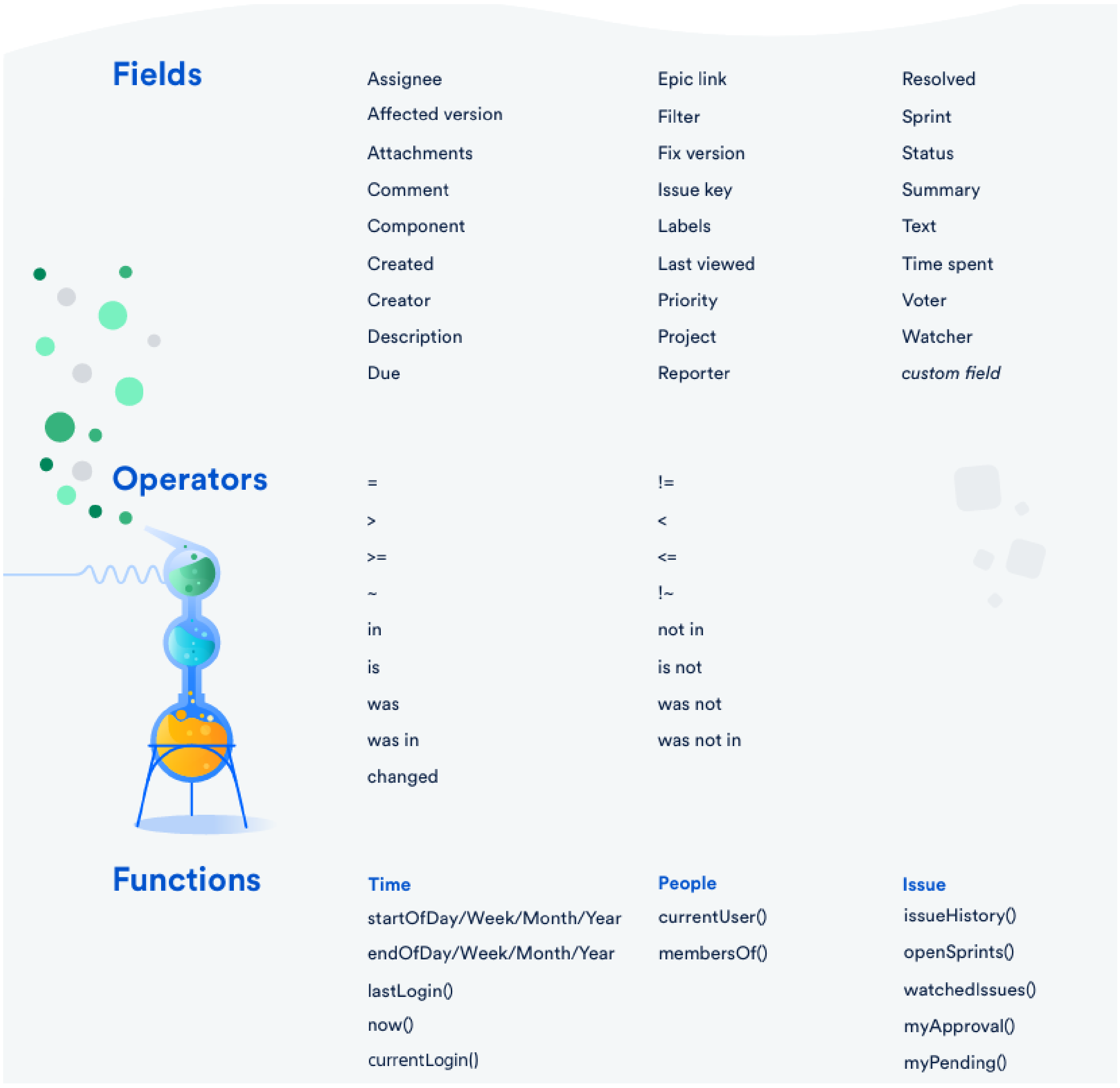
- **Find issues linked to a specific issue:**issue in linkedIssues("ISSUE-123")
- **Find issues linked with a specific link type:**issue in linkedIssues("ISSUE-123", "blocks")
  (Common link types include "blocks", "relates to", "is caused by", "duplicates")
- **Find all issues that have any links:**issueFunction in hasLinks()
- **Find issues linked to a set of issues returned by another query (subquery):** issueFunction in linkedIssuesOf("project = 'Project Name' AND status = 'Open'")
- **Find unresolved issues blocked by open issues (example of linkedIssuesOf):** issueFunction in linkedIssuesOf ("status = 'to do'","blocks") AND resolution = empty
- **Recursive linked issues:** issueFunction in linkedIssuesOfRecursive("issue = JIRA-1") (Requires ScriptRunner or similar app)

Hierarchy JQL Queries

Hierarchy queries filter issues based on their position within the Jira hierarchy (e.g., Epic, Story, Sub-task). Jira's native JQL has limited hierarchy support beyond Epics and sub-tasks, with most multi-level hierarchy features requiring Jira Advanced Roadmaps or marketplace apps.

Standard Jira Hierarchy (Epic/Story/Sub-task)

- **Find all stories/tasks under a specific Epic:**"Epic Link" = "EPIC-123" (For Jira Server/Data Center)
  parent in "EPIC-123" (For some Jira Cloud configurations)
- **Find all sub-tasks for a specific story/task:**parent = "STORY-456"
- **Find all issues (including stories/tasks and sub-tasks) within an Epic:**"Epic Link" = "EPIC-123" OR issuekey = "EPIC-123"



Fields

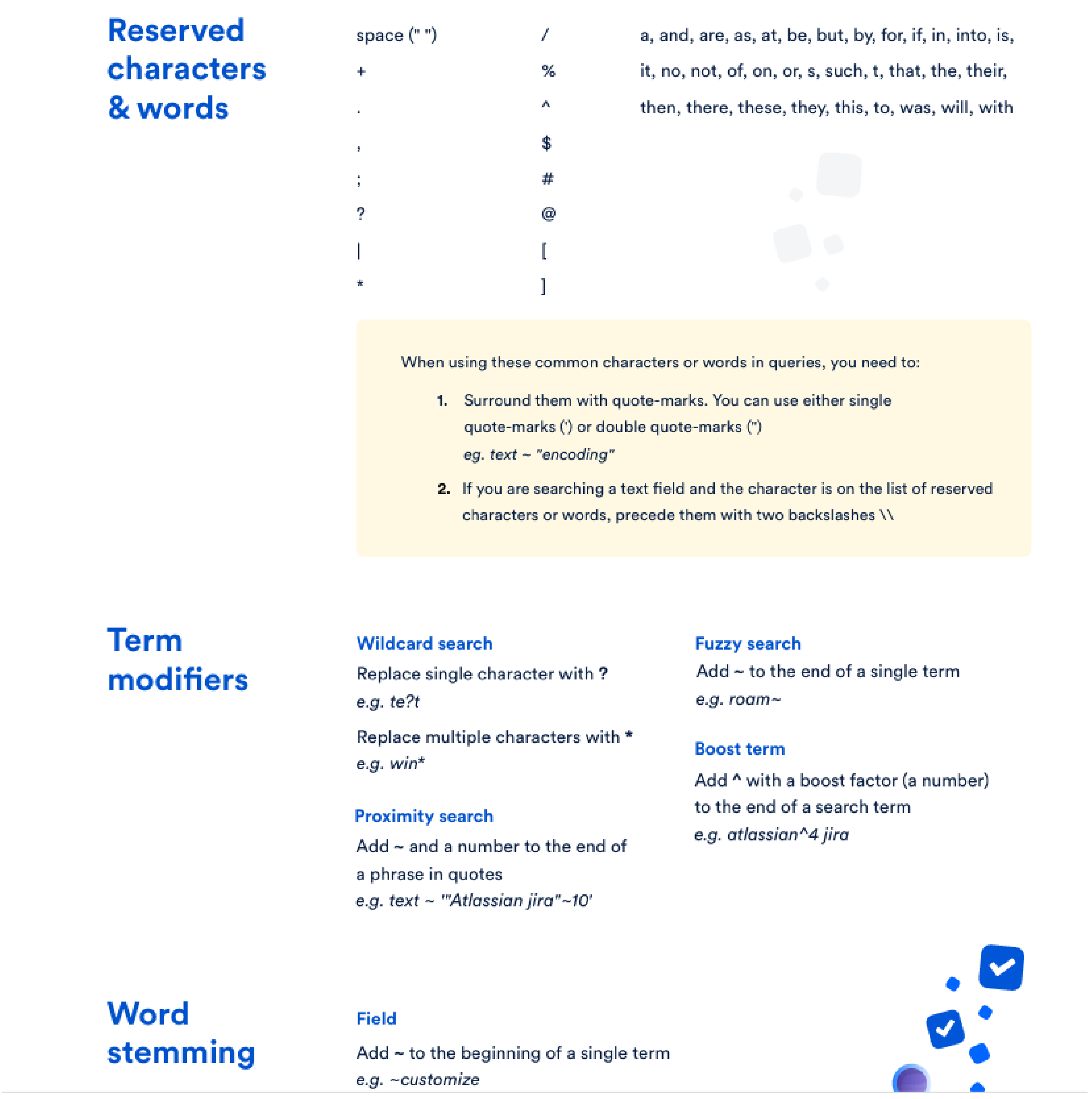
Assignee	Epic link	Resolved
Affected version	Filter	Sprint
Attachments	Fix version	Status
Comment	Issue key	Summary
Component	Labels	Text
Created	Last viewed	Time spent
Creator	Priority	Voter
Description	Project	Watcher
Due	Reporter	*custom field*

Operators

=	!=
>	<
>=	<=
~	!~
in	not in
is	is not
was	was not
was in	was not in
changed	

Functions

Time	People	Issue
startOfDay/Week/Month/Year	currentUser()	issueHistory()
endOfDay/Week/Month/Year	membersOf()	openSprints()
lastLogin()		watchedIssues()
now()		myApproval()
currentLogin()		myPending()



Reserved characters & words

space (" ")	/	a, and, are, as, at, be, but, by, for, if, in, into, is,
+	%	it, no, not, of, on, or, s, such, t, that, the, their,
.	^	then, there, these, they, this, to, was, will, with
,	$	
;	#	
?	@	
|	[	
*	]	

When using these common characters or words in queries, you need to:

1. Surround them with quote-marks. You can use either single quote-marks (') or double quote-marks (")
   eg. text ~ "encoding"
2. If you are searching a text field and the character is on the list of reserved characters or words, precede them with two backslashes \\

Term modifiers

Wildcard search
Replace single character with **?**
e.g. te?t

Replace multiple characters with *
*e.g. win**

Proximity search
Add ~ and a number to the end of a phrase in quotes
e.g. text ~ '"Atlassian jira"~10'

Fuzzy search
Add ~ to the end of a single term
e.g. roam~

Boost term
Add ^ with a boost factor (a number) to the end of a search term
e.g. atlassian^4 jira

Word stemming

Field
Add ~ to the beginning of a single term
e.g. ~customize

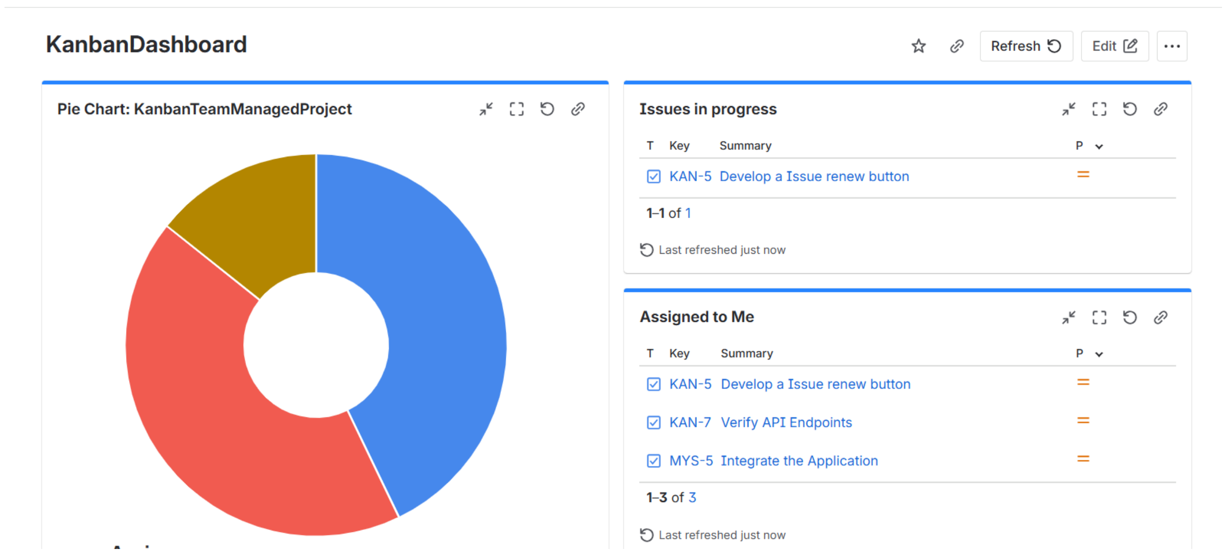
Dashboards

Jira dashboards are customizable, visual homepages that provide a real-time overview of project progress, team performance, and key metrics using configurable "gadgets" like charts, graphs, and filter results, helping teams track work, spot bottlenecks, and stay aligned, with options to create personal dashboards or share them across teams with specific permissions. You create them by selecting "Dashboards > Create Dashboard," naming it, adding gadgets (like Pie Charts or Filter Results) that use JQL filters, customizing the layout, and setting viewing permissions.

Key Features & Uses

- **Visibility:** Offer a snapshot of project status, issue distribution, and workload.
- **Customization:** Add/remove gadgets, change layouts (columns), and drag-and-drop to organize information.

- **Gadgets:** Use widgets like Pie Charts (status/assignee), Two-Dimensional Filter Statistics, or Filter Results to display data**.**
- **Filters:** Gadgets rely on JQL (Jira Query Language) filters; these filters must also be shared for others to see the data.
- **Sharing & Permissions:** Can be private, or shared with specific users, groups, projects, or publicly, requiring correct filter permissions**.**
- **Wallboards:** Can be set up to display on large screens for continuous team updates.



KanbanDashboard

Pie Chart: KanbanTeamManagedProject

Issues in progress

T	Key	Summary	P
☑	KAN-5	Develop a Issue renew button	=

1–1 of 1

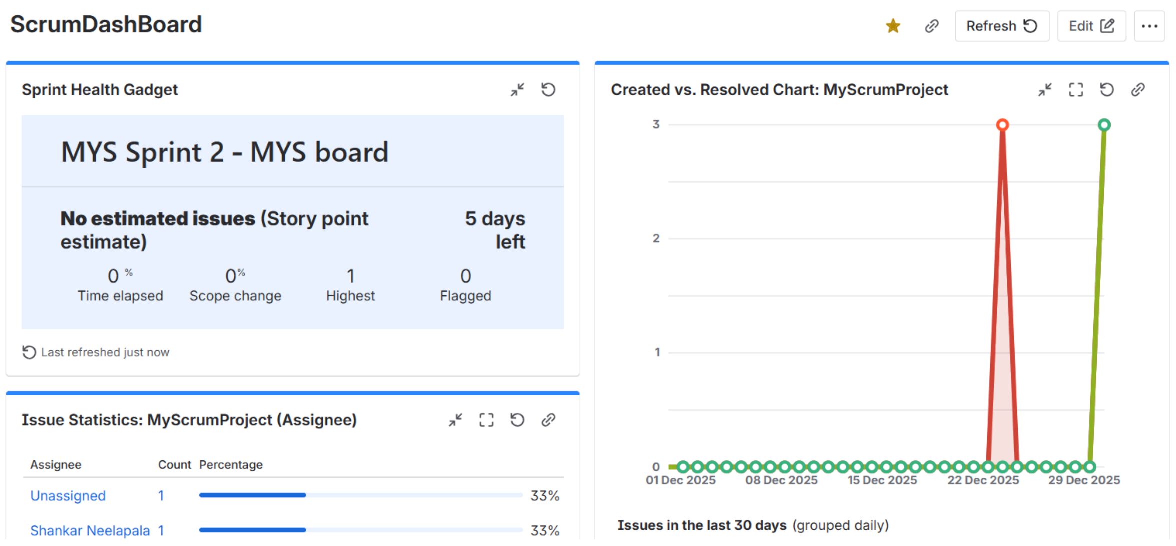
Last refreshed just now

Assigned to Me

T	Key	Summary	P
☑	KAN-5	Develop a Issue renew button	=
☑	KAN-7	Verify API Endpoints	=
☑	MYS-5	Integrate the Application	=

1–3 of 3

Last refreshed just now



ScrumDashBoard

Sprint Health Gadget

MYS Sprint 2 - MYS board

No estimated issues (Story point estimate)

5 days left

0 %	0%	1	0
Time elapsed	Scope change	Highest	Flagged

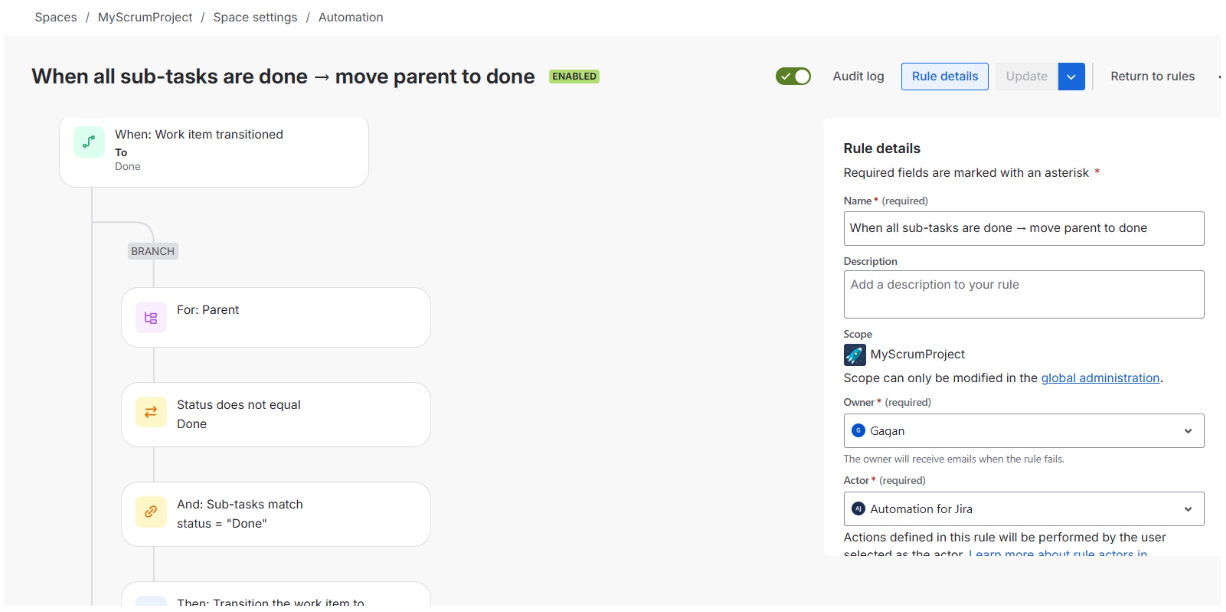
Last refreshed just now

Created vs. Resolved Chart: MyScrumProject

Issues in the last 30 days (grouped daily)

Issue Statistics: MyScrumProject (Assignee)

Assignee	Count	Percentage
Unassigned	1	33%
Shankar Neelapala	1	33%



Spaces / MyScrumProject / Space settings / Automation

When all sub-tasks are done → move parent to done ENABLED

Audit log Rule details Update Return to rules

When: Work item transitioned
To
Done

BRANCH

For: Parent

Status does not equal
Done

And: Sub-tasks match
status = "Done"

Then: Transition the work item to

Rule details

Required fields are marked with an asterisk *

Name * (required)
When all sub-tasks are done → move parent to done

Description
Add a description to your rule

Scope
MyScrumProject
Scope can only be modified in the global administration.

Owner * (required)
Gagan

The owner will receive emails when the rule fails.

Actor * (required)
Automation for Jira

Actions defined in this rule will be performed by the user selected as the actor. Learn more about rule actors in