Introduction

Jira is a project and issue management system that was made by atlassian, and is a globally used system in software development and enterprise environments. It provides full support for agile methodologies through its availability of built in scrum and kanban boards, reporting tools and automation features and an extensive ecosystem of third party applications. Jira is primarily delivered as a cloud based service and exposes rich REST APIs that enable integration with other development tools as well as external systems.

On the other hand, Redmine is an open source issue and project tracking platform that is built on Ruby on Rails. It supports multiple projects, configurable workflows, granular role based access control, time tracking, gantt charts, calendars and built in wiki as well as a forum feature. Redmine is usually deployed in a self-hosted environment and although commercial, hosting options exist. It is distributed under the GNU GPLv2 license which allows full access and modification to its source code.

Shared Characteristics

Despite differences in its ideas and deployment, Jira and Redmine has many similar fundamental capabilities:

- Both of the platforms provide robust issue tracking with customizable workflows, statuses, fields as well as permission schemes.
- They support management of several projects in one system
- Both of them offer API access to enable integrations and automation, Jira provides a lot of REST APIs and Redmine offers more JSON/XML-based endpoints.
- Notifications like emails and RSS feeds for redmine are also available so that the users that are using the system are informed when there are changes.
- Both Redmine and Jira can be extended through plugins or add-ons either by the Atlassian Marketplace or the Redmine plugin community.

Key Differences

Licensing and Cost Structure

Jira follows a proprietary licensing model. While a free tier is available for small teams, most organizations must subscribe to paid plans that are priced per user and vary by feature set and region.

Redmine is completely free to use for its licensing. However, organizations must factor in for infrastructure, hosting, and maintenance costs, either by managing them internally or using third party softwares.

Deployment Approach

Jira’s strategic focus is on cloud deployment. Atlassian has announced that the data center offerings will be discontinued, with the full discontinuation set for 2029. Jira Cloud now supports a huge amount of installations, and can house up to 100,000 users on one instance.

Redmine is mainly built for self-hosting and runs across multiple operating systems and database backends. Even though this encourages flexibility, it also makes a user dependency when it comes to installation, upgrades, and system reliability on the adopting organization.

Agile Functionality

Jira includes native Agile tooling, such as Scrum and Kanban boards, sprint planning, and built-in reports like burndown and velocity charts.

Redmine provides timeline-oriented features such as Gantt charts and calendars by default, but Agile boards and related analytics typically require additional plugins.

Automation and Query Capabilities

Jira distinguishes itself with its visual automation engine and Jira Query Language (JQL), which allow users to define complex rules and perform advanced searches without external tooling.

Redmine does not include equivalent built-in automation or query languages. Customization in this area usually depends on plugins, scripting, or API-based solutions.

Integrations and Ecosystem

Jira offers tightly integrated support for popular development and collaboration tools such as GitHub, GitLab, Bitbucket, and CI/CD platforms, supported by thousands of marketplace applications.

Redmine includes basic source control integration out of the box and can be extended through community plugins. However, plugin quality and maintenance vary, often requiring additional administrative effort.

Security, Identity, and Compliance

Jira Cloud provides enterprise-grade identity management features, including SAML-based single sign-on, SCIM provisioning, audit logging, and expanding data residency options.

With Redmine, security and compliance depend entirely on how the system is deployed and managed. Features such as SSO, backups, and compliance controls must be implemented and maintained by the hosting organization, often via external tools or plugins.

Selection Considerations

When Jira Is the Better Fit

Jira is well suited for organizations that require a managed cloud solution with minimal operational overhead, strong Agile support, seamless integrations, and enterprise-level governance features. It is particularly attractive to teams that prioritize rapid onboarding, standardized workflows, and vendor-supported scalability.

When Redmine Is the Better Fit

Redmine is an appropriate choice for teams that value full control over their infrastructure, require on-premise deployment, or operate under strict budget constraints. It is especially suitable for organizations with sufficient technical expertise to manage servers, customize features at the source-code level, and maintain the platform independently.

Total Cost of Ownership (TCO)

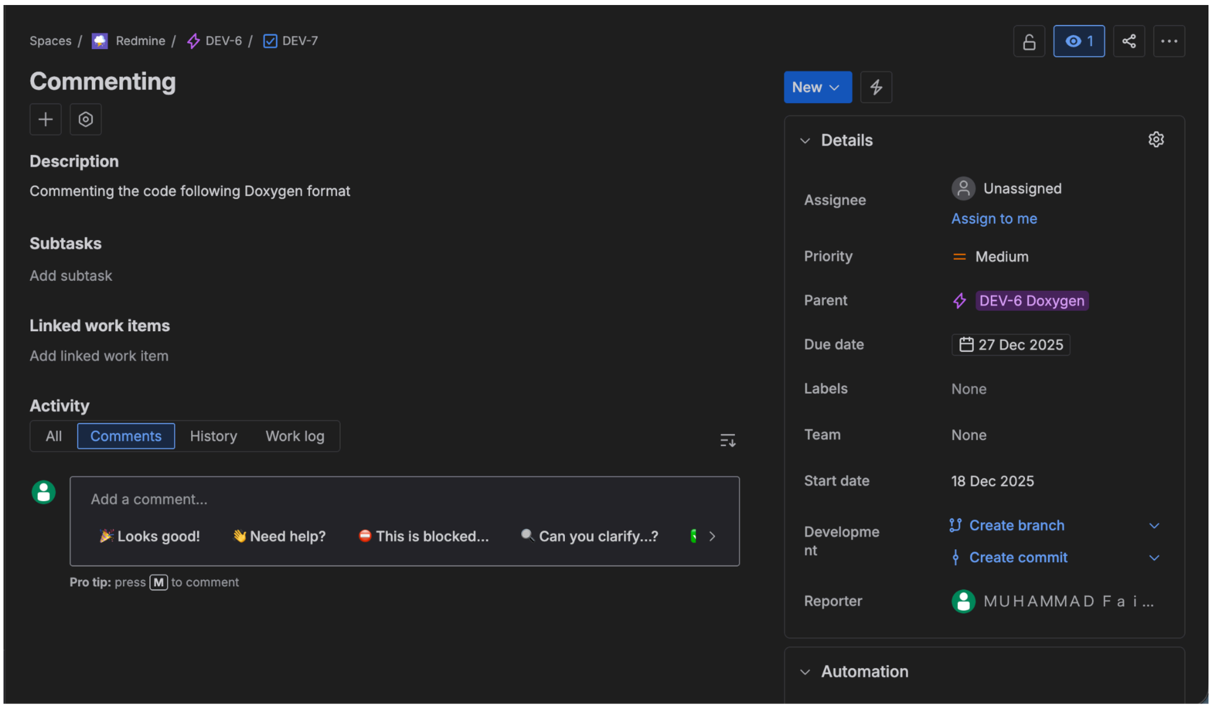
Jira’s cost model is based on recurring subscription fees, which provide predictable expenses and include hosting, maintenance, and automatic updates. Additional costs may arise from premium marketplace applications.

Redmine eliminates licensing fees but shifts costs toward infrastructure, system administration, upgrades, and potential commercial plugins or managed hosting services. As a result, its total cost of ownership varies significantly depending on internal capabilities and deployment choices.

Creating Epics and Tasks in Jira

Planning in Jira uses a timeline, which is done through Epics and Tasks. Epic and Tasks are like the Issues in Redmine. However, there is a parent-child relationship that helps to group the Tasks together in one Epic. The Epics and Tasks can then be viewed within the timeline which looks like a Gantt chart. This feature is also like Redmine. It is worth noting that for the free version of Jira, a non-direct child of Epics (child of Tasks, Tasks not linked to an Epic, etc.) will not be shown within the timeline, but will be shown on the Jira Boards section.

To create an Epic and Task, first the user needs to navigate to the project board, click “Create” and choose “Epic” as the issue type. For Tasks, the user should click “Tasks” as issue type. To link Tasks to an Epic, while creating Tasks the user needs to select an Epic in the “Epic Link” field


Spaces / Redmine / DEV-6 / DEV-7

Commenting

Description

Commenting the code following Doxygen format

Subtasks

Add subtask

Linked work items

Add linked work item

Activity

All | Comments | History | Work log

Add a comment...

🎉 Looks good! 👋 Need help? ⛔ This is blocked... 🔍 Can you clarify...?

Pro tip: press M to comment

New

Details

Assignee	Unassigned Assign to me
Priority	Medium
Parent	DEV-6 Doxygen
Due date	27 Dec 2025
Labels	None
Team	None
Start date	18 Dec 2025
Development	Create branch Create commit
Reporter	MUHAMMAD F a i ...

Automation

Comparison: Jira vs Redmine

When comparing Jira and Redmine for project management, both tools support hierarchical task organization but with different approaches and philosophies. **Jira** is highly specialized for Agile methodologies, offering native support for Scrum and Kanban workflows with dedicated Epic, Story, and Task issue types that align with Agile terminology. Its interface is polished and feature-rich, with advanced reporting through dashboards, burndown charts, and velocity tracking. However, Jira can feel complex for smaller teams and typically requires paid licensing for meaningful features. **Redmine**, conversely, takes a more generalized approach to project management, using a simpler Parent-Child relationship model where any issue can be a parent to another, regardless of type. It's open-source and free, making it accessible for budget-conscious teams, and offers straightforward issue tracking with customizable workflows, Gantt charts, and time tracking built-in. While Redmine's interface feels more utilitarian and less modern than Jira's, it provides excellent flexibility through plugins and doesn't lock teams into Agile-specific frameworks. Ultimately, Jira excels for Agile teams needing sophisticated sprint management and reporting, while Redmine suits teams wanting a free, flexible, and straightforward project tracking system without methodology constraints.

Gantt Chart Timeline Views in Jira and Redmine

Jira's Timeline/Gantt Chart Approach

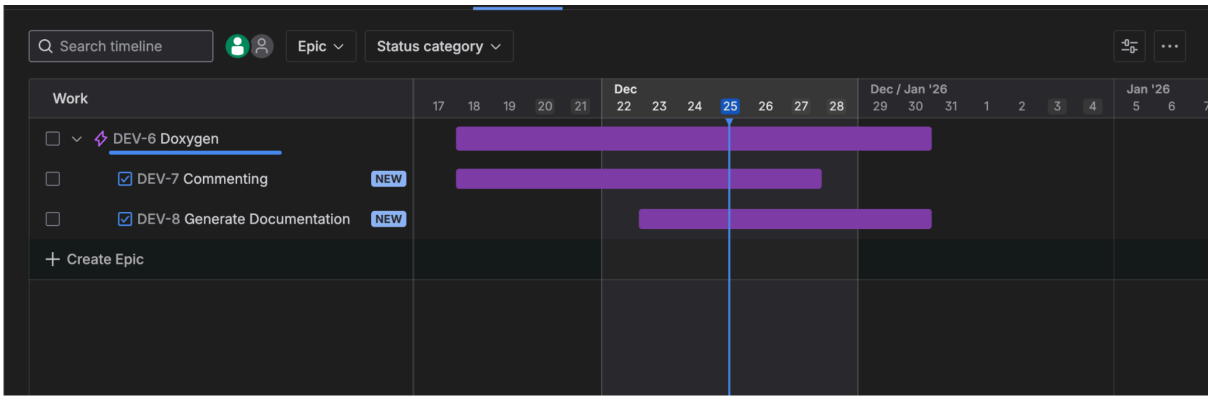
In **Jira**, Gantt chart functionality is not available in the basic version and requires additional solutions. For Jira Cloud, you can use the built-in **Timeline view** (formerly called Advanced Roadmaps in premium plans), which provides a visual roadmap showing Epics, Stories, and Tasks across a time-based horizontal layout. To access it, navigate to your project sidebar and select "Timeline" or create a plan in Jira Premium/Enterprise editions. The Timeline displays issues as horizontal bars spanning their start and end dates, with dependencies shown as connecting lines between items. Epics appear as parent bars with their child Tasks nested beneath or linked visually. You can drag bars to adjust dates, create dependencies by connecting issues, and adjust timelines dynamically. The view color-codes issues by status and allows filtering by assignee, label, or component. However, for more traditional Gantt chart features like critical path analysis or resource leveling, many teams install third-party apps from the Atlassian Marketplace such as **BigPicture**, **ActivityTimeline**, or **Tempo Planner**, which provide comprehensive Gantt functionality with baseline tracking, progress percentages, and resource management overlays.

Redmine's Native Gantt Chart

Redmine includes a **native Gantt chart** out of the box, accessible directly from the project menu by clicking "Gantt" in the left sidebar. This displays all project issues as horizontal bars on a timeline, automatically calculating the span based on start dates, due dates, and estimated hours. Parent issues (Epics) appear as summary bars that encompass all their child tasks, creating a clear hierarchical visualization. The Gantt chart shows task dependencies with arrows connecting related issues, displays completion percentage as filled portions within each bar, and color-codes issues by priority or tracker type. You can zoom the timeline view from days to months to years, and export the chart as a PNG image for sharing. Redmine's Gantt chart automatically reflects the parent-child relationships you've established, so when you set an issue's "Parent task" field, it nests visually beneath that parent in the timeline. The chart updates in real-time as you modify issue dates, progress percentages, or relationships, making it straightforward for traditional waterfall or hybrid project management approaches.

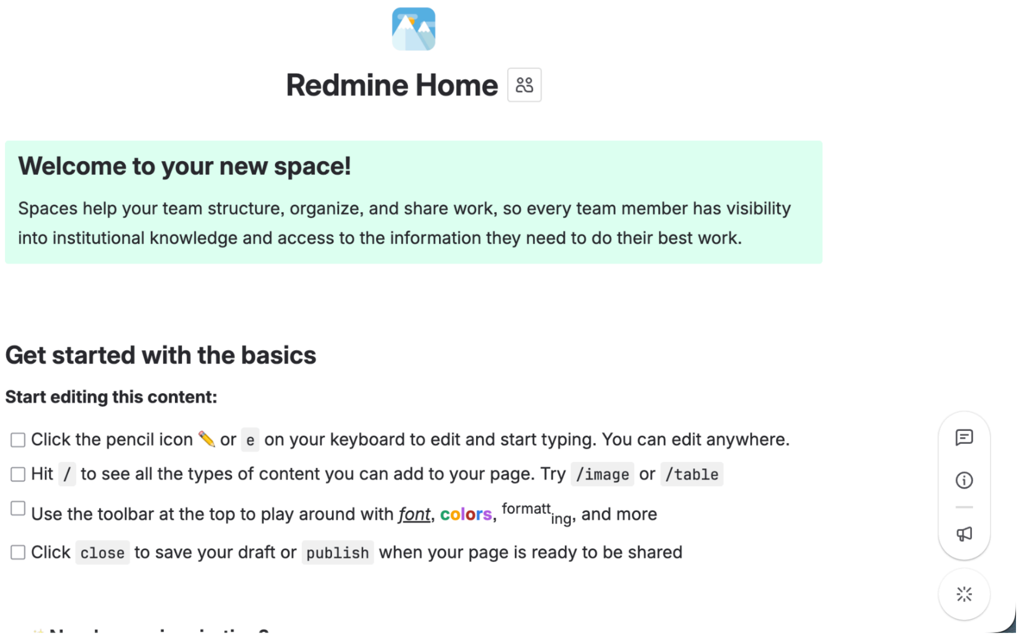
Key Differences in Timeline Display

The primary distinction is that **Redmine provides Gantt charts as a core feature** requiring no additional plugins or premium tiers, making it immediately accessible for teams needing classic project scheduling views. **Jira requires either a premium subscription** for Timeline views or third-party apps for full Gantt capabilities, but offers more sophisticated roadmap planning and Agile-oriented visualizations when properly configured. Redmine's Gantt chart tends to be more technical and utilitarian, ideal for detailed project scheduling, while Jira's Timeline emphasizes strategic planning and release coordination with a modern, interactive interface.



DEV-6 Doxygen
DEV-7 Commenting NEW
DEV-8 Generate Documentation NEW
Create Epic

Pages in Jira are used for project documentation. Pages can be used to store information such as team member information, project requirements, meeting notes, and more. While similar to Wiki in Redmine, the most significant difference is the use of templates in Jira. The user is able to select the templates given by the website or select from the community-created templates. Examples of templates include templates for home pages, about the team members section, and project requirements page templates. It is worth noting that Pages uses Confluence, which is a different tool from Jira, as Confluence which stores pages. Even though it is a different tool, it has a seamless integration with Jira as they are both from Atlassian.



Redmine Home

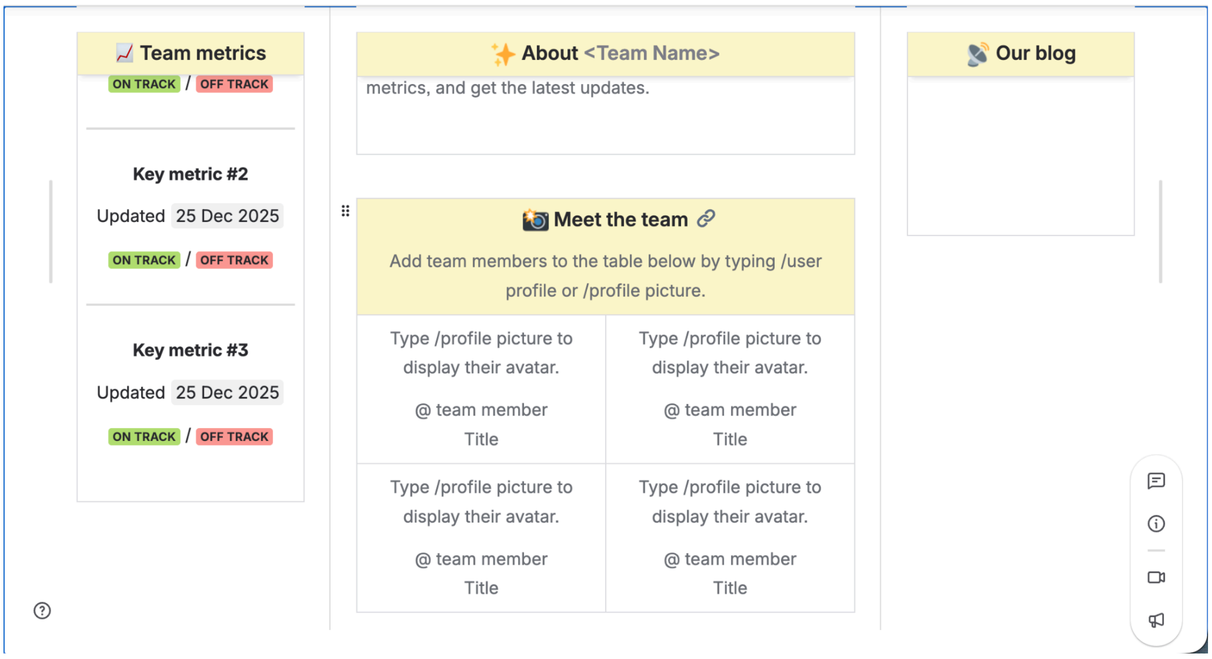
Welcome to your new space!

Spaces help your team structure, organize, and share work, so every team member has visibility into institutional knowledge and access to the information they need to do their best work.

Get started with the basics

Start editing this content:

- ☐ Click the pencil icon ✏️ or `e` on your keyboard to edit and start typing. You can edit anywhere.
- ☐ Hit `/` to see all the types of content you can add to your page. Try `/image` or `/table`
- ☐ Use the toolbar at the top to play around with *font*, colors, formatting, and more
- ☐ Click `close` to save your draft or `publish` when your page is ready to be shared



📈 Team metrics

ON TRACK / OFF TRACK

Key metric #2

Updated 25 Dec 2025

ON TRACK / OFF TRACK

Key metric #3

Updated 25 Dec 2025

ON TRACK / OFF TRACK

✨ About <Team Name>

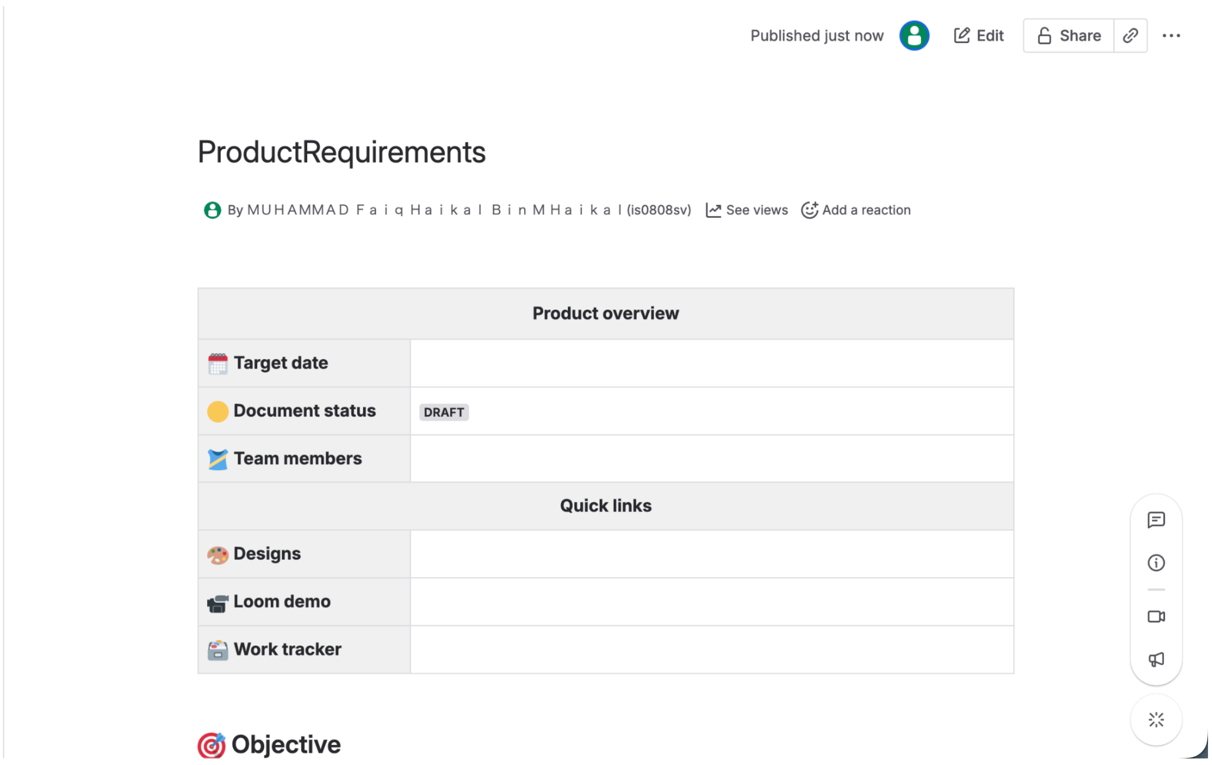
metrics, and get the latest updates.

📸 Meet the team

Add team members to the table below by typing /user profile or /profile picture.

Type /profile picture to display their avatar. @ team member Title	Type /profile picture to display their avatar. @ team member Title
Type /profile picture to display their avatar. @ team member Title	Type /profile picture to display their avatar. @ team member Title

📡 Our blog



ProductRequirements

By M U H A M M A D F a i q H a i k a l B i n M H a i k a l (is0808sv) See views Add a reaction

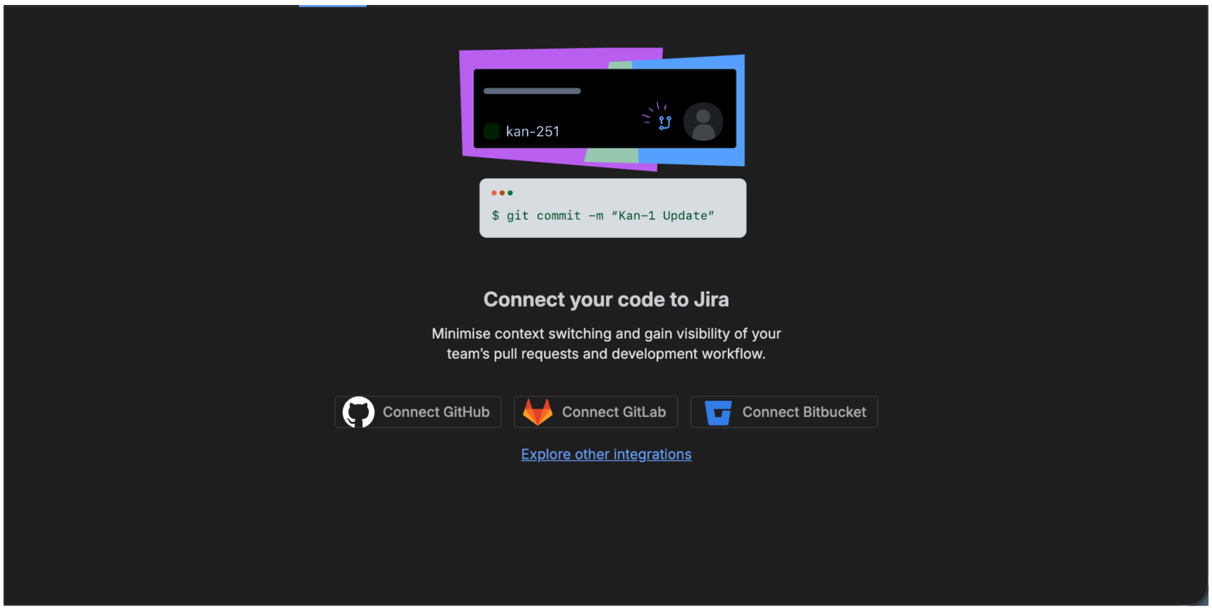
Product overview	
📅 Target date	
🟡 Document status	DRAFT
👥 Team members	
Quick links	
🎨 Designs	
📹 Loom demo	
🗃️ Work tracker	

🎯 Objective

Steps to Connect Jira and GitHub

1. **Open a Jira Issue**
   Firstly, navigate to a task in Jira. This is where the development progress will be tracked. In the tasks's **Details** panel, locate the **Development** section.

2. **Initiate the GitHub Integration**
   Under the Development section, select **Create branch** or **Connect GitHub**. Jira will prompt you to connect a source code management tool.



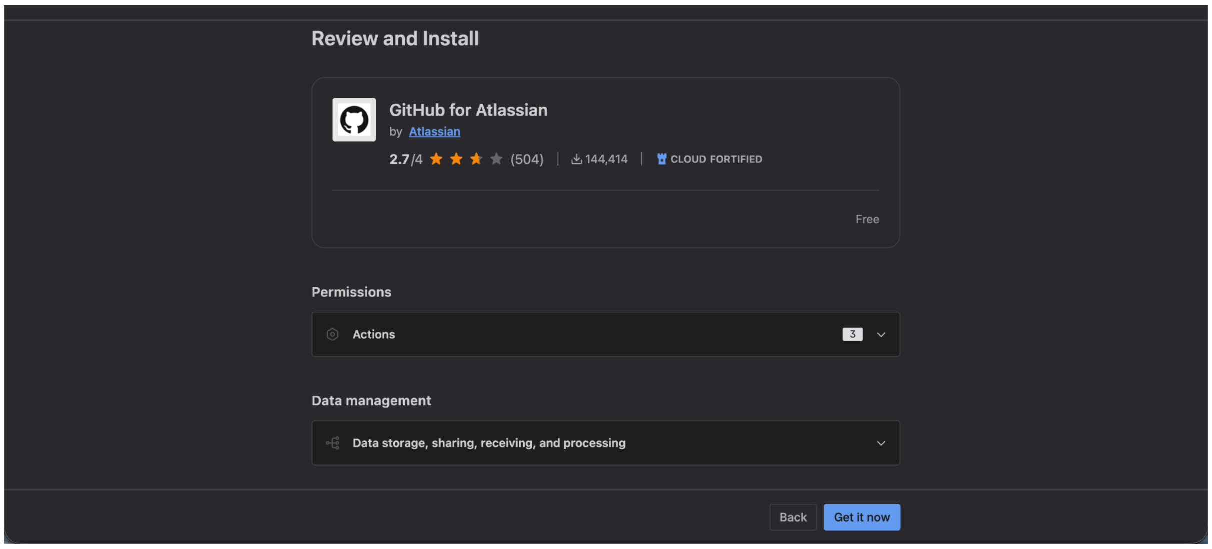
kan-251

```
$ git commit -m "Kan-1 Update"
```

Connect your code to Jira

Minimise context switching and gain visibility of your team's pull requests and development workflow.

3. **Select GitHub as the Repository Provider**
   From the available integrations, choose **GitHub**. You will be redirected to the Atlassian Marketplace to install the **GitHub for Atlassian** application if it is not already installed.



Review and Install

GitHub for Atlassian
by Atlassian
2.7/4 ★★★★ (504) | 144,414 | CLOUD FORTIFIED

Free

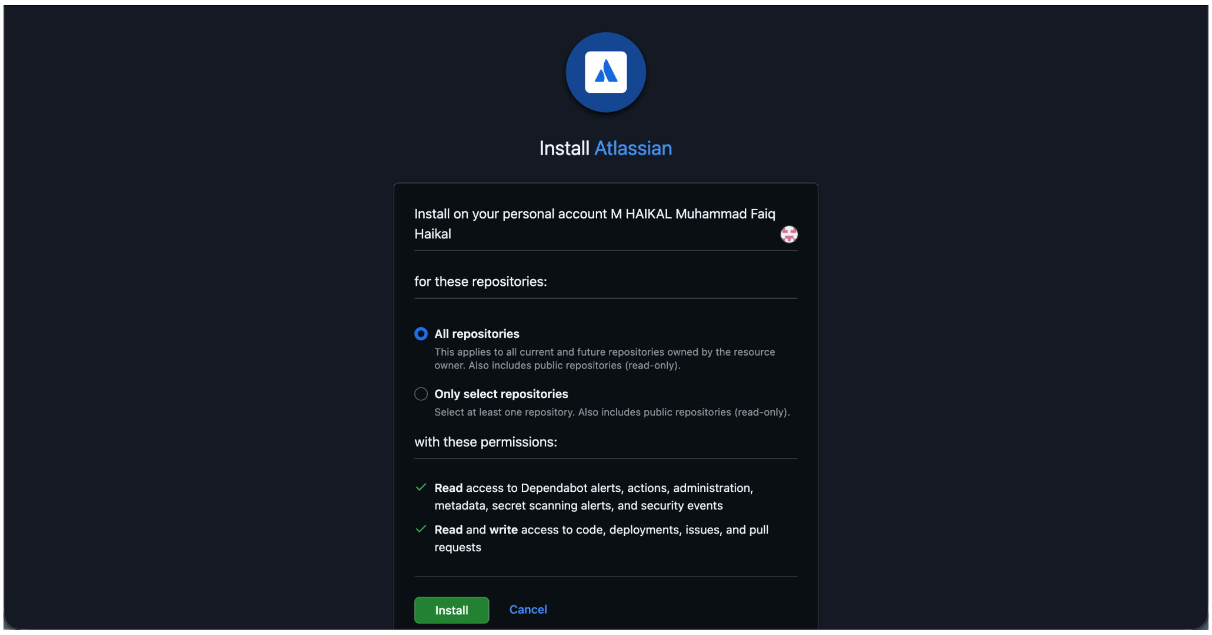
Permissions

Actions 3

Data management

Data storage, sharing, receiving, and processing

4. **Install GitHub for Atlassian**
   Click **Get it now**, review the requested permissions, and proceed with the installation. Select the appropriate Atlassian site (Jira workspace) where the integration should be enabled.



Install Atlassian

Install on your personal account M HAIKAL Muhammad Faiq Haikal

for these repositories:

[✓ All repositories]
This applies to all current and future repositories owned by the resource owner. Also includes public repositories (read-only).

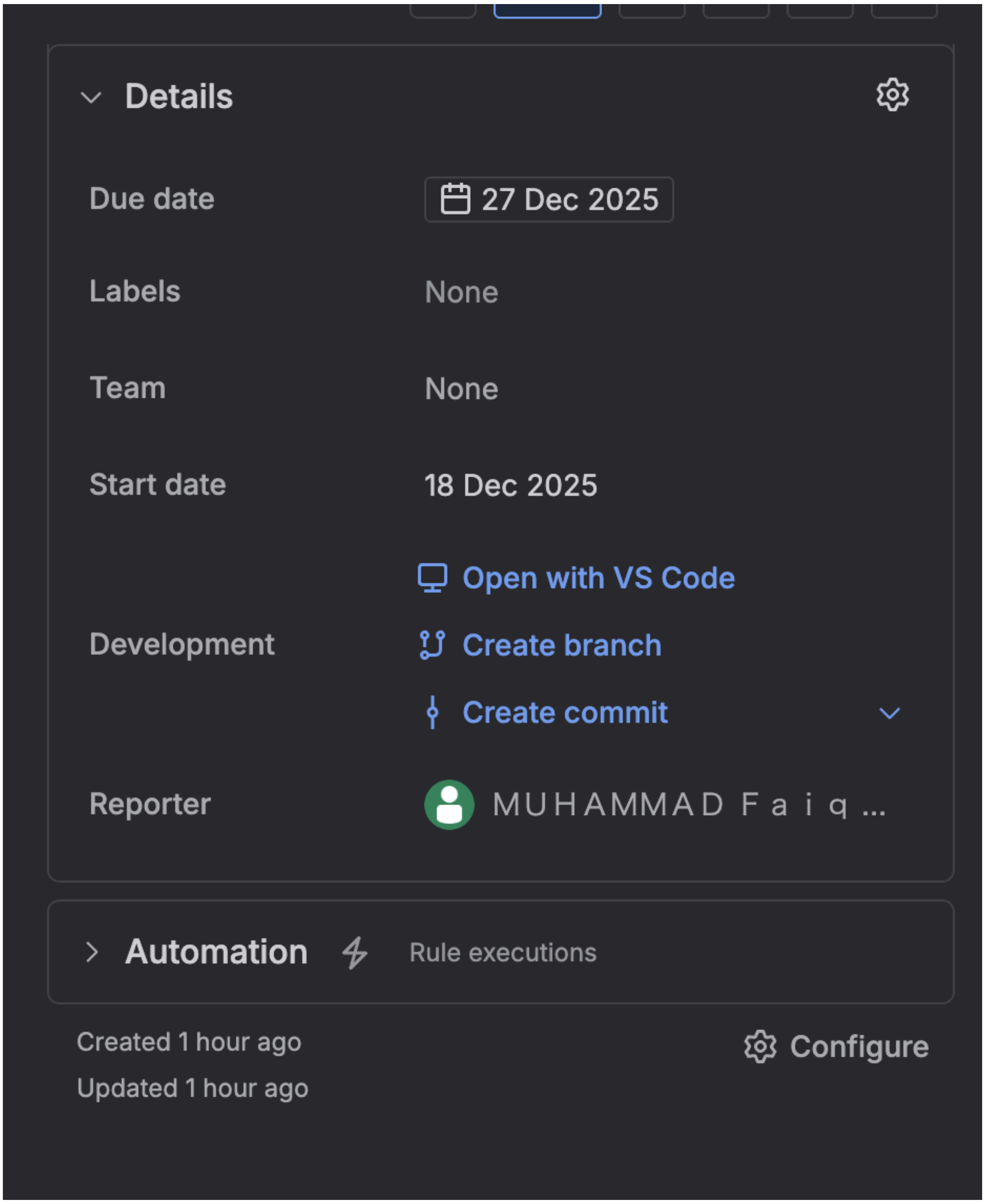
Only select repositories
Select at least one repository. Also includes public repositories (read-only).

with these permissions:

✓ **Read** access to Dependabot alerts, actions, administration, metadata, secret scanning alerts, and security events

✓ **Read** and **write** access to code, deployments, issues, and pull requests

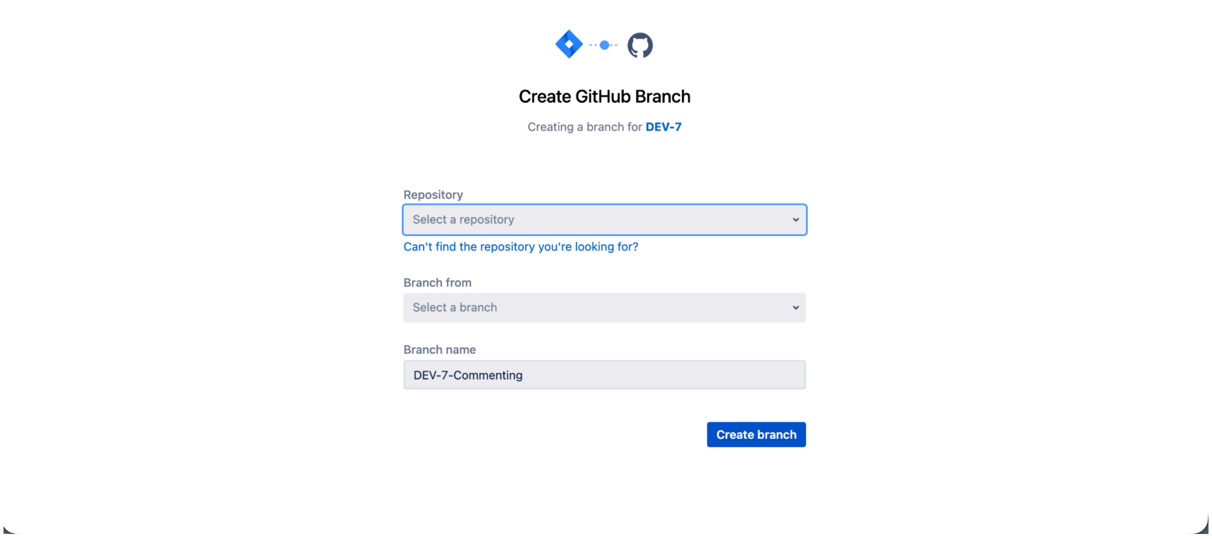
5. **Authorize GitHub Access**
   Log in to your GitHub account when prompted. Choose whether to grant access to **all repositories** or **only selected repositories**, depending on project requirements. Confirm the requested read and write permissions.
6. **Complete the Installation**
   Finalize the setup by reviewing the configuration and confirming the installation. Once completed, Jira and GitHub will be linked at the workspace level.
7. **Create a Branch from Jira**
   Return to the Jira issue and select **Create branch**. Choose the repository and base branch (e.g., main or develop). Jira will automatically generate a branch name that includes the issue key (e.g., DEV-7-commenting).



Details

Due date	27 Dec 2025
Labels	None
Team	None
Start date	18 Dec 2025
Development	Open with VS Code Create branch Create commit
Reporter	MUHAMMAD Faiq ...

Created 1 hour ago
Updated 1 hour ago



Create GitHub Branch

Creating a branch for DEV-7

Repository

Select a repository

Can't find the repository you're looking for?

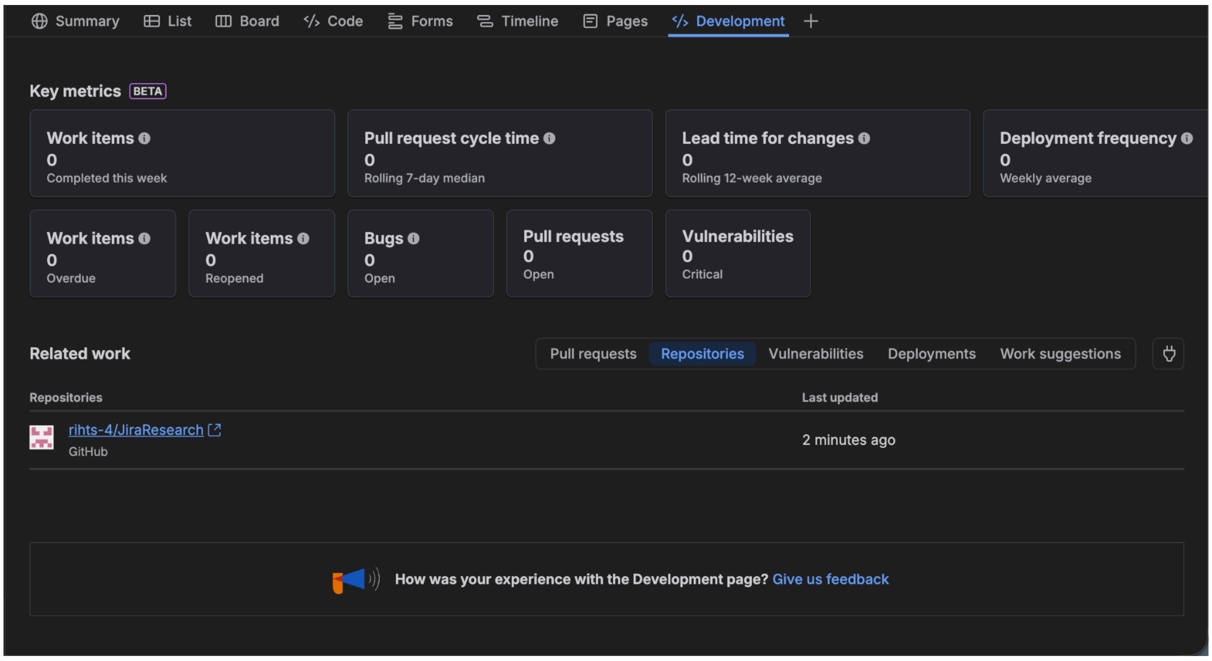
Branch from

Select a branch

Branch name

DEV-7-Commenting

Create branch



Key metrics BETA

Work items 0 Completed this week	Pull request cycle time 0 Rolling 7-day median	Lead time for changes 0 Rolling 12-week average	Deployment frequency 0 Weekly average

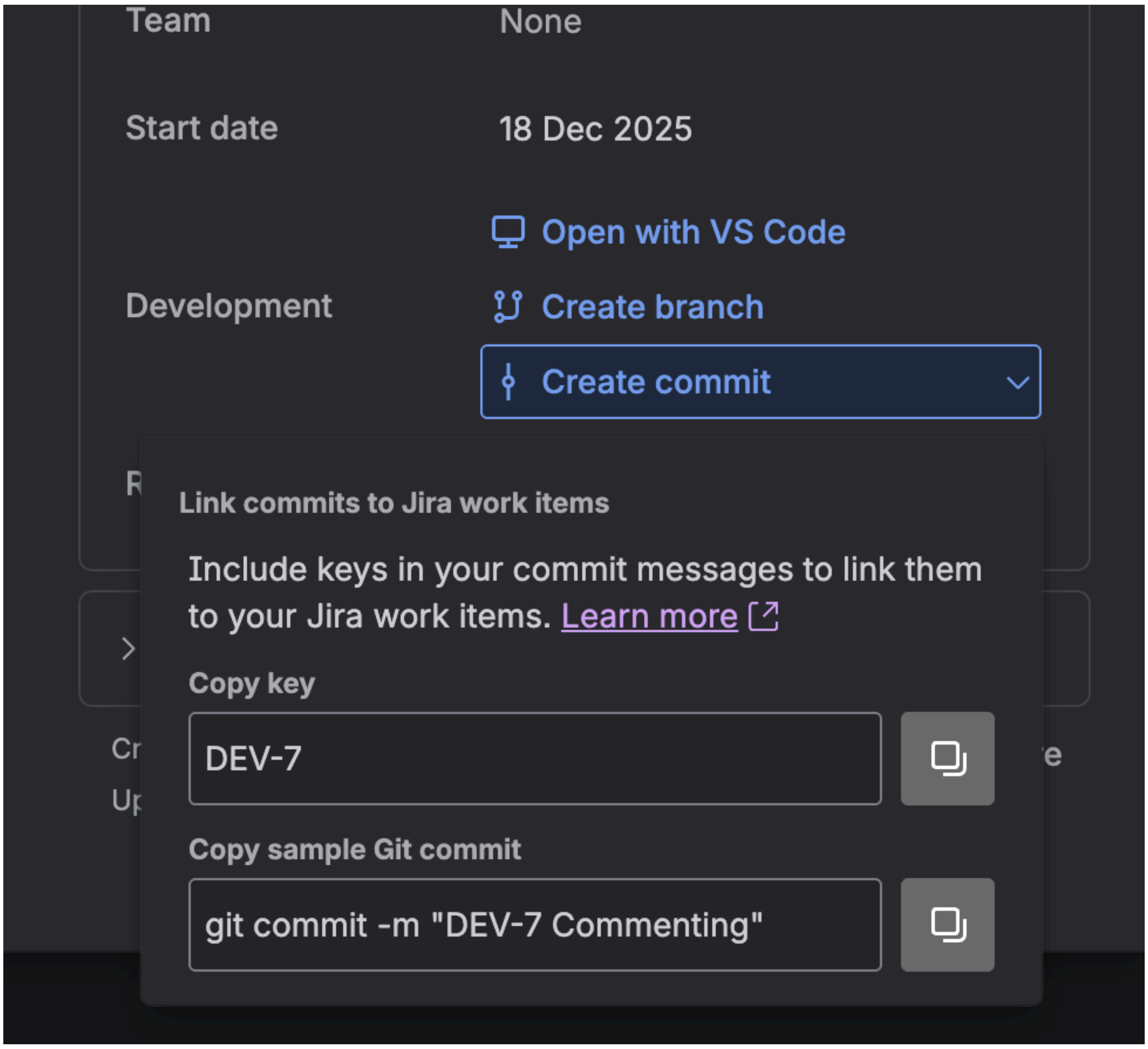
Work items 0 Overdue	Work items 0 Reopened	Bugs 0 Open	Pull requests 0 Open	Vulnerabilities 0 Critical

Related work

Repositories	Last updated
rihts-4/JiraResearch GitHub	2 minutes ago

How was your experience with the Development page? Give us feedback

8. **Link Commits and Pull Requests**
   When committing code, include the Jira issue key in the commit message (e.g., git commit -m "DEV-7 Add commenting feature"). Jira will automatically detect and display commits, branches, and pull requests under the issue's Development section. This part is important as the **issue key helps Jira trace GitHub commits.**



Team None

Start date 18 Dec 2025

Development

Link commits to Jira work items

Include keys in your commit messages to link them to your Jira work items. Learn more

Copy key

DEV-7

Copy sample Git commit

```
git commit -m "DEV-7 Commenting"
```

Outcome

By completing these steps, Jira issues become directly linked to GitHub branches, commits, and pull requests. This integration improves traceability between project requirements and implementation, reduces context switching for developers, and provides real-time visibility into development progress directly from Jira.