Jira Overview

JIRA is an **Incident Management Tool** used for Project Management, Bug Tracking, Issue Tracking and Workflow. JIRA is based on the following three concepts – Project, Issue and Workflow.

Important Points to Note

The following points explain some interesting details of JIRA.

- JIRA is an incident management tool.
- JIRA is developed by **Atlassian Inc.**, an Australian Company.
- JIRA is a platform independent tool; it can be used with any OS.
- JIRA is multi-lingual tool – English, French, German, Japanese, Spanish, etc.
- JIRA supports MySQL, Oracle, PostgreSQL and SQL server in the backend.
- JIRA can be integrated with many other tools – Subversion, GIT, Clearcase, Team Foundation Software, Mercury, Concurrent Version System and many more.

License and Free Trial

The following points describes the legalities of the JIRA Tool.

- JIRA is a commercial tool and available as a Trial version for a limited time.
- To utilize JIRA services, a license is required.
- JIRA provides free license for academic projects.
- A 15-day trial version is available for an individual person to use.

Use of JIRA

Following are some of the most significant uses of JIRA.

- JIRA is used in Bugs, Issues and Change Request Tracking.
- JIRA can be used in Helpdesk, Support and Customer Services to create tickets and track the resolution and status of the created tickets.
- JIRA is useful in Project Management, Task Tracking and Requirement Management.
- JIRA is very useful in Workflow and Process management.

JIRA – Core Features

The following are some of the most important and commonly used features in detail for better understanding.

Boards

JIRA supports Scrum and Kanban boards.
These boards provide an immediate snapshot of the project to the team.
Helps to quickly review the progress of the project and see the status of the individual tasks.
Board workflow can be customized to fulfil the way a team wants to proceed.

Business Project Template

JIRA supports n number of business templates to manage simple tasks and complex tasks like workflow.
Template can be customized based on the team and their approach. Ex: Workflow can be customized based on each team's approach.
Every step is accounted and team can move to achieve their goals.

Task Details

Tasks can be defined at the individual level to track the progress.
Status of every task, comment, attachment and due dates are stored in one place.

Notifications

An email can be sent for a particular task to the users.
Voting and watching features to keep an eye on the progress for the stakeholders.

Use @mention to get the attention of a specific team member at Comments/Description. User will instantly notify if something is assigned or if any feedback is required.

Power Search

JIRA supports a powerful search functionality with Basic, Quick and Advanced features. Use the search tool to find answers like due date, when a task was last updated, what items a team member still needs to finish.
Project information at one place, search within a project.

Reports

JIRA supports more than a dozen reports to track progress over a specific timeframe, deadlines, individual's contribution, etc.
Easy to understand and generate different reports those help to analyse how the team is going on.
Easy to configure these reports and display the matrices to the stakeholders.

Scale with Team Growth

JIRA supports any business team and any project irrespective of size and complexity.

Add –Ins

JIRA supports more than 100 add-ins to connect with different software to make work easy. Wide range of add-ins makes it as universal across the globe.

Multilingual

JIRA supports more than 10 languages those are widely used as English (US, UK, India), French, German, Portuguese, Spanish, Korean, Japanese and Russian.

Mobile App

JIRA is available as a Mobile Application as well.
It is available on Google Play Store and App Store (iTunes) of Apple.
Easy to stay connected with the team while moving anywhere with notification, comments and project activity.

JIRA – Installation

In this chapter, we will learn how to install JIRA on your system.

Important Points to Note

- JIRA is a web application that provides a private website to an individual or a set of requested users belonging to the same company/project.

- JIRA can be run as a Windows Service at the server side.

- JIRA is a pure Java based application and supports all OS platforms like Windows, Linux of different versions or MAC, etc., those satisfy JDK/JRE requirements.

- JIRA supports all famous browsers like Chrome, IE, Mozilla and Safari.

- It supports Mobile browsers as well in mobile views.

System Requirements

Since JIRA is a web-application, it follows the concept of client/server. It means that JIRA can be installed centrally on a server and users can interact with it through web-browsers using a website from any computer.

- **Browser:**

JavaScript should be enabled, the user is recommended not to use any script-blocking tool like NoScript to access full functionality of JIRA.

- **JDK/JRE:**

It is recommended to update JRE/JDK with the latest version. JIRA 6.4 recommends using JRE/JDK version 8.
Since our scope is to consume the JIRA application as end users, we can ignore the server side requirements.

Installation at the Server Side

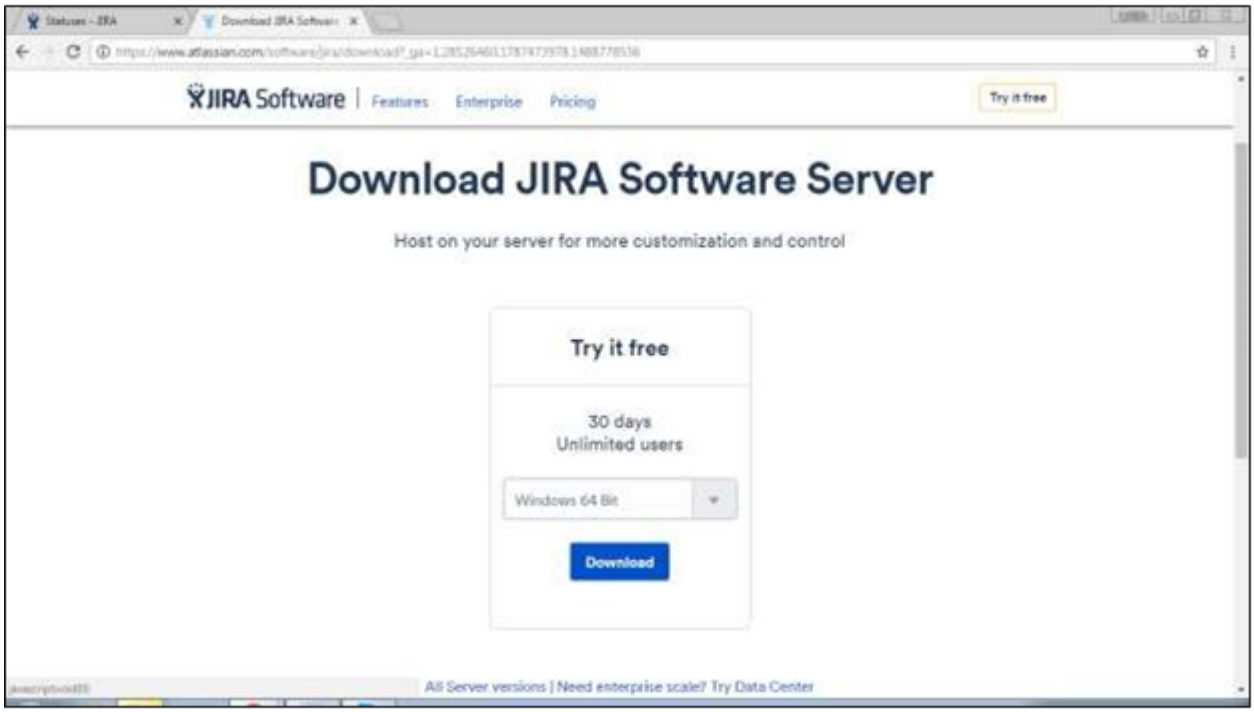
- JIRA follows the Client/Server concept. At the server side, JIRA must be installed before using it as end user.

- At the server side, JIRA must connect with relation database to store issues/application data.

- Download the JIRA Windows Installer.exe file from the following link – https://www.atlassian.com/software/jira/download?_ga=1.28526460.1787473978.1488778536

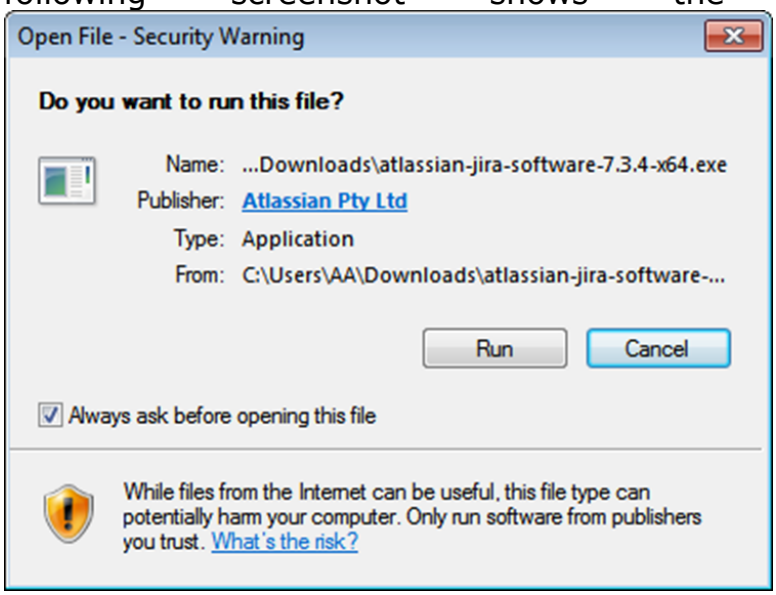
- Select the OS type and click on Download

The following screenshot shows how to download the .exe file for a specific



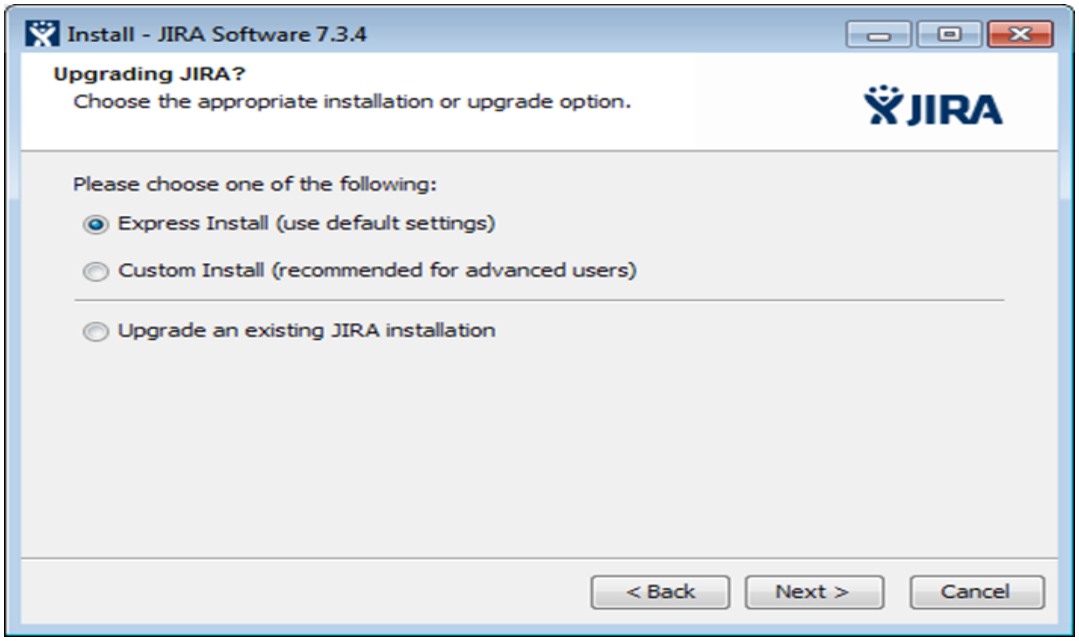
Run the .exe file to run the installation wizard. The following screenshot shows the downloaded .exe file.

After clicking the .exe file, the Run confirmation pop-up displays, click on RUN. The following screenshot shows the RUN confirmation pop-up.

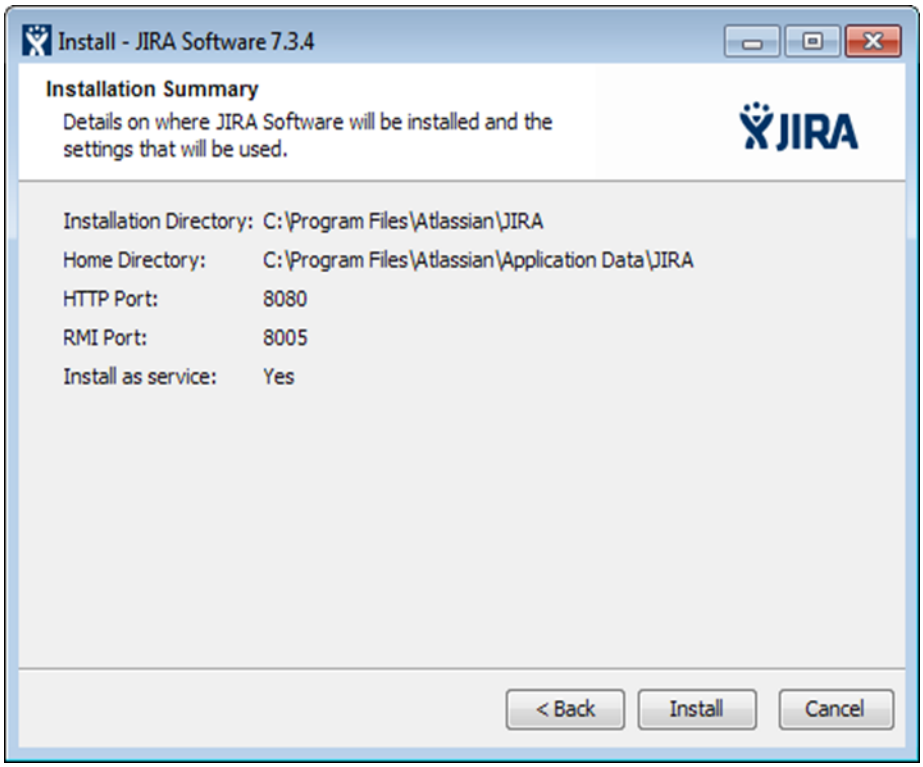


The following JIRA installation wizard displays, click on Next.

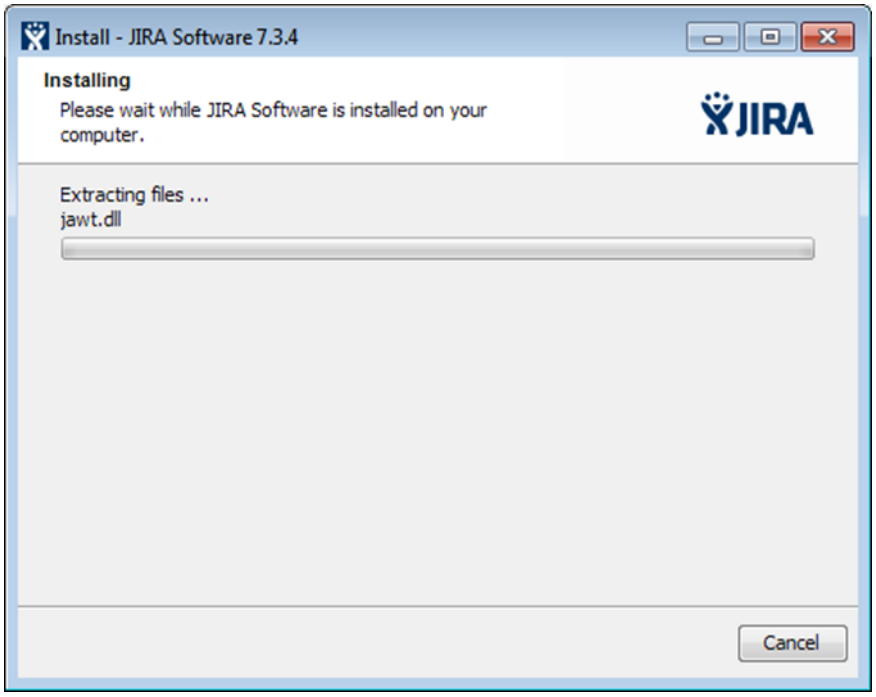
Choose the appropriate installation option as shown in following screenshot and then click on Next.



The installation summary is displayed with the Destination Directory, Home Directory, TCP Ports, etc., as shown in the following screenshot.



Click on Install. JIRA will start installing as displayed in following screenshot. It takes a couple of minutes to finish the installation.



After installation, JIRA will be started automatically if the check box to "Start JIRA Software

7.x.x now” is checked. Then click on Next, if not, it can be accessed using the appropriate Windows Start Menu shortcut.

What is JIRA ?

JIRA is a tool developed by Australian Company Atlassian. It is used for bug tracking, issue tracking, and project management. The name "JIRA" is actually inherited from the Japanese word "Gojira" which means "Godzilla".

The basic use of this tool is to track issue and bugs related to your software and Mobile apps. It is also used for project management. The JIRA dashboard consists of many useful functions and features which make handling of issues easy. Some of the key features are listed below.

About the Jira platform

Products and apps built on top of the Jira platform help teams plan, assign, track, report and manage work. The Jira platform brings teams together for everything from agile software development and customer support to managing shopping lists and family chores.

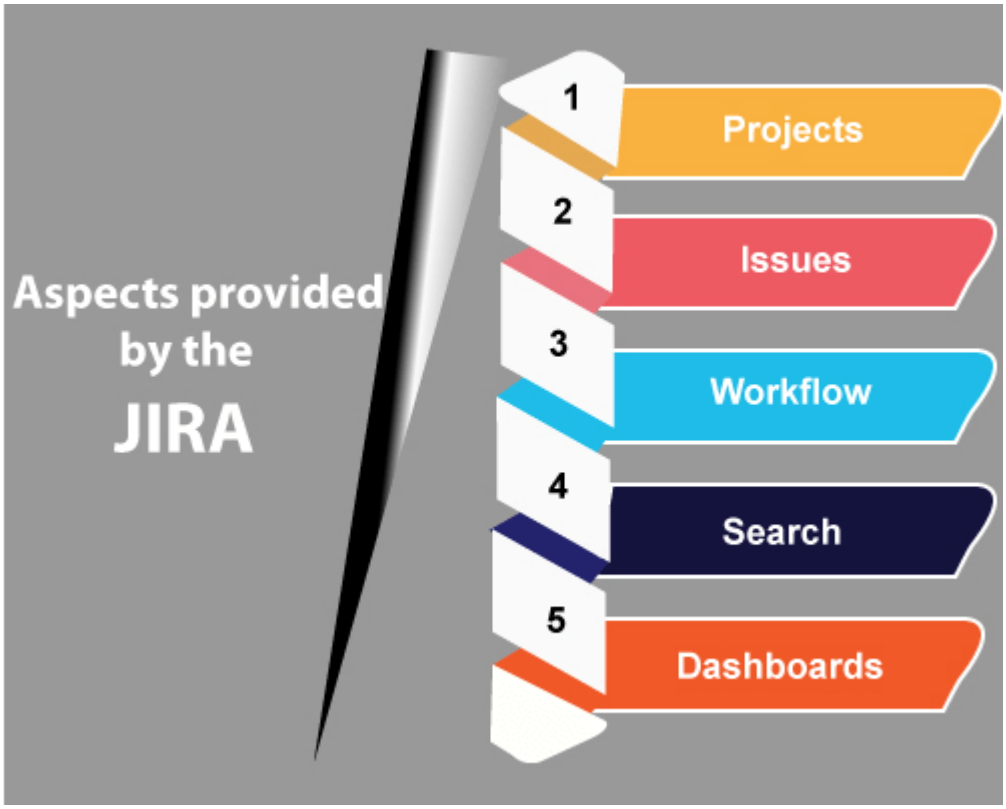
Four products are built on the Jira platform: Jira Software, Jira Service Desk, Jira Ops, and Jira Core. Each product comes with built-in templates for different use cases and integrates seamlessly, so teams across organizations can work better together.

JIRA Software Use cases

- Bug tracking
- Project management
- Product management
- Process management
- Task management

- Software development
- Agile software development

JIRA is based on the Agile methodology. The following are the useful aspects provided by the Jira:



Projects:
It is used to manage the defects very effectively.

Issue:
It is used to track and manage the defects/issues.

Workflow:
Processes the Issue/Defect life cycle. Suppose we have a business requirement, we create the technical design and from the technical design, we create the test cases. After creating the test cases, coding is done, and then testing is performed on the project. This design workflow is possible by using Jira.

Search:
Find with ease. Suppose we have done with a project at the beginning of the December and its version is 1.0. Now, we move to version 1.1 and completed at the end of December. What we are doing is that we are adding new versions. Through Jira, we can get to know that what happened in the earlier versions, how many defects occurred in the earlier projects and the learning we achieve from the earlier projects.

Dashboards:

Dashboard is a display which you see when you log in to the Jira. You can create multiple dashboards for multiple projects. You can create the personal dashboard and can add the gadgets in a dashboard so that you can keep track of the assignments and issues that you are working on.

Why JIRA

JIRA tool is used because of the following reasons:

Plan, Track and Work Faster

JIRA is a bug-tracking tool mainly used to track, organize, and prioritize the bugs, newly added features, improvements for certain software releases. Projects are subdivided into issues and issues can be of multiple types such as bug, new feature, improvement, and documentation tasks.

When the release date of software comes near, then software developers need to focus on the remaining issues which are to be fixed before the specified date. It also becomes difficult for the QA to maintain the status of the documentation, i.e., sometimes it becomes hard to keep track of everything.

JIRA is a good choice for handling the above issues. It enables software developers to track issues and improvements. It manages the projects as well as maintain the technical documentation.

The main source of information

JIRA is the primary source of information for the next software release. On JIRA, the whole team of the software developers can plan for the new features which are to be added and bugs to be fixed in the next release.
It also helps the QA team in writing the technical documentation. Through JIRA, the QA team can check the status of each feature that is newly added by the software developers, and according to that, they can plan how to document for the new version.

Organize the documentation tasks

JIRA tool is used to organize the documentation tasks. It is useful in grouping the multiple tasks by using the component functionality, and even you can create your own documentation. In this way, you can create a structured way of documentation.

Track the progress of our documentation

It is a very useful tool in tracking the progress of our documentation. JIRA tool provides a very important feature, i.e., pie chart macro. In the pie chart macro, you can view tasks such as Open tasks, Closed tasks, Resolved tasks.
Helps to meet the deadlines of a documentation release.

You can define the specific due date or deadline for the release of documentation, and even you can configure the JIRA tool with the notifications so that you can finish your documentation in time.

Measures the time spent on documentation

JIRA tool does not have the default functionality for measuring the time spent on documentation. JIRA tool is bundled with the Tempo Timesheets, which measures how much time has been spent on the documentation.

Provides feedback faster

JIRA tool provides the Confluence pages where you can connect to the issues in just a few clicks. If something needs to be updated, then you can create the issues directly from the Confluence page.

JIRA Waterfall Model

Before understanding the agile and JIRA, you must be aware of the waterfall model.

What is a Waterfall model?

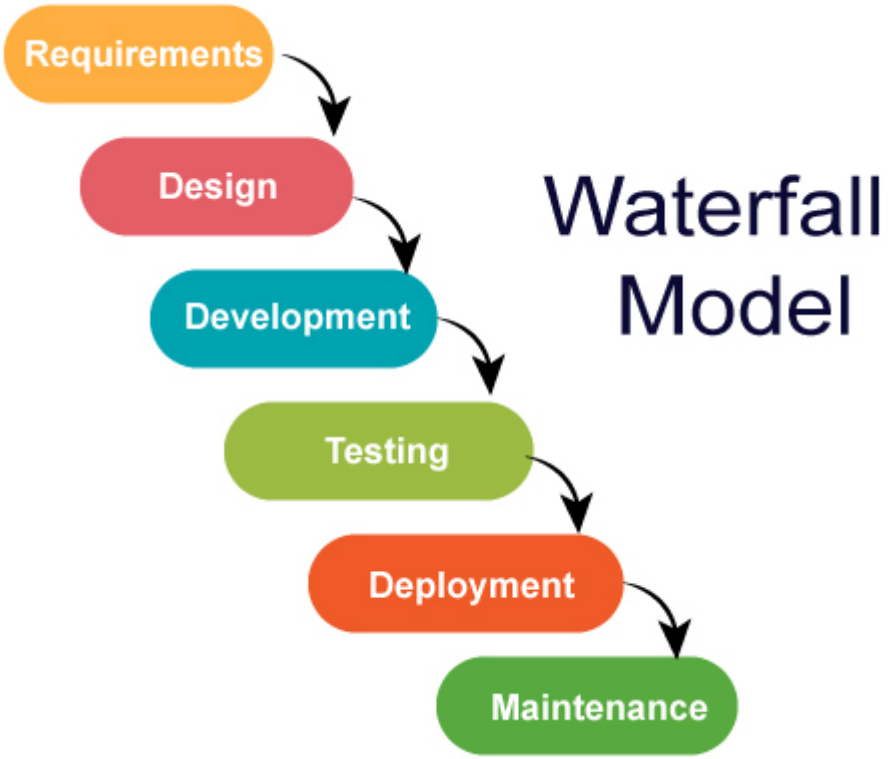
Waterfall model is the oldest model used by an IT industry to develop the software.

There are various models and processes which are used by different companies, but the waterfall model is the oldest, safest, and easy model in the IT industry.

It is the easiest model for building software that represents real life.

Let's understand the Waterfall model.

Waterfall model is broken down into multiple phases:



Requirements

Consider if there is a client and he wants to develop software, then the client reaches out to the company. Suppose he reaches out to the service-based company and asks the company to build the software for me.

What does the company do?

Company will collect all the requirements, the knowledge that the customer has or the client want to have on his software, the company will collect all the information from the client and prepare the documentation. Once this activity is performed, then the design phase gets started.

Design

In this phase, we prepare the high-level and low-level designs. Before developing the software, the design of the software is required. Suppose the customer wants e-commerce website similar to the Amazon, then UI(User Interface) of the website will be made by the designer and dataflows are also designed in this phase that how data will flow. Once this phase is completed, the development phase will get started.

Development

In the Development phase, the software development team starts coding and developing the software. This is the longest phase of the waterfall model as developers need more time to build the software. Once the development of the software is completed, then the project is handed over to the testers.

Testing

The testing team will test the software, and if any bug is found, then they inform the developers about the issue they found and make sure that the bug is fixed. They ensure that the end-to-end software is completed.

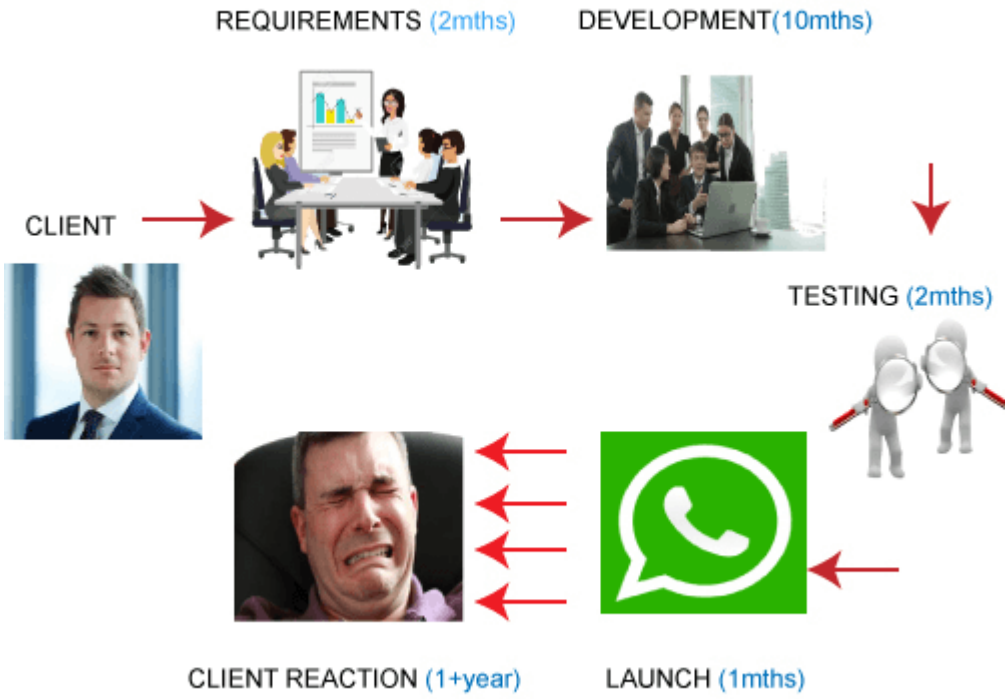
Deployment

Once the project is tested, the project is deployed so that it becomes live to the real-time users.

Maintenance

Finally, the project is deployed and available to the clients. Clients want the maintenance period for one or two years because if any bug is found or want a slightly enhanced feature in the project, so they need some team to handle such stuff. Due to this reason, they go for the maintenance period.

Example of Waterfall model



Suppose the client wants an app like a WhatsApp, so he reaches to the company where both the company and the client had a discussion for 2 months. The company made the documentation of all the requirements in 2 months. Now, the development team starts developing the software and suppose it took around 10 months to develop the software. It means that 12 months have been used, i.e., 2 months in requirement phase and 10 months in a development phase, but still the client does not have the idea about the internal phases. Once the development is completed, testing is done, and it will take around 2 months for software quality testing. Once the testing is done, it goes to the integration and launch so that WhatsApp will become live. However, when it reaches to the client, then the client says that it has taken more than a year and the software that I received was not what I expected. This happened because the client had only verbal communication with the

software team. If the client wants some changes in the software, then the whole process will be executed again.

Advantages of the Waterfall model

- **Simple and easy to understand and use**
  It represents all the tasks that you want to do in real life. For example, you need the requirements of a client. It contains different phases, and each phase is started only when the previous phases get completed.
- **Specific deliverable and review process**
  Each phase has a specific deliverable and review process. After the requirement phase, we have all the requirements of what the customer needs. Once the software is developed, we have its deliverable.
- **Phases do not overlap**
  In this model, phases do not overlap, i.e., they are completed once at a time. Once the previous phase is completed, then only the next phase gets started. For example, the Development phase will start only when the design phase is completed.

Disadvantages of the Waterfall model

- **Time to market is high**
  Product is released only when all the phases are completed. Therefore, this model takes a long time to release the product.
- **Unexpected results**
  What you expect and you receive are mostly different as the customer has an idea as per the documents only about the product. The client has only an idea, which is a documented idea.
- **Not suitable for changing requirements**
  This model is not suitable for the projects where requirements are at a moderate to high risk of changing. If the requirements are changing, then this model is not recommended as all the requirements are done at the requirement phase, which is a very time taking process.

Agile

- Agile is a time-boxed, iterative approach to build the projects incrementally instead of all at once.
- Agile is a practice that promotes continuous iterations of development and testing throughout the software.

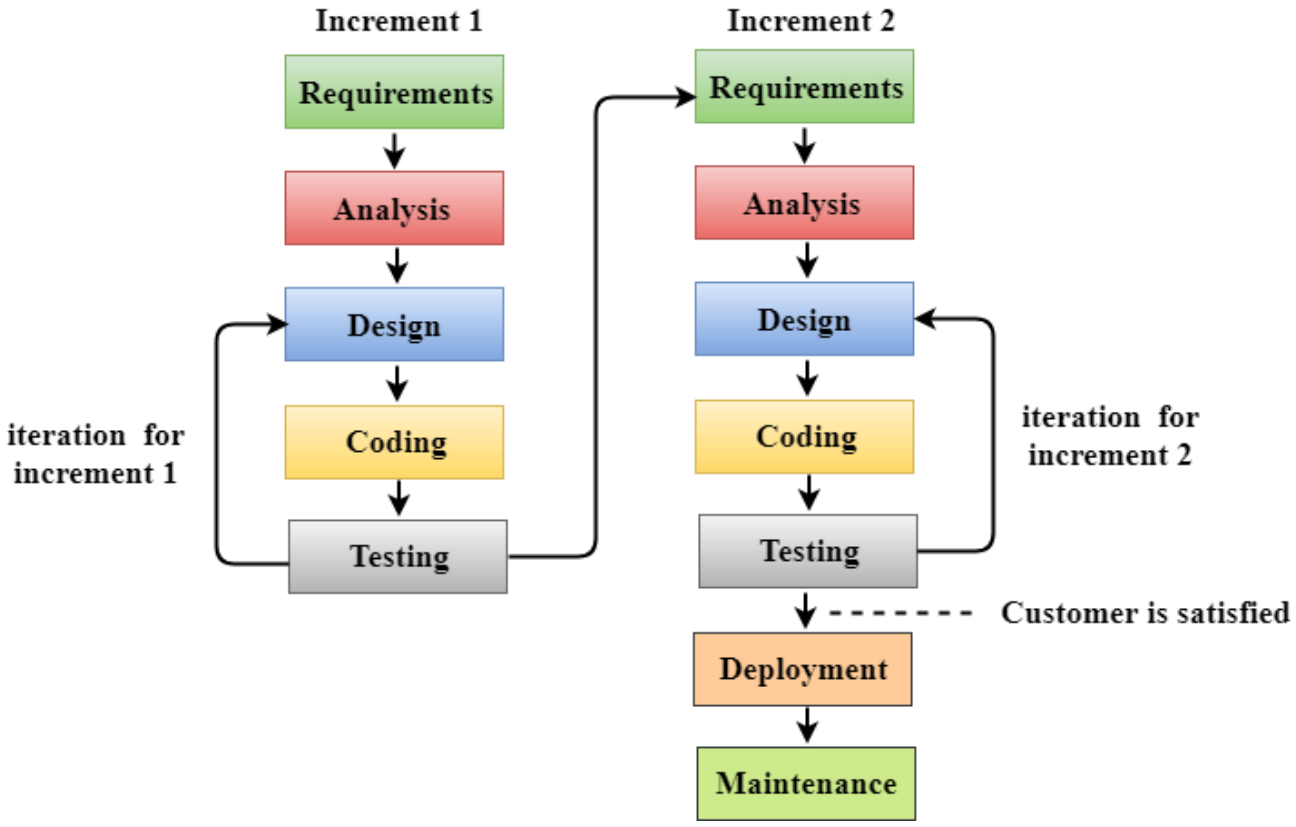
What is NOT AGILE?

- **Conducting meetings**
  The team conducts frequent meetings for 10-15 minutes daily, and they think that conducting frequent meetings will be Agile. However, only the following meetings will not be Agile.
- **Requirements changing anytime**
  Requirements can be changed at any time, then it is not Agile. For example, a client wants to add some new features and want the changes to be updated at the same time, then this will not be Agile.

- **Unstructured development**
  Suppose you are not following any plan and you are working on Adhoc basis then it is not Agile wherein Adhoc testing, testers randomly test the application without following any documents and test design techniques.
- **No documentation**
  If the company does not make the documentation, then it is not Agile.

What is Agile?

- Agile is a philosophy, i.e., a set of values and principles to make a decision for developing software.
- Agile is based on the iterative-incremental model. In an incremental model, we create the system in increments, where each increment is developed and tested individually.



The above diagram shows how agile model works incrementally.

What are values?

Individuals and interactions	**over processes and tools.**
Working software	**over comprehensive documentation.**
Customer collaboration	**over contract negotiation**

Responding to change	**over following a plan**

In Agile, you need to perform all the eight tasks which are mentioned in the above table. However, we have to make sure that the left task should be given higher priority than the right tasks.

- **Individuals and interactions, Over processes and tools**
  Suppose the team finds any issue in software then they search for another process or tool to resolve the issue. But, in Agile, it is preferable to interact with client, manager or team regarding issue and make sure that the issue gets resolved.
- **Working software, Over comprehensive documentation**
  Documentation is needed, but working software is much needed. Agile is not saying that documentation is not needed, but working software is much needed. For example, you have 20-page documents, but you do not have a single prototype of the software. In such a case, the client will not be happy because, in the end, the client needs a document.
- **Customer collaboration, Over contract negotiation**
  Contract negotiation is important as they make the budget of software, but customer collaboration is more important than over contract negotiation. For example, if you stuck with the requirements or process, then do not go for a contract which we have negotiated. You need to interact with the customer, gather their requirements.
- **Responding to change, over following a plan**
  In the waterfall model, everything is planned, i.e., at what time, each phase will be completed. Sometimes you need to implement the new requirements in the middle of the software, so you need to be versatile to make changes in the software.

Agile Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. According to the Agile principle, the customer is everything for them. Whatever the customer needs, they have any issues or want to add new requirements, they always give priority to the customers. Listen to the customer what they are saying and provide the quality software to them.
- It welcomes changing requirements, even late in development. Agile processes harness change for the customer?s competitive advantage. In the waterfall model, if any new changes to be made in the middle of the software, the whole process is to be done again. Therefore, the waterfall model is rigid and not versatile. Agile says work like so that new changes can be easily incorporated in software.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. As in a waterfall model, when the whole system is developed then only it is delivered to the customer while Agile says that do not wait for too long, wait only for some weeks or months. Whatever you have developed, give a demo to the client, and this gives the glimpse of the software to the client that what you are developing at the initial phases.
- Business people and developers must work together daily throughout the project. This means that the customer, client, and team should interact daily.
- Build projects around motivated individuals, give them the environment and support they need, and trust them to get the job done. Agile says that believe on your team, customer,

company. Suppose a task is given to the team member then provide all the resources whatever he needs such as documentation, system, information study, etc.

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. Suppose there are situations when you need to interact with the client, development team which is usually done through mails or phone calls, but it is better to have face-to-face conversation.
- Working software is the primary measure of progress. Agile says that whatever has been developed is neither through documentation or what your project manager is saying, how much software has been developed or working is the measure of the progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. Agile says that make your team constant in delivery so that the team should have fixed working hours means if the company working timing is 8 hours then the team should work for 8 hours in a day.
- Continuous attention to technical excellence and good design enhances agility means that the team members should be technically sound so that they can make good designs if any changes are made, then they can be easily incorporated in the software.
- The best architectures, requirements, and designs emerge from the self-organizing teams. Whatever the architecture team made designs, they make sure that they sit with the development team and discuss the architecture of the software.
- At regular intervals, the team reflects on how to become more effective then tunes and adjusts its behavior accordingly. This principle says that the teams should frequently meet so that they can discuss the issues that they are facing and can be resolved effectively.

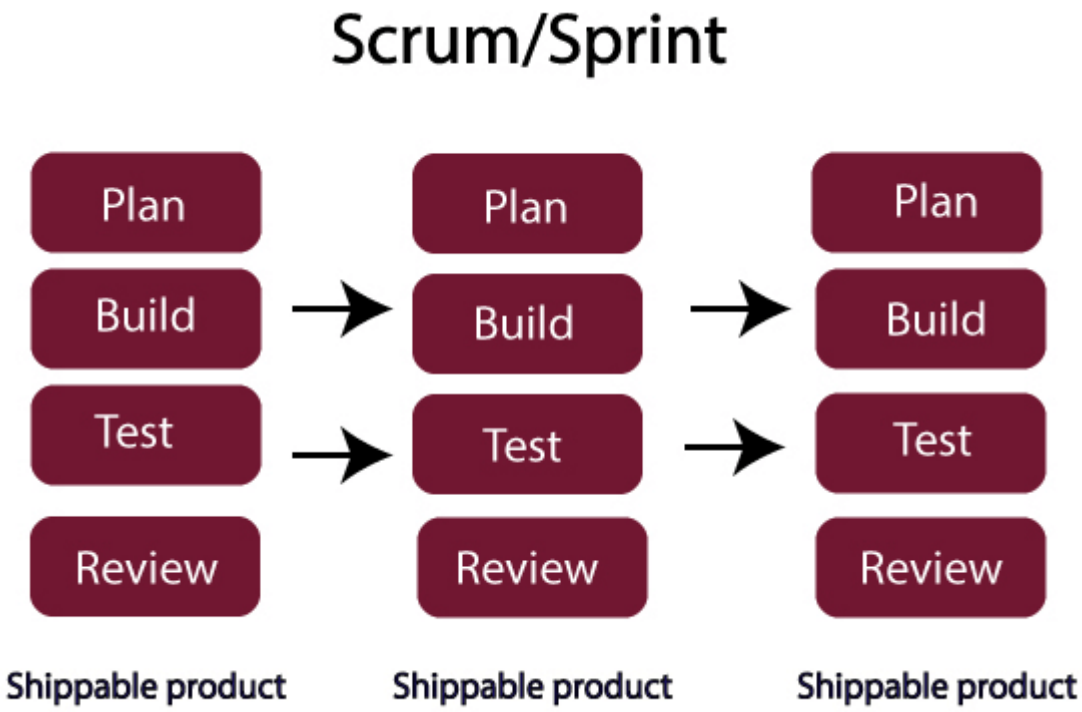
Scrum

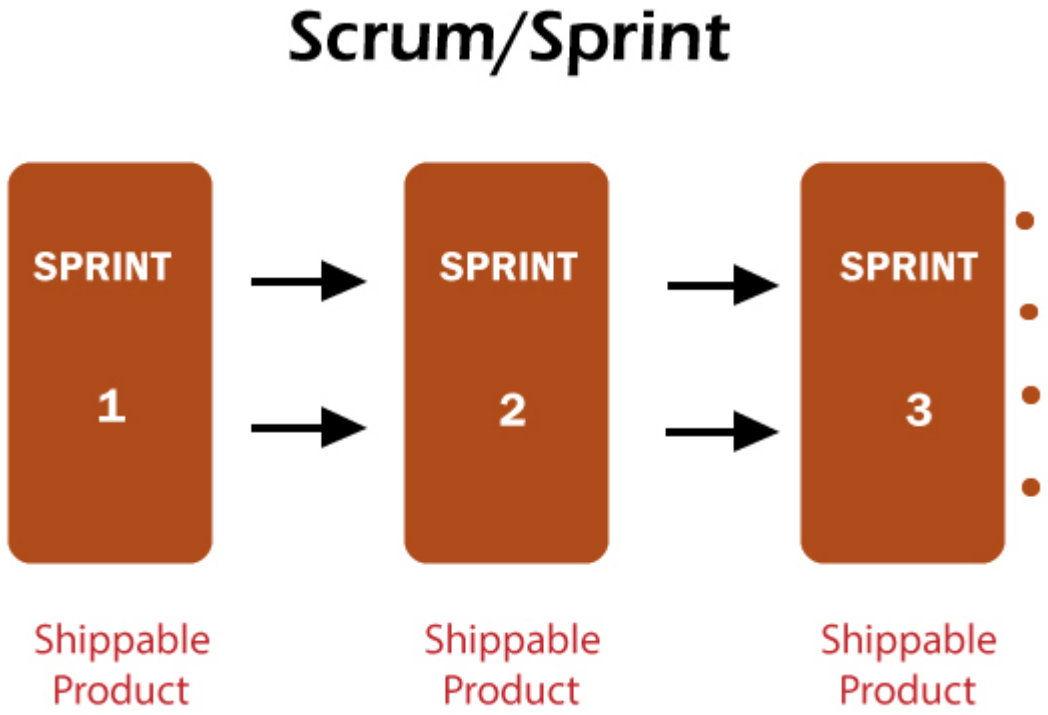
We have studied the Agile methodology where Agile is a set of beliefs which should be followed to develop the software development project. On these beliefs or values, there is many models have developed, and in which one of the models is a **scrum**.

Before going to deep in Scrum, you should know the meanings of some basic terms:

- **Scrum:** Scrum is an agile framework that helps you to organize, iterate, and continue the same project that you are working on. In scrum, a product is built in the series of iterations known as sprints or parts.
- **Sprint:** Sprint is a time-boxed period in which the scrum team needs to finish the set amount of work. Each sprint has a specified timeline, i.e., 2 weeks to 1 month. The scrum team agrees with this timeline during the sprint planning meeting.
- **Scrum Master:** Scrum Master is defined as a facilitator or servant-leader to the Scrum development team. Scrum Master must ensure that scrum principles are followed.
- **Scrum development team:** A scrum development team is a collection of individual members that includes developers, QA, and scrum master. It decides and provides the effort estimate. The recommended size of the scrum team is between 5 and 9 members.

How does Scrum work



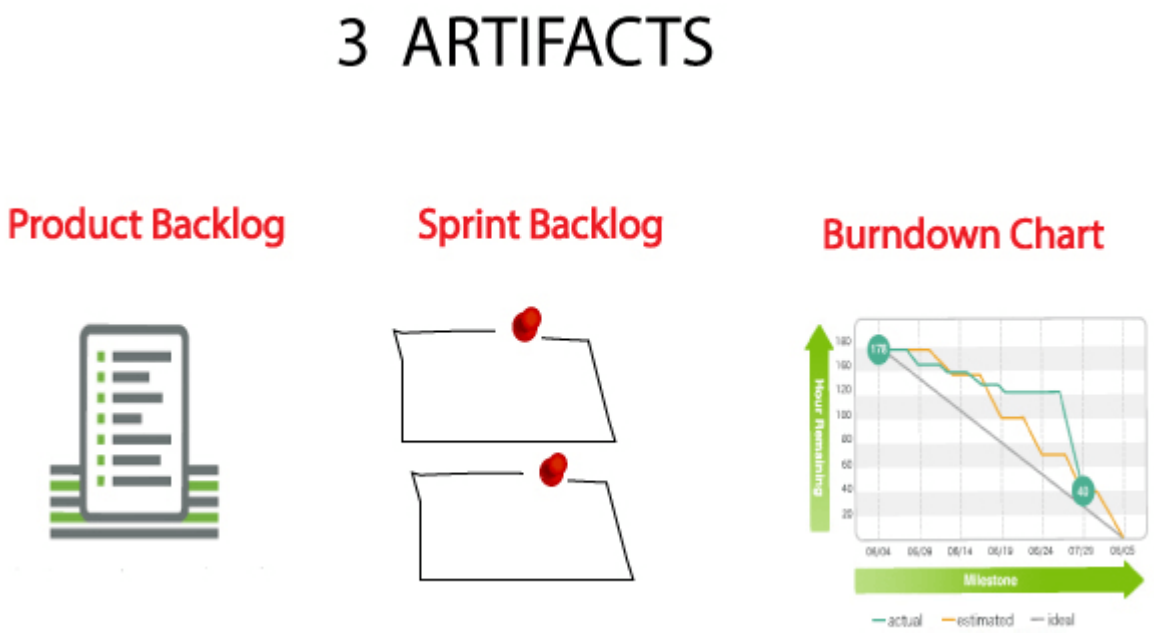


In the Waterfall model, we have read that the first whole requirements are done, then the whole designing is done, then whole development is done, and then whole testing and deployment is done. This model takes full lifecycle of the product, and then the only product is viewable to the client. While Scrum says that consider a small part of the software and then plan it, build it, test and finally review it. This small module which has been developed will be shown to the customers. For example, we need to develop the e-commerce website, which can be broken into a number of sprints or modules such as login page, payment page, cart page, etc. Then, each module is developed individually and shown to the customers simultaneously. Therefore, we can say that after the completion of each sprint, the product is shipped to the client, though not the complete product but the part of the functionality.

Artifacts of Scrum

The documentation and stuff which are prepared in scrum are known as Artifacts.

Following are the artifacts of Scrum:



- **Product Backlog**
  Product Backlog is a collection of activities that need to be done within the project. When we want to develop software, then we need to perform the 'n' number of activities. For example, we need to develop the e-commerce website then we have to do the 'n' number of activities such we need to create the login page, payment system, cart system, etc. and these 'n' number of activities which are needed to develop the software is known as the product backlog.
- **Sprint Backlog**
  We know that in a scrum, we break the scrum into 'n' number of sprints and the objective of a sprint is to bring the small functionality of the software and ship it to the client for demo. In Product backlog, we have to do all the activities which are required to develop the software while in the sprint backlog, a small set of product backlog activities are performed within that sprint. The 'n' number of sprint backlogs is equal to the 1 product backlog.
- **Burndown chart**
  Burndown chart is the outcome of the sprint, which shows the progress in a sprint. After each sprint, we need to examine the progress of each sprint. The burndown chart tells how you are working on the sprint. In the burndown chart, the graph starts from some time, i.e., where the activity gets started, and at the end of the sprint, the graph reaches to zero where the activity ends. It is generally an inclined line from top to bottom.

Scrum Roles

There are three scrum roles:

- **Product Owner**
  There is a client who wants to develop his software, so he approaches to the company who can develop his software. What does the company do? The Company assigns a role, i.e., Product Owner. Product Owner is the person who communicates with the clients understands their requirements. Product Owner is the responsible person from the company for software development.
- **Scrum Master**
  During the sprint, Agile says that the team should meet together once daily. When the team is following scrum means that they are conducting meetings daily for 10 to 15 minutes. This meeting is known as a scrum meeting. Scrum Master is the person who handles the scrum meeting.
- **Team**
  The team comprises of persons who work on the project. It can be developers, testers or designers. When we talk about Agile or Scrum then we talk about the team, we do not talk about developers, or testers as an individual. Agile says that developers can work as a tester or testers can work as a developer when the need arises.

Scrum Ceremonies

Let's look at the following Scrum Ceremonies:

- **Sprint Planning**
  Scrum consists of a number of sprints which have a different set of modules used to deliver the software. Before starting the sprint planning, we have a meeting known as sprint planning, and in sprint planning, we discuss what we are going to do in a sprint. In sprint planning, product owner discusses about each feature of a product and estimates the effort involved by the development team.
- **Daily Scrum**
  In Scrum, meetings are conducted daily for 15 minutes by Scrum Master, where Scrum Master is the person who manages the meeting. Meeting consists of scrum master, developers, testers, designers, product owner, the client where product owner and client are optional.
- **Sprint Review**
  After the completion of each sprint, the meeting is conducted with a client in which a product is shown to the client for demo and team discuss the features they added in the project.

Kanban Methodology

Kanban is the most popular agile framework after Scrum for software development. It provides the real-time and transparency of work. In Kanban board, all the tasks are visible that allows the team members to see the state of every task at any time.

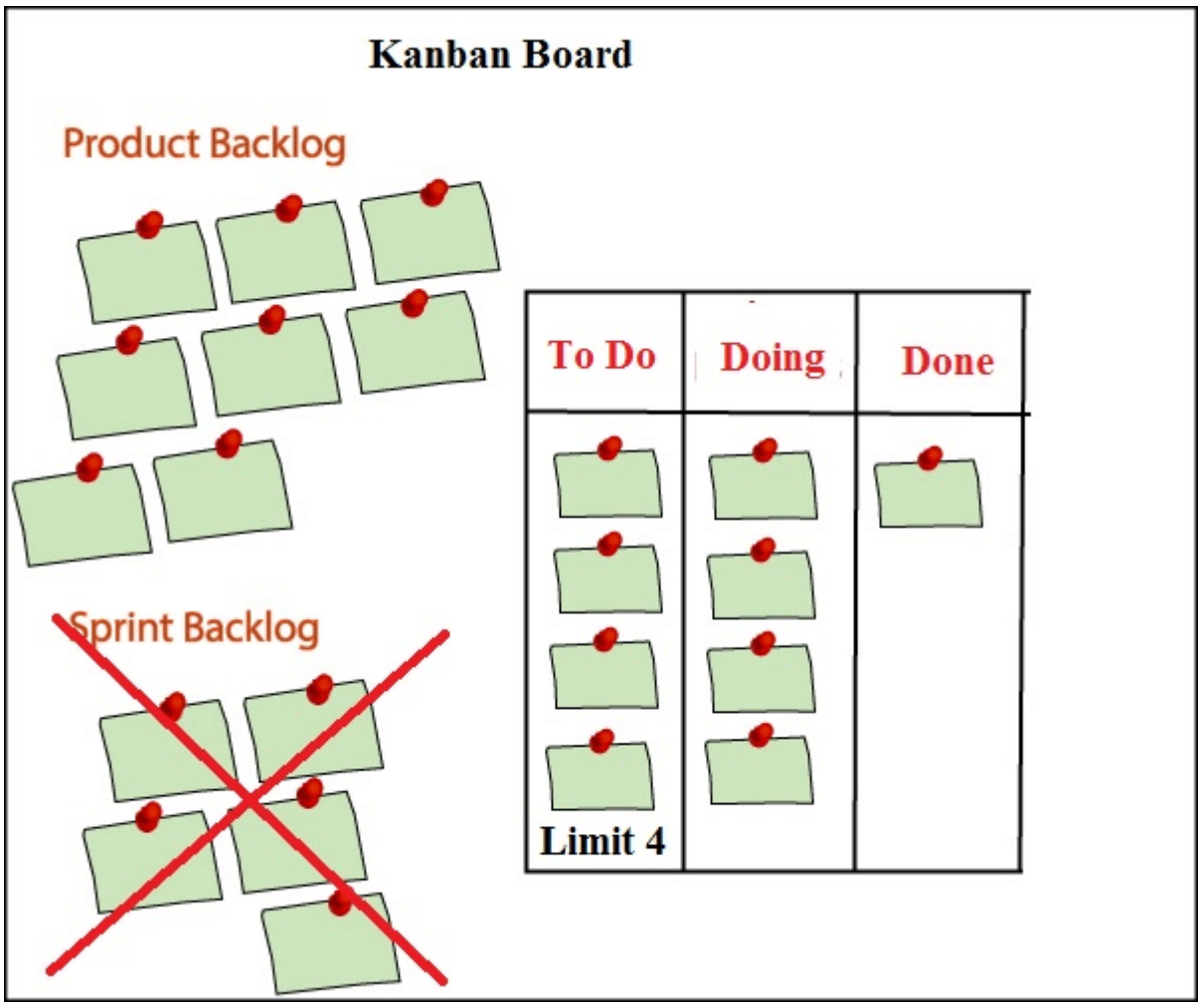
Characteristics of Kanban methodology:

- **Flexibility**
  In Kanban methodology, a team focusses on the work that is '**in progress**' state. Once the team completed its task, then it pulls the next first task of the product backlog. A product owner reprioritizes the tasks or made changes in a product backlog outside the team, so this

will not disrupt or impact the team. A product owner keeps the most important task on the top of the product backlog, so the development team assures that they will produce the most valuable output. In Kanban, we do not need to do the fixed-length iteration as we did in the scrum.

- **Minimize time cycles**
  The cycle time is the amount of time taken by the work to travel from the moment it gets started to the moment it gets shipped to the customers. An overlapping skill set can minimize the cycle time. In this, developers not only write the code but can also test the code whenever required. This type of sharing skills means the team members can take the heterogeneous work which optimizes the cycle time.
- **Visual metrics**
  Visual metrics is a way of improving team efficiency and team effectiveness. A visual metric is shown through charts, and team members can view the data in charts, and can spot the issues arises in their process. The main goal of the visual metric is to reduce the amount of time taken by the issues to move through the entire process.
  There are two types of charts used by the kanban team:
  Control charts: It shows the cycle time taken by each issue.
  Cumulative flow diagrams: It shows the number of issues present in each state.
- **Continuous delivery**
  The main aim of continuous delivery to deliver the product with low risk rapidly. The transition from the agile methods to continuous delivery move the two-three weeks sprint to the Kanban methodology. Both Kanban methodology and continuous delivery complement each other by delivering the product to the customers faster. Software development teams are used to develop, test, and review the new features in a continuous manner. Therefore, we can say that Kanban is a continuous flow methodology.

Kanban Board



Kanban board is a tool used to visualize the work and limit work-in-progress.

As in scrum, we are taking some activities from a product backlog and adding in a sprint backlog. However, in Kanban, we do not have sprint, so sprint backlog activity will not be performed. This is the main difference between scrum and Kanban that scrum contains sprint backlog while kanban does not contain the sprint backlog.

Kanban board consists of three states:

- To Do
- Doing
- Done

When the project is started, then we put all the activities from the product backlog to the '**To Do**' state. When the team member starts working on an activity, then that activity is put in a '**Doing**' state, and when the activity is placed, then it is placed in a '**Done**' state.

From the Kanban board, one can get to know which activities have been done and which activities they need to develop.

One of the most important features of the Kanban board is a **Limit** option. In the above figure, we have eight tasks in a product backlog and limit set is 4. At a time, it will take only four tasks in a '**To Do**' state, and if any of the tasks come in a '**Doing**' state, then one more task from the product

backlog will be placed in a '**To Do**' state. In this way, we can set the **limit** depending on the availability of the resources.

Kanban vs Scrum

There are many differences between Kanban and Scrum. A list of differences between Kanban and Scrum are given below:

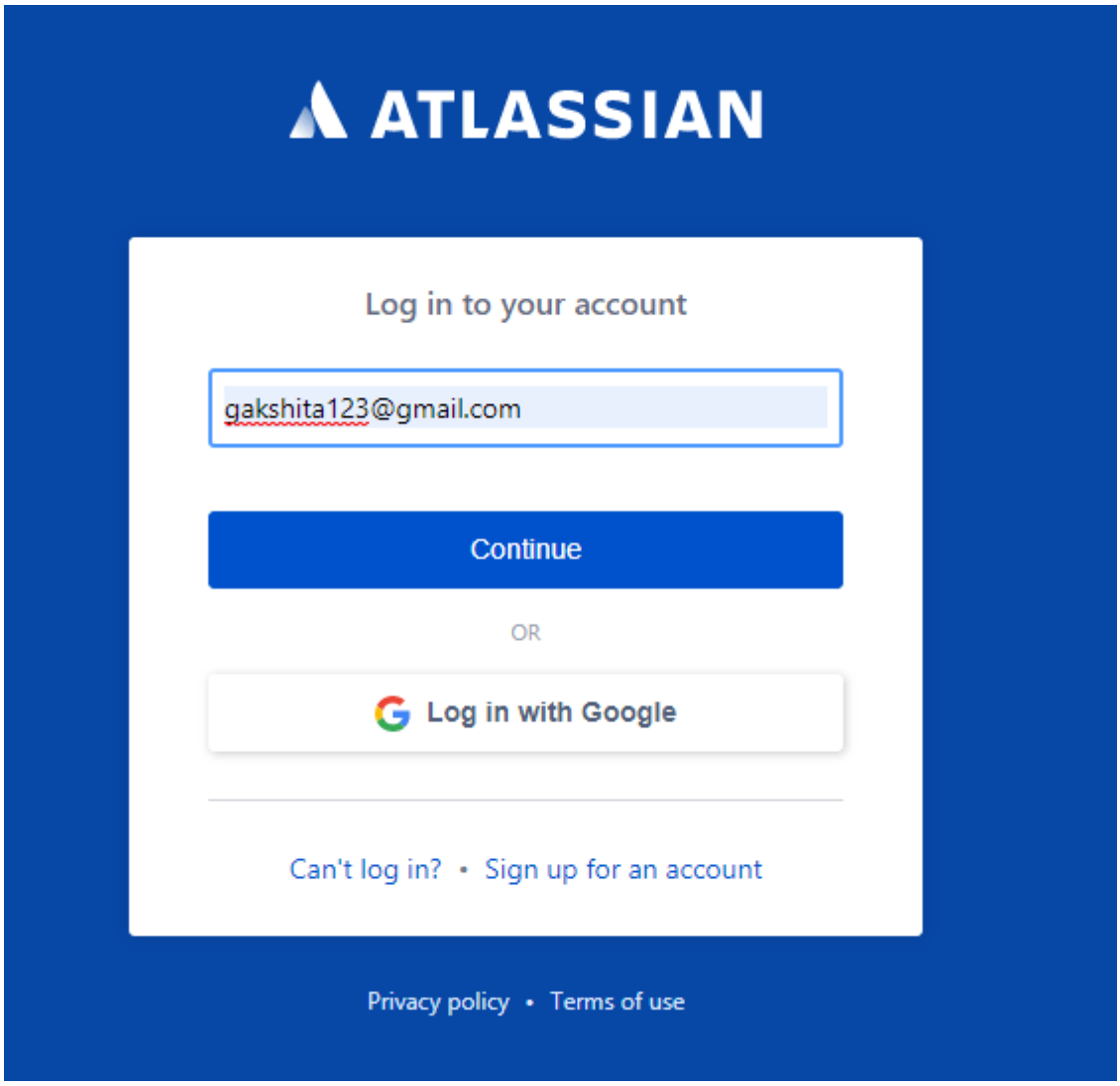
	Scrum	Kanban
Planning	It has fixed planning. It focussed on planning. It starts with the sprint planning and ends with the sprint review, retrospective. The daily meeting is held so that the team knows the next steps, priorities, and the learnings from the previous steps.	It has no fixed planning, and no daily meetings are conducted. In Kanban, changes can occur at any time, i.e., frequent changes occur.
Timeline	In scrum, we work on the sprint that has the fixed-time duration means that after some fixed-time, we are delivering something to the client.	Kanban does not have the concept of a sprint, so it has no fixed timeline for delivering the product to the client.
Task estimates	During sprint planning, it is decided that how many activities are to be pulled from the product backlog and add in a sprint backlog. For example, if the sprint is for two weeks, then the number of activities are selected in such a way that they can be completed within the sprint, i.e., in two weeks.	It does not estimate the task.
Scrum Master	In scrum methodology, we have one scrum master who handles the team and conducts the meeting on a daily basis.	In Kanban methodology, we do not have any scrum master. It's the responsibility of each individual to deliver a valuable product.
Suitability	This methodology is suitable for large-sized projects as large-sized projects can be divided into multiple	It is mainly suitable for small-sized projects.

	sprints.	
Constant changes	In Scrum, constant changes can be adapted easily in shorter sprints.	If any major change occurs, then Kanban methodology gets failed.
Cost	In Scrum, the task is estimated, i.e., a fixed number of activities are taken in a sprint, so the total cost of the project is minimal.	In Kanban, the task is not estimated, so the total cost of the project is not accurate.
Roles and responsibilities	In Scrum, a specific role is assigned to the team members by the Scrum Master while the product owner tells the objectives of the product on which team members have to work.	No predefined role is assigned to the team members. It's the responsibility of all the team members to work in collaboration to deliver a valuable product.
Measurement of Productivity	The productivity is measured by using cycle time or the time taken to complete the whole project from start to the end.	Productivity is measured by using velocity through sprints.
Release Methodology	Small release after the end of each sprint.	It provides continuous delivery.

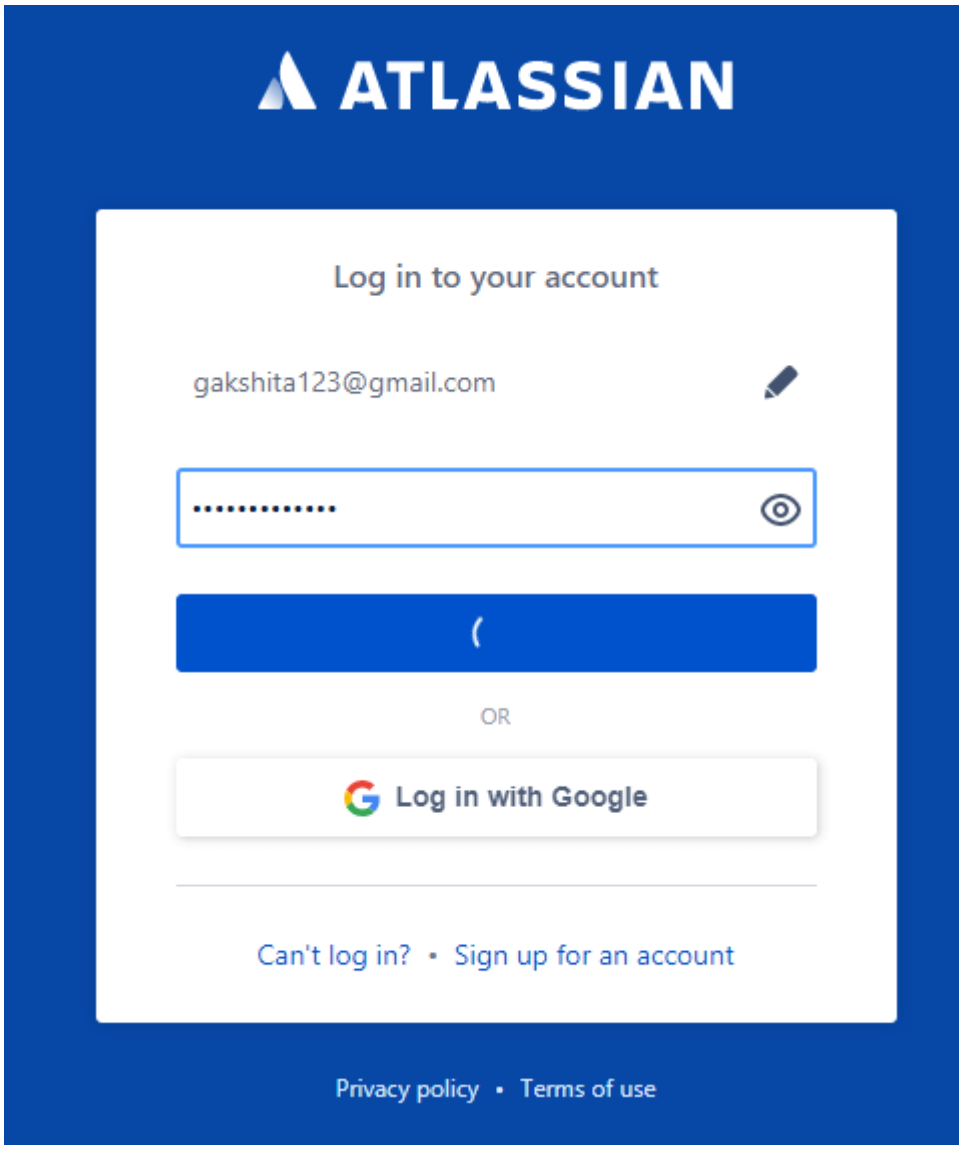
JIRA Login

The following are the steps to login to Jira:

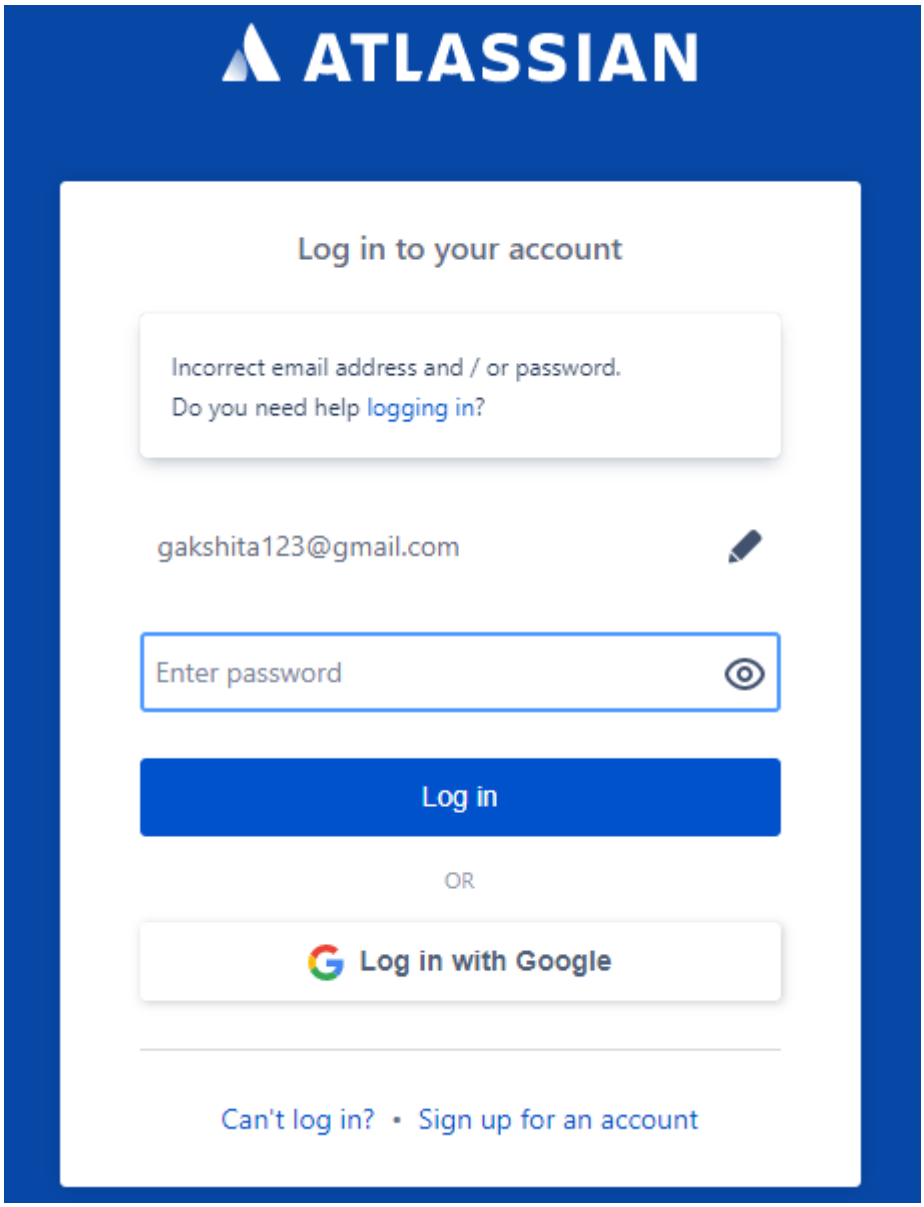
- To Login to Jira, move to the website that we created in **Jira**. The login page appears and then you need to fill the credentials such as username and password.
  Click on the continue button.



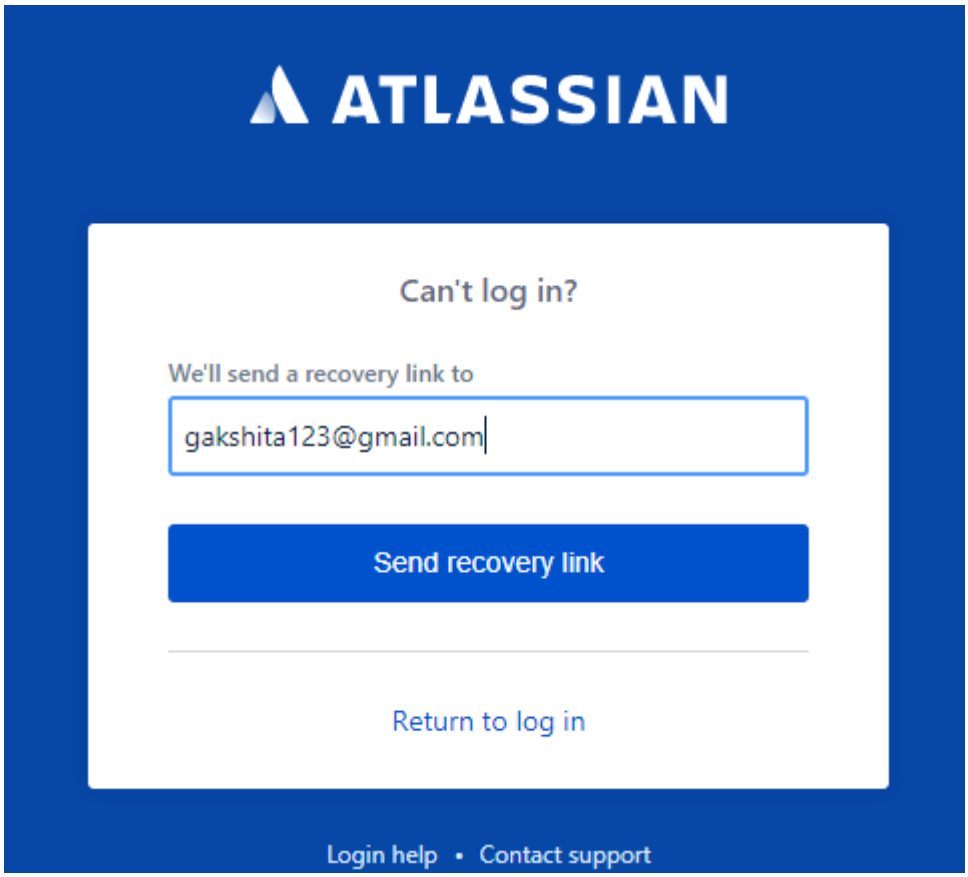
Enter the password, and then click on the **Login** button.



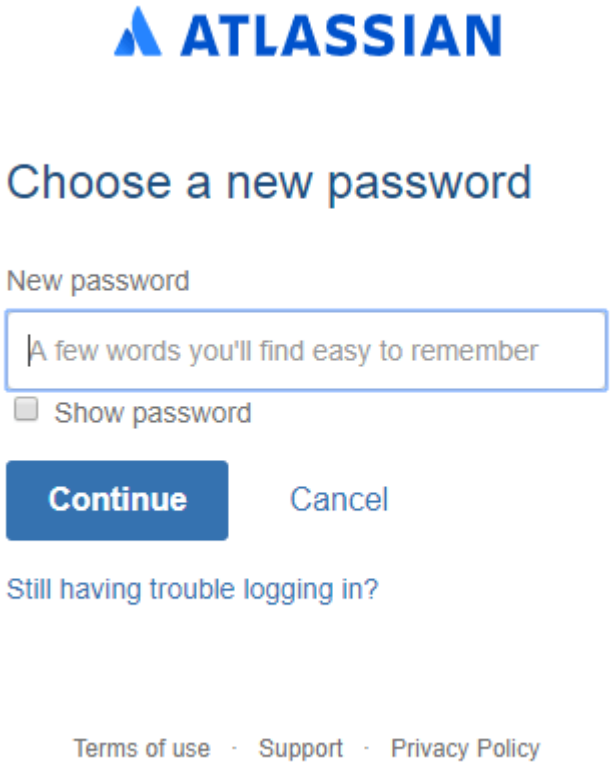
If the password is incorrect, then it displays the validation message "incorrect email address or password". This is shown in the below screenshot:



To overcome the above error, click on the **Can't log in?** link. On clicking on the link, we will get the screen which is shown below:

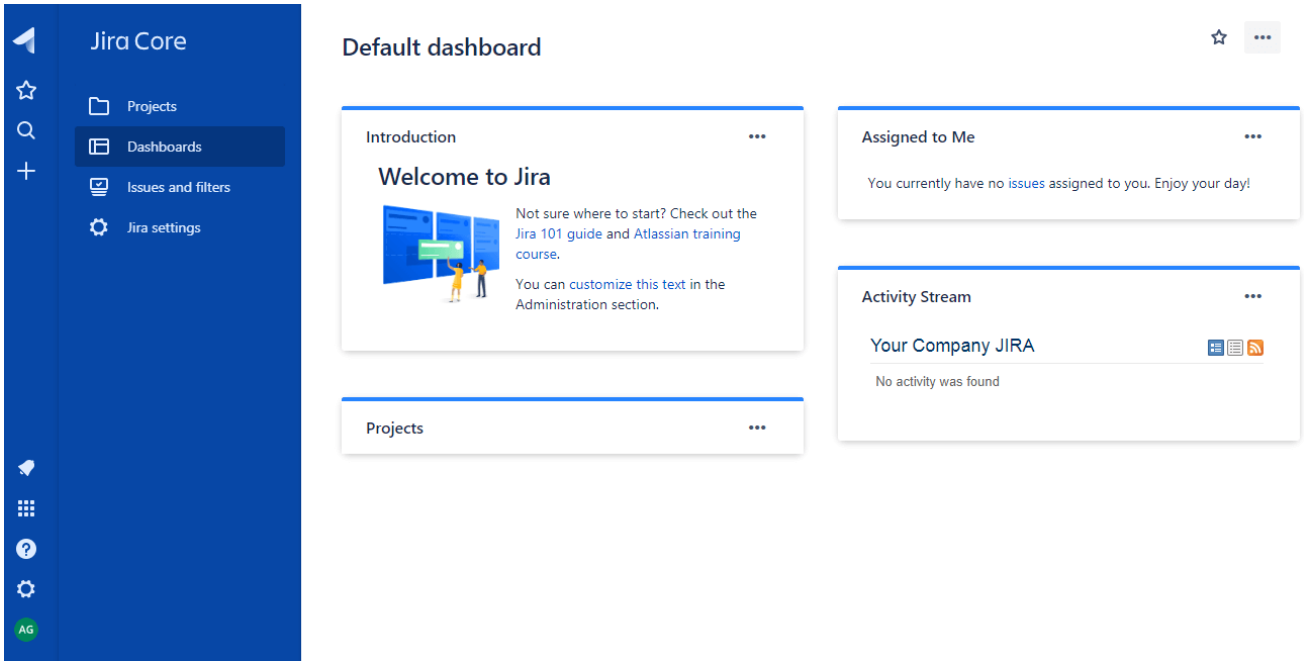


- Click on the **Send recovery link** button. After clicking on the **Send recovery link** button, the mail is sent to the registered email id and ask you to **Reset my password**. When you click on the **Reset my password** button, we will get the screen which is shown below:



- Enter your new password, and then click on the **Continue** button.

- The above cases occur when we provide either incorrect username or password. On successful login, your jira account will get opened and the screen appears which is shown below:

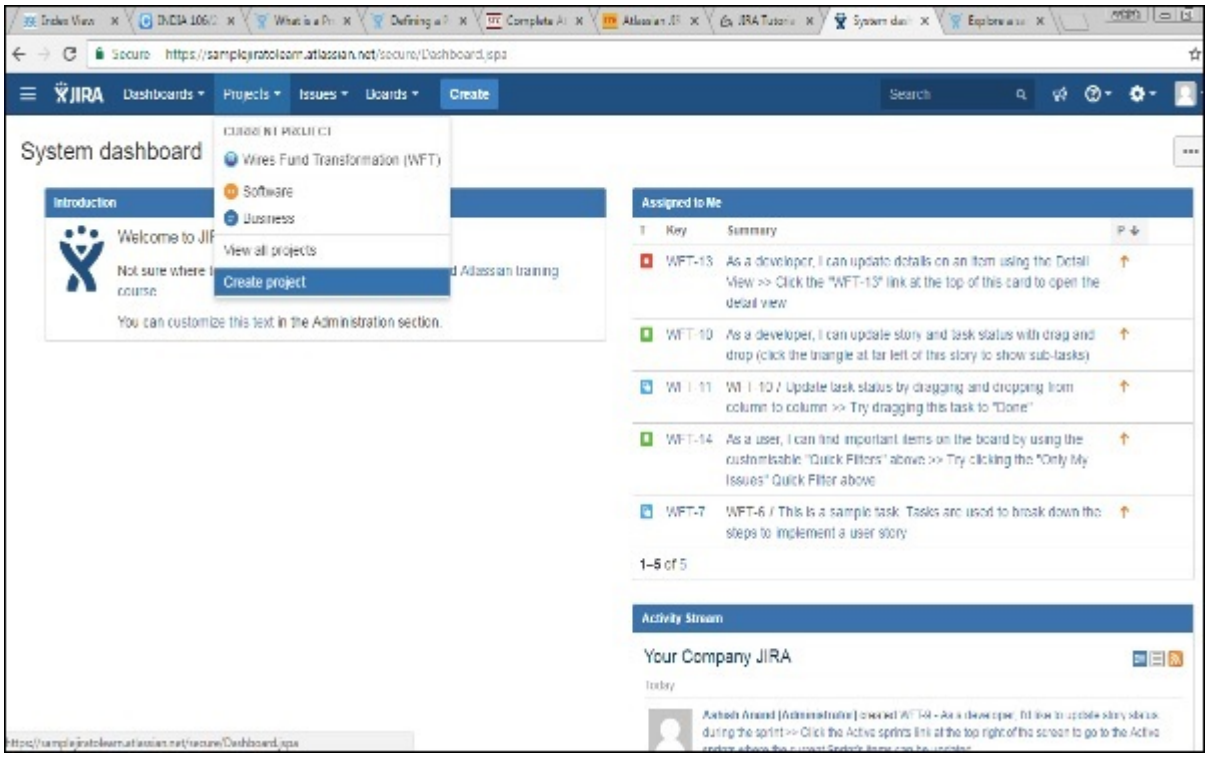


JIRA Project

A Project contains issues; a JIRA project can be called as a collection of issues. A JIRA Project can be of several types. For example –

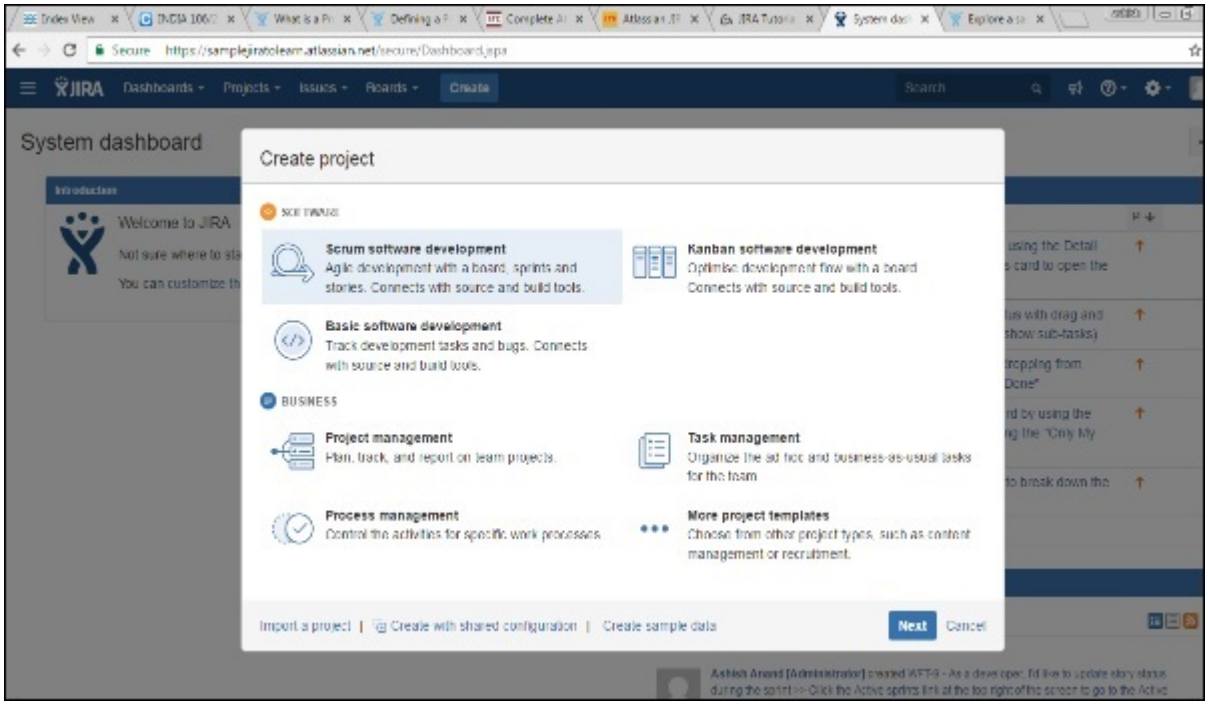
Software Development Project
Marketing Project
Migration to other platform project
Help Desk Tracking Project
Leave Request Management System
Employee Performance System
Website Enhancement
Create a New Project
To create a project, the user should login as a JIRA Service Desk Admin and then Click on Project → Create Project.

The following screenshot shows how to reach to the Create Project button from the Dashboard.



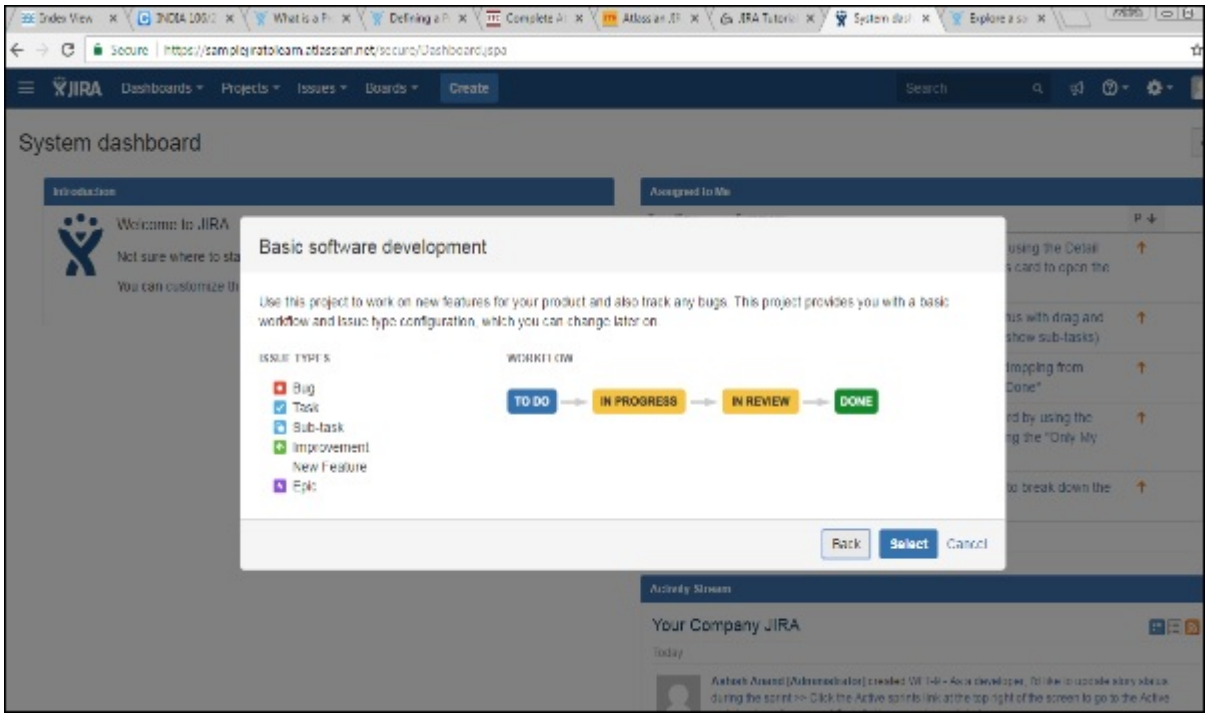
Choose the type of Project that suits your requirement and the process it should follow.

The following screenshot displays the type of projects available in JIRA.



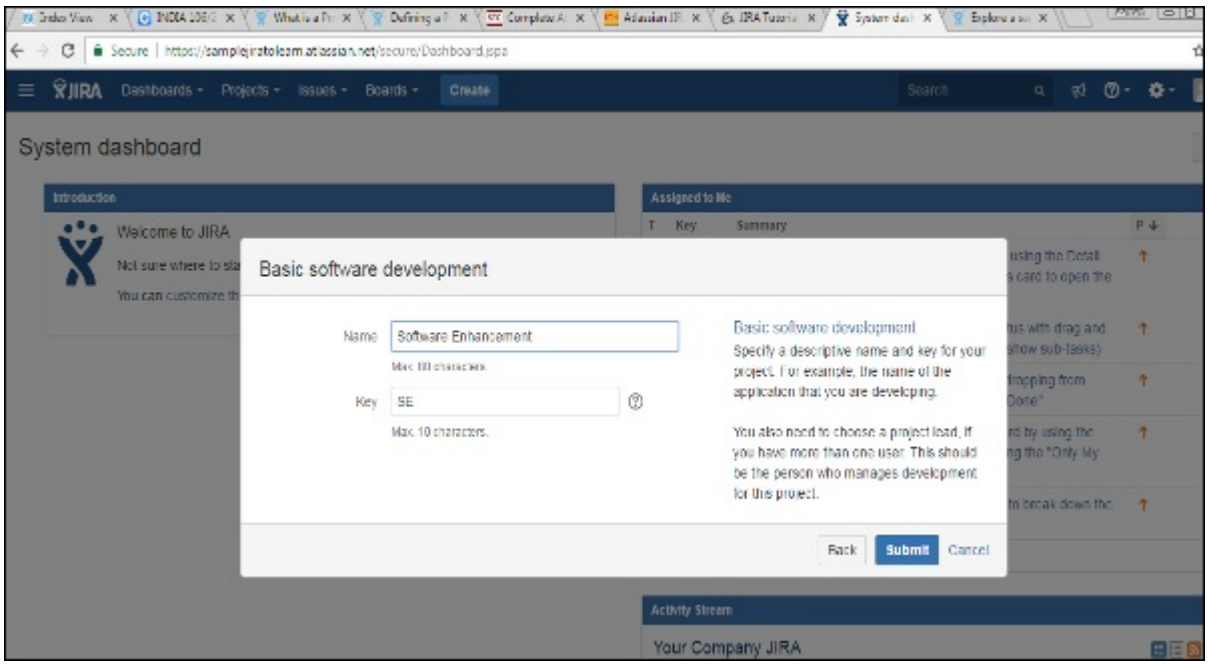
Once the type of project is selected, click on Next. The user will see the flow of the project based on the selection. Here, we have selected Basic Software development.

The following screenshot displays the available issue types and the workflow for the chosen project in the step mentioned above –

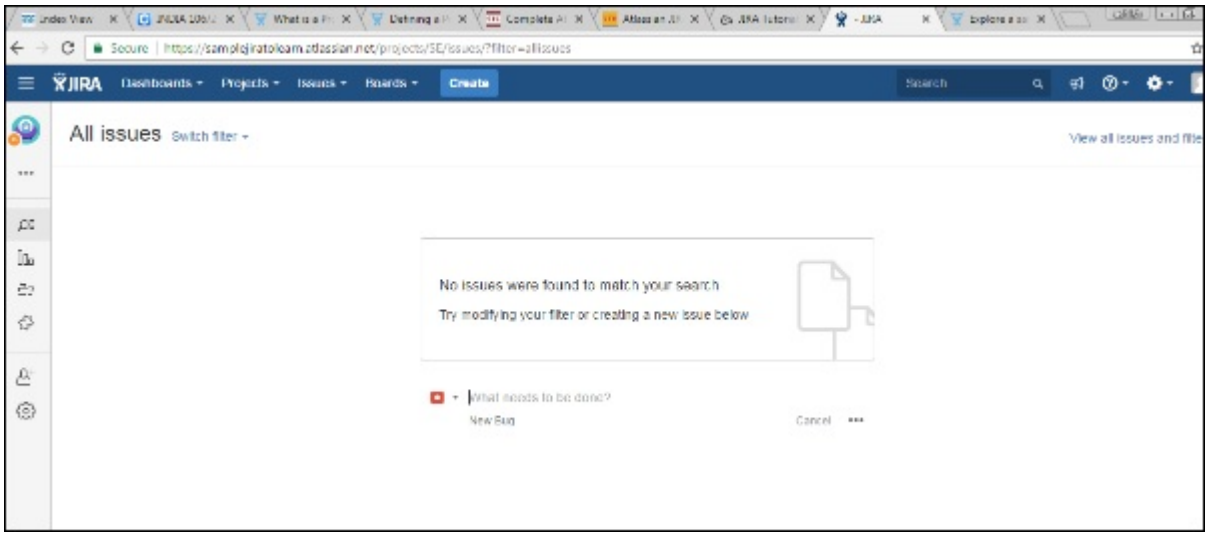


Click on the Select button, enter the name of the project and confirm the Key that the user wants to display as a reference in all the issues. Once this is done, click on the Submit button.

The following screenshot displays the fields to provide details before the creation of a project.



The page having issues will display. The following screenshot displays whether any issues are linked with a new created project.



What is an Issue in JIRA?

JIRA Issue is used to track the individual pieces of work that has been completed or not. JIRA issue can be a software bug or any other issue. Once you have imported the project in Jira, you can create issues.

Issue types

Teams work together to break down the pieces of work into issues. Issues can represent software bugs, a project task, subtasks, or any other project work. Jira project comes with the default issue type, and you can also create your own issue type.

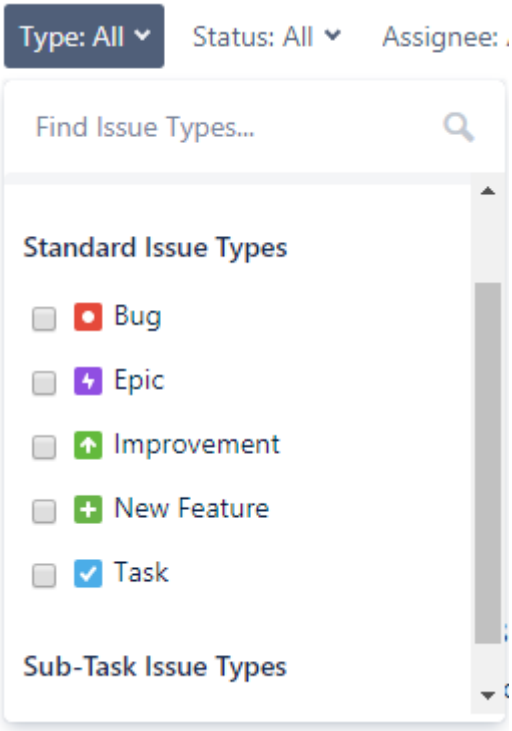
An issue type is a generic name for the unit of work. In Jira, we have different units of work in a project, and issue types field differentiate this.

When you create an issue, then you need to specify the issue type. You can even change the issue type after creating the issue.

Each issue has an icon associated with it to identify the issue type.

Custom Issue type: It is also used to create the customization issue type that provides the flexibility to the team to create their own issue types according to their projects

The below screen shows the list of Issue types:



There are three types of Issue types:

- **JIRA Core Default Issue types**
  - **Task:** The task is a work item done by the team but not connected directly to the user's requirements. For example, to upgrade the version of a product used by the teams.
  - **Subtask:** It is a part of another issue. It is used to break an issue into different pieces of work. While creating an issue, sub task issue is not given in the issue type field drop-down as it contains some parent issue, so we can say that subtask issue cannot be created independently.
- **JIRA Software default issue types**
  - **Story:** It is a requirement from the user's perspective.
  - **Bug:** It is a flaw in a product that needs to be fixed by the developers. It can be tracked with its own issue type to differentiate from other types of work.
  - **Epic:** An epic is a big issue that contains other issues.
- **JIRA Service desk default issue types**
  - Incident
  - Service Request
  - Change
  - Problem

Why Issue types?

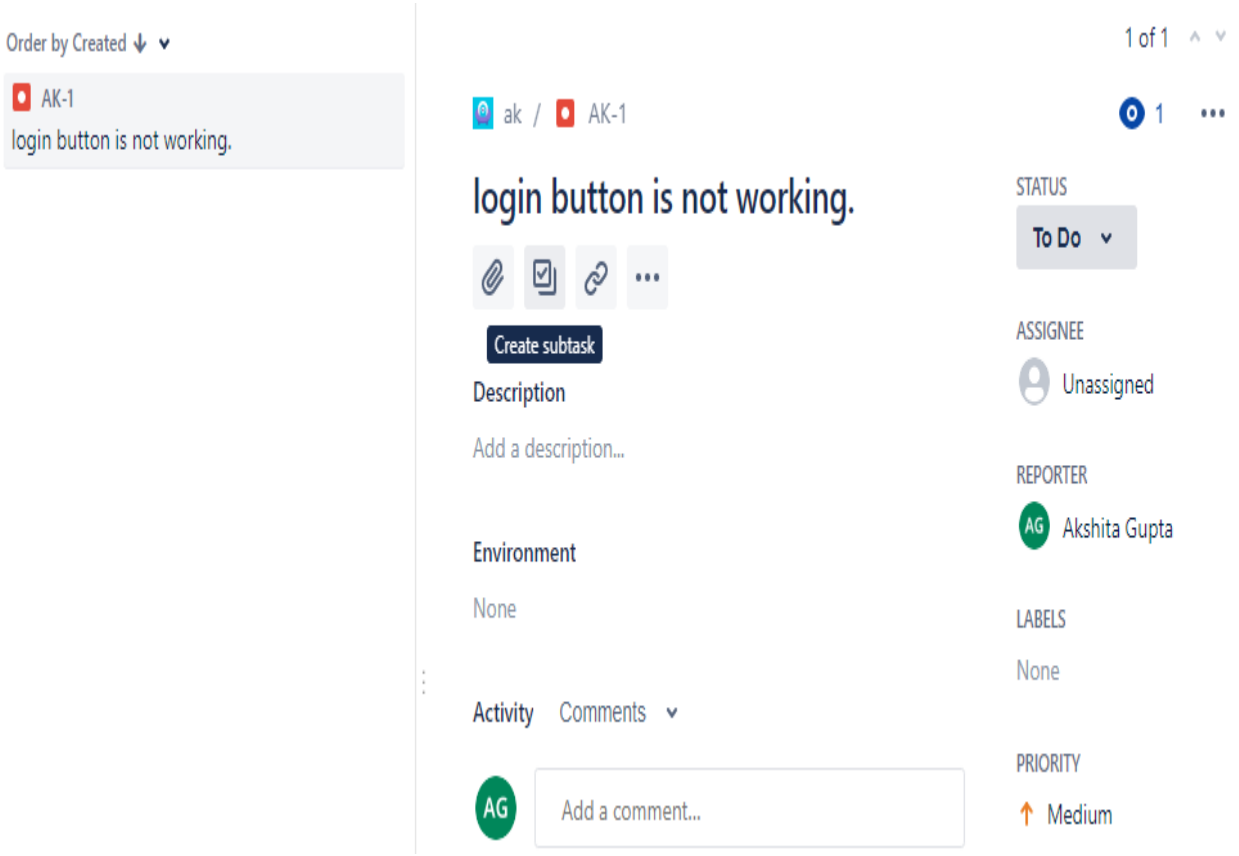
Issue types are used because of the following reasons:

- They support multiple work items. Usually, teams contain multiple work items, and issue types are used to differentiate these work items.

- Each Issue type can have different fields, screens, and workflows. For example, the bug appears on the top of the project board.
- You can report issue types separately. Issues are categorized by issue types; for example, you want to report the progress of your work of the previous week.

Subtask

- Subtask is an issue type that must have a parent issue.
- To create a subtask, click on the create subtask icon for the parent issue.



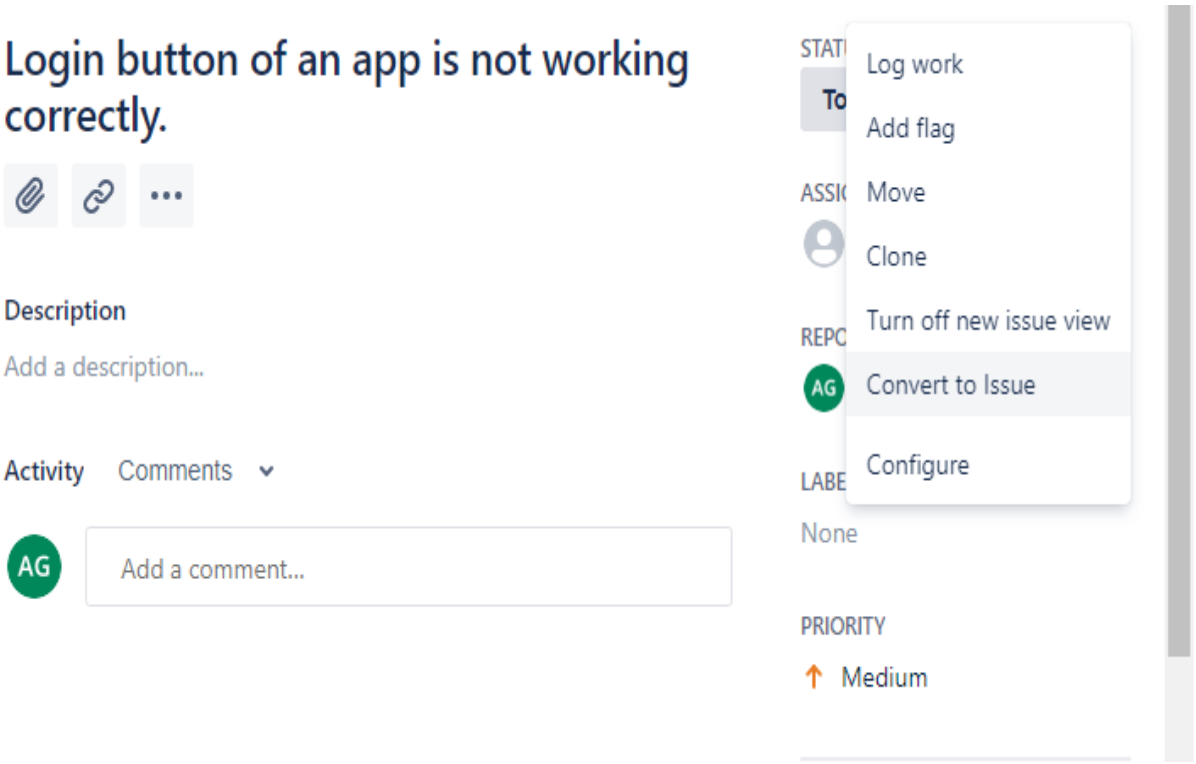
- It converts an issue into individual manageable tasks, and each task is assigned to the team members.
- Subtask is more technical than the parent issue. For example, the type of issue is the Story, then the lines written in story will be non-technical, so that team members and stakeholders can understand. But the subtasks are written in technical line implementing subtasks.

Subtask Characteristics

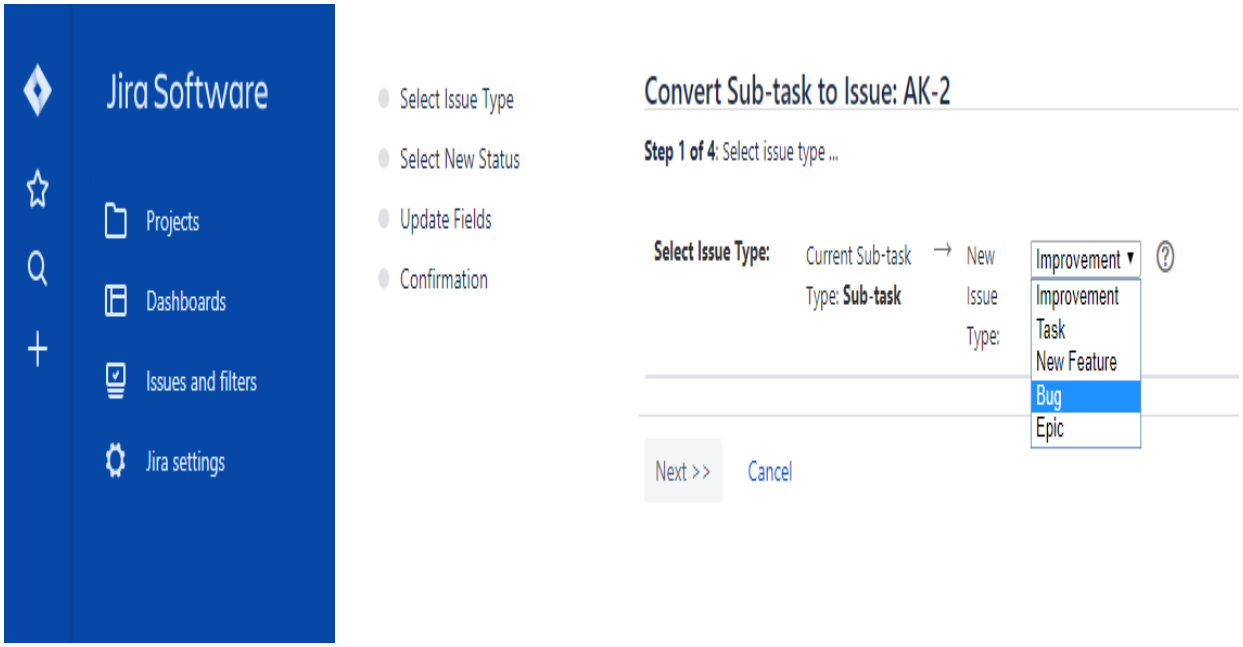
- Subtask has its own issue key and fields.
- Subtasks have independent workflow status
- Subtask issue type can be converted into another issue.

Converting a Sub-task to issue

- Move to the details of a subtask and then click on the **more(...)** button, and then click on the **Convert to Issue**.



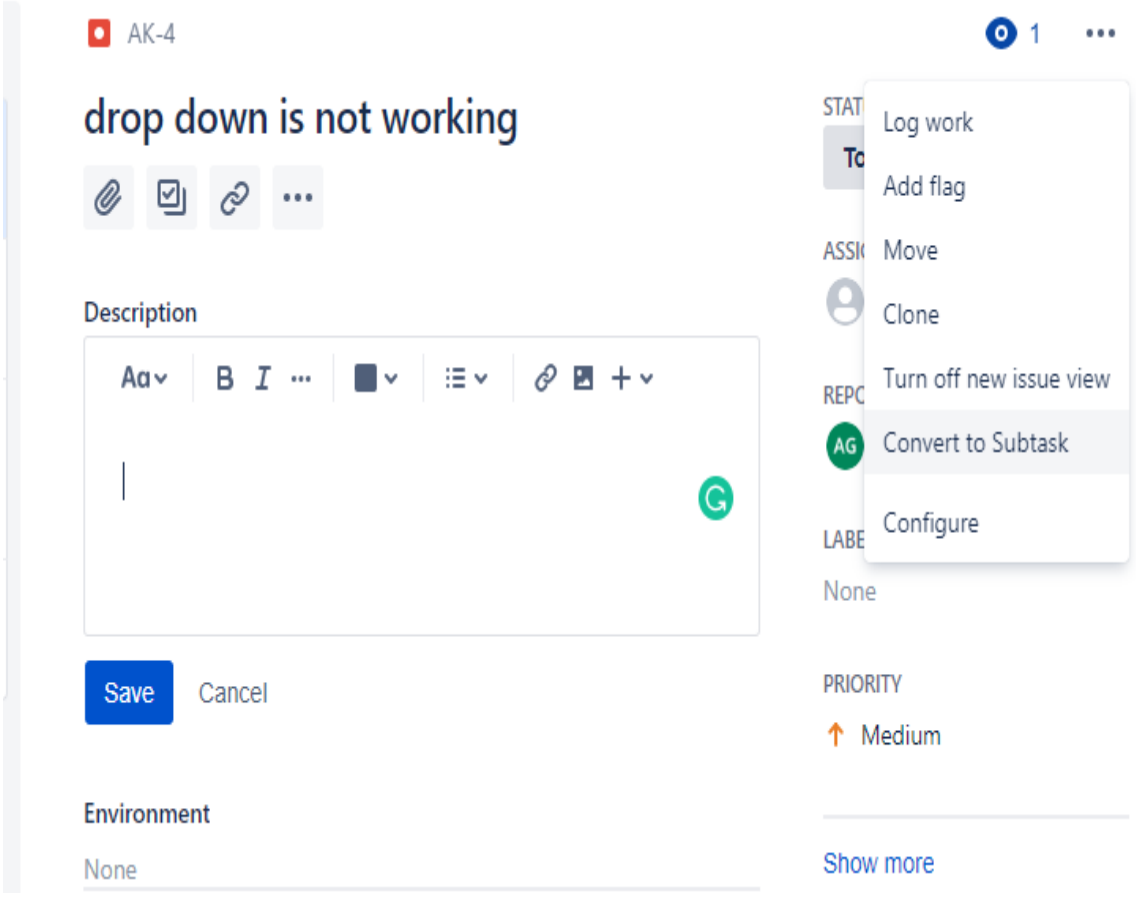
- On clicking on the **Convert to Issue**, the screen appears which is shown below. You can select the issue type of your choice.



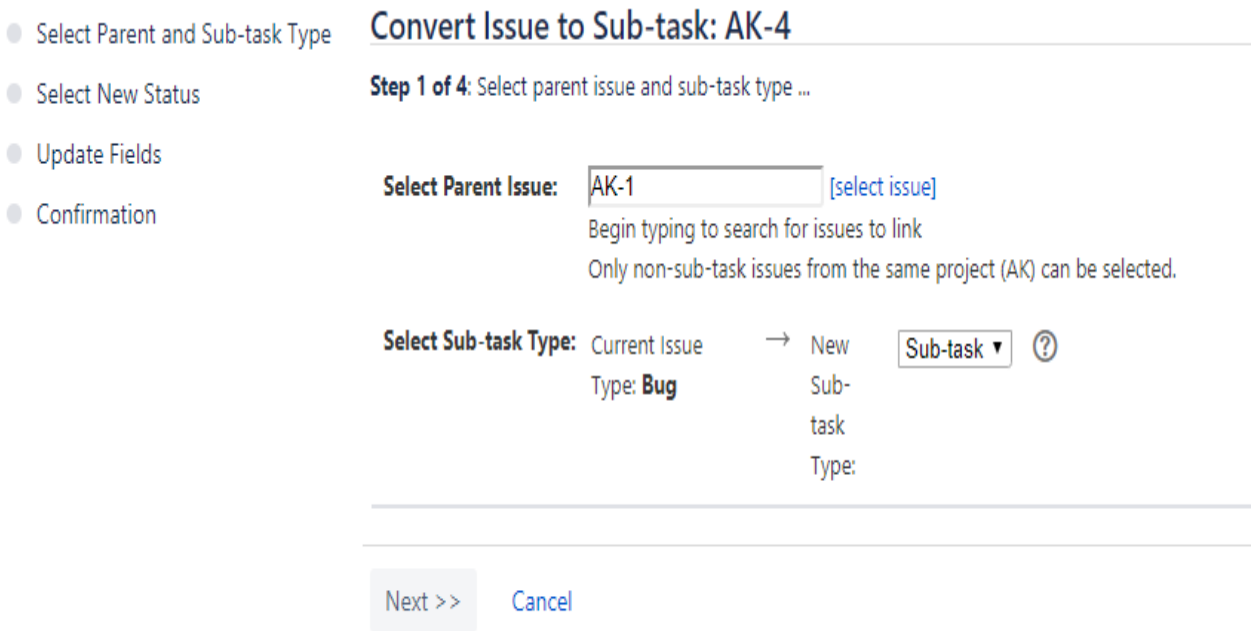
Converting an issue into Subtask

- Open the issue that you want to convert into a subtask.

- Click on the **on** (...) button, then drop down menu will be opened.

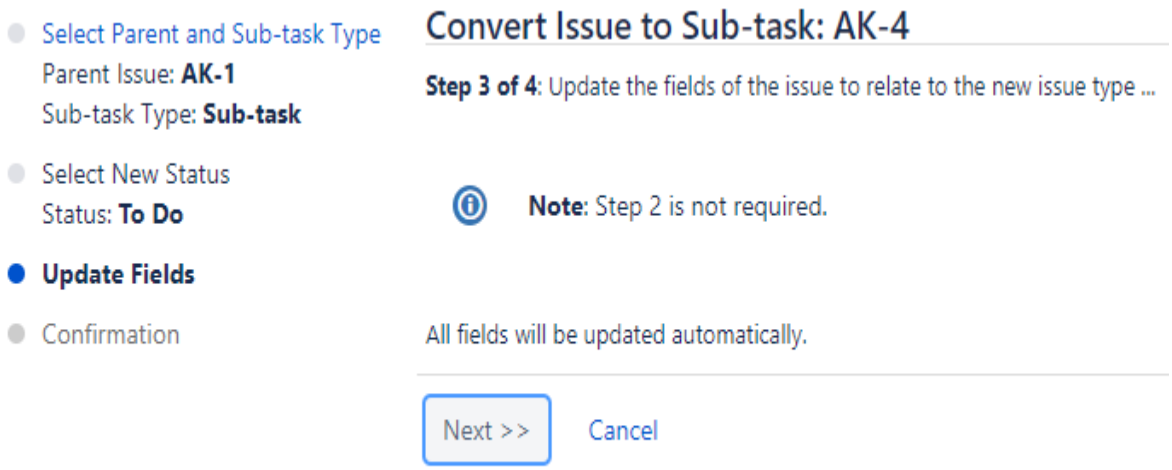


- Click on the **Convert to Subtask** menu item. On clicking on the Convert to Subtask, the screen appears which is shown below:

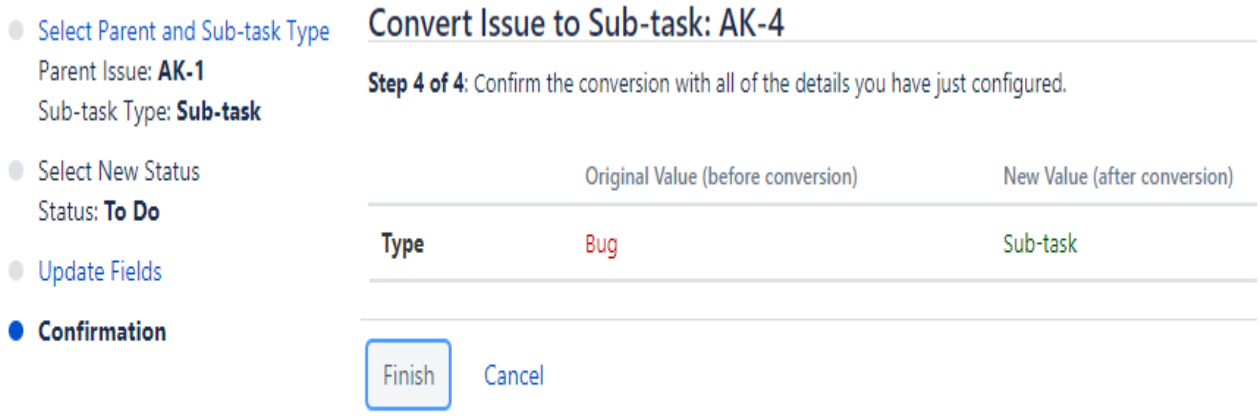


In the above screen, select the parent issue and then click on the Next>>.

- The below screen shows that Step 2 is not required. Click on the Next>>.



- The below screen is Confirmation. Click on the **Finish** button.



JIRA Workflow

Workflow is a set of activities which are performed to track the status and the transition of an issue during the lifecycle of an issue.

Where **transition** represents some work in the form of link between the two statuses when an issue moves from one status to another.

Status: Status determines the impact of the work on the issue which is filed by the tester.

In Jira tool, following are the phases that occur in the workflow:

- **TODO state**
- **In Progress state**
- **Done state**

There are two activities performed in the TODO state:

- **Issue creation**
  When the tester finds a defect, then they log the defect in Jira tool. Once the defect is logged in a Jira tool, the unique ticket identification number is generated by a Jira tool. This process is known as issue creation.
- **Summary and additional details**
  Issue creation requires some additional information which is to be added or updated to an issue such as issue description, priority, severity, components impacted, subtasks, upload screenshots, email history, etc. After adding all the details to an issue, Jira tool assigns the status as a TODO state.

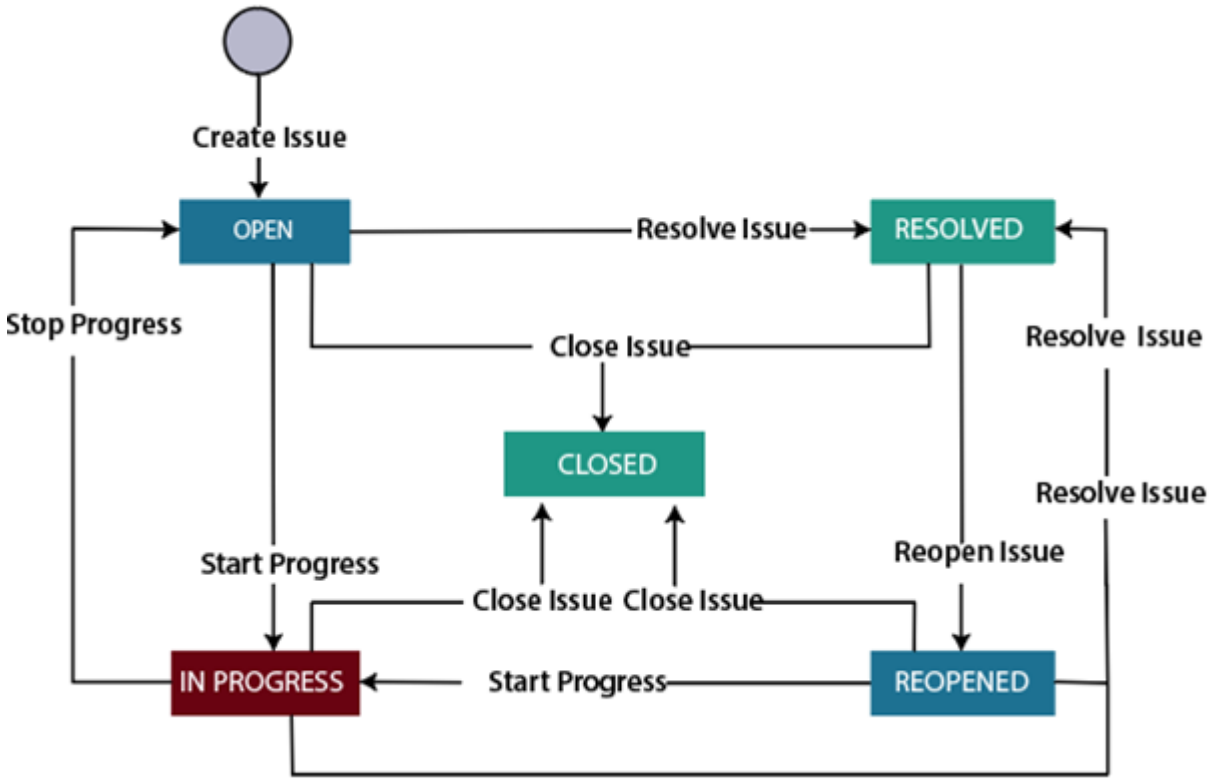
There are three activities performed in the In Progress state:

- **Assignee**
  When the issue is created, then it is assigned to a person or a team. After assigning the issue to a person, then the status changes to **In Progress** state.
- **Work Review**
  Work on the issue is first reviewed and monitored by the issue reporter, assignee, and other project's management folks. When the developer removes an issue, then it is first unit tested by the developer, and then the code review team reviews it. During the work review activity, the status remains in the **In Progress** state.
- **Quality Analysis**
  After code review, the changes are verified by the quality analysis team, which includes regression testing. If the QA team finds any problem in the changes, then the issue is reassigned to the developer otherwise QA team will close the issue which states that the issue has been fixed. During the Quality Analysis, the status remains in the **In Progress** state.

One activity is performed in the **Done** state:

- **Release to production** Once the product has been developed and tested, then the product is released in the market. When the product is released in the market, then the status is changed to **Done** state.

JIRA Workflow can also be referred to as a defect lifecycle. The defect lifecycle is shown below:



- **Open issue:** When the issue is created, then the issue is assigned to the software developer, and they start working on it.
- **In Progress issue:** The software developers start working on the issue.
- **Resolved issue:** Issue is resolved by the software developer but waiting for the verification by the tester. If the verification is successful, then the issue is closed; otherwise, the issue gets reopened.
- **Reopened issue:** When the software developer did not resolve the issue as per the requirements, then the issue comes in a reopened state.
- **Close issue:** When the software developer correctly resolves the issue and verified by the software tester, then the issue will be closed.

What is an Issue?

JIRA Issue is used to track the individual pieces of work that has been completed or not. JIRA issue can be a software bug or any other issue. Once you have imported the project in Jira, you can create issues.

Issue types

Teams work together to break down the pieces of work into issues. Issues can represent software bugs, a project task, subtasks, or any other project work. Jira project comes with the default issue type, and you can also create your own issue type.

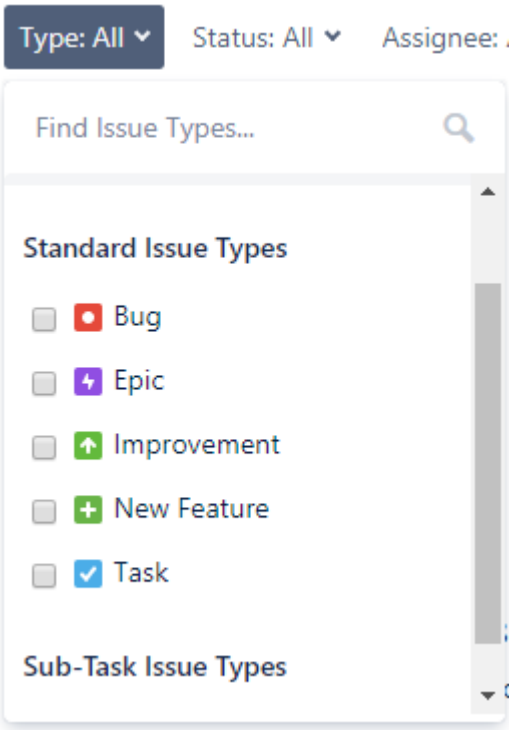
An issue type is a generic name for the unit of work. In Jira, we have different units of work in a project, and issue types field differentiate this.

When you create an issue, then you need to specify the issue type. You can even change the issue type after creating the issue.

Each issue has an icon associated with it to identify the issue type.

Custom Issue type: It is also used to create the customization issue type that provides the flexibility to the team to create their own issue types according to their projects

The below screen shows the list of Issue types:



There are three types of Issue types:

- **JIRA Core Default Issue types**
  - **Task:** The task is a work item done by the team but not connected directly to the user's requirements. For example, to upgrade the version of a product used by the teams.
  - **Subtask:** It is a part of another issue. It is used to break an issue into different pieces of work. While creating an issue, sub task issue is not given in the issue type field drop-down as it contains some parent issue, so we can say that subtask issue cannot be created independently.
- **JIRA Software default issue types**
  - **Story:** It is a requirement from the user's perspective.
  - **Bug:** It is a flaw in a product that needs to be fixed by the developers. It can be tracked with its own issue type to differentiate from other types of work.
  - **Epic:** An epic is a big issue that contains other issues.
- **JIRA Service desk default issue types**
  - Incident
  - Service Request
  - Change
  - Problem

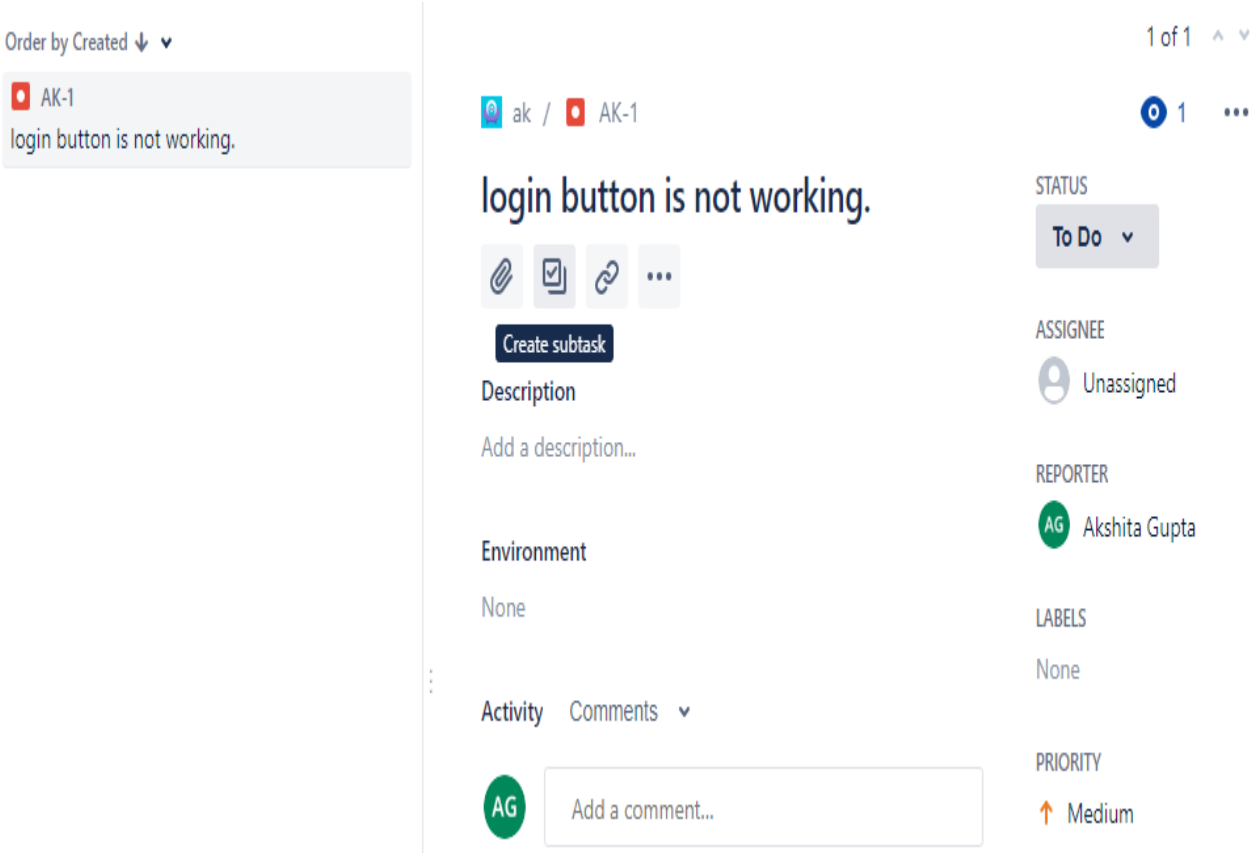
Why Issue types?

Issue types are used because of the following reasons:

- They support multiple work items. Usually, teams contain multiple work items, and issue types are used to differentiate these work items.
- Each Issue type can have different fields, screens, and workflows. For example, the bug appears on the top of the project board.
- You can report issue types separately. Issues are categorized by issue types; for example, you want to report the progress of your work of the previous week.

Subtask

- Subtask is an issue type that must have a parent issue.
- To create a subtask, click on the create subtask icon for the parent issue.



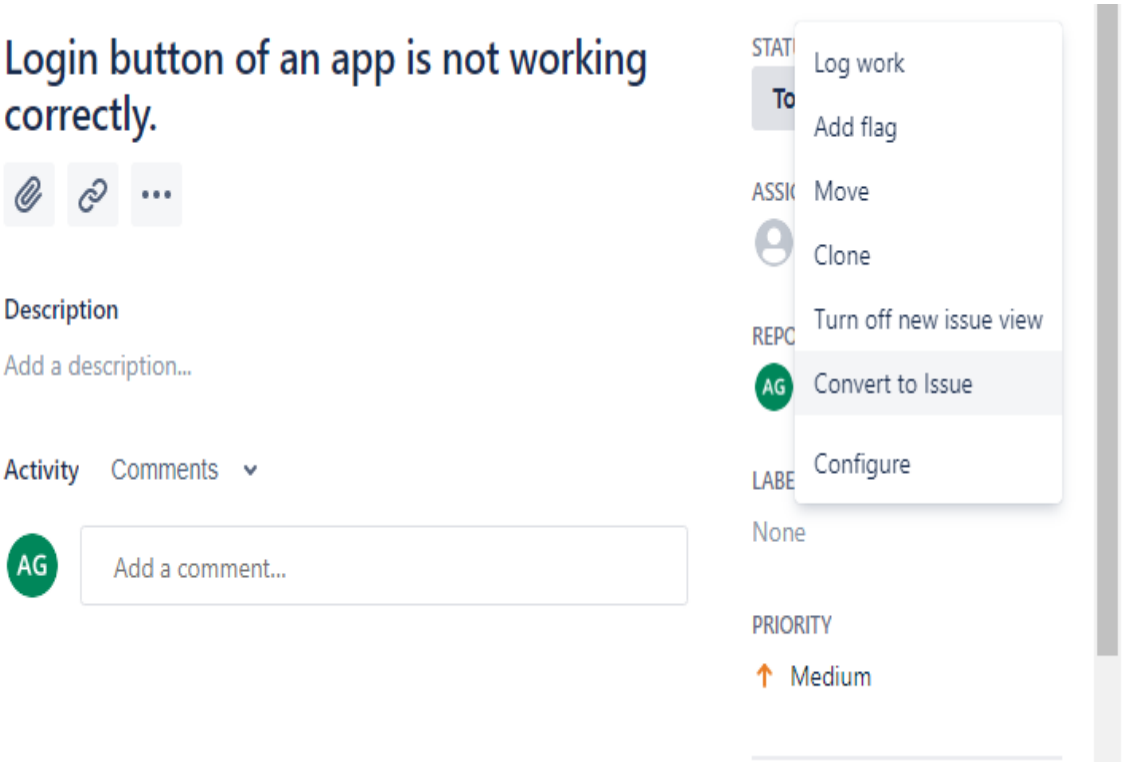
- It converts an issue into individual manageable tasks, and each task is assigned to the team members.
- Subtask is more technical than the parent issue. For example, the type of issue is the Story, then the lines written in story will be non-technical, so that team members and stakeholders can understand. But the subtasks are written in technical line implementing subtasks.

Subtask Characteristics

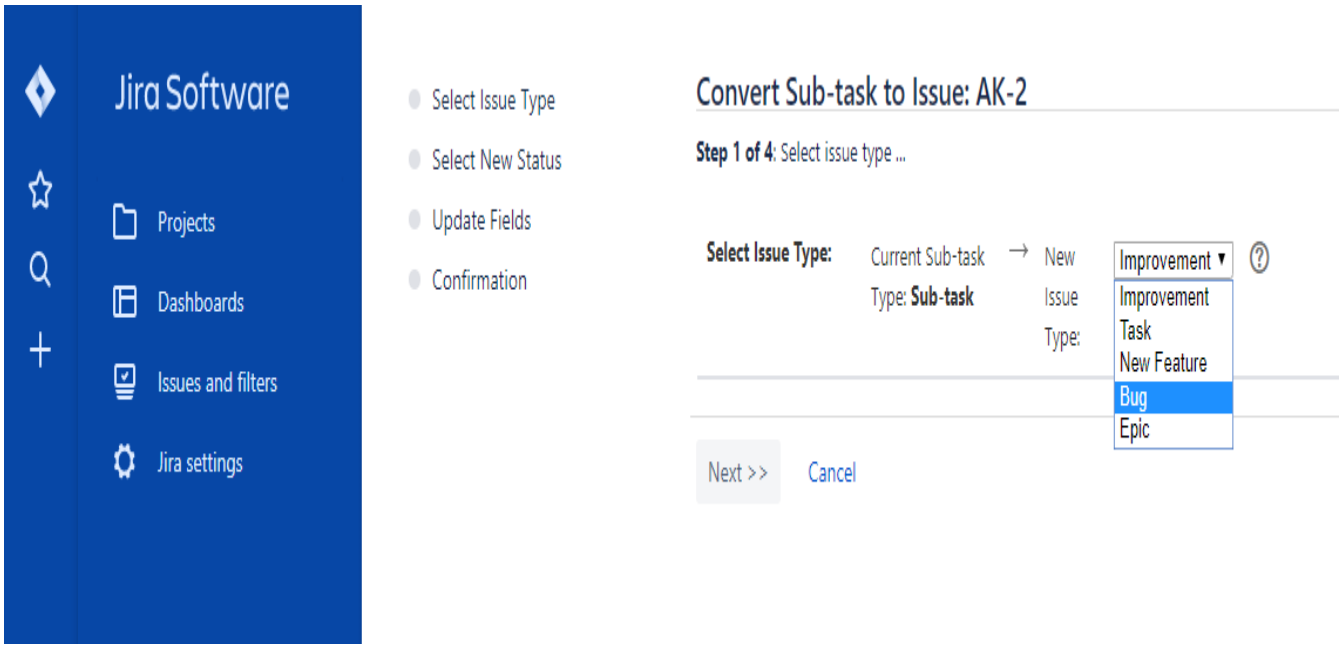
- Subtask has its own issue key and fields.
- Subtasks have independent workflow status
- Subtask issue type can be converted into another issue.

Converting a Sub-task to issue

- Move to the details of a subtask and then click on the **more(...)** button, and then click on the **Convert to Issue**.

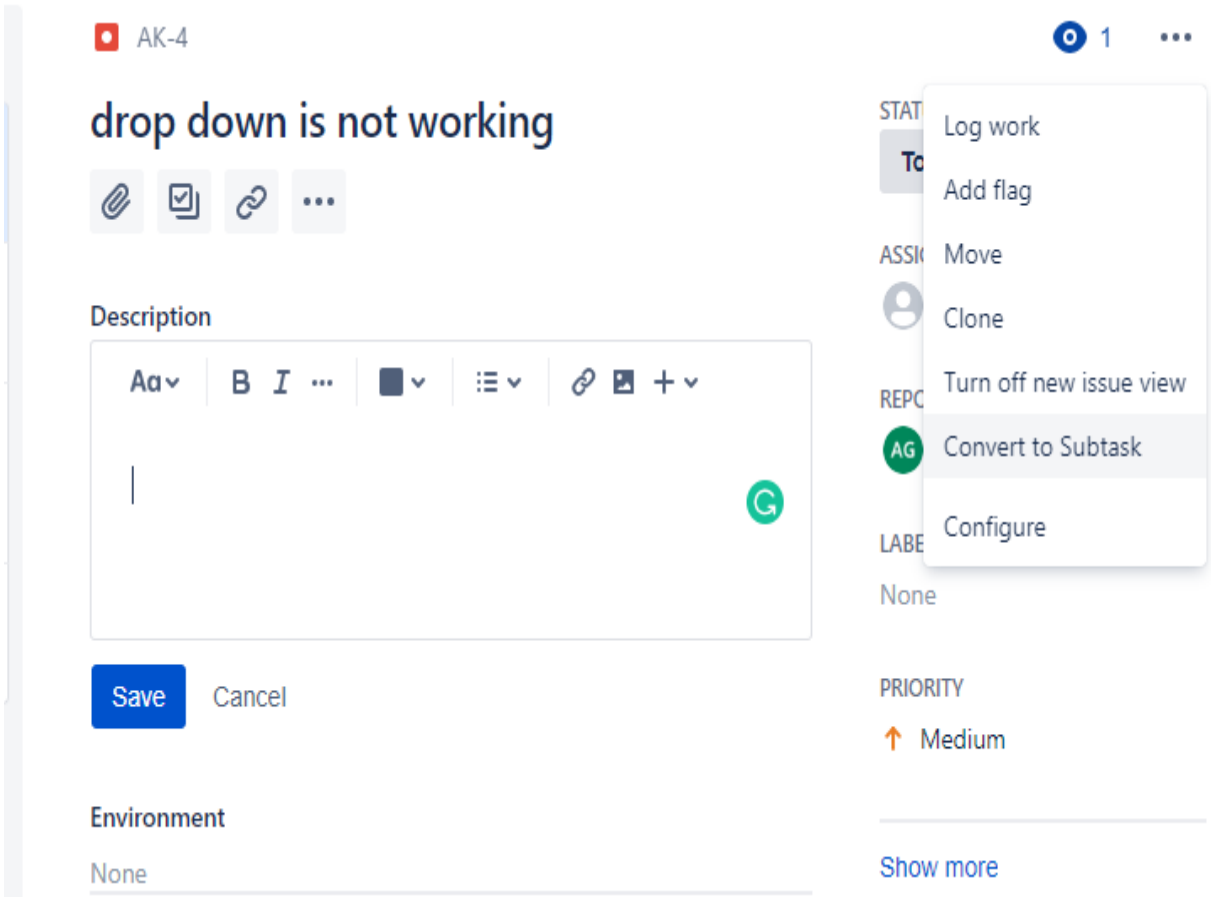


- On clicking on the **Convert to Issue**, the screen appears which is shown below. You can select the issue type of your choice.

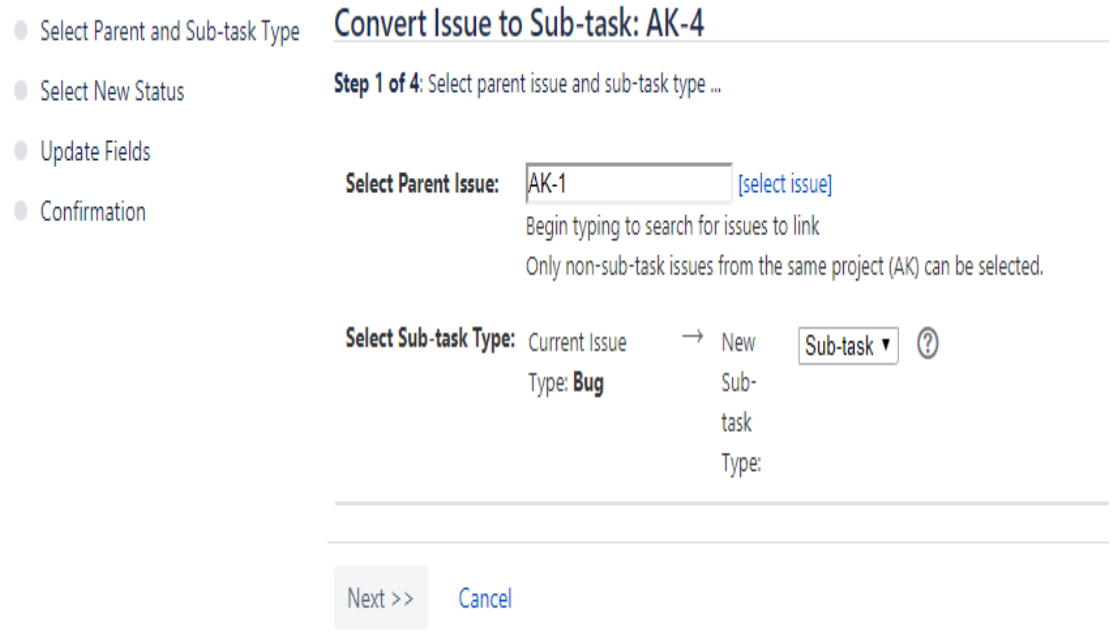


Converting an issue into Subtask

- Open the issue that you want to convert into a subtask.
- Click on the **on** (...) button, then drop down menu will be opened.

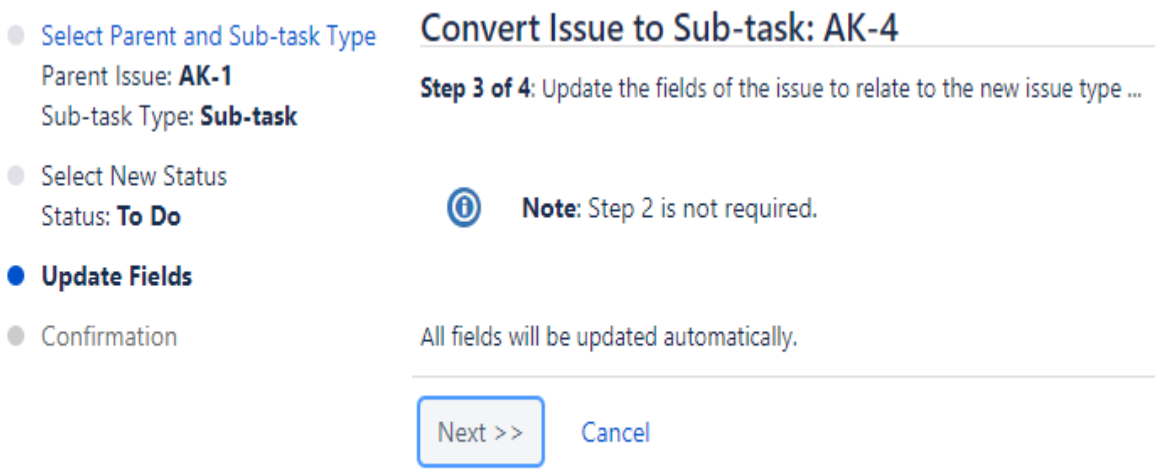


Click on the **Convert to Subtask** menu item. On clicking on the Convert to Subtask, the screen appears which is shown below:

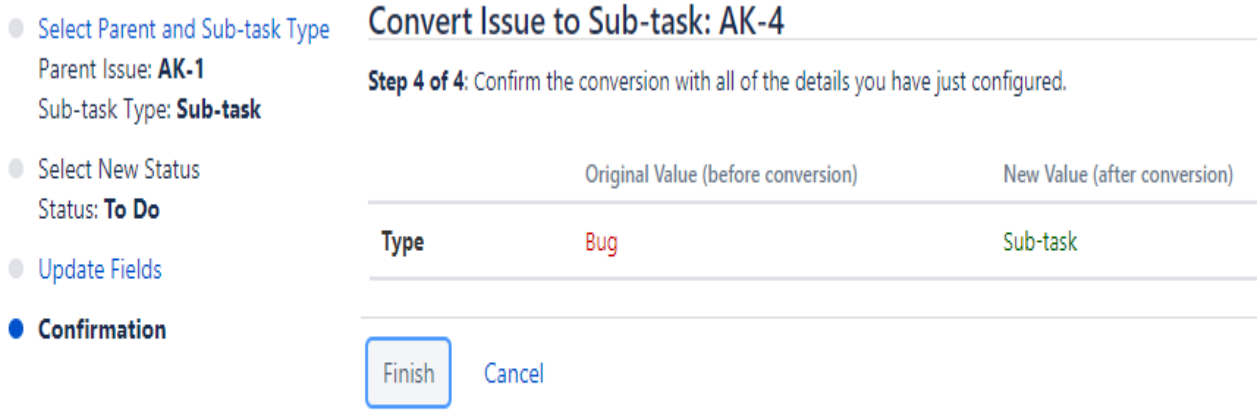


In the above screen, select the parent issue and then click on the Next>>.

- The below screen shows that Step 2 is not required. Click on the Next>>.



- The below screen is Confirmation. Click on the **Finish** button.



JIRA Dashboard

When you log in to Jira, the first screen that will appear is Dashboard. The Dashboard can only be customized by the Admin, and based on the roles, an admin can access to Jira.

An admin can even change the color and the Jira logo.

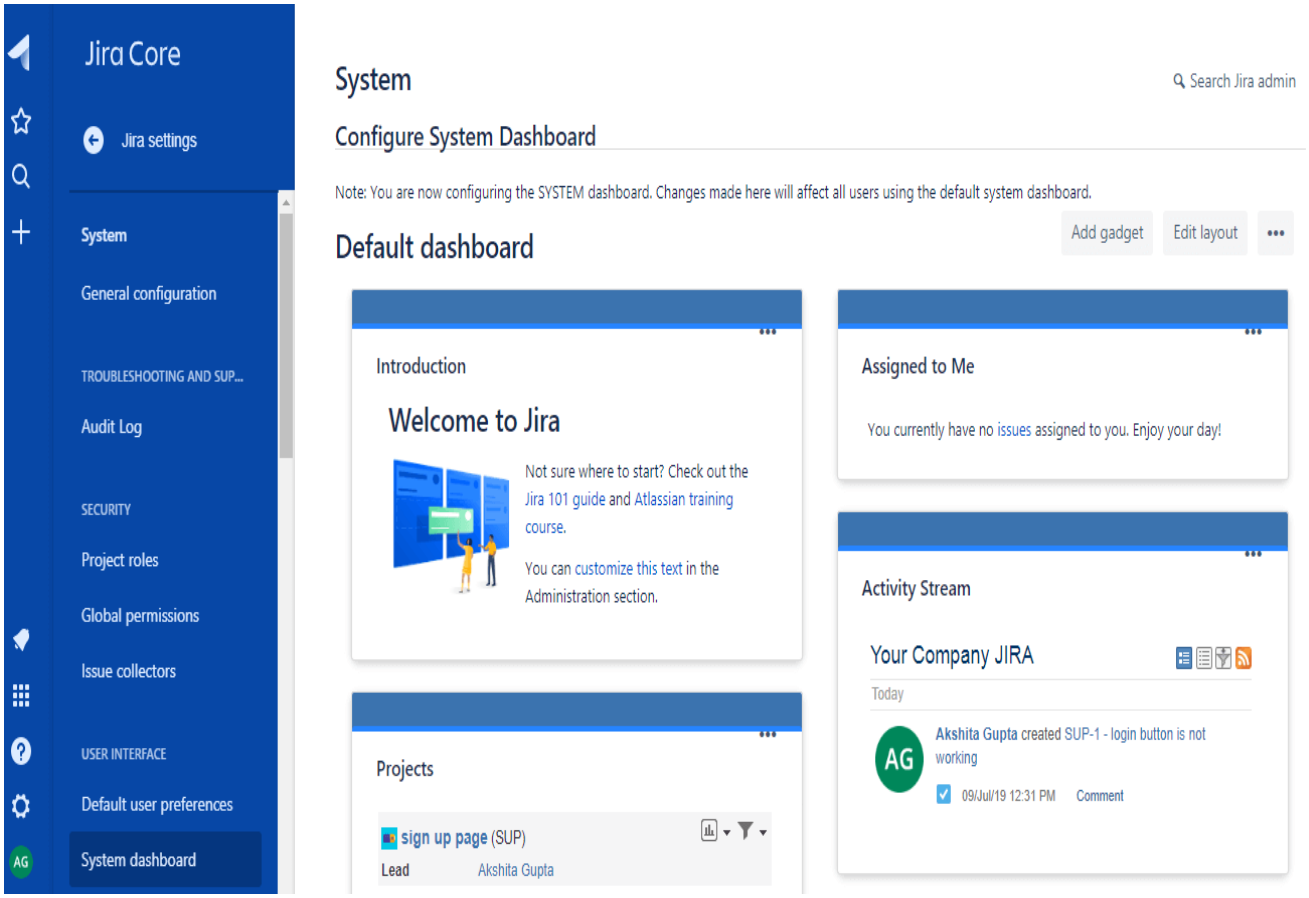
You can create multiple dashboards that help you to organize your projects, assignments, and achievements in different charts.

Default Dashboard

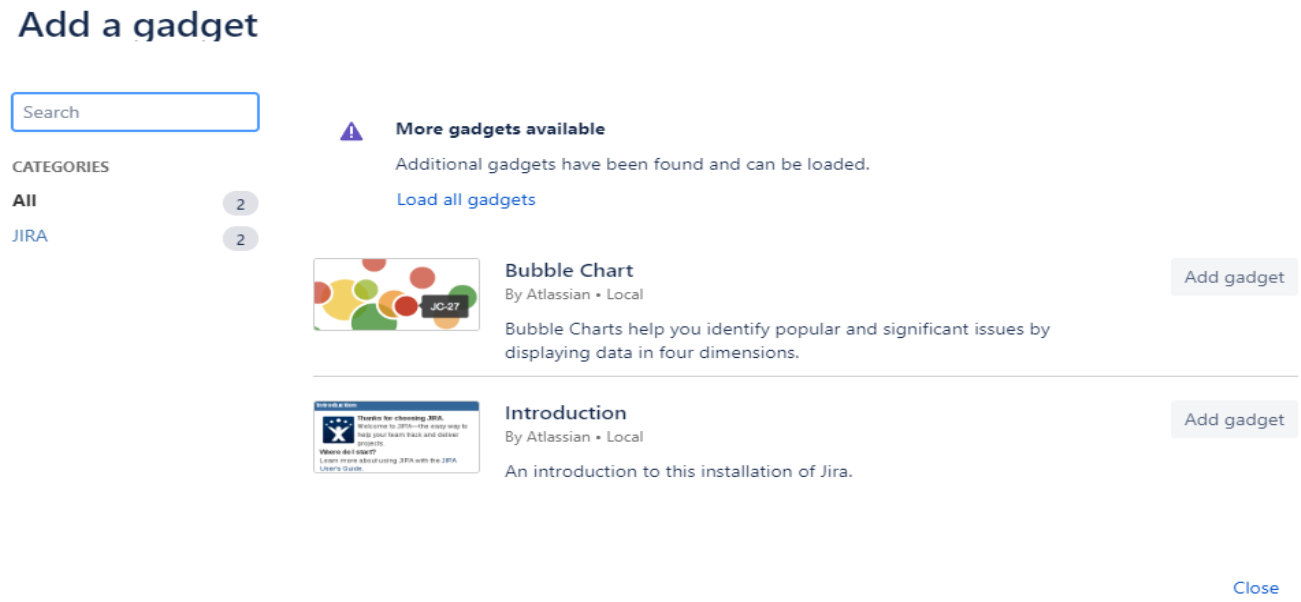
You can add the gadgets in a default dashboard as well as you can configure the default dashboard. The layout of the dashboard can also be configured, such as a number of columns. All the changes made to the default dashboard will also reflect the changes in the dashboard of the users that are currently using the default dashboard. For example, the Administration gadget exists in the Default dashboard, but it is visible to only admin.

Following are the steps which can be used to add the gadget in default dashboard:

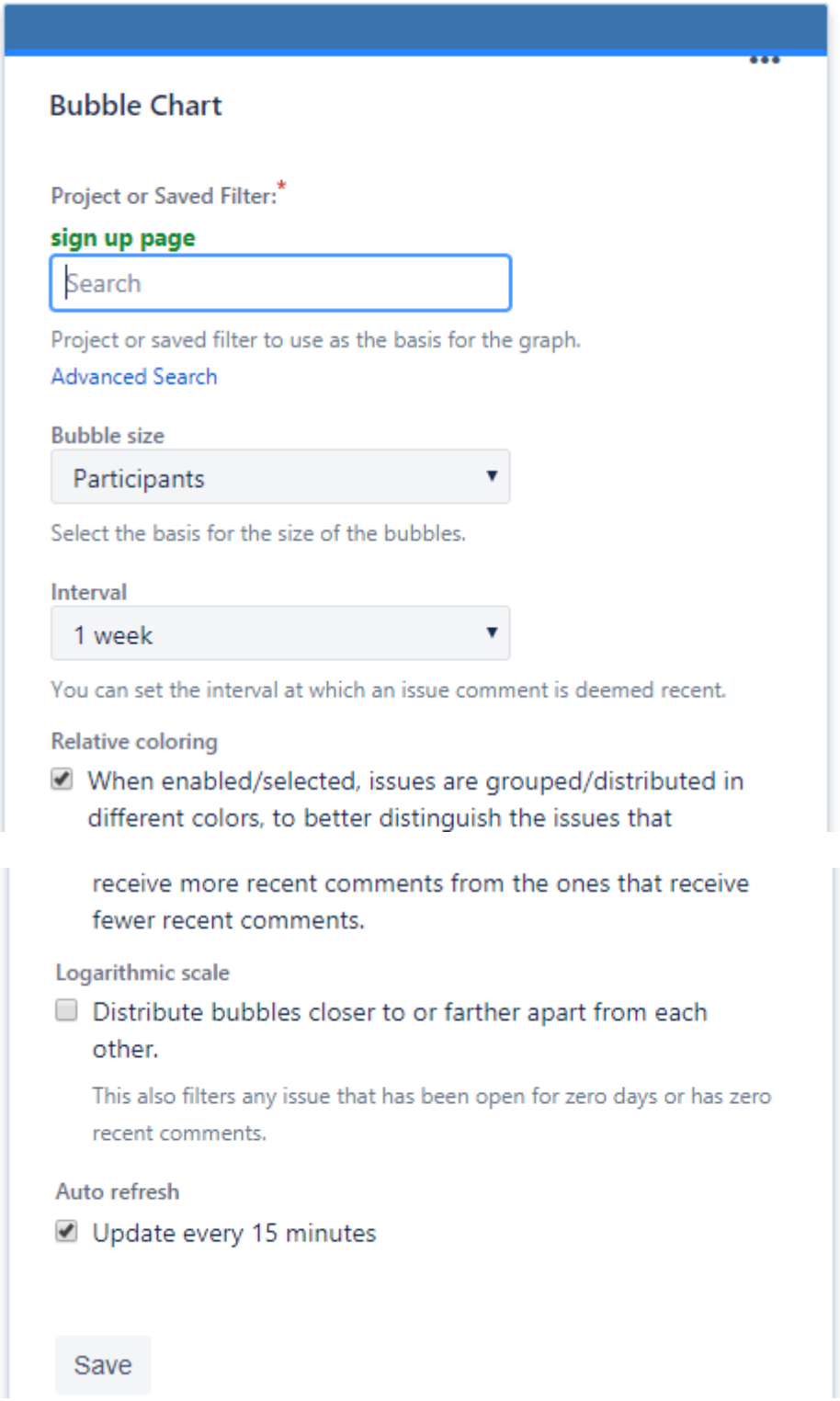
- Login to the Jira website.



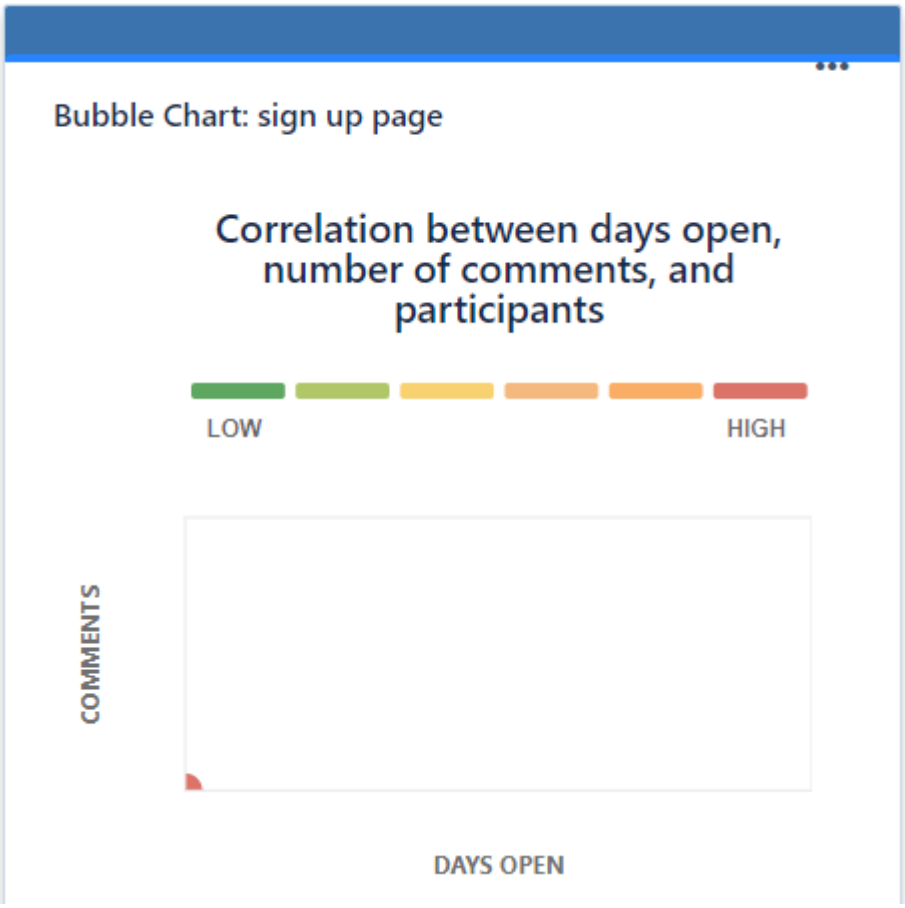
- Click on the **Add gadget** link.



- After adding the gadget, the gadget is added in your dashboard and then click on the Save button.



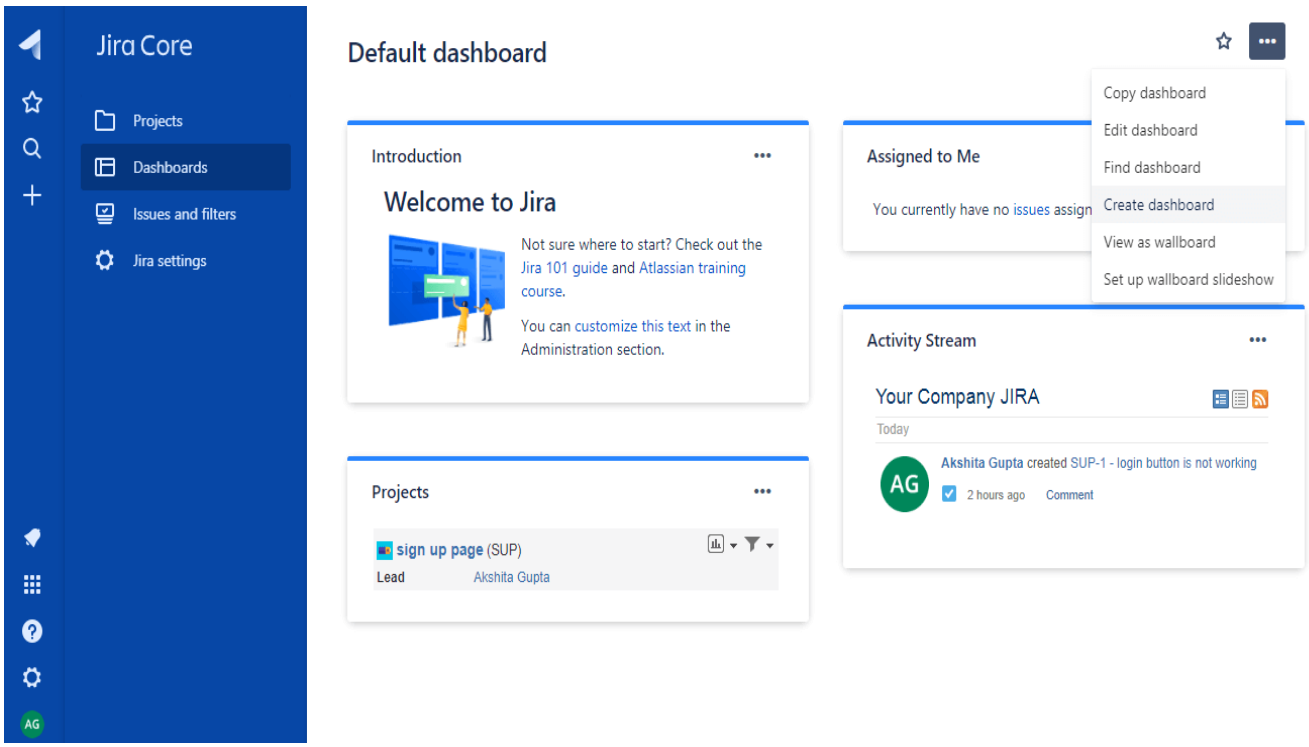
- When you save the gadget, then the bubble chart is created, which is shown below:



Create a Dashboard

The following are the steps to create a dashboard:

- Click on the '...' button.

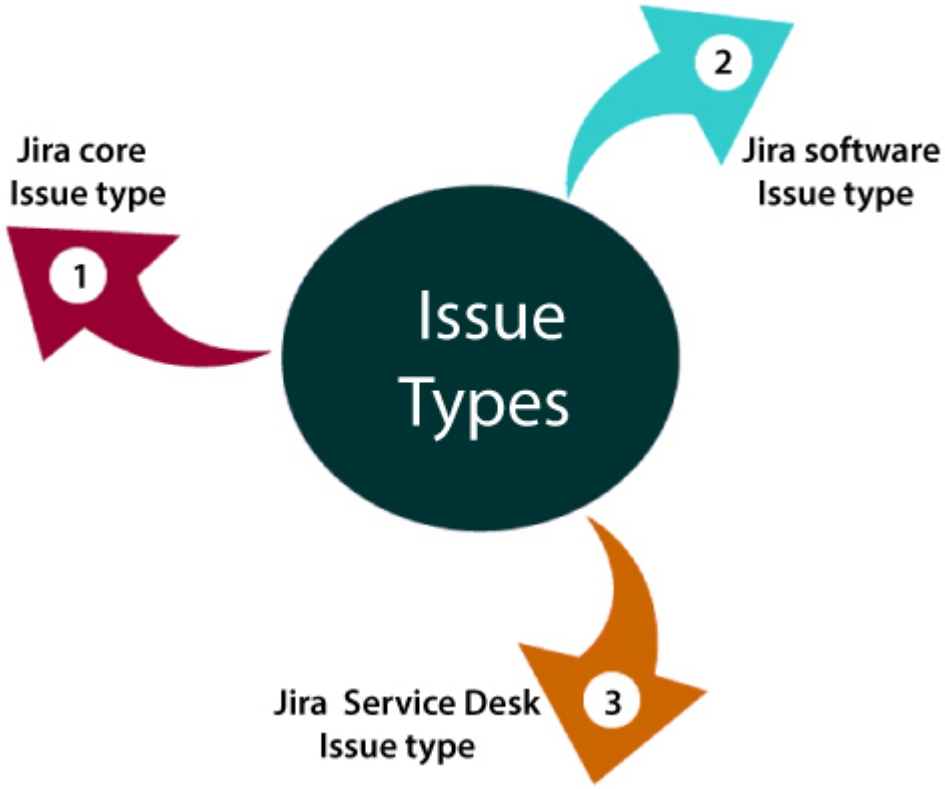


- Click on the Create dashboard from the drop-down menu of "...". On clicking on the Create dashboard, the screen appears, which is shown below:

Jira Issue types

JIRA application can be used to break the pieces of work into issues. Issues can be represented as tasks, subtasks, bud, epic, feature requests, or other pieces of work. Each Jira software comes with some default issue types that suit your projects and teams.

There are three types of default Jira issue types that come with the JIRA software:



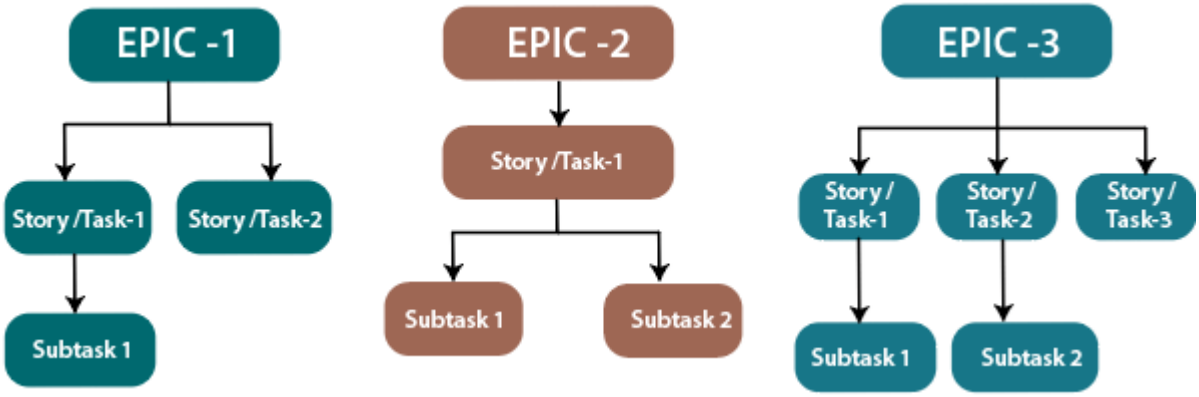
- Jira Core (business projects) issue types
- Jira Software (software projects) issue types
- Jira Service desk (service desk projects) issue types

Jira Core issue types

- **Task**
  The task is the work which is to be completed or done to achieve the team's goal.
- **Subtask**
  It is the subtask of an issue. All the tasks that come under the logged issue are known as subtasks.

Jira Software issue types

- **Bug**
  A bug is the problem or defect that occurs in the functions of a product.
- **Epic**

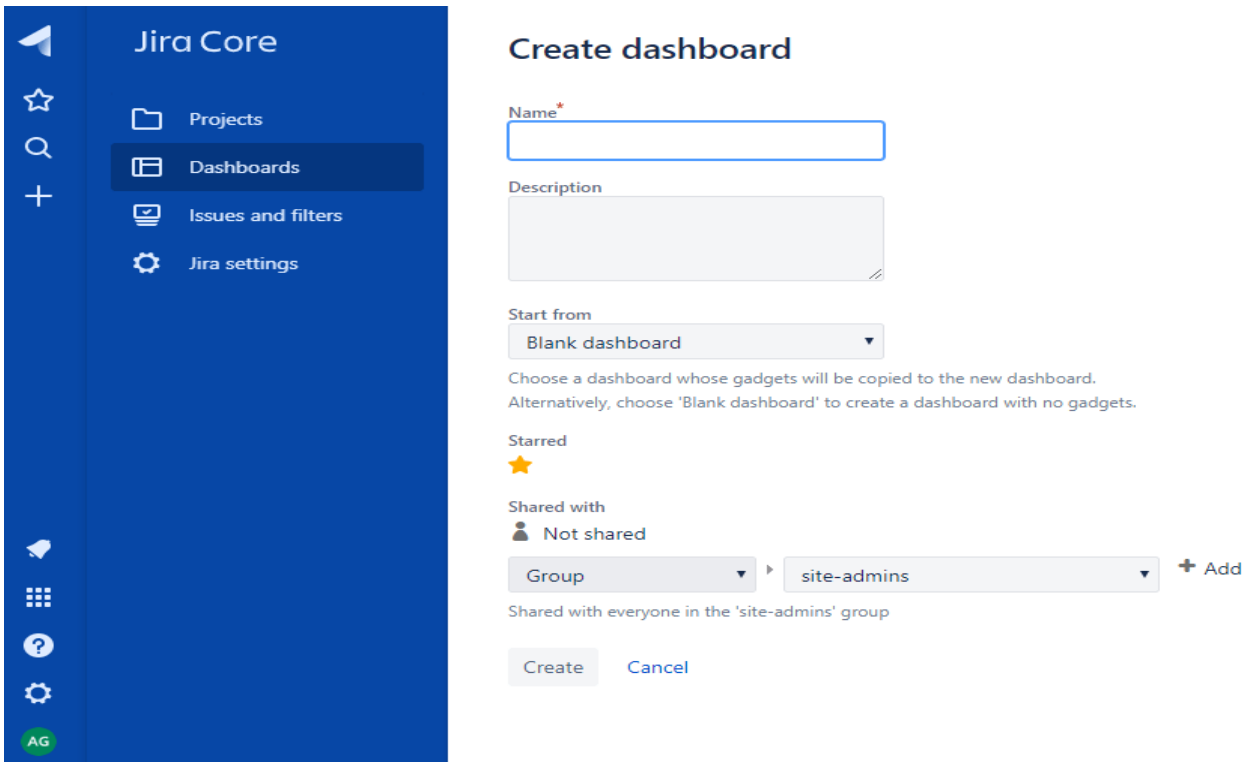


  - Epic is a large user story which is to be broken into smaller stories
  - It cannot be achieved in a single sprint.
  - The whole epic is completed in months.
  - Epic refers to a set of activities which are not converted into user stories yet.
  - First, Epics are converted into user stories, and then user stories are converted into several tasks on which agile team work.
  - Epics are broad in scope and lacking details, and they split into smaller and multiple stories on which agile team work on.
  - Epic is referred to as a 'top-tier in the work hierarchy.
- **Subtask**
  Subtask is a piece of work which is to be done to complete the whole work.
- **Task**
  The task is the work which is to be completed or done to achieve the team's goal.
- **Story**
  - The story is a list of tasks that need to be completed within a project.
  - It defines the high-level design of project requirements.
  - It defines the short and simple descriptions of the whole project.
  - It is owned by the product owner of a company, but anyone can write the user story.
  - It is written in simple language so that customers can understand the final product.
  - User stories can be considered as the "**heart of scum**" as it serves as the building blocks for the sprint.

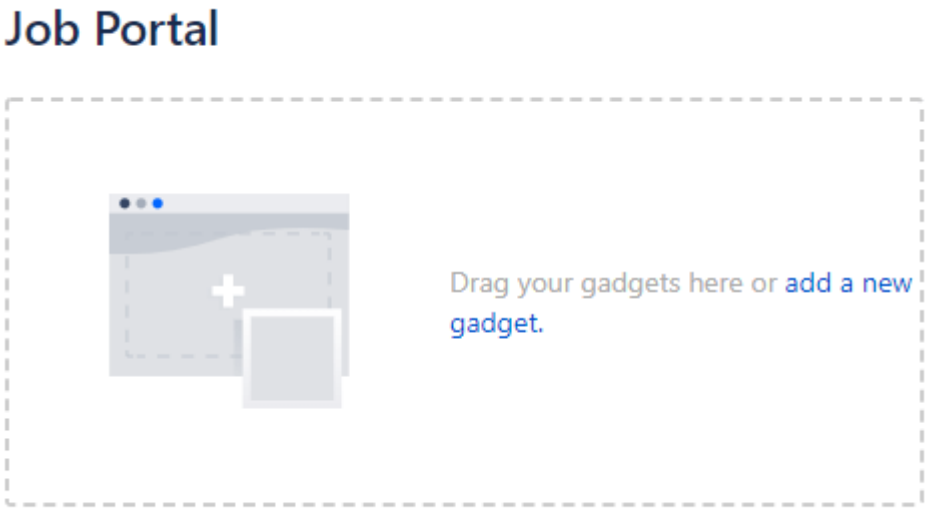
Jira Service Desk issue types

- **Change**
  It requests a change in the current IT profile.
- **IT help**
  It is used to request for help for an IT related problem.

- **Incident**
  It reports an incident or service IT outage.
- **New feature**
  It is requesting to add a new feature or software capability.
- **Problem**
  It is used to report the root cause of multiple incidents.
- **Service Request**
  It is used for requesting help from an internal or customer service desk.
- **Service Request with approval**
  It is used for requesting help that requires manager or board approval.
- **Support**
  It is used for requesting help for the customer support issues.



- Enter the details of the dashboard such as Name of the dashboard, description, etc and then click on the create button. Suppose I provide the name of the dashboard as Job Portal. The below screen shows that the dashboard has been created:

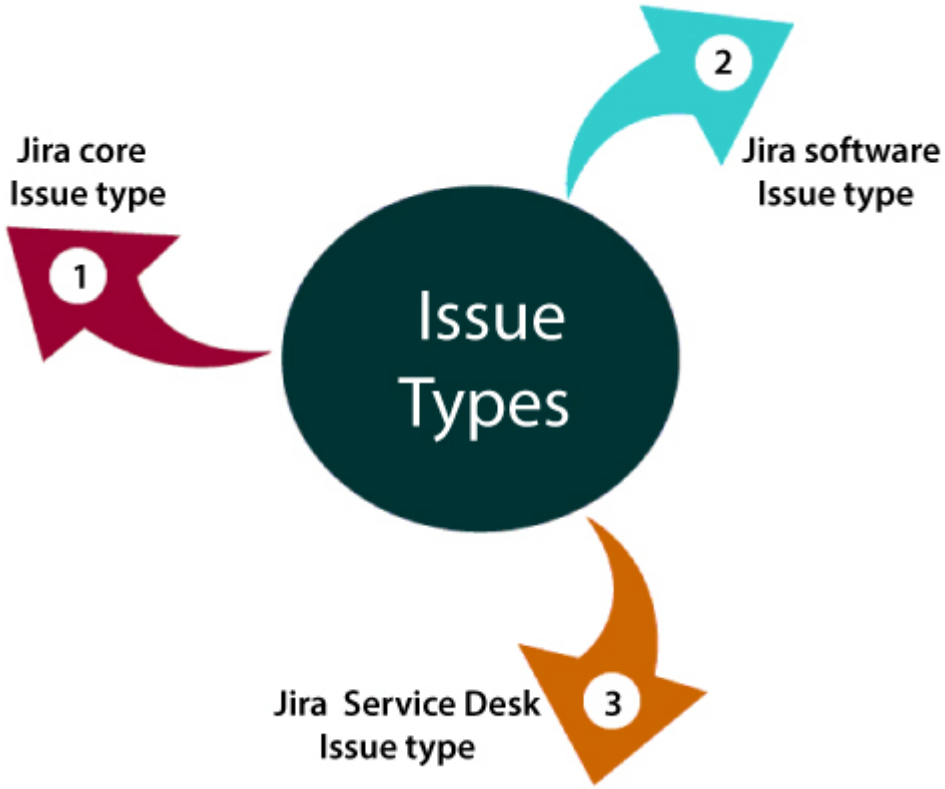


Note: We can also edit the Dashboard by clicking on the '...' button and then click on the Edit Dashboard.

Jira Issue types

JIRA application can be used to break the pieces of work into issues. Issues can be represented as tasks, subtasks, bud, epic, feature requests, or other pieces of work. Each Jira software comes with some default issue types that suit your projects and teams.

There are three types of default Jira issue types that come with the JIRA software:



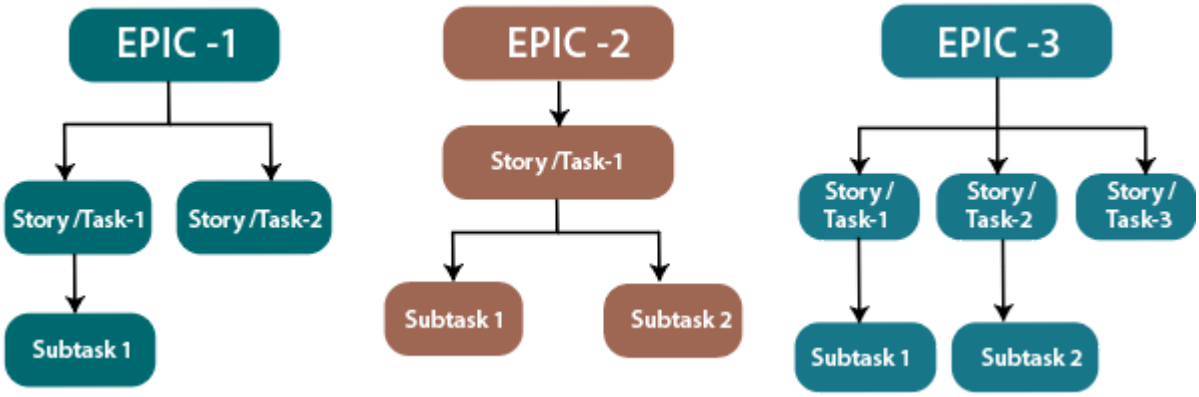
- Jira Core (business projects) issue types
- Jira Software (software projects) issue types
- Jira Service desk (service desk projects) issue types

Jira Core issue types

- **Task**
  The task is the work which is to be completed or done to achieve the team's goal.
- **Subtask**
  It is the subtask of an issue. All the tasks that come under the logged issue are known as subtasks.

Jira Software issue types

- **Bug**
  A bug is the problem or defect that occurs in the functions of a product.
- **Epic**



- Epic is a large user story which is to be broken into smaller stories
- It cannot be achieved in a single sprint.
- The whole epic is completed in months.
- Epic refers to a set of activities which are not converted into user stories yet.
- First, Epics are converted into user stories, and then user stories are converted into several tasks on which agile team work.
- Epics are broad in scope and lacking details, and they split into smaller and multiple stories on which agile team work on.
- Epic is referred to as a 'top-tier in the work hierarchy.

- **Subtask**
  Subtask is a piece of work which is to be done to complete the whole work.
- **Task**
  The task is the work which is to be completed or done to achieve the team's goal.
- **Story**
  - The story is a list of tasks that need to be completed within a project.
  - It defines the high-level design of project requirements.
  - It defines the short and simple descriptions of the whole project.
  - It is owned by the product owner of a company, but anyone can write the user story.
  - It is written in simple language so that customers can understand the final product.
  - User stories can be considered as the "**heart of scum**" as it serves as the building blocks for the sprint.

Jira Service Desk issue types

- **Change**
  It requests a change in the current IT profile.
- **IT help**
  It is used to request for help for an IT related problem.
- **Incident**
  It reports an incident or service IT outage.
- **New feature**
  It is requesting to add a new feature or software capability.

- **Problem**
  It is used to report the root cause of multiple incidents.
- **Service Request**
  It is used for requesting help from an internal or customer service desk.
- **Service Request with approval**
  It is used for requesting help that requires manager or board approval.
- **Support**
  It is used for requesting help for the customer support issues.

Jira Versions

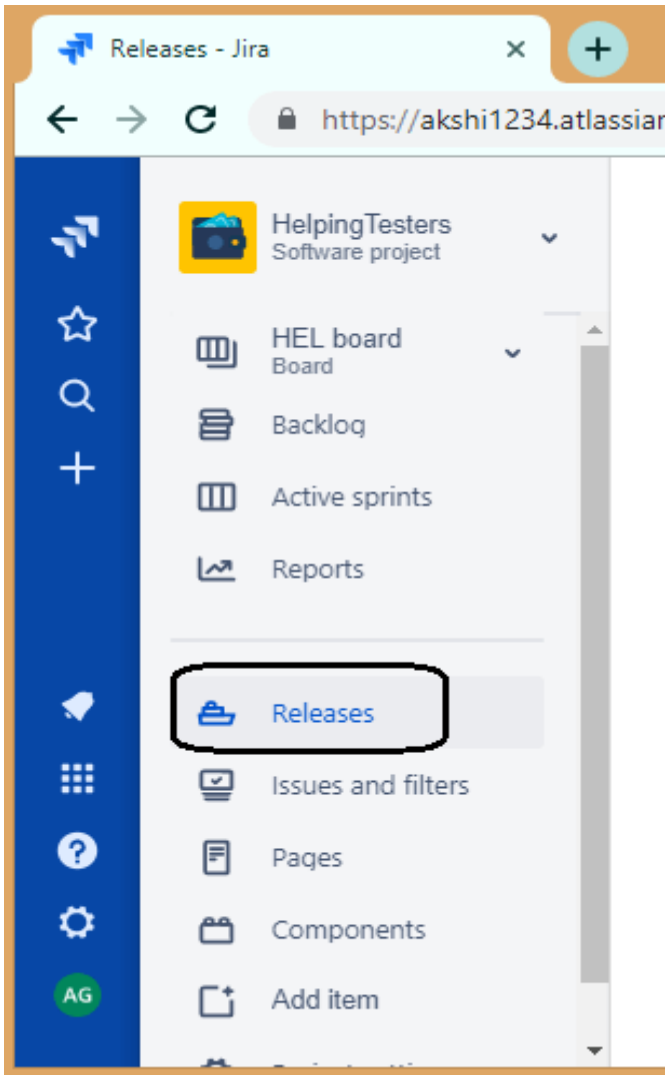
Versions are used to organize and schedule the releases. If the version is created and issues are assigned to it, then we can use the version to filter information in various reports.

You can assign issues to a specific version and can organize the sprints in that version.

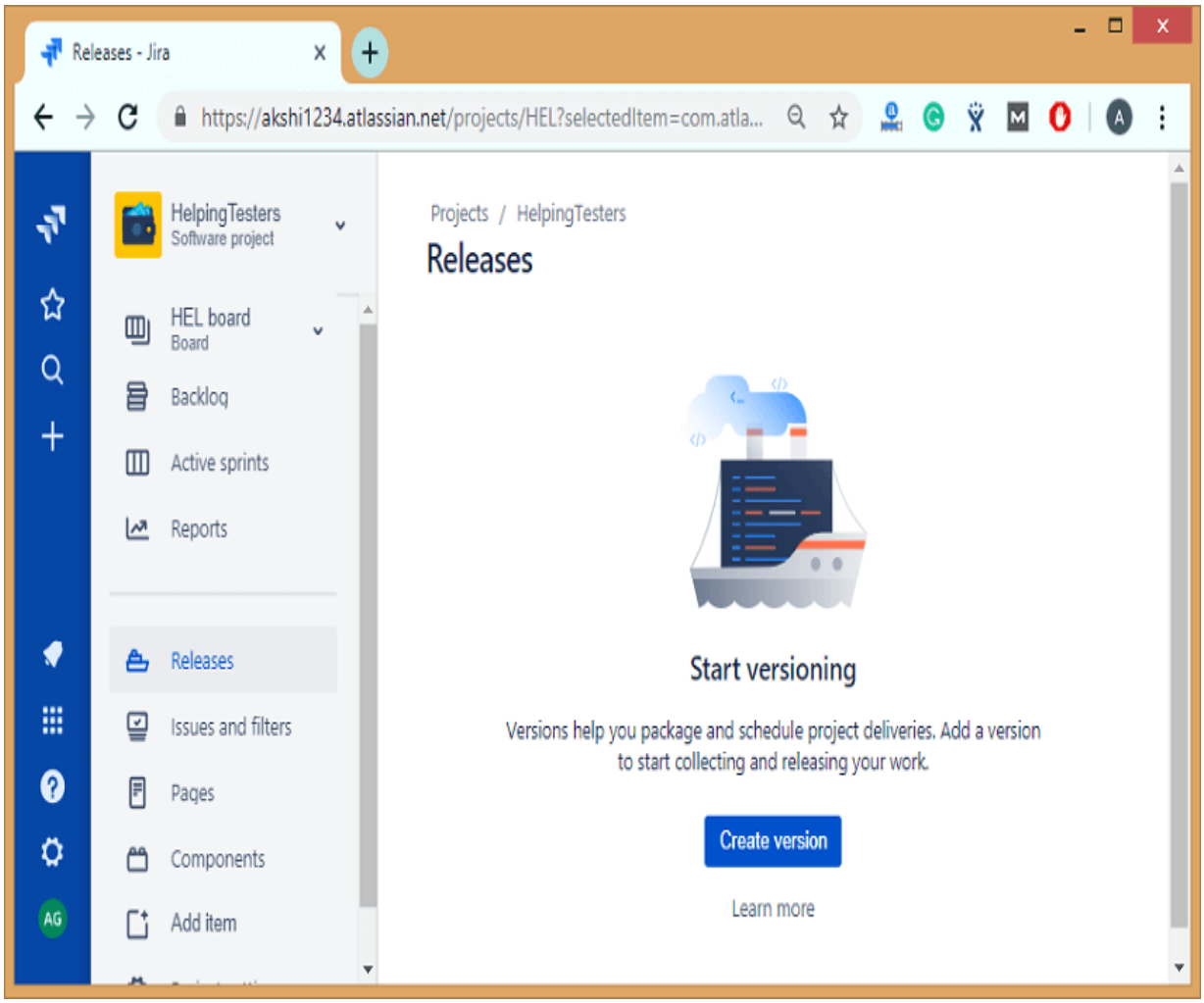
Creating a Version

Step 1: Move to your project.

Step 2: Click on the **Releases** appearing at the left side of the panel.

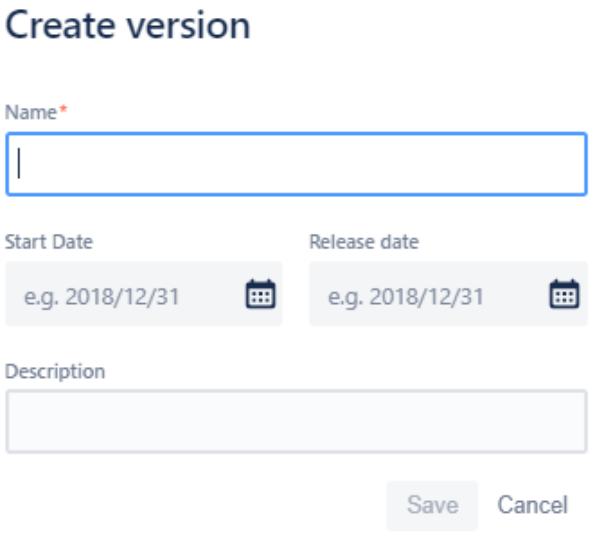


Step 3: After clicking on the **Releases link**, the screen appears, which is shown below:



Step 4: Click on the Create version button.

Step 5: On clicking on the **Create version** button, the pop-up window appears asking for the



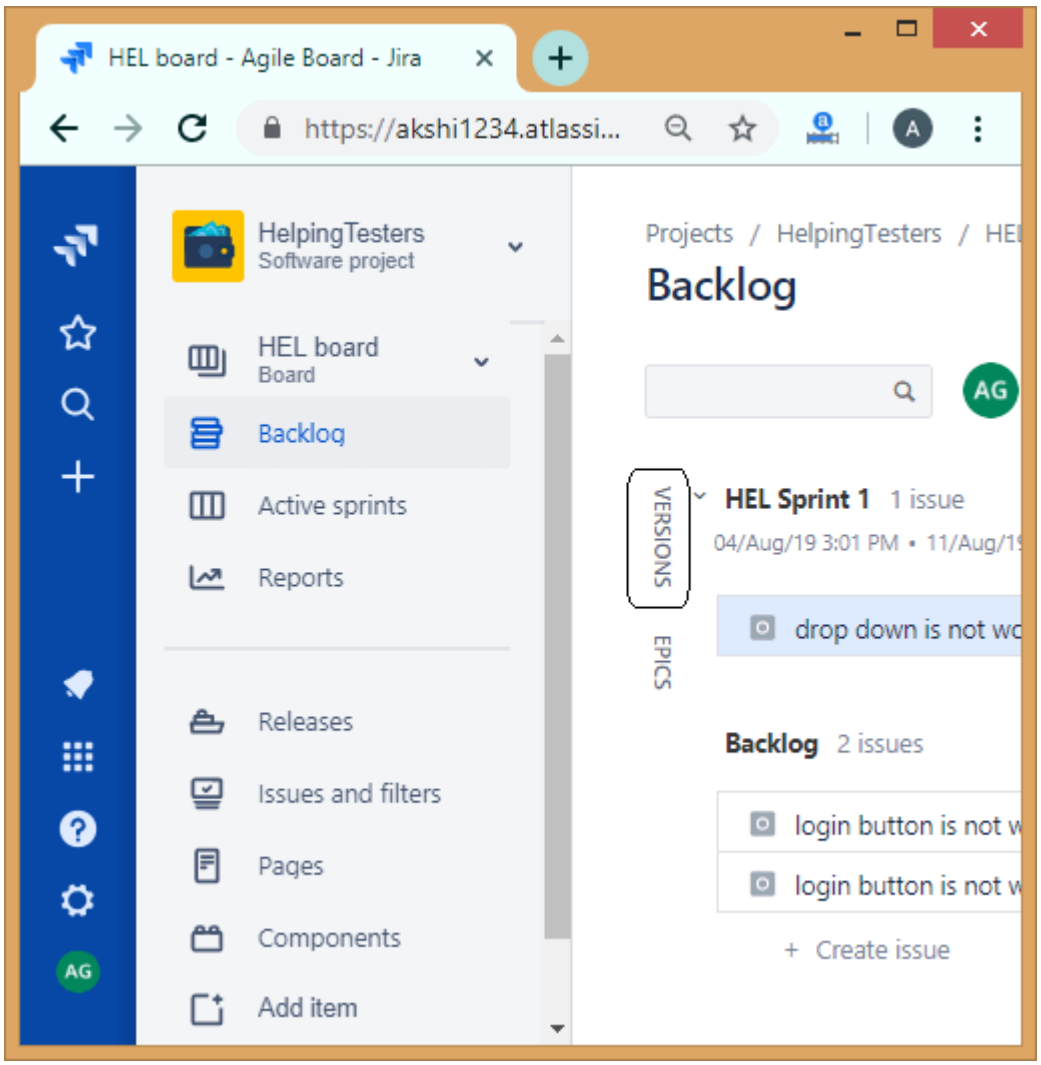
version name and its description.

Adding issues to version

Step 1: Move to your project.

Step 2: Click on the Backlog appearing at the left side of the panel.

Step 3: Click on the Versions panel.



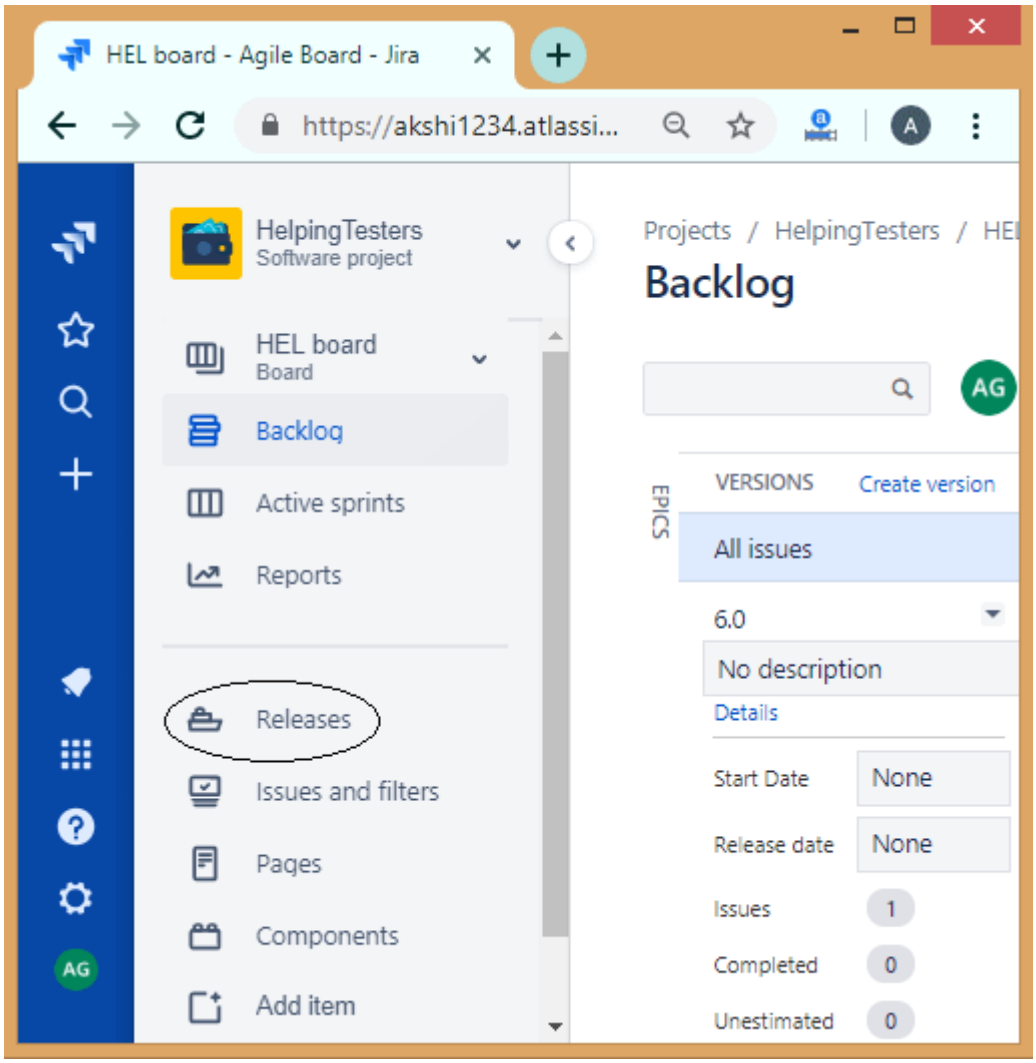
Step 4: Drag the issue to a version that you want to add in a version.

Complete a version

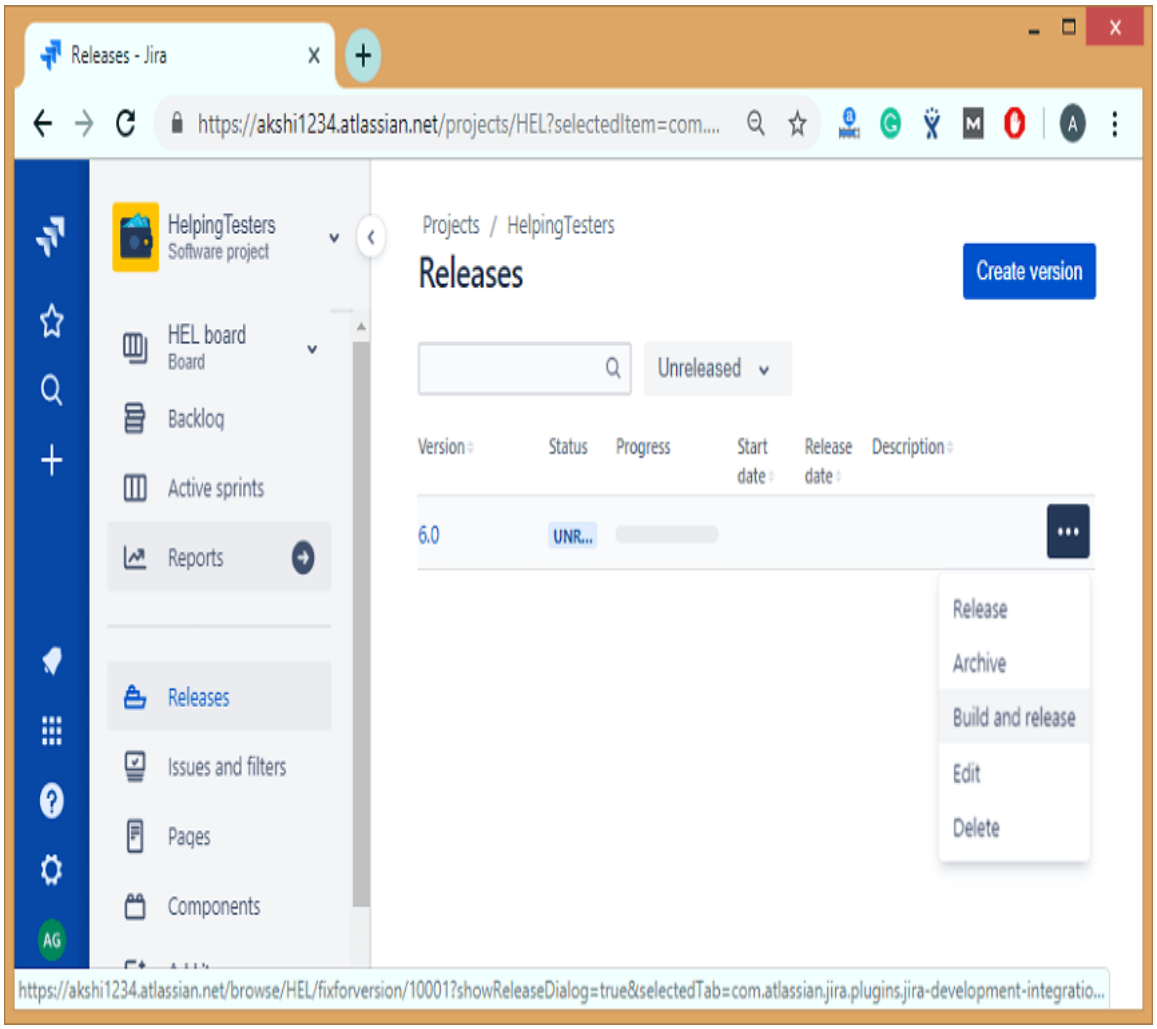
You can complete a version when your version is ready to be released. The version is ready to be released when issues are completed, and code is checked and reviewed.

Step 1: Navigate to your project.

Step 2: Click on the **Releases link** appearing at the left side of the panel.



Step 3: To release the version, go to ellipsis (...) > Release.



Jira Backlog

A backlog is a set of activities or issues that the team needs to be resolved within a specific iteration. All the issues of your project are grouped in a backlog and sprint.

In scrum backlog, you can perform the multiple activities such as create and update issues, drag and drop the issues to prioritize them, assign them to sprints, epics, versions, manage epics, etc.

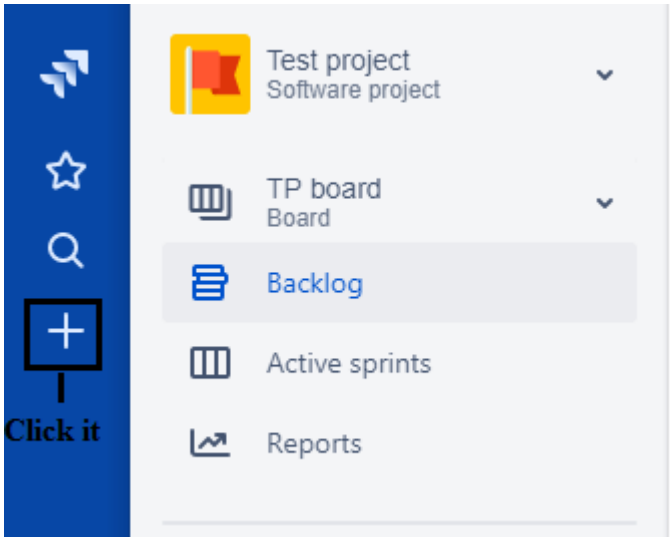
The following are the activities that can be performed in the scrum backlog:

- **Add issues to the backlog**
  To add an issue in the backlog, you need to create an issue. Click on the + icon in the global sidebar to create an issue. Fill all the details related to an issue, and then click on the create button. An issue which you have created will be added to the backlog.
- **Prioritize the backlog**
  You can also drag and drop an issue so that you can prioritize the issue in a backlog. You can also right-click on the issue to open the menu that allows you to move the issue to the top or the bottom in a backlog.
- **View and edit an issue's details**
  Click an issue on the board to view all the details of the issue. If you want to see the issue in a separate window, then right-click on the issue key.

- **Identify the workload for users**
  The users to whom the work assigned are shown at the top of the sprint. Click on the '...' option to view the workload for assignees.
- **Create subtasks**
  Click on an issue, and then click on the icon to create the subtask. Subtasks are useful for breaking a story into implementable parts.
- **Transition an issue**
  You can drag and drop an issue from the Backlog section to the relevant sprint, or we can say that we can move the issue from the Backlog section to the relevant sprint.
- **Split an issue**
  Right click on the issue and then select the split issue. It can split the issue in Backlog as well as in the sprint section. Splitting is useful when you want to break the big task into two or more issues to make the work more manageable.
- **Delete an issue**
  You can also delete an issue by selecting an issue, and then select **...>Delete**.
- **Find issue**
  You can create your own Quick Filters to view only those issues that you want.

Steps to create a backlog

- Click on the + appearing on the leftmost side of the pane.



- Fill all the details to create an issue.

Create issue

Import issues | Configure fields

Project*

Test project (TP)

Issue Type*

Bug

Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

Summary*

Components

None

Description

Style | B | I | U | A | ³A | | | | | |

Fix versions

None

Priority

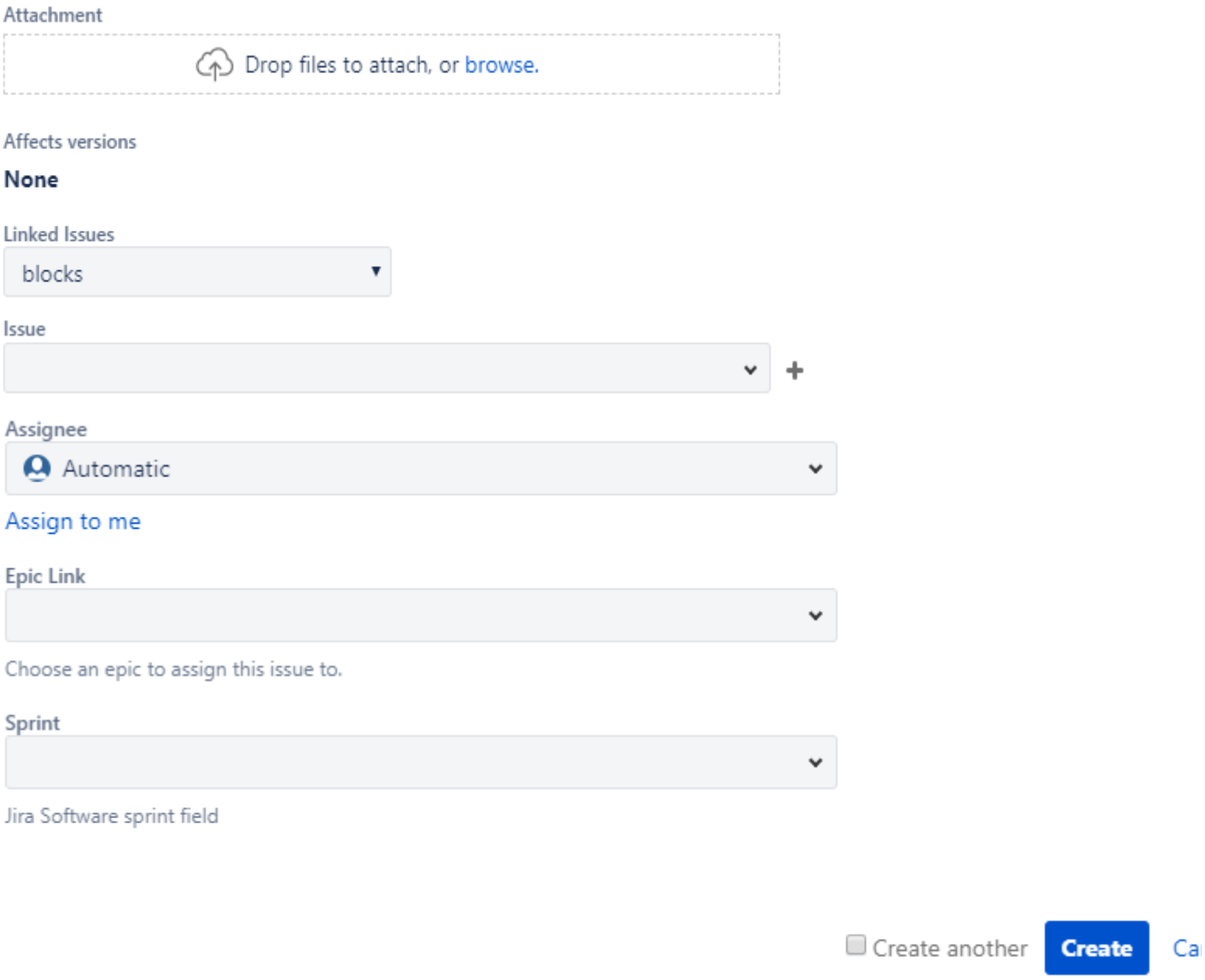
Medium

Labels

Begin typing to find and create labels or press down to select a suggested label.

Environment

Style | B | I | U | A | ³A | | | | | |

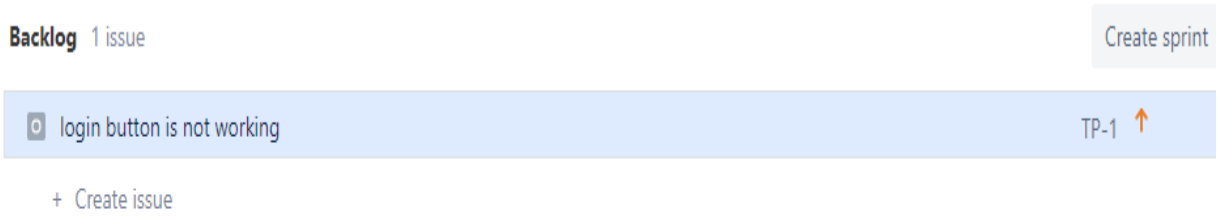


In the above form, fill all the mandatory details, and leave other fields with their blank or default values.

Mandatory fields are:

- **Project:** It defines the name of the project. Suppose I have given the name of the project as '**Test project**'.
- **Issue Type**
  It defines the type of issue. It can be a bug, epic, story, etc.
- **Summary**
  It tells the overall description of an issue.
- **Priority**
  It defines the priority of an issue. It can be either medium, low, or high.

Suppose I have created an issue whose type is Bug and description is "Login Button is not working", and priority of an issue is high.



In the above screen, we can see that the issue is created and it is placed in the Backlog.

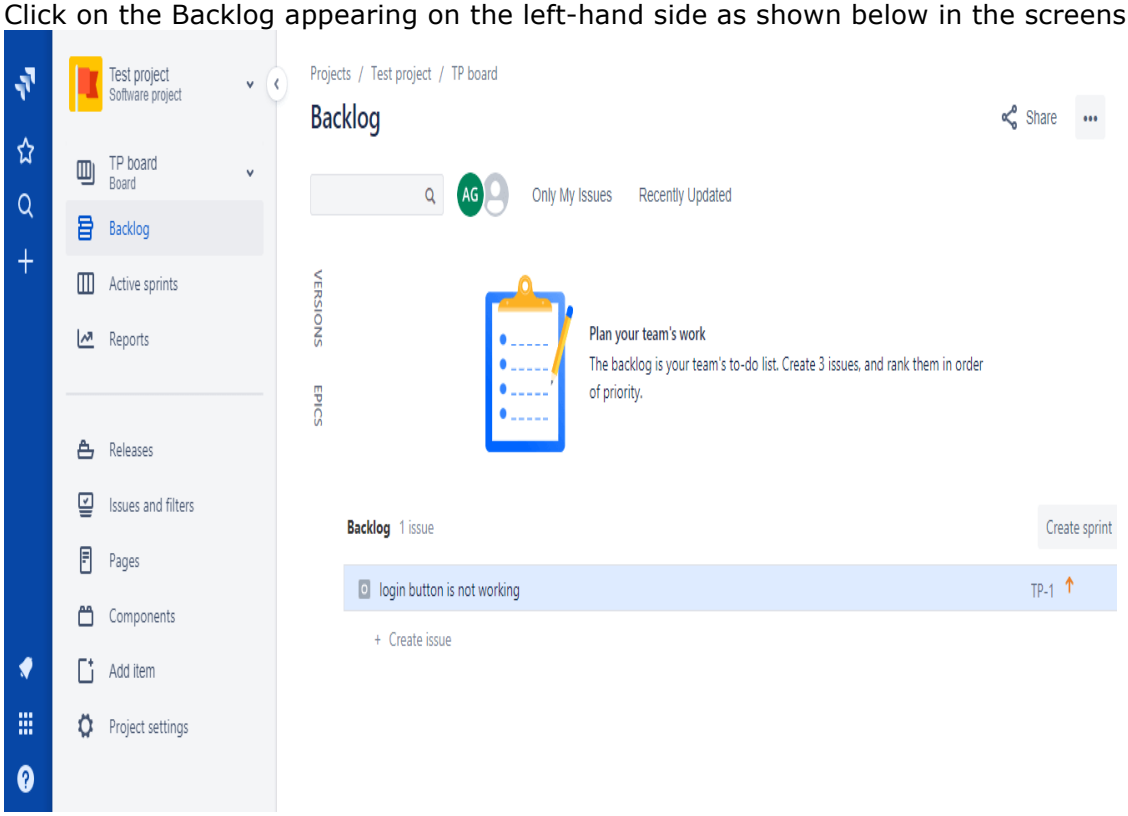
What is Sprint?

The team needs to complete a set of issues within a specified time period, known as a sprint. The duration of the sprint is decided by both the team and the product owner.

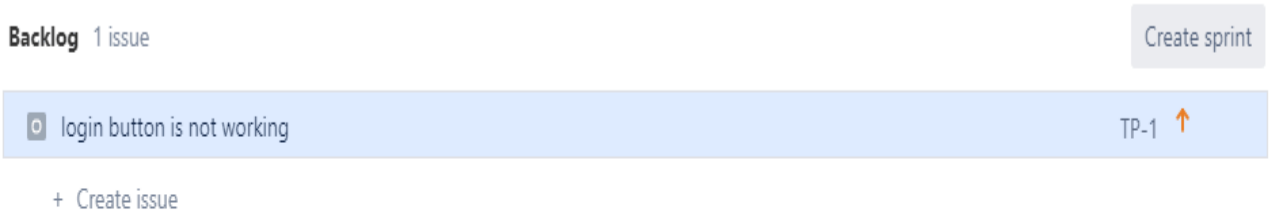
Creating a Sprint

Following are the steps to create a sprint:

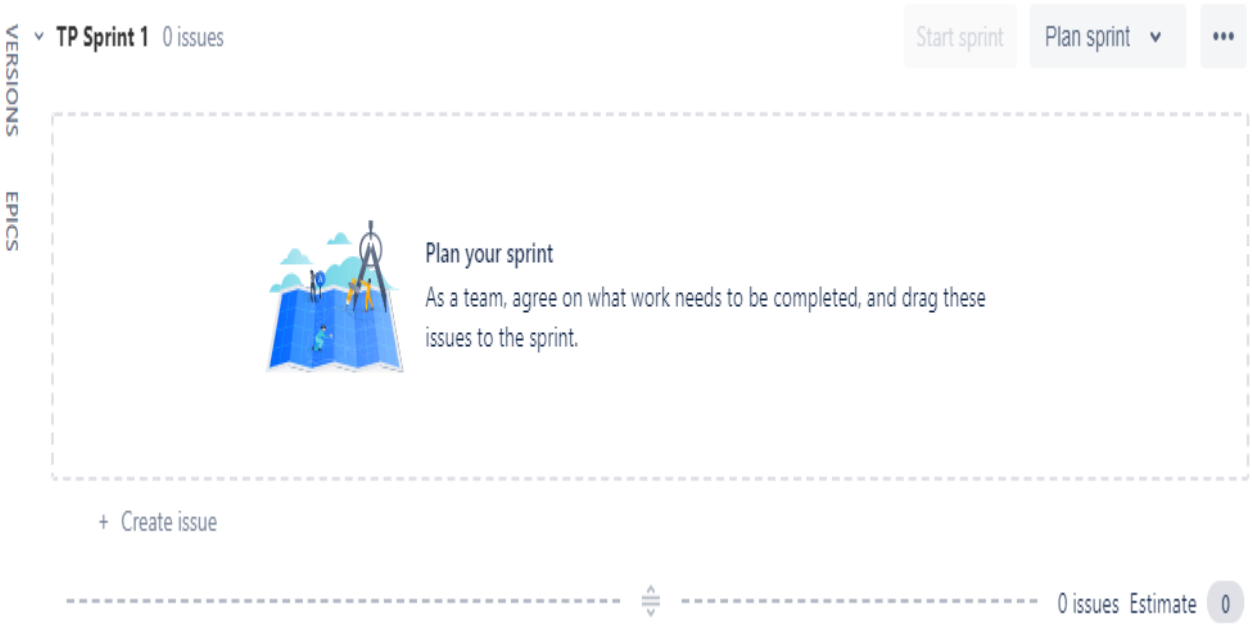
- Click on the Backlog appearing on the left-hand side as shown below in the screenshot:



- Click on the **Create sprint button** appearing on the right corner as shown in the below screenshot:



- After clicking on the **Create sprint button**, the screen appears, which is shown below:

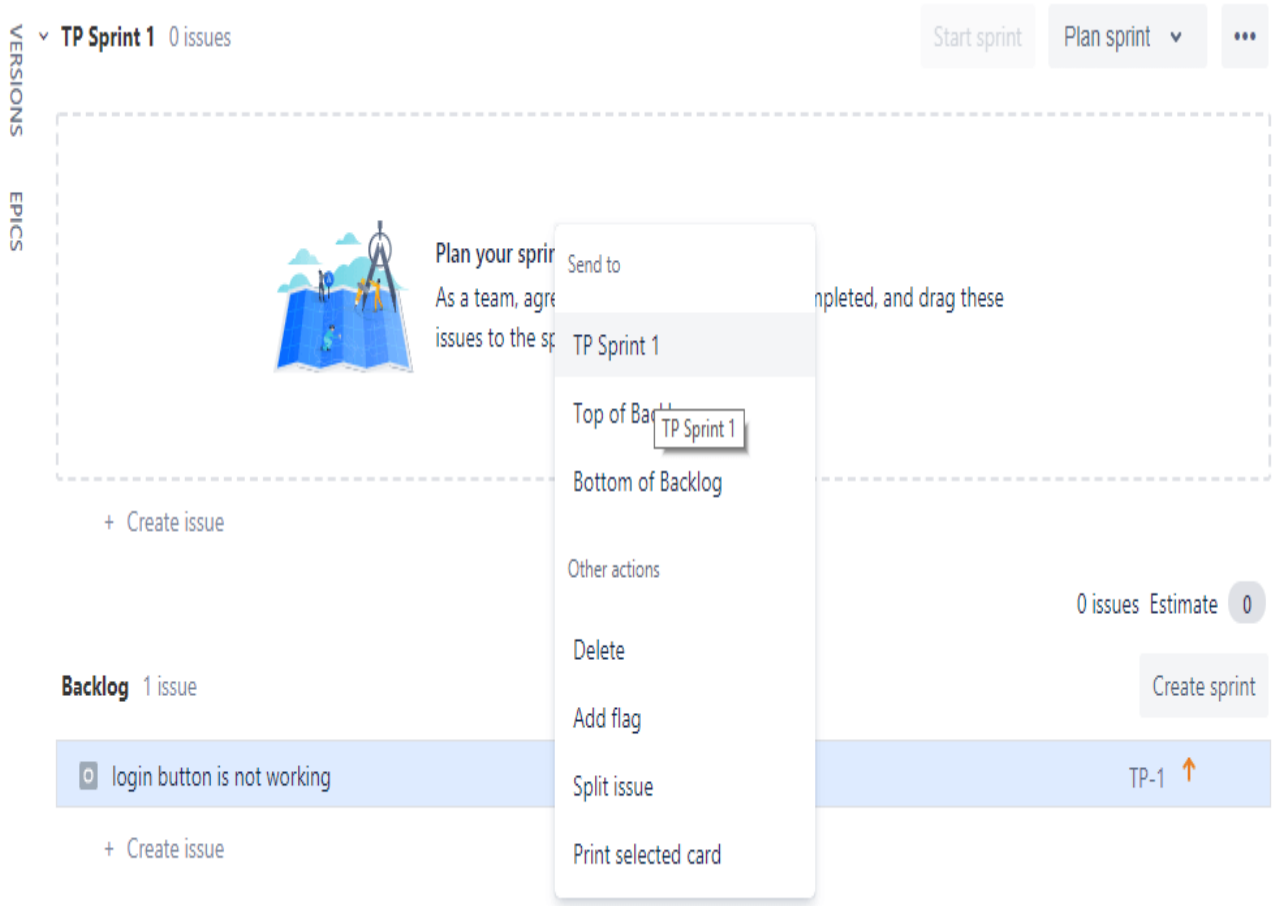


The above screen shows that the sprint has been created with a name "Sprint 1", now we can add the issues to this sprint.

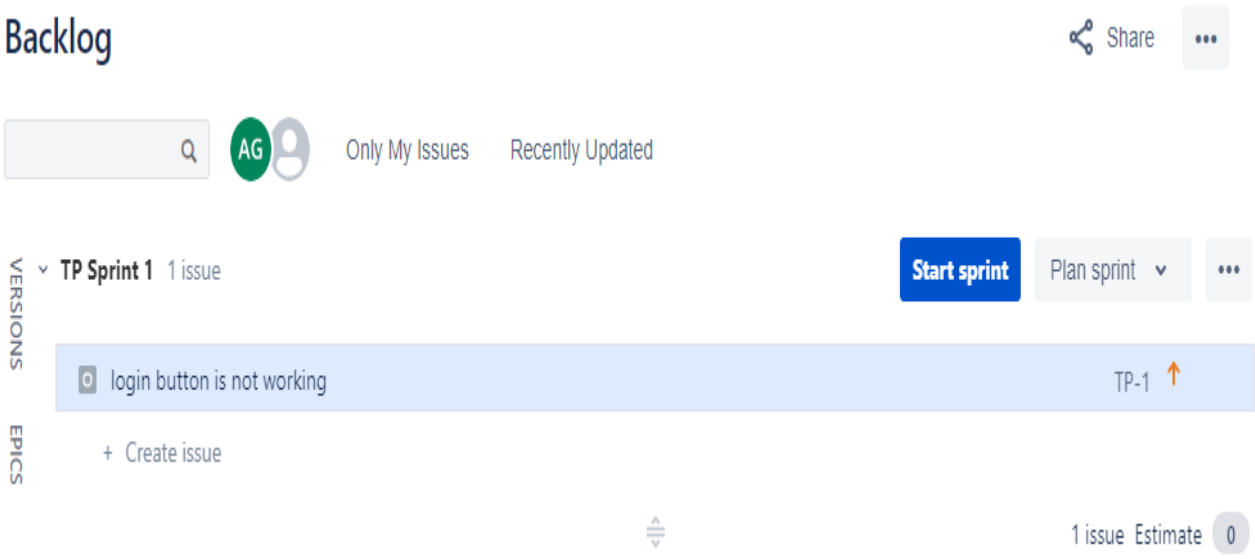
Adding an issue to a sprint

The following are the steps required to add an issue to a sprint:

- Click on the backlog.
- From the Backlog list, right-click on the issue and then select the sprint in which you want to add your issue. We have just now created the sprint named as Sprint 1, select the Sprint 1.



- The issue TP-1 is added to the sprint, i.e., Sprint 1.

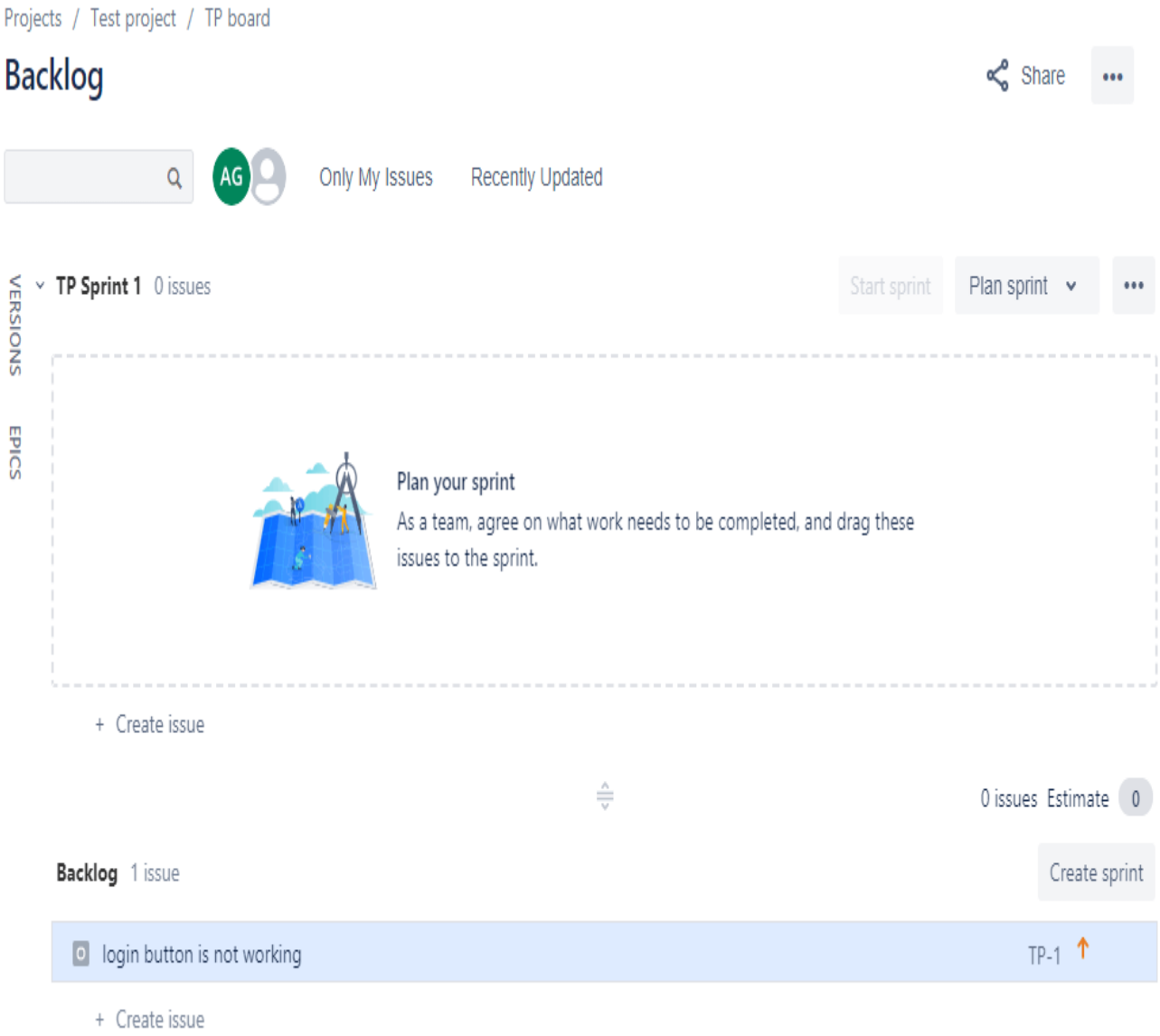


From the above screen, we observe that TP-1 issue is added to the Sprint 1.

Removing an issue from a sprint

The steps to be followed to remove an issue from a sprint:

- Click on the Backlog.
- Drag and drop the issue which is to be moved from the Sprint 1 back to the backlog list.

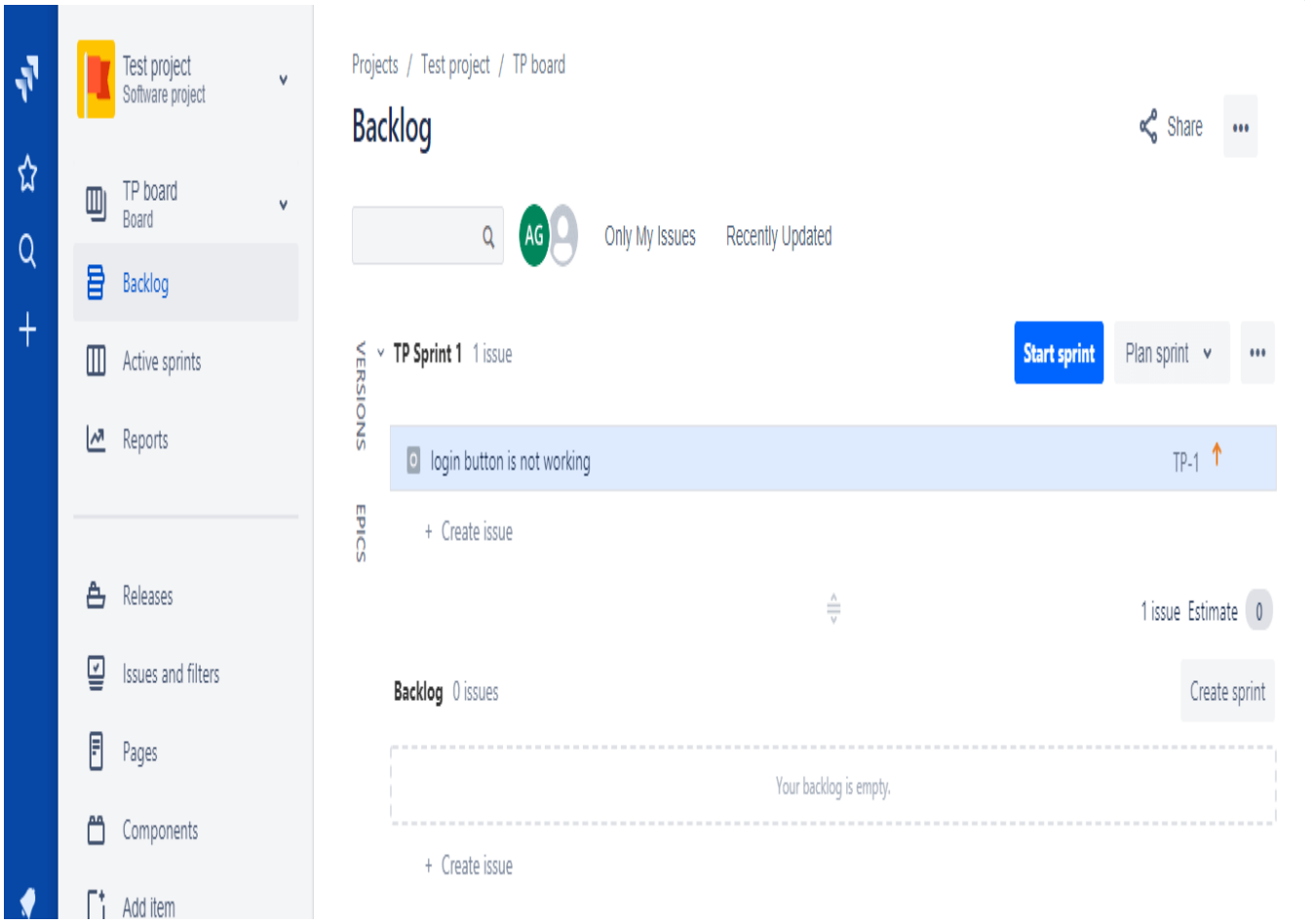


The above screen shows that the issue TP-1 is moved from the Sprint 1 to the Backlog.

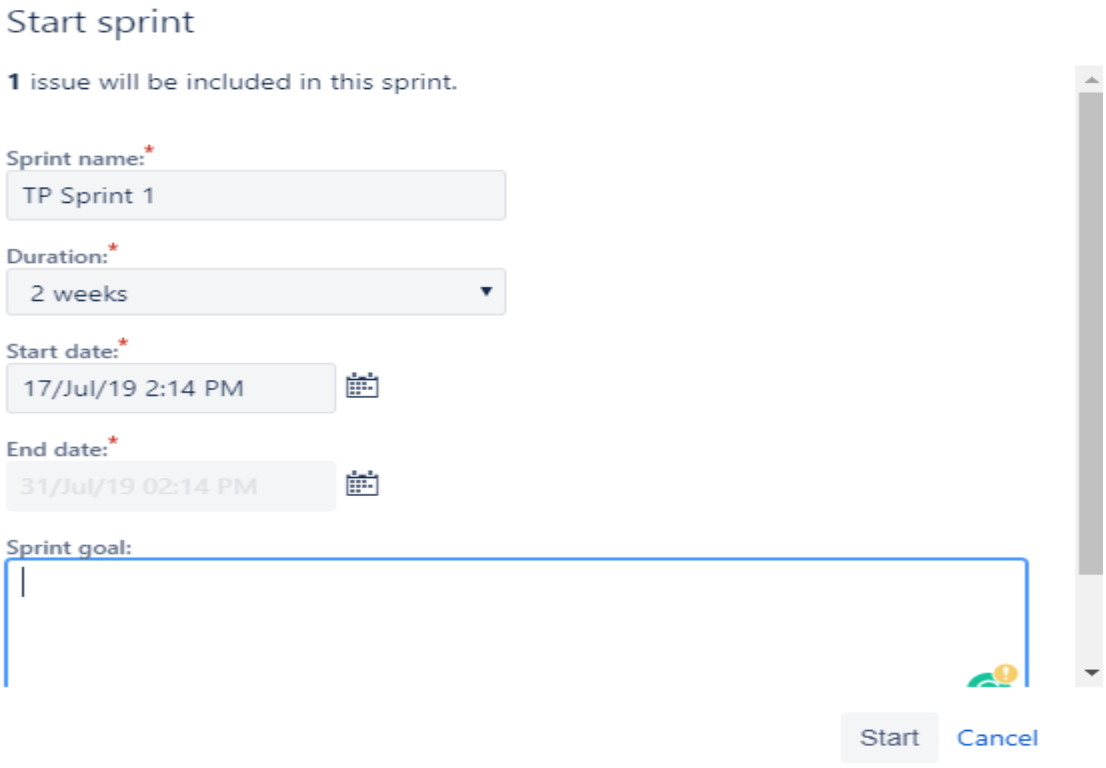
Starting a sprint

The steps to start a sprint:

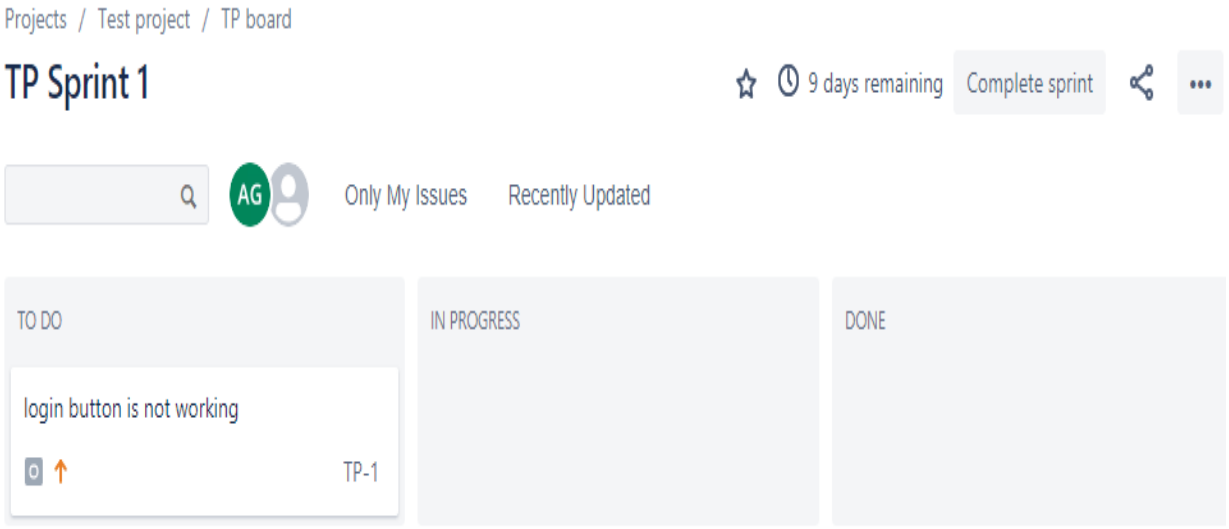
- Click on the Backlog.
- Click on the **start sprint** button to start a sprint.



- Fill the following details to start a sprint such as a sprint name, sprint duration, end date, start date, sprint goal.



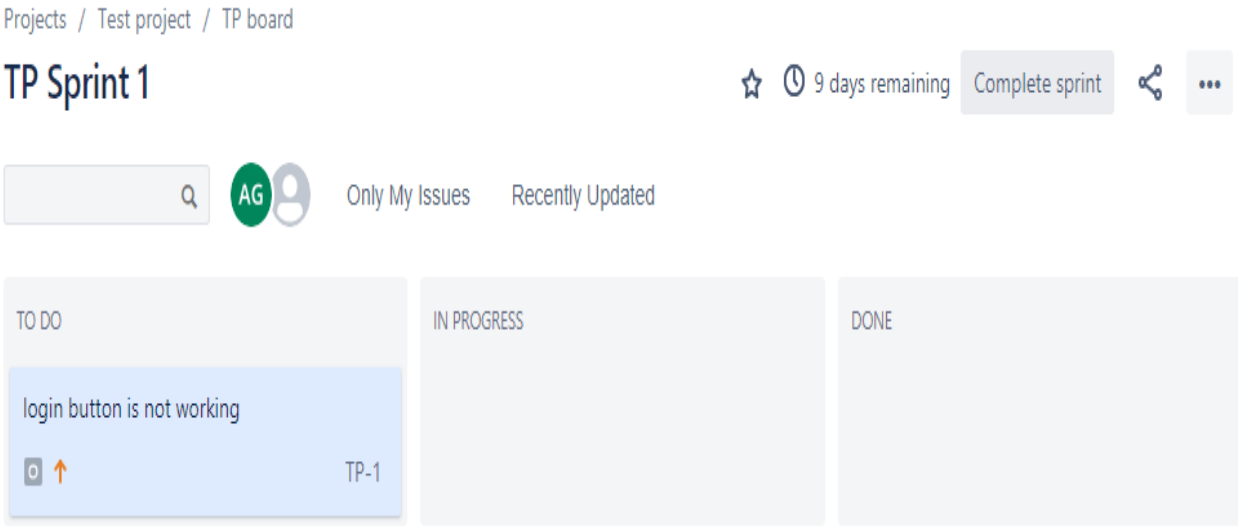
- Now, **Sprint 1** is started. Initially, the issue comes under the **TO DO** state.



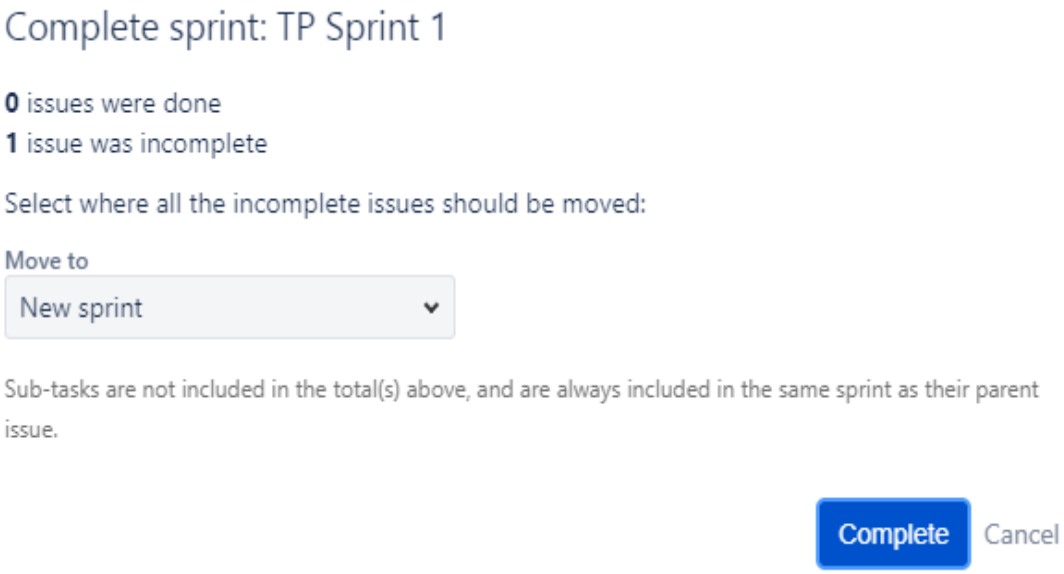
Ending a sprint

Steps to be followed to start a sprint:

- Click on the Backlog.
- Click on the **Active sprints** appearing on the left side of the project.
- Choose the sprint that you want to complete or end the sprint.
- Click on the **complete sprint button** appearing at the top right corner when all the issues have been completed.



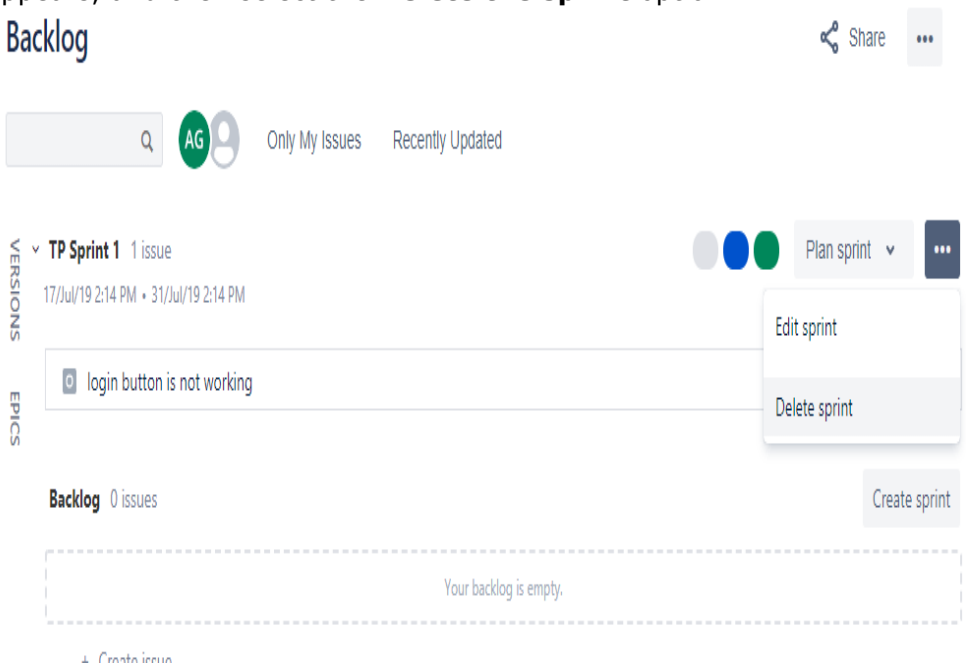
- It will show you whether all the issues have been completed or not, If not, then it asks you to move the issues to another sprint which are not yet completed.



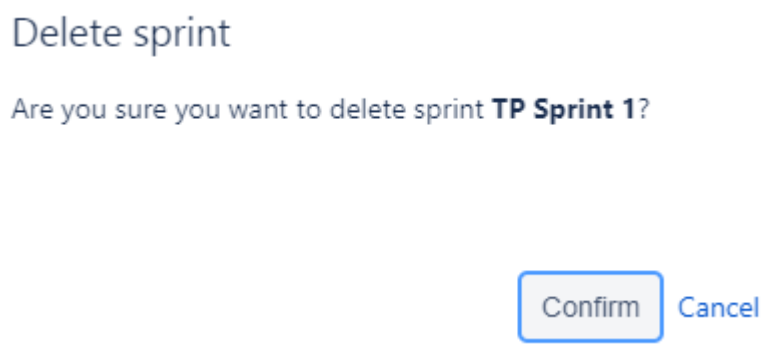
Deleting a sprint

Steps to be followed to delete a sprint:

- Click on the **Backlog**.
- Choose the sprint that you want to delete, and then click on the '...' icon. The dropdown appears, and then select the **Delete the sprint** option.



- On clicking on the **Delete sprint** option, click on the **Confirm** button.



Jira Scrum Board

Jira Scrum Board is a tool used to unite the teams to achieve a single goal and incremental-iterative delivery.

Functions of Scrum Board

- **Increase communication and transparency**
  Jira Scrum board is the single source through which all the work of a team can be accessed by a team member at any time. This increases communication among the team members and transparency.
- **Promote sprint planning and iterative development**
  The heart of the Scrum framework is a sprint, which is a fixed amount of time for teams to build a releasable product increment. Jira scrum board is mainly designed so that the teams can organize their work within the sprint timeframe.
- **Improve team focus and organization**
  Sometimes teams forget the project deadlines, and they are over-committed on their workload. Jira Scrum Boards provide transparency to the team's work by dividing the work into multiple stages and providing the burndown and velocity reports.

Following are the important terms related to a Scrum Board:

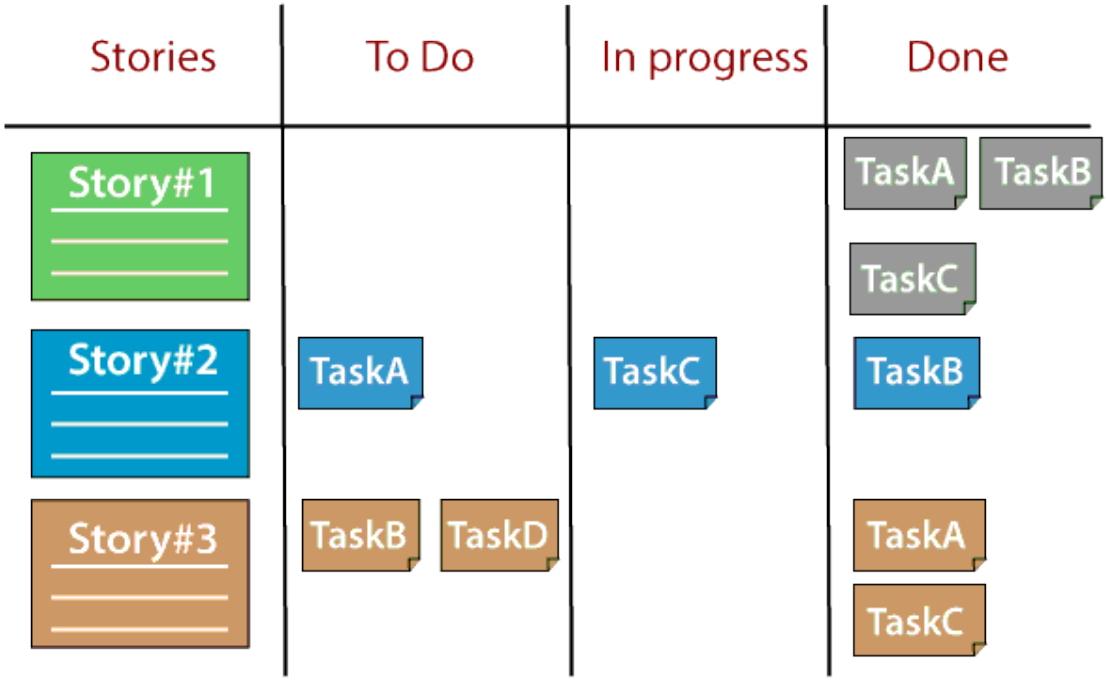
- **Sprint**
  A sprint is a time-boxed iteration of time which is mainly of 2 weeks. When the sprint is completed, then it produces the most valuable, useable, and releasable product in the market.
- **Backlog**
  The Backlog is owned by the Product owner. Product Backlog is a list of features, defects, enhancements, experiments that are to be considered in the product.
- **User story**
  - A user story is the smallest unit of work in an agile framework. A User story is a goal, not a feature which is expressed in terms of the user's perspective.

- The purpose of a user story is to show how a piece of work will deliver a particular value back to the customer.
- User stories are the sentences written in simple language to provide the desired outcome.
- It should be in a granular format so that it can be delivered within a sprint. Sprint is nothing but a time-bound iteration to deliver the working software.
- User story comes under the epic, and epic is a large chunk of work, user stories come under the epic. Some user stories are taken from the epic and placed in a particular sprint. This cycle goes on until and unless all the user stories within a sprint are completed.

- **Issue**
  An issue is also known as user story. In scrum board, an issue contains all the tasks, dependencies, and other relevant information required to perform a single piece of work.
- **Epic**
  Epic is a large chunk of work. It is basically a large user story which can be broken into a number of smaller stories. All the user stories in the epic might take several sprints to complete.
- **Swimlane**
  Swimlane is a means of categorizing of issues in an active sprint of a scrum board or a kanban board. It helps you to distinguish the tasks of different categories, such as users, workstreams, etc.

Layout of the Scrum Board

The Scrum Board is a physical board on which the user stories present in the current sprint backlog are displayed.



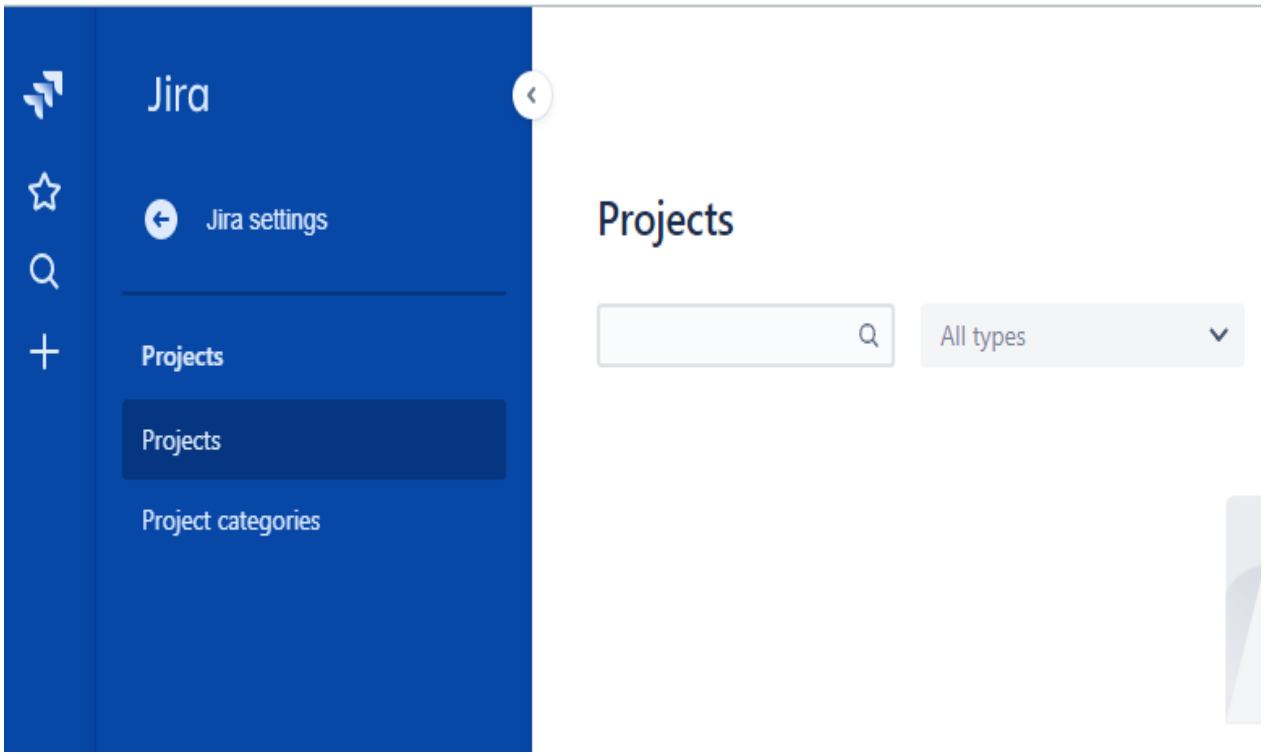
Scrum board is divided into columns such as:

- **Stories:** This column contains all the user stories available in the current sprint backlog.

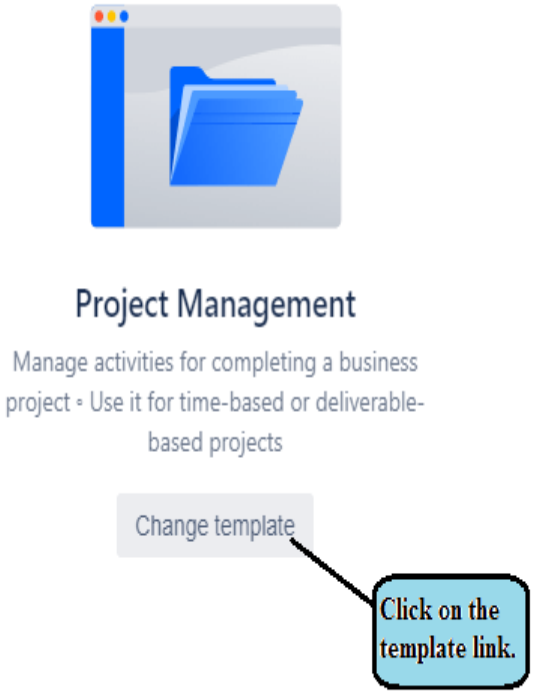
- **TODO:** This state contains the subtasks of the stories on which the work has not started on.
- **In Progress:** This state contains all the tasks on which the work has started.
- **Done:** This state contains all the tasks on which the work have been completed.

Steps to create a Scrum project

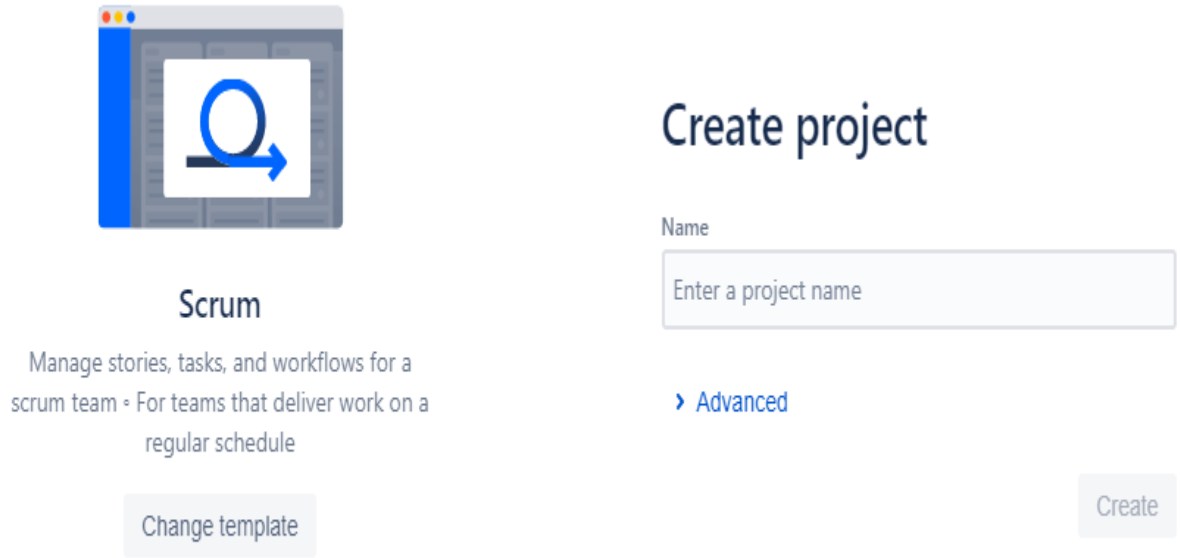
- Click on the **projects** appearing on the left side of the page.



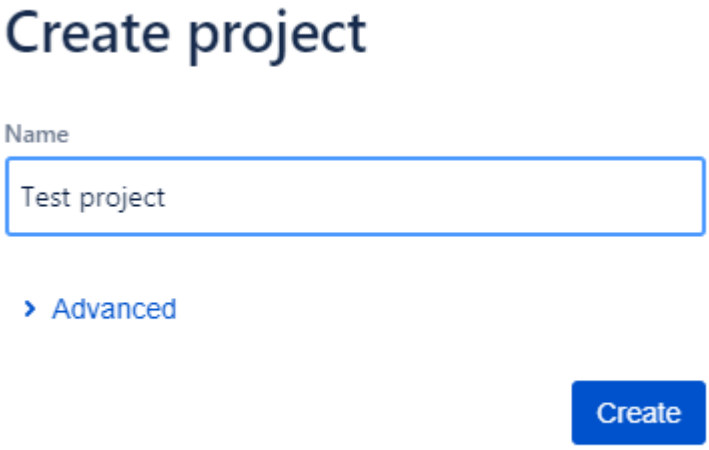
- Click on the **Create Project** button to create a new project appearing at the top right corner of the page.
- Click on the template link.



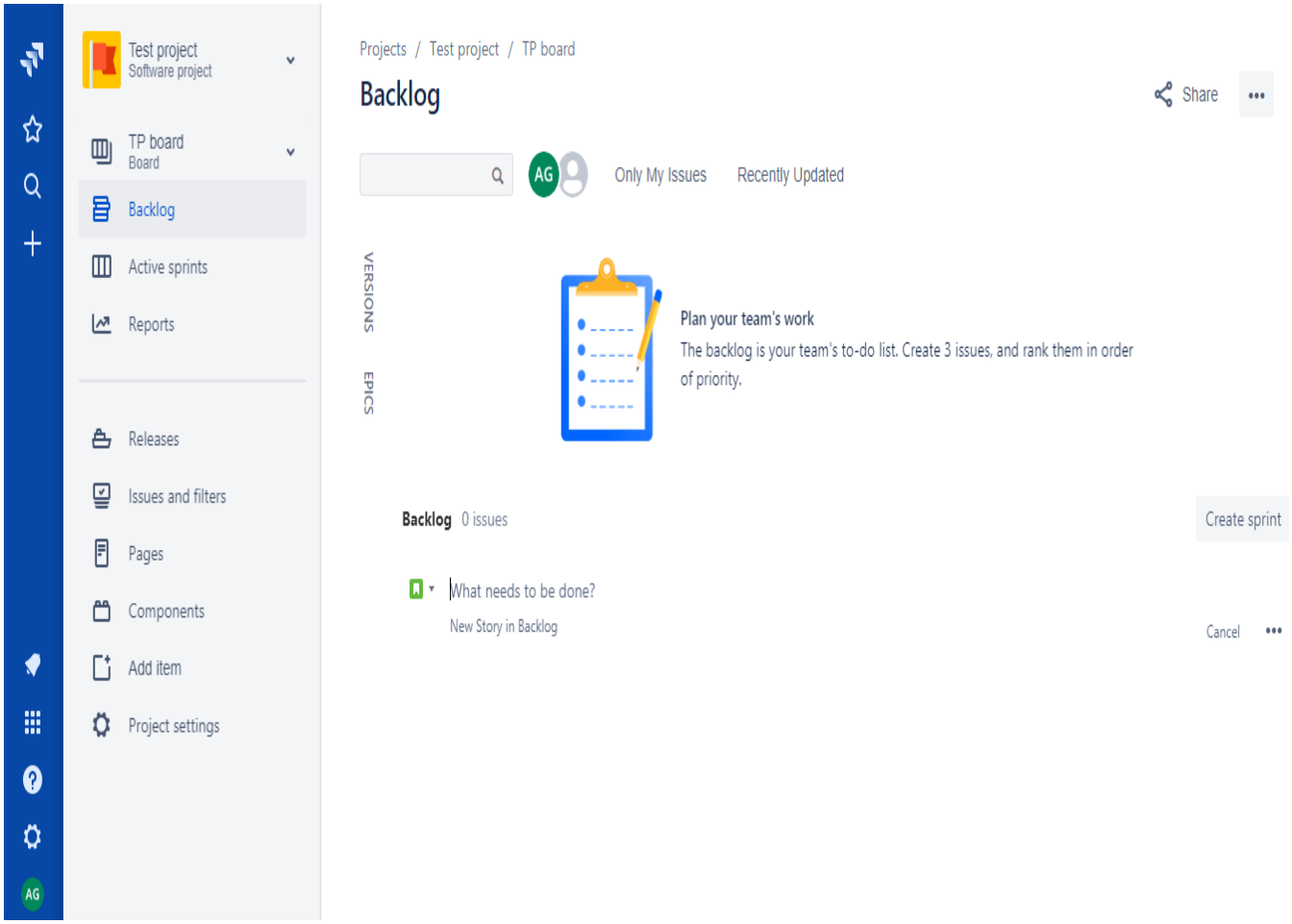
- Select the scrum project from the template.



- Enter the project name.



- In the below image, TP Board is created, which is the short name for the project named as "**Test project**".



The whiteboard shown in the above image is the Jira board, which consists of issues related to a particular project.

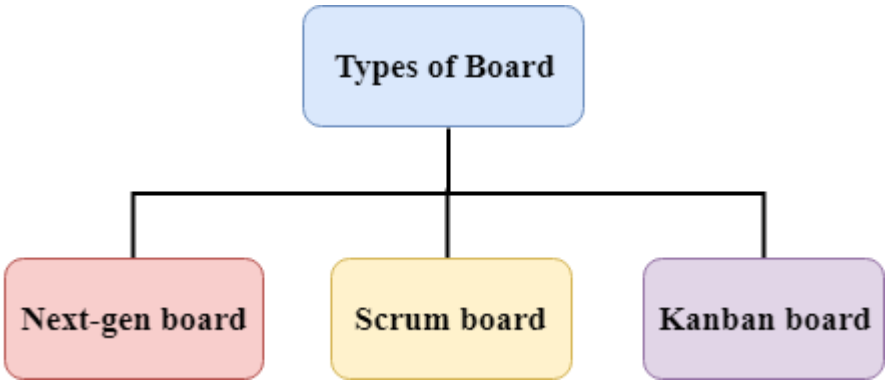
Displaying the issues on a scrum board provides the flexibility for viewing, managing, and tracking the progress of an issue.

The backlog is empty for the newly created project. It is filled only when you add the tasks or an issue in the Backlog.

What is Jira board?

A board displays all the issues that occurs within the project, providing you a flexible way of viewing, managing, and reporting the progress on work.

There are three types of boards that exist in Jira:

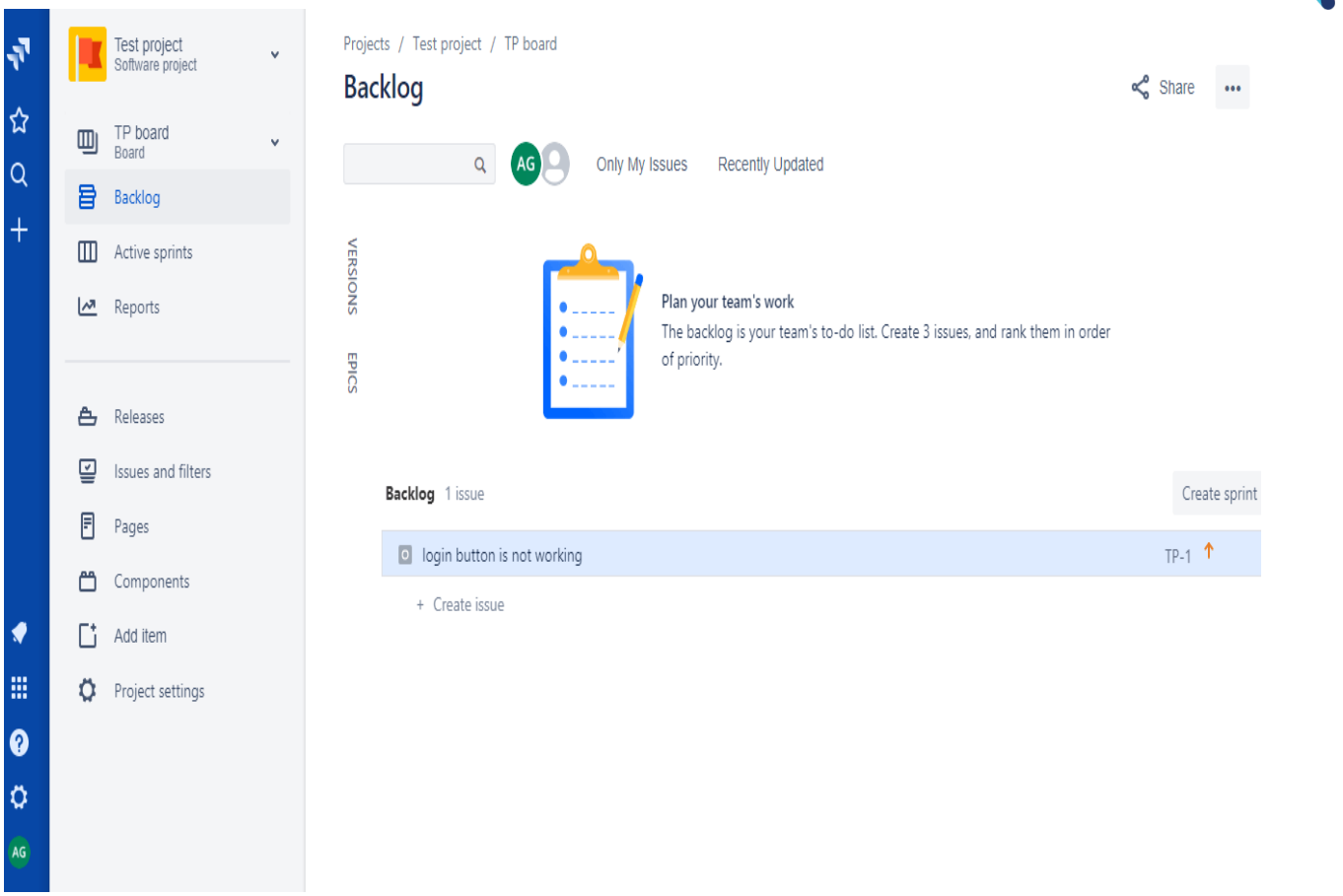


- **Next-gen board**
  This board is useful for those who are new to the agile. It is a very simplified, straight-forward, and streamlined board.
- **Scrum board**
  This board is useful when teams work on sprints that includes a backlog.
- **Kanban board**
  Kanban board is an agile project management tool designed for the visualization of work, limit work-in-progress, and maximize efficiency.

Scrum board

Scrum board is divided into three parts:

- **Backlog**
  The backlog is a set of activities or issues available in a project. All the issues are grouped in a backlog and sprint. In scrum backlog, you can create and update the issue, drag and drop issue and assign them to sprints, drag, and drop the issue to rank them.

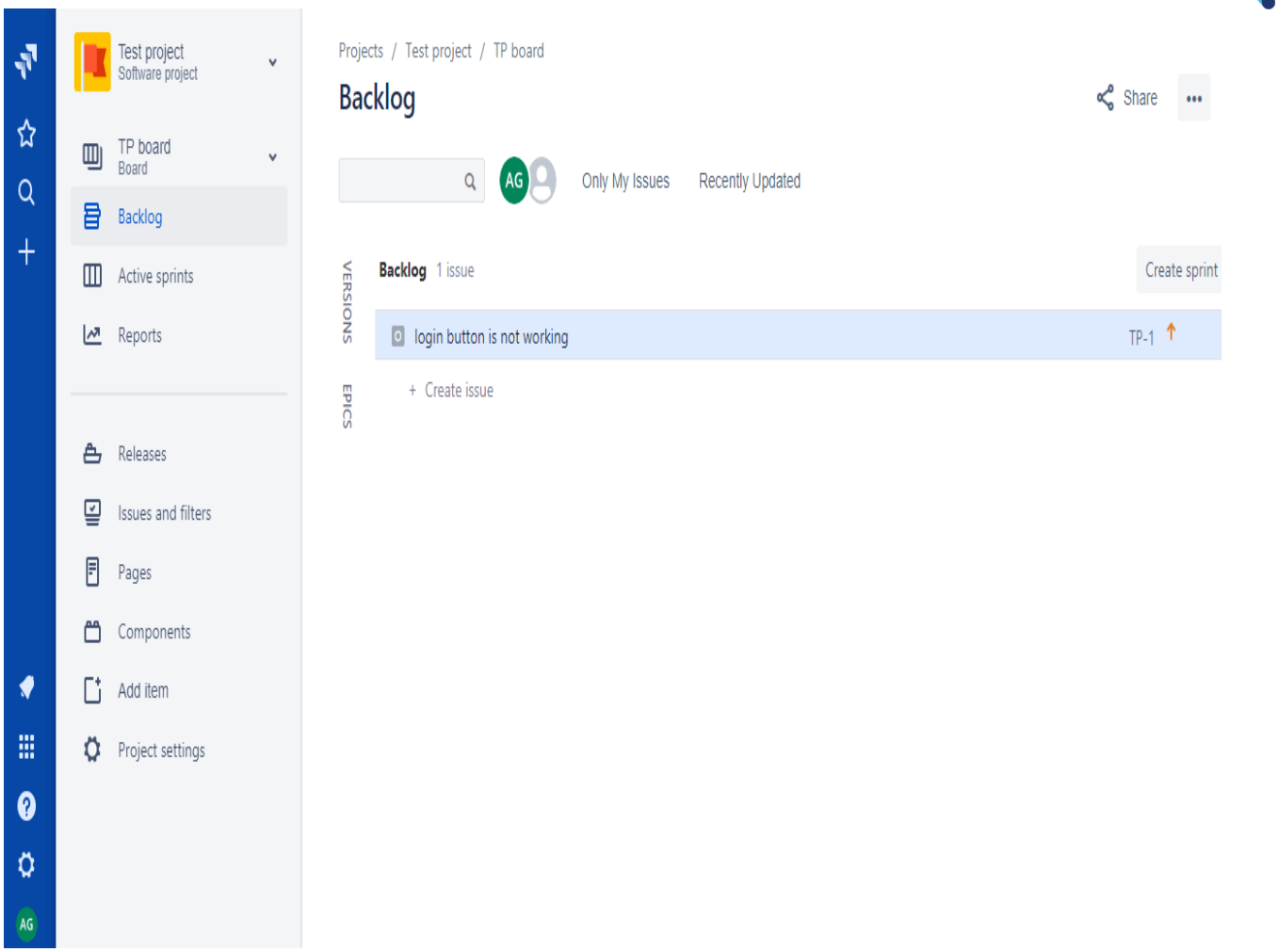


The above screen shows the backlog which contains an issue **TP-1** whose description is that **login button is not working**.

- **Active sprint**
  Active sprint is a sprint that contains the issues on which the team is currently working on. You can also drag and drop the issue and assign it back to the backlog.

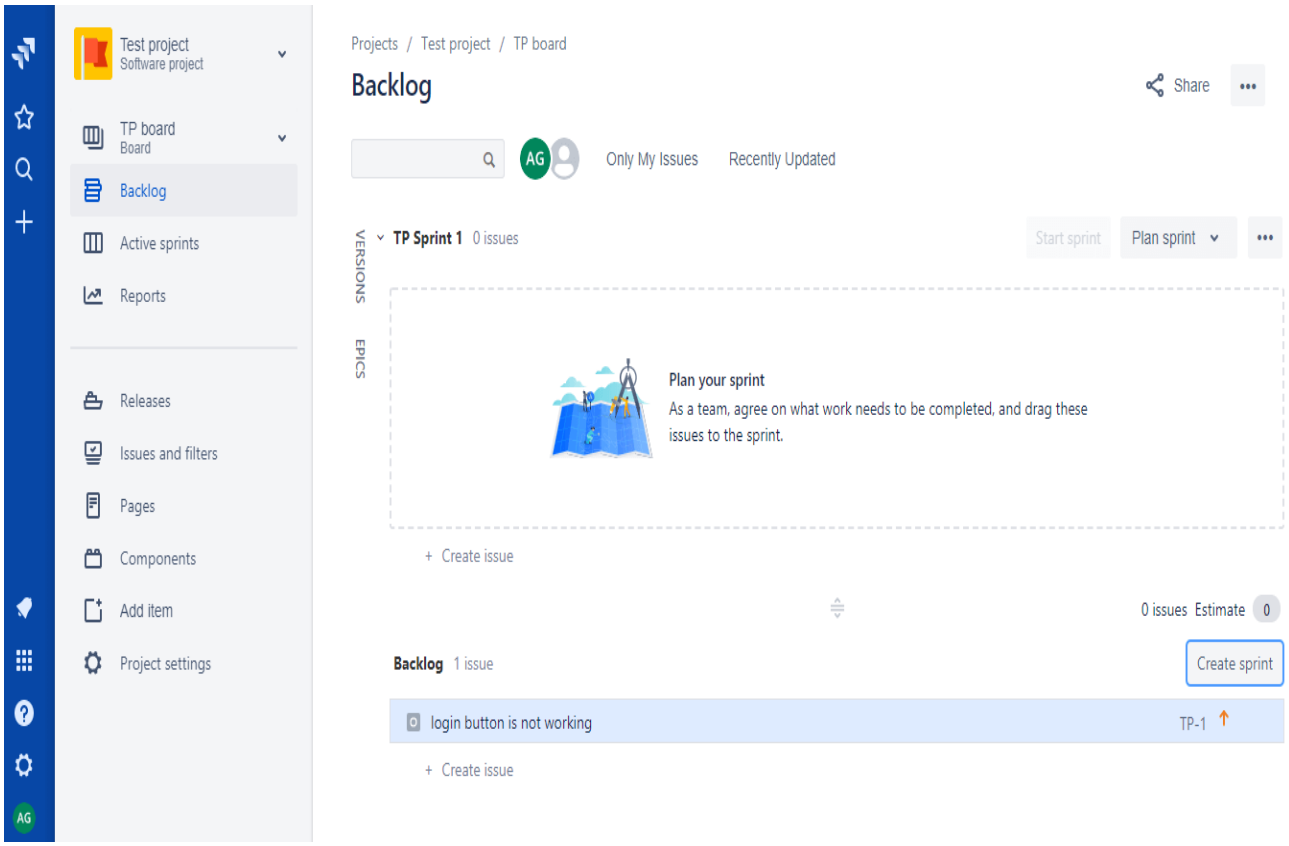
Steps to create an active sprint:

- Click on the Backlog. After clicking on the Backlog, the screen appears, which is shown below:



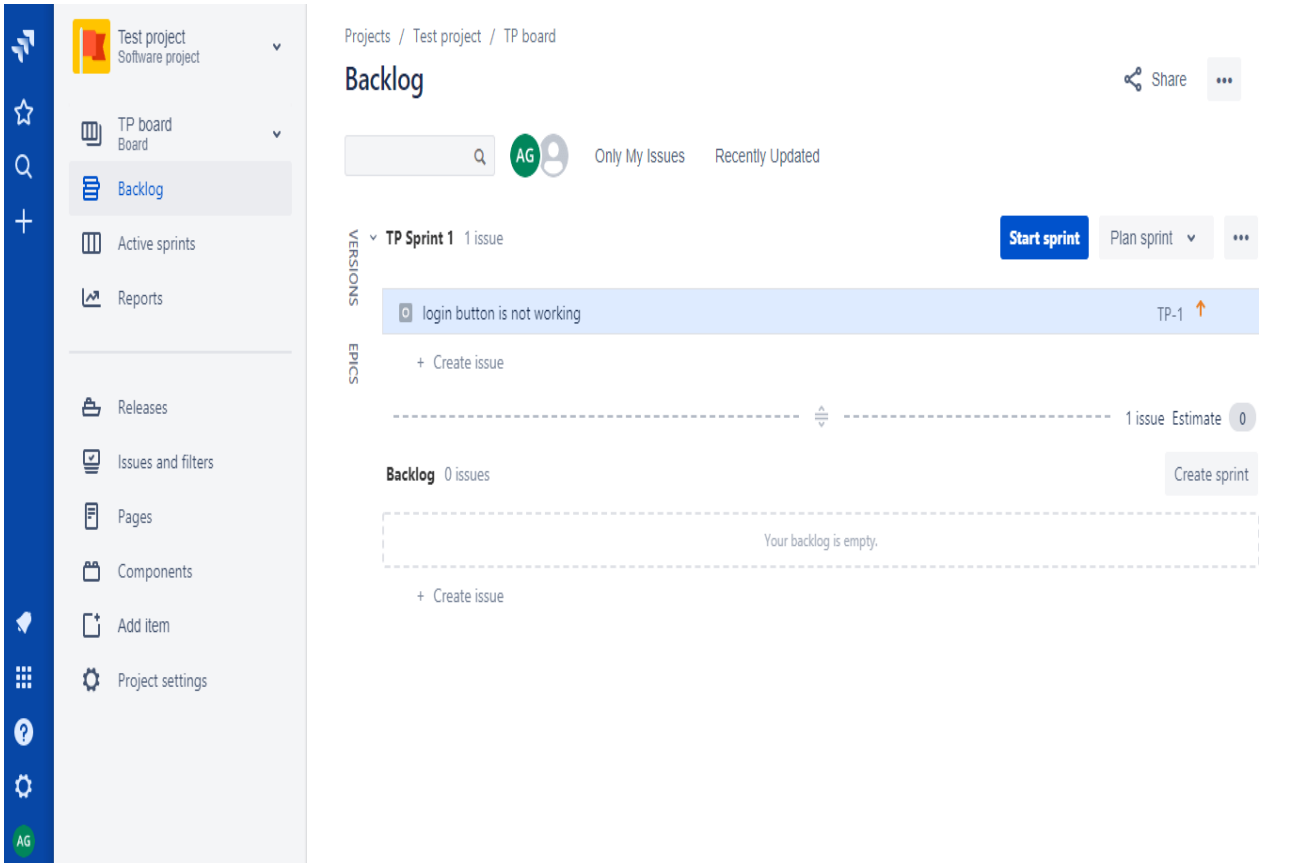
The above screen shows the Backlog which contains an issue "**Login button is not working**".

- Click on the **Create sprint** button to create a sprint. On clicking on the Create sprint button, the screen appears, which is shown below:



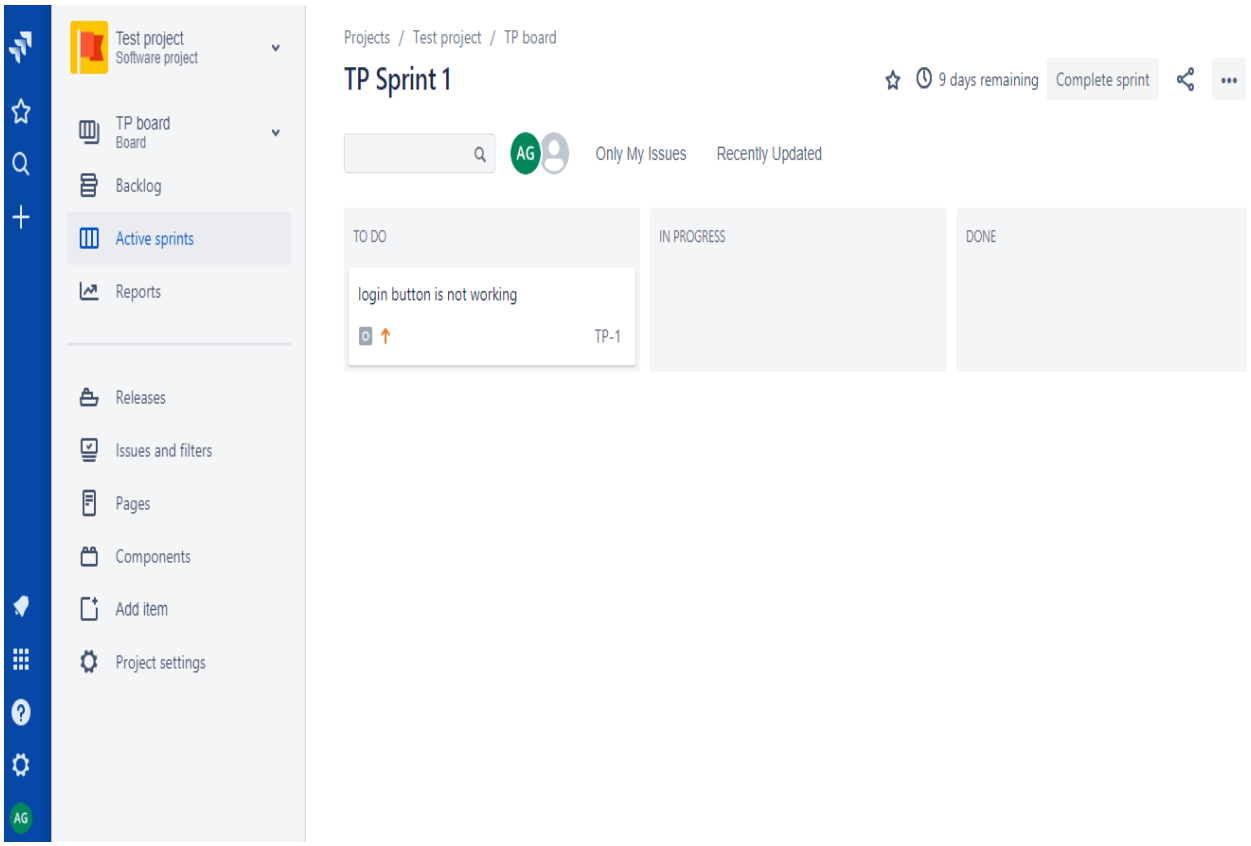
The above screen shows that sprint is created named as **Sprint 1**.

- Drag and drop the issue from backlog to Sprint 1.



In the above screen, we observe that the issue has been dragged from the backlog to Sprint 1, and the issue is removed from the backlog.

- Click on the **start sprint** button to make an active sprint.
- Click on the **Active sprint** appearing on the left-side of the panel.



The above screen shows that Sprint 1 is an active sprint on which the team is currently working on.

- **Reports**
  Reporting is an activity that will be performed throughout the project. Reports are made which are used to show the information about your project, versions, epic, sprints, and issues.

Kanban board

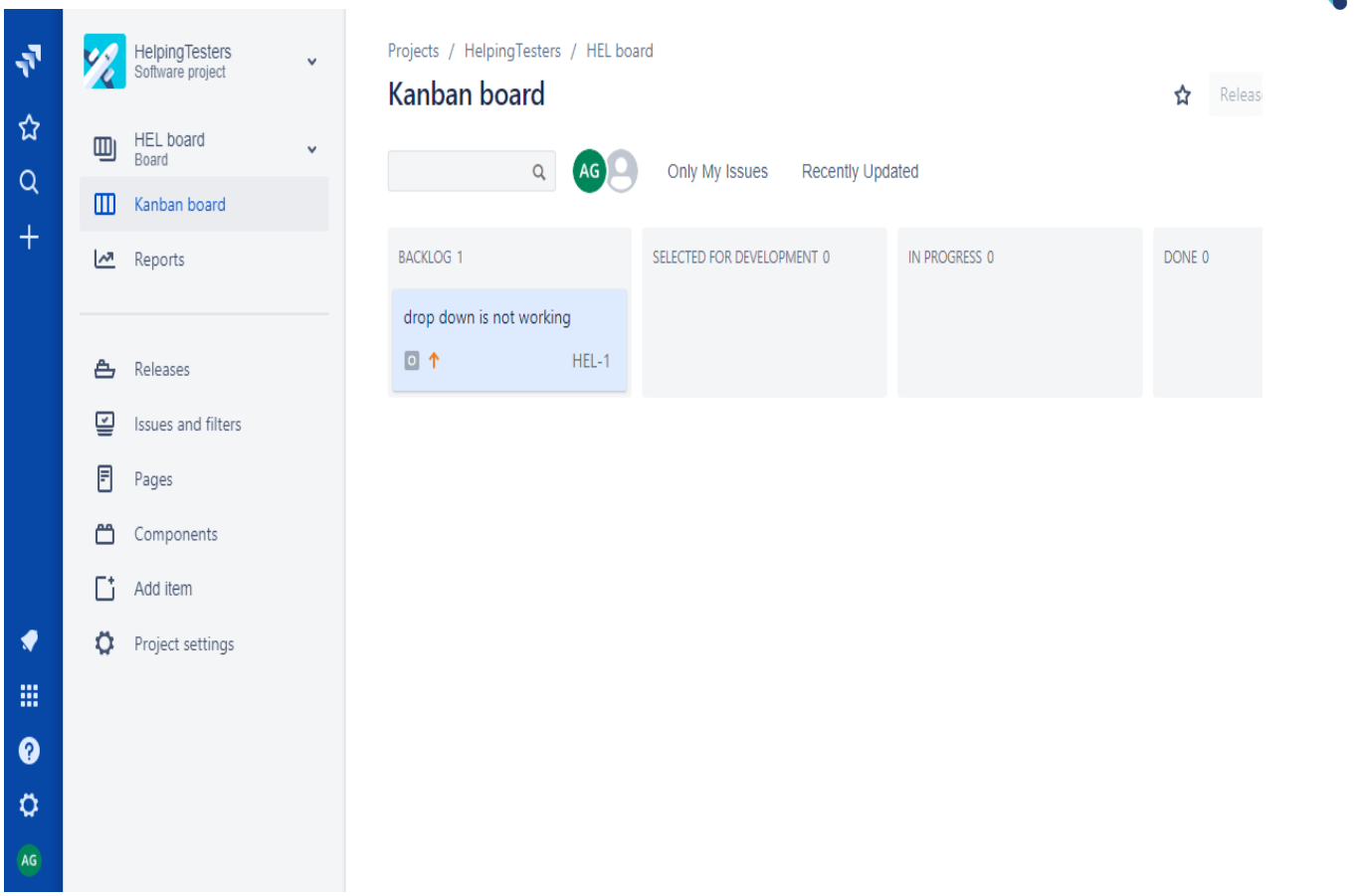
Kanban is a process that provides the visualization of all the work which you are doing today.

Kanban consists of three parts:

- **Backlog**
  Kanban has a separate column of Backlog from where we plan the work for our team. Planning of work in a small column becomes very difficult, so by using the kanban backlog that provides a bigger space for handling the issues.

Steps to create a backlog in the kanban framework:

- Create an issue. When we create an issue, then it is added to the Backlog automatically.



The above screen shows that HEL-1 issue that we created is added to the backlog.

- **Kanban board**
  Kanban is a process based on the continuous delivery of work. In Kanban framework, rather than planning the iterations, the flow of work is continuously monitored means that when the tasks are completed, then the new task is added in **In Progress** state.
  Kanban board consists of three states:
  - **Selected for development**
  - **In Progress**
  - **Done**

Reports

Reporting is an activity that will be performed throughout the project. Reports are made which are used to show the information about your project, versions, epic, sprints, and issues.

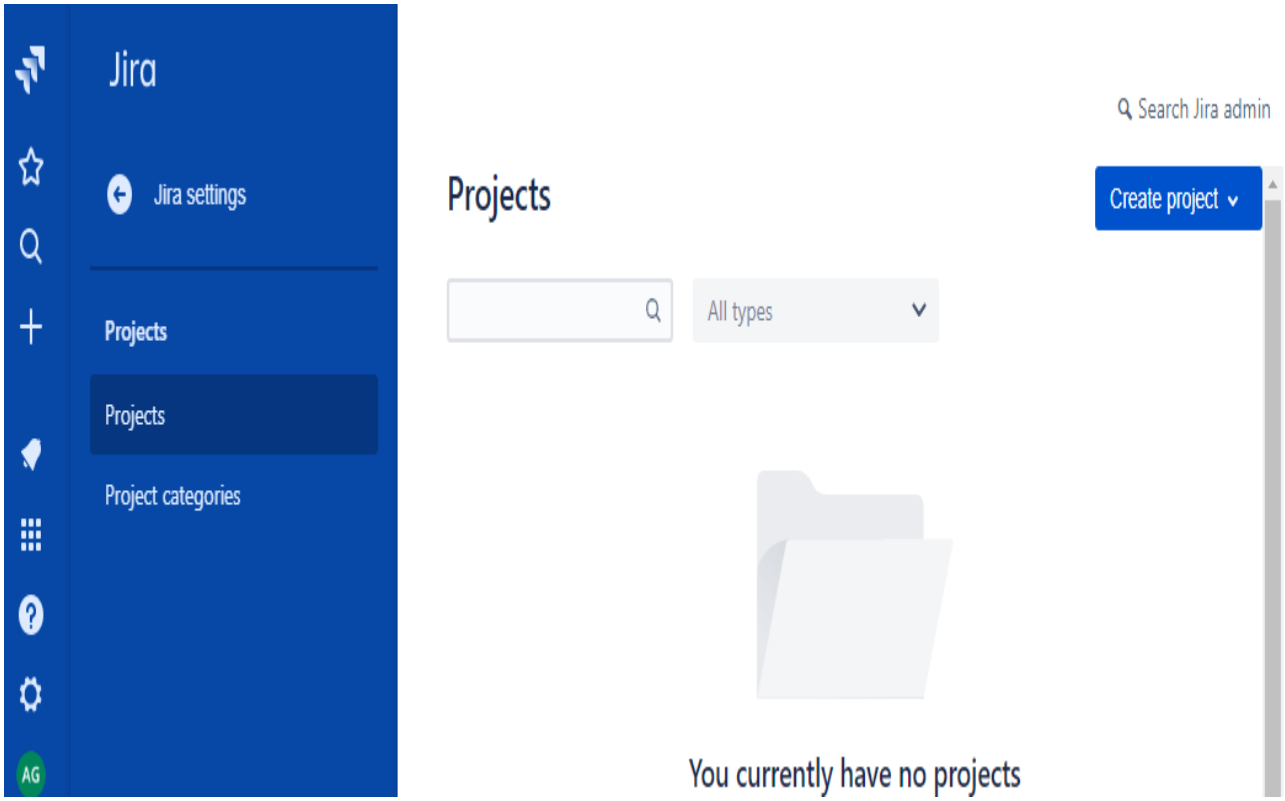
Next-gen board

Next-gen board is the same as classic scrum and kanban board. However, next-gen does not provide all the features in the beginning, it provides the feature only when you need. The Next-gen project can be done in two templates, i.e., Next-gen Scrum and Next-gen Kanban.

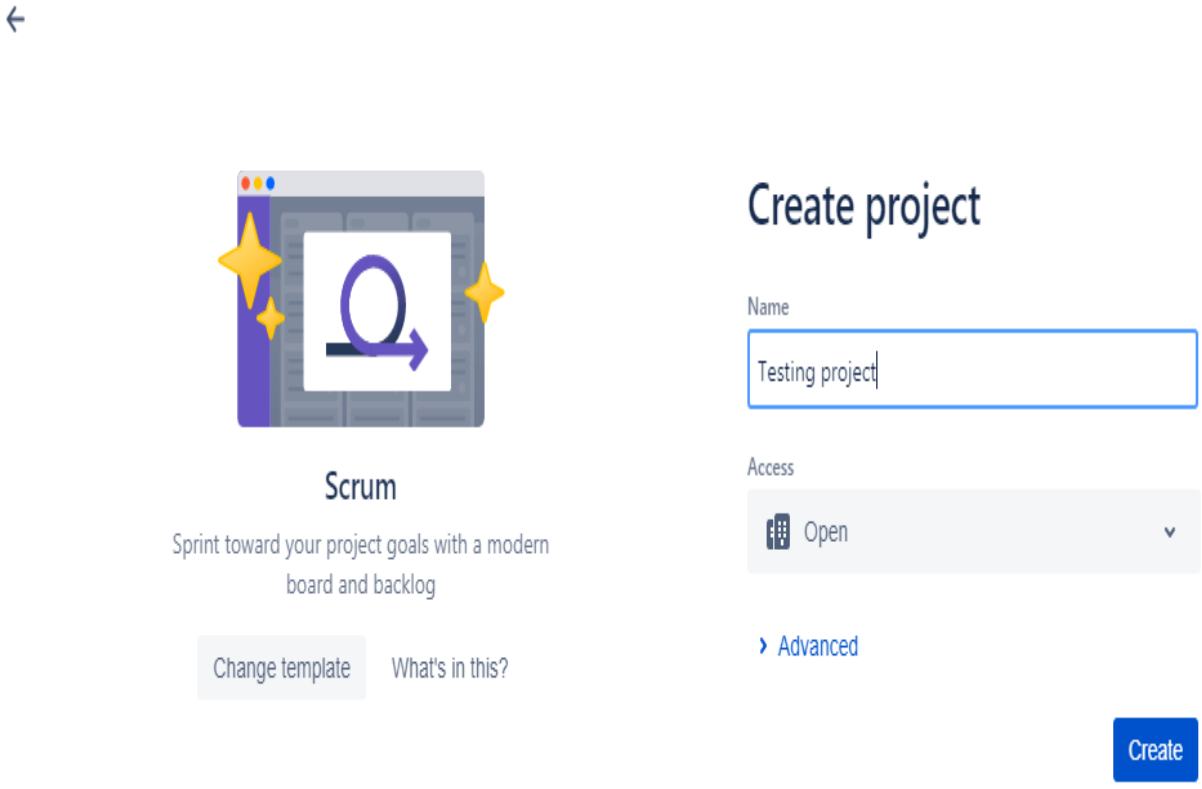
How to create a Next-gen project

Next-gen project can be created in the following steps:

- Click on the **Create project** drop-down menu and then select the **Try next-gen project** option.

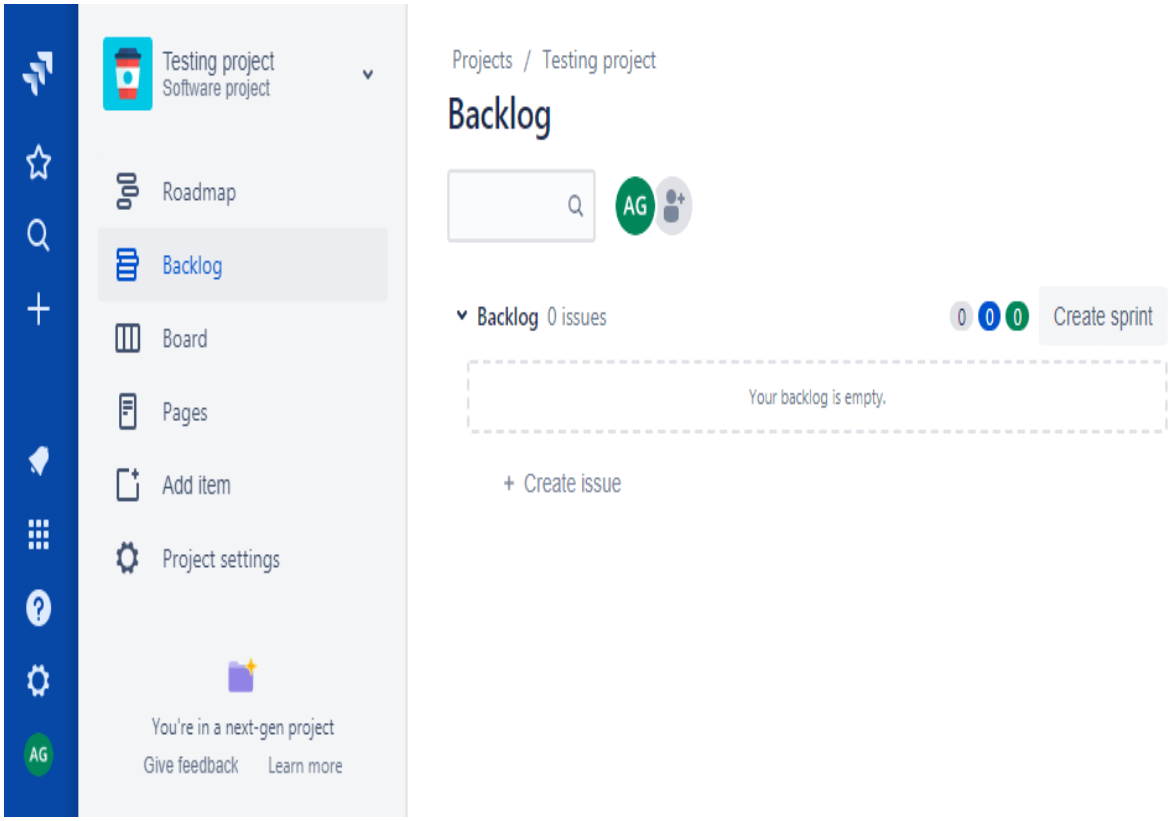


- After clicking on the Try next-gen project, the screen appears, which is shown below:



The above screen shows that we have chosen the Scrum template, we can also change the template by clicking on the link Change template. After selecting the template, enter the project name, we have given the Testing project as a project name. Click on the Create button.

- After clicking on the **create** button, Next-gen Scrum project is created, which is shown below:

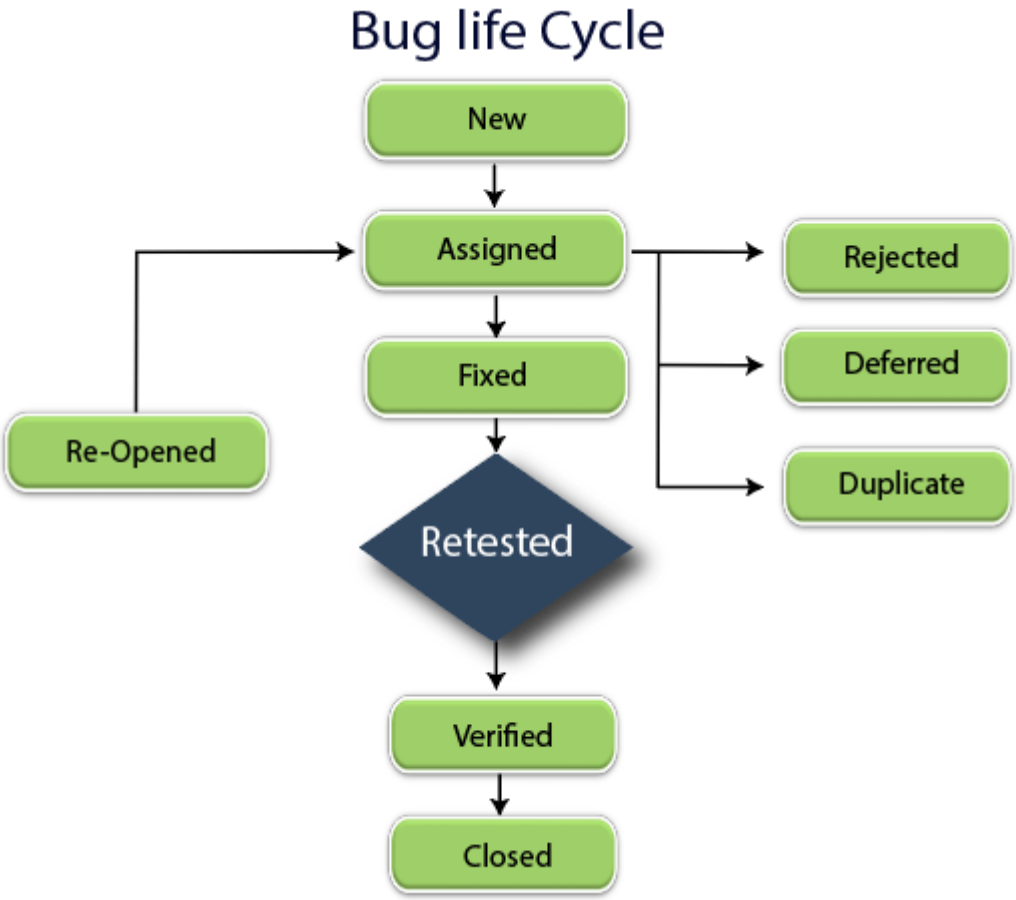


JIRA Bug Life Cycle

JIRA bug life cycle is also known as a defect life cycle. The bug life cycle consists of a set of states that a bug goes through. The number of states that the bug goes through varies from project to project. We can define the bug as an error, flaw or we can say that when the actual output does not match with the expected output, it is known as bug or defect. Both the terms, i.e., bug and defect are commonly used but the most popular is a bug. A bug can be generated at any stage of the **SDLC**(Software Development Life Cycle), it could exist in requirements gathering, designing phase where SRS document is designed, development phase, testing phase or user acceptance testing done by the end-user at the time of using the application.

A bug has its life cycle from the point when the bug is logged in to the point the bug is closed. Bug undergoes the following states:

- New
- Assigned
- Open
- Fixed
- Retesting
- Reopen
- Verified
- Closed



New

During the time of the testing phase, the bug or defect is identified by the tester and it is logged in to the bug tracking tool such as Jira, Bugzilla, etc. The bug which is detected by the tester will be posted for the first time in a bug tracking tool. This status is assigned as a **New** status.

Assigned

Bug with **New** status is assigned to the software developers and they will look into the bug to check whether the bug is valid or invalid. If the bug is invalid then they change the status to invalid. If the bug is valid then the status is changed to **assigned** then the software developers start working on the defect to get fixed.

Open

When the bug is assigned to the software developers then they start analyzing on it and works on the defect fix. The bug or defect can be opened in three stages:

- **Duplicate**
  If the defect is repeated twice or the defect corresponds to the same concept of the previous bug, then it changes the status to Duplicate.
- **Rejected**
  If the developer feels that the defect is not a genuine defect, then it changes the status to Rejected.

- **Deferred**
  If the bug is not of higher priority and can be solved in the next release, then the status changes to Deferred. The deferred state is also known as postpone state.

Fixed

When a developer makes a necessary code changes and verifies the change, then he/she can make the bug status as fixed. When the bug is fixed by the developers then the status is changed to either Reopened or Verified.

Retest

Once the bug is fixed by the software developers then it is assigned back to the testing team to check whether the bug has been fixed or not.

Reopen

If the bug persists even after the developer has fixed the bug, then tester changes the status to Reopen and once again bug goes through the whole bug life cycle.

Verified

The tester retests the bug after it got fixed by the developer if no bug found then it changes the status to Verified.

Closed

If the bug is no longer exists, then it changes the status to Closed.

Participants of Bug life cycle

- **Bug Reporter**
  The person who identifies the bug is known as Bug Reporter. Bug reporter validates the bug and enters all the bug related details into the bug tracking tool such as correct subject, bug priority, application component, test environment, bug assignee, bug description. When required, the tester needs to send the attached screenshot to clarify the bug details.
- **Bug tracking tool**
  A bug can be logged in to the bug tracking tool and bug tracking tool can be Jira, Bugzilla, Assembla, etc.
- **Bug Group**
  Bug Group is a group of people who can see the bug details. Bug Group can include tester or end user who reported the bug, the developers to whom the bug is assigned, project manager, QA manager.
- **Bug Owner**
  Bug owner is the person who reviews and owns the bug. Bug owner checks whether the bug information is sufficient or not, if not then the bug is assigned back to the bug reporter to provide more information. Based on the priorities given to the bug, Bug owner takes the ownership of the bug and get fixed it within the deadline.

Introduction to JQL (Jira Query Language)

JQL stands for Jira Query Language and is the most powerful and flexible way to search for your issues in Jira. JQL is for everyone: developers, testers, agile project managers, and business users. This blog is intended to be a tutorial for those who have no experience with database queries to those who want faster access to information in Jira.

Searching in Jira: how do I start?

The search box

The most simple search feature in Jira is the search box at the top right of your screen.

You can use this search box to:

search a particular issue

```
JiraKEY-15
```

search on text (searches in issue summary, description and comments)

```
atlassian
```

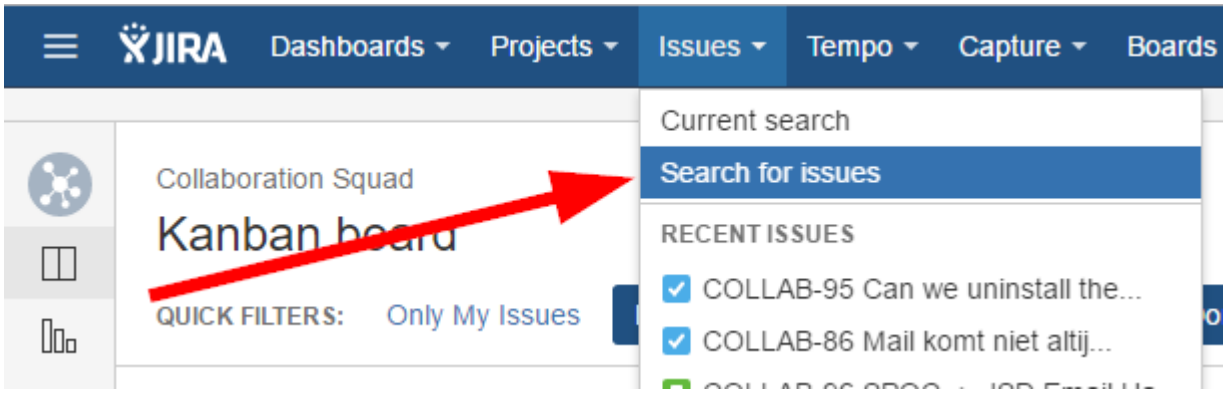
Finally, the field supports Smart Querying

```
my open bugs
```

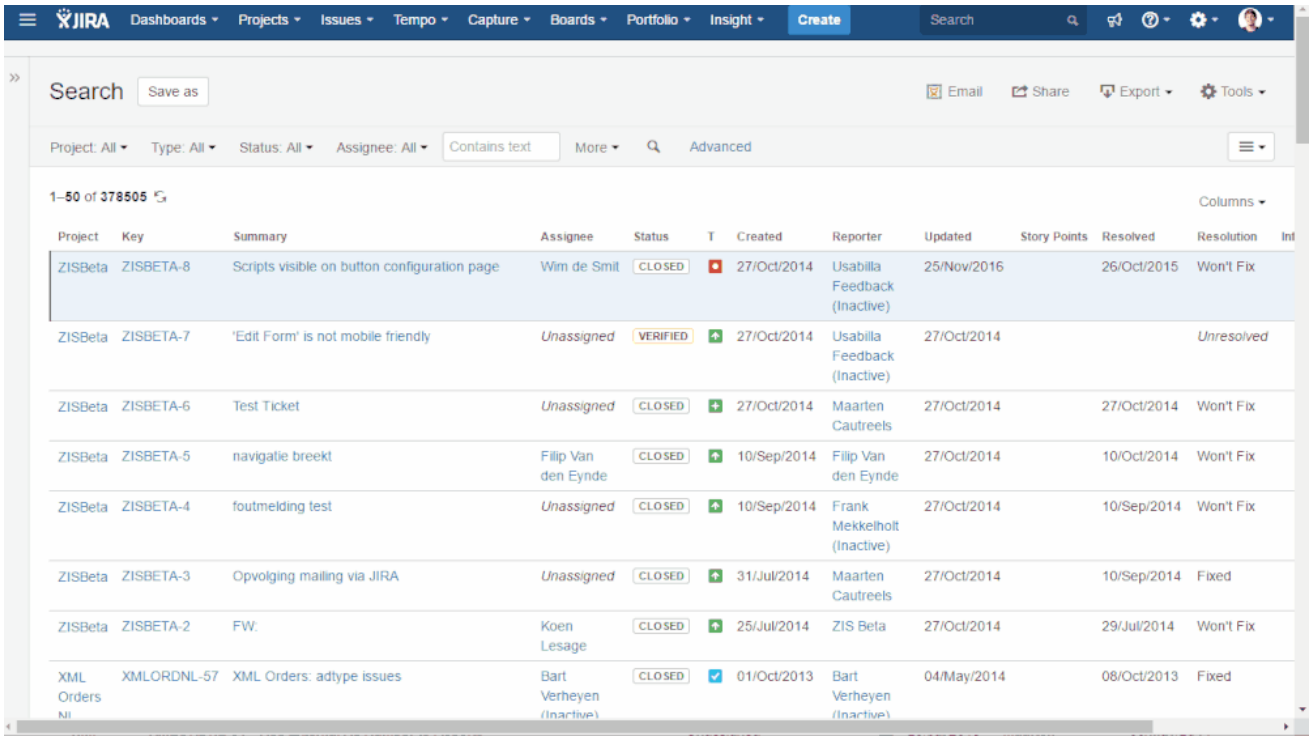
This will give you a list with all Bugs that are in the status Open and assigned to you

Search for Issues

You can also search issues from the “Issues” section in the dropdown menu.



In the screen that just opened you’ll be able to easily filter issues based on the value(s) of certain fields. A quick example search:



You can filter on any field available in Jira (depending on the selected project(s)).

Basic and Advanced Searches

There are two types of searches in Jira: basic and advanced. Basic searches, like the ones above, present you with a set of forms that you can fill in, such as Project name, Issue Type, Status, and Assignee. Basic search can be useful for getting a high-level view of your issues and status.

Now Advanced Searching is where you will get into JQL, using it to form queries. Queries are a series of simple elements strung together to form a more complex question. A query has three basic parts: fields, operators, and values.

Field – Fields are different types of information in the system. Jira fields include priority, fixVersion, issue type, etc.
Operator – Operators are the heart of the query. They relate the field to the value. Common operators include equals (=), not equals (!=), less than (<), etc.
Value – Values are the actual data in the query. They are usually the item for which we are looking.
Keyword – Keywords are specific words in the language that have special meaning. In this post we will be focused on AND and OR.

Important keywords to know

Atlassian has created a JQL reference where you can find all keywords, operators, etc that can be used in JQL. This is your go-to-guide when you want to discover new search capabilities. Here are some of the most common keywords and operators you will use:

AND

```
This will only return issues that match both clauses (are part of the Collaboration
project and have their status set to "In Progress")
project = Collaboration AND status = "In Progress"
```

OR

```
Returns all issues from either the Collaboration project or that have their status set
to "In Progress".
project = Collaboration OR status = "In Progress"
```

IS

```
This will return all issues that have no description.
```

```
description IS EMPTY
```

Operators

!=

```
status != "To Do"
```

Important keywords to know

Atlassian has created a JQL reference where you can find all keywords, operators, etc that can be used in JQL. This is your go-to-guide when you want to discover new search capabilities. Here are some of the most common keywords and operators you will use:

=” style=”margin: 30px 0px 0px;padding: 0px;color: #333333;font-size: 16px;font-weight: bold;line-height: 1.5;font-family: Arial, sans-serif;font-style: normal;letter-spacing: normal;text-align: start;text-indent: 0px;background-color: #ffffff”>>=

```
"Story Points" >= 5
```

Find all issues that have Story Points which are greater than or equal a given value

Similar

- > | greater than
- < | smaller than
- <= | smaller than or equal

IN

```
status IN ("To Do", "In Progress", "Closed")
```

is the same as

```
status = "To Do" OR status = "In Progress" OR status = "Closed"
```

Find all issues which are either have the status “To Do”, “In Progress” or “Closed”

Reverse

- NOT IN

Putting this into practice

Let's start with a simple example. Say you want to see what bugs your teammate filed within a certain project. You query would be:

```
reporter = jsmith AND project = Pipeline
```

Here we used the keyword "AND" to query Jira for tickets reported by John Smith in the project "Pipeline"

Other times you may want to have the query reference a set of items. For instance:

What issues are blocking or critical in Projects A, B, and C?

```
priority in (Blocker, Critical) AND project in (ProjA, ProjB, ProjC)
```

What issues are blocking or critical in Projects A, B, and C?

This is extremely helpful for organizations that have service level agreements (SLA) with their customer base. A JQL query can easily find the issues that are not meeting that SLA.

What issues are unassigned and have not been updated in the last day?

```
assignee is EMPTY and created < -1d
```

In this query the "is empty" statement only includes issues where the value of the assignee field is blank. This query also shows how Jira supports relative dates. The value -1d evaluates to 1 day behind the current date when the query is run. As a result, the above query will return all issues that do not have an assignee that are at least one day old.

JQL: Functions, history, and sorting

Functions: What do they do?

Like in math class, functions boil down complex logic and makes it easy for you to access and use in a simple way.

For example, Jira supports a function called membersof() that you can use to see all the issues assigned to members of a group. Groups can be defined inside of Jira or come from existing groups in your company's preexisting directory servers.

project = Pipeline AND assignee in membersof('test-engineering')

This query returns all of the issues that are assigned to test engineers. Functions also react dynamically as the environment changes. If more users are added to the test-engineering group, this query will dynamically update.

Let's see how we might see what issues got fixed in the last release.

project = Pipeline AND status in (resolved, closed) and fixversion = "Sprint A"

This will return all of the issues that were fixed in that particular release. What if we want a list that always shows the issues fixed in the last release whenever it was run? That's very easy to do with a function.

status in (resolved, closed) and fixVersion = latestReleasedVersion(Pipeline)

What's the difference here? Eventually the team will release sprints B, C, and D. The first query only returns the list of issues resolved in Sprint A. The second one queries Jira to find the last released version for project Pipeline. Thus as sprints B, C, and D are released this query will pull the right data for those releases. Pretty cool, aye?

History: What happened back then?

You can also use JQL to see how our work has been done over time. For instance, if you wanted to see the bugs that John Smith resolved in the Pipeline project, you can use the following query:

project = Pipeline AND status CHANGED FROM "In Progress" TO "Resolved" BY jsmith

Sometimes issues that fail verification or get reopened are of special interest to see why a change didn't work as intended. To find those bugs it's easy to run this query:

status CHANGED FROM "In Progress" TO "Open"

To see what issues were in flight during the current week we can use the following JQL:

status was ("In Progress") DURING (startofweek(), endofweek());

At the end of the year it can be nice to see how many issues you resolved during the year. To do so, all it takes is a little JQL:

resolution changed to "Fixed" by currentUser()

during (startOfYear(), endOfYear())

Scoping and Sorting:

Oftentimes when you start searching your Jira issues, you can come back with A LOT of information. Scoping and sorting your queries will make it easy to see the exact information you need.

Scoping – focusing your query so it pulls the right amount of data so the user sees only the information relevant to the current item at hand

Sorting – ordering your data such that the most critical set of data is listed first
Scoping

Starting with open issues:

project = Pipeline AND status = open

You might find that query returns too many issues as it includes everything from your backlog as well as issues you are currently working on. You can tighten it up a bit by eliminating the backlog issues.

project = Pipeline AND status = open AND fixVersion = "Current Sprint"

This result is better, but you can narrow it down even more. Let's see what didn't make last sprint and got moved into this sprint.

project = Pipeline AND status = open AND fixVersion = "Current Sprint"
AND fixVersion WAS "Last Sprint"

Now that you have the issues you really care about in front of you, it's easy to look at each of these issues with a high degree of focus and see why they didn't make the last sprint. Was it an estimation problem, requirements, or something else? The issues were there in the first query, but they were too hard to find manually. Good query creation is an iterative process: query, review, see if you can focusing the results. The more you do, the easier it becomes.

Sorting

Let's say you want to see all the bugs open for the team in a particular sprint:

project = Pipeline AND fixVersion = "Current Sprint" AND status = open

What are you trying to glean from this data? If you are trying to understand risk you might want to see the list ordered by priority and then by engineer so that you can easily see if one engineer has two high priority bugs. JQL has a keyword ORDER

BY that tells Jira to sort the results. The above query could be extended to include sorting:

project = Pipeline AND fixVersion = “Current Sprint” AND status = open ORDER BY priority, assignee

Jira will first order the list by priority and then sort by assignee for all of the issues with the same priority. Let’s look at another example examining the incoming bugs to our project. We want to see any new critical or blocking bugs that have come in recently to see if recent checkins have decreased stability.

project = Pipeline AND priority in (blocker, critical) AND created > -2w ORDER BY created DESC

The query controls for priority and limiting the created time properly scope the query. What the sorting does is show us the most recent issues first. Use the DESC keyword to sort in reverse (newest to oldest). That way you focus your audience on the most important issues first. Not everyone will look at each issue in your reports. Always sort your queries so that the most important issues show up first.

Advanced searching

1. Choose the Jira icon (or) > **Filters**.
2. Select **Search issues**.

- If basic search is shown instead of advanced search, click **Advanced** (next to the icon). If advanced is already enabled, you'll see the option of switching to basic.

3. Enter your JQL query. As you type, Jira will offer a list of "auto-complete" suggestions based on the context of your query. Note, auto-complete suggestions only include the first 15 matches, displayed alphabetically, so you may need to enter more text if you can't find a match.
4. Press Enter or click to run your query. Your search results will display in the issue navigator.

Understanding advanced searching

`Constructing JQL queries`

A simple query in JQL (also known as a 'clause') consists of a *field*, followed by an *operator*, followed by one or more *values* or *functions*. For example:

```
project = "TEST"
```

This query will find all issues in the "TEST" project. It uses the "project" *field*, the EQUALS *operator*, and the *value* "TEST".

A more complex query might look like this:

```
project = "TEST" AND assignee = currentuser()
```

This query will find all issues in the "TEST" project where the assignee is the currently logged in user. It uses the "project" *field*, the EQUALS *operator*, the *value* "TEST",the "AND" keyword and the "currentuser()" function.

Setting the precedence of operators

You can use parentheses in complex JQL statements to enforce the precedence of operators.

For example, if you want to find all resolved issues in the 'SysAdmin' project, as well as all issues (any status, any project) currently assigned to the system administrator (bobsmith), you can use parentheses to enforce the precedence of the boolean operators in your query, i.e.

```
(status=resolved AND project=SysAdmin) OR assignee=bobsmith
```

Note that if you do not use parentheses, the statement will be evaluated left-to-right.

You can also use parentheses to group clauses, so that you can apply the NOT operator to the group.

Restricted words and characters

Reserved characters

JQL has a list of reserved characters:

space (" ") + . , ; ? | * / % ^ $ # @ [ ]

If you wish to use these characters in queries, you need to:

- surround them with quote-marks (you can use either single quote-marks (') or double quote-marks ("));
  and, if you are searching a text field and the character is on the list of reserved characters for text searches,
- precede them with two backslashes.

For example:

- `version = "[example]"`

  You can't search for many special characters in text fields using this method. For advanced text searches, see Search syntax for text fields.

Reserved words

JQL also has a list of reserved words. These words need to be surrounded by quote-marks (single or double) if you wish to use them in queries:

"abort", "access", "add", "after", "alias", "all", "alter", "and", "any", "as", "asc", "audit", "avg", "before", "begin", "between", "boolean", "break", "by", "byte", "catch", "cf", "char", "character", "check", "checkpoint", "collate", "collation", "column", "commit", "connect", "continue", "count", "create", "current", "date", "decimal", "declare", "decrement", "default", "defaults", "define", "delete", "delimiter", "desc", "difference", "distinct", "divide", "do", "double", "drop", "else", "empty", "encoding", "end", "equals", "escape", "exclusive", "exec",

"execute", "exists", "explain", "false", "fetch", "file", "field", "first", "float", "for", "from", "function", "go", "goto", "grant", "greater", "group", "having", "identified", "if", "immediate", "in", "increment", "index", "initial", "inner", "inout", "input", "insert", "int", "integer", "intersect", "intersection", "into", "is", "isempty", "isnull", "join", "last", "left", "less", "like", "limit", "lock", "long", "max", "min", "minus", "mode", "modify", "modulo", "more", "multiply", "next", "noaudit", "not", "notin", "nowait", "null", "number", "object", "of", "on", "option", "or", "order", "outer", "output", "power", "previous", "prior", "privileges", "public", "raise", "raw", "remainder", "rename", "resource", "return", "returns", "revoke", "right", "row", "rowid", "rownum", "rows", "select", "session", "set", "share", "size", "sqrt", "start", "strict", "string", "subtract", "sum", "synonym", "table", "then", "to", "trans", "transaction", "trigger", "true", "uid", "union", "unique", "update", "user", "validate", "values", "view", "when", "whenever", "where", "while", "with"

Apart from the type of searches explained in the previous chapter, JIRA also has a few advanced search options, which can be performed using the following three ways.

Using Field Reference
Using Keyword Reference
Using Operators Reference

These above-mentioned three ways have been explained in detail below.

The user should consider the following points while performing any advanced search.

Advanced search uses structured queries to search for JIRA issues.

Search results displays in the Issue Navigator.

Search results can be exported to MS Excel and many other available formats.

Save and Subscribe features are available to advanced searches.

An advanced search uses the JIRA Query Language known as JQL.

A simple query in JQL consists of a field, operator, followed by one or more values or functions. For example, the following simple query will find all issues in the "WFT" project –

```
Project = "WFT"
```

JQL supports SQL like syntax such as ORDER BY, GROUP BY, ISNULL() functions, but JQL is not a Database Query Language.

Using Field Reference

A field reference means a word that represents the field name in the JIRA issue including the custom fields. The syntax is –

```
<field name> <operators like =,>, <> “values” or “functions”
```

The operator compares the value of the field with value presents at right side such that only true results are retrieved by query.

Using Keyword Reference

Here, we will understand how to use a keyword reference and what its advantages are

A keyword in JQL –

joins two or more queries together to form a complex JQL query.
alters the logic of one or more queries.
alters the logic of operators.
has an explicit definition in a JQL query.
performs a specific function that defines the results of a JQL query.
List of Keywords –

AND – ex - status = open AND priority = urgent And assignee = Ashish.
OR – ex – duedate < now() or duedate is empty.
NOT – ex – not assignee = Ashish .
EMPTY – ex - affectedVersion is empty / affectedVersion = empty.
NULL – ex – assignee is null.
ORDER BY – ex – duedate = empty order by created, priority desc.
Similar to field reference, as soon as the user starts typing, the auto-complete functionality helps to get the correct syntax.

Jira Reports

Reports are used to track and analyze the team's performance throughout a project.

Jira has a range of reports that shows the information about your project, versions, sprints, epics, and issues.

Following are the reports generated by report:

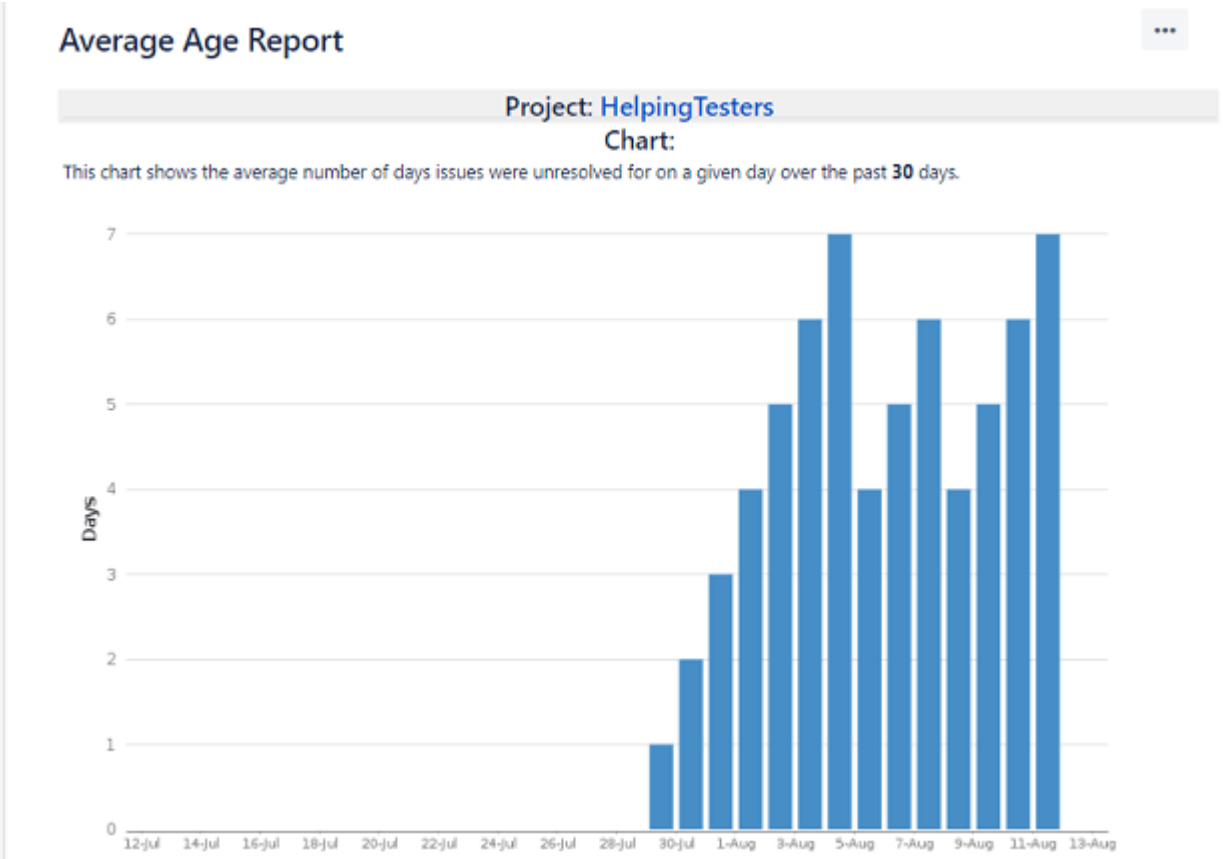
Average age report

An average age report shows the average age of unresolved issues for a project or a filter. It helps you to show whether the backlog is kept up-to-date or not.

Basically, it finds the average number of days for which the issues are kept unresolved.

The Average Age report is generated that depends upon the selected project, the type of issue selected in the filter, and time is chosen (hours/days/week/months).

Average age report can be shown as below:

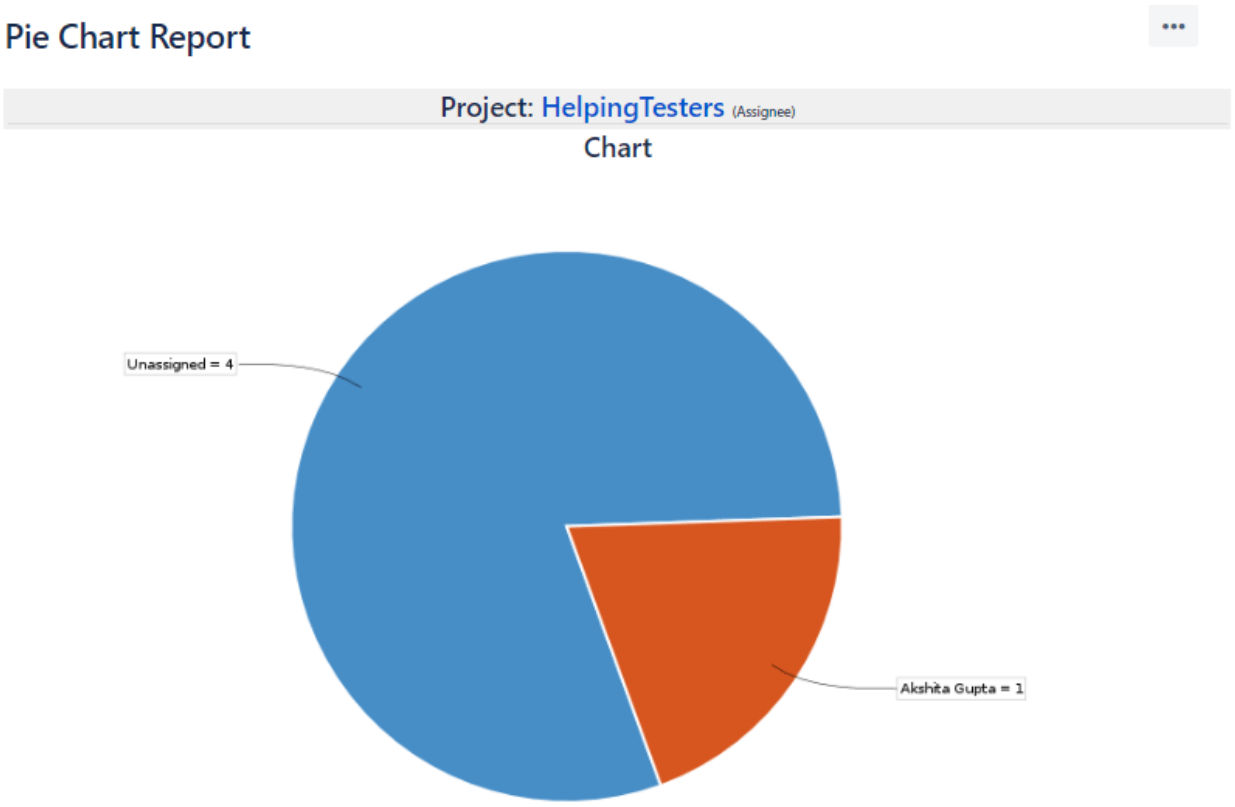


The above screen shows the average age of all the unresolved issues over the past 30 days.

Pie chart

The pie chart displays the issues which are returned from the specified project or an issue filter. We can create an issue filter that retrieves all the issues belonging to a specific version of a particular project. The pie chart is created to group all the issues belonging to a specific statistic type, and statistic type can be Assignee, project, etc.

Suppose I choose the statistic type as **Assignee**.



The above screen shows the pie chart based on the "Assignee" type. The above chart shows that the one issue is assigned to the Akshita Gupta, and the other four issues are unassigned.

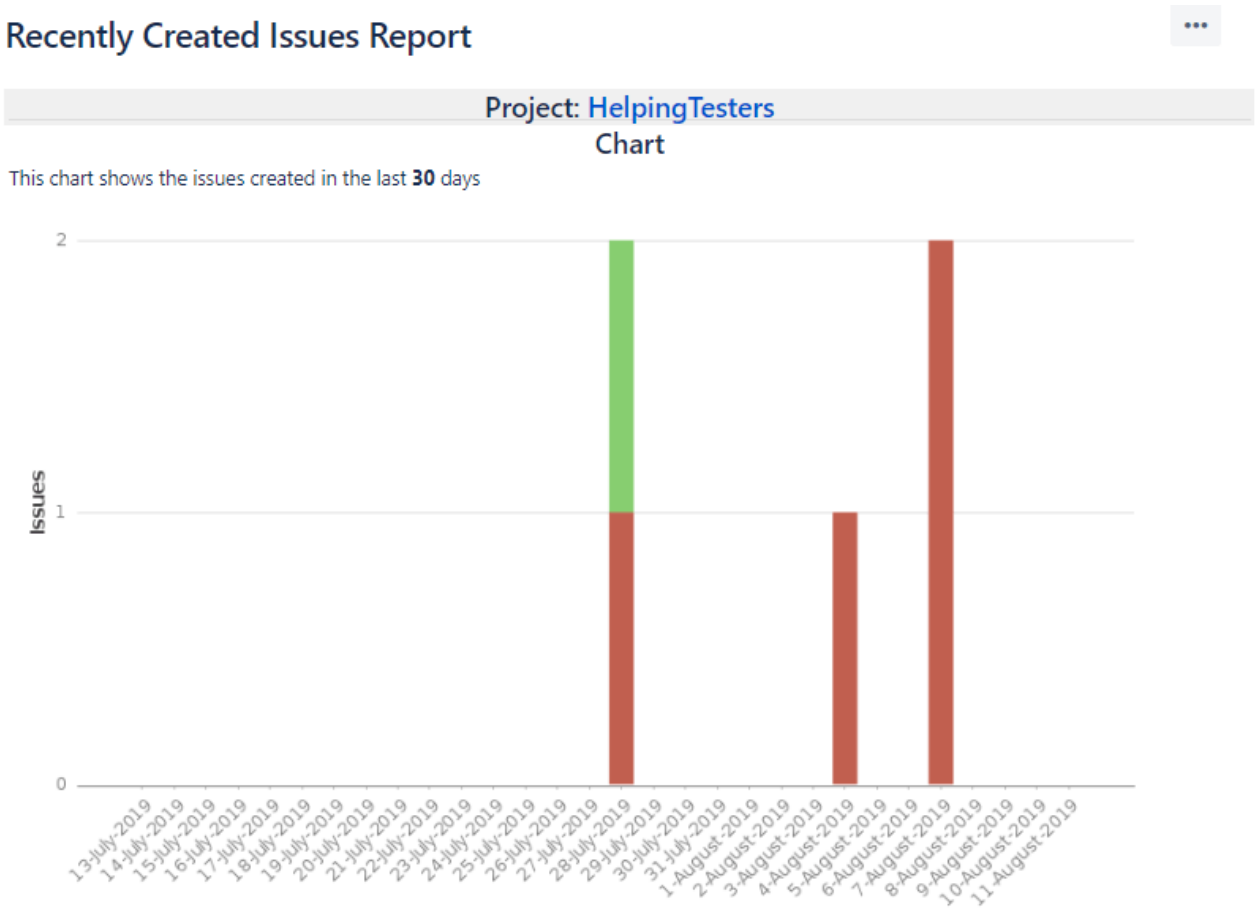
Created vs. Resolved issue

The created vs. resolved issue is a report that displays the number of issues which are created and resolved over a given period of time.

This report is created based on the project and issue filter that the user chooses and the chart can either be cumulative or not.

Recently Created Issues Report

The recently created issues report displays the number of issues that are created, and some of the issues are resolved.



In the above screen, the red portion shows that that issues are created and they are unresolved while the green color shows that the issues are created, but they are resolved.

Resolution Time Report

The resolution time report determines the length of the time taken to resolve the issues for a project/filter.

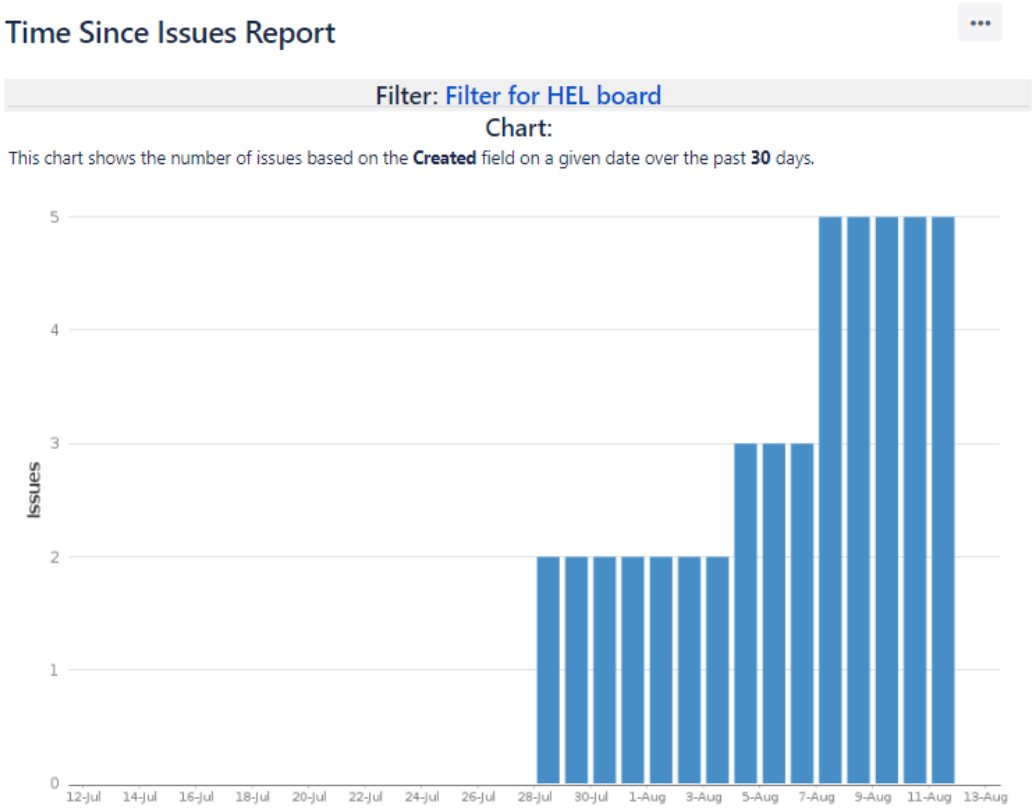
Single Level Group By Report

Single level group by report categorizes the issues by a particular field for an issue filter, but it does not display the chart.

The above screen shows that one issue, i.e., HEL-1, is assigned while other four issues, i.e., HEL-5, HEL-3, HEL-2, HEL-4, are unassigned. The Single Level Group By Report shows the progress of both assigned and unassigned issues.

Time Since Issues Report

The Time Since Issues Report generates the report that shows the number of issues for the chosen data field, and the data field can be Created, Updated, Due, Resolved.



User management

You can use JIRA to manage it's own users, or you can connect JIRA to an external user management system.
You can also use JIRA as a user management system for other Atlassian products, so that your users have the
same login details for all their Atlassian products.

Managing users

This section covers all aspects of using JIRA for user management.
Learn everything from how to create and view a user, to deactivation and monitoring user activity.

Managing groups

Learn which groups exist by default when you install a JIRA application as

well as how to create, edit, and delete groups, add users to groups,. This section also covers assigning permissions to JIRA functions.

Advanced user management

Learn about the advanced user management features in JIRA, such as allowing other Atlassian products to connect to JIRA for user management, enabling public signup, and user management limitations and recommendations.

User directories

Learn more about your JIRA user directory and how to connect to external directories for external user management.

Managing users

As an administrator, you can manage users directly in JIRA, or enable public signup so users can create their own accounts. You can refer to these pages for information on managing users across multiple projects and applications.

Create, edit, or remove a user

Users in JIRA applications can be managed manually or via External User Management. This page helps you manage these users manually, and references external user management systems where required.In order for a user to log in and access a JIRA application, they must have application access. Application access is obtained by being a member of a group assigned to an application. Membership to these groups can be changed at any
time on a per user basis.

Note: Before you begin You must have the JIRA Administrator or JIRA System Administrator global permission to be able to manage users in JIRA applications

Create a user in JIRA

Create a user directly in JIRA if you have a small team.Consider external user management (LDAP or Active Directory) if you have a lot of hands on deck. Maintaining permissions for individual user ID's can be messy if you have too many users, so there are other options for your large staff.

To create a user:

Select > User Management.
In the User browser, click Create User.
Enter the Username, Password, Full Name and Email address.
Optionally, select the Send Notification Email checkbox to send the user an email containing:
their login name.

a link from which to set their password (this link is valid for 24 hours).

If you have more than one JIRA application installed, you will need to select the JIRA application you would like to give the user access to.

Click the Create button.

After the user is created, you'll brought to a screen where you can view the user information and perform additional functions such as edit details, edit groups or properties, and delete the user.

To Edit a user

Modifying user information, such as name, email, address, and password, is easy with the JIRA internal directory. If you are using an external authentication method such as LDAP or Active Directory, you'll have to make changes in that system rather than in JIRA.

Edit a username, full name, or email address

These three attributes can be modified together, in a few simple clicks, if you're using the JIRA internal directory to manage users. When updating a username, it's important to note that:

JIRA cannot update external usernames - for example, users that are coming from an LDAP server or Crowd instance. However, JIRA can update JIRA users stored in an "Internal Directory with LDAP Authentication."

If you are using your JIRA instance as a JIRA User Server for other applications, (e.g., Confluence), you will not be able to use this feature. If you aren't sure about this, check under **User Management > JIRA**

User Server to confirm that no external applications have been configured to use JIRA as a Crowd Server

Here is example of JIRA Admin Project Management from Creation of Project.

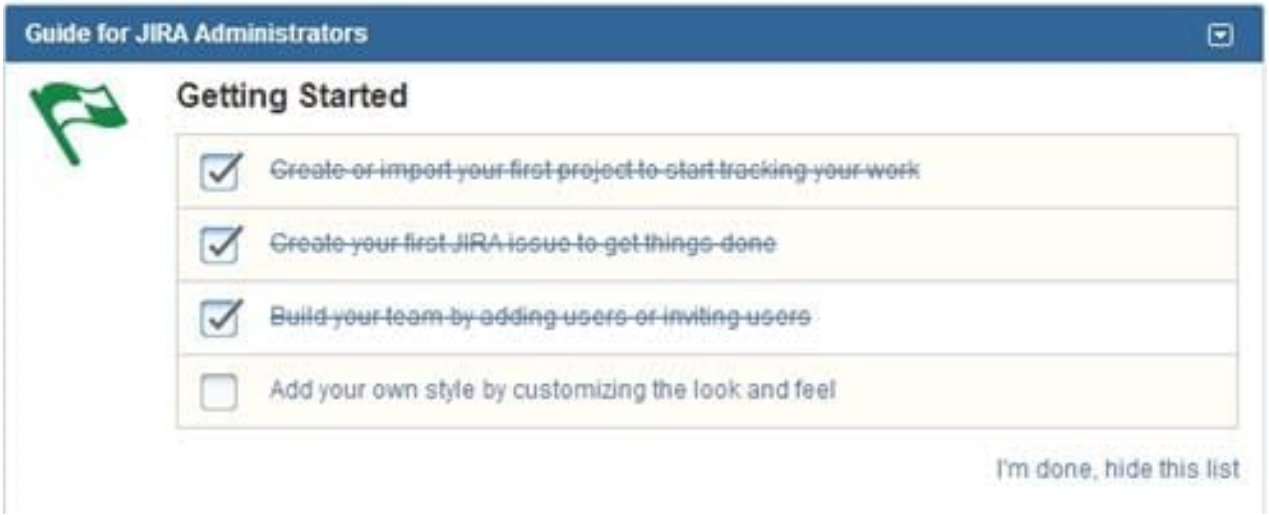
Project Management is one of the first and most important concepts in administration.

A JIRA Admin can:

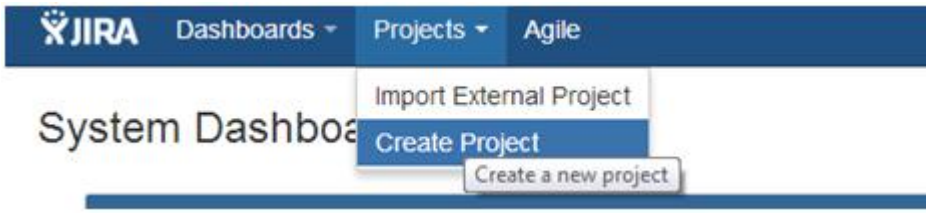
Create a project
Configure an already created Project.

Create A Project

#1) Once you log in, on the Dashboard itself you will have a few options to get started:



#2) You can either click on the item in the list or you can choose:



Or go to the Administration->Projects menu option:

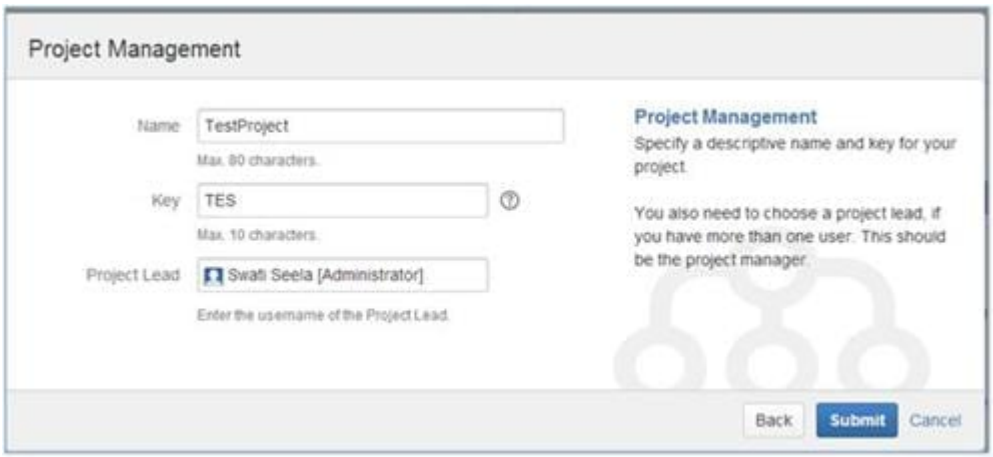
And then on this page, you will have an option to "Add Project"

#3) Choose any of the above 3 methods to add a project. In the below window, choose the type of Project.

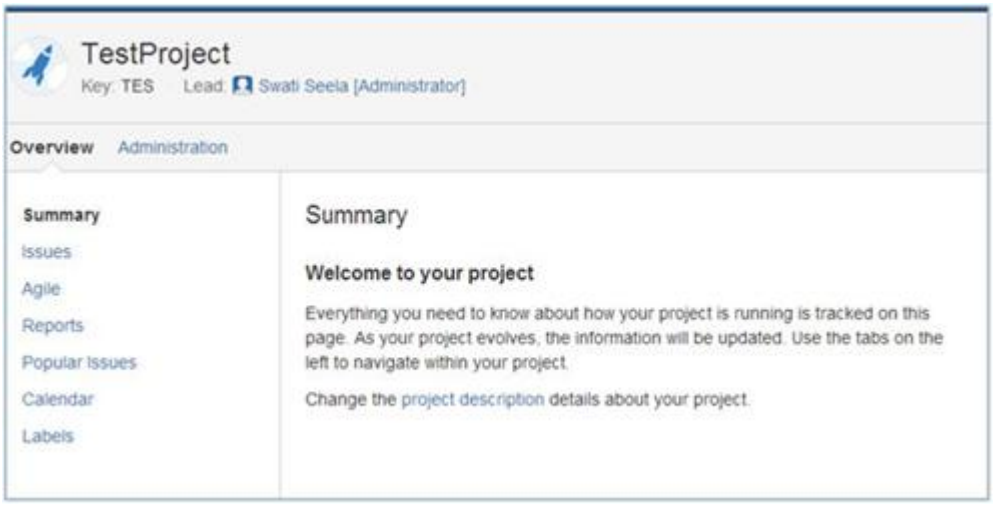
Pro Tip: The Project type will determine the type of workflow your issues will go through.

Note: Please note the "Import from external systems" link at the bottom of the page. If you have been using a different bug/issue tracking system and would like to migrate to JIRA. This is the option you could use. Also, if all you had earlier is an excel sheet for all your issues, you can create a project by importing all the issues in the CSV file into JIRA.

#4) Enter the Project Name, Key (once chosen cannot be changed) and assign a Project Lead (the person who is responsible for the overall project). Click Submit when done.

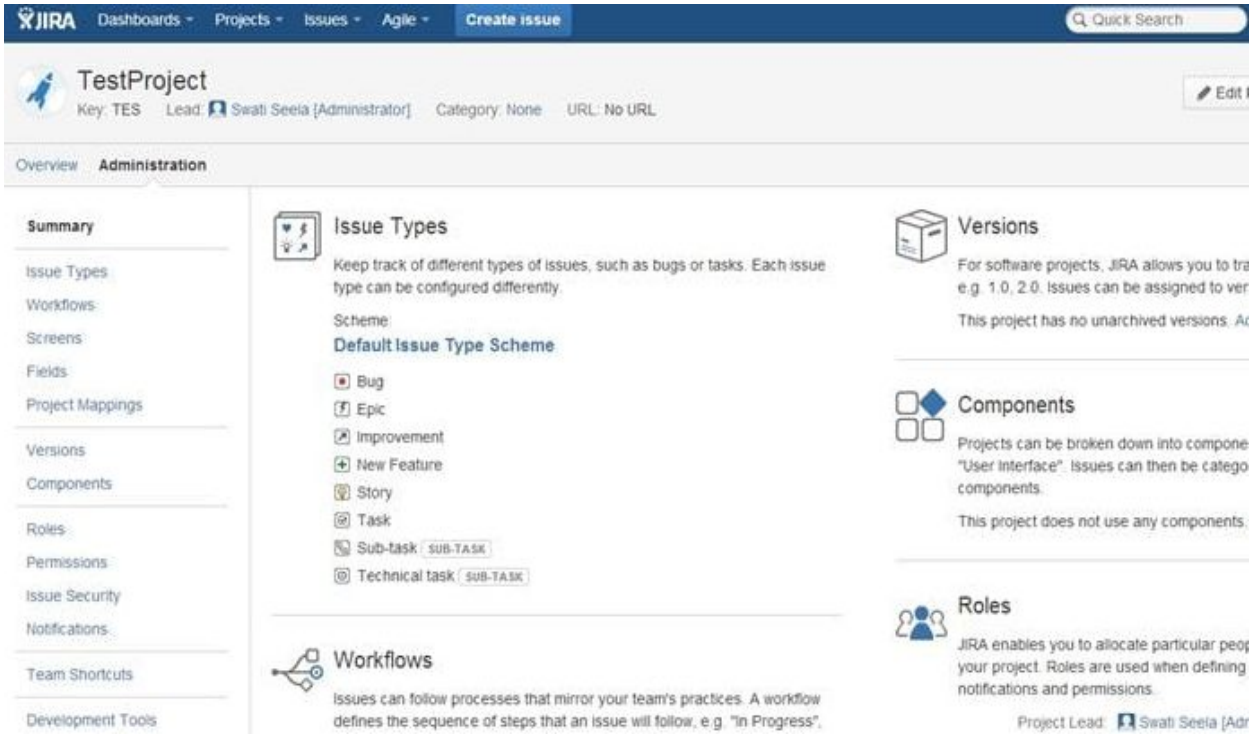


#5) The project gets created and the following details are displayed. By going to the "Administration" link you will be able to configure the project (in case of an already configured project, you can edit the configurations).



Configuring A Project

As you can see below, you can choose the issue types that need to show up, the workflow that the project follows, versions and components, people involved and the level of access permissions they have, etc.



User Management In JIRA

User creation is the most important part of the User Management process, it is not limited to just that.

The important user-related activities performed by a JIRA Admin are:

Edit a particular user information
Delete a user
Creating users groups
Creating Roles
Permissions
Setting Preferences

We will see the creations, deleting and editing of an issue in detail below. For the rest of the operations, we would encourage you to try them on the site with your trial ID.

Note: With the trial, you can add up to 10 users to your account i.e., 9 additional users and 1 admin, yourself.

Firstly, Go to **Administration->User Management**:

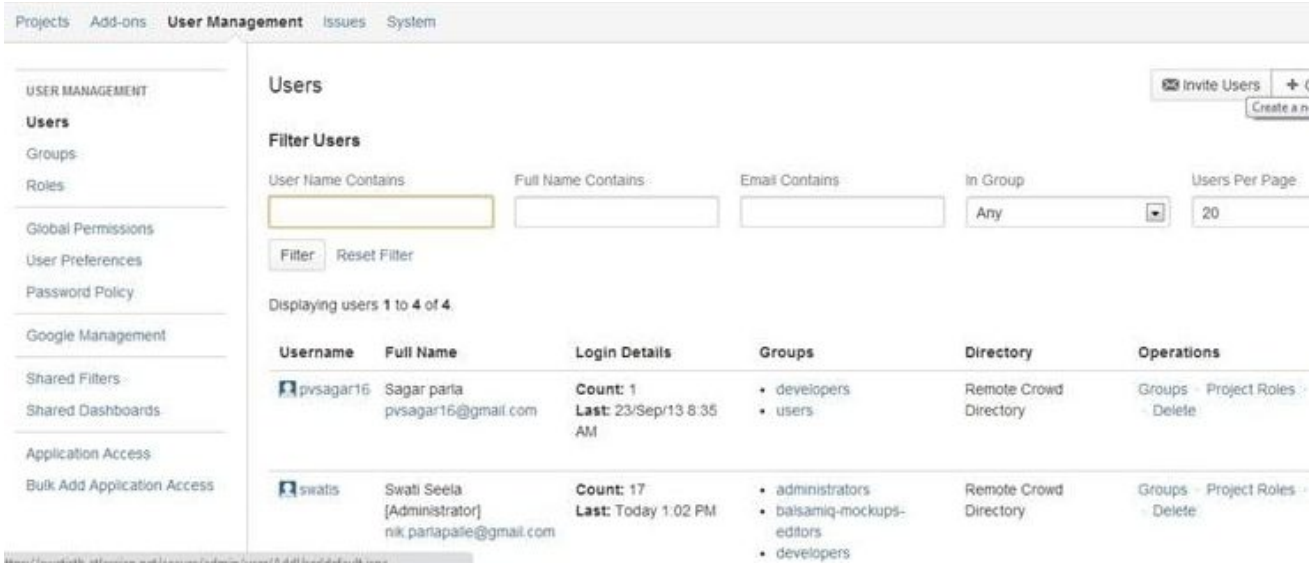
There are two ways to which you can add users to JIRA.

Create them manually.

Send an invite to join JIRA to a user's email ID

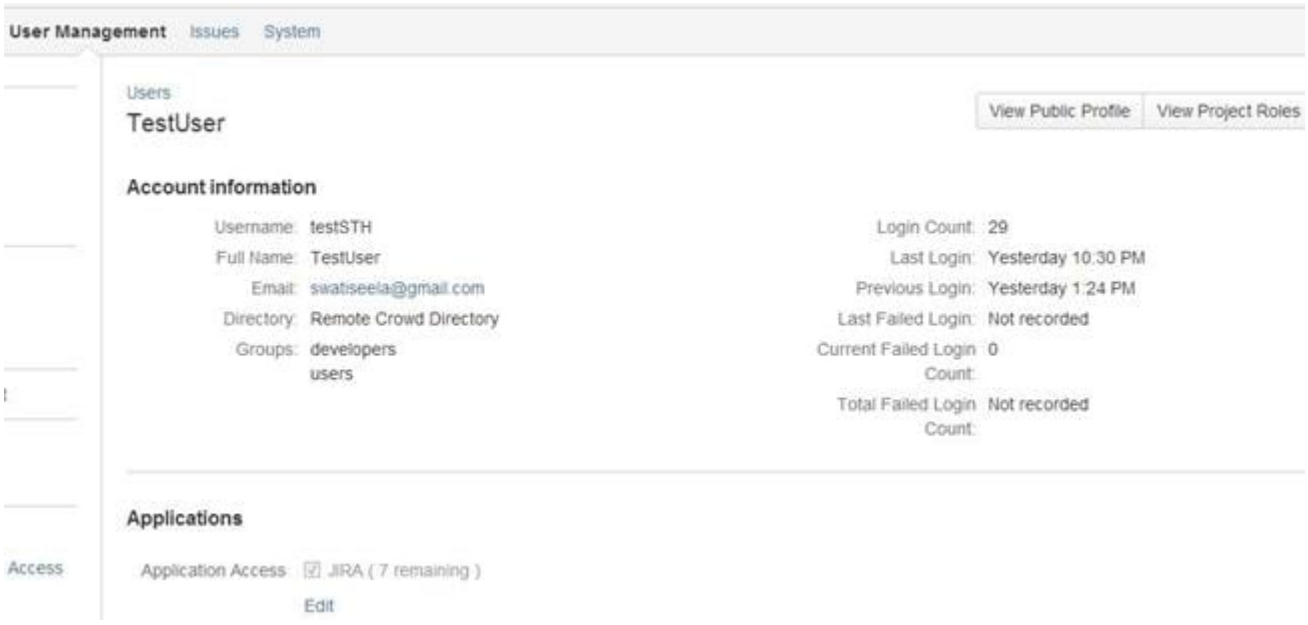
Let us look at creating manually in detail:

#1) Click on Create User button in the below page:



#2) Enter the user details in the "Create New User" dialogue that opens up. In the below example I have entered only mandatory fields. The others are self-explanatory and can be set as required.

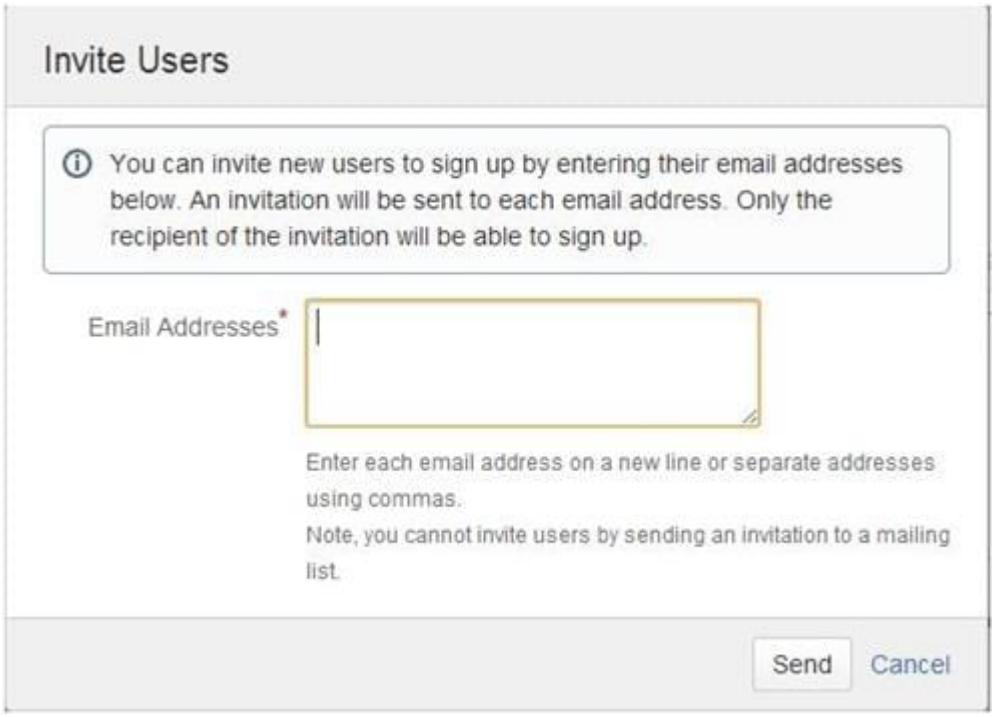
#3) On Clicking create, the user gets added and the following confirmation page comes up for the user.



An email is sent to the ID provided to the user with the information on how to set up a password that would enable the user to login and use JIRA.

Method 2 of User creation, via Email invites:

#1) Click on the "Invite Users" button in the User Management->Users screen and enter one or more email ID's in the below dialogue that opens up. When done, click Send.



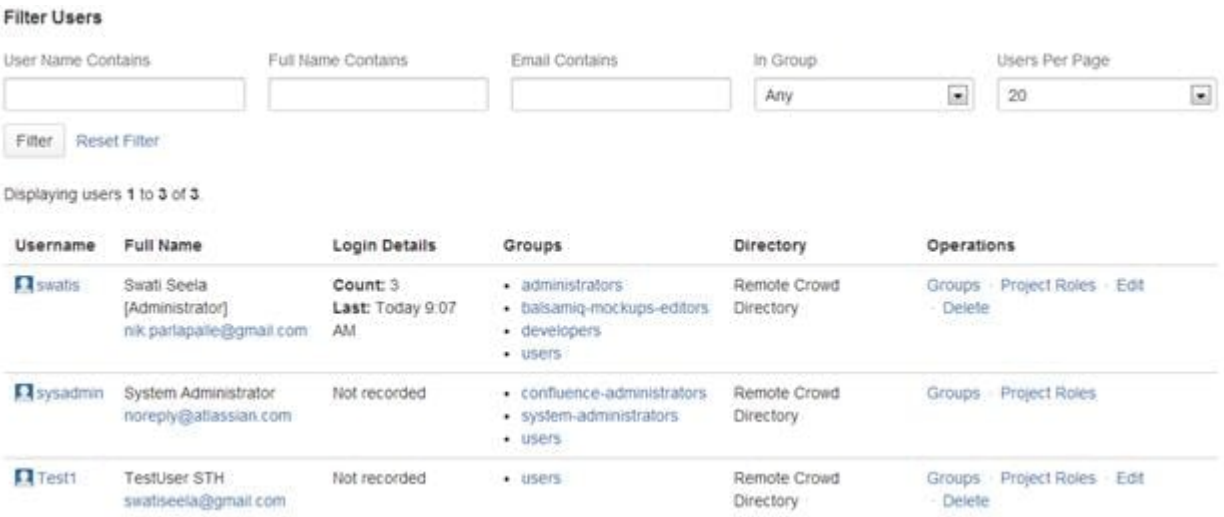
#2) The invitation sent message comes up when the message is successfully sent.

#3) In case of an email invite, the user is not added until the recipient of the email invite acts on the received invitation.

Editing A User

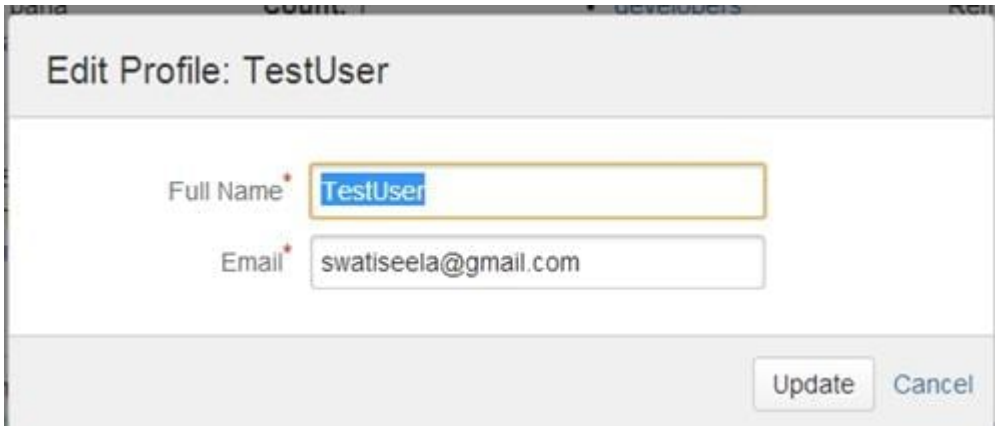
#1) Go to the **User Management -> Users page.** All the users available will be displayed. The top portion of this page contains a "Filter" section. This can be used to manipulate how/what user information needs to be displayed. You can choose to keep it empty, in which case it displays all the users available.

#2) Below the filter is a list o fusers organized in a tabular form is displayed.



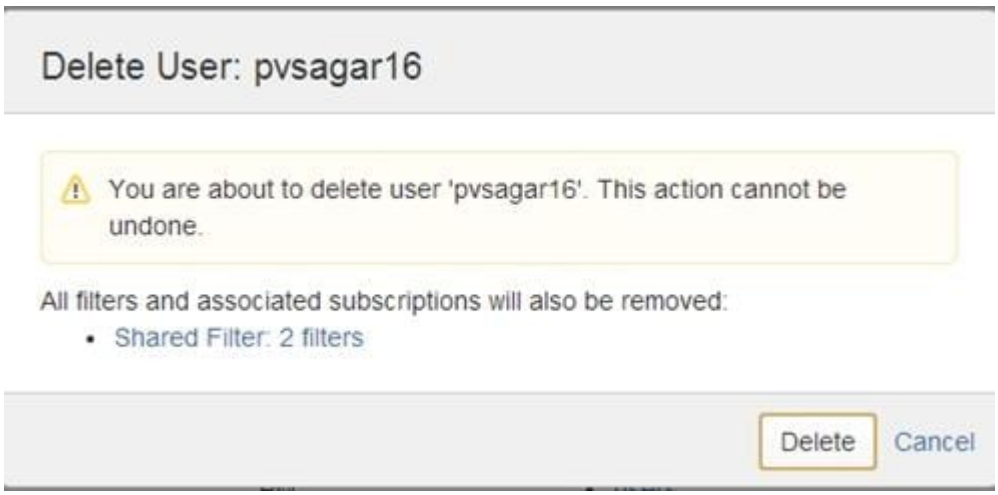
#3) Note the "Edit" link in the "Operations" column. Click on the corresponding "Edit" link for the user whose information is to be modified.

You will be able to change the following profile information.



Delete A User

Choose the "Delete" link corresponding to the user you would like deleted from JIRA. The following confirmation message is displayed and you can choose to delete a user or cancel the operation.



A JIRA Admin can additionally:

Configure the look and feel
Configure workflows
Set Project/issue-level security details.
Can add custom fields/screens
Integrate a JIRA project with development tools to make sure that the commit, revert, changes, etc. can reflect immediately in JIRA.
Configure Dashboard settings
Set time/time zone information.
Configure and set email preferences
Please note that all the admin aspects of any tool should more or less support the activities we have detailed above.

Also, admin access to a tool is very powerful and, “with great power comes great responsibility”. :)

Typically, Admin access is limited to just one person in an organization to make sure that accidental inconsistencies do not happen. All the requests for changes, new projects or new users are directed through the admin.

Configuring JIRA applications to receive email from a POP or IMAP mail server

To enable JIRA to create comments and issues from email, you need to first configure JIRA to receive email from a POP or IMAP mail server as described below.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Add or edit a POP or IMAP mail server

1. Choose  > System.
2. Select **Mail > Incoming Mail** to open the Incoming Mail page.
3. Click either the **Configure new POP / IMAP mail server** button to define a new POP / IMAP mail server, or the **Edit** link at the right of an existing POP / IMAP mail server configuration, which will open the **Add/U pdate POP / IMAP Mail Server** page.
4. Complete the fields on this page as follows:

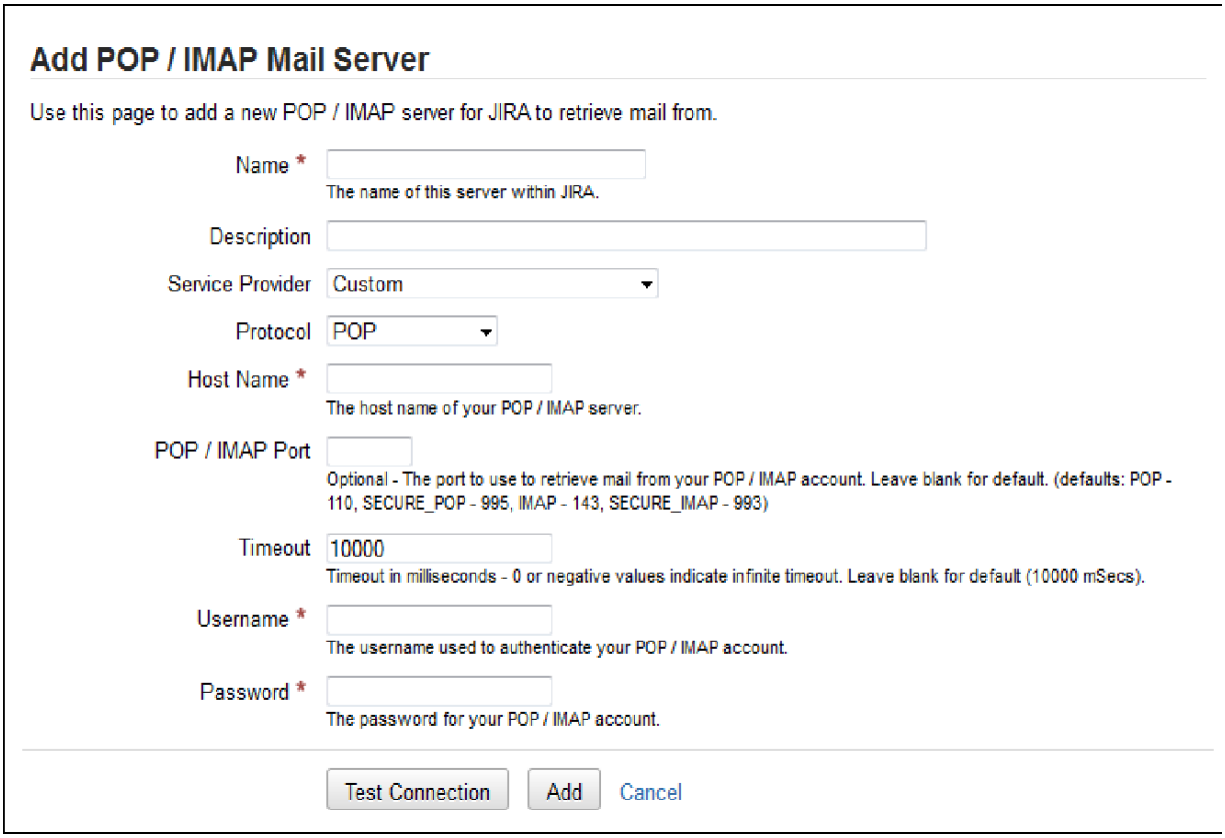
Name	Specify a short, arbitrary name to identify your POP or IMAP mail server configuration. You could possibly just specify the email address of the POP / IMAP mail server.
Description	(Optional) Specify an arbitrary description that describes the POP or IMAP mail server configuration and/or what it is used for. For example, 'Email Issue Creation/Comments for <Project>'. This description appears below the **Name** of the POP / IMAP mail server on the **POP / IMAP Mail Servers** configuration page.

Service Provider (not available when updating an existing POP / IMAP mail server)	Choose between using your own POP / IMAP mail server (i.e. **Custom**), Gmail POP / IMAP (i.e. **Google Apps Mail / Gmail [POP3 / IMAP]**) or Yahoo! POP (i.e. **Yahoo! MailPlus**) as the service provider for your POP / IMAP mail server. ℹ If you choose any of the Gmail or Yahoo! options and then switch back to **Custo m**, some of the key fields in this section will automatically be populated with the relevant POP / IMAP mail server settings for these service providers.
Protocol	Choose between whether your POP / IMAP mail server is a standard (i.e. **POP or IMAP**) or a secure (i.e. **SECURE_POP or SECURE_IMAP**) one. ⚠ JIRA Cloud does not support self-signed certificates.
Host Name	Specify the hostname or IP address of your POP / IMAP mail server. Eg. pop.your company.comor imap.yourcompany.com
POP / IMAP port	(Optional) The port to use to retrieve mail from your POP / IMAP account. Leave blank for default. Defaults are: POP: 110; SECURE_POP: 995; IMAP: 143; SECURE_IMAP: 993.

Timeout	(Optional) Specify the timeout period in milliseconds, which is treated as 10000 if this field is left blank. Specifying 0 or a negative value here will result in JIRA waiting indefinitely for the POP / IMAP server to respond.
Username	The username used to authenticate your POP / IMAP account.
Password	The password for your POP / IMAP account. ℹ When editing an existing POP / IMAP mail server, select the **Change Password** checkbox to access and change this field.

1. *(Optional)* Click the **Test Connection** button to check that JIRA can communicate with the POP / IMAP mail server you just configured.
2. Click the **Add** (or **Update**) button to save the POP / IMAP mail server configuration.

Screenshot: Add/Update POP / IMAP Mail Server



POP / IMAP over SSL

You can encrypt email communications between JIRA and your mail server via SSL, provided your mail server supports SSL.

Firstly, you will need to **import the mail server certificate** into a Java keystore. The process is described on the Connecting to SSL Services page.

⚠️ **Important Note:** Without importing the certificate, JIRA will not be able to communicate with your mail server.

Upgrading JIRA applications

If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

There are several ways to upgrade JIRA, and the method you choose to use depends on which version of JIRA you use, and the type of environment

If you need to move JIRA to a new server, or use it in a new environment that has a different operating system, different database type, or different location of attachment or index files, follow the instructions for Migrating JIRA

you use it in. Use this table to determine which steps to follow to complete your JIRA upgrade:

Upgrading JIRA applications manually

If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

This page describes how to upgrade JIRA installations that don't support the rapid upgrade method or fallback method. You should use this method to upgrade JIRA if you meet any of the following criteria:

- You use JIRA 4.0.0 or later on Solaris.
- You use JIRA 4.0.0 – 4.2.x on Windows or Linux.

See Upgrading JIRA applications for more information on the methods you can use to upgrade JIRA

1. Before you start

- **Read about the new version** - Review the release notes and upgrade notes for the version of JIRA that you are upgrading to. See our Release notes for JIRA Server. If you plan to skip a few JIRA versions during your upgrade, **we strongly recommend** that you read the upgrade guides for all major versions between your current version and the version to which you are upgrading.
- **Check your license** - Verify that your license support period is still valid.
- **Check for known issues** - Use the JIRA Knowledge Base to search for any issues in the new version that will affect you.
- Check for compatibility:
  - Confirm that your operating system, database, other applicable platforms and hardware still comply with the requirements for JIRA 7.3. The End of Support Announcements page also has important information regarding platform support for future versions of JIRA.
  - If you have installed JIRA plugins (i.e. not included with JIRA), verify that they will be compatible with the version of JIRA you are upgrading to. You can find a plugin's compatibility information from the plugin's home page on the Atlassian Plugin Exchange.
  - Some anti-virus or other Internet security tools may interfere with the JIRA upgrade process and prevent the process from completing successfully. If you experience or anticipate experiencing such an issue with your anti-virus/Internet security tool, disable this tool before proceeding with the JIRA upgrade.
- **Prestaging and testing your new version of JIRA** - We strongly recommend performing your upgrade in a test environment first. Do not upgrade your production JIRA server until you are satisfied that your test environment upgrade has been successful.
  - If you have any problems with your test environment upgrade which you

cannot resolve, create an issue at our support site so that we can assist you.

- If you have any problems during the upgrade of your production JIRA server, do not allow your users to start using this server. Instead:
  - Continue to use your old JIRA server — this will help ensure that you do not lose production data.
  - Also create an issue at our support site so that we can help you resolve the problems with your upgrade.

2. Backing up

Before you begin the JIRA upgrade, we strongly recommend that you back up your existing JIRA installation.

2.1 Stop users from updating JIRA data

During the upgrade process, you'll export JIRA's database from your existing JIRA installation (via an XML backup) and then restore this backup into a new JIRA installation. To ensure that the data in the XML backup is consistent with the latest data in the system, you must temporarily restrict access to JIRA so users can't update the data. Refer to the Preventing users from accessing JIRA applications during backups page for more information.

⚠️ **Be aware!** Inconsistent XML backups cannot be restored!

2.2 Back up your database

Perform an XML backup of your existing JIRA installation's external database. For large JIRA installations, this process may require several hours to complete.

> The 'embedded database' is the H2 database supplied with JIRA for evaluation purposes only. If you accidentally use the H2 database in a production system, perform an XML backup of this database and continue on with this procedure.

2.3 Back up your JIRA home directory

1. Shut down JIRA.
2. Locate the JIRA home directory. You can find information about the location of the directory by navigating to the <jira-application-dir>/WEB-INF/classes/jira-application.proper ties file in your JIRA application installation directory. Alternatively, you can open the JIRA configuration tool to see the directory that is set as your JIRA Home.
3. Navigate to the directory specified in the configuration file and create a backup of it in another directory.
4. ❌ Delete the file <jira-home>/dbconfig.xml from the original folder as soon as the backup is complete.

2.4 Back up your attachments and index directories if located outside your JIRA home directory

If the attachments and index directories are located outside of your JIRA home directory, you must back them up separately. These pages describe how to find out where these directories are located in your implementation:

- Your attachments directory — Refer to Configuring file attachments page in the documentation for your version of JIRA.
- Your index directory — Refer to Search indexing page in the documentation for your version of JIRA. Also refer to Backing up data for more information about backing up attachments in JIRA.

2.5 Back up your JIRA installation directory

The 'JIRA Installation Directory' is the directory into which the JIRA application files and libraries were extracted when JIRA was installed.

3. Setting up your new JIRA installation

> If you are running a 'mission-critical' JIRA server, we highly recommend performing the remaining steps of this guide in a test environment (e.g. using a separate test JIRA database and a copy of your JIRA Home directory) before performing the upgrade in production.

3.1 Install the new version of JIRA

Download and extract the JIRA distribution you require to a new directory. **Do not overwrite your existing JIRA installation. Ensure this has been shut down and install the new JIRA version to a new location.**

Follow the installation instructions Installing JIRA applications.

3.2 Point your new JIRA to (a copy of) your existing JIRA Home directory

If your new JIRA 7.3 installation is on a new server, **copy the backup of your existing JIRA Home Directory** from the old server to the new server before proceeding.

To set up a distribution:

1. Open the JIRA configuration tool.
2. Click the **JIRA Home** tab.
3. Update the **JIRA Home Directory** field:
   - If your JIRA 7.3 installation is on a new server, update the **JIRA Home Directory** field to the path of your copied JIRA Home directory.
   - If your JIRA 7.3 installation is on the same server, update the **JIRA Home Directory** field to the path of your **existing** JIRA Home directory.
     For more information about this directory, see JIRA home directory.

You can also set your JIRA Home Directory's location by defining an operating system environment variable JIRA_HOME. This value of this variable takes precedence over the value of the jira.homeproperty in the jira-application.properties file in your JIRA installation directory. See Setting your JIRA home directory for details.

3.3 Connect the new version of JIRA to a new, empty database

Create a new, empty database that your new JIRA installation will use to store its data.

Jira Schemes and Workflow configurations

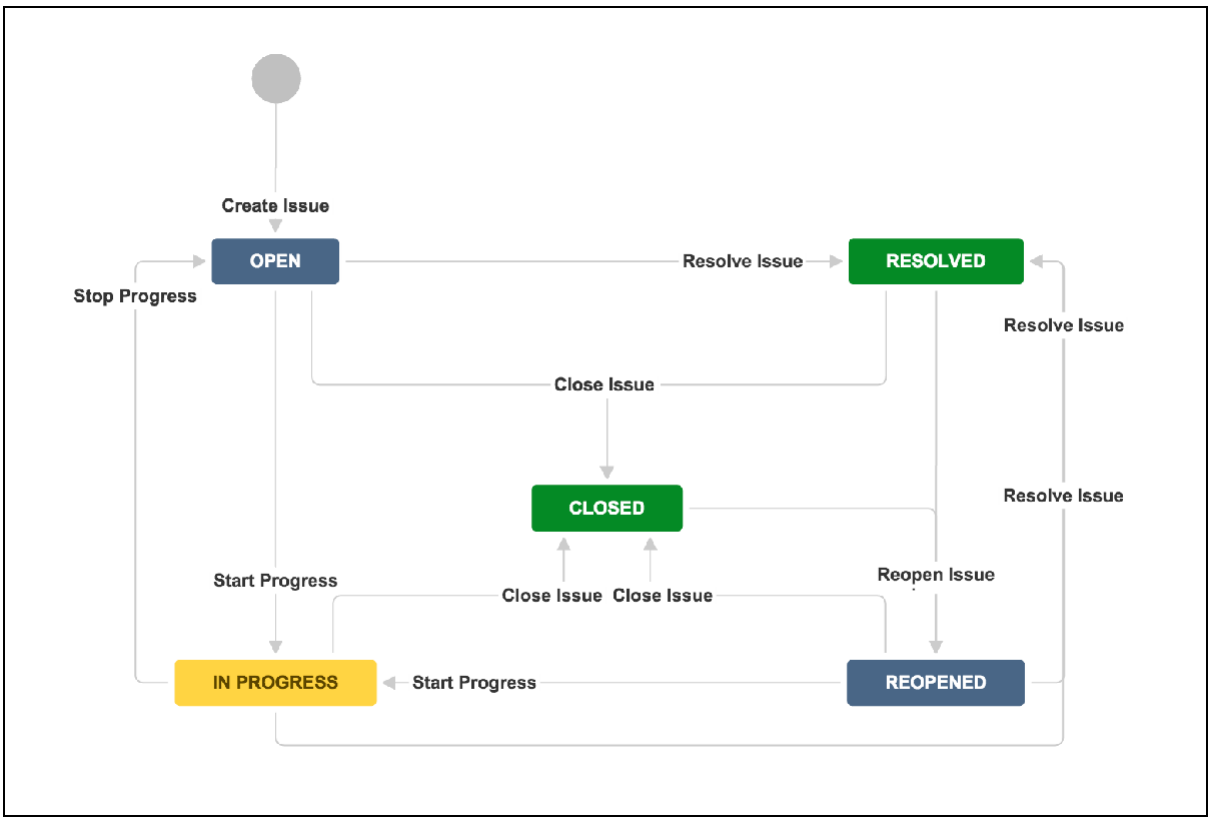
Working with workflows

A JIRA workflow is a set of *statuses* and *transitions* that an issue moves through during its lifecycle and typically represents processes within your organization. There are default built-in workflows that cannot be edited; however, you can copy and use these workflows to create your own.

You can also create your own workflows from scratch, or import workflows from Atlassian Marketplace. Workflows can be associated with particular projects and, optionally, specific issue types, by using a workflow scheme.

You will need to log in as a user with the 'JIRA Administrators' global permission to access and manage workflows. If the workflow isn't the default workflow, or shared with another project, project administrators also have limited editing permission to the workflow.

Here's an example of a default workflow:



Statuses and transitions

A status represents the state of an issue at a specific point in your workflow. An issue can be in only one status at a given point in time. When defining a status, you can optionally specify *properties*.

A transition is a link between two statuses that enables an issue to move from one status to another. In order for an issue to move between two statuses, a transition must exist.

A transition is a *one-way* link, so if an issue needs to move back and forth between two statuses, two transitions need to be created. The available workflow transitions for an issue are listed on the View issue screen.

Active and inactive workflows

There are slight differences between editing an inactive and an active workflow. We place restrictions on the modifications you can make to an active workflow, due to the impact the changes will have on projects and/or issue types that use this workflow.

Workflow status	Description
Inactive workflow	An *inactive workflow* is a workflow that is not currently being used by any projects. Because there are no issues currently transitioning through an inactive workflow, you can edit the workflow's steps and transitions directly. For details on this, see Working in text mode.
Active workflow	An active workflow is a workflow that is currently being used by one or more projects. When you edit an active workflow, JIRA first creates a draft of it, that you can then modify as you see fit. When you've finished, you can publish your draft and, optionally, save your original workflow as an inactive backup. The following limitations apply when editing the draft for an active workflow: Editing limitations... It is not possible to edit the workflow name (only the description) if a workflow is active. Workflow statuses cannot be deleted. The step ID cannot be changed. See Cannot Add Transitions or Delete Steps in Draft Workflows. To make any of the modifications listed above, you need to copy the workflow (see Creating a workflow), modify the copy, and then activate it.

Workflow designer

The workflow designer is a graphical tool that allows you to see the layout of your workflow and to create and edit a workflow's steps and transitions. You will need to log in as a user with the 'JIRA System Administrators' global permission to access the functionality described below.

With the workflow designer, you can:

- Manage status and transitions: add, click and drag, or select to edit properties (Workflow properties) to rename, or delete (from the workflow but not JIRA).
- Add a global transition that allows every other status in the workflow to
-

transition to the selected status. Select **Allow all statuses to transition to this one** in the properties panel for the transition. Change the screen that a transition uses. See Working in text mode for details.

- Configure advanced transition options, such as triggers, conditions, validators, and post functions. See the Advanced workflow configuration page.

Expand for workflow designer tips...

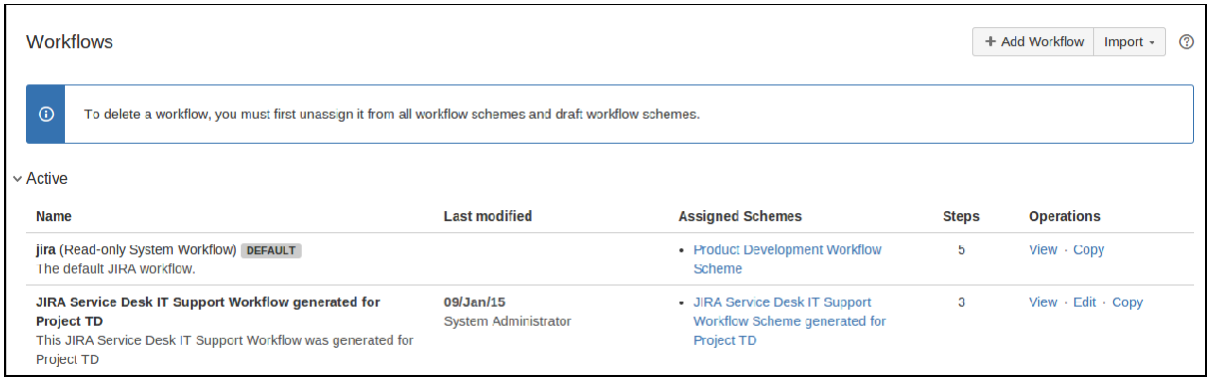
- Statuses are *global objects*. Changing the name of a status on one workflow also changes it in *all workflows that use that status*.
- Hover over a transition or a status to see the relevant transition labels.
- Zoom the diagram with your mouse wheel. Pan the diagram by clicking and holding the mouse while on white space, then moving your mouse across the diagram.
- You cannot clone transitions in the workflow
- designer. You cannot create annotations in the workflow designer.
- You cannot directly set the issue.editableproperty. To do this, simply add the issue.edita ble property to the status properties.
- The workflow designer will automatically validate your workflow and highlight any statuses that have no incoming or outgoing transitions. The workflow validator will also highlight all transitions that have an invalid permission condition that you don't have available in JIRA. The validator is particularly useful if you import workflows, or deal with complex workflows.

Creating workflows

There are a few ways you can start a new workflow. These include cloning an existing workflow, creating a new workflow, and importing a workflow.

Clone an existing workflow

1. Choose  > Issues.
2. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.



3. Copy an existing workflow using the **Copy** link in the Operations column (shown above). Enter a name and description and select the **Copy** button.

4. Customize it by adding or editing steps and transitions.

When you have finished customizing your workflow, see Managing your workflows for details on how to use it with a JIRA project.

Create a new workflow

For advanced administrators

1. Click **Workflows** in the left-hand nav panel, then **Add Workflow** at the top of the screen.
2. Enter a name and description for your workflow. Click **Add**.
   The workflow opens in edit mode, and contains a step called **Open** and an incoming transition called
   Create.
3. Continue with your workflow customizations, by adding and editing steps and transitions.

Import a workflow

Please see the documentation on Importing workflows.

Configuring a workflow

Editing a project's workflow

Whenever you create a new JIRA project, your project automatically uses the default workflow scheme. The scheme associates all available issue types in the project with the JIRA system workflow. Since neither the JIRA system workflow nor the default workflow scheme are editable, JIRA creates an editable copy of the system workflow and workflow scheme for your project.

1. Choose
   
   > Projects, and select the relevant project.
2. On the Administration page for the project, click **Workflows**.
3. Click the 'edit' icon at the top-right of the box, and JIRA automatically does
   - the following: Creates a draft copy of the system workflow named
   - '*Your Project Name* Workflow (Draft)'. Creates a new workflow scheme for the workflow named '*Your Project Name* Workflow Scheme'.
   - Associates any existing issues in your project with the new workflow.
4. You can now edit your draft workflow. Click on a status or transition to see editing options in the panel that appears.
5. When you are finished, click **Publish Draft**. The dialog allows you to publish your draft and, optionally, save your original workflow as an inactive backup.

Expand for performance notes about modifying workflows...

- The number of issues impacts the speed when configuring a workflow - for small numbers of issues, this process is relatively quick, however if you have many (e.g. thousands of) existing issues in your JIRA project, this process may take some time.
- Once this process begins, *it cannot be paused or canceled*. Please avoid editing or transitioning any issues within your project while this process is taking place.

In JIRA, an issue is either open or closes, based on the value of its 'Resolution' field — not its 'Status' field.

- An issue is open if its resolution field has not been set.
- An issue is closed if its resolution field has a value (e.g. Fixed, Cannot Reproduce).

This is true regardless of the current value of the issue's status field (Open, In Progress, etc). Therefore, if you need your workflow to force an issue to be open or closed, you will need to set the issue's resolution field during a transition. There are two ways to do this:

- Set the resolution field automatically via a post function.
- Prompt the user to choose a resolution via a screen. See Working in text mode for details on this.

Renaming workflow transition buttons

If you copied the system workflow and you wish to rename the workflow transition buttons on the View Issue page, you must delete the following properties from all transitions in the copied workflow:

- jira.i18n.title
- jira.i18n.description

Otherwise, the default names (i.e. values of these properties) will persist. Read more about transition properties.

Working in text mode

Text mode is an advanced way of working with workflows, and it shows the difference between steps and statuses. In text mode, you work directly with steps. For details, see Working in text mode.

Managing your workflows

Workflows need to be activated to use them in JIRA. Activating a workflow is the process of mapping the workflow to a workflow scheme, and then associating the workflow scheme with a project. To configure a workflow scheme, see Configuring workflow schemes.

A workflow scheme defines a set of associations – or mappings – between a workflow and an issue type. Workflow schemes are associated with a project and make it possible to use a different workflow for every combination of project and issue type.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Activating a workflow

Active workflows are those that are currently being used, while inactive workflows are those that are not associated with any workflow schemes, or are associated with workflow schemes that are not associated with any projects. Active workflow

schemes are also those associated with projects, while inactive workflow schemes are not.

1. Create a workflow scheme or find an existing workflow scheme. See Configuring workflow schemes fo r instructions.
2. Configure the workflow scheme to use your workflow. See Configuring workflow schemes for instructions.
   Associate your workflow scheme with a project, as described in the Associating a workflow scheme with a project section below.

Managing workflows for projects

You can manage your workflows by associating workflow schemes, importing, exporting, uploading, and sharing.

Associating a workflow scheme with a project

You can associate a single workflow scheme with more than one project, although only one workflow scheme can be associated with a given project. The issue type scheme associated with a project defines the issue types that are available to that project. If an issue type is not defined in the project's issue type scheme, its workflow is not used.

1. Choose  > **Projects**, and select the relevant project. The Project Summary page is displayed.
2. Click **Workflows** on the left of the Project Summary page (you can also click the **More** link in the Workflows section in the middle of the screen). This is the current workflow scheme used by the project.
3. Click the **Switch Scheme** link to display the Associate Workflow Scheme to Project page.
4. Select the relevant workflow scheme from the Scheme list and click the **Associate** button to begin the migration process.
   Each issue has to be in a valid status. The valid statuses for an issue are defined by its workflow. This means that when changing a workflow, you may need to tell JIRA the status for specific issues after the change.
5. A screen displays that indicates the progress of migrating all the project's issues to the updated scheme's workflows. **Acknowledge** to finish the process.

Disassociating a workflow scheme from a project

A JIRA project must always be associated with a workflow scheme, since all issues must move through a workflow, even if that workflow only consists of a single *Create Issue* transition. By default all JIRA projects with unmodified workflows use JIRA's system workflow. *Disassociating* a workflow scheme re-associates your project's workflow with JIRA's default workflow scheme.

1. Follow the instructions in Associating a workflow scheme with a project above.
2. When selecting the workflow scheme from the Scheme list, select the **Default** workflow scheme

3. Click the **Associate** button, and follow the wizard, which guides you through migrating all of the project's issues.

Exporting your workflow

The workflow sharing feature allows you to share your team's workflow with other teams in your organization on different JIRA instances, or external parties in other organizations via the Atlassian Marketplace. This feature allows you to easily share and use workflows that other people have published, or to move a workflow from staging to production in your own organization. you wish to share your JIRA Workflow with another instance of JIRA or upload it to the Atlassian Marketplace, you first need to download it.

1. Choose  > Issues.
2. Find the workflow you wish to share by clicking on the Workflows section in the left-hand panel.
3. Click **View** or **Edit** under the Operations column.
4. Select **Export > As Workflow** and click **Next** to continue.
5. In the Add Notes field, add any special configuration notes; for example, information about plugins that should be installed. JIRA auto-populates these notes for you when it discards parts of your workflow (for example, plugins, post functions, conditions, validators).
6. Click **Export** and select a download location. Ensure the location is publicly accessible.

Uploading to Atlassian Marketplace

To share your workflow with other JIRA users, upload it to the Atlassian Marketplace. Create an account on Atlassian Marketplace, or log in and choose **Manage Add-ons**

1. Click **Create new add-on**.
2. Choose **My add-on is not directly installable** (ensure that 'Add-on Type' is listed as 'Not a Plugin'). You will need to host the workflow on your own servers, and add information about where the workflow export can be accessed in the Binary URL textbox. This should be the location you specified in step 6 of the prior instruction set.
3. Fill out the submission form, be sure to note the following:
    a. The Summary field contains the information that will be displayed to users searching the Marketplace.
    b. The Category for your workflow must be Workflow Bundles. Choosing Workflow Bundles ensures other JIRA users will have visibility to your workflow.
    c. The Add-on Key must be unique, as it uniquely identifies your application; it will become the application URL.

You don't have to complete the form in one session. You can save your form and come back to it later. Once you accept the Atlassian Marketplace Vendor Agreement, the system submits your add-on for review by Atlassian's Developer Relations team.

Importing workflows

Custom fields in workflow imports

If your workflow contains custom fields that are disabled, the workflow importer will not create these fields unless they are enabled before importing. You will receive a warning about this. To fix this, you need to enable the missing custom fields before proceeding with the import.

1. Click on the highlighted **Custom Field Types & Searchers** plugin in the displayed warning. This opens the plugin in a new window and scrolls to the right place to make the necessary changes.
2. Click to expand the list of enabled modules.
3. Find the modules that are disabled and enable them.

After enabling the corresponding modules of the Custom Field Types & Searchers plugin, return to the summary page and proceed. You may need to refresh the page first. For information on installing add-ons, see Viewing installed add-ons.

Importing from Atlassian Marketplace

This procedure covers importing a workflow from Atlassian Marketplace.

1. Choose  > Issues.
2. Click on the **Workflows** section in the left-hand panel.
3. Select **Import > Import Workflow** in the top right of the screen.
4. The **From Atlassian Marketplace** option should be selected by default.
5. Find the workflow you want and click the **Select** button.
6. Follow steps 5 through 8 of the **Importing from a local instance** procedure.

Importing from a local instance

This procedure covers importing a workflow from a local instance. You must be logged in as System Administrator to perform this function.

1. Click on the **Workflows** section in the left-hand panel.
2. Select **Import > Import Workflow**.
3. Select a workflow from your computer to upload, and then click **Next**.
4. JIRA automatically generates a workflow name, but you can change this if you like. Click **Next**.
5. Next, you are presented with a screen that details your workflow statuses, as shown below. You can map the steps of the workflow to your existing workflow statuses or create new statuses at this point. When you are finished, click **Next** to continue.
6. At the Preview of Import screen, click **Import** at the bottom of this screen to accept the changes and import the workflow.

Your workflow is imported and you are presented with a screen with additional configuration details

Click **Done** to exit this process.

All custom fields will have brand new custom fields created. This is regardless of a custom field of the same name / type already existing. See:

JRA-37358 - Workflow import creates duplicate custom fields

OPEN

for the request to improve this.

Configuring workflow schemes

A workflow scheme defines a set of associations – or mappings – between a workflow and an issue type. Workflow schemes are associated with a project and make it possible to use a different workflow for every combination of project and issue type.
By default, projects use JIRA's system workflow. The default workflow scheme:

- Associates JIRA's system workflow *jira* with all issue types (available to the JIRA project).
- Appears as the default workflow scheme for your selected project type.

In addition, you can share an existing project's workflow scheme when you are creating a new project by selecting **Create with shared configuration** i n the Project Creation Wizard. This allows you to reuse your existing schemes without having to recreate them for new projects. Keep in mind that changing shared workflow schemes will affect all projects that are using that theme.

This page describes how to configure workflows and issue type workflow associations in the scheme.

To associate a workflow scheme with a project (part of activating a workflow), see Managing your workflows.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Adding a workflow scheme

1. Choose  > **Issues**. Select **Workflow Schemes** to open the Workflow Schemes page.
2. Click the **Add Workflow Scheme** button.
3. Enter the name and description of the new workflow scheme.
4. Click the **Add** button. The new workflow scheme is created.
5. Follow the instructions in Configuring workflows for a workflow scheme below.

Configuring workflows for a workflow scheme

If your scheme is associated with a project, follow the instructions in Configuring a workflow scheme associated with a project. Otherwise, follow the instructions in Configuring a workflow scheme outside of a project.

Configuring a workflow scheme associated with a project

> JIRA's default workflow scheme cannot be modified. If you attempt to modify it, a copy of the scheme is created with the name of the project you are administering. You cannot configure a workflow scheme shared by multiple projects using this method; follow the instructions in Configuring a workflow scheme outside of a project instead.

1. Choose  Projects
2. Select a project from the displayed list.
3. Click **Workflows** on the left of the Project Summary page. The Workflows page is displayed, indicating the current workflow scheme used by the project. Configure the workflow scheme using the table below.
4. At the Publish Workflows screen, click **Associate** to begin the migration process. Each issue has to be in a valid status. The valid statuses for an issue are defined by its workflow. This means that when changing a workflow, you may need to tell JIRA the status for specific issues after the change.
5. A screen displays that indicates the progress of migrating all the project's issues to the updated scheme's workflows.
6. Click **Acknowledge** to finish the process. A message displays letting you know that 'your workflows have been published.'

Configure the issue types for the workflow scheme as desired. This is not the same as editing the workflow (clicking the **Edit** button in the workflow diagram at the center of your screen). If you do that, you will be asked to publish your *draft workflow scheme*.

What do you want to do?	Instructions
Add a workflow to the scheme	1. Click **Add Workflow**, and select **Import From Bundle** or **Add Existing**. After selecting Import From Bundle, you can select a workflow **From Atlassian Marketplace**. See Sharing your workflow for more information. 2. Select the desired workflow and issue types.
Edit a workflow	Hover over the desired workflow and click the **Edit** button. See Working with workflows for further instructions. The **Edit** button only displays if you have the edit permission.
Remove a workflow from the scheme	Click the cross icon under Operations to remove the workflow from the scheme.
Change the issue types associated with a workflow	1. Click the **Assign** link under Issue Types for the desired workflow. 2. Select the desired issue types in the dialog that appears. 3. Click **Finish**.
View the text-based representation	Hover over the desired workflow, and click the **View as Text** link.

of a workflow	
Change the workflow scheme associated with the project	Click the **Switch Scheme** button next to the scheme name. See Managing your workflows for further instructions.

Configuring a workflow scheme outside of a project

You can use this procedure to edit any workflow scheme in the system, including those shared by multiple projects. The workflow scheme can be either active or inactive.

- If your workflow scheme is associated with a project, you may want to follow the instructions above ins tead. When a workflow scheme is used by more than one project, you must use this configuration method.
- When a workflow scheme is active, it creates a *draft workflow scheme* when you edit it.

1. Choose  > **Issues**. Select **Workflow Schemes** to open the Workflow Schemes page.
2. Click the **Edit** link under the Operations column for the desired workflow.
3. Edit your workflow scheme, as described in the table below.
4. If your workflow is active, you need to publish it to make your changes active.

What do you want to do?	Instructions
Add a workflow to the scheme	1. Click **Add Workflow**, and select **Import From Bundle** or **Add Existing**. After selecting Import From Bundle, you can select a workflow **From Atlassian Marketplace**. See Sharing your workflow for more information. 2. Select the desired workflow and issue types.
Remove a workflow from the scheme	Click the **Remove** link in the Operations column.
Change the issue types associated with a workflow	1. Click the **Assign** link under Issue Types for the desired workflow. 2. Select the desired issue types in the dialog that appears. 3. Click **Finish**.
View a representation of a workflow	Click either the text or diagram link next to the Workflow name.

Remove an issue type from the scheme	Click the **x** next to the name of the issue type to remove it.

Editing, copying, and deleting workflow schemes

Choose  **> Issues**. Select **Workflow Schemes** to open the Workflow Schemes page.

Operation	Instructions
Edit the name and description of a workflow scheme	Click the **Edit** link. Use inline edit mode – click in the associated field – to update the name and description.
Copy a workflow scheme	Click the **Copy** link to create a workflow scheme with the prefix "Copy of (name of current workflow)" and placed in the inactive workflow schemes.
Delete a workflow scheme	Click the **Delete** link and confirm the deletion. You cannot delete an active workflow scheme. You must first disassociate it from all projects.

Sharing your workflow

The new Workflow Sharing feature allows you to share your team's workflow with other teams in your organization on different JIRA instances, or external parties in other organizations via the Atlassian Marketplace. This feature allows you to easily share and use workflows that other people have published, or to move a workflow from staging to production in your own organization.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Exporting your workflow

If you wish to share your JIRA Workflow with another instance of JIRA or upload it to the Atlassian Marketplace, you first need to download it. Follow this procedure.

1. Choose  > Issues.
2. Find the workflow you wish to share by clicking on the Workflows section in the left-hand panel.
3. Click **View** or **Edit** under the Operations column.
4. Select **Export > As Workflow**.
5. Click **Next** to continue.

6. In the Add Notes field, add any special configuration notes; for example, information about plugins that should be installed. JIRA auto-populates these notes for you when it discards parts of your workflow (for example, plugins, post functions, conditions, validators).
7. Click **Export** and select a download location. Ensure the location is publicly accessible.

Uploading to Atlassian Marketplace

To share your workflow with other JIRA users, upload it to the Atlassian Marketplace.

1. Create an account on Atlassian Marketplace.
2. Log in to the Atlassian Marketplace and choose **Manage Add-ons**. See this page for more details: Ste p-by-step Paid-via-Atlassian Listing.
3. Click **Create new add-on**.
4. Choose My add-on is not directly installable.
5. Ensure 'Add-on Type' is listed as 'Not a Plugin.'
6. You will need to host the workflow on your own servers, and add information about where the workflow export can be accessed in the Binary URL textbox. This should be the location you specified in step 7 of the prior instruction set.
7. When you fill out the submission form, be sure to note the following:
    a. The Summary field contains the information that will be displayed to users searching the Marketplace.
    b. The Category for your workflow must be Workflow Bundles.
    Choosing Workflow Bundles ensures other JIRA users will have visibility to your workflow.
    c. The Add-on Key must be unique.

Importing from Atlassian Marketplace

1. Choose

> Issues.
2. Click on the Workflows section in the left-hand panel.
3. Select **Import > Import Workflow** in the top right of the screen.
4. The **From Atlassian Marketplace** option should be selected by default.
5. Find the workflow you want and click the **Select** button.
6. Follow steps 5 through 8 of the 'Importing from a local instance' procedure.

Importing from a local instance

This procedure covers importing a workflow from a local instance. For importing from Marketplace, see the procedure above, **Importing from Atlassian Marketplace**. You must be logged in as System Administrator to perform this function.

1. Click on the Workflows section in the left-hand panel.
2. Select **Import > Import Workflow**.
3. Select a workflow from your computer to upload, and then click **Next**.
4. JIRA automatically generates a workflow name, but you can change this if you like. Click **Next**.
5. Next, you are presented with a screen that details your workflow statuses, as shown below. You can map the steps of the workflow to your existing workflow

statuses or create new statuses at this point. When you are finished, click **Next** to continue.

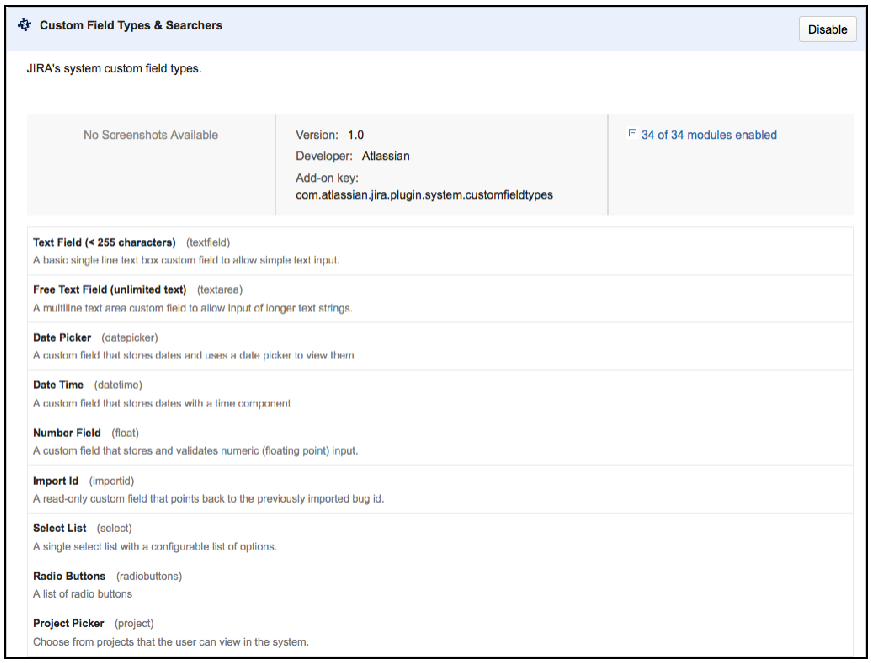
6. You will be presented with a screen that presents a summary of the workflow changes, as shown below. Click **Import** at the bottom of this screen to accept these changes and import the workflow.
7. Your workflow is imported and you are presented with a screen with additional configuration details. Click **Done** to exit this process.
   All custom fields will have brand new custom fields created. This is regardless of a custom field of the same name / type already existing. See:
   JRA-37358 - Workflow import creates duplicate custom fields
   OPEN
   for the request to improve this.

Custom fields in workflow imports

If the workflow that you are importing contains custom fields that are disabled, the workflow importer will not create these fields unless they are enabled before importing.

You will receive a warning about this. To fix this, you need to enable the missing custom fields before proceeding with the import.

1. Click on the highlighted Custom Field Types & Searchers plugin in the displayed warning. This opens the plugin in a new window and scrolls to the right place to make the necessary changes:



1. Click to expand the list of enabled modules.
2. Find the modules that are disabled and enable them.

After enabling the corresponding modules of the Custom Field Types & Searchers plugin, return to the summary page and proceed. You may need to refresh the page first.

Advanced workflow configuration

This page describes configuring transitions in JIRA workflows. For information about the basics of workflows
– see Working with workflow.

As a JIRA administrator, you can control the following aspects of a transition's behavior:

- Triggers – transition JIRA issues when certain events occur in a connected development tool, such as Atlassian's Bitbucket or Stash.
- Conditions – check that a transition should be performed by the user.
- Validators – check that any input to the transition (for example, by a user) is valid, *before* the transition is performed.
- Post functions – carry out additional processing, *after* a
- transition is performed. Properties – are key-value pairs that can be used to further customize transitions.

Triggers

JIRA administrators can configure triggers in JIRA workflows that respond to events in your linked development tools. This allows you to set up your development tools and JIRA workflows so that, for example, when a developer creates a branch to start work on an issue in Atlassian's Bitbucket or Stash, the issue will automatically be transitioned from 'Open' to 'In progress'.

If you haven't set up a trigger before or you want to learn about triggers in more detail, see our guide on triggers here: Configuring workflow triggers. The guide also shows you how to configure a workflow with triggers, similar to this sample development workflow: Development Workflow with Triggers (from Atlassian Marketplace).

Configure triggers

To see, or to set, triggers for a transition, edit the workflow that contains the transition, select the transition, then click **Triggers** in the properties panel for the transition.

To add a trigger to a transition:

1.
2. Log in as a user with the 'JIRA Administrators' global permission.
3. Choose  > **Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
4. Click **Edit** for the workflow that has the transition you wish to change.
5. In the Workflow Designer, select the transition.
6. Click **Triggers** in the properties panel to show the triggers configured for the transition.
7. Click **Add trigger** on the **Triggers** tab to configure a trigger.

On the **Conditions** tab, you can see any conditions that have already been set.

When you click **Add condition**, you can choose from the available conditions, and set any necessary parameters for the condition. Additional conditions may be available from installed plugins. or you can create your own conditions using the plugin system; see the Workflow Plugin Modules for details.

Note that you can also edit the transition in 'text' mode.

Grouping conditions

You can construct complex conditions by grouping and nesting conditions. Change any condition into a group by clicking the 'Add grouped condition' icon for the condition. Now you can add further conditions to this new group, as described above.

You can toggle the logic for how the conditions in a group are applied between **All** and **Any.**

Validators

Validators check that any input made to the transition is valid *before* the transition is performed. Input can include that gathered from the user on the transition's screen.

If a validator fails, the issue does not progress to the destination status of the transition, and the transition's p ost functions are not executed.

Adding a validator

To add a validator to a transition, edit the workflow that contains the transition, select the transition, then click **Validators** in the properties panel for the transition.

Not sure about that? Click here to see how...

To add a validator to a transition:

1. Log in as a user with the 'JIRA Administrators' global permission.
2. Choose 
   > Issues. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click **Edit** for the workflow that has the transition you wish to change.
4. In the Workflow Designer, select the transition:
5. Click **Validators** in the properties panel

On the **Validators** tab, you can see any validators that have already been set.

When you cllck **Add validator** you can choose from the available validators and set any necessary parameters for the validator.

Note that you can also edit the transition in 'text' mode.

Post functions

Post functions carry out any additional processing required after a transition

- is executed, such as: updating an issue's fields
- generating change history for
- an issue adding a comment to
  an issue
- generating an event to trigger email notifications

The following sections

- describe: Essential
- post functions
  Optional post
  functions
- Using post functions with the initial
- transition Using a post function to
  set a field
- Using a post function to send HipChat
- notifications Using a post function to
  send email notifications

Essential post functions

Every JIRA transition has the following essential post functions, which are performed in this order:

1. Set issue status to the linked status of the destination workflow status.
2. Add a comment to an issue if one is entered during a transition.
3. Update change history for an issue and store the issue in the database.
4. Reindex an issue to keep indices in sync with the database.
5. Fire an event that can be processed by the listeners.

These essential post functions cannot be deleted from a transition or reordered. However, you can insert other (optional) post functions between them.

Optional post functions

Optional post function	Description
Assign to Current User	Assigns the issue to the user who is executing the transition. This post function is ignored unless the user has the Assignable User permission. Create a condition to give the logged-in user this permission before executing the transition.
Assign to Lead Developer	Assigns the issue to the component lead, if one exists, or project lead.
Assign to Reporter	Assigns the issue to the user who created the issue.
Create Perforce Job Function	Creates a Perforce Job (if required) after completing the workflow transition.
Notify HipChat	Sends a notification to one or more HipChat rooms. See Using a post function to send HipChat notifications for more information.
Trigger a Webhook	Triggers the specified webhook after completing the workflow transition. When you add this post function, you will be asked to specify a webhook. This webhook must already be defined in JIRA (see Managing webhooks).
Update Issue Field	Updates one of the issue's fields to a given value. Fields that can be updated include: Assignee Description Environment Priority Resolution Summary Original Estimate Remaining Estimate This post function cannot update custom fields and must be positioned after the other optional post functions.

To add a post function to a transition, edit the workflow that contains the transition, select the transition, then click **Post functions** in the properties panel for the transition.

Not sure about that? Click here to see how...

To add a post function to a transition:

1. Log in as a user with the 'JIRA Administrators' global permission.
2. Choose  **> Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click **Edit** for the workflow that has the transition you wish to change.
4. In the Workflow Designer, select the transition:
5. Click **Post functions** in the properties panel.

On the **Post functions** tab, you can see any post functions that have already been set. When you click **Add**

post function you can choose from the available post functions, and set any necessary parameters. Options for editing or deleting a post function, and for changing the execution order, are at the right of the tab (hover there to see them).

Note that you can also edit the transition in 'text' mode.

Using post functions with the initial transition

You can add post functions to a workflow's initial transition when you need to perform processing tasks – such as setting a particular field's value – when an issue is created. The initial transition is called 'Create' (if you created a blank workflow) or 'Create Issue' (if you copied the system workflow).

JIRA includes the following essential post functions that are specific to a workflow's initial transition and that are performed in this order:

1. Create the issue.
2. Fire an event that can be processed by the listeners.

The following optional post functions are available specifically for the initial transition:

Optional post function (initial transition only)	Description
Create Comment	Adds a comment to an issue if one is entered during a transition.
Update Issue Status	Sets the issue's status to the linked status of the destination workflow status.
Store Issue	Stores updates to an issue (no change history is created).

Additionally, the standard optional post functions can also be added to an initial transition,

Optional post functions added to the Create transition must be placed *before* the

'Create the issue originally' post function.

If you wish, you can configure the initial status for your workflow to go to a different initial transition. See Conf iguring the initial status for details.

Notes

If you need to set the 'Resolution' field when creating an issue, add the 'Update Issue Field' post function *afte r* the 'Create the issue' post function and *after that*, use the 'Store Issue' post function. The 'Store Issue' post function is useful for setting the Resolution field during issue creation.

However, only use the Store Issue post function where

- necessary, since it: does not generate change history
- is unable to persist fields that have a one-to-many relationship with the issue (for example, 'Version' or
  'Component')

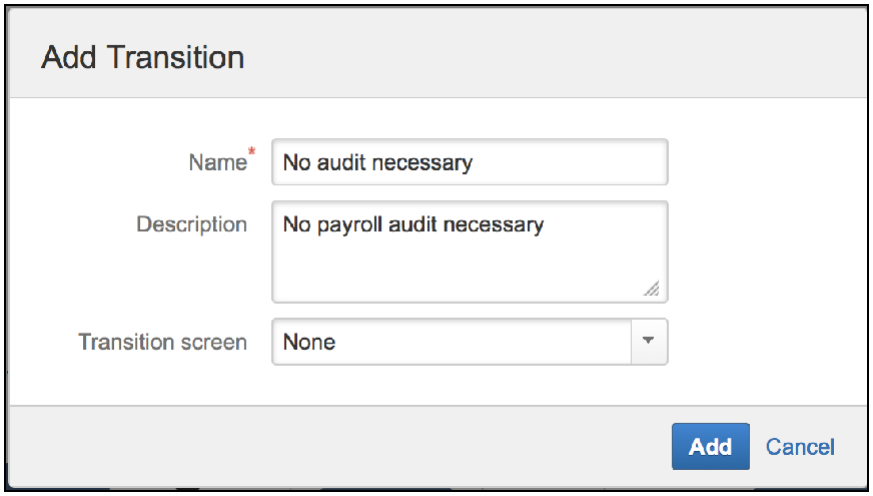
Using a post function to set a field

You can use the 'Update Issue Field' post function to set the value of an issue's field after a particular transition is executed.

For example, you might want a transition that moves the issue to a *closed* status to automatically set the 'Resolution' field.

Example: Using a post function to set the Resolution field:

1. Edit the workflow that has the transition, and drag from status to another to create a new transition.
2. Select either **None** or a screen that does not contain the **Resolution** field:



3. Add a new post function of type 'Update Issue Field' and:
   a. Select **Resolution** from the **Issue Field** list.
   b. Select a suitable resolution from the **Field Value** list.

To create a transition that clears the **Resolution** field, follow the same steps above

for adding an 'Update Issue Field' post function to your transition. However, select **None** from the **Field Value** list.

The list of post functions for this transition includes the

- following statement: The **Resolution** of the issue will

  be **cleared**.

Each time one of these transitions is executed, the **Resolution** of the issue is automatically set or cleared, as specified in these post functions.

Using a post function to send HipChat notifications

You can use a 'Notify HipChat' post function to send a notification to one or more HipChat rooms whenever an issue passes through a transition with this post function. You can also add a JQL query to the 'Notify Hipchat' post function to filter for the issues that will trigger the HipChat notification.

To send HipChat notifications:

1. Create or edit your transition.
2. Add a new post function of type 'Notify HipChat'.
3. On the 'Add Parameters to Function' page:
   a. Optionally, specify a JQL query. Only issues that match the query will send notifications. Leave this field empty to send notifications to *all* issues that pass through this transition.
   b. Select the HipChat rooms you want to link with your workflow transition.

Using a post function to send email notifications

Use the 'Fire an event that can be processed by the listeners' post function to fire the 'Generic Event', which is a built-in JIRA event that can be used to trigger the sending of email notifications after a particular transition is executed.

Alternatively, you could fire a custom event that you've created

specifically for this transition. When a transition is performed, JIRA will:

- Look up the notification scheme associated with the issue's project and identify the users associated with the fired event;
- Send an email notification to each user.

The fired event is also propagated to all registered listeners.

Example: Using a post function to fire the Generic Event to send email notifications:

1. Create or edit your transition.
2. Click the transition's **Post Functions** tab and edit the 'Fire an event that can be processed by the listeners' post function.
3. Select **Generic Event** from the list of events.

Transition properties

Properties are key-value pairs that can be used to further customize transitions. For example, transition properties can help to extend a copied system workflow to allow language translations.

To view and edit the properties of a transition:

1. Select a transition in the diagram.
2. Click **Properties** in the Properties panel.
3. Either:
   - Add a new property to the transition.
   - Delete a property, by clicking the icon to the right of the property.

> **Important**
> It is not possible to edit a transition's properties on this page. To change any property's key or value (or both), you must first delete the property you wish to change and add the new updated property.

Note that you can also edit the transition in 'text' mode.

It is possible to implement restrictions on transitions using transition properties. For more information, see W orkflow properties.

Customize how transitions appear

When viewing an issue, most of the operations and workflow transitions are available from a row of buttons at the top of the issue.

To change the number of transition buttons from the default of two:

By default, the first two transitions appear as separate buttons in the set of transition buttons. Additional transitions appear in the **Workflow** menu. The order in which these buttons appear is based on the order defined in the system workflow.

1. Shutdown JIRA.
2. Edit the jira-config.propertiesfile in your JIRA application home directory. See Making changes to the jira-config.propertiesfile for more information.
3. Change the value of 'X' in the ops.bar.group.size.opsbar-transitions = Xproperty of this file to be the number of transition buttons required *before* the **Workflow** menu.
   If this property does not exist in your jira-config.propertiesfile, add it. Otherwise, a default value of 2 is assumed.
4. Save the updated jira-config.propertiesfile.
5. Restart JIRA.

To change the order of transition buttons:

To change the order of transition buttons, including additional transitions in the **Workflow** menu, add the property key opsbar-sequenceto each workflow transition that you wish to reorder. Each opsbar-seque nceproperty key requires a property value that defines the order of the transition action on issue views.

1. Go to the transition's properties, as described in Transition properties above.
2. Type opsbar-sequenceinto the **Property Key** field, under 'Add New Property'.

3. Type a value In the **Property Value** field, The value must be a positive integer (starting at '0'); it defines the order of the transition buttons on issue views.
   Consider using a sequence of opsbar-sequenceproperty values like 10, 20, 30... to allow new transitions to be easily added later.
4. Click **Add**.

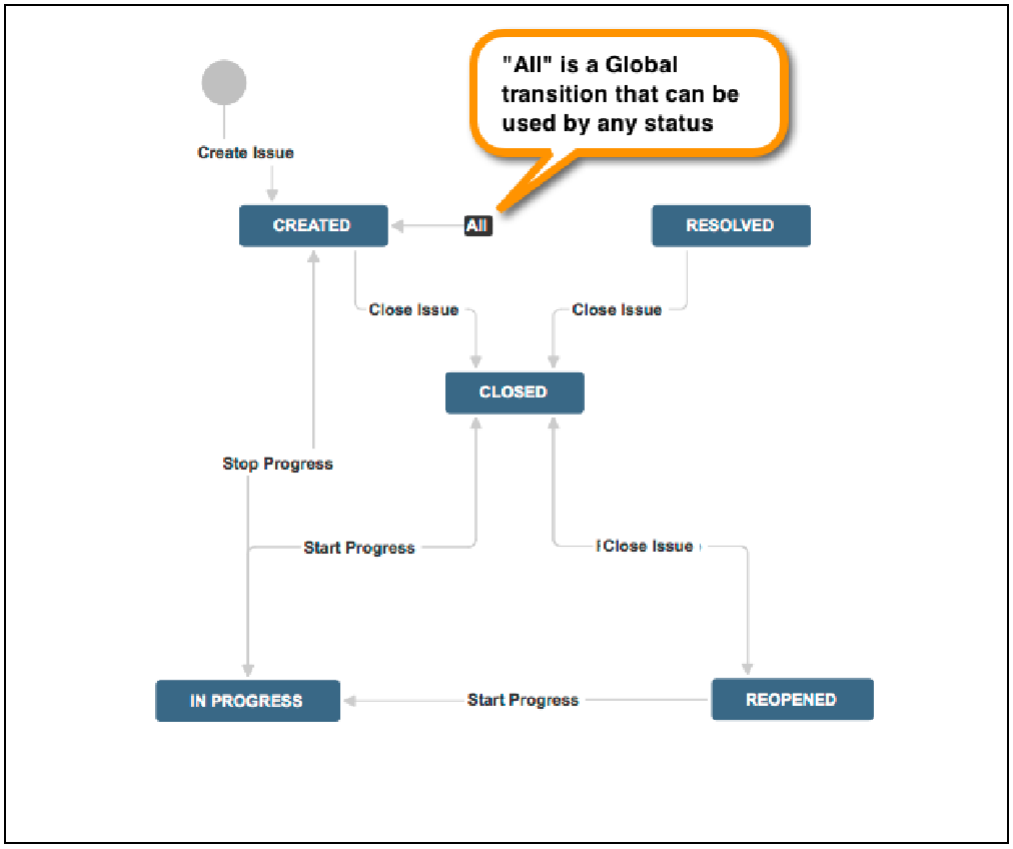
Adding the opsbar-sequence property to a workflow transition does not change the order of these transitions in the workflow in Text edit mode. The addition of this property only affects the order of transitions on the **View issue** page.

Global transitions

Global transitions allow any status in a workflow to transition to

a particular status. You can add a global transition:

- When creating a new status (adding an existing status) – check the **Add global transition to status** o ption.
- By selecting a status and checking **Allow all statuses to transition to this one** in the properties panel for the status.



To create two global transitions that point to the same destination step:

1. From the workflow designer, create the first global transition as normal by selecting a step and checking "Allow all statuses to transition to this one"
2. Create the second global transition on any *other* step that does not currently have a global transition pointing to it
3. Then from text editor, select the second global transition that you created
4. Click on the 'Edit' button and change the 'Destination Step' to the same step that you selected for your first global transition, and then click 'Update'

Integrating with collaboration tools

Integrating with Confluence

Give your team the ability to share, discuss and work with JIRA application issues in Confluence, and create knowledge articles for your service desk customers. Here are some of the ways you can benefit from integrating Confluence with your JIRA applications:

For...	You can...
Bugs	Create a knowledge base article to document a workaround for a bug.
New Features	Create a product requirements document for a new feature.
Self-service	Create knowledge articles that customers can view on the customer portal to find solutions themselves
General JIRA Use Case	Document and collaborate with your team on an issue in Confluence.

And here are just a few of the things Confluence

- allows you to do: Share pages
- Watch pages
- Create knowledge articles from service desk issues
- Collaborative commenting, especially through the use of @mentions
- Form a team network and let them know what you are doing
- via a status update Add images, picture galleries, videos, and more
- Enable various content macros

Integrating Jira and Confluence

Jira applications and Confluence complement each other. Collect your team's thoughts, plans and knowledge in Confluence, track your issues in your Jira application, and let the two applications work together to help you get your job done.

Use Jira applications and Confluence together

Confluence and Jira are like bacon and eggs; coffee and cake; Simon and Garfunkel. Separately, they're great, but together, they're amazing!

If your Confluence and Jira sites are connected using Application Links, you can display and create Jira issues and more from within Confluence.

What you can do with Confluence and Jira depends on the Jira application and version you have.

For every project or team Display issues on a page

You can display Jira issues on a Confluence page using the Jira Issues macro. Display a single issue, a list of issues, or show the total number of issues.
The simplest way to add a Jira issue to Confluence is to paste a Jira URL on a Confluence page.

Here's some examples...

<yourjirasite.com>/browse/CONF-1234 will insert the Jira Issues macro and display a single issue.

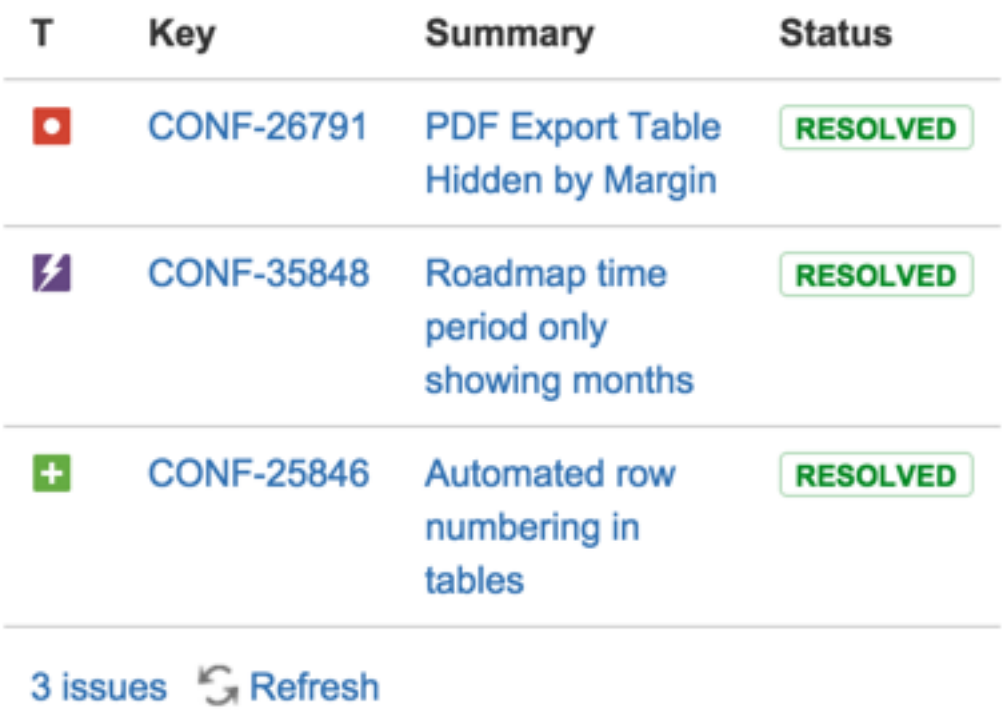
<yourjirasite.com>/issues/?filter=56789 will insert the Jira Issues macro and display a list of issues matching the saved filter.

<yourjirasite.com>/issues/?jql=project%20%3D%20CONF will insert the Jira Issues macro and display a list of issues matching the Jira search.

Alternatively, you can add the Jira Issues Macro to the page and search for issues directly:

In the editor choose Insert > Jira Issue.

Follow the prompts in the macro browser to choose a project and search for an issue – you can even use JIRA Query Language (JQL).
Once you've added the macro, you can customize how the issue or list of issues appears on the page, including how much information to display, how many issues, and more.



Create reports and charts

Reporting on information stored in Jira is simple in Confluence. In addition to the Jira Issues Macro, you can use the Jira Report blueprint or Jira Chart macro to show information from your Jira application visually. It's the best way to give your stakeholders a snapshot of your team or project's progress.

You can:

Use the JIRA Report blueprint to create a Change Log or Status report.

Use the Jira Chart Macro to display data as a chart, including pie charts, created vs resolved, and two dimensional charts.

Use JIRA Gadgets to display detailed Jira reports and charts on pages.

Create issues from inside Confluence

You can create issues while viewing a page or from the within the editor. This is really useful if you use Confluence for planning and gathering requirements.

To create an issue when viewing a page:

Highlight some text on your page and choose the Create Jira issue icon that appears above the highlighted text.

Enter your server (if you have multiple Jira sites connected to Confluence), project, issue type and description. Your highlighted text will populate the issue summary automatically.

Choose Create.

The issue will be created in Jira and added to your page. If your text is in a table, you'll have the option to create multiple issues using text from the same column.

If you don't see a popup when you highlight text, check that Text Select is enabled in your profile settings.

To create an issue in the editor:

In the editor choose Insert > Jira Issue > Create new issue.
Enter your server (if you have multiple Jira sites connected to Confluence), project, issue type, summary, and description.

Choose Insert.

The issue will be created in Jira and added to your page.

There are some limitations when creating Jira issues from Confluence. The Jira Issues macro or Create Jira Issue dialog will notify you if it's unable to create an issue in the selected project. You can find out more in the Jira Issues Macro page.

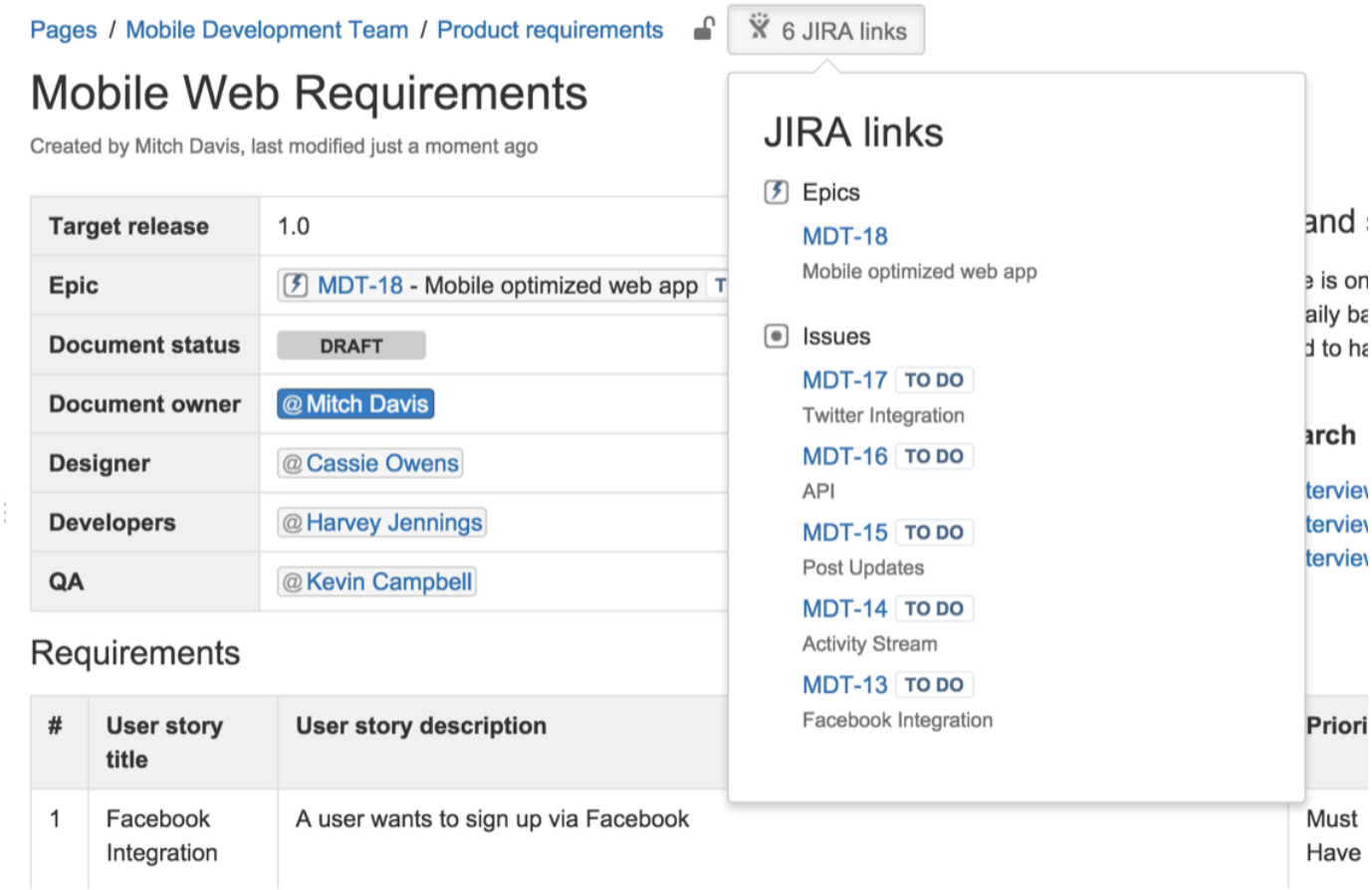
Move between Jira and Confluence

Whenever you add a link to JIRA issues in Confluence, or link to a Confluence page from your Jira application, the Jira Links button appears at the top of the Confluence page. This makes it really easy to jump from Confluence to Jira and vice versa, speeding up your workflow.

The number on the Jira Links button indicates the total number of issues, epics, and sprints connected to that page, regardless of whether you have permission to view them. The

dropdown, however, will only show details of issues, epics, and sprints that you have Jira permissions to view.

The button doesn't detect links from issues displayed in the Jira Issues macro in table format.



For software teams

Here's some suggestions to help you get the most out of Confluence and Jira Software and unleash the potential in your agile development team.

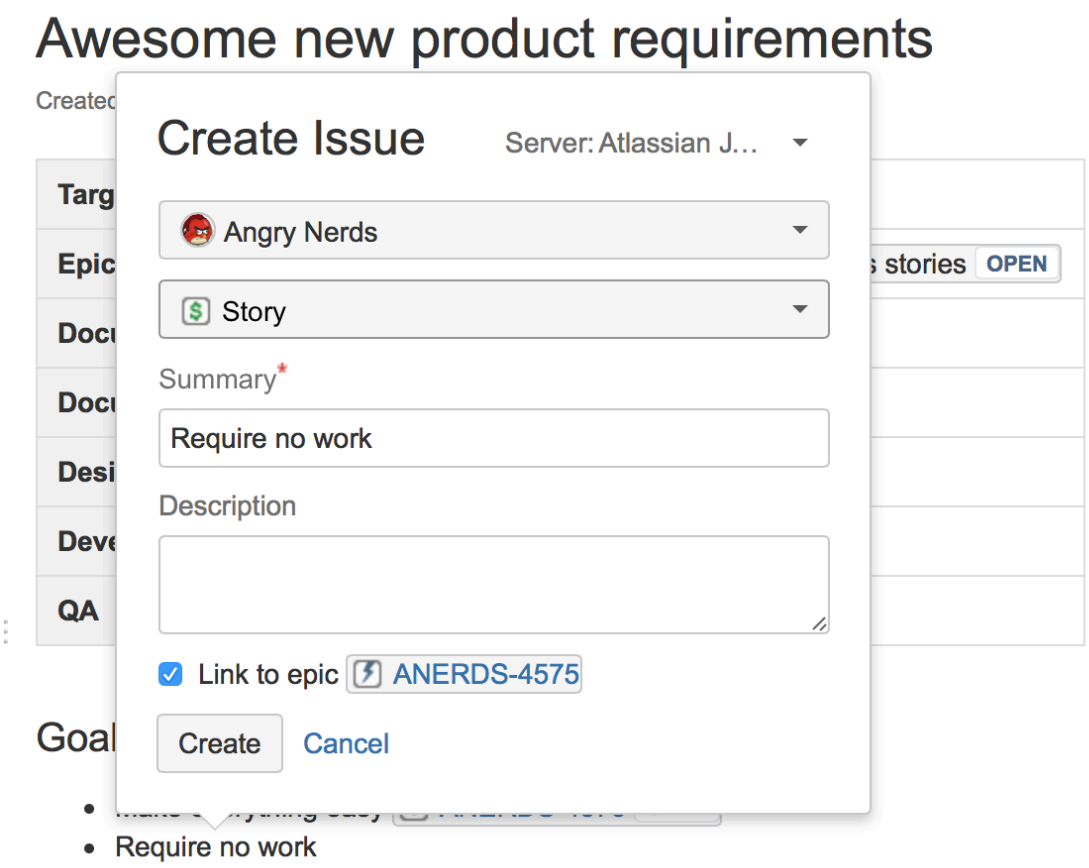
Define your requirements

Confluence is the perfect place to start defining your requirements. You can use the Product Requirements Blueprint to capture your requirements, then create your Jira epic and other issues right from the requirements page in Confluence.

Here's how it works:

1. Create a Confluence page using the Product Requirements Blueprint.
2. Choose the placeholder text '*Link to Jira epic or feature*' and choose **Create new issue** to create your epic in Jira.
3. Collaborate with your team to define your stories and save the page.

4. Highlight text on your requirements page and choose the **Create Jira issue** link to create stories in Jira, and automatically link them to your epic.
5. Track the progress of the stories from the Confluence page or from within Jira.



The tight integration between Confluence and Jira Software means you can easily access issues from the Confluence page and see their status at a glance, and from within Jira Software you can see links to related Confluence pages. All the information you need is right there.

Manage your sprints

There's often a lot of material in Confluence that provides useful context for your team during a sprint. These might be requirements documents, designs, tech specs, customer research and more. By linking these pages to epics, you make them easy for your team to find during the sprint.

Here's how you can use Confluence to support your sprint from within Jira Agile:

- In Jira Software, create a Confluence page to plan your sprint. The page is created using the Meeting Notes Blueprint – a handy template that helps capture the details you need – and is automatically linked to the sprint.
- In an epic, link to useful Confluence pages, including requirements, designs, and more.
- Report on your progress to stakeholders using the JIRA Report blueprint in Confluence.

- Use the Retrospective Blueprint in Confluence at the end of your sprint to take stock of what went well and not so well.

For people who work mostly in Jira Software, the integration means that useful Confluence pages are only a click away.

Jira integration in Crucible

When Crucible is integrated with Jira Software, you and your team get all the benefits described on this page:

In Crucible, you can:

- See all the Crucible reviews related to a Jira Software issue
- Create a Crucible review directly from an issue in Jira Software
- Link your Crucible review to a Jira Software issue
- Create a Jira Software issue from a review comment
- Transition Jira Software issues automatically
- Transition Jira Software issues from within Crucible
- See issues from multiple instances of Jira Software

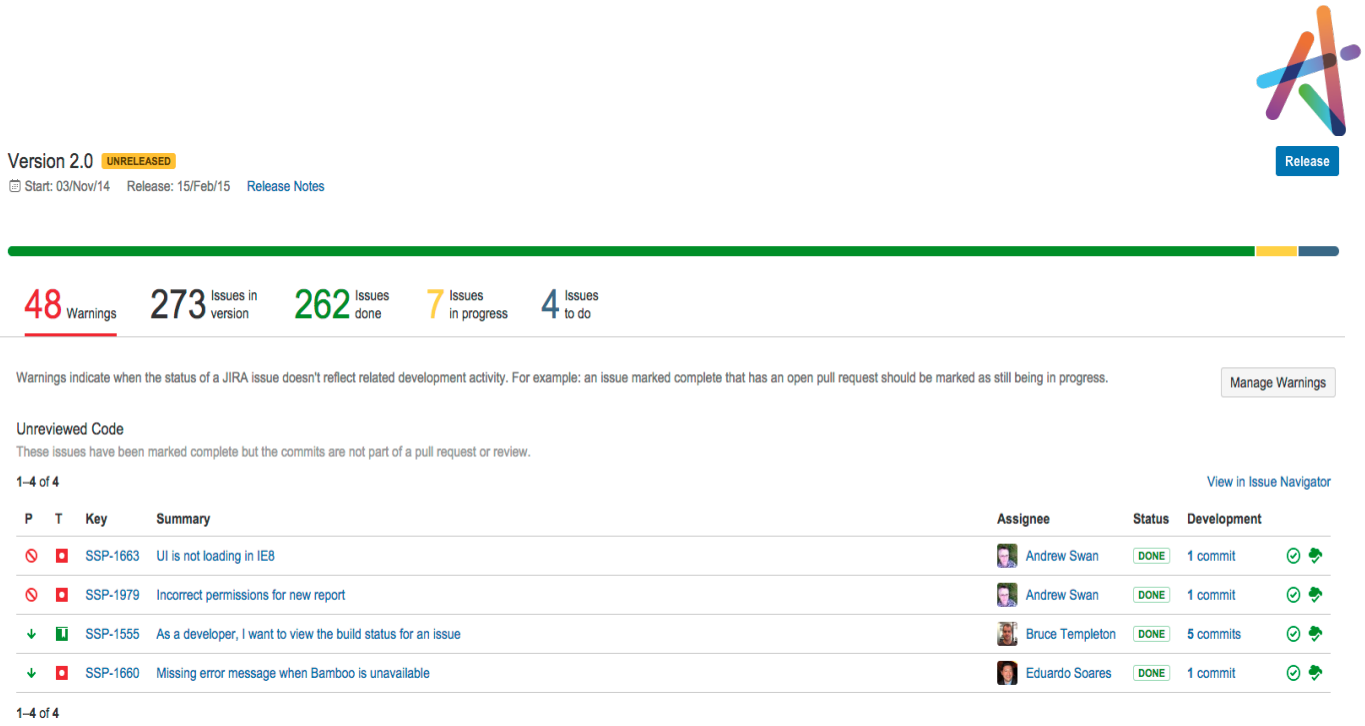
See open reviews or unreviewed commits for an entire version within Jira Software

Note that your Crucible and Jira Software instances must be linked to make use of these Jira Software integration features.

Check development progress of a version in Jira Software

CRUCIBLE 3.3+ JIRA 6.4+

The Release Hub in Jira Software shows the relevant issues and development information for a version – so you can determine which issues are likely to ship at a glance. With Jira Software and Bitbucket Server connected, the release page can warn you about open reviews or unreviewed commits that could cause problems for your release.



From the Release Hub you can also:

- Release a version
- Mark a version as complete
- Move incomplete issues to other versions
- Trigger release builds (if Jira Software is connected to Bamboo)
- Warnings that help you reconcile what is happening in development with Jira data.

To view the Release Hub (with the project sidebar enabled), navigate to a project, click on **Releases**, then select a version listed.

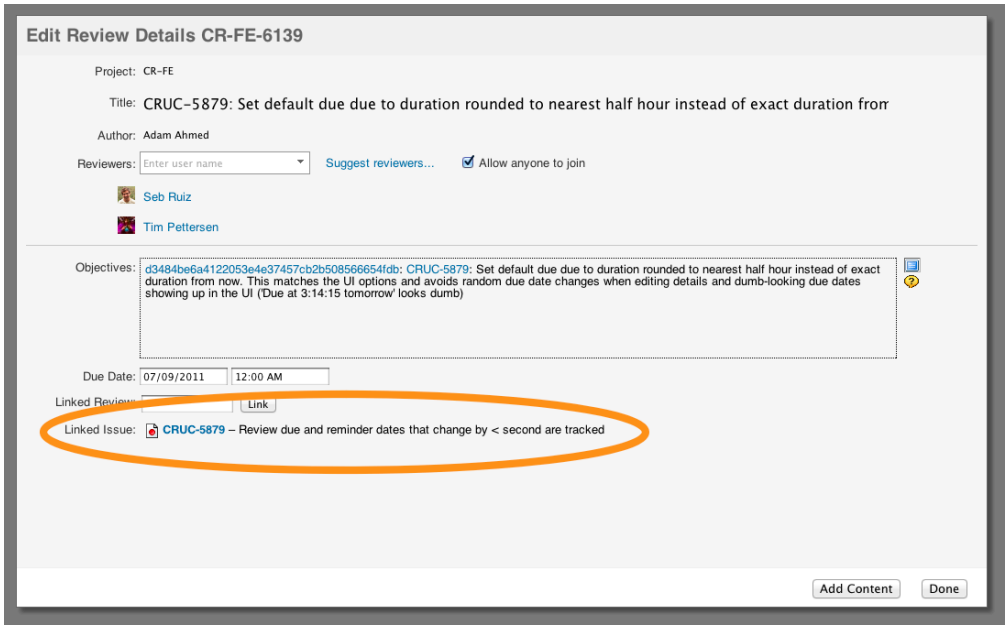
See all the Crucible reviews related to a Jira Software issue

In a Jira Software issue, the Development panel shows the number of reviews that are linked to the issue. Click the **reviews** link to see details of those reviews.

Link your Crucible review to a Jira Software issue

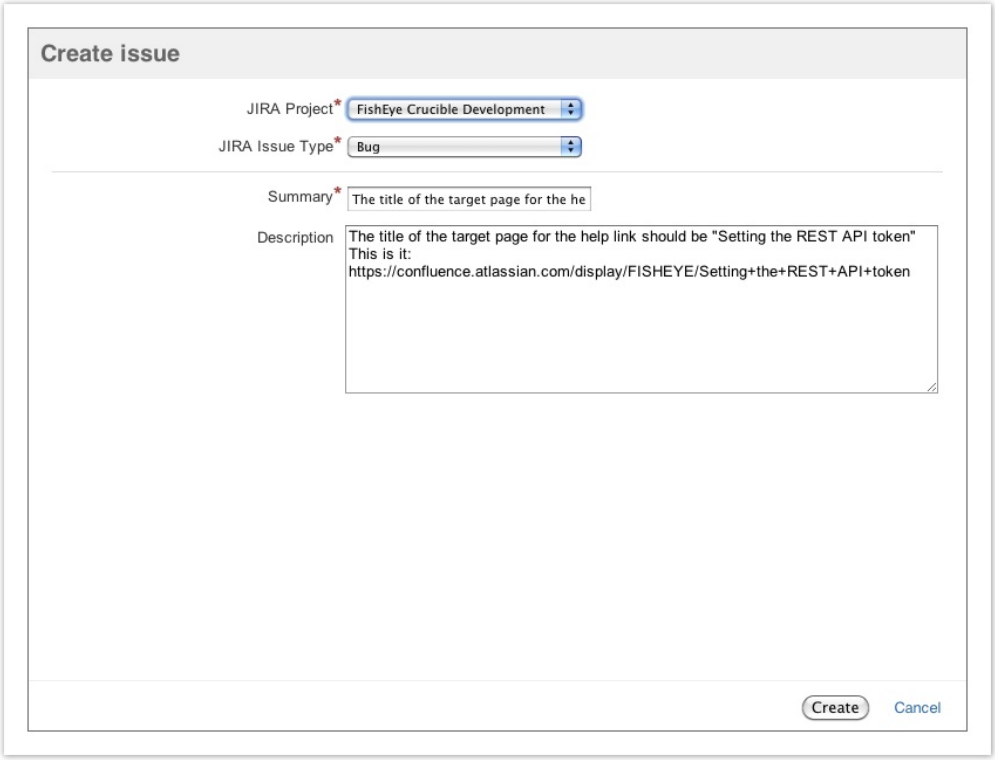
When creating, or editing, your review, Crucible will suggest a Jira Software issue that can be linked to the review, if a Jira Software issue key is found in the review title. You can:

- click the suggested Jira Software issue key, to link it to the review
- delete the suggested Jira Software issue and specify a different issue key and click **Link** to save it.



Create a Jira Software issue from a review comment

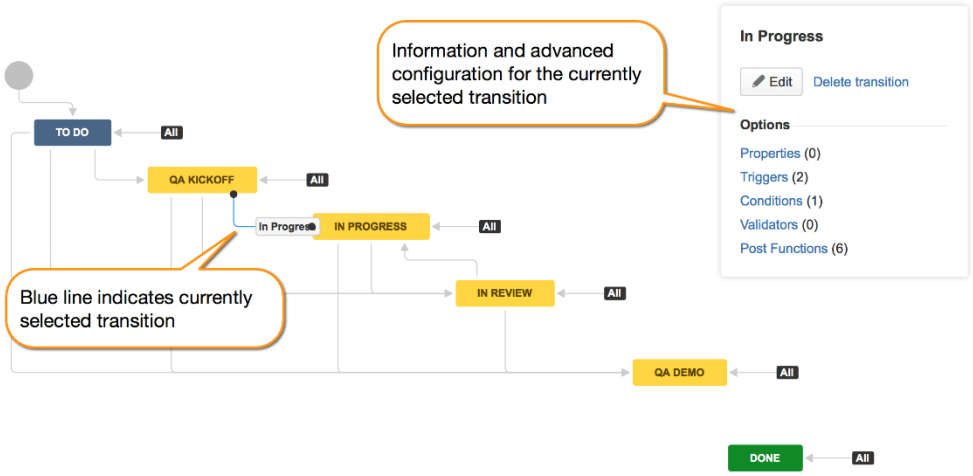
When viewing any review comment (genaral, file, inline), you can click **Create Issue** in the comment to create a Jira Software issue. Crucible suggests the Jira Software instance, project and issue type, but you can modify these. This requires Jira 5.0, or later, and is disabled if Crucible is integrated with an earlier version of Jira Software.



Transition Jira Software issues automatically

Your Jira Software workflow can now respond to events in your linked development tools, for when a review is started, your Jira Software workflow can be configured to automatically

transition the related issue. Configure this from transitions within the Jira Software workflow editor. (Available with Jira 6.3.3 and later.)



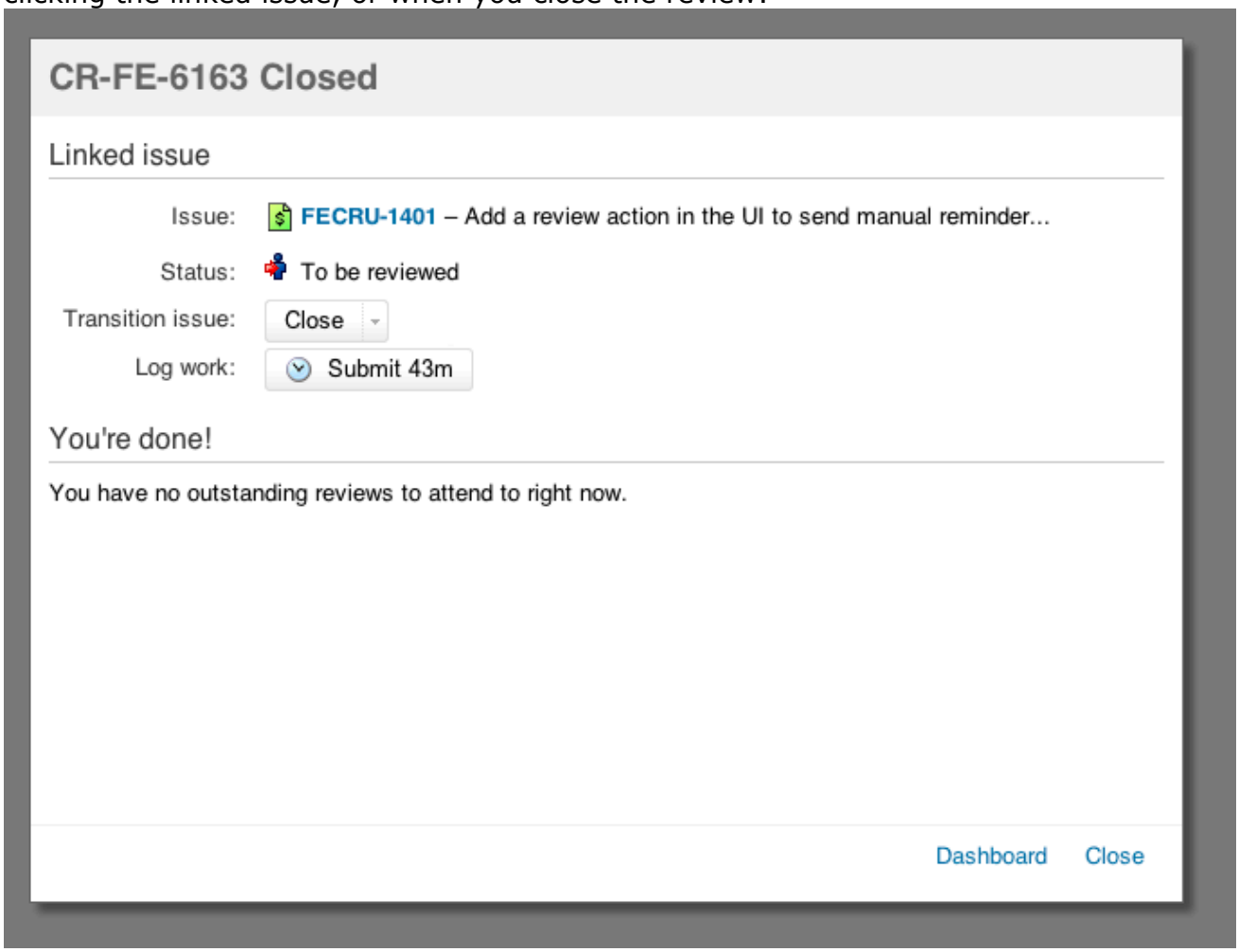
The events available in Crucible are:

- Review started
- Submitted for approval
- Review rejected
- Review abandoned
- Review closed
- Review summarized

Transition Jira Software issues from within Crucible manually

For Crucible reviews that have linked Jira Software issues, you can advance the Jira Software workflow for the issue from within Crucible. You can do this at any time by

clicking the linked issue, or when you close the review:



Jira Integration in Fisheye

When Fisheye is integrated with Jira Software, You can also use Jira Software for delegated management of your Fisheye users.

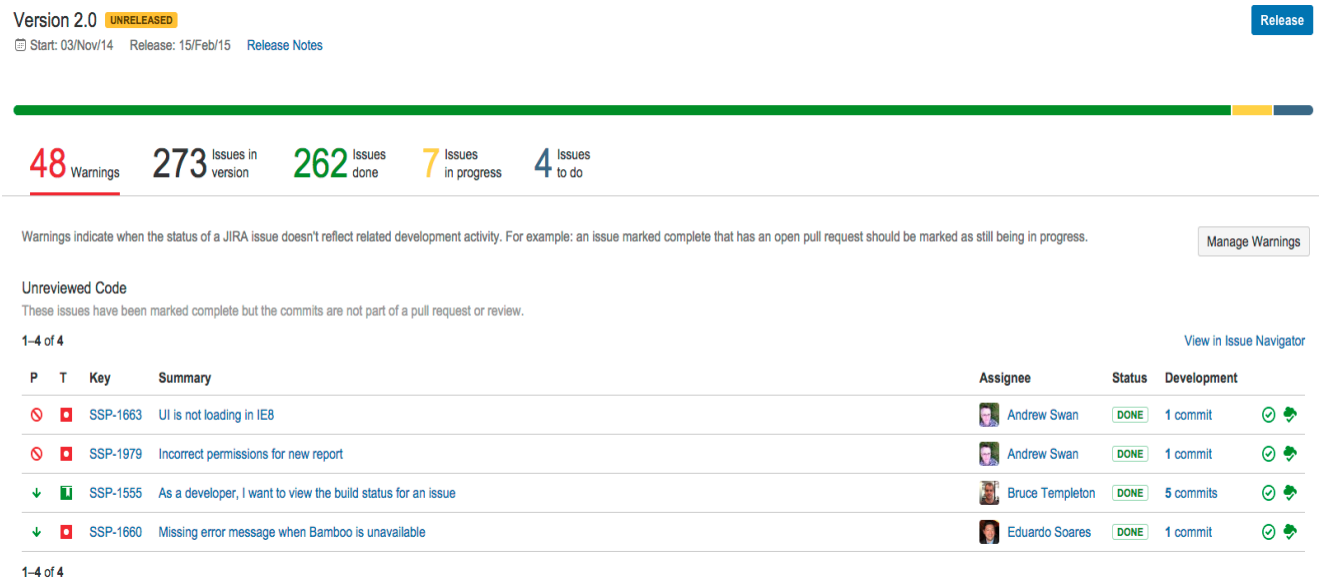
Starting in Jira 6.2.2 the Source and Reviews tabs are only displayed if Jira Software is unable to display the associated information in the Development Tools panel.

```
Your user tiers don't need to match between Jira Software and Fisheye/Crucible in order to integrate them. Jira Software users that are not Fisheye users will see the same view as Fisheye users within Jira Software, but will not be able to log in to Fisheye to view the source/reviews.
```

Check development progress of a version in Jira Software

FISHEYE 3.3+ JIRA 6.4+

The Release Hub in Jira Software shows the relevant issues and development information for a version – so you can determine which issues are likely to ship at a glance. With Jira Software and Fisheye connected, the release page can warn you about potential development issues that could cause problems for your release.



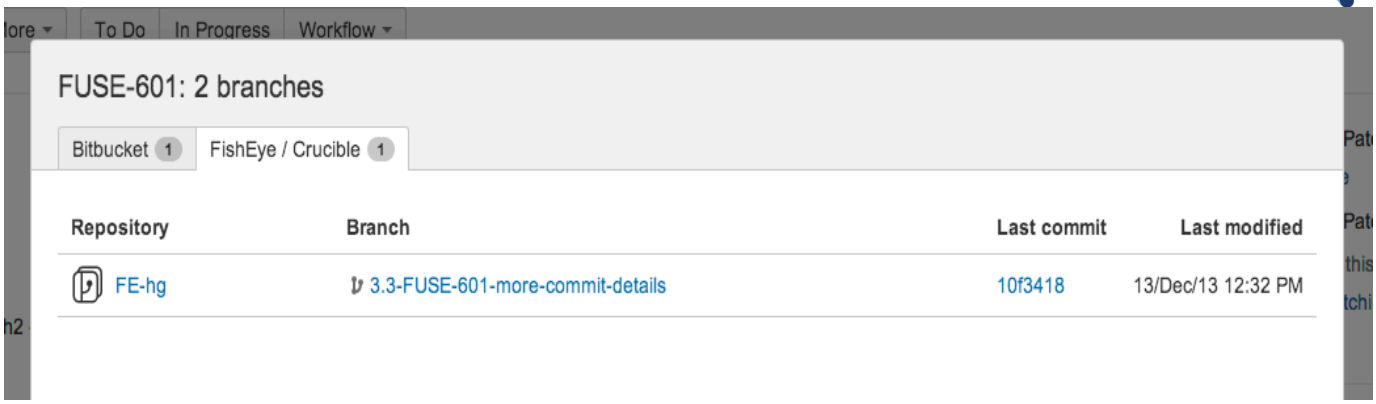
From the Release Hub you can also:

- Release a version
- Mark a version as complete
- Move incomplete issues to other versions
- Trigger release builds (if Jira is connected to Bamboo)
- Warnings that help you reconcile what is happening in development with Jira Software data.

Jira Software the Fisheye repository branches related to an issue

FISHEYE 3.3+ JIRA 6.2+

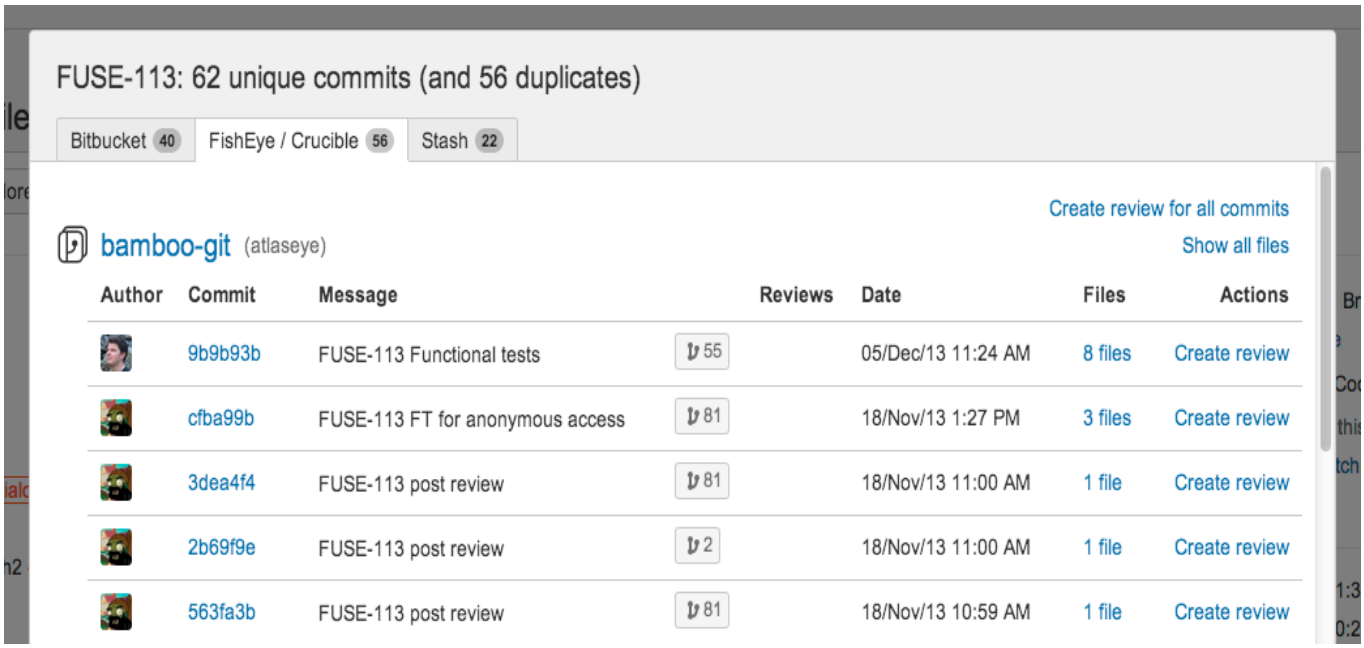
For Fisheye 3.3 and later, the Fisheye repository branches related to a Jira Software issue are summarized in the Development panel for the issue, when Jira Software and Fisheye are connected with an application link. To see details of the branches, simply click the **branches** link. You can see which repository each branch is in and when the last commit was made. As long as the issue key is included in the branch name the branch is automatically linked to the Jira Software issue.



Jira Software the commits related to an issue

FISHEYE 3.3+ JIRA 6.2+

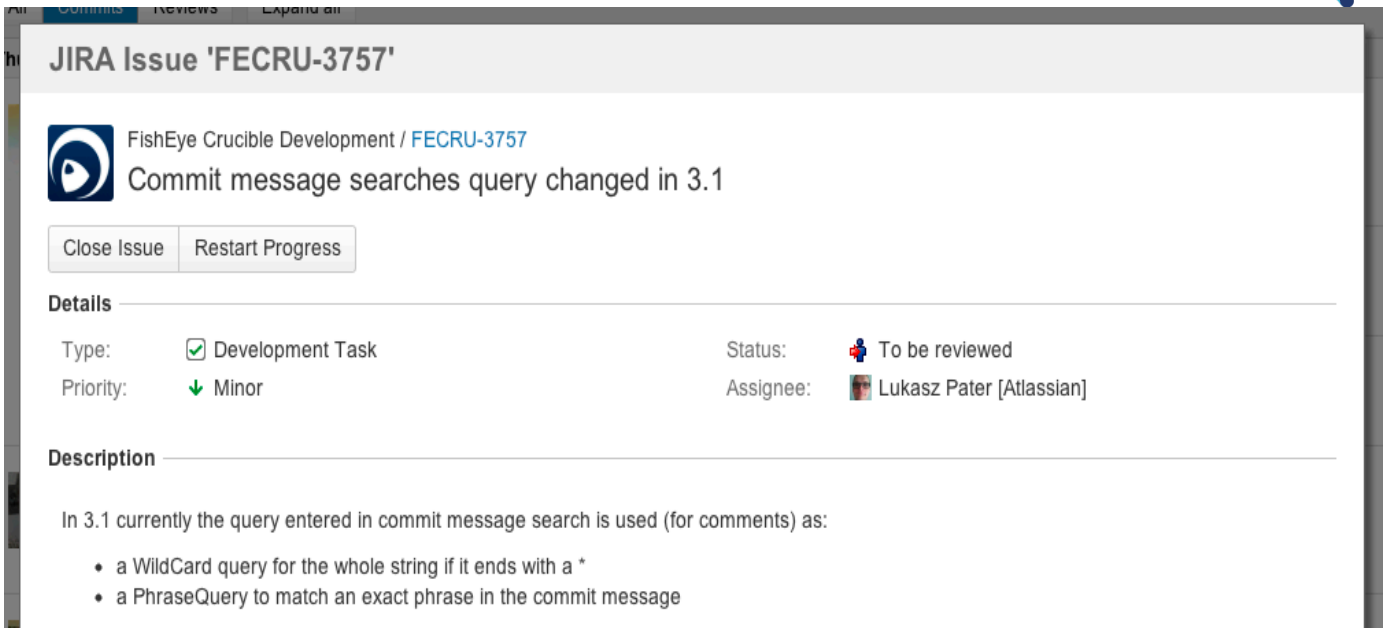
For Fisheye 3.3 and later, the Fisheye repository commits related to a Jira Software issue are summarized in the Development panel for the issue, when Jira Software and Fisheye are connected with an application link. You can click the **commits** link to see detailed information such as who made each commit, when they commited, and how many files were changed. Click through to see a particular commit in the Fisheye instance where the commit was made. A developer only needs to add the issue key to the commit message for that commit to be automatically linked to the Jira Software issue.



Jira Software issues related to commits

FISHEYE 3.1+

Fisheye recognizes Jira Software issue keys, and displays those as links in places such as the activity stream, side-by-side diffs, and commit messages:



Click on the linked issue key to see details for the issue, as described next.

See the details for Jira Software issues

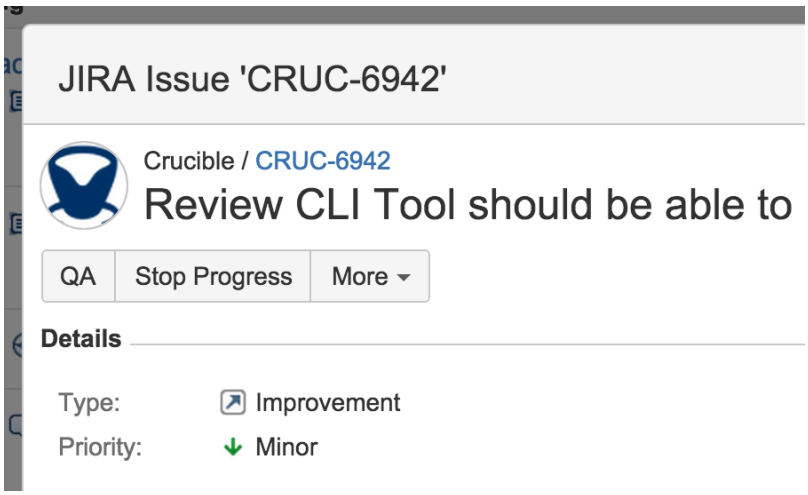
FISHEYE 3.1+

Click a linked issue key anywhere in Fisheye to see the details of that issue in a dialog. And you can click the issue key at the top of the dialog to go straight to the issue in Jira Software:

Transition Jira Software issues from within Fisheye

FISHEYE 3.1+

You can easily transition a Jira issue from within Fisheye. For example, when viewing a commit, you may want to transition the related Jira Software issue into QA. Click on a linked Jira issue anywhere in Fisheye to see a dialog with the available workflow steps:



Click on a step in the dialog, and complete any displayed fields as required. If there are custom required fields that are unsupported by Fisheye, just click **Edit this field in Jira** to transition the issue directly in Jira.

Issues from multiple instances of Jira Software

FISHEYE 3.1+

Fisheye can link to more than one Jira server at a time, so different teams can work with their own projects in different Jira Software instances, or a single team can link to issues across multiple Jira Software servers.

Thank You