

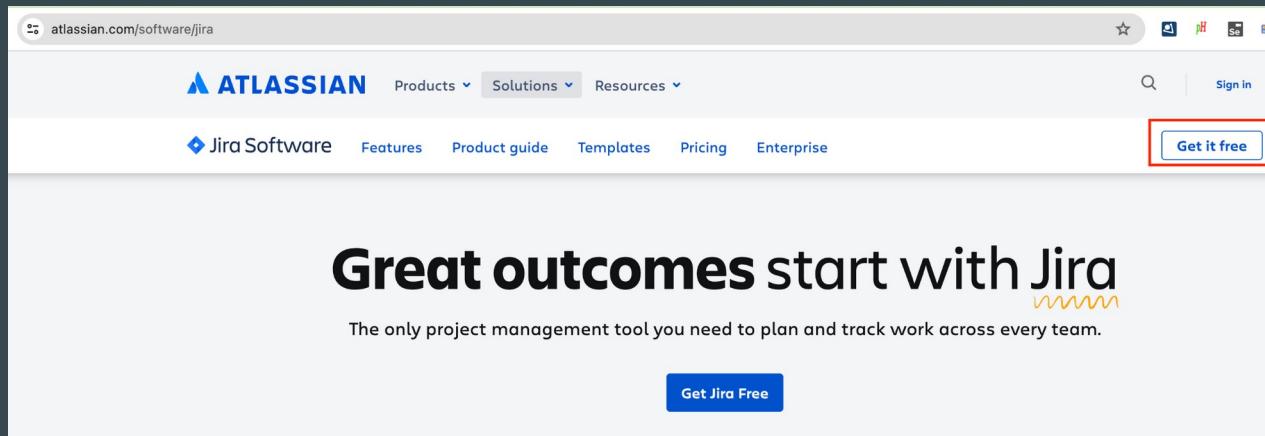
Jira and Xray cloud setup

...

Quick setup instructions of a Jira Software + Xray cloud instance

Jira cloud setup

<https://www.atlassian.com/software/jira>



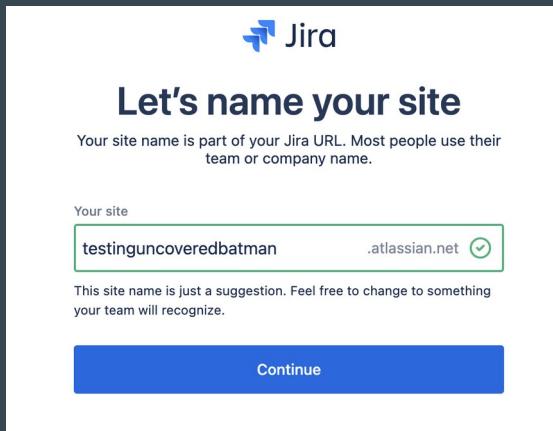
Jira cloud signup

The image shows three sequential screenshots of the Jira Cloud sign-up process:

- Initial Sign-up Screen:** The Jira logo is at the top. The heading "Get started with Jira" is displayed. A subtext says "It's free for up to 10 users — no credit card required". Below is a "Work email" input field containing "testing.uncovered+batman@gmail.com". A note below the input says "Find teammates, plus keep work and life separate by using your own email." At the bottom, there's a checkbox for "I agree to the [Atlassian Customer Agreement](#), which references the [AI Product-Specific Terms](#), and acknowledge the [Privacy Policy](#)." A large blue "Sign up" button is at the bottom.
- Verification Code Email:** The Jira logo is at the top. The heading "Your verification code is:" is followed by the code "513 137". Below it, a note says "If you didn't try signing up, you can safely ignore this email." At the bottom, there's a note "This message was sent to you by Atlassian Cloud" and the "ATLASSIAN" logo.
- Verification Code Entry Screen:** The Jira logo is at the top. The heading "We've emailed you a code" is displayed. Below it, the text "To complete your account setup, enter the code we've sent to:" and the email address "testing.uncovered+batman@gmail.com" are shown. There are six input fields for the verification code: the first five contain "5", "1", "3", "1", and "3" respectively, while the last one is empty. A large blue "Verify" button is at the bottom. Below the "Verify" button is a link "Didn't receive an email? Resend email".

Jira cloud signup

Take note of your Jira Cloud instance URL. You'll be asked some questions that you may skip.

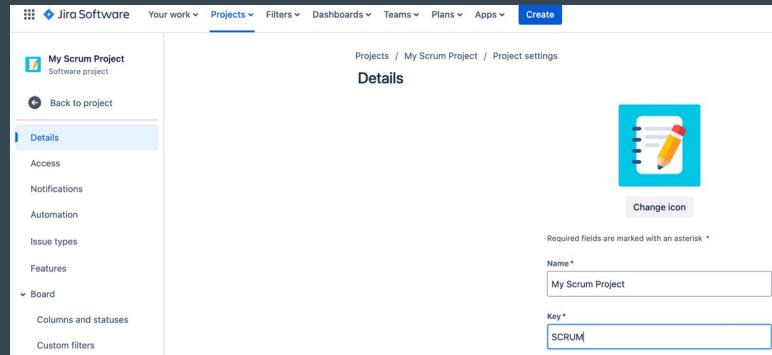


This screenshot shows the second step of the Jira Cloud setup process. The heading "What kind of work do you do?" is at the top. Below it, a sub-instruction "This helps us suggest templates that help your team do their best work." is shown. A grid of categories is displayed, with "Software development" highlighted by a red box and a red arrow pointing to it. Other categories include Marketing, Product management, Project management, Design, Operations, IT support, Customer service, Finance, and Data science. To the right, a section titled "What types of work do you need?" lists building blocks of projects: Task (A small piece of work), Story (A requirement expressed from the user's perspective), Feature (A broad piece of functionality), Request (An ask for assistance), and Bug (A problem that needs fixing). A note at the bottom says "Don't worry, you can change these later." At the very bottom are "Step 2 of 3" and a "Next" button.

This screenshot shows the third step of the Jira Cloud setup process. The heading "How does your team plan to use Jira?" is at the top. Below it, a sub-instruction "These form the building blocks of your project." is shown. A grid of checkboxes for team planning is displayed, with several checked: "Track bugs", "Manage tasks", "Work in scrum", "Prioritize work", "Run sprints", and "Map work dependencies". At the bottom are "Step 3 of 3" and a "Next" button.

Project creation during signup

During the Jira cloud instance signup, you'll be asked to create your first project; this will create a *team-managed* project named "My Scrum Project", having the key "SCRUM"; therefore, all issues on that project will have keys like SCRUM-<int> (e.g., "SCRUM-1"). You can change the project name and key on the project settings panel.



Creation of projects

You can also create your project later on, from the top Projects menu.

Have in mind though that there are 2 main project types in Jira Cloud:

- **Team-managed** (simpler but not so powerful; good enough for us)
- Company-managed (more complex and more powerful)

The screenshot shows the Jira Cloud interface for creating new projects. On the left, a sidebar lists categories like Project templates, Software development, Service management, etc. The Software development section is selected. In the center, a modal window titled "Choose a project type" offers two options: "Team-managed" and "Company-managed". The "Team-managed" section is highlighted with a yellow background. It includes a warning message: "⚠ You'll need to create a new project if you decide to switch project types later." Below this, it describes "Simplified configuration" and "Standardized configuration". The "Company-managed" section is described as "Set up and maintained by experts". On the right, the "Add project details" form is partially visible, showing fields for Name (set to "DEMO"), Key (set to "DEMO"), and Access (set to "Open"). Below the modal, there are cards for "Scrum" and "Kanban". A legend at the bottom indicates that the "Team-managed" card is blue and the "Company-managed" card is orange.

Project templates

Software development

Service management

Work management

Product management

Marketing

Human resources

Finance

Design

Personal

Project templates

Software development

Plan, track and release great software. Get up and running quickly with templates that suit the way your team works. Plus, integrations with tools like GitHub, Bitbucket, and Jira.

Kanban
Jira

Visualize and advance your project forward using work items on a Kanban board.

Scrum
Jira

Sprint toward your project goals with a board, backlog, and timeline.

Choose a project type

⚠ You'll need to create a new project if you decide to switch project types later.

Team-managed

Set up and maintained by your team.

For teams who want to control their own working processes and practices in a self-contained space. Mix and match agile features to support your team as you grow in size and complexity.

Simplified configuration

Get up and running quickly, with simplified configuration.

Anyone on your team can set up and maintain

Settings do not impact other projects

Company-managed

Set up and maintained by experts.

For teams who want to work on projects in a standard way, following organizational best practices and shared configuration.

Expert configuration

Benefit from configuration, customization, and shared configuration.

Set up and maintained by experts.

Standardized configuration

Add project details

Explore what's possible when you collaborate with your team. Edit project details anytime in project settings.

Template

Change template

Scrum
Jira

Sprint toward your project goals with a board, backlog, and timeline.

Type

Change type

Team-managed

Control your own working processes and practices in a self-contained space.

Sergio Freire

Xray Installation

Installation of Xray

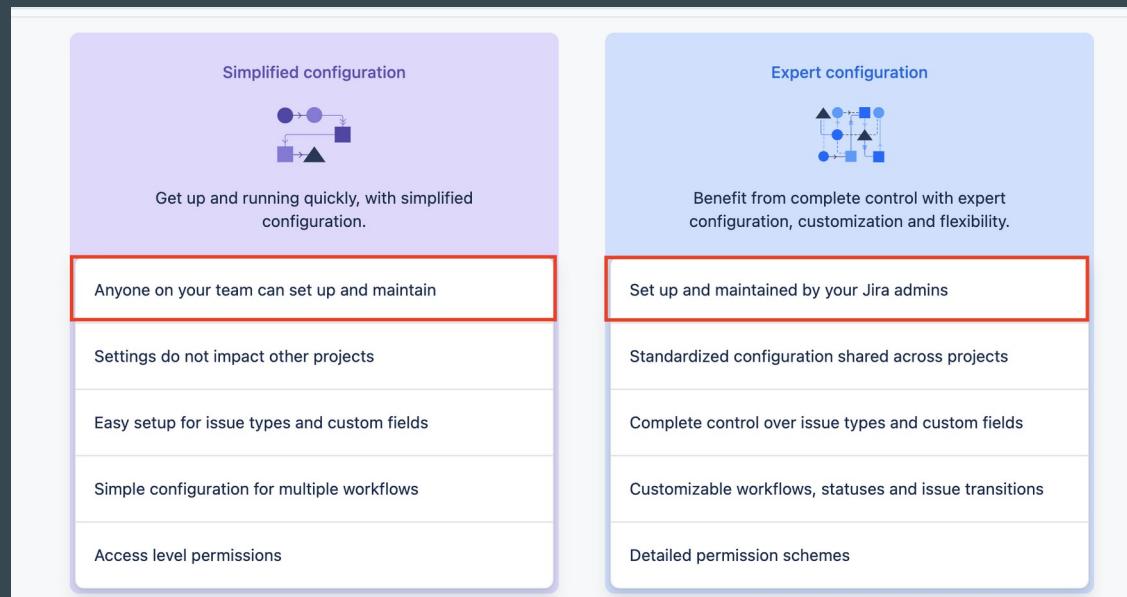
Installation is quite straightforward; go to Apps > Explore more apps > Xray and then “Try it free”. You need to use the account that created the Jira

The image consists of three screenshots from the Atlassian Marketplace:

- Search results for Jira:** A screenshot of the marketplace search interface. A search bar at the top contains "xray". Below it, a filter bar includes "Pricing", "Trust signals", "Categories", "Use cases", and "More filters". The main area shows search results for "Xray Test Management for Jira" and "Xray Enterprise". The first result, "Xray Test Management for Jira" by Xblend, is highlighted with a red border. It shows a green icon, the app name, developer name, a brief description, a 3.4/4 star rating, 515 reviews, and 32k installs.
- Xray Test Management for Jira product page:** A screenshot of the product page for "Xray Test Management for Jira". It features a large green header with the app logo and name. Below the header, there's a brief description: "Native Test Management. Built for every member of your team to plan, test, track and release great software". A "TRY FREE" button is prominently displayed. The page also shows a 3.5/4 star rating, 401 reviews, and 28,227 installs. It includes sections for "Overview" and "Support".
- Add to Jira dialog box:** A screenshot of the "Add to Jira" dialog box. It displays the same information as the product page: the app logo, name, developer, rating, reviews, and installs. It also lists the actions the app will perform: "Act on a user's behalf, even when the user is offline", "Administer the host application", "Administer Jira projects", and "Delete data from the host application". A "Start free trial" button is at the bottom right.

Enabling Xray on Jira projects

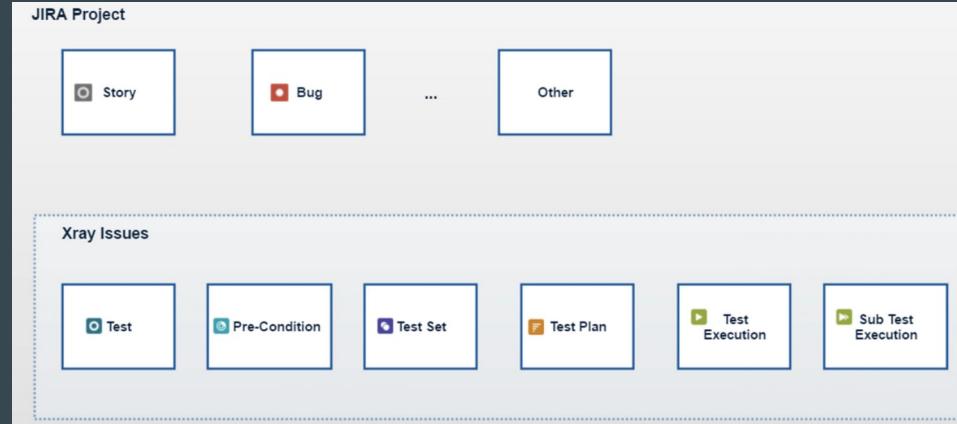
- Note: there are 2 main project types in Jira Cloud:
 - Team-managed
 - Company-managed
- We can enable Xray in any of these project types but the configuration steps are slightly different



Organization strategy: “All-in-One”

A single project to manage your “Requirements” (stories, epics), Defects (bugs), and Test related issues and also have all your Test Executions.

The idea is to use one project to manage everything related to it, including testing.



Enabling Xray on team managed projects

Setting a team-managed project in a nutshell

Create or use an existing team-managed project. You can use the project you created during the Jira signup.

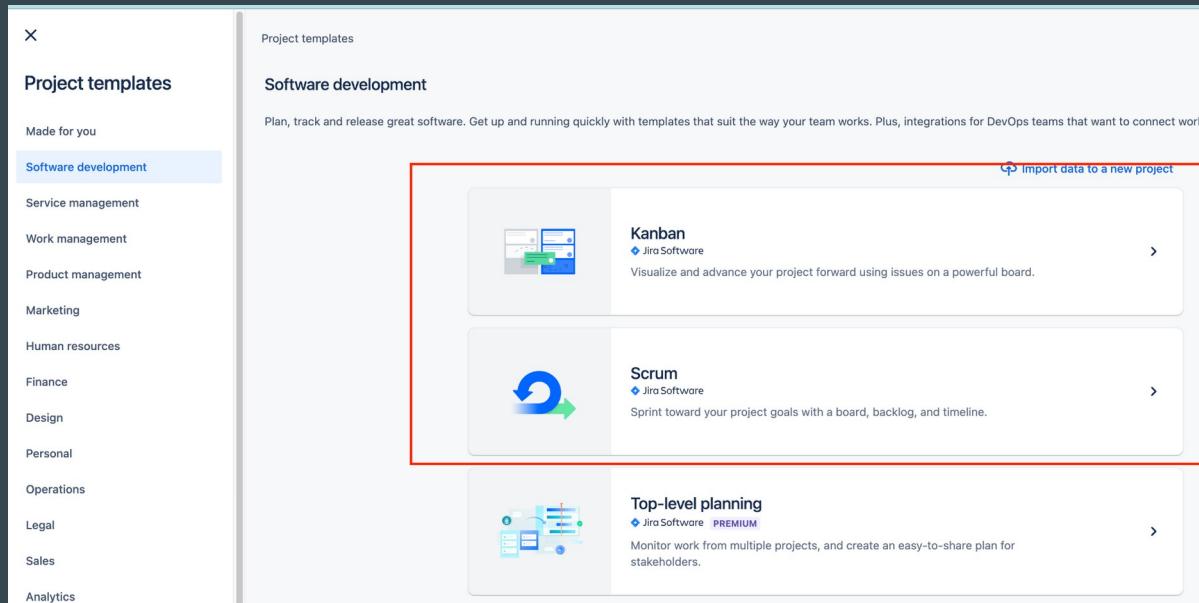
Then:

1. Add (i.e., create) issue types to the project and map them to the Xray concepts
 - a. Precondition, Test, Test Set, Test Execution, Test Plan
2. Define which items you consider to be “requirements” like
3. Define which items you consider to be “defects” like

That's it.

Create project (unless you have created one already)

- Create a Jira project using a Kanban or Scrum template



Create project (unless you have created one already)

- We'll use the Scrum template

Add project details

Explore what's possible when you collaborate with your team. Edit project details anytime in project settings.

Name *

Access *

Open

Key ⓘ *

Template

Scrum

Jira Software

Sprint toward your project goals with a board, backlog, and timeline.

Type

Team-managed

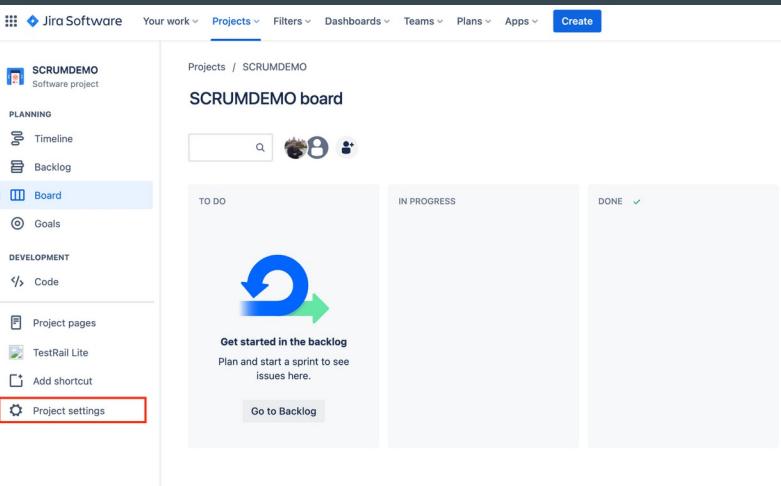
Control your own working processes and practices in a self-contained space.

Cancel

Next

Xray project setup

- Go to **Project settings > Apps > Xray Settings**
- There are a few configurations to perform



Jira Software Projects / SCRUMDEMO SCRUMDEMO board

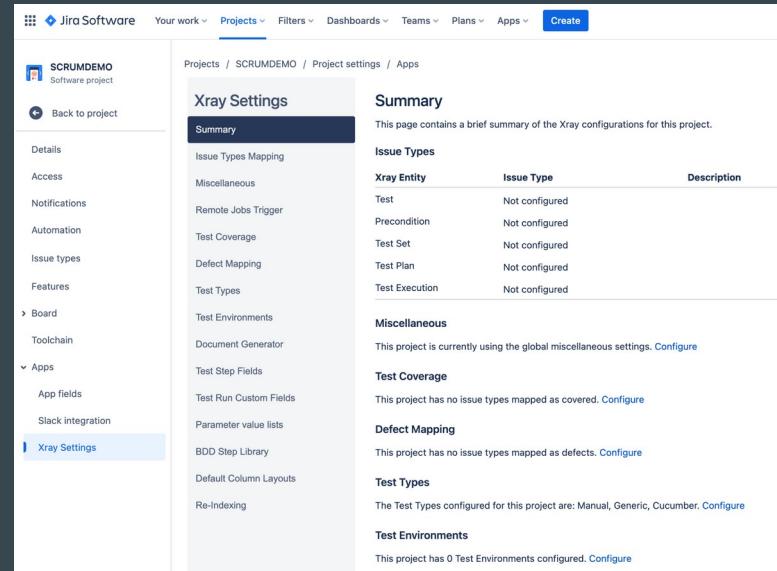
PLANNING Timeline Backlog

Board Goals

DEVELOPMENT Code

Project pages TestRail Lite Add shortcut

Project settings



Jira Software Projects / SCRUMDEMO / Project settings / Apps

Xray Settings

Summary

This page contains a brief summary of the Xray configurations for this project.

Issue Types

Xray Entity	Issue Type	Description
Test	Not configured	
Precondition	Not configured	
Test Set	Not configured	
Test Plan	Not configured	
Test Execution	Not configured	

Miscellaneous

This project is currently using the global miscellaneous settings. [Configure](#)

Test Coverage

This project has no issue types mapped as covered. [Configure](#)

Defect Mapping

This project has no issue types mapped as defects. [Configure](#)

Test Types

The Test Types configured for this project are: Manual, Generic, Cucumber. [Configure](#)

Test Environments

This project has 0 Test Environments configured. [Configure](#)

Xray project setup

- Go to **Issue Types Mapping** to configure the issue types
- Then go back to **Project settings > Issue Types** using the “project configuration” link to create the issue types used by Xray

Projects / SCRUMDEMO / Project settings / Apps

Xray Settings

- Summary
- Issue Types Mapping**
- Miscellaneous
- Remote Jobs Trigger
- Test Coverage
- Defect Mapping
- Test Types
- Test Environments
- Document Generator
- Test Step Fields
- Test Run Custom Fields
- Parameter value lists
- BDD Step Library
- Default Column Layouts
- Re-Indexing

Team-Managed Project Issue Types Mapping

The issue type configuration for this team-managed project.

Manual configuration required

In order to configure Xray issue types for this team-managed project, you first need to create the issue types manually on the [project configuration](#).

After the creation of the issue types for the Xray entities you need, you must associate them below.

Test

Select an issue type

Precondition

Select an issue type

Test Set

Select an issue type

Test Plan

Select an issue type

Test Execution

Select an issue type

Jira Your work Projects

DEMO Software project

Project settings

Work types

- Epic
- Bug**
- Story
- Task
- Subtask

+ Add work type

Xray project setup: creation of issue types

- Create the following issue types (also called “work types”) with these names:
 - Precondition, Test, Test Set, Test Plan, and Test Execution
- Optionally: configure the related icons ([download them here](#))

The image shows two screenshots of the Jira interface. On the left, the 'Work types' section of the 'Project settings' page is displayed. It lists several work types: Epic, Bug (which is selected), Story, Task, and Subtask. A red box highlights the '+ Add work type' button at the bottom. On the right, a modal window titled 'Create issue type' is open. It has fields for 'Name*' (containing 'Tes') and 'Description' (containing 'Let people know when to use this issue type'). Below these is an 'Icon' section with a 'Change icon' button. At the bottom right of the modal are 'Create' and 'Cancel' buttons.

Xray project setup: associate the issue types

This step is for Xray to be aware of the issue types we want to use for its own entities; we could have created “Test Case”, “Test List” issue types (for example) and associate them with the Test and Test Set concept of Xray.

However, in our case, we'll keep it simple, using issue types with the same names of the expected entities:

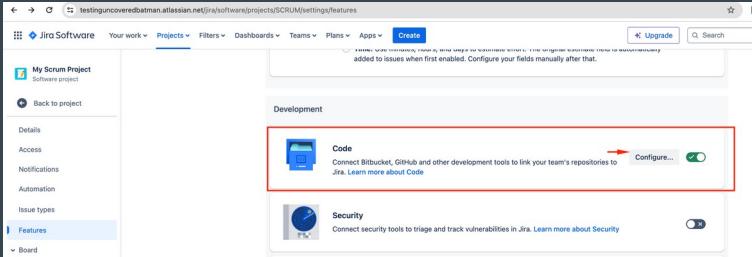
- Test > Test
- Precondition > Precondition
- Test Set > Test Set
- Test Plan > Test Plan
- Test Execution > Test Execution

The screenshot shows the Jira Software interface with the following details:

- Header:** Jira Software, Your work, Projects, Filters, Dashboards, Teams, Plans, Apps, Create.
- Left Sidebar (Project SCRUMDEMO):** Details, Access, Notifications, Automation, Issue types, Features, Board, Toolchain, Apps (with sub-options: App fields, Slack integration), Xray Settings.
- Central Content Area:** Projects / SCRUMDEMO / Project settings / Apps. Sub-section: Xray Settings (with sub-options: Summary, Issue Types Mapping). The "Issue Types Mapping" option is highlighted with a blue background.
- Right Panel:** Team-Managed Project Issue Types Mapping. Description: The issue type configuration for this team-managed project. It lists several entities with their corresponding issue types:
 - Test:** Test
 - Precondition:** Precondition
 - Test Set:** Test Set
 - Test Plan:** Test Plan
 - Test Execution:** Test Execution
- Bottom Right:** Save button.

Nice-2-Have

In project settings > **Features**, enable Code and add the code repository tool (e.g., GitHub).



Jira Software - Your work - Projects - Filters - Dashboards - Teams - Plans - Apps - Create - Upgrade - Search

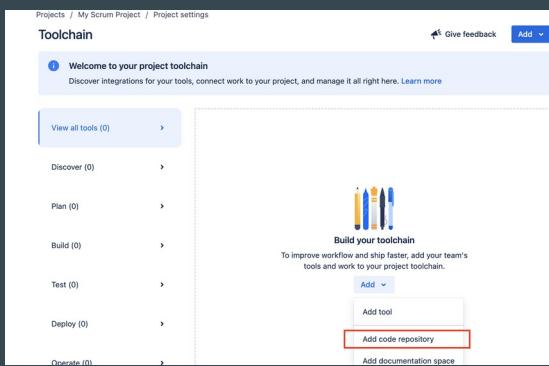
My Scrum Project Software project Back to project

Details Access Notifications Automation Issue types Features Board

Development

Code Connect Bitbucket, GitHub and other development tools to link your team's repositories to Jira. Learn more about Code **Configure...** **Switch**

Security Connect security tools to triage and track vulnerabilities in Jira. Learn more about Security **Switch**



Projects / My Scrum Project / Project settings

Toolchain

Welcome to your project toolchain Discover integrations for your tools, connect work to your project, and manage it all right here. Learn more

View all tools (0) >

Discover (0) >

Plan (0) >

Build (0) >

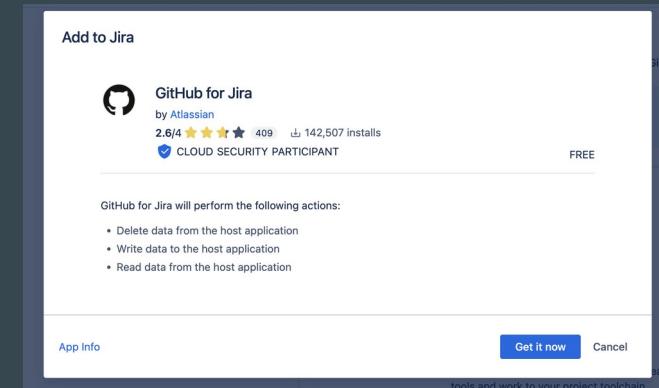
Test (0) >

Deploy (0) >

Operate (0) >

Build your toolchain To improve workflow and ship faster, add your team's tools and work to your project toolchain.

Add tool Add code repository Add documentation space



Add to Jira

GitHub for Jira by Atlassian 2.6/4 ★★★★ 409 142,507 installs CLOUD SECURITY PARTICIPANT FREE

GitHub for Jira will perform the following actions:

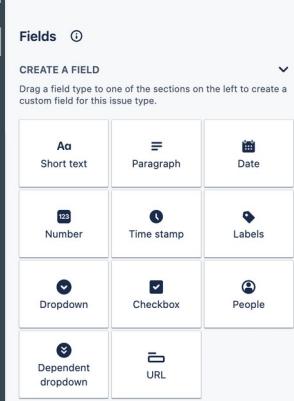
- Delete data from the host application
- Write data to the host application
- Read data from the host application

App Info Get it now Cancel

OPTIONAL: Associate some custom fields to Xray issue types

On **Project settings > Issue types**, for the following issue types add the fields

- Test Execution: Revision
 - As short text
- Test Plan: Begin Date and End Date
 - As date



The screenshot shows the 'Issue types' configuration for the 'Test Execution' issue type. It includes sections for 'Description fields' (containing 'Summary' and 'Description') and 'Context fields' (containing 'Status', 'Assignee', 'Labels', 'Parent', 'Story point estimate', 'Sprint', and 'Fix versions'). At the bottom, a new field 'Aa Revision' is being added, indicated by a red border around its input field. A 'HIDE WHEN EMPTY' link is at the bottom right.

Xray: define “requirements” / issues coverable by tests

- **Coverable Issue Types:** *Which issue types do you aim to cover with tests? (e.g., Story, Epic)*

The screenshot shows the Jira Xray Settings page for the SCRUMDEMO project. The left sidebar has a 'Xray Settings' section selected. The main content area is titled 'Test Coverage' and contains a description: "A testable/coverable entity is an issue that you can cover with tests in order to validate its acceptance criteria. It can be a Story, Epic, Bug or any custom issue type such as a Requirement." Below this, there are two columns: 'Available Issue Types' and 'Coverable Issue Types'. A red box highlights the 'Coverable Issue Types' column, which lists 'Story' and 'Epic'. At the bottom, there are dropdowns for 'Issue Links relation' and 'Issue Link Column Direction' (Outward or Inward), and a note about Xray using the issue link type configuration for hierarchy between issues.

Xray: define defects

- **Defect Issue Types:** Which issue types do you want to be handled as defects? (e.g., Bug)

The screenshot shows the 'Xray Settings' page in Jira Software, specifically the 'Defect Mapping' section. The left sidebar lists various settings like Details, Access, Notifications, Automation, Issue types, Features, Board, Toolchain, Apps, App fields, Slack integration, and Xray Settings. The 'Xray Settings' section is currently selected. The main content area has a title 'Defect Mapping' with a sub-section 'Available Issue Types'. This list includes Story, Task, Epic, Subtask, Test, Precondition, Test Set, Test Plan, and Test Execution. To the right, under 'Defect Issue Types', the 'Bug' option is selected, indicated by a checked checkbox. A 'Save' button is at the bottom.

Xray: final checkup

Go to Project settings > Apps > Xray Settings > Summary.

The screenshot shows the Jira Software interface with the following details:

- Header:** Jira Software, Your work, Projects, Filters, Dashboards, Teams, Plans, Apps, Create, 13 days left, Search.
- Left sidebar:** SCRUMDEMO (Software project), Back to project, Details, Access, Notifications, Automation, Issue types, Features, Board, Toolchain, Apps (App fields, Slack integration, Xray Settings), You're in a team-managed project, Learn more.
- Current page:** Projects / SCRUMDEMO / Project settings / Apps / Xray Settings / Summary.
- Content Area:**
 - Summary:** This page contains a brief summary of the Xray configurations for this project.
 - Issue Types:**
 - Xray Issue Types in Project:** There are 5 of 5 Xray issue types configured for this project. Go to the [Issue Types Mapping](#) to configure the issue types for this project.
 - Xray Entity** | **Issue Type** | **Description**

Test	Test	
Precondition	Precondition	
Test Set	Test Set	
Test Plan	Test Plan	
Test Execution	Test Execution	
 - Miscellaneous:** This project is currently using the global miscellaneous settings. [Configure](#)
 - Test Coverage:** This project has the following issue types mapped as covered: Story, Epic. [Configure](#)
 - Defect Mapping:** This project has the following issue types mapped as defects: Bug. [Configure](#)
 - Test Types:** The Test Types configured for this project are: Manual, Generic, Cucumber. [Configure](#)
 - Test Environments:** This project has 0 Test Environments configured. [Configure](#)

If all goes well...

- You should see a “Testing Board” shortcut on the side menu
- Whenever creating issues, you should see the issue types that were created earlier

This screenshot shows the Jira Software interface for the project 'SCRUMDEMO'. The left sidebar contains navigation links for Planning, Backlog, Board, Goals, Issues, Development, Code, Releases, Project pages, and Testing Board. The 'Testing Board' link is highlighted with a red box. The main content area displays the 'Test Repository' for the project, showing one entry: 'SCRMDEMO-3 manual test for valid login scenario'. A 'Create Test' button is visible at the bottom.

This screenshot shows the 'Create issue' dialog in Jira. The 'Project' dropdown is set to 'SCRUMDEMO (SCRUMDEMO)'. The 'Issue type' dropdown is set to 'Test', which is also highlighted with a red box. Other available issue types listed in the dropdown include Story, Task, Bug, Epic, Precondition, Test Set, Test Plan, and Test Execution. The 'Create' button is located at the top right of the dialog, also highlighted with a red box.

Try it out, just in case :)

1. Create project with your name in uppercase (e.g., "SCRUMDEMO"); notice the project key (e.g., "SCRUMDEMO")
2. Create Epic "client area" and a Story "login"; associate Story to the parent Epic
3. Create a Test from the Story screen

The image consists of three side-by-side screenshots of the Jira software interface, illustrating the process of creating a test from a story screen.

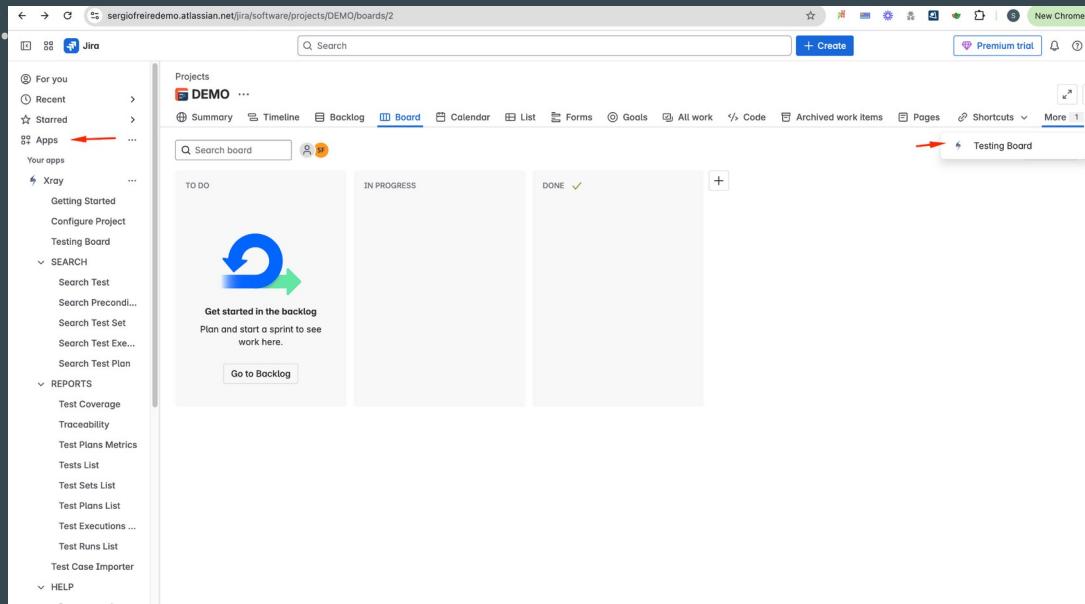
Screenshot 1: Story Screen
This screenshot shows the "client area" story screen. The story is titled "As a user, I can login the web application". It has one child issue, "SCRUMDEMO-1 As a user, I can login the web application", which is linked to the story. The "Test Coverage" section shows a single test named "SCRUMDEMO-3 manual test for valid login scenario" with a status of "TO DO".

Screenshot 2: Story Screen with Test Attached
This screenshot shows the same story screen after a test has been attached. The test "SCRUMDEMO-3 manual test for valid login scenario" is now listed under "Linked issues" with a status of "IS TESTED". The "Test Coverage" section still shows the same test with a status of "TO DO".

Screenshot 3: Test Details Screen
This screenshot shows the "Test details" screen for the test "SCRUMDEMO-3 manual test for valid login scenario". The screen indicates "There are no steps defined". It includes sections for "Test details", "Preconditions", "Test Sets", "Test Plans", and "Test Runs". A note at the bottom states: "A test is a sequence of steps covered with conditions, test inputs and automated results. Please define such steps to define the test".

NOTE: new Jira UI coming during 2025...

Jira is changing the UI so the “Testing Board” page from Xray will appear on a different place. On the left side menu, Xray related pages are available from within Apps > Xray.



Applying promo code

Your Xray trial ends in 1 month, so you must use the “promo code” given by your teacher. Each group can only use the promo code once! Go to Jira > Administration > Billing > Enter promo code.

The screenshot shows the Jira administration interface. On the left, there's a sidebar with links like Home, Jira, Goals, Teams, and Administration (which is highlighted with a red box). The main area has tabs for Admin, sergiofreiredemo, Overview, Directory, Products, Security, Billing (which is also highlighted with a red box), and Settings. Below these tabs, there's a message about the new overview and a "Quick actions" section with an "Invite users" button. Under "For review", there's a warning about the trial ending soon. At the bottom, there's a "Monitor" section. To the right, a larger window shows the "Subscription details" for "sergiofreiredemo". It includes sections for "Payment details needed" (warning about needing a payment method before May 25, 2025), "Subscription details" (listing "Xray Test Management for Jira" and "sergiofreiredemo.atlassian.net | E-43F-LVQ-HPQ-K2X"), and "Standard" billing information (Billing cycle: Monthly, Next bill date: May 25, 2025, Next bill estimate: USD 10.00). A red box highlights the "Enter promo code" input field in the "Payment details needed" section.

Xray Entities & Concepts

...

Xray entities & concepts (a few)

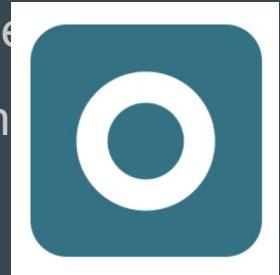
Xray provides several concepts; we'll focus just on a few ones.

- Test
- Test Run
- Test Execution
- Test Plan
- “Requirements”
- “Defects”

All of the previous entities will be issues in Jira/Xray, except the Test Run which is

Test

An abstraction of a test idea/scenario, automated test, and thus re
It is essentially a way to verify/validate the associated requiremen



- Can be a scripted (e.g. test case, automated test) or exploratory test
- Can be specified using Gherkin (Scenario)
- Can be executed manually in Jira/Xray or through automation
- May be linked to/cover 1 (or more) requirements, using the “tests” issue link type
- Has one type: “Manual”, “Cucumber”, “Generic”

Test: Manual

A traditional test case composed by a list of steps, thus scripted.

- Each step is mainly defined by:
 - Action/step
 - Expected result

The screenshot shows the Xray software interface for managing test cases. The main area displays a 'Test details' card for a test case titled 'CALC-1389'. The 'Test Type' is set to 'Manual'. Below this, there is a table listing three test steps:

Step	Action	Data	Expected Result
1	Action press 1	None	1
2	Action press key	+	+
3	Action press 2	None	2

On the right side of the screen, there is a detailed sidebar panel for the test case, showing information such as Assignee (Unassigned), Reporter (Sérgio Freire), Development (Create branch, Create commit), Releases (None), Labels (None), Revision (None), ExternalRef (None), Priority (Medium), Xporter (Open Xporter), Test Status (Open Test Status), and Automation (Rule executions). The sidebar also indicates the test was created on March 9, 2022, at 5:53 PM and last updated on March 9, 2022, at 5:53 PM.

If we don't have test automation yet, or the test scenario is hard/costly to automation, we can create a "manual" test and link it to the related Story, for example. We can then manually record its result in Xray.
Sergio Freire

Test: Cucumber

A Cucumber Scenario/Scenario Outline that provides one or more examples of an acceptance criteria.

- Write tests in a business-readable domain-specific language (Gherkin)
- Specify Cucumber Scenarios and Scenario Outlines with Gherkin syntax highlighting
- Ability to export to .feature files and execute during Continuous Integration

Test Type
Cucumber

Scenario

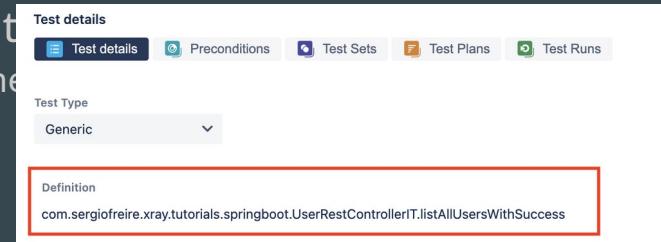
```
1 Given I have entered "<input_1>" into the calculator
2 And I have entered "<input_2>" into the calculator
3 When I press "<button>"
4 Then the result should be "<output>" on the screen
5
6 Examples:
7 | input_1 | input_2 | button | output |
8 | 20     | 30      | add    | 50   |
9 | 2      | 5       | add    | 7    |
10 | 0     | 40      | add    | 40   |
11 | 4     | 50      | add    | 54   |
12 | 5     | 50      | add    | 55   |
```

We can use Xray as the master for the Cucumber/Gherkin scenarios or we can use Git instead. The flows for fully integrating the results from automation are slightly different as shown in [this tutorial](#).

Test: Generic

A way to abstract and have visibility of traditional automated tests or exploratory tests.

- Allows you to track the results of automated tests that are non-Cucumber (e.g. JUnit)
 - Automate tests in any framework and report results back to Xray
 - These tests are usually auto-provisioned whenever importing the results the first time
 - Definition field contains a unique identifier:
`<package_name>.<class_name>.<method_name>`

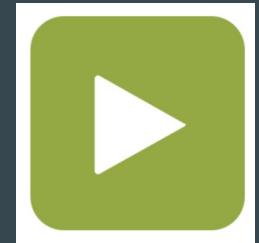


Note: Generic tests could also be used to have visibility of Cucumber results if upload JUnit XML reports. In that case we lose visibility of the individual Gherkin sentences; we would just have visibility of the whole test result.

Test Execution

A “task” for executing a group of tests on a given version of the system, on a given environment. This task will also contain the results when they’re reported. Sometimes the Test Execution is created alongside with its results (i.e., the Test Runs).

- Contains a list of Tests and their results (i.e. Test Runs)
- May be planned (especially for manual tests) or ad hoc
- May be created manually (i.e., “I want to run these tests...”);
 - Later on someone will “execute” it manually, by going to each test and record the actual results and whether the test passed or failed
- May be created during Continuous Integration
 - in this case it will already contain the results (i.e., a Test Run for each Test that was executed)



Test Run

An instance of a Test in the context of a Test Execution. A run of a Test in some scope.

- Contains the results obtained for a Test, including evidence and linked/reported defects
- For compliance, also contains a copy of the Test specification at the moment of execution
- It's an internal entity; not a Jira issue

Test Plan

A way to define the scope for testing and track its progress; what testing we'll be performing and its latest status.



- Tracks a group of tests and their results independently of the number of executions
- Can be used in a planned way, with its scope for testing (i.e. the Tests) defined beforehand
- Can be used in an Agile way, acting as a testing guidance result aggregator, by allowing you to create/add Tests along with their results at any time
- Test Plans may be assigned to versions, sprints and users, as they're a Jira issue
- A version or a sprint may have multiple Test Plans

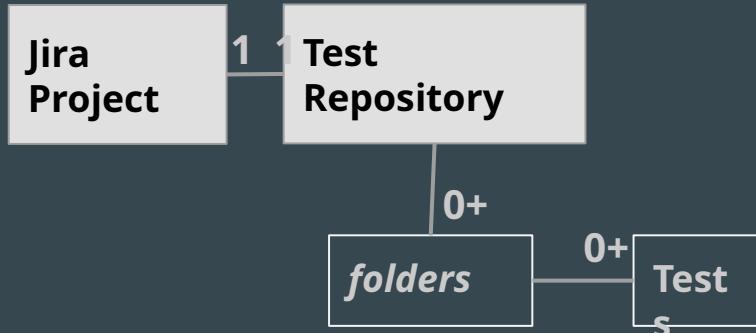
Relation between entities

- “Requirements” are coverable by one or more Tests
- A Test Plan has a list of Tests it tracks and usually several Test Executions related to those Tests
- A Test Execution contains Test Runs, one for each Test that is on the Test Execution

Additional entities... (non issue type based)

Test Repository

- The place where we can see all Tests in a project, organized in folders
- No information about results
- Mostly useful for manual testing



The screenshot shows the 'Test Repository' page for the 'Calculator' project. The left sidebar has tabs for 'FOLDERS' and 'TEST SETS'. The main area displays a tree view of test sets and their counts:

Folder	Count
Test Repository	1909 (1933)
Performance testing	0 (0)
Postman F1	14 (14)
Postman F2	1 (1)
core	1 (1)
UI	0 (8)
login	1 (1)
basic arithmetic operations	7 (7)

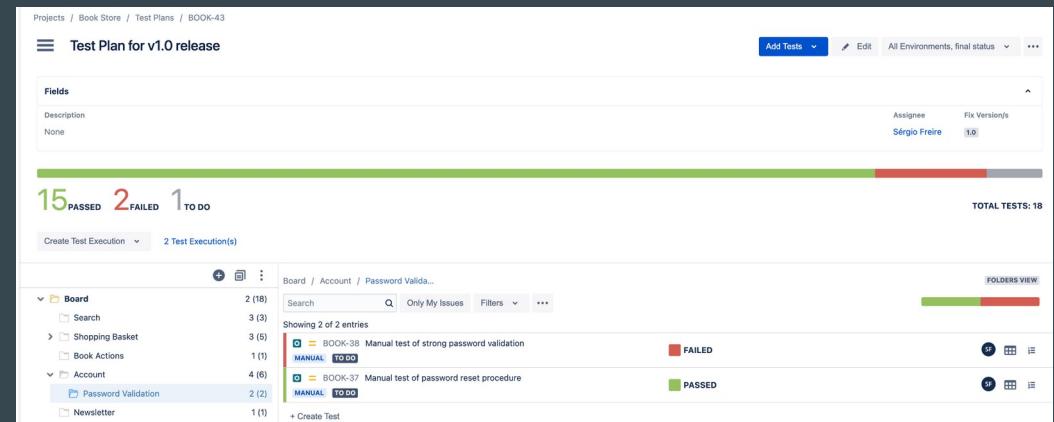
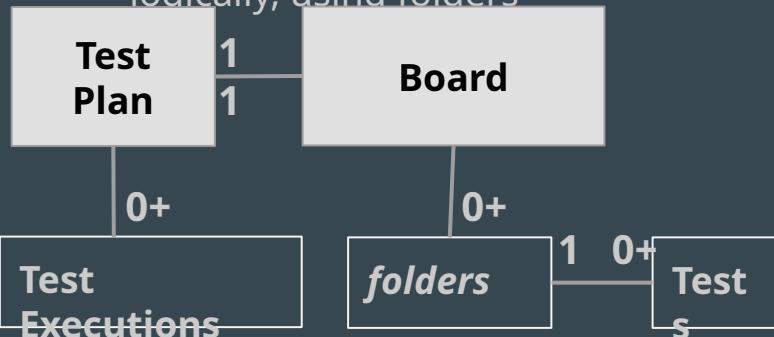
Below the tree view is a table of test cases:

Test Case	Status
CALC-2706 dummy gherkin precondition	CUCUMBER TO DO
CALC-2666 xx	CUCUMBER TO DO
CALC-1450 organizations exist	CUCUMBER TO DO
CALC-864 Background for: CALC-861	CUCUMBER TO DO
CALC-393 Calculator exists	CUCUMBER TO DO

Additional entities... (non issue type based)

Test Plan' Board

- An implicit structure in each Test Plan to organize the Tests in folders, considering priorities and what makes sense from a execution perspective
- Can be useful to group the results logically, using folders



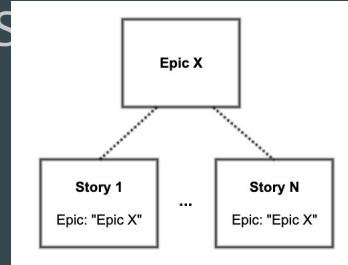
Defects (e.g., “Bug”)

A defect is something that negatively impacts quality (the value perceived to some stakeholder).

- Xray uses the “defect” word in a broader sense, i.e. some deficiency/imperfection related to the product. Usually, these are also known as bugs
- In practice, a “defect” is something we report manually during testing, or after testing whenever analyzing the test results, as a Jira issue (e.g., “Bug”)
- You can define one or more issue types to be treated as “defects”; they can include Bug or any other custom issue types you may want for that purpose

“Requirements” (e.g., Story, Epic)

- Xray uses “requirement” word in broad sense: something that the SUT must meet
- In practice, a “requirement” is any issue in Jira that can be covered (i.e., verified or validated) with Tests
- You can define the issue types to be treated as “requirements”; they can include Story, Epic, or any other custom issue type
- Xray is aware of the hierarchical relation between Epics and Stories; if you create a Test for a Story, you can see it at the related Epic level (i.e., covering the Epic)



Coverage in Xray

Xray provides an heuristic, called (test) coverage; sometimes people may refer to it as requirement coverage, depending on the perspective.

This is different from code coverage, which is focused on covering code.

Coverage in Xray is at higher level; it's a way to understand if a deliverable/requirement (Story, Epic) has some tests associated and if so, if all those tests have been run and were successful.

Coverage: UNCOVERED

In Xray a “requirement” (Story, Epic) is UNCOVERED if it has no Tests linked/covering it.

The screenshot shows a requirement card in the Xray application. At the top, there are navigation links: Projects, Calculator, Add parent, and CALC-2855. Below these are several buttons: Attach, Create subtask, Link issue, Test Coverage (which is highlighted with a green icon), and three vertical dots. The main title of the requirement is "As a user, I can login the web application". Underneath the title are buttons for Attach, Create subtask, Link issue, and three vertical dots. A "Description" section contains the placeholder "Add a description...". The "Test Coverage" section features a central icon of a clipboard with a magnifying glass over a checklist, surrounded by green arrows pointing in various directions. Below this icon is a red-bordered box containing the word "UNCOVERED" in white capital letters. Inside the box, the text "No Tests are associated with this issue." is displayed. At the bottom of the requirement card is a blue button labeled "Add Tests" with a dropdown arrow.

Coverage: NOTRUN

In Xray, a requirement is “NOTRUN” if:

- *any* of the linked tests need yet to be run

Remember: it's an heuristic! Tests don't cover everything and not every test (problems found during their execution) has the same importance.

The screenshot shows a requirement in the Xray application. The requirement title is "As a user, I can login the web application". Below the title, there are buttons for "Attach", "Create subtask", "Link issue", "Test Coverage", and more. The "Test Coverage" button is highlighted with a yellow background. The "Description" section has a placeholder "Add a description...". The "Linked issues" section shows one linked issue: "CALC-2856 test for successful login" with a status of "TO DO". In the "Test Coverage" section, there is a "NOTRUN" status indicator. The "Analysis & Scope" section shows "Scope: Latest Final Status". The "Test Status" table at the bottom lists one test: "CALC-2856 test for successful login" with a status of "TO DO".

Coverage: OK

In Xray, and also in some tools, a requirement is “OK” if ...

- All linked tests passed, according with latest results

Remember: it's an heuristic! Tests passing don't mean we can ensure the requirement doesn't have problems.

The screenshot shows a Jira issue page for requirement CALC-2855. At the top, there are buttons for Attach, Create subtask, Link issue, Test Coverage (which is highlighted in green), and more. Below that is a Description field with placeholder text "Add a description...". Under "Linked issues", it says "is tested by" and lists "CALC-2856 test for successful login" with a status of "TO DO". In the "Test Coverage" section, there are "Add Tests" and "Execute" buttons. The "Analysis & Scope" section shows "Scope: Latest Final Status". A large green button labeled "OK" is prominently displayed. Below this, a table provides a detailed view of the coverage:

Status	Key	Summary	Test Status
TO DO	CALC-2856	test for successful login	PASSED

At the bottom, there are navigation links for Prev, 1, and Next.

Coverage: NOK

In Xray, a requirement is “NOK” if ...

- *any* of the linked failed, according with latest results

Remember: it's an heuristic! Tests don't cover everything and not every test (problems found during their execution) has the same relevance.

The screenshot shows a requirement titled "As a user, I can login the web application". Under "Linked issues", two issues are listed: "CALC-2856 test for successful login" and "CALC-2858 test for invalid login", both marked as "TO DO". In the "Test Coverage" section, a summary table shows two tests: "test for successful login" (Status: TO DO, Key: CALC-2856) and "test for invalid login" (Status: TO DO, Key: CALC-2858). A red box highlights the "NOK" status indicator in the top right corner of the coverage summary. A legend at the bottom right indicates that green means "PASSED" and red means "FAILED".

Status	Key	Summary
TO DO	CALC-2856	test for successful login
TO DO	CALC-2858	test for invalid login

Test Status :
PASSED
FAILED

OTHER, OPTIONAL INFO

• • •

Just in case you need it...

Enabling Xray on company managed projects

Setting a company-managed project in a nutshell

Create or use an existing company-managed project and then:

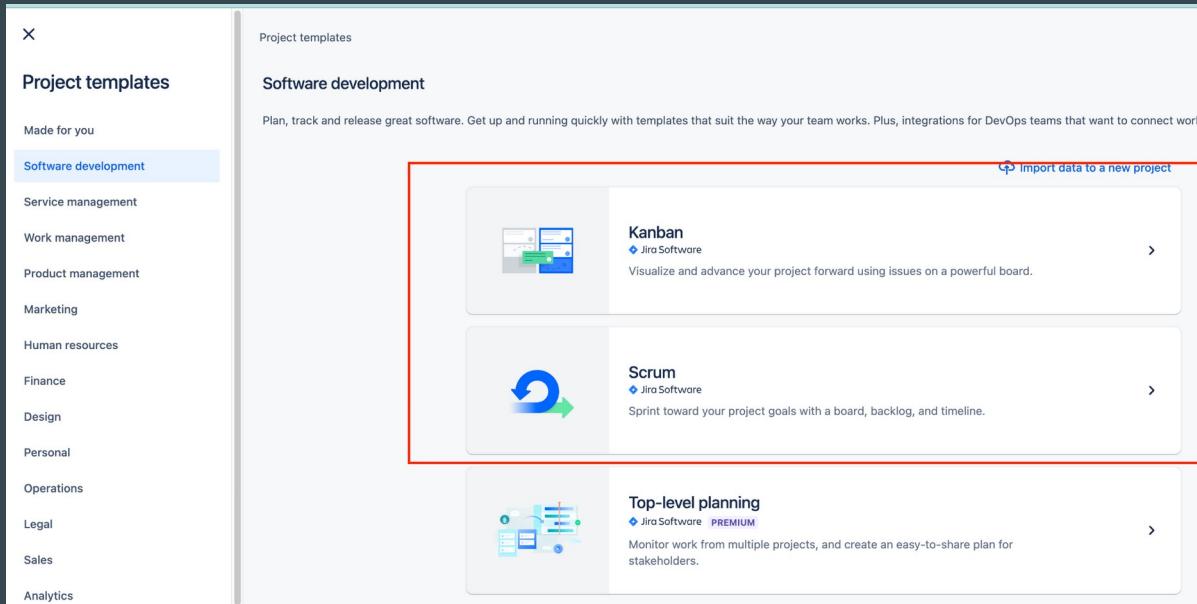
1. Add issue types to the project and map them to the Xray concepts
2. Define which items you consider to be “requirements” like
3. Define which items you consider to be “defects” like

That's it.

There are additional settings that we can fine tune ahead.

Create project

- Create a Jira project using a Kanban or Scrum template



Select type of project

Team-managed	Company-managed
<p>Set up and maintained by your team.</p> <p>For teams who want to control their own working processes and practices in a self-contained space. Mix and match agile features to support your team as you grow in size and complexity.</p> <p>Simplified configuration</p>  <p>Get up and running quickly, with simplified configuration.</p> <ul style="list-style-type: none">Anyone on your team can set up and maintainSettings do not impact other projectsEasy setup for issue types and custom fieldsSimple configuration for multiple workflowsAccess level permissions <p>Essential features</p>  <p>A modern Jira experience for teams who don't need advanced features.</p> <ul style="list-style-type: none">Only show your project's issues on your boardEssential agile reporting	<p>Set up and maintained by your Jira admins.</p> <p>For teams who want to work with other teams across many projects in a standard way. Encourage and promote organizational best practices and processes through a shared configuration.</p> <p>Expert configuration</p>  <p>Benefit from complete control with expert configuration, customization and flexibility.</p> <ul style="list-style-type: none">Set up and maintained by your Jira adminsStandardized configuration shared across projectsComplete control over issue types and custom fieldsCustomizable workflows, statuses and issue transitionsDetailed permission schemes <p>Advanced features</p>  <p>All the power and features that Jira Software is known for.</p> <ul style="list-style-type: none">Pull in issues from other projects on your boardComprehensive agile reporting
Select a team-managed project	Select a company-managed project

Add project details

Add project details

Explore what's possible when you collaborate with your team. Edit project details anytime in project settings.

Required fields are marked with an asterisk *

Name *

SAMPLEPROJ

Key ⓘ *

SAM

Share settings with an existing project

Template

Change template



Scrum

◆ Jira Software

Sprint toward your project goals with a board, backlog, and timeline.

Type

Change type



Company-managed

Work with other teams across many projects in a standard way.

Cancel

Next

Xray project setup

- Go to **Project settings > Apps > Xray Settings**
- There are several configurations to perform

The screenshot shows the Jira Software Project Settings interface for a project named "SAMPLEPROJ". The left sidebar lists various settings categories like Summary, People, Permissions, Notifications, Automation, Features, Toolchain, Workflows, Issues, Types, Layout, Screens, Fields, Collectors, Security, Components, Apps, Microsoft Teams Integration, and Slack Integration. The "Xray Settings" category is highlighted with a red box. The main content area is titled "Summary" and contains a brief summary of Xray configurations. A modal window titled "Xray Issue Types in Project" is open, stating there are 0 Xray issue types configured. It includes a button "Add Xray Issue Types" and a link "Configure the Issue Type Scheme manually". Below this, a table lists Xray Issue Types:

Name	Description	Present in Project
Test	This is the Xray Test Issue Type. Used to define test cases of different types that can be executed multiple times using Test Execution issue.	●
Precondition	This is the Xray Precondition Issue Type. Used to abstract common actions that must be ensured before the test case execution. A Precondition can be associated with multiple test cases.	●
Test Set	This is the Xray Test Set Issue Type. Creates a group of test cases. Used to associate all included Tests with other Xray issue types like Test Execution and Test Plan. A Test Set can also be associated with a requirement issue to provide coverage and test status.	●
Test Plan	This is the Xray Test Plan Issue Type. Used to define the scope of test cases for a given test campaign and to aggregate all executions for those tests displaying the latest result for each test case.	●
Test Execution	This is the Xray Test Execution Issue Type. Used to execute test cases already defined.	●
Sub Test Execution	This is the Xray Sub Test Execution Issue Type. Used to execute test cases already defined. A Sub Test Execution can be created for a parent issue like a requirement in order to execute the test cases associated with it.	●

Below the table, sections for "Miscellaneous", "Test Coverage", "Defect Mapping", and "Test Types" are shown, each with a "Configure" link. At the bottom, it states "The Test Types configured for this project are: Manual, Generic, Cucumber. Configure".

Xray: define “requirements” / the coverage

● Coverable Issue Types: Which issue types do you aim to cover with tests?

(e.g., Story, Epic)

The screenshot shows the 'Xray Settings' interface with the 'Test Coverage' tab selected. On the left, a sidebar lists various settings like Miscellaneous, Remote Jobs Trigger, and Defect Mapping. The main area is titled 'Test Coverage' with a sub-section header: 'A testable/coverable entity is an issue that you can cover with tests in order to validate its acceptance criteria. It can be a Story, Epic, Bug or any custom issue type such as a Requirement.' Below this, there are two columns: 'Available Issue Types' and 'Coverable Issue Types'. The 'Available Issue Types' column contains: Task, Sub-task, Bug, Test, Test Set, Test Plan, Test Execution, Precondition, and Sub Test Execution. The 'Coverable Issue Types' column contains: Story (selected) and Epic. A red arrow points from the 'Story' checkbox in the 'Available Issue Types' list to the 'Coverable Issue Types' list. At the bottom, there are configuration options for 'Test Coverage Hierarchy' (checkbox for 'Epic - Issues(Stories) relation' is checked), 'Issue - Sub-tasks relation' (checkbox is checked), 'Issue Links relation' (dropdown set to 'Select...'), 'Issue Link Type Direction' (radio buttons for 'Outward' and 'Inward'), and a note about Xray using the issue link type configuration for hierarchy between issues. A large red box highlights the 'Save' button at the bottom.

Xray: define defects

- **Defect Issue Types:** *Which issue types do you want to be handled as defects? (e.g., Bug)*

The screenshot shows the 'Xray Settings' interface with the 'Defect Mapping' tab selected. The main section is titled 'Defect Mapping' and contains a brief description: 'A defect represents an error, flaw, failure or fault in the SUT (System Under Test) that produces an incorrect or unexpected result. All the issue types mapped as Defect Entities are handled as defects.' Below this is a 'Defect Issue Types' configuration panel. On the left, a list of 'Available Issue Types' includes: Task, Sub-task, Story, Epic, Test, Test Set, Test Plan, Test Execution, Precondition, and Sub Test Execution. On the right, a list of 'Defect Issue Types' shows 'Bug' selected, indicated by a red arrow pointing from the left list to the right list. A 'Save' button is located at the bottom of the panel.

Xray: final checkup

Go to Project settings > Apps > Xray Settings > Summary.

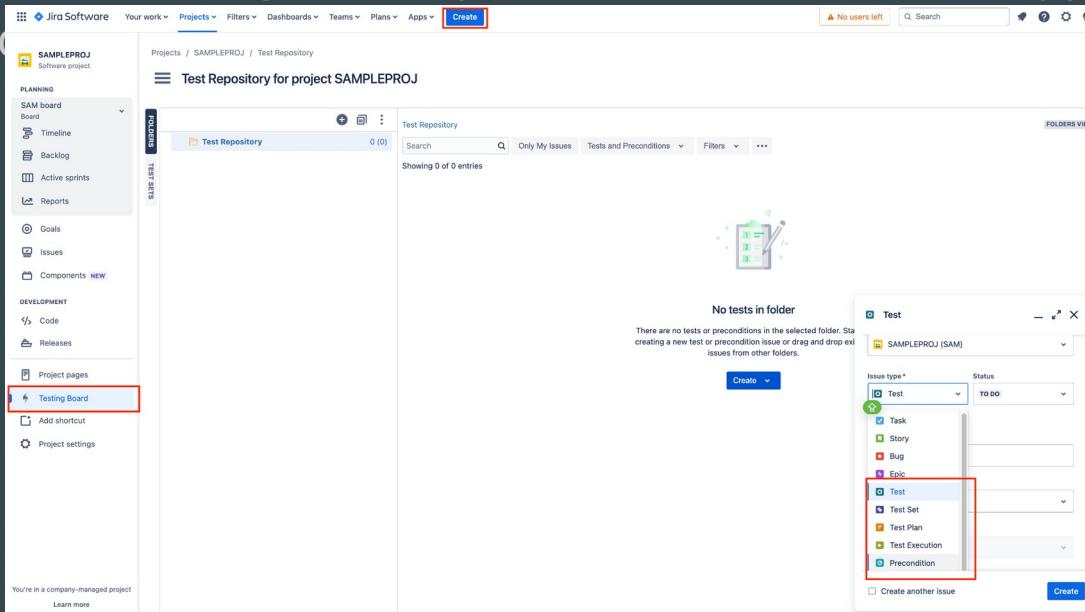
The screenshot shows the 'Xray Settings' section of a project's configuration interface. On the left, a sidebar lists various project settings like Summary, People, Permissions, Notifications, Automation, Features, Toolchain, Workflows, Issues, Types, Layout, Screens, Fields, Collectors, Security, Components, Apps, Microsoft Teams integration, Slack integration, and Development tools. The 'Xray Settings' option is highlighted. The main content area is titled 'Summary' and contains a brief description: 'This page contains a brief summary of the Xray configurations for this project.' Below this is a section titled 'Issue Types' with a sub-section 'Xray Issue Types in Project'. It states there are 6 Xray issue types configured for the project and provides a link to edit them. A modal window titled 'Xray Issue Types in Project' is open, showing a table of issue types:

Name	Description	Present in Project
Test	This is the Xray Test Issue Type. Used to define test cases of different types that can be executed multiple times using Test Execution issues.	✓
Precondition	This is the Xray Precondition Issue Type. Used to abstract common actions that must be ensured before the test case execution. A Precondition can be associated with multiple test cases.	✓
Test Set	This is the Xray Test Set Issue Type. Creates a group of test cases. Used to associate all included Tests with other Xray issue types like Test Execution and Test Plan. A Test Set can also be associated with a requirement issue to provide coverage and test status.	✓
Test Plan	This is the Xray Test Plan Issue Type. Used to define the scope of test cases for a given test campaign and to aggregate all executions for those tests displaying the latest result for each test case.	✓
Test Execution	This is the Xray Test Execution Issue Type. Used to execute test cases already defined.	✓
Sub Test Execution	This is the Xray Sub Test Execution Issue Type. Used to execute test cases already defined. A Sub Test Execution can be created for a parent issue like a requirement in order to execute the test cases associated with it.	✓

Below the table, there are sections for 'Miscellaneous', 'Test Coverage', 'Defect Mapping', 'Test Types', and 'Test Environments', each providing a brief description and a 'Configure' link.

If all goes well...

- You should see a “Testing Board” shortcut on the side menu
- Whenever creating issues, you should see the issue types that were created



DEMO / Try it out, just in case :)

1. Create Epic and a Story; associate Story to the parent Epic
2. Create a Test from the Story screen

You should see coverage information on the Epic and also on the Story issue

The screenshot displays the Jira interface across three main sections: the left sidebar, the Epic screen, and the Story screen.

Left Sidebar: Shows the "client area" selected, with a "Description" field and a "Child issues" section containing "SAM-2".

Epic Screen (SAM-1): Shows the epic "As a user, I can login the web application". It includes a "Test Coverage" section with a table:

Status	Key	Summary	Test Status
TO DO	SAM-3	test for valid login scenario	TO DO

Story Screen (SAM-2): Shows the story "As a user, I can login the web application". It includes a "Test Coverage" section with a table:

Status	Key	Summary	Test Status
TO DO	SAM-3	test for valid login scenario	TO DO

The "Linked issues" section shows "SAM-3" linked to the story. The "Test Coverage" section shows "NOTRUN".

Test Coverage Details: A separate window shows the test "test for valid login scenario" with the status "NOTRUN".

Bottom Right: A note states "There are no steps defined" and provides instructions for creating test steps.

Integrating Xray with Test Automation

Integration with Jira and Xray

Goal 1: Track (i.e., have visibility of) test automation results in Jira, using Xray

Test Execution

(i.e., a batch of test results, composed of multiple Test Runs, one per each Test that was executed)

Details of a Test Run

(i.e., result of this Test in the scope of the Test Execution)

Projects / Spring Tutorial / Test Plans / ST-3 / Test Executions / ST-61 / Test: ST-10

Test 6 of 11

listAllUsersWithSuccess

Execution Status: FAILED | Timer: 00:00:00 | Started On: 9/Apr/2024 04:28 PM | Assignee: Sérgio Freire | Versions: - | Test Version: V1 | Test Environments: java21

Execution Status: Failed | Timer: 00:00:00 | Started On: 9/Apr/2024 04:28 PM | Assignee: Sérgio Freire | Versions: - | Test Version: V1 | Test Environments: java21

No time logged.

Assigned By: Executed By: Revision: 1|merge

Findings (0)

Test details (General)

Test Issue Links (1)

ST-42 REST API endpoint is not returning all users (To Do)

Created

ST-2 As third party system, I can manage users through the REST API (In Progress)

Tests

Results (1)

Context	Output	Duration	Status
Test Suite JUnit Jupiter	<pre>Expecting actual not to be null
AssertionError: Expecting actual not to be null
at com.kmskiveira.java.tutorial.springBoot.UsersControllerTest.listAllUsersWithSuccess(UsersControllerTest.java:116)
at java.base/java.lang.reflect.Method.invoke(Method.java:580)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1586)
at java.base/java.util.Arrays.forEach((Arrays.java:1586)</pre>	-	FAILED

Integration with Jira and Xray

Goal 2: Track coverage related with test automation

- What “requirements” are covered by automated tests?
- What “requirements” given

Test Run details screen

The screenshot shows a test run titled "listAllUsersWithSuccess" that has failed. The execution status is red, indicating failure. The test was started and finished on the same day at 04:28 PM. The assignee is Sérgio Freire. A single test result is shown under the "Results" section, which also includes context and output logs.

Test	Status	Message
ST-2 As third party system, I can manage users through the REST API	FAILED	ST-2 REST API endpoint is not returning all users

Story (or Epic) issue screen

(in this case it is shown as NOK because the latest result failed for one of the Tests

The screenshot shows a story issue titled "As third party system, I can manage users through the REST API". The status is "NOK" because the latest final status is "FAILED". The "Linked issues" section lists several other stories, all marked as "TO DO". The "Test Coverage" section shows a summary table with three entries, all of which are marked as "TO DO".

Status	Key	Summary
TO DO	ST-6	createUserWithSuccess
TO DO	ST-7	deleteUserUnsuccess
TO DO	ST-10	listAllUsersWithSuccess

What test automation frameworks/tools are supported?

You can use any language for developing your automated tests.

Xray processes the reports generated by your test runner, which is usually part of the automation framework.

Your automated scripts can be developed in any language (or tool) if you generate a supported test report ([see here](#)).

Many test runners and tools can generate reports in the "standard" JUnit XML format, which Xray is able to process. Xray also processes a custom, enhanced JUnit XML report containing additional information tailored for it (e.g., issue key of covered Story issue, comments, evidence).

What's the flow?

There are 2 main flows:

1. Common flow (i.e., based on JUnit XML reports)
2. Cucumber specific flows
 - a. To be able to have full integration with Cucumber, it requires some additional step; however, we can follow the “common flow” to keep it simple and if having visibility of overall status of test result is enough; more info ahead

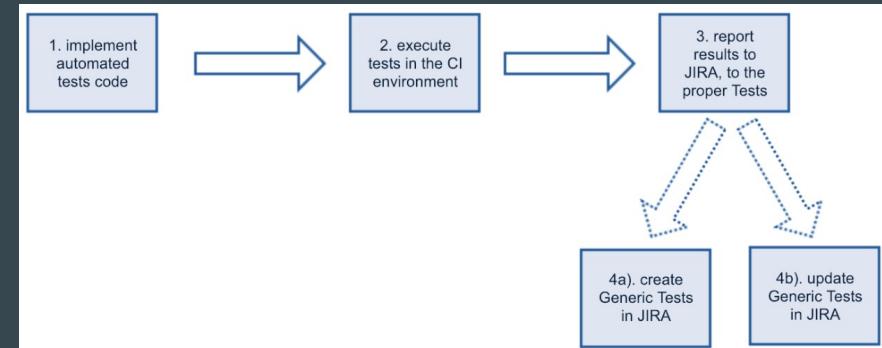
Common flow for having visibility of test automation results

1. Implement test automation code (e.g., in Java + JUnit)
 - a. Use whatever IDE, language and framework; store it in the SCVS (e.g., git)
2. Run the tests, usually in CI/CD
 - a. Using Maven, for example
 - b. Produce a compatible test report; most test runners can output JUnit XML reports as a last resource
3. Push results to Xray
 - a. From the pipeline (or even from your local machine, invoking [Xray's REST API](#))

Common flow for JUnit, TestNG and similar test results

In Xray we don't need to do anything beforehand; we just import the results to it.

- Tests are auto-provisioned the first time results are imported; from them onwards, Tests are reused (just new Test Runs will be created for these Tests)

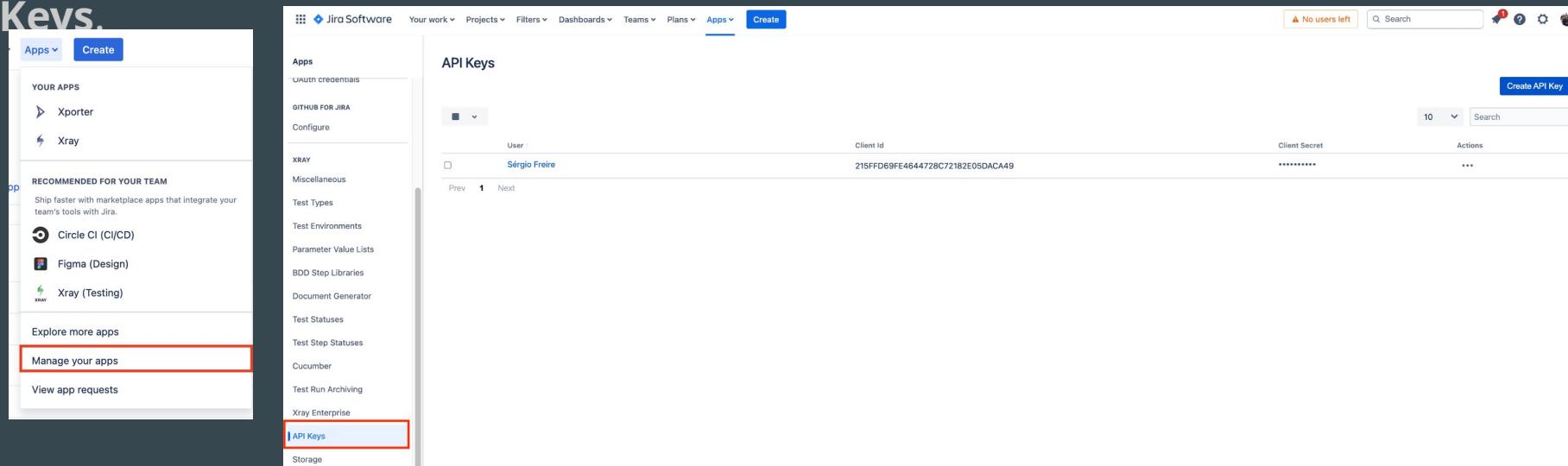


Optionally (depends on framework),

- We can link tests to existing Story issues (i.e., cover them)
- We can report results against existing Tests

Before importing results

Create an “API key” (i.e., a pair of *client id* and *client secret*) to be used on Xray API authentication requests. Go to **Apps > Manage your apps > Xray > Api Keys**.



The screenshot shows the Jira Software interface with the 'Manage your apps' section highlighted by a red box. The 'Xray' app is selected, and its configuration page is displayed. The 'API Keys' section is also highlighted by a red box. The table shows one API key entry:

User	Client Id	Client Secret	Actions
Sérgio Freire	215FFD69FE4644728C721B2E05DACA49	*****	...

Submitting test results

To submit test automation results to Xray/Jira we have several options:

- a. Using a plugin for CI/CD tools, like the free [xray-action](#) GitHub action (see a [tutorial](#))
- b. Using a Maven plugin: [xray-maven-plugin](#) (
[example of a GH workflow in a sample project](#))
- c. Invoking Xray's [REST API](#) endpoints for importing results directly

Submitting test results through the REST API (using “curl”)

To submit results to Xray using its REST API, we need to obtain a token, based on the client id/secret, and then use it on the request that sends the test results to Xray on Jira.

We need to specify the target Jira project, by its key. Optionally, we can also specify a Test Plan by its issue key; this Test Plan needs to exist beforehand (create it manually in Jira). Test Plan is used to group the results from multiple “builds”/iterations.

```
token=$(curl -H "Content-Type: application/json" -X POST --data '{ "client_id": "<CLIENT_ID>" , "client_secret": "<CLIENT_SECRET>" }' https://xray.cloud.getxray.app/api/v2/authenticate)

curl -H "Content-Type: text/xml" -X POST -H "Authorization: Bearer $token" --data @"reports/TEST-junit-jupiter.xml" https://xray.cloud.getxray.app/api/v2/import/execution/junit?projectKey=XPT0&testPlanKey=XPT0-123
```

Note: this way of submitting results may be mostly useful for debugging purposes.

What happens in Jira side?

- A Test Execution issue is created on the target Jira project, containing results for the tests contained in the submitted test report file
- Test issues will be created as abstraction of the original test methods; these are only created unless they don't exist yet

The image shows three screenshots of the Jira interface illustrating the import process:

- Left Screenshot:** Shows the "Execution results [708453141445]" screen. A red box highlights the list of test cases in the "Tests" section, and an orange arrow points from this list to the "Test Plans" field in the middle screenshot.
- Middle Screenshot:** Shows the "Details" screen for a newly created issue. The "Test Plans" field is highlighted with a red box, and an orange arrow points from the "Test Plans" field in the first screenshot to this one.
- Right Screenshot:** Shows the "Test Executions" screen for the target project. It lists two entries: "XT-677 Test Execution for Test Plan XT-674" and "XT-677 Execution results [708453141445]".

Test Results from Left Screenshot:

Rank #	Key	Description	Test Type	Dataset	#Defects	TestRun Assignee	Priority	Linked Issues	Status
1	XT-678	getPersonalizedGreeting	Generic		0	Sérgio Freire	=		PASSED
2	XT-679	don'tCreateUserForInvalidData	Generic		0	Sérgio Freire	=		PASSED
3	XT-680	getUserUnsuccess	Generic		0	Sérgio Freire	=		PASSED
4	XT-681	deleteUserWithSuccess	Generic		0	Sérgio Freire	=		PASSED
5	XT-682	createUserWithSuccess	Generic		0	Sérgio Freire	=		PASSED

If you associated the results to a Test Plan, on it you can see the all the previous results (i.e., Test Executions).

Auto-provisioning of Tests: how does it work?

The diagram illustrates the process of auto-provisioning tests from Java code to a testing tool like Xray. It consists of two main parts: a Java code editor and a Xray test management interface.

Java Code Editor: On the left, a code editor shows a Java file named `UserRestControllerIT.java`. A specific test method, `listAllUsersWithSuccess()`, is highlighted with a red box. The code uses the `MockMvc` framework to test a REST API endpoint for user retrieval.

```
104  
105 @Test  
106 void listAllUsersWithSuccess() {  
107     createTempUser("Amanda James", "amanda", "dummypassword");  
108     createTempUser("Robert Junior", "robert", "dummypassword");  
109  
110     ResponseEntity<List<User>> response = restTemplate  
111         .exchange("/api/users", HttpMethod.GET, null, new ParameterizedTypeReference<List<User>>() {  
112     });  
113  
114     assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);  
115     assertThat(response.getBody()).extracting(User::getName).containsExactly("Sergio Freire", "Amanda James", "Robert Junior");  
116 }  
117
```

Xray Test Management Interface: On the right, a screenshot of the Xray interface shows a test card for `listAllUsersWithSuccess`. The test card includes fields for `Description` (with placeholder text "Add a description...") and `Linked issues` (listing `ST-42` and `ST-2`). The `Test details` section is expanded, showing the `Test Type` as `Generic` and the `Definition` as the full Java code of the `listAllUsersWithSuccess` method.

Two orange arrows point from the highlighted code in the Java editor to the corresponding sections in the Xray test card: one arrow points from the package name in the Java editor to the `Test Type` dropdown in Xray, and another arrow points from the test method name in the Java editor to the `Definition` code block in Xray.

Auto-provisioning of Tests is based on the package name and the test method name.

If you change any of those, you'll have duplicated Tests (you'll need to delete them in Xray).

Linking to user stories

You're writing the test automation code; how do you link it to the user stories, to let Xray know that you're covering that user story with that test?

There are 2 options:

1. Use the standard JUnit XML report produced by surefire or failsafe, upload it as usual, and then link each Test to the Story manually (you'll need to do this for each Test, just once)
2. Use a JUnit 5 extension "xray-junit-extensions", that provides additionally features

Option 1: manually linking Tests to the user stories

After you imported results at least once, go to the Story issue, and choose **Add Tests > Existing tests**. After this, the Test will appear under the Coverage section.

Projects / Xray Tutorials / Add parent / XT-675

As a user, I can see a welcome message

Description
Add a description...

Linked issues
is tested by

XT-676 Test as a user, I can see a welcome message on the root

Test Coverage

Add Tests Execute

New test Existing tests Final Status

Existing test sets

Status Key Summary Test Status

TO DO XT-676 Test as a user, I can see a welcome message on the ... PASSED

Prev 1 Next

OK

Add tests to issue XT-675

Select Search JQL

Start typing to get a list of possible matches or press down to select from a list of existing issues

XT-683 X

Projects / Xray Tutorials / Add parent / XT-675

As a user, I can see a welcome message

Description
Add a description...

Linked issues
is tested by

XT-676 Test as a user, I can see a welcome message on the root

XT-683 getDefaultGreeting

Test Coverage

Add Tests Execute

Analysis & Scope Scope: Latest Final Status

Status Key Summary Test Status

TO DO XT-676 Test as a user, I can see a welcome message on the ... PASSED

TO DO XT-683 getDefaultGreeting PASSED

OK

Note: If we wish to remove/dissociate in terms of coverage, we need to remove the proper issue link in the "Linked Issues".

Option 2: linking Tests to the user stories right from the test code

Use the [xray-junit-extensions](#) which provides the ability to link a test to an existing “requirement” (e.g., Story, Epic) in Jira, using the **@Requirement annotation**

```
import org.xray.junit.customjunitxml.annotations.Requirement;  
...  
  
@Test  
@Requirement("ST-2") // issue key of related Story issue covered by this test  
void listAllUsersWithSuccess() {  
    createTempUser("Amanda James", "amanda", "dummypassword");  
    createTempUser("Robert Junior", "robert", "dummypassword");  
  
    ResponseEntity<List<User>> response = restTemplate  
        .exchange("/api/users", HttpMethod.GET, null, new ParameterizedTypeReference<List<User>>() {  
    });  
  
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);  
    assertThat(response.getBody()).extracting(User::getName).containsExactly("Sergio Freire", "Amanda James", "Robert  
Junior");  
}
```

Ultimately, this Maven plugin will generate an enhanced JUnit XML like report (alternative to the the surefire/failsafe XML reports) with additional information that Xray can take advantage of. It doesn't communicate with Jira or Xray at all!

Option 2: setting up the “xray-junit-extensions”

Add the following dependency to your pom.xml file.

```
<dependencies>
    <dependency>
        <groupId>app.getxray</groupId>
        <artifactId>xray-junit-extensions</artifactId>
        <version>0.8.0</version>
        <scope>test</scope>
    </dependency>

    ...
    <dependency>
        <groupId>app.getxray</groupId>
        <artifactId>xray-maven-plugin</artifactId>
        <version>0.7.5</version>
    </dependency>
</dependencies>
```

Add the optional
[xray-maven-plugin](#)
dependency, if you
want to push the
results to Xray directly
from Maven using a
specific task.

Option 2: setting up the “xray-junit-extensions”

In order to generate the enhanced, customized JUnit XML report we need to register a specific Test Execution Listener:
`EnhancedLegacyXmlReportGeneratingListener`.

- Create a file `src/test/META-INF/services/org.junit.platform.launcher.TestExecutionListener` with the following content:

```
app.getxray.xray.junit.customjunitxml.EnhancedLegacyXmlReportGeneratingListener
```

- By default, the report will be generated under the directory `target/` (`./target/TEST-junit-jupiter.xml`)

More info on the [xray-junit-extensions GitHub project](#).

Tutorials

Check these resources:

- [Tutorial](#) showcasing Spring and integration with Xray
- [Sample project on GitHub](#) showcasing testing Spring apps, fully integrated with Xray (using the xray-junit-extensions to provide specific annotations and to generate an enhanced JUnit XML report with this information that Xray can process)

What about Cucumber?

Cucumber and similar frameworks are a bit different; Cucumber and other Gherkin-based frameworks have specification (based on Gherkin) and the underlying implementation (code).

We have two options:

- I. Treat them as typical automated tests (as mentioned earlier for JUnit), where we just aim to have overall visibility of the test results in Jira
 - A. Generate JUnit XML test report (has less detail, but easier to implement)
 1. Xray will process it as described earlier for standard JUnit tests; in Xray we'll have overall information about the test result but we won't have Gherkin steps based information
 2. **This is simpler to implement in CI as we just need to push the JUnit XML report(s)**
- II. Handle Cucumber specifics
 - A. Generate a Cucumber JSON test report (has more detail, but more work for setup)
 1. This report contains Gherkin statement level information; however, we cannot simply import the results
 2. We need to have Cucumber' Scenarios as Test issues in Xray; we decide the master for editing the Scenarios and then implement the proper flow as consequence of that (see [tutorial](#))

1. Handling Cucumber similar to regular JUnit tests

We can upload the surefire/failsafe JUnit XML tests, as usual; Xray will create Test entities (of type “generic”), if needed, one per each Cucumber Scenario.

The screenshot shows the Xray interface for a project named 'Calculator'. On the left, the 'Execution results' page displays two passed tests: 'Successful purchase flight' and 'Successful purchase flight using declarative style'. A red arrow points from the 'Actions' column of the first test to its detailed view. The detailed view (top right) shows the test is of type 'Generic' with the definition 'Purchase flight.Successful purchase flight'. Another red arrow points from the 'Actions' column of this detailed view to the 'Test Run' section of the execution results page, which lists 'TOTAL TESTS: 2'.

The Test Run (the result report for the Test) just has overall status and elapsed time; there is no Gherkin statement level information; this may be enough

The screenshot shows the 'Test Run' result report for the 'Successful purchase flight' test. It displays the overall status as 'PASSED', the start and end times as '29/04/2024 04:49 PM' and '29/04/2024 04:49 PM', and the duration as '0s 138ms'. The report also includes sections for 'Findings' and 'Test details'.

2. Handling Cucumber specifics

Scenarios (and Backgrounds) must exist in Xray, so that you can report results against them; these cannot be created automatically during the import of test results.

Where are you going to make the Gherkin specification of the Cucumber (test) Scenarios?

Specification may be managed/edited with Xray or not; use one of these approaches (not both):

- A. Xray as the master for editing Scenarios/Backgrounds (as Test and Precondition issues, respectively)
- B. VCS (e.g. Git) as the master to store Scenarios/Backgrounds, while the edition is done by some external IDE (e.g. Eclipse, IntelliJ IDEA, Sublime, VSCode, etc)

In sum, we need to decide (A or B) where we will edit the Cucumber related tests; that will affect the flow we need to implement.

Cucumber: generating the JSON report

To be able to show Gherkin statement/step level information in Xray, we need to generate a Cucumber JSON report.

We can specify that we want the Cucumber JSON report right from the maven CLI, or using the @ConfigurationParameter annotation on the test class.

```
mvn test -Dtest=pt.ua.sergiofreire.blazedemo.BlazedemoTest  
-Dcucumber.plugin="json:target/report.json" ...
```

```
import static io.cucumber.junit.platform.engine.Constants.GLUE_PROPERTY_NAME;  
  
import org.junit.platform.suite.api.ConfigurationParameter;  
import org.junit.platform.suite.api.IncludeEngines;  
import org.junit.platform.suite.api.SelectClasspathResource;  
import org.junit.platform.suite.api.Suite;  
  
@Suite  
@IncludeEngines("cucumber")  
@SelectClasspathResource("pt/ua/sergiofreire/blazedemo")  
@ConfigurationParameter(key = GLUE_PROPERTY_NAME, value = "pt.ua.sergiofreire.blazedemo")  
@ConfigurationParameter(key = Constants.PLUGIN_PROPERTY_NAME,value = "pretty, json:target/report.json")  
public class BlazedemoTest {  
}
```

Warning: Cucumber with JUnit 4

If you see code similar to this, then it's for JUnit 4 and not JUnit 5.

```
package webdemo;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(plugin = {"pretty"})
public class RunCucumberTest {

}
```

Cucumber related tutorials

Check these:

- Tutorial detailing the 2 possible flows related with Cucumber and Xray integration
- GH repo using Cucumber + Selenium, supporting the 2 flows

Reporting

The different ways of reporting

Xray provides several ways of reporting:

- Built-in, tailored reports ([more info](#)) => mostly for your awareness
 - Test Coverage and Traceability Reports are 2 of the most important ones, but there's more
- Gadgets to be used on standard Jira dashboards ([more info](#))
 - The way to share information in and between teams, in near real-time

Some built-in reports are also available as gadgets but not all of them. Teams usually use a mix of approaches, depending on their needs.

Built-in Reports (a few examples)

Accessing built-in reports

Built-in reports can be accessed from several places.

- Testing Board > (hamburger menu) > *report*
- Apps > Xray > Reports > *report*

The screenshot shows the Jira Software interface with the 'Testing Board' feature highlighted. On the left, there's a sidebar with various project management sections like Planning, Development, and Components. The main area displays a 'usability test' board with execution status (FAILED), findings (Defects, Evidence), and comments. A red box highlights the 'Testing Board' option in the sidebar.

The screenshot shows the Xray application integrated into Jira Software. The 'Test Coverage Report' is selected in the sidebar. The main panel displays analysis and scope settings, including 'Scope: Latest Final Status' and 'Project: Calculator Issue Type: Epic, Story'. A large central area features a laptop icon with gears, and a message at the bottom says 'Please choose the calculation scope and set the filters above in order to generate the report.' A 'Generate Report' button is visible at the bottom right.

Typical actions/inputs on built-in reports

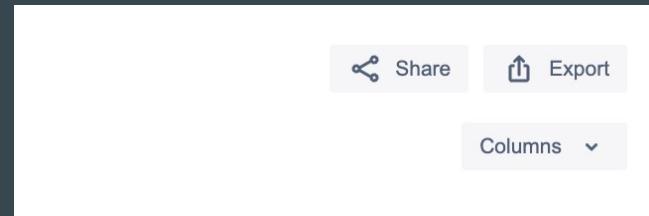
Left side

- Analysis & Scope
 - How to analyze the given data
- Filters
 - What data to use as source data
- Group by
- View



Right side

- Export (to CSV)
- Share
 - Create a unique URL that can be shared
- Columns / Show
 - show/hide some information



Built-in report: Traceability

Provides traceability from “requirements” up to defects.

Traceability works in the forward way:

**Parent requirement > sub requirement(s) > Test(s) > Test Run(s) >
Defect(s)**

Epic > Stories > Tests > Test Runs > Bugs

Built-in report: Traceability

It's dynamic: depends on scope (i.e., the version and environment we want to analyze it on)

Quick filters for requirements based on their coverage status on the selected "scope" (e.g., version and/or environment; or even just using a Test Plan)

- Show me just the requirements having problems...
- Show me just the uncovered issues...

The screenshot shows a traceability report interface with the following sections:

- Analysis & Scope:** Scope: Version Version: v1.0 Final Status, Project: Calculator Issue Type: Epic, Story Labels: MOT, Show Test Runs.
- Filters:** Scope: Version Version: v1.0 Final Status, Project: Calculator Issue Type: Epic, Story Labels: MOT.
- QUICK FILTERS:** OK (1), NOK (1), NOTRUN (0), UNKNOWN (0), UNCOVERED (0).
- REQUIREMENTS:** A list of requirements with their status (e.g., TO DO, IN PROGRESS, UNCOVERED) and descriptions. Requirements include:
 - CALC-1120: TO DO, Fix Version/s: v1.0, Description: As a user, I can perform restricted operations in my ... (Status: NOK)
 - CALC-2657: TO DO, Fix Version/s: v1.0, Description: As a user, I can manage my profile details (Status: UNCOVERED)
 - CALC-795: IN PROGRESS, Fix Version/s: v1.0, Description: As a user, I can login the web application (Status: NOK)
- TESTS:** A list of test cases with their status (e.g., TO DO, IN PROGRESS, PASSED). Tests include:
 - CALC-797: TO DO, Fix Version/s: v1.0, Description: Valid Login using Gherkin style (Status: PASSED)
 - CALC-1449: TO DO, Fix Version/s: v1.0, Finished On: 05May/22 03:15 PM, Executed By: Sérgio Freire, Test Environments: -, Test Version: v1 (Status: PASSED)
 - CALC-1428: TO DO, Fix Version/s: v1.0, Finished On: 28Apr/22 03:42 PM, Executed By: Sérgio Freire, Test Environments: -, Test Version: v1 (Status: PASSED)
 - CALC-1427: TO DO, Fix Version/s: v1.0, Finished On: 27Apr/22 04:02 PM, Executed By: Sérgio Freire, Test Environments: -, Test Version: v1 (Status: PASSED)
 - CALC-2604: TO DO, usability test, Fix Version/s: v1.0, Finished On: 25Jul/22 03:19 PM, Executed By: Sérgio Freire, Test Environments: -, Test Version: v1 (Status: FAILED)
 - CALC-2661: TO DO, login button too small, Fix Version/s: v1.0, Finished On: 25Jul/22 03:19 PM, Executed By: Sérgio Freire, Test Environments: -, Test Version: v1 (Status: FAILED)
- DEFECTS:** A section showing defects related to the requirements and tests.

Built-in report: Test Coverage

Provides a birds-eye overview of the (coverage) status of your deliverables (e.g., stories, epics) based on their latest testing results.

It answers this simple question: **What are the impacts of my testing?**

But also,

- *How are my stories on version 2.0 of the SUT?*
- *Are my highest priority stories covered by tests?*
- *How are the requirements related to some component?*
- *Which is the test completeness of certain requirements?*
- ...

Built-in report: Test Coverage

It's dynamic: depends on scope (i.e., the version and environment we want to analyze it on), considering the latest results obtained on it

- Output will most likely be different on two different versions



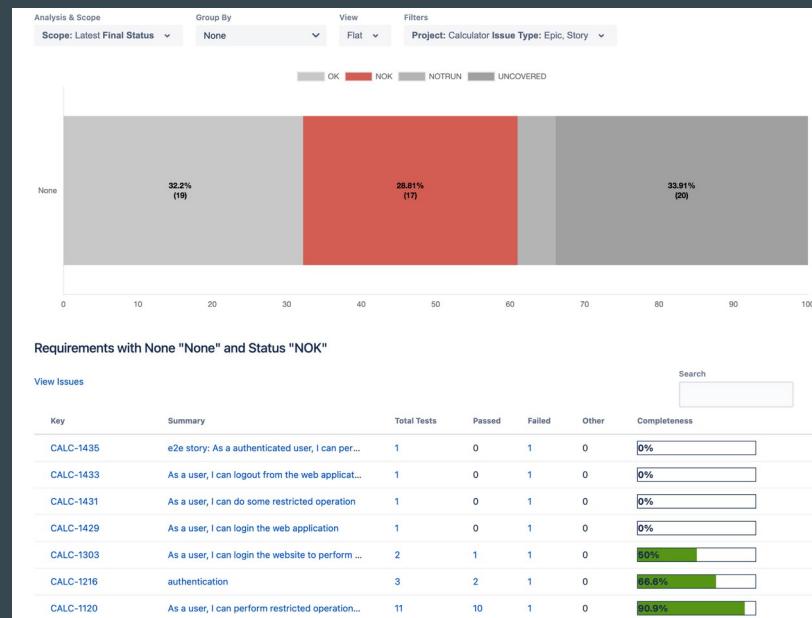
Status meaning:

- OK: covered, and all related tests are passing
- NOK: covered, but one of the related tests is failing
- NOTRUN: covered, but some of the related tests are yet to be run
- UNCOVERED: has no tests

Built-in report: Test Coverage

Drill-down, by clicking on a bar, to see:

- List of related “requirements”
- Test completeness of each requirement
- Total tests that cover each requirement



Built-in report: Test Coverage

Group by component, priority, or other compatible field on the “requirements”.



Jira dashboards with Xray Gadgets

Quality related information: quick considerations

What do we want to answer?

Who is going to look at it?

Is that information understandable and actionable by the team?

Quality related information

Focus on: **testing results at high-level**

- *What were the testing results, namely from test automation ran in the pipeline?*
- How to achieve it? In different ways depending on what we aim to show...
 - "Test Executions List" gadget, if we want to see each "batch" of test results (i.e., Test Execution issues), usually coming from a specific build on the pipeline
 - Here we see each build result
 - "Test Plans List" gadget, if we want to see the consolidated results from multiple "batch" of test results (i.e., from multiple builds on the pipeline)
 - If you create a Test Plan manually in Jira, and then report the results back to it, the Test Plan will show the consolidated results and on each test you can track the runs you obtained for it
 - Here we see the consolidated results from multiple builds
 - "Overall Test Results" gadget, if you want to show how are your tests, in a pie-chart, grouped by their latest result

Quality related information

Focus on: “**Requirements**” / epics & user stories

- *How are our deliverables given the latest testing results? Are they ready to be released?*
- How to achieve it?
 - “Overall Test Coverage” gadget, for an overall perspective at high-level
 - “Requirements List” gadget, for a detailed, per user story/epic, information

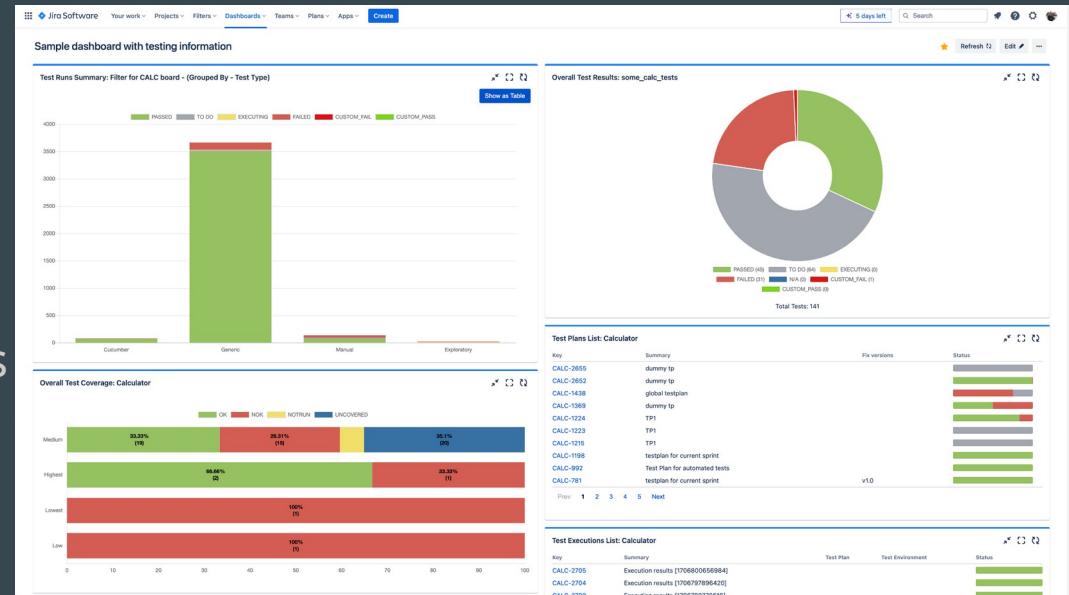
In this case we’re focused on the impacts of our testing. What does it tell about the quality of our deliverables?

Jira dashboards

The typical way to make shareable, near real-time reports, composed of multiple gadgets. Go to **Dashboards** top menu shortcut and create a new dashboard.

In a dashboard, you can use:

- Jira default gadgets
- Xray gadgets
- Gadgets from other apps



Xray gadgets

Listings:

- Requirements List
- Tests List
- Test Sets List
- Test Executions List
- Test Plans List
- Test Runs List

Aggregated or Calculated:

- Overall Test Coverage
- Overall Test Results
- Test Runs Summary
- Tests by Test Type

Xray gadgets: how source data is specified

Depends on gadget, but source data for gadgets can include:

- Saved filter
- Project
- Issue picker

And eventually...

- Test Runs related info (e.g., status, executed by, dates)
- Tests related info (e.g., priority, component)
- ...

Xray gadgets: Requirements List

Show list of requirements, some attributes, and the coverage status for some scope.

Define the scope for calculating coverage status

Requirements List: Calculator

Key	Summary	Priority	Components	Status
CALC-2703	As a user, I can login the web application	=		OK
CALC-2667	As a user, I can calculate the sum of 2 numbers	=		OK
CALC-2664	dummy story	=		UNCOVERED
CALC-1435	e2e story: As a authenticated user, I can perform so...	=		NOK
CALC-1433	As a user, I can logout from the web application	=		NOK
CALC-1431	As a user, I can do some restricted operation	=		NOK
CALC-1429	As a user, I can login the web application	=		NOK
CALC-1388	dummy story	=		NOTRUN
CALC-1345	dummy story	=		OK
CALC-1222	As system, the calculator changes to sleep mode wit...	=		UNCOVERED

Prev 1 2 3 4 5 6 Next

Requirements List: Calculator

Project or Saved Filter: Calculator

Number of results: 10

Columns to display: Key, Summary, Priority, Components

Flat Test Coverage View

Analysis & Scope

Calculate the Test Coverage for the following scopes.

Latest Version Test Plan

Project: Calculator

Version: v1.0

Test Environment: All Environments

Final statuses have precedence over non-final.

Auto-generate gadget name

If this option is enabled, Xray will generate the gadget name based on its 'Rename' action.

Save Cancel

Xray gadgets: Test Plans List

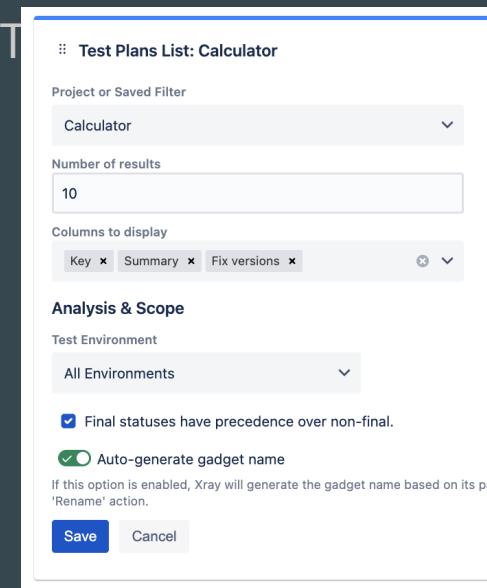
Show a list of Test Executions, some attributes, and overall progress.

- Track consolidated progress considering the results on all Test Environments or just progress on a specific Test Environment
- Similar to the Test Plans List Report

Test Plans List: Calculator

Key	Summary	Fix versions	Status
CALC-2655	dummy tp		
CALC-2652	dummy tp		
CALC-1438	global testplan		
CALC-1369	dummy tp		
CALC-1224	TP1		
CALC-1223	TP1		
CALC-1215	TP1		
CALC-1198	testplan for current sprint		
CALC-992	Test Plan for automated tests		
CALC-781	testplan for current sprint	v1.0	

Prev 1 2 3 4 5 Next



Xray gadgets: Test Executions List

Show a list of Test Executions, some attributes, and overall progress.

- Track in which Test Plan, Test Env., Fix Version, Revision they were performed
- Similar to

Test Executions List: Calculator				
Key	Summary	Test Plan	Test Environment	Status
CALC-61	Test Execution of XRAYintegration of the test WANI...	CALC-12	DEV	<div style="width: 50%;"></div>
CALC-60	Test Execution of XRAYintegration of the test WANI...	CALC-12	DEV	<div style="width: 50%;"></div>
CALC-58	Exploratory session for CALC-57		CHROME	<div style="width: 100%; background-color: #f08080;"></div>
CALC-56	Ad-hoc execution for CALC-55			<div style="width: 100%; background-color: #9acd32;"></div>
CALC-53	Exploratory session for CALC-52		FIREFOX	<div style="width: 100%; background-color: #f08080;"></div>
CALC-49	Exploratory session for CALC-46			<div style="width: 100%; background-color: #ffff00;"></div>
CALC-47	Ad-hoc execution for CALC-46		CHROME	<div style="width: 100%; background-color: #f08080;"></div>
CALC-45	Ad-hoc execution for CALC-44			<div style="width: 100%; background-color: #ffff00;"></div>
CALC-42	Exploratory session for CALC-41			<div style="width: 100%;"></div>
CALC-39	Test Execution of XRAYintegration of the test WANI...	CALC-12	DEV	<div style="width: 50%;"></div>
Prev	1	...	59	60
	61	62	63	Next

Xray gadgets: Test Runs List

Show a list of Test Executions, some attributes, and overall progress.

- Track in which Test Environment, Fix Version, Revision they were performed
- Tracked by who executed by, etc

Key	Test Execution Key	Summary	Test Execution Version	Test Environment	Status
CALC-3	CALC-8	Cucumber Test As a user, I can sum two numbers	v3.0		PASSED
CALC-4	CALC-8	Generic Test As a user, I can sum two numbers	v3.0		PASSED
CALC-2	CALC-11	Manual Test As a user, I can sum two numbers	v3.0		PASSED
CALC-14	CALC-13	tete	v3.0		TO DO
CALC-29	CALC-30	Test epic1			CUSTOM_FAIL
CALC-17	CALC-33	WANImpact Local	v3.0	DEV	FAILED
CALC-44	CALC-45	risks around screensize			EXECUTING
CALC-46	CALC-47	assess login usability risks	v3.0	CHROME	FAILED
CALC-46	CALC-49	assess login usability risks	v3.0		EXECUTING
CALC-52	CALC-53	assess usability risks in the login page	v3.0	FIREFOX	FAILED

Test Runs List: Calculator

Project or Saved Filter: Calculator

Test Filter: Priority: Select... Component: Select...

Contains: Status: Select...

Test Run Filter: Assignee: Executed By: Type to search

Status: Select...

Enable date range filtering

Number of results: 10

Columns to display: Key, Test Execution Key, Summary, Test Execution Version, Test Environment

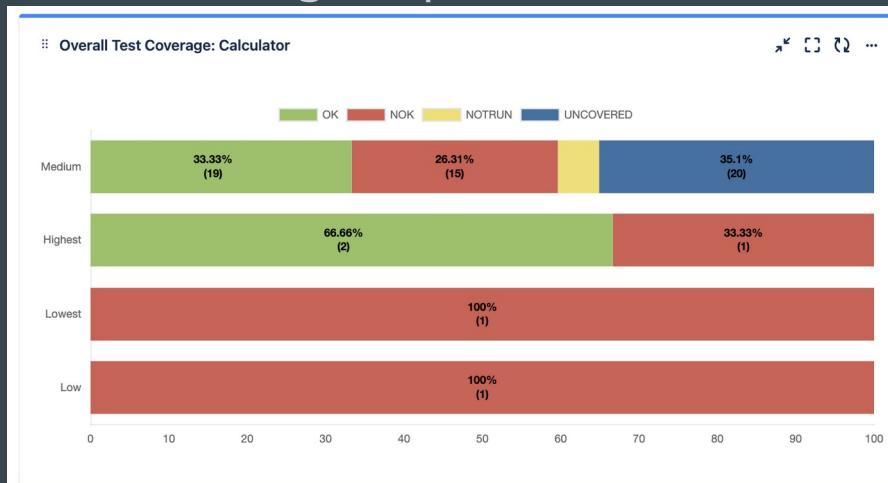
Auto-generate gadget name: If this option is enabled, Xray will generate the gadget name based on its parameters. Otherwise, you can explicitly use the 'Rename' action.

Save Cancel

Xray gadgets: Overall Test Coverage

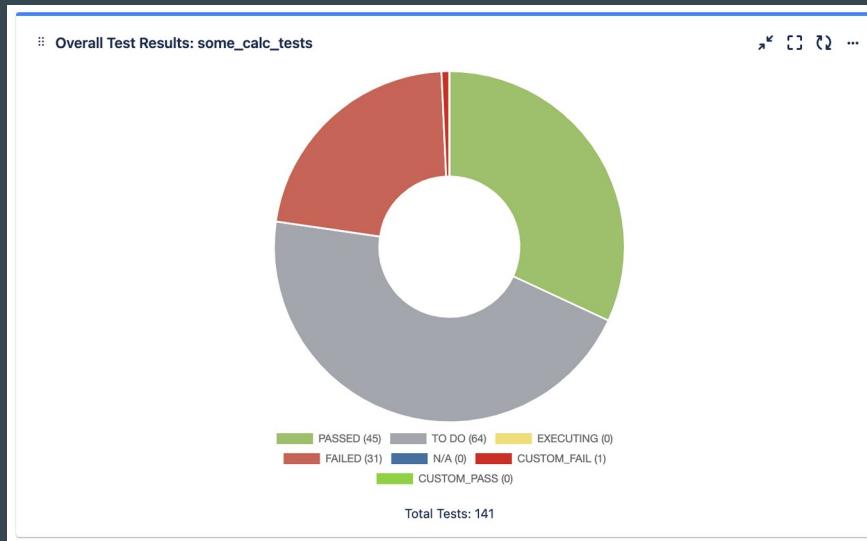
Provides a birds-eye overview of the (coverage) status of your deliverables (e.g., stories, epics) based on their latest testing results.

- Similar to the Test Coverage Report



Xray gadgets: Overall Test Results

Shows how the given Tests are in some “scope” (e.g., version and/or Test Env.).



Overall Test Results: some_calc_tests

Project or Saved Filter

some_calc_tests

Analysis & Scope

Calculate the Test Coverage for the following scopes.

Latest Version Test Plan

Test Environment

All Environments

Final statuses have precedence over non-final.

Auto-generate gadget name

If this option is enabled, Xray will generate the gadget name based on its explicitly define a name for the gadget using the 'Rename' action.

Save Cancel

Example of dashboard

Jira Your work Projects Filters Dashboards Teams Plans Apps Create

No users left Q Search

Refresher Edit ...

TQS: Spring Tutorial Dashboard

Overall Test Coverage: Spring Tutorial

Component	Status	Count
backoffice	UNCOVERED	1
None	NOK	1
REST	NOK	1
frontend	OK	1

Requirements List: Spring Tutorial

Key	Summary	Priority	Components	Status
ST-53	As a admin, I can manage users in a web based back...	!	backoffice	UNCOVERED
ST-30	User management	!		NOK
ST-2	As third party system, I can manage users through t...	!	REST	NOK
ST-1	As a user, I can access a landing page	!	frontend	OK

Prev 1 Next

Filter Results: ST_defects

T	Key	Summary	P
ST-42	REST API endpoint is not returning all users	!	

1-1 of 1 Just now

Overall Test Results: Spring Tutorial

Status	Count
PASSED	10
EXECUTING	0
TO DO	0
FAILED	1
N/A	0
CUSTOM FAIL	0
CUSTOM PASS	0

Total Tests: 11

Test Executions List: Spring Tutorial

Key	Summary	Test Plan	Test Environment	Revision	Status
ST-63	Execution results [1712676935799]	ST-3	JAVA17	1/merge	
ST-62	Execution results [1712676935927]	ST-3	JAVA21	1/merge	
ST-61	Execution results [1712676509957]	ST-3	JAVA21	1/merge	
ST-60	Execution results [1712676503076]	ST-3	JAVA17	1/merge	
ST-59	Execution results [1712676503453]	ST-3	JAVA21	1/merge	
ST-58	Execution results [1712590471087]	ST-3	JAVA17	1/merge	
ST-57	Execution results [1712590126808]	ST-3	JAVA21	1/merge	
ST-56	Execution results [1712590123672]	ST-3	JAVA17	1/merge	
ST-55	Execution results [1712589857267]	ST-3	JAVA21	1/merge	
ST-54	Execution results [1712589852634]	ST-3	JAVA17	1/merge	

Prev 1 2 3 4 Next

Test Plans List: Spring Tutorial

Key	Summary	Fix versions	Status
ST-3	testing for current sprint		