**JIRA**

**(Only this answer from copilot)**

**Jira** is a powerful project management tool developed by Atlassian, widely used for tracking issues, bugs, and managing projects, especially in software development. Jira is a tool used by teams to manage their work. Think of it as a digital whiteboard where you can keep track of tasks, bugs, and projects. It’s especially popular in software development but can be used in many other fields too

**Key Features of Jira:**

1. **Boards:** 
   * **Scrum Boards:** It uses Sprints. A sprint is a predetermined amount of time where teams determine the work to get done. Imagine you have a list of tasks for a two-week period (called a sprint). A Scrum board helps you organize these tasks, showing what needs to be done, what’s in progress, and what’s completed.
     + **Example:** Your team is developing a new app. You create a sprint with tasks like “Design login screen,” “Develop login functionality,” and “Test login feature.” The Scrum board helps you track these tasks over the two weeks.
   * **Kanban Boards:** These are like a continuous to-do list. Tasks move from “To Do” to “In Progress” to “Done” without a set time frame.
     + **Example:** Your customer support team uses a Kanban board to handle support tickets. New tickets go into “To Do,” get worked on in “In Progress,” and move to “Done” once resolved.
2. **Issues/Tickets:** These are individual work items or tasks. Each issue can be a bug, a task, or a new feature request.
   * + **Example:** A user reports a bug in your app. You create an issue in Jira titled “Fix login bug,” assign it to a developer, and track its progress until it’s resolved.
3. **Workflows:** Workflows define the steps an issue goes through from start to finish. You can customize these steps to match how your team works.
   * + **Example:** For a bug, the workflow might be “Open” -> “In Progress” -> “Code Review” -> “Testing” -> “Done.”
4. **Reports and Dashboards:** Jira provides visual reports and dashboards to help you see how your team is doing. You can track progress, identify bottlenecks, and make data-driven decisions.
   * + **Example:** You create a dashboard showing the number of bugs fixed each week, the progress of current sprints, and the workload of each team member.
5. **Integrations:** Jira can connect with other tools you use, like Confluence for documentation or Bitbucket for code repositories.
   * + **Example:** When a developer commits code to Bitbucket, it automatically updates the related Jira issue, showing that the code is ready for review.

**Use Cases:**

* **Software Development:** Managing tasks, tracking bugs, and planning releases.
  + **Example:** Your development team uses Jira to plan sprints, track progress, and release new features on time.
* **Project Management:** Organizing and tracking projects across different departments.
  + **Example:** Your HR team uses Jira to manage recruitment processes, from posting job ads to onboarding new hires.
* **Service Management:** Handling customer support tickets and IT service requests.
  + **Example:** Your IT team uses Jira to track and resolve technical issues reported by employees.

**Understanding the Basics of Agile**

Agile is a way of managing projects that helps teams work more efficiently and adapt to changes quickly. It’s like a set of rules and practices that guide how teams plan, develop, and deliver projects.

* **Key Principles of Agile:**

1. **Adaptive Planning:** Plans can change as the project progresses.
2. **Early Delivery:** Deliver small parts of the project early and often.
3. **Continuous Improvement:** Always look for ways to improve the process.
4. **Flexible Response to Change:** Be ready to change plans based on new information.

* **Agile Boards:** An Agile board is a visual tool that helps teams track their work. It has columns that represent different stages of the work process.

**Scrum vs. Kanban:** Agile has two popular ways to organize work: Scrum and Kanban.

1. **Scrum:** A **Scrum board** is a visual tool used to manage and track the progress of tasks during a sprint. A sprint is a fixed time period (usually 1-4 weeks) during which specific tasks are completed.
2. **Kanban:** A Kanban board is a visual tool used to manage and track tasks in a project. It uses columns to represent different stages of work, such as “To Do,” “In Progress,” and “Done,” allowing teams to see the status of tasks at a glance and ensure a steady flow of work.

**Stories and Epics in Agile**

* **Stories:** A **story** is a short, simple description of a feature or functionality from the perspective of the end user. It follows a specific format to ensure clarity and focus on the user’s needs.

**Format of a User Story:** As a [type of user], I want [some goal] so that [some reason].

**Examples:** As a web developer, I want to add users to Jira so that my coworkers can report bugs

* **Epics:** An **epic** is a large body of work that can be broken down into smaller tasks, known as user stories. Epics are used to organize and manage large projects that span multiple sprints or iterations.

**Example:** Let’s say your team is developing a note-taking app. The **epic** could be “Develop core functionality of the note-taking app.”

**Issues and Projects in JIRA**

* **Issues:** In Jira, an **issue** is any task, bug, feature request, or any other work item that you want to track. Think of issues as digital post-it notes where you can write down what needs to be done.

**Key Points About Issues:**

* **Fields:** Issues contain fields that hold specific information, such as the assignee (who is responsible for the task), summary (a brief description), due date, and detailed description.
* **Types of Issues:** Jira comes with several predefined issue types like stories, bugs, and tasks. You can also create custom issue types to fit your needs.

**Example:** Imagine you have a post-it note for a task:

* **Summary:** Fix login bug
* **Assignee:** John Doe
* **Due Date:** October 20, 2024
* **Description:** Users are unable to log in with their credentials.

This post-it note represents an issue in Jira. You can have different types of post-it notes (issues) for different purposes, like bugs, tasks, or feature requests.

* **Projects:**

A **project** in Jira is a collection of issues. Projects help you organize and manage related issues together. Think of a project as a folder that contains all the post-it notes (issues) related to a specific goal or area of work.

**Key Points About Projects:**

* **Containers for Issues:** Projects hold issues. Each project can have its own set of issue types and workflows.

**Example:**

Let’s say you have three projects:

1. **Web Development Project:** Contains issues related to developing a website.
   * **Issue Types:** Stories, bugs, tasks.
   * **Example Issue:** “Fix login bug” (bug type).
2. **Marketing Project:** Contains issues related to marketing campaigns.
   * **Issue Types:** Tasks, stories.
   * **Example Issue:** “Create social media plan” (task type).
3. **Customer Service Project:** Contains issues related to customer support.
   * **Issue Types:** Tasks, incidents.
   * **Example Issue:** “Resolve customer complaint” (incident type).

Each project has its own set of issues that are relevant to the work being done in that project. For example, the web development project might have bugs and feature requests, while the marketing project might have tasks related to campaign planning

**Types of Project in Jira**

There are 2 types of project in Jira:

* 1. **Team-Managed Projects:** Team-managed projects are designed to be simple and flexible, allowing teams to set up and manage their own projects without needing special permissions.

**Key Points:**

* **Introduced in 2018:** Originally called next-gen projects, renamed to team-managed projects in 2021.
* **No Admin Permissions Needed:** Anyone on the team can create and manage these projects.
* **Self-Contained:** All settings and configurations (like issue types, workflows, and fields) are specific to that project and cannot be shared with other projects.

**Example:** Imagine your marketing team wants to track their campaigns. They can quickly set up a team-managed project with issue types like “Campaign,” “Task,” and “Event.” They can customize the workflow to fit their needs without needing help from a Jira administrator.

* 1. **Company-Managed Projects:** Company-managed projects are more structured and require administrative permissions to set up and manage. They are ideal for larger organizations that need consistency and shared configurations across multiple projects.

**Key Points:**

* **Originally Called Classic Projects:** Renamed to company-managed projects in 2021.
* **Admin Permissions Required:** Only Jira administrators can create and manage these projects.
* **Shared Configurations:** Settings like issue types, workflows, and fields can be shared across multiple projects.

**Example:** Your IT department needs to track software development and support tickets. They use a company-managed project where the issue types “Bug,” “Task,” and “Story” are shared across all IT projects. This ensures consistency and allows for centralized management by Jira administrators

**Team Managed Project Settings**

**Overview of Project Settings:**

1. **Details:**
   * **Icon:** You can change the project icon to a custom image or select from pre-made icons. This icon helps you quickly identify the project in Jira.
   * **Name:** The name of your project, which you can change if needed.
   * **Key:** A unique identifier for your project. This key appears in the issue IDs (e.g., WEB2-1).
   * **Project Lead:** Assign a person who will be the main point of contact for the project.
   * **Default Assignee:** Decide who will be assigned new issues by default. You can set it to a specific person or leave issues unassigned.
2. **Board Settings:** **Columns and Statuses:** Customize the columns on your board and map them to issue statuses. For example, “To Do,” “In Progress,” and “Done.”
   * **Example:** If you move an issue from “To Do” to “In Progress,” its status changes accordingly.
3. **Custom Filters:** Create filters to view specific types of issues.
   * **Example:** Create a filter to show only bugs or only tasks assigned to a specific person.
4. **Card Cover Images:** Enable or disable images on issue cards. If an issue has an attached image, it will show up on the board.
   * **Example:** Attach an image to an issue, and it will appear as a cover image on the board.
5. **Access:** Control who can see and edit the project.
   * **Options:**
     + **Private:** Only invited users can see the project.
     + **Limited:** Anyone with access to Jira can see the project but only in read-only mode.
     + **Open:** Anyone with access to Jira can view and edit the project.
6. **Issue Types:** Manage the types of issues in your project (e.g., Task, Bug, Story).
   * **Example:** You can create custom issue types like “Idea” and add custom fields to them.
7. **Automation:** Set up automated rules to perform actions based on triggers.
   * **Example:** Automatically assign new issues to a specific person or send notifications when an issue is updated.
8. **Notifications:** Customize who gets notified about changes in the project.

**Example:** Set notifications for issue creation, updates, and comments.

1. **Features:** Enable or disable features like the roadmap, sprints, or backlogs.
   * **Example:** Turn on the roadmap feature to plan and visualize your project’s timeline.

**Jira for Managers**

**(Notes aren’t that much helpful, you’ll need to practice JIRA)**

**Scrum and Kanban Boards**

**Scrum Board:**

1. **Structure:**
   * **Sprints:** Scrum boards use sprints, which are fixed time periods (usually 1-4 weeks) during which specific tasks are completed.
   * **Columns:** Common columns include “To Do,” “In Progress,” and “Done.”
2. **Workflow:**
   * **Sprint Planning:** At the beginning of each sprint, the team plans which tasks will be completed.
   * **Daily Standups:** Short daily meetings to discuss progress and obstacles.
   * **Sprint Review:** At the end of the sprint, the team reviews what was completed and plans for the next sprint.
3. **Focus:**
   * **Time-Boxed:** Work is divided into sprints, providing clear deadlines and goals.
   * **Predictability:** Helps teams predict how much work they can complete in a sprint.
4. **Example:**
   * **Sprint Goal:** Complete the “Homepage Redesign” in a two-week sprint.
   * **Tasks:** “Design homepage layout,” “Develop homepage,” “Test homepage.”

**Kanban Board:**

1. **Structure:**
   * **Continuous Flow:** Kanban boards do not use sprints. Instead, tasks are continuously added and moved through stages.
   * **Columns:** Common columns include “To Do,” “In Progress,” and “Done.”
2. **Workflow:**
   * **No Fixed Time Periods:** Work is not divided into sprints. Tasks are added and completed as needed.
   * **WIP Limits:** Work In Progress (WIP) limits restrict the number of tasks in each column to prevent bottlenecks.
3. **Focus:**
   * **Flexibility:** Allows for continuous delivery and adaptation to changes.
   * **Flow Efficiency:** Focuses on maintaining a steady flow of work.
4. **Example:**
   * **Ongoing Tasks:** “Fix login bug,” “Update user profile page,” “Improve site performance.”
   * **WIP Limit:** Only 3 tasks can be “In Progress” at any time to ensure quality and focus.

**Key Differences:**

| **Feature** | **Scrum Board** | **Kanban Board** |
| --- | --- | --- |
| **Time Management** | Uses fixed sprints (1-4 weeks) | Continuous flow, no fixed time periods |
| **Planning** | Sprint planning at the start of each sprint | Continuous planning, tasks added as needed |
| **Meetings** | Daily standups, sprint reviews | No required meetings, but can have regular check-ins |
| **Focus** | Time-boxed work, predictability | Flexibility, flow efficiency |
| **WIP Limits** | Not typically used | Essential to prevent bottlenecks |
| **Adaptability** | Less flexible within a sprint | Highly adaptable, changes can be made anytime |

**When to Use Each:**

* **Scrum:**
  + Best for teams that benefit from structured, time-boxed work periods.
  + Ideal for projects with clear goals and deadlines.
  + Useful when you need to measure progress and predictability.
* **Kanban:**
  + Best for teams that need flexibility and continuous delivery.
  + Ideal for projects with ongoing tasks and frequent changes.
  + Useful when you want to maintain a steady flow of work and avoid bottlenecks.

**Navigating the Scrum Board in Jira:**

1. **Breadcrumb Navigation:** At the top, you have a breadcrumb trail that shows the hierarchy of your project and board.
   * **Example:** “All Projects > Based on a True Story > BOATS Scrum Board.”
2. **Active Sprint:** The Scrum board shows only the issues that are part of the current active sprint.
   * **Example:** If the current sprint includes “Design homepage” and “Develop login feature,” only these tasks will appear on the board.
3. **Backlog:** The backlog contains all the tasks that need to be done but are not yet part of an active sprint.
   * **Example:** Tasks like “Create user profile page” might be in the backlog, waiting to be added to a future sprint.

**Insights on the Scrum Board**

**What are Insights on a Scrum Board?**

**Insights** provide a quick overview of the progress and performance of your current sprint. They help you understand how well your team is doing and identify any potential issues early.

**Key Components of Insights:**

1. **Sprint Progress:** Shows the overall progress of the sprint.
   * **Example:** If your sprint goal is to complete 10 tasks and 6 are done, the progress bar will show 60% completion.
2. **Burndown Chart:** A visual representation of the work remaining in the sprint over time.
   * **Example:** If you start with 100 hours of work and complete 20 hours each day, the chart will show a downward trend.
3. **Work Logged:** Displays the amount of work logged by team members.
   * **Example:** If a task is estimated to take 5 hours and 3 hours have been logged, it shows the progress on that task.
4. **Remaining Work:** Shows the amount of work left to complete in the sprint.
   * **Example:** If there are 40 hours of work left and 5 days remaining, it helps you gauge if the team is on track.

**Using Epics in Jira**

**What is an Epic?**

An **epic** is a large body of work that can be broken down into smaller tasks, known as user stories or issues. Epics typically span multiple sprints and help organize and manage large projects.

**Steps to Use Epics in Jira:**

1. **Creating an Epic:**
   * Go to your project and click on the **Create** button.
   * Select **Epic** as the issue type.
   * **Example:** Name your epic “Q1 Episodes” if you’re planning to release multiple podcast episodes in the first quarter.
   * **Epic Name vs. Summary:** The **Epic Name** is a short identifier that appears in the backlog and on boards, while the **Summary** is a more detailed description.
2. **Adding Issues to an Epic:**
   * Once the epic is created, you can add related issues (tasks, stories, bugs) to it.
   * **Example:** Add tasks like “Record Episode 1,” “Edit Episode 1,” and “Publish Episode 1” to the “Q1 Episodes” epic.
3. **Viewing Epics on the Scrum Board:**
   * In the **Backlog** view of your Scrum board, you will see the epic panel on the left side.
   * This panel shows all the epics and the issues associated with them.
   * **Example:** You can see “Q1 Episodes” with all its related tasks listed under it.
4. **Tracking Progress:**
   * As you work on the issues, you can track the progress of the epic.
   * **Example:** If 3 out of 9 tasks are in progress and 1 is done, you can see the overall progress of the epic.
5. **Using Epics in Kanban Boards:**
   * Epics can also be used in Kanban boards, but they are not shown by default.
   * To enable the epic panel in a Kanban board, go to **Board Settings** > **Columns** and turn on the **Epics Panel**.
   * **Example:** Once enabled, you can see the “Q1 Episodes” epic and its tasks on the Kanban board.

**Using Roadmaps in Jira**

**What is a Roadmap?**

A **Roadmap** in Jira is a visual tool that helps you plan and track the progress of your epics over time. It provides a high-level view of your project, showing how different pieces of work fit together and when they are expected to be completed.

**Steps to Use Epics in Roadmaps:**

1. **Enable the Roadmap:**
   * Go to your board settings.
   * Find the **Roadmap** option and enable it.
   * **Note:** Roadmaps can only be used if your board is displaying issues from a single project.
2. **Creating an Epic:**
   * Click on the **Create** button.
   * Select **Epic** as the issue type.
   * **Example:** Name your epic “Q1 Episodes” if you’re planning to release multiple podcast episodes in the first quarter.
3. **Viewing the Roadmap:**
   * Once the Roadmap is enabled, you can see a visual timeline of your epics.
   * **Example:** You will see “Q1 Episodes” on the Roadmap with a timeline showing its duration.
4. **Adding Issues to an Epic:**
   * Add related tasks (issues) to your epic.
   * **Example:** Add tasks like “Record Episode 1,” “Edit Episode 1,” and “Publish Episode 1” to the “Q1 Episodes” epic.
5. **Filtering and Searching:**
   * Use the search bar to find specific epics or issues.
   * Filter by assignee, status, or specific epics to narrow down the view.
   * **Example:** Filter to see only tasks assigned to “Gretchen” or only tasks that are “In Progress.”
6. **Tracking Progress:**
   * The Roadmap shows the progress of each epic.
   * **Example:** If 1 out of 9 tasks in “Q1 Episodes” is done and 3 are in progress, you can see this progress visually on the Roadmap.
7. **Adjusting Timelines:**
   * You can adjust the timeline of an epic by dragging its bar on the Roadmap.
   * **Example:** If you expect “Q1 Episodes” to take three months, you can extend its timeline accordingly.
8. **Customizing the View:**
   * Use the view settings to show or hide details like progress and releases.
   * **Example:** Hide the progress bar if you want a cleaner view or show it to see detailed progress.

**Dashboard in JIRA**

**What is a Dashboard?**

A **dashboard** in Jira is a customizable page where you can display various reports and data visualizations, called gadgets. It helps you quickly see important information about your projects at a glance.

**Steps to Create a Dashboard:**

1. **Create a Dashboard:** Go to the **Dashboards** dropdown menu and select **Create dashboard**.
   * **Example:** Name your dashboard “Podcast Team’s Dashboard”.
2. **Set Permissions:** Decide who can view and edit the dashboard.
   * **Options:**
     + **Private:** Only you can see and edit it.
     + **Project:** Anyone with access to the project can see it.
     + **Public:** Anyone on the internet can see it.
   * **Example:** Set it to “Project” so anyone with access to the project can view it.
3. **Choose a Layout:** Select a layout for your dashboard.
   * **Example:** Choose a layout with a larger left column and a smaller right column.
4. **Add Gadgets:** Gadgets are the building blocks of your dashboard. They display different types of data and reports.
   * **Example:** Add the “Assigned to Me” gadget to show tasks assigned to you.

**Recap:**

Creating a dashboard in Jira involves setting up a new dashboard, choosing a layout, adding gadgets, and configuring them to display the data you need. You can also create and apply filters to customize the information shown in your gadgets. Dashboards help you centralize and visualize project data, making it easier to track progress and collaborate with your team.

**Report in JIRA**

**Types of Reports:**

1. **Agile Reports (Scrum Board):**
   * **Burndown Chart:** Shows the amount of work remaining in the sprint over time.
     + **Example:** Helps you see if your team is on track to complete all tasks by the end of the sprint.
   * **Sprint Report:** Provides details about the completed and incomplete tasks in a sprint.
   * **Velocity Chart:** Shows the amount of work completed in each sprint, helping to predict future sprint performance.
2. **DevOps Reports:**
   * **Cycle Time Report:** Measures the time taken to complete tasks from start to finish.
   * **Deployment Frequency:** Tracks how often code is deployed to production.
3. **Issue Analysis Reports:**
   * **Average Age Report:** Shows the average age of unresolved issues.
     + **Example:** Helps identify issues that have been open for a long time and may need attention.
   * **Created vs. Resolved Issues:** Compares the number of issues created versus resolved over time.
4. **Forecast and Management Reports:**
   * **Time Tracking Report:** Compares the original estimated time versus the actual time spent on tasks.
   * **Workload Report:** Shows the distribution of tasks among team members, helping to balance the workload.

**Jira for Administrators (Admin)**

**User Access in Jira**

**User Management in Jira:**

1. **Understanding Active Users:**
   * In Jira, you can have as many users as you want, but you only pay for active users.
   * **Active Users:** Users who have access to Jira and count towards your subscription cost.
   * **Inactive Users:** Users who are suspended and do not count towards your subscription cost but still exist in the system.
2. **Accessing User Management:**
   * User management is handled in the **Atlassian administration** portal, not directly in Jira.
   * To access it, click on your profile icon in the top right corner of Jira and select **User management**.
3. **Inviting a New User:**
   * Click on **Invite users**.
   * Enter the email address of the user you want to invite.
   * **Example:** Invite Eleanor by entering her email and selecting Jira access.
   * You can add the user to a group or personalize the invitation email if needed.
   * Click **Invite**.
4. **User’s Perspective:**
   * The invited user receives an email with a link to join Jira.
   * **Example:** Eleanor receives an email, clicks the link, and logs in to Jira.
5. **Managing Active Users:**
   * You can see the total number of users and the number of active users in the Atlassian administration portal.
   * **Example:** If you have 24 users but only 8 are active, you only pay for those 8 active users.
6. **Suspending and Removing Users:**
   * To suspend a user, find the user in the list and click **Suspend access**. This makes them inactive.
   * To completely remove a user, click **Remove user**. This deletes their account from the system.
   * **Example:** Suspend Mary to make her inactive or remove her completely if she no longer needs access.
7. **Using Groups:**
   * Groups help manage users more efficiently by assigning permissions and roles to multiple users at once.
   * To add a user to a group, go to the **Groups** section and add the user to the desired group.
   * **Example:** Add Eleanor to the “Team Leaders” group.
8. **Product Access and Permissions:**
   * You can control which products (like Jira or Confluence) users and groups have access to.
   * **Example:** Set the “Team Leaders” group to have admin access in Confluence and user access in Jira.
9. **Handling Access Requests:**
   * Users can request access to your Jira instance. You can approve or reject these requests.
   * Go to **User access settings** to manage these requests.
   * **Example:** Approve or reject access requests from users who want to join your Jira instance.

**Company Managed Project**

**Steps to Create a Company-Managed Project:**

1. **Administrative Permissions:** You need to have administrative permissions to create a company-managed project. Not everyone can create or customize these projects.
2. **Access the Project Creation Page:** Go to the **Projects** dropdown menu. Click on **Create Project**.
3. **Choose a Project Template:** You will see various templates based on different needs.
   * **Example Templates:**
     + **Software Development:** Choose between Scrum or Kanban workflows.
     + **Service Management:** Requires Jira Service Management license.
     + **Business Projects:** Templates for project management, design, etc.
4. **Select a Template:** **Example:** Let’s create a software project using the Scrum template. Click on the **Scrum** template under Software Development.
5. **Configure the Project:**
   * **Project Name:** Enter a name for your project.
     + **Example:** “Mobile App Team”
   * **Project Key:** This is a unique identifier for your project.
     + **Example:** “MOBILE”
   * **Project Type:** Ensure it is set to **Company-managed**.
   * Click **Create**.
6. **Project Setup:**
   * Once created, you will be taken to your new project.
   * You can customize the project settings, workflows, and more.

**Customization of Company based Project**

**Accessing Project Settings:**

1. **Navigate to Project Settings:**
   * Go to your project.
   * Click on **Project settings** in the left-hand menu.
   * Alternatively, go to **Projects** > **View All Projects** and click on the three dots next to your project, then select **Project settings**.

**Customization Options:**

1. **Details:**
   * **Project Name:** You can change the name of your project.
   * **Project Key:** This is the unique identifier for your project, used in issue IDs (e.g., “PROJ-1”).
   * **URL:** A reference link for your project (e.g., a link to a related website).
   * **Avatar:** Change the project icon for easy identification.
   * **Description:** Add a brief description of your project.
   * **Project Lead:** Assign a person as the main point of contact for the project.
   * **Default Assignee:** Decide who new issues are assigned to by default (e.g., the project lead or leave unassigned).
2. **Summary:**
   * Provides an overview of the project’s configurations, including issue types, workflows, and screens.
3. **Issue Types:**
   * **Issue Type Scheme:** Controls what types of issues (e.g., Task, Bug, Story) are available in the project.
   * **Example:** You can add custom issue types like “Feature Request.”
4. **Workflows:**
   * **Workflow Scheme:** Defines the statuses and transitions for issues (e.g., To Do, In Progress, Done).
   * **Example:** Customize workflows to add statuses like “In Review.”
5. **Screens:**
   * **Screen Scheme:** Controls which fields are displayed when creating, viewing, or editing an issue.
   * **Example:** Customize screens to include fields like “Due Date” or “Priority.”
6. **Fields:**
   * **Field Configuration:** Manage which fields are available and whether they are required.
   * **Example:** Make the “Due Date” field mandatory for all issues.
7. **Versions:**
   * Manage different versions or releases of your project.
   * **Example:** Create versions like “1.0,” “2.0” to track progress and releases.
8. **Components:**
   * Organize issues into different components or sections of the project.
   * **Example:** Create components like “Frontend,” “Backend,” “Database.”
9. **People and Roles:**
   * **People:** Add or remove users and assign roles (e.g., Administrator, Developer).
   * **Roles:** Define what each role can do within the project.
   * **Example:** Add Christian as an Administrator to give him project-specific admin access.
10. **Permissions:**
    * Control who can do what within the project.
    * **Example:** Set permissions for who can create issues, manage sprints, or view the project.
11. **Notifications:**
    * Configure who gets notified about changes in the project.
    * **Example:** Notify the assignee, reporter, and watchers when an issue is created.

**Jira’s System Administration Overview**

**Accessing Global Administration:**

1. **Navigate to Settings:**
   * Click on the **Settings** icon (gear) in the top right corner of Jira.
   * The options you see here depend on your permissions. As a global Jira administrator, you have access to all settings.

**Key Sections in Global Administration:**

1. **Atlassian Admin Area:**
   * **User Management:** Manage users and groups, invite new users, and assign roles.
   * **Billing:** Update billing information and manage subscriptions.
2. **Jira Settings:**
   * **System Settings:** Control general settings for Jira, such as custom dashboards and the look and feel of Jira.
     + **Example:** Customize the logo and favicon for your organization.
   * **Import and Export:** Import and export Jira data. Be aware of compatibility issues between different Jira versions (e.g., Cloud vs. Server).
3. **Permission and Notification Helpers:**
   * **Permission Helper:** Check why a user does or does not have certain permissions.
     + **Example:** If Gretchen cannot delete an issue, use the Permission Helper to see why.
   * **Notification Helper:** Check why a user is or is not receiving notifications.
     + **Example:** If Gretchen is not getting notifications for an issue update, use the Notification Helper to find out why.
4. **Product Settings:**
   * **Jira Software Settings:** Configure settings specific to Jira Software.
     + **Example:** Enable multiple sprints for company-managed projects if needed.
5. **Project Settings:**
   * **Manage Projects:** View and manage all projects in Jira.
     + **Example:** Move a project to the trash or restore a deleted project.
   * **Archive Projects:** Archive projects to hide them from search results without deleting them (requires a premium or enterprise license).
6. **Apps:**
   * **Manage Apps:** Add and manage apps to extend Jira’s functionality.
     + **Example:** Install apps from the Atlassian Marketplace to customize Jira.
7. **Issues:**
   * **Issue Types:** Create and manage different types of issues (e.g., Task, Bug, Story).
   * **Workflows:** Define the statuses and transitions for issues.
   * **Custom Fields:** Create and manage custom fields to capture additional information.
   * **Screens:** Control which fields are displayed when creating, viewing, or editing an issue.
   * **Field Configurations:** Manage which fields are available and whether they are required.

**Schemes in Jira**

**What are Schemes in Jira?**

Schemes in Jira are configurations that link projects to various settings like issue types, workflows, permissions, and more. They allow you to apply the same configuration across multiple projects, making management easier and more consistent.

**Key Types of Schemes:**

1. **Workflow Scheme:** Defines the statuses and transitions for issues in a project.
   * **Example:** A workflow with statuses like “To Do,” “In Progress,” and “Done.”
2. **Issue Type Scheme:** Controls which issue types (e.g., Task, Bug, Story) are available in a project.
   * **Example:** A scheme that includes issue types like “Story,” “Epic,” and “Bug.”
3. **Permission Scheme:** Manages who can do what within a project.
   * **Example:** A scheme that allows certain users to manage sprints or delete issues.
4. **Field Configuration Scheme:** Determines which fields are available and whether they are required.
   * **Example:** A scheme that makes the “Due Date” field mandatory.
5. **Notification Scheme:** Configures who gets notified about changes in the project.
   * **Example:** A scheme that sends notifications to the assignee and reporter when an issue is updated.

**Automation in Jira**

**What is Automation in Jira?**

Automation in Jira allows you to automate repetitive tasks and processes, making your workflow more efficient. It can perform actions based on specific triggers and conditions.

**Steps to Create an Automation Rule:**

1. **Access Automation:** You can access automation directly from your Agile board by clicking on the lightning bolt icon.
2. **Create a New Automation Rule:** Click on **Create rule**.
3. **Set a Trigger:** A trigger is an event that starts the automation. **Example:** Choose the trigger **Issue assigned**.
4. **Add a Condition:** A condition checks if certain criteria are met before performing an action. **Example:** Add a condition to check if the assignee is “Christian.”
5. **Add an Action:** An action is what happens when the trigger and condition are met. **Example:** Set the action to **Edit issue** and change the priority to “High.”
6. **Name and Save the Rule:** Give your rule a name for easy identification.
   * **Example:** Name it “Set to High.”
   * Turn on the rule.

**Jira Course 2**

**Jira Deployment Types**

There are three main types of Jira deployments:

* **Cloud**: Everything is hosted and managed by Atlassian. It's easy to set up—you just sign up and start using it without worrying about maintenance. But, it comes with a subscription fee.
* **Server**: You host Jira on your own servers. This gives you more control but means you have to handle all the maintenance. Note: This option is being discontinued by 2024
* **Data Center**: Similar to Server, but designed for large enterprises. It offers high availability and performance.

**Why is Server going away?**

Atlassian is ending support for Server by 2024. So if you're using it, you need to switch to either Cloud or Data Center.

**Cloud Versions**

There are different versions of Jira Cloud:

* **Free**: For up to 10 users. Limited features, support, and storage.
* **Standard**: What we’ll use in this course. More features and better support for a reasonable cost.
* **Premium**: Offers almost unlimited automation, more storage, advanced project planning tools, and better uptime guarantees. It’s pricier.
* **Enterprise**: Includes everything in Premium plus more—like unlimited instances and 24/7 dedicated support. Meant for very large companies.

**Why Cloud?**

Atlassian is pushing for Cloud because it's easier to manage and scale. They handle all the updates, security, and performance issues for you.

**Agile Methodology and Scrum in Jira**

**Agile Methodology**

* **Agile**: Think of it as a way of developing software in small, manageable parts rather than doing everything in one go. Each part is built, tested, and delivered step-by-step.
* **Iteration**: Each step is called an iteration, where the team plans, develops, and delivers a piece of the project. Feedback is collected and used to improve the next iteration.
* **Flexibility**: This method is flexible and allows changes to be made easily based on early feedback, ensuring the final product meets customer needs.

**Key Agile Concepts**

* **Iterative Development**: Continuous planning, development, and feedback loops until the product is polished.
* **Customer Satisfaction**: Frequent delivery of software ensures customers see progress and can provide feedback early.
* **Collaboration**: Close teamwork, especially between business people and developers, is crucial.

**Scrum Methodology:** Scrum is a popular type of Agile that breaks down projects into manageable tasks.

**Key Roles in Scrum:**

* **Product Owner**: Defines what needs to be built and why. Manages the product backlog (a  list of all desired features).
* **Development Team**: Builds the product.
* **Scrum Master**: Ensures the Scrum process is followed and helps remove any obstacles.

**Main Components of Scrum:**

1. **Epics and User Stories**:
   * **Epic**: A large task that’s broken down into smaller tasks.
   * **User Story**: The smallest unit of work, representing a feature to be developed.
2. **Product Backlog**: A prioritized list of all the features needed in the product. The most important features are at the top.
3. **Sprint**:
   * **Sprint**: A fixed time period (usually 2 weeks) where a set of tasks (User Stories) from the backlog is completed.
   * **Sprint Planning**: Meeting to decide which tasks will be worked on during the Sprint.
   * **Daily Scrum**: A 15-minute meeting each day to discuss progress and any blockers.
   * **Sprint Review and Retrospective**: At the end of the Sprint, the team reviews what’s  completed and discusses ways to improve.
4. **Sprint Board (Scrum Board)**:
   * **Columns**: Tasks move through columns like To Do, In Progress, and Done. The goal is to  move all tasks to the Done column by the end of the Sprint.

**Kanban Methodology**

**Kanban Methodology**

* **Kanban**: This is a visual way to manage tasks in a team. Think of it like a board that shows all the work that needs to be done and helps track progress.
* **No Sprints**: Unlike Scrum, Kanban doesn’t work in fixed time periods (sprints). Instead, it’s about continuous delivery.

**Key Concepts of Kanban**

1. **Visualizing Work**:**Kanban Board**: A board with columns representing different stages of work (like To Do, In Progress, Done). The goal is to move tasks from left to right until they’re completed.
2. **Work In Progress (WIP) Limits**:**WIP Limits**: This is a limit on the number of tasks that can be in any column at one time. It helps prevent bottlenecks by ensuring the team isn't overloaded.
3. **Smooth Flow**:The main goal is to have a smooth flow of work without any stoppages. The team defines the steps a task goes through from start to finish and ensures tasks move efficiently through these stages.
4. **Monitoring and Improving**:**Track Progress**: The team keeps an eye on the board to see how tasks are moving. If there are bottlenecks (e.g., too many tasks in one stage), they work together to resolve them and improve the process.

**Key Jira Terms**

**Key Jira Terms**

1. **Issue**:
   * **Definition**: An issue is a work item that needs action. Think of it as a task or ticket.
   * **Examples**: Epics, Stories, Bugs, Feature Requests, and Tasks are all types of issues.
   * **Details**: Each issue has fields like title, description, due dates, priorities, and status.
2. **Project**:
   * **Definition**: A project is a collection of issues.
   * **Use**: Projects can be anything from software development projects to help desk systems.
   * **Customization**: You can customize who has access, the issue types, and the workflows in a  project.
3. **Workflow**:
   * **Definition**: A workflow is the set of statuses an issue can go through, and the transitions between those statuses.
   * **Example**: A simple workflow might have statuses like To Do, In Progress, and Done.
4. **Version**:
   * **Definition**: Versions help track software releases.
   * **Use**: You can create versions at the end of each sprint or after several sprints to track the group of features being released.
5. **Component**:
   * **Definition**: Components are subsections of a project used to group issues.
   * **Example**: You could have components like Backend Development and Frontend Development to categorize work.
   * **Advantage**: You can set default assignees for each component to streamline task delegation.

**JQL Searching**

**Searching in Jira**

1. **Basic Search**:
   * Use simple filtering options provided in Jira to find issues.
   * Example: Filter by project, issue type, status, assignee, etc.
2. **Advanced Search with JQL (Jira Query Language)**:
   * **JQL**: A powerful way to search using a query language similar to SQL.
   * You can switch to JQL from the basic search to see your filters as a query.

**Example of Using JQL**

**Basic Search**

* You start by filtering issues. For example, if you want to see only stories in a project that are either "In Progress" or "To Do", you set these filters.
* Jira shows you the issues that match these filters.

**Switching to JQL**

* When you switch to JQL, you see a query like:

issuetype = Story AND project = SSP AND status in ("In Progress", "To Do")

**Swimlanes in JIRA**

**Swimlanes** in JIRA are horizontal rows that divide a board into sections, allowing you to group issues based on specific criteria. **Imagine a board** with many tasks listed on it. You can use swimlanes to **divide** these tasks into different **sections** based on who is doing them, what kind of work they are, or how important they are. **Here are some common ways to use swimlanes:**

* **By assignee:** Group issues based on the individuals responsible for completing them.
* **By project:** Organize issues from different projects within a single board.
* **By label:** Categorize issues based on shared characteristics or tags.
* **By component:** Group issues related to specific parts of a project.
* **By priority:** Prioritize issues within swimlanes based on their urgency.

**For example,** you could have a swimlane for each team member, so you can see who is working on what. Or, you could have a swimlane for "High Priority" tasks and another for "Low Priority" tasks.

**Software Version in Jira**

A software version in Jira is like a milestone or checkpoint that helps track and manage different stages of your project’s progress. Think of it as a label for a particular phase of your project where you plan to release or deliver a set of features or bug fixes.

**Creating Versions**

1. **Where to Create Versions**:
   * You can create versions in the **Backlog** view by expanding the Versions panel.
   * Or, you can go to the **Releases** section of the project to manage all your versions.
2. **Creating a Version**:
   * **Name**: Give your version a name. Example: "Team Lead Section".
   * **Dates**: Specify a start date and a target release date.
   * Click **Create** to add the version.

**Adding Issues to Versions**

1. **In Backlog View**:
   * Drag and drop issues (tasks, stories, etc.) into the version.
   * Example: Drag issues for the “Leading and Agile Team” epic into the version named "Team Lead Section".
2. **Viewing Version Details**:
   * Go to the **Releases** section and click on the version name.
   * Here, you’ll see all issues in the version, their statuses, and progress.

**Releasing a Version**

1. **Release When Ready**:
   * Once all issues in the version are complete, click **Release**.
   * If there are unresolved issues, Jira will ask if you want to:
     + **Ignore** them and proceed with the release.
     + **Move** them to a future release.

**Using Release Notes**

1. **What Are Release Notes?**:
   * Release notes are a summary of all issues included in the release.
   * Useful for sharing progress with stakeholders.
2. **Configuring Release Notes**:
   * In the version’s details, you can configure the format of release notes.
   * Example: Format it for copying into a document.

**Sprint Report**

**What is a Sprint Report?**

A sprint report provides a summary of what happened during a sprint. It shows completed tasks, unfinished tasks, and other useful information like burndown charts to help the team understand their performance.

**Key Components of a Sprint Report**

1. **Completed Issues**: Tasks or stories that were finished during the sprint.
2. **Incomplete Issues**:Tasks that were not completed and need to be moved to the next sprint or back to the backlog.
3. **Burndown Chart**:A visual representation of the remaining work in the sprint. It helps track if the team is on track to complete the sprint on time.
4. **Issues Added/Removed**: Any new tasks that were added to or removed from the sprint after it started.
5. **Velocity**:The total amount of work (usually in story points) completed in the sprint. It helps estimate how much work can be done in future sprints.

**JIRA Admin**

**Inviting Users**

**Inviting Users:**

* The admin can invite users by entering their email addresses.
* You can create unique email addresses for testing by adding "+something" before the "@" symbol in your email address (e.g., test@email.com + user1 becomes test@email.com+user1).

**Adding Users to Groups:** JIRA automatically adds users to a group that gives access to the products they're granted access to (e.g., "jira-software-users" group for Jira software).

**Admin Permissions:**

* An organization admin can grant other users admin privileges for Jira products or the entire organization.
* Admins can also manage user access (suspend, remove, reset passwords).
* Interestingly, admins can even log in as other users (but there might be a delay before this works).

**Other Ways Users Can Get Access:**

* Approved domains: Users with email addresses from specific company domains can request access (needs to be set up by the admin).
* User invites: Other users in your JIRA instance can invite people (needs admin approval by default).
* Invitation links: The admin can generate a link that anyone can use to sign up for JIRA (link expires after 30 days).

**Groups in JIRA**

**Groups** in JIRA are like teams or categories that you can use to organize users. By putting users into groups, you can easily manage their permissions and access to different parts of JIRA.

**In JIRA:**

* **Create groups:** You can create different groups based on your needs (e.g., "Developers," "Project Managers," "External Consultants").
* **Add users to groups:** Assign users to the appropriate groups.
* **Manage permissions:** Grant or restrict permissions for groups, rather than for individual users. This saves time and ensures consistency.

**Types of permissions you can manage for groups:**

* **Global permissions:** These apply to the entire JIRA instance. For example, you might give a group permission to create new projects.
* **Project permissions:** These apply to specific projects. You might give a group permission to edit issues in a particular project.
* **Application access:** You can grant groups access to different JIRA applications (e.g., Jira Software, Jira Service Desk).

**Example:**

If you have a group of external contractors, you might create a group called "External Consultants." You can then give this group specific permissions, like the ability to view and create issues in certain projects, but not the ability to edit project settings.

**Global Permissions in JIRA**

**Global permissions** in JIRA are like master keys that control what users can do across the entire JIRA instance. Here's a breakdown of the key global permissions:

* **Administer JIRA:** This is the highest level of permission, allowing users to manage JIRA settings and configuration.
* **Browse users and groups:** This lets users see a list of other users and groups in JIRA.
* **Share dashboards and filters:** This enables users to share their dashboards and filters with others.
* **Manage group filter subscriptions:** Allows users to create and manage subscriptions for filters.
* **Bulk changes:** Permits users to make changes to multiple issues at once.

**Project Roles**

**Project roles** in JIRA are like titles or job positions that define what users can do within a specific project. They're a more efficient way to manage permissions compared to assigning permissions directly to individual users or groups.

**Here's how project roles work:**

1. **Create project roles:** You can create custom project roles like "Developer," "Tester," or "Product Owner."
2. **Assign roles to users:** Assign users or groups to these project roles.
3. **Define permissions for roles:** Determine what actions users with that role can perform in the project (e.g., create issues, edit issues, assign issues).
4. **Assign roles to projects:** Assign the project roles to specific projects.

**Example:** If you have a project with developers, testers, and a project manager, you can create project roles for each. Then, assign users to these roles. The project manager might have permissions to create and edit issues, while developers can create and work on issues, and testers can test issues.

**Roles and Permissions**

**Create Users and Groups**

* Create users for different project roles: Consultant (Jack Parker), Customer (Monica Hall - new user), Project Owner (Adam Smith), Developer (KS Developer).
* Create groups for project roles (optional): Customer, Developer.
* Grant access to Jira software for the Customer group.

**Manage Project Roles**

* Delete any unnecessary project roles (e.g., Issue Creator).
* Set the Jira administrator group as the default member of the Administrators project role (optional).
* Set the Developer group as the default member of the Developers project role (optional).
* Create new project roles: Consultant, Customer, Project Owner.

**Create Permission Scheme**

* Create a new permission scheme instead of modifying the default one.
* Assign permissions to project roles instead of individual users or groups.

**Define Permissions**

* **Administer Projects:** Assign to Administrators project role.
* **Browse Projects:** Assign to Project Team project role (to be created).
* **Manage Sprints:** Assign to Project Owner project role.
* **View Development Tools:** Assign to Project Team project role (to be created) (can be adjusted based on your needs).
* **View Read-only Workflow:** Assign to Project Team project role (to be created).
* **Assignable User:** Assign to Project Team project role (to be created).
* **Assign Issues:** Assign to Project Team project role (to be created).

**Testing:**

* **Logged in as Adam Smith (Project Owner):**
  + Able to access both projects.
  + Can manage sprints, create and edit issues, and view project details.
* **Logged in as KS Developer (Developer):**
  + Able to access both projects.
  + Can create, edit, and work on issues (depends on other permissions).
* **Logged in as Monica Hall (Customer):**
  + Able to access the project where she is assigned the Customer project role.
  + Has read-only access (can view issues but not edit).

**Key Takeaways from the Testing:**

* **Project roles simplify permission management:** The use of project roles made it easy to assign permissions to groups of users.
* **Permission scheme configuration is crucial:** The default permission scheme needed to be modified to restrict access to specific projects and actions.
* **Global permissions impact project access:** The "Browse Projects" global permission controls which projects users can see.

**Additional Observations:**

* **Trusted user privileges:** The trusted user status allowed Adam Smith to access project settings but not necessarily the project itself.
* **Project visibility:** The "Next Gen Project" being private prevented unauthorized access.

**Recommendations:**

* **Review and adjust permissions regularly:** As your project evolves, revisit your permission scheme to ensure it remains aligned with your needs.
* **Consider using more specific project roles:** For larger projects, create additional roles (e.g., "Lead Developer," "Quality Assurance") to further refine permissions.
* **Document your permission structure:** Create documentation outlining the roles and permissions for future reference.

**JIRA Schemes**

**JIRA Schemes** are like blueprints that define how different aspects of your JIRA instance work. They control things like how issues are tracked, how projects are organized, and what permissions users have.

**Here are the main types of schemes in JIRA:**

* **Issue type schemes:** Determine the types of issues that can be created in your instance (e.g., Bug, Task, Story).
* **Workflow schemes:** Define the lifecycle of issues, including the different statuses they can go through and the transitions between them.
* **Permission schemes:** Control what users can do in JIRA, such as create, edit, or delete issues.
* **Field schemes:** Specify the custom fields that are available for issues in your instance.
* **Screen schemes:** Determine the screens users see when creating, editing, or viewing issues.

**How schemes work:**

* **Schemes are associated with projects:** Each project uses a specific set of schemes that define its behavior.
* **Schemes can be copied and modified:** You can create new schemes based on existing ones to customize your JIRA instance.

**Example:**

Imagine you're managing a software development project. You might use a scheme that defines issue types like "Bug," "Task," and "Story." The workflow scheme would outline the stages of development, such as "To Do," "In Progress," "Testing," and "Done." The permission scheme would control who can create, edit, and resolve issues.

**Screen Scheme in JIRA**

* **Screens**: Define which fields are shown for creating, viewing, and editing issues.
* **Screen Scheme**: Maps screens to specific operations (Create, Edit, View).
* **Issue Type Screen Scheme**: Links screen schemes to specific issue types.
* **Example**: Create screens for spike issues, configure them, and test to ensure they work as expected.

**Custom Fields in JIRA**

**Custom fields** in JIRA allow you to create additional data fields for issues, tailoring them to your specific project needs. They can be used to collect information, track progress, or automate workflows.

**Key Steps to Create a Custom Field:**

1. **Choose a field type:** Select the appropriate field type based on the data you want to collect (e.g., text, number, date, dropdown).
2. **Configure the field:** Set the field name, description, and other relevant properties.
3. **Assign the field to screens:** Determine which screens the field should appear on (create, edit, view).
4. **Define field behavior:** Specify whether the field is required, has default values, or has specific validation rules.
5. **Map the field configuration to the issue type:** Associate the field configuration with the appropriate issue type.

**Example: Creating a "Priority" Field**

1. Create a new Select List (Single Choice) field named "Priority."
2. Define the options: High, Medium, Low.
3. Assign the field to the "Create Issue" and "Edit Issue" screens for relevant issue types.
4. Make the field required for those issue types.

**Workflow in JIRA**

**JIRA Workflows** define the lifecycle of issues, including the statuses they can go through and the transitions between them. They are essential for managing project tasks and ensuring smooth progress.

**Key Components of Workflows:**

* **Statuses:** Represent different stages of an issue's lifecycle (e.g., To Do, In Progress, Done).
* **Transitions:** Define the possible movements between statuses (e.g., Start Work, Complete).
* **Conditions:** Specify requirements for a transition to occur (e.g., only allow the transition if the issue is assigned).
* **Post-functions:** Trigger actions after a transition (e.g., send a notification, update a field).

**Example: Creating a "Review" Workflow**

* **Statuses:** To Do, In Review, Done
* **Transitions:** Start Work (To Do -> In Review), Complete (In Review -> Done)
* **Conditions:** Require the issue to be assigned before starting work.
* **Post-function:** Send a notification to the assignee when the issue is completed.

**Modifying Workflows**

* **Draft workflows:** JIRA creates a draft when you edit an active workflow to prevent disrupting ongoing work.
* **Limited changes to live workflows:** Some changes (e.g., renaming statuses) can only be made to inactive workflows.
* **Workflow schemes:** Use workflow schemes to assign workflows to different issue types.

**Updating a Workflow:**

1. **Create a copy:** Make a copy of the existing workflow to avoid affecting live issues.
2. **Modify the workflow:** Add, remove, or edit statuses and transitions.
3. **Update the workflow scheme:** Assign the modified workflow to the appropriate issue types.
4. **Publish the workflow:** Make the changes active.

**JIRA Workflows and Agile Boards**

**Synchronizing Workflows and Agile Boards**

* **Map statuses to columns:** Ensure that statuses in your workflow are mapped to columns on your agile board.
* **Update board settings:** When you modify workflows, update the board settings to reflect the new statuses.
* **Handle status changes:** If an issue's status changes, it may need to be moved to a different column on the board.

**Additional Considerations:**

* **Board configuration:** Customize your board settings to suit your team's needs (e.g., swimlanes, filters).
* **Workflow transitions:** Ensure that transitions between statuses align with your project's process.
* **Issue types:** Consider using different workflows for different issue types if necessary.

**Understanding Workflow transitions - Conditions, Validators, Post-Functions**

**Conditions:**

* **Restrict transitions:** Prevent users from moving issues to certain statuses unless specific conditions are met.
* **Examples:**
  + Require the issue to be assigned to a specific user or group.
  + Check if a custom field has a particular value.
  + Verify that a linked issue is in a specific status.

**Validators:**

* **Validate user input:** Ensure that data entered into custom fields meets specific criteria.
* **Examples:**
  + Validate email addresses.
  + Check for required fields.
  + Enforce numerical ranges.

**Post-functions:**

* **Automate actions after transitions:** Trigger actions when an issue moves to a new status.
* **Examples:**
  + Send notifications.
  + Update custom fields.
  + Create sub-tasks.
  + Link issues.

**Configuring Conditions, Validators, and Post-functions:**

1. **Select the transition:** Click on the transition in the workflow designer.
2. **Access properties:** Open the properties panel for the transition.
3. **Add conditions, validators, or post-functions:** Use the available options to configure the desired behavior.
4. **Save and publish:** Save your changes and publish the workflow.

**Example: Requiring a Review Before Completion**

* **Condition:** Add a condition to the "Complete" transition that requires the issue to be in the "Reviewed" status.
* **Post-function:** Add a post-function to send a notification to the project lead when the issue is completed.

**Project Settings**

**Project Settings** in JIRA allow you to configure various aspects of your project, including access restrictions, roles, and categories.

**Key Project Settings:**

* **Project details:** Name, key, description, and icon.
* **Project access:** Open, limited, or private.
* **Project roles:** Define custom roles with specific permissions.
* **Project categories:** Organize projects into categories for better management.

**Understanding Project Access:**

* **Open:** Anyone with access to your JIRA instance can view, create, and edit issues.
* **Limited:** Anyone can view issues, but only members can create and edit them.
* **Private:** Only members and administrators can access the project.

**Managing Project Roles:**

* **Create custom roles:** Define specific permissions for different user groups.
* **Assign roles to users:** Grant users appropriate permissions based on their roles.
* **Customize permissions:** Tailor permissions to meet your project's specific needs.

**Project Settings: Issue Type, Notifications**

**Issue Types:**Issue types in Jira are the different kinds of work items you can create and manage in your projects. Examples include tasks, bugs, stories, and custom types like investigation

**Example Scenario**

1. **Default Issue Types**:When you create a project using the Kanban template, it comes with default issue types like Epic, Task, and Sub-task.
2. **Adding Default Issue Types**:You can add builtin issue types like Bug and Story. Simply click on **Add Issue Type** and select from the list.
3. **Creating Custom Issue Types**:
   * You can create your own custom issue type. For example, create an issue type called “Investigation” for any investigative work needed.
   * Give it an appropriate icon and click **Create**.

**Notifications**

Notifications in Jira keep team members informed about changes or updates to issues. You can customize these notifications for different events.

**Example Scenario**

1. **Default Notifications**:
   * Jira has a fixed list of events for which you can send notifications, like issue creation, assignment, and resolution.
2. **Adding Notifications**:
   * You can add notifications for specific events. For example, add a notification for when a comment is deleted.
   * Select the event and specify the recipients, such as watchers, assignees, reporters, or project roles.

**Team Managed Project Features**

**Key Features**

1. **Roadmap**:
   * **Purpose**: Helps visualize the project timeline and plan.
   * **Enabling/Disabling**: You can turn it on/off in the project settings.
   * **Example**: When disabled, the roadmap option disappears from your project view.
2. **Backlog**:
   * **Purpose**: Organize and prioritize tasks before moving them to the board.
   * **Enabling**: Activates a backlog view where you can manage pending work.
   * **Example**: Move tasks from the first column of the board into the backlog for better management. The board becomes less cluttered.
3. **Sprints**:
   * **Purpose**: Manage work in time-boxed iterations.
   * **Enabling**: Converts a Kanban project to a Scrum project.
   * **Example**: Once enabled, the board will only show tasks that are part of an active sprint.
4. **Reports**:
   * **Purpose**: Analyze progress and performance.
   * **Enabling**: Adds a reports section to the project.
   * **Example**: Provides access to sprint reports, velocity charts, and other metrics once a sprint is completed.
5. **Story Points**:
   * **Purpose**: Estimate effort for tasks using story points.
   * **Enabling**: Adds a story point field to tasks.
   * **Example**: Estimate the effort for tasks directly within the issue details.
6. **Pages (Confluence)**:
   * **Purpose**: Link and manage Confluence pages within the project.
   * **Enabling**: Connects Confluence pages to your project for better documentation.
   * **Example**: Collaborate and document work using Confluence pages linked to your Jira project.
7. **Releases and Versions**:
   * **Purpose**: Track software releases.
   * **Enabling**: Adds a releases section where you can create and manage versions.
   * **Example**: Assign tasks to specific versions for better release management.
8. **Issue Navigator**:
   * **Purpose**: Search and filter issues.
   * **Enabling**: Adds an issue navigator to the project.
   * **Example**: Search for specific issues using predefined filters or advanced search options.

**Workflow Editor**

**What is the Workflow Editor?**

The Workflow Editor allows you to define and manage the workflow for issues in your Jira project. A workflow is essentially the steps (statuses) that an issue goes through from start to completion.

**Key Features of the Workflow Editor**

1. **Managing Rules**:
   * You can set rules for how issues move between statuses.
   * Rules help ensure that issues follow specific paths, like needing approval before moving to "Done".
2. **Editing Workflows**:
   * You can now edit workflows directly within Next-Gen projects.
   * This includes adding new statuses and transitions between those statuses.
3. **Applying Changes**:
   * Any changes you make to the workflow apply to all issue types in the project for now.
   * Atlassian plans to allow different workflows for each issue type in the future.