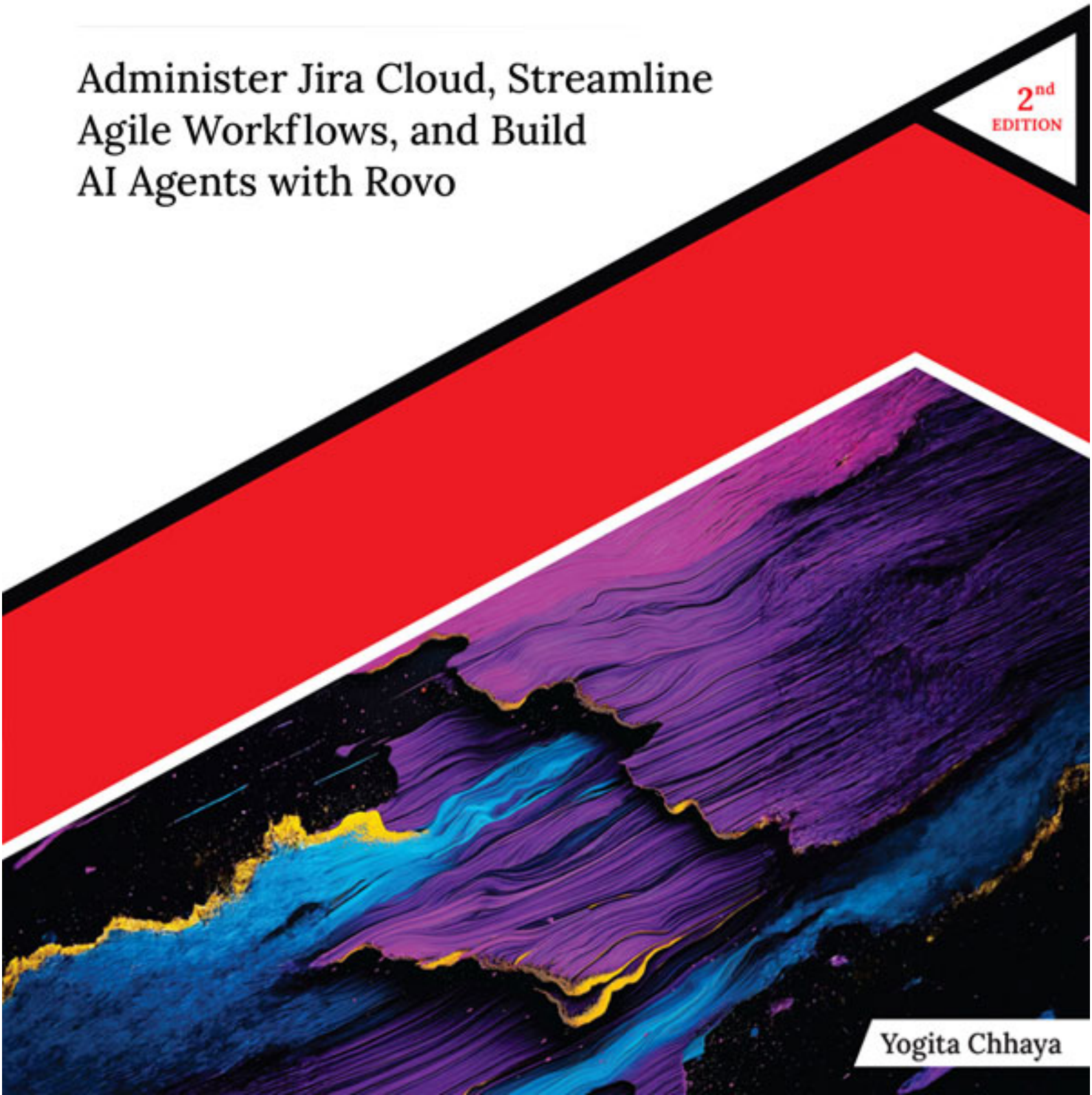
ULTIMATE

Agile Administration with Jira

Administer Jira Cloud, Streamline Agile Workflows, and Build AI Agents with Rovo

2nd EDITION



Yogita Chhaya

First Published: April 2026

Published by: Orange Education Pvt Ltd, AVA®

Address: 9, Daryaganj, Delhi, 110002, India

275 New North Road Islington Suite 1314 London,
N1 7AA, United Kingdom

ISBN (PBK): 978-93-49887-95-4

ISBN (E-BOOK): 978-93-49887-02-2

Scan the QR code to explore our entire catalogue

www.orangeava.com

Table of Contents

CHAPTER 1

Getting Started with Agile, Jira, and Jira Terminologies

Introduction

This chapter will cover the basics of Agile methodology, including the history of Agile, and how it differs from the waterfall approach. We will then understand Scrum and Kanban, which are two popular methodologies of software development. This will help build a strong foundation for understanding modern software practices.

We will explore Jira, different Jira products, and reasons why Jira is being used by different industries. At the end of the chapter, we will cover Jira terminology commonly used in the field. This will give you a practical understanding of how Jira is applied in real-world scenarios.

Structure

In this chapter, we will cover the following topics:

- History of Agile
- Difference between Waterfall and Agile
- Agile Mindset
- 4 Values and 12 Principles of Agile
- Various Agile Frameworks
- Defining Scrum
- Creating User Stories and Product Backlogs
- Defining Kanban
- Estimating and Prioritizing Work in Agile
- Agile Metrics and Performance Tracking
- About Jira
- Step-by-Step Guide to Create a Jira Site

- The Popularity of Jira
- Terminologies

History of Agile

Back in the 1990s, products were not solely dominated by software applications. They were a combination of hardware and software products. It is important to note that software was becoming increasingly complex and important in its own right. This is one of the factors that led to the need for a new way of developing software. It used to take more than a few years to complete the development cycle and launch a product. By the time it was handed over to the customer, their requirements had changed due to the long lead time and the products had already entered the market. In addition to that, some of the features were of no use to the customers. In software development, there were many processes to be followed, and a lot of documentation was required to be done as a part of the process.

As years passed, software projects were becoming more important, and it was felt that a new process or new way of developing software was required by everyone. The same thing was experienced in other industries such as automotive, aerospace, healthcare, and many others.

Many industries, beyond software development, faced challenges with long project cycles. Issues such as change in customer requirements, late feedback, increasing cost, time spent on documentation, and most importantly, how to deliver continuously concerned the project teams. The need for a more responsive and iterative approach became evident as businesses sought to stay competitive in rapidly evolving markets.

The introduction of the Toyota Production System (TPS) by Toyota, which revolutionized manufacturing, was the beginning of the foundations of Agile. It was founded between 1945 and 1975 by Japanese industrial engineers. TPS prioritized flexibility, waste minimization, and ongoing improvement. To encourage flexibility and effectiveness in software development, agile techniques drew inspiration from TPS and adopted its ideas. While Agile drew inspiration from TPS, it was not directly founded during that period; instead, it was formalized in 2001 with the Agile Manifesto.

Evolution toward the Agile Manifesto

The Agile Manifesto was not created suddenly. It was developed over time as people looked for better ways to build software. Traditional methods were rigid

and slow, so developers began looking for more flexible and collaborative approaches. These ideas eventually came together in 2001 as the Agile Manifesto.

- **1980s:** In the early 1980s, the seeds of Agile methodology were sown with the emergence of lightweight software development approaches. The Waterfall model, a sequential and document-heavy methodology, dominated the scene. However, cracks begin to show as projects face delays, scope changes, and poor communication.
- **1990s - Iterative and Incremental Practices**: As the 1990s dawned, software practitioners started experimenting with iterative and incremental practices.
- **Late 1990s - Crystal and DSDM**: Towards the late 1990s, two notable methodologies, Crystal and Dynamic Systems Development Method (DSDM), emerged.
- **1990s - Extreme Programming (XP)**: In the late 1990s, Kent Beck introduced Extreme Programming (XP).
- **2001- The Agile Manifesto Takes Shape:** In 2001, 17 people from the software industry met at a place, and they agreed upon a way of working which is known as the Agile Manifesto. There were people from the software industry who were following Scrum, XP, FDD, and many different methodologies. They did not like the word **Lightweight** for this new methodology and agreed and decided to name it **Agile,** meaning responding to change.

Implementing an Agile way of working organization-wide is known as Business Agility. It refers to the ability of an organization to adapt quickly to market changes, seize opportunities, and deliver value to customers. It enables businesses to respond quickly to changing customer needs. It is implemented by many organizations worldwide to stay ahead in the currently dynamic business environment.

Difference between Waterfall and Agile

Agile and waterfall project management strategies are two separate approaches to project management. In the waterfall model, the next phase starts only after the earlier phase is completed. In waterfall methodology, requirements gathering, design, development, testing, and deployment phases are accomplished in sequence. It emphasizes extensive planning, documentation, and a fixed scope, making it suitable for projects with well-defined requirements and limited

changes. It is therefore considered to be rigid and not able to manage unplanned changes.

Agile, on the other hand, is a flexible and iterative strategy that emphasizes adaptive planning and collaboration. It divides work into discrete sprints or iterations, allowing for continual feedback and adjustments. Agile values change and encourage client participation throughout the project. It encourages self-organizing teams and frequent communication, and focuses on providing incremental value. Agile teams use documentation to communicate and collaborate, but they do not produce as much documentation as waterfall teams.

Table 1.1 illustrates the difference between Waterfall and Agile Methodology:

Aspects	Waterfall Methodology	Agile Methodology
Project Phases	Distinct phases – requirements, design, development, and more	Continuous cycles - sprints
Development Approach	Sequential and linear	Iterative and incremental
Project roles	Specific roles are assigned for each phase (for example, Business Analyst, Designer, Developer, Tester, Project Manager)	Product owner, Scrum Master, Development teams
Quality Assurance	Testing at the end of the cycle	Continuous testing and quality assurance
Team Structure	Hierarchical with predefined roles	Cross-functional self-organizing teams
Communication	Less customer involvement	Close collaboration with customers
Documentation	Extensive documentation throughout	Documentation as needed
Flexibility	Limited flexibility to changes in requirements	Welcomes changes in requirements throughout the project
Feedback	Limited customer feedback during development	Regular customer involvement and feedback
Delivery time	Longer delivery time	Shorter delivery time
Change Implementation	Difficult to accommodate changes	Welcomes changes, even in later stages of development
Risk Management	Addressed in early planning, may not handle unforeseen risks well	Ongoing risk management
Project outcome	Well-defined scope and deliverables	Evolving scope and adaptable deliverables
Suitability	Projects with stable and known requirements	Projects with dynamic or evolving requirements

Table 1.1: *Waterfall versus Agile Methodology*

Agile Mindset

Being agile is more about beliefs, values, attitudes, and behaviors. For example, in a game of Rugby, the entire team works towards achieving a goal by adapting to the situation, collaborating, and with the "everyone is equal" attitude. It can be applied to any industry and any functional group, such as marketing, human resources, operations, and more. It is a way of working in which one must co-create, explore new ideas, experiment and experience, innovate, and remain adaptable.

4 Values and 12 Principles of Agile

In the 1990s, there was a period when projects were frequently delayed due to lengthy processes and sequential approaches to software development. The released products did not meet customer expectations due to the time lag, resulting in numerous order cancellations. These frustrations led to the creation of the Agile Manifesto by 17 leaders. This manifesto includes 12 principles and 4 values, which we will cover in this section.

4 Agile Values

There are multiple Agile methodologies, each of which applies the four values of the Agile Manifesto in different ways to guide teams in developing good quality software.

1. **Individuals and Interactions over Processes and Tools:** In the waterfall model, more importance was given to rigid processes. Even after spending a lot of time following these processes, the final software products were not of the best quality and error-free. It is possible to design and develop innovative software products if more importance is given to smart and competent people who can sit together, discuss, share, and solve problems. Processes should be followed, but they need to be simplified. Tools have to be customized to follow processes easily and make the development faster.
2. **Working Software over Comprehensive Documentation:** There was a time when very detailed documents were prepared for the features, requirements, test cases, diagrams, and releases. This was not helpful to provide value to the customer, and it used to delay the product releases. Delayed releases and obsolete features were the cause of the loss of business. More significance is given to working software, which is provided

to the customer, and based on the feedback, further development is done. As per Agile Manifesto, providing value first is of more importance than creating outdated documents.

3. **Customer Collaboration over Contract Negotiation:** In the old ways of project development, customers used to come into the picture at the beginning, once during the development cycle, and finally at the end. The products were developed based on what is written in the contract documents. In an agile way of working, customers come into the picture to provide feedback on a small piece of software, and based on the feedback and their actual needs, further development is carried out. This way it is possible to do immediate corrections if it is going in the wrong direction with the help of a customer. Continuous interaction with the customer is helpful to find out the right needs than what is written in the legal documents.
4. **Responding to Change over Following a Plan:** Traditionally, plans were prepared well in advance. However, it was not possible to incorporate any changes in that plan. Even a small change in the requirement was considered impossible. When a plan is made with agility in mind, it provides direction initially. Based on a rough plan, product development is started and then there is a scope for change. A small change in specification is discussed by all the stakeholders and taken up as an opportunity rather than an obstacle.

Agile Principles

The 12 principles act as guiding principles for the Agile methodologies. They outline a culture that embraces change and emphasizes customer-centric product development. These principles emphasize the importance of aligning new products with business needs:

1. **Customer Satisfaction through Early and Continuous Delivery:** By iteratively delivering working software, the customer is satisfied as he is getting some valuable features early in the cycle. At the same time, in every release cycle, he is going to receive something new which makes the customer happy.
2. **Welcome Changing Requirements, Even Late in Development:** In this dynamic world, everything changes continuously. Similarly, by incorporating changes in the requirements during the entire software development life-cycle, people can create the right products. By doing so, the final released software application is aligned with the actual and

evolving requirements of the customer. Agile allows scope for changes even late in the development process.

3. **Deliver Working Software Frequently:** This principle recommends delivering new versions in a shorter time span, as opposed to the earlier longer time span. By doing this, the number of bugs per release is significantly reduced, leading to an overall improvement in the success of the software release. The release cycles are measured in the number of days rather than months.
4. **Collaborate Daily between Business People and Developers:** In the Agile way of working, there are no barriers to communication between business teams and developers. Business teams must communicate about the business requirements, any changes from the side of the customer, and also whether these changes can be managed or not by the developers. By collaborating with developers, any misunderstandings can be avoided. Business analysts have to convert any business language to a language that developers can understand. Even if the teams are geographically far from each other, providing feedback on a daily basis helps in getting good outcomes.
5. **Motivated Individuals:** It is about creating a conducive work culture and keeping the workforce motivated. Every member is allowed, trusted, and encouraged to give new ideas and find better ways of designing new products. If the team members are motivated, they can eventually create innovative architectures, designs, and products.
6. **Face-to-face Conversations:** Communication is the key to success. The same thing is emphasized in Agile. It could be physical meetings in the office or video conference meetings where people can communicate efficiently. Developers and teams must communicate daily to understand the initial and evolving requirements.
7. **Measure of Progress through Working Software:** Customers want a working and good quality product. It is the measurement of the overall performance of the team. The software is considered to be of good quality based on the feedback from the end-user.
8. **Promote Sustainable Development:** The teams should work at the same pace irrespective of the number of changes introduced. Any member or employee should not be burdened by the work more than his/her capacity. Otherwise, it can result in burnout of the teams and hence, can affect the quality of the product.
9. **Continuous Attention to Technical Excellence:** Agile emphasizes on the good design of the product. This is a must-have for each and every team. It

ensures that the software application is adaptable, maintainable, and scalable for future enhancements.

10. **Simplicity:** This principle suggests keeping the processes simple. Simplicity does not mean skipping important processes but it means avoiding complexity in design and coding. A working software product with less number of features is better than a problematic product having multiple features.
11. **Self-organizing Teams:** If the teams are empowered and autonomous, they can get better results. The agile teams consider themselves accountable and are ready to take responsibility for their work. Teamwork, Commitment collaboration, and Competency are the core skills to become more agile. The teams require a coach or a mentor but not a manager.
12. **Regularly Reflect on Continuous Improvement:** During the meeting, teams reflect on how they can improve the processes, productivity, and performance by making small changes. It is a continuous process to create a culture of continuous learning and working on improvement areas.

Various Agile Frameworks

We have covered Agile Scrum and Kanban in this chapter, among the many Agile frameworks listed as follows:

- **Kanban**: We have covered the details of this framework in the next section.
- **Scrum**: We have covered the details of this framework in the next section.
- **Feature-driven Development (FDD)**: FDD is a software development methodology that follows an iterative and incremental approach. In FDD, the development process is broken down into manageable pieces — features — and then it is built incrementally. In this process, an overall model is developed, followed by creating a feature list, feature-wise planning, features-based design, and features development.

  In FDD, the development process is structured around feature teams. Each of these teams is assigned the responsibility of delivering specific features. This way, the development effort is divided into smaller, more manageable tasks, making it easier to track progress and ensure that each feature is developed effectively. It maintains the iterative approach.

  One of the important characteristics of FDD is its emphasis on collaboration. It promotes close interaction among team members, stakeholders, and clients. It helps to understand the project requirements correctly.

It also emphasizes well-defined processes. To summarize, it is a structured and flexible approach to software development.

- **Extreme Programming**: Extreme Programming (XP) is an Agile software development methodology that promotes collaboration, adaptability, and high-quality code. It emphasizes practices like continuous testing, pair programming, and frequent releases. With a focus on customer involvement and iterative development, XP makes sure that software remains responsive to changing requirements. By fostering teamwork, automation, and streamlined processes, XP enables developers to deliver reliable software efficiently while maintaining a sustainable work pace. Extreme Programming is suitable for teams working on dynamic projects with evolving requirements.
- **Adaptive Software Development (ASD)**: ASD is a software development approach that embraces change as an inherent part of the process. ASD involves three distinct phases: speculation, collaboration, and learning. Speculation includes creating a preliminary plan based on what is known, while collaboration encourages open communication and active participation from both the development team and stakeholders. The learning phase emphasizes adapting the plan based on new information and insights gained throughout the process. ASD recognizes that software projects often encounter unexpected shifts and encourages a flexible and iterative approach to development, ensuring that the end result aligns closely with evolving needs and requirements.
- **Dynamic Systems Development Method** (**DSDM**): It is an Agile methodology that focuses on delivering functional solutions within a fixed timeframe and budget. DSDM divides development into specific time-bound phases, promoting regular user involvement and feedback. It encourages collaboration between developers, users, and business representatives to ensure that the software aligns with business needs. DSDM emphasizes the importance of delivering the most valuable features first, and it provides a set of principles and practices to guide teams in building high-quality solutions while accommodating changes. Overall, DSDM provides a structured framework for Agile software development, aiming to strike a balance between fixed constraints and adaptability.
- **Scrumban**: Scrumban is a hybrid approach that combines the practices of both Scrum and Kanban methodologies. In Scrumban, teams use the structured framework of Scrum but add elements from Kanban to enhance flexibility and continuous improvement. This allows teams to transition smoothly between planned iterations (sprints) and a more flow-based

approach. Scrumban is especially useful for teams that are already familiar with Scrum and want to introduce the focus of Kanban on optimizing workflow and minimizing bottlenecks. Scrumban can be implemented when a company wants to allow more flexibility to the teams or for a team facing problems implementing scrum.

Defining Scrum

Scrum is an Agile project management framework to create solutions for complex problems. It can be compared to the game of Rugby in which the goal is set and achieved by working on the principles of collaboration, teamwork, transparency, and agility. In scrum, work is divided into small time intervals called sprints, which can last for 1–4 weeks. The work items are also divided into small chunks called stories, tasks, and more. Each sprint begins with a planning meeting where the priority of the tasks to be taken is decided. The team meets daily to discuss the tasks done, tasks to be taken up, and to report any obstacles in completing the tasks planned to achieve the sprint goal. The teams are empowered, motivated, and work collaboratively to adapt to changing requirements at the same pace.

Scrum Roles, Ceremonies, and Artifacts

This section explains scrum roles, ceremonies, and artifacts.

In scrum methodology, there are three defined roles: scrum master, product owner, and the development team, which collaborate with each other:

- **Scrum Master:** Scrum Master is a facilitator and a scrum expert. He makes sure that everybody in the team understands how the team has to collaborate, set the sprint goal, and achieve the same. He helps the team to remove any impediments and leads the scrum meetings.
- **Scrum Team:** The scrum team is a team of developers and testers working together, collaborating, and working towards achieving the sprint goal. Team members can communicate with the product owner to understand the product specifications and task priorities. The team follows the guidance provided by the scrum master in case of any obstacles such as resources, server availability, or any other problems faced during the sprint.
- **Product Owner:** Product owner is the single point of contact between the customer/end user and the scrum team. He manages the product backlog, helps in sprint planning, and participates in scrum meetings. He is also responsible for product backlog grooming and deciding the priorities of the user stories.

Scrum ceremonies provide opportunities for planning, tracking, and feedback. During these ceremonies, scrum teams interact with each other to understand the requirements, plan sprints, communicate about the obstacles, demonstrate the working software, and move forward with the lessons learned and action plan.

- **Sprint Planning:** It is a collaborative planning done by the product owner, scrum master, and development team. They discuss the priorities and decide which tasks to be included in the sprint backlog. During this ceremony, the team decides the sprint goal, which means deciding what they will deliver/develop in this sprint. The team estimates the effort required for each task and creates a plan to achieve the sprint goal.
- **Daily Stand-up:** It is conducted daily in which team members and the scrum master discuss the progress done.

  Points to discuss in Daily Scrum are as follows:

  - What did I do yesterday as per the current sprint plan?
  - What will I do today to meet the Sprint goal?
  - Share about any impediment that prevents me/team towards completing the ongoing task in the sprint.

  The members discuss the work done on the tasks, which task they will be taking up next, and share any impediments with the scrum master. It builds a culture of transparency, collaboration, and accountability among the team.
- **Sprint Review:** It is done at the end of the sprint. In this ceremony, team members demonstrate the features and functionality developed to the stakeholders. Based on the feedback from the stakeholders, improvements can be made in the next sprint.
- **Sprint Retrospective:** This ceremony is held to reflect on the completed sprint. In this ceremony, what went well, what could be improved, and actions to be taken are discussed. The team and developers identify the areas for improvement. A culture of continuous improvement and learning is created by conducting this ceremony.

There are three main artifacts in Scrum: Product backlog, Sprint backlog, and Increment.

- **Product Backlog:** Product backlog is a list of features, bugs, and any other type of requirements to develop a product and is managed by the product owner.
  - It is a complete list of items to design and develop the product.

    - It is an ordered list and evolves continuously.
    - The product owner is fully responsible for this artifact.
    - It includes features, functionalities, fixes, and any other details required.
- **Sprint Backlog:** Sprint Backlog is a complete list of tasks to be taken up in the next sprints and is managed collectively by everyone on the team.
    - It is a complete list of work items to be taken up in a sprint.
    - It is a detailed list that includes estimation and who will work on what details.
    - It is updated or maintained by the developers.
- **Increment:** In Scrum methodology, increment is the combination of the software developed earlier plus the software developed in the recent sprint, which is integrated, tested, and ready to be released.
    - It is work that is complete and meets the definition of done.
    - It must be a working feature/functionality that can be used by the customer.
    - An increment does not necessarily equal a release.

Agile scrum artifacts are the details used by the team and stakeholders. It is generated during the planning, breaking the tasks, working on the sprints, and at the end of the sprint. There are also other artifacts such as the definition of Done and the Burndown Chart.

Creating User Stories and Product Backlogs

User stories are a way of writing customer requirements in a simple language that anyone can understand. The number of such user stories, along with other tasks and features, are added to create the product backlog. The product backlog can be visualized and prioritized in a project management tool and is regularly updated by product owners to meet new business needs.

User Stories:

- It is a document that contains information about the product requirements of the end-user in an understandable format.
- It contains information such as a description of the user, what exactly the user wants from that feature/functionality, and how the user will benefit from that feature.

- For example: As a user, I can find important items on the board by using the customizable “Quick Filters” so that I don’t have to search for those work items every time and work efficiently.
- For each and every user story, there is an acceptance criterion written, which makes sure the specific conditions are met. Based on the criteria, it is accepted as a working functionality or else it gets rejected.
- User stories can be divided into sub-tasks to simplify the work by the developers.
- It is user-centric and a new way of writing requirements in Agile methodology.
- User stories will evolve based on received feedback, providing clear specifications for both technical and non-technical team members.

Product Backlog:

- A product backlog is a prioritized list of features, functionalities, improvements, suggestions, and bugs in detail.
- It is a bucket of tasks from which one can take up tasks and create a sprint backlog based on the priority decided by the scrum team.
- It is continuously refined during the development cycle and tasks can be added and modified based on the change in the requirements and other conditions.
- Priority of the work items is decided based on the value it provides to the customer.
- It is created and maintained by the product owner and is updated to meet the final objective of the product to be developed.
- The team and product owner estimate the tasks in the product backlog and break them down based on the complexity of the tasks.
- In a project management tool, it can be viewed on boards and then moved to the sprint backlog based on priority.

Defining Kanban

Kanban, derived from the Japanese word for **signboard**, was originally implemented by Toyota in Japan as part of their manufacturing processes. This innovative system served as a signaling mechanism, effectively enhancing the efficiency of their manufacturing units by visualizing bottlenecks within the system.

Kanban is a **Pull system**. In this method, work is done when there is a demand. Known as Just in Time (JIT), Kanban methodology employs a board where tasks or work items progress from left to right as the work unfolds. This concept has been embraced not only by manufacturing but also by software development teams. On a Kanban board, tasks are categorized into three main statuses: **To do**, where prioritization occurs before moving on to design, coding, testing, and release tasks in progress; and finally, tasks are moved to the **Done** status, signifying completion.

Here are some of the Kanban practices, principles, benefits, and disadvantages:

- **Core Practices:**
    - Visualize the workflow
    - Limit WIP (Work in Progress)
    - Manage flow
    - Make process policies explicit
    - Take feedback
    - Improve collaboratively, evolve experimentally
- **Principles:**
    - Start with what you know
    - Agree to pursue incremental, evolutionary change
    - Respect the current process, roles, responsibilities, titles
    - Encourage acts of leadership at all levels in your organization
- **Benefits:**
    - Identifying issues
    - Flexibility
    - It can save time
    - It empowers teams
    - Balance productivity
    - Increased customer satisfaction
- **Disadvantages:**
    - Cannot be used independently
    - May be difficult where dependencies are involved

- Inability for an iteration
- Too simple board
- Lack of time representation

Approaching Kanban

The Kanban approach should be selected in the following situations:

- When the teams are new to the Agile way of working and the team is very small.
- When the nature of work varies continuously, such as changing priorities, task durations, and inflow of new requirements.
- Optimizing flow is more important than fixed timeframes.
- When work items cannot be easily broken down into sprints.
- When there is a need to be flexible and adapt to changes in requirements, market conditions, and customer feedback.
- When there is a need for managing work in progress to identify bottlenecks and manage capacity.
- When there is a need for continuous improvement in processes and more visibility of task status and progress through Kanban boards.

Comparison of Scrum and Kanban Frameworks

Scrum and Kanban frameworks are similar; however, there are some distinctions, as mentioned in *Table 1.2*:

Factors	Scrum	Kanban
Planning	Done at the beginning of the sprint/iteration	JIT Planning means - no planning is done
Sprint	It can be of 1-4 weeks time	There are no sprints/iterations
Roles	PO, SM, and team	No Roles defined: There might be an Agile Coach
Boards	Indicates the workflow steps	Indicates the workflow steps and WIP limits
Commitment	Team commits for each sprint: Sprint may fail if capacity is not measured	Commitment is based on Capacity: WIP limits help team members working on multiple tasks
Workload	Workload: Limits WIP per iteration	Workload: Limits WIP per status

Modifications	Once the sprint begins, not allowed to add tasks in the iteration…scope creep	You can add tasks in the backlog and take up based on the priorities
Industries	Software Development, Construction, Data Analytics, Advertising, Pharmaceuticals	Marketing, Content creation, Manufacturing, Healthcare, Fashion

Table 1.2: *Comparison of Scrum and Kanban Frameworks*

Estimating and Prioritizing Work in Agile

Estimating and deciding the priority of tasks is an important aspect of Agile planning. The task estimation is done with the help of techniques such as story points, t-shirt sizing, Fibonacci sequence, and many more. In these techniques, estimation is done based on the complexity of the tasks and not on the time taken. Tasks are moved to the sprint after deciding the priority. Priority may change throughout the cycle and the decision is taken by the team based on the value it provides to the customer. The most important features are delivered first to the customer. This way, it is a value-driven approach that takes into account the feedback provided by the customer.

Agile Metrics and Performance Tracking

Agile metrics are a quantitative measurement of the agile project life cycle used to track and improve performance. We have defined some must-learn Agile metrics as follows:

- Agile metrics are used to track the project and improve the performance of the team and develop better products.
- Burndown charts and burn-up charts measure the amount of work remaining and completed, respectively. It helps to understand the progress towards the sprint -goal.
- Cycle-time is a measurement of the time it takes from start to finish the project. It helps to understand the speed and efficiency of the teams.
- Lead-time is a measurement of the time it takes a feature to enter the backlog, then design-coding-release, and finally reach the customer.
- Velocity is a measurement of how much work a team can complete in a given time frame. It is measured in story points or number of stories completed.
- Epic and release burndown charts track the progress over an epic or release, which is the larger body of the work.

- There are many other metrics and charts such as Control chart, Cumulative flow diagram, No. of defects, Customer support requests, Customer retention rates, and many more. These are used to measure the performance of the Agile project life cycle.

About Jira

Jira is a robust project management tool used by Agile teams worldwide. It was designed by an Australian software company - Atlassian in 2002. It has evolved as a unique tool, which has many variants and can be customized for all types of projects and teams. Jira offers features to plan, track, and release software and collaborative platforms for teams. It is used for work item management, bug tracking, agile project management, and IT service management with the help of customizable workflows and integration with other add-ons. Jira is a popular tool for many other tasks, such as requirements gathering, test case management, and documentation.

A Step-by-Step Guide to Create a Jira Site

We will learn how to set up Jira on Cloud.

These are the steps to create a Jira site free for 10 users:

1. Go to: https://www.atlassian.com/software/jira/pricing

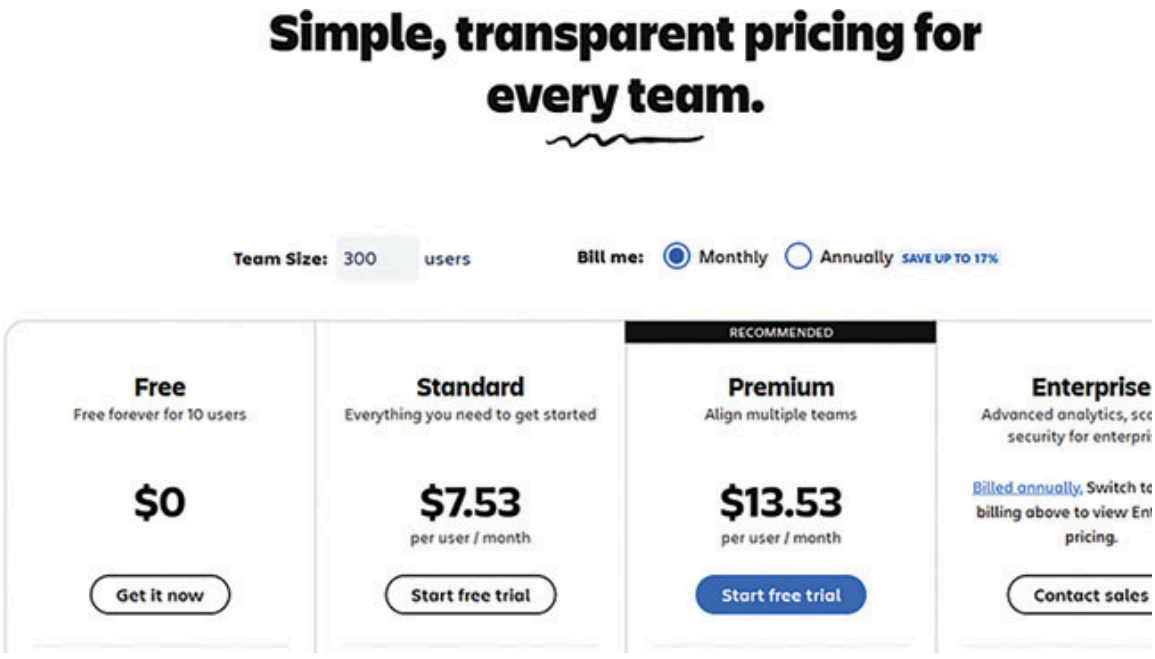


Figure 1.1: *Step 1 to Create a Jira Site*

2. Choose the free version and click on "`Get It now`"
3. Enter your work email or other email ID
4. Follow the prompts to sign up for an Atlassian account

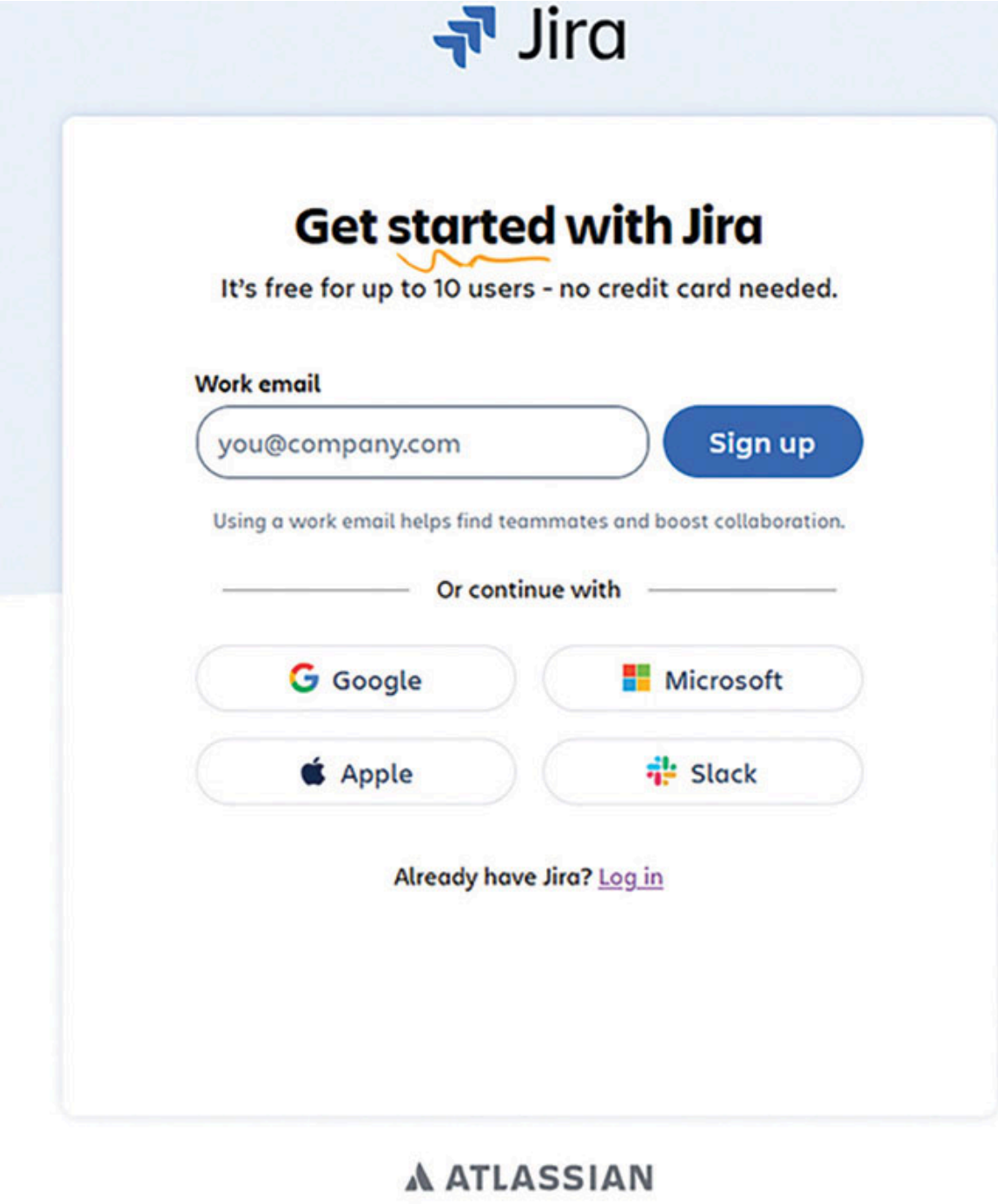


Figure 1.2: *Step 2 to Create Jira Site*

5. After signing up, login to your Jira Cloud Account

6. Insert the site name for your site to be created

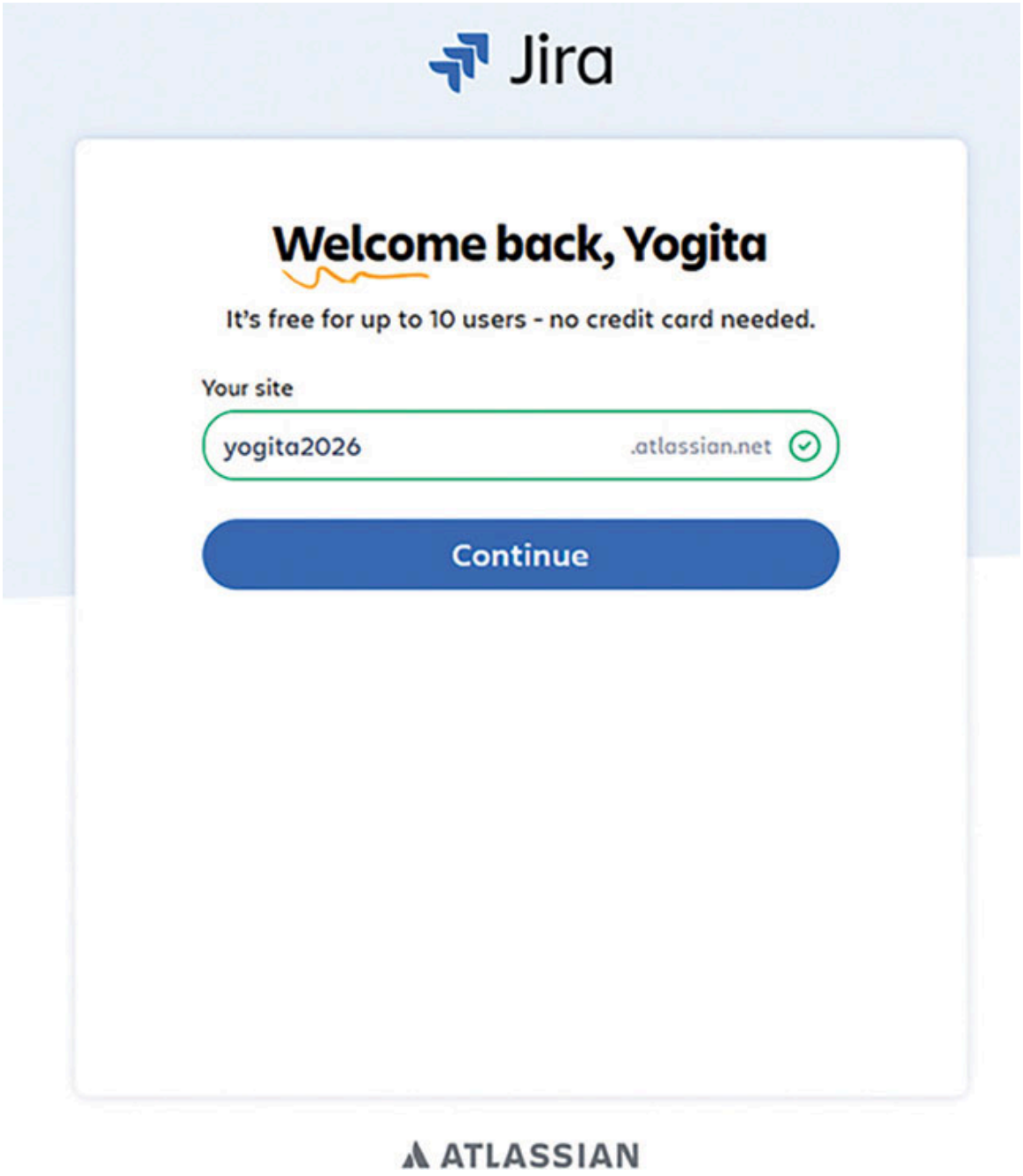


Figure 1.3: *Step 3 to Create Jira Site*

7. Click `Continue`
8. Jira will create your site and your Jira site is ready.

Different Atlassian Products

Let us explain some Jira products as follows:

- **Jira**: A tool used worldwide to manage projects for software teams, featuring agile development, time tracking, work management, program management with the help of features like customizable templates, artificial intelligence and agentic AI. Jira is a solution designed for business teams in marketing, sales, HR and finance. It is a friendly tool built for cross team collaboration and coordination for all types of teams.
- **Jira Service Management**: It is a Service desk and ITSM solution with ticketing, incident management, problem management, and self-service features.
- **Jira Align**: It is an Enterprise Agile Planning platform that helps improve visibility, strategic alignment, and enterprise adaptability to accelerate your digital transformation.
- **Jira Product Discovery**: A tool for product teams to organize and prioritize ideas, share product roadmaps, and connect business and tech teams, all in Jira.

Additionally, there are other Products that are used for version control repository, code review, team collaboration, and incident management:

- **Bitbucket**: A source-code management and version control repository with code collaboration and CI/CD capabilities.
- **Opsgenie**: Incident management and alerting tool for prompt response and resolution of critical issues.
- **Statuspage**: Communication tool for real-time status updates, incident notifications, and customizable incident response pages.
- **Trello**: Visual project management tool using boards, lists, and cards for organizing tasks and projects.
- **Sourcetree**: Sourcetree is a free Git client for Windows and Mac, which simplifies how you interact with your Git repositories.
- **Bamboo**: Bamboo is a continuous delivery pipeline that offers resilience, reliability, and scalability for teams of any size.
- **Fisheye**: Fisheye makes it easy to search, track, and compare code changes.
- **Crucible**: Crucible is a collaborative code review platform that helps you find bugs and improve code quality.
- **Confluence**: Team collaboration software for creating, organizing, sharing documents, project plans, and knowledge bases.

- **Loom**: Loom makes it easy to create and share videos directly within Atlassian apps like Jira, Confluence and Bitbucket, helping teams communicate visually. The app includes AI-powered features and simple editing tools for quick and professional videos.
- **Rovo**: Rovo is the AI-powered digital assistant of Atlassian that helps teams find, learn and act on information across Atlassian and connected apps faster.
- **Focus**: Atlassian Focus is a strategic planning hub that connects goals, work, teams and funds in real time to align execution with organizational priorities.
- **Talent**: It is an app that helps leaders plan, align and manage their workforce with real-time insights.

The Popularity of Jira

Jira is a very famous and customizable tool that has been used by Agile teams and other teams for 20 years. Here are some features and functionalities of Jira as follows.

- **Designed for All Types of Teams:** It is a unique tool that is the right fit for software development and various other teams, such as HR, marketing, sales, service, and more. It was originally designed for bug tracking in software development, but it has evolved for all types, sizes, and industries.
- **Pricing and Plan:** Jira products are available in a free plan for 10 users and three different variants. There are three variants available based on the pricing, number of users, features, and other criteria.
- **Customizable Space Templates:** Jira provides built-in space templates for various purposes, such as:
    - Scrum
    - Kanban
    - Bug Management

  Each space can be of two types:
    - Company-managed space, and
    - Team-managed space.

In addition to this, there are many other templates to manage various processes and projects. We will understand the same in *Chapter 2, "Working with Space*

Templates".

- **Time Tracking:** Jira has a time tracking feature that shows the time estimated, time spent, and time left for a task.
- **Bug Management:** In Jira, a template for bug management is readily available. It can be customized easily to manage the bugs for agile software development teams. Users can add fields available to customize the process in more detail.
- **Work Management:** Users can create work item types/task types as per the requirements of the spaces. Work items can be created, edited, and managed based on permissions and roles.
- **Search Filters:** The searching feature in Jira is helpful to find out any work items by setting criteria to filter work items. There is a basic search and an advanced search. Basic search helps filter work items based on task types, spaces, assignees, status, and more, while advanced search uses Jira Query Language (JQL), which consists of a set of keywords, functions, operators, and values.
- **Timeline:** Jira has a feature called Timeline View that has been used for planning, tracking, and creating tasks and map dependencies. In this view, one can create parent and child work items and schedule the same. The linking is done automatically between the tasks created in the timeline view. So, all the tasks created are child work items under an epic, which is the parent task.
- **Advanced Roadmap:** Advanced Roadmap is a feature of planning across multiple teams and spaces. It is an inbuilt feature in Premium and Enterprise versions. You can plan based on capacity, priorities, set dependencies, and explore different scenarios in the Advanced roadmap.
- **Automation in Jira:** Jira has rich templates for automation rules. Automation rules help teams automate the repetitive tasks and simplify the processes. There is an automation library for beginners to learn and implement the rules. The rules are of two types: space-specific and Global rules.
- **Community Support:** Atlassian community is a group of users of Atlassian products and Community Champions. It is a very big community where users can ask questions and clear doubts. Beginners can get help from community champions and other people in the group as and when it is required. By collaborating with experienced people, anyone can learn everything about Atlassian products.

- **Boards:** Jira boards are powerful tools used for visualizing, updating, and tracking purposes. There are two types of boards: Scrum and Kanban boards. Jira boards can be customized for sorting the work items based on priority and other criteria. One can also match the statuses in Jira workflow and Jira boards. It also provides insights about the epics, sprints, and work items to be worked on. Users can move tasks between statuses as per the customized workflow.
- **Dashboards and Reporting:** Jira is a rich project management tool with customizable dashboards and reporting features. One can add gadgets and create their own dashboards, making it easy to present critical information to management during meetings. The reporting functionality of Jira provides Agile metrics, such as burndown charts and cumulative flow diagrams, allowing teams to track project progress and identify bottlenecks. Jira offers reports for time tracking, workload, and resolution time, enabling teams to make data-driven decisions and optimize their processes.
- **Atlassian Intelligence and Rovo:** Jira became richer with the introduction of AI features and Rovo Agents. AI helps teams generate Work Breakdown Structures, refine work item descriptions and summarize updates instantly. Users can create no code/low code Agents. With Rovo Agents, everyday tasks get automated, tickets are managed smoothly and right information can be found easily. Together, they make Jira easier to use and free up your time for real work.

Conclusion

We had started by learning about the necessity of a new way of working in various industries to manage projects. We explored the importance of adaptability and collaboration through the values and principles of Agile. We also discussed when to use Scrum and Kanban, including practices such as user stories, backlogs, estimation, and prioritization. Finally, we covered Agile metrics for performance tracking.

We are now familiar with the Agile manifesto and concepts of Scrum and Kanban methodologies in Agile software development. We also learned about the various Jira products and other product solutions offered by Atlassian including the incorporation of Artificial Intelligence and Agentic AI. We now know the reasons behind the popularity of Jira and also its terminologies.

In the next chapter, we will learn how to create the first space in Jira, configure different spaces, and explore a few examples of space templates in Jira.

Terminologies

- **Attachment**: A file attached to a Jira work item.
- **Backlog**: A list of user stories, bugs, and features for a sprint or a product to be worked upon.
- **Board**: It is the place from where information about the work items can be read and updated. It provides status-based information, and the tasks can be moved from one status to another on the board.
- **Burndown Chart**: It shows the amount of total work estimated and remaining in a sprint.
- **Component**: It is a subcategory of the space; for example, Modules of a space.
- **Field**: Jira provides its own system fields and other customizable fields. The custom field is a field used by the users to capture additional information about the work item.
- **Control Chart**: It provides information about the lead-time and cycle-time for a space or sprint.
- **Dashboard**: Using this feature, one can present information about the key metrics of a space.
- **Dependency**: A relationship between two or more work items when they rely on each other to get completed.
- **Epic**: A large task that has to be broken down into smaller tasks or stories.
- **Fix Version**: A version of a software application in which a bug is fixed.
- **Group**: A collection of users/Administrators in Jira who have common permissions and responsibilities.
- **Work Item**: It is a unit of work, which is also named as a task. It can be based on the work a person is supposed to do. For example, tasks, bugs, epic, story, and more, which can be tracked through workflow.
- **Impediment**: An obstacle that can prevent work on a work item.
- **JQL (Jira Query Language)**: It is the own language of Jira used to search and display a work item.
- **JSM**: It stands for Jira Service Management.
- **JWM**: It stands for Jira Work Management.
- **Key**: A unique identifier for each and every work item in Jira.
- **Label**: A tag that can be used to search and sort the work items in Jira.

- **Notification**: An email sent from Jira to notify events like create, edit, delete work items, and many other events.
- **Space**: A combination of work items, components, workflow steps, and other resources in Jira to work towards achieving a goal.
- **Priority**: It defines the level of importance assigned to the work items used to decide its order in the backlog.
- **Parent Work Item**: A work item in Jira, which is related to one or multiple child work items.
- **Release**: A version of a software application that is delivered to the customer.
- **Resolution**: The way of fixing or completing a work item in Jira and sending it to the resolved status.
- **Screen**: It is the combination of fields that creates a form to accept information about a work item.
- **Story**: It is the task type in Agile project management, which is used to describe the needs of the end user.
- **Sprint**: It is the time span during which the software development teams design, develop, and implement the small working piece of software. It is also called an iteration and can be 1-4 weeks.
- **Story Point**: It is a unit of estimation used by Agile teams. It is the measurement of the relative complexity of a story.
- **Swimlane**: It is the way to categorize the work items. Based on the categorization, teams can see which work items to work on.
- **Sub-task**: It is the child work item of task, story, and more, which is smaller than the parent task.
- **Task**: A piece of work in Jira that needs to be done to complete a process.
- **Time-tracking**: The way of logging and tracking the time spent on different tasks in Jira.
- **Transition**: When a task moves from one status to another in Jira, it is called transition.
- **User**: Any person who works to manage and track work items in Jira.
- **Version**: A release of a software application, which is a set of work items and bug-fixes.
- **Velocity**: It is the measure of how much work a team can take up in a sprint. It could be based on story points, work item counts, value, and more.

- **Work-flow**: It is the process-steps to complete a task from the start to resolution.
- **Work-log**: A record of the time spent while working on a work item in Jira.