**Basics of Jira**

Jira is a popular project management tool developed by Atlassian, commonly used for **issue tracking, bug tracking, and agile project management**. It’s widely used in software development but can be applied to other project types too.

**1. What is Jira?**

* **Purpose:** Tracks tasks, bugs, and project progress.
* **Popular Use Cases:**
  + Agile/Scrum project management.
  + Software development lifecycle (SDLC) management.
  + Bug and issue tracking.

**2. Jira Key Concepts**

**Project** – A container for tasks, issues, and workflows related to a specific goal.

**Issue** – A task, bug, or feature that needs to be worked on. Issue types include:

* **Task** – Work that needs to be done.
* **Bug** – Error/issue in the software that needs fixing.
* **Story** – A user requirement or feature.
* **Epic** – A larger body of work that can be broken down into stories or tasks.

**Board** – A visual representation of your project using Scrum or Kanban workflows.

**Backlog** – List of issues/tasks that need to be prioritized and worked on.

**Sprint** – A set period (usually 1-2 weeks) where a specific amount of work is completed.

**Workflow** – The lifecycle of an issue, showing different stages (To Do → In Progress → Done).

**3. Basic Jira Workflow**

1. **Create a Project**
   * Choose from Scrum, Kanban, or other templates.
2. **Create Issues/Tasks**
   * Define tasks and assign them to team members.
   * Add descriptions, attachments, and due dates.
3. **Move Issues Through Workflow**
   * Move tasks from **To Do** → **In Progress** → **Done**.
   * Customize workflows if needed.
4. **Track Progress**
   * Use boards and reports to monitor sprint progress.

**4. Basic Jira Features**

**Dashboard** – Overview of projects, issues, and sprint status.  
**Boards (Scrum/Kanban)** – Visual representation of issues and workflows.  
**Epics and Stories** – Break down large tasks into manageable units.  
**Reports and Analytics** – Track team performance and sprint progress.  
**Filters and JQL (Jira Query Language)** – Search and filter issues based on conditions.

**5. Jira Agile Boards**

1. **Scrum Board** – Best for iterative work in sprints.
2. **Kanban Board** – Ideal for continuous workflows.

**6. Basic Jira Reports**

* **Burndown Chart** – Tracks remaining work in a sprint.
* **Velocity Chart** – Measures work completed in past sprints.
* **Cumulative Flow Diagram** – Tracks the progress of issues over time.

**Agile and Scrum Basics**

**1. What is Agile?**

**Agile** is a **project management methodology** focused on delivering small, incremental changes quickly while responding to feedback. It’s widely used in software development but applies to many industries.

**Core Principles (from Agile Manifesto):**

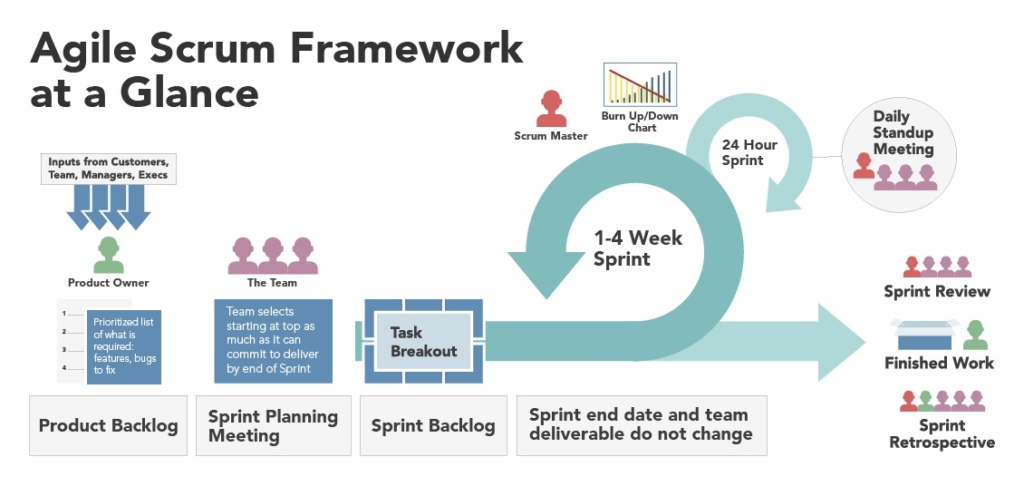
* **Customer collaboration** over contract negotiation.
* **Responding to change** over following a plan.
* **Working software** over comprehensive documentation.
* **Individuals and interactions** over processes and tools.

**Top 5 Reasons to Use Agile**

1. **Flexibility and Adaptability** – Easily accommodates changing requirements, even late in the development process.
2. **Faster Time to Market** – Delivers small, incremental releases, allowing quicker feedback and early product delivery.
3. **Continuous Feedback and Improvement** – Regular reviews ensure the product aligns with customer expectations.
4. **Better Collaboration and Transparency** – Promotes teamwork and visibility through daily standups and regular communication.
5. **Higher Product Quality and Reduced Risk** – Frequent testing and integration catch defects early, minimizing project failure.

**Popular Agile Methodologies:**

* **Scrum** – Uses sprints (short development cycles), daily stand-ups, and roles like Scrum Master and Product Owner.
* **Kanban** – Focuses on visualizing work (via boards) and limiting work in progress (WIP).
* **Extreme Programming (XP)** – Emphasizes coding best practices, pair programming, and frequent releases.
* **Lean** – Aim to maximize value while minimizing waste.

**2. What is Scrum?**

**Scrum** is a **framework** within Agile used to implement Agile principles. It organizes work into **sprints** and promotes team collaboration to achieve goals.

**Scrum Core Concepts:**

* **Sprints:** Time-boxed periods (usually 1-4 weeks) where a set amount of work is completed.
* **Iterative Approach:** Deliver small, functional increments frequently.
* **Inspect and Adapt:** Regular reviews to improve processes.

**3. Scrum Framework Structure**

**Scrum Roles:**

1. **Product Owner (PO):**
   * Defines the product vision.
   * Prioritizes the backlog.
   * Ensures the team is building what the customer wants.
2. **Scrum Master (SM):**
   * Facilitates Scrum processes.
   * Removes blockers.
   * Coaches the team to follow Scrum practices.
3. **Development Team:**
   * Cross-functional team responsible for delivering the work.
   * Self-organized and collaborative.

**Scrum Artifacts:**

1. **Product Backlog:**
   * List of all features, bugs, and tasks that need to be completed.
   * Managed by the Product Owner.
2. **Sprint Backlog:**
   * Subset of prioritized tasks from the product backlog for the current sprint.
3. **Increment:**
   * The usable, shippable product delivered at the end of a sprint.

**Scrum Events (Ceremonies):**

1. **Sprint Planning:**
   * Goal: Define the scope of work for the sprint.
   * Input: Product backlog items.
   * Outcome: Sprint backlog.
2. **Daily Standup (Daily Scrum):**
   * 15-minute meeting to discuss:
     + What was done yesterday?
     + What will be done today?
     + Any blockers?
3. **Sprint Review:**
   * Demo the completed work to stakeholders.
   * Gather feedback for improvement.
4. **Sprint Retrospective:**
   * Reflect on the sprint.
   * Discuss what went well and what can be improved.
   * Plan changes for the next sprint.

**4. Scrum Workflow Overview**

1. **Product Owner** creates and maintains the product backlog.
2. **Sprint Planning** selects high-priority tasks for the sprint.
3. Team works on tasks and updates progress in the **Daily Standup**.
4. At the end of the sprint:
   * Completed tasks are reviewed in the **Sprint Review**.
   * Team reflects on the process in the **Sprint Retrospective**.
5. Process repeats for the next sprint.

**5. Key Scrum Metrics**

**Velocity:** Amount of work completed per sprint.  
 **Burndown Chart:** Visualizes remaining work vs. time.  
 **Sprint Goal:** Objective to be achieved in a sprint.

**6. Differences Between Agile and Scrum**

| **Feature** | **Agile** | **Scrum** |
| --- | --- | --- |
| **Scope** | Broad philosophy | Specific framework |
| **Flexibility** | Highly adaptable | Defined processes |
| **Delivery** | Continuous | Incremental sprints |
| **Role Focus** | Flexible roles | Defined roles |

**Basics of Git**

**Introduction to Git**

Git is a version control system (VCS) that helps developers track and manage changes to their codebase. It allows multiple people to collaborate on the same project without overwriting each other’s work.

**Why Use Git?**

* Track Changes: Keeps a history of modifications.
* Collaboration: Multiple people can work on a project simultaneously.
* Branching and Merging: Create separate branches for different features and merge them later.

**Basic Git Concepts**

1. Repository (Repo): A directory where Git stores the project’s files and history.  
   You can create a repo locally or clone an existing one.
2. Commit: A snapshot of your changes. Every commit has a unique ID (hash).
3. Branch: A branch is a separate line of development. The default branch is usually main or master.
4. Merge: Combines changes from different branches.
5. Staging Area: A place where you prepare your changes before committing.

**Basic Git Workflow**

1. **Initialize a Repository**

git init - Creates a new Git repository in the current directory.

1. **Clone an Existing Repository**

git clone <repository\_url> - Clones a remote repository to your local machine.

1. **Check Status**

git status - Shows the current status of your working directory.

1. **Add Files to Staging Area**

git add <filename> # Add a specific file

git add . # Add all files in the directory

1. **Commit Changes**

git commit -m "Your commit message"

Saves the changes to the local repository with a message describing the changes.

1. **View Commit History**

git log - Shows the commit history.

1. **Create a New Branch**

git branch <branch\_name>

1. **Switch to Another Branch**

git checkout <branch\_name>

1. **Merge Branches**

git merge <branch\_name>

1. **Push Changes to Remote Repository**

git push origin <branch\_name>

1. **Pull Changes from Remote Repository**

git pull

**Common Commands**

* git config --global user.name "Your Name"
* git config --global user.email "youremail@example.com"
* git remote add origin <repository\_url>
* git diff – Show changes between commits
* git reset <file> – Unstage file

**Undoing Changes**

1. **Unstage a File**

git reset <file>

1. **Undo Last Commit (Soft Reset)**

git reset --soft HEAD~1

1. **Discard All Local Changes**

git checkout -- .

**Basics of Bitbucket**

Introduction to Bitbucket

Bitbucket is a web-based platform that hosts Git repositories. It is used for version control, collaboration, and code management. Bitbucket supports Git and Mercurial (though Mercurial support was removed after 2020).

**Why Use Bitbucket?**

* Version Control: Keeps track of all code changes.
* Collaboration: Allows multiple developers to work on the same project.
* Pull Requests and Code Reviews: Ensure better code quality through reviews.
* Integration with Jira: Links issues and tasks with the code.

**Key Concepts**

1. Repository (Repo):  
   A repository stores all your project’s files and history. It can be either:
   * Public: Accessible to everyone.
   * Private: Accessible only to invited users.
2. Branch:  
   A branch is an independent line of development. The default branch is usually named main or master.
3. Commit:  
   A commit is a snapshot of your changes with a message describing the modifications.
4. Pull Request (PR):  
   A pull request is a request to merge changes from one branch into another. It allows team members to review and discuss changes before merging.
5. Merge:  
   Merging combines changes from different branches into a single branch.
6. Pipeline:  
   An automated process that builds, tests, and deploys code using a bitbucket-pipelines.yml file.

**Getting Started with Bitbucket**

1. Create a Bitbucket Account

* Sign up at [bitbucket.org](https://bitbucket.org/).
* Create a workspace or join an existing one.

2. Create a Repository

* Click on Repositories → Create Repository.
* Fill in the details:
  + Repository name
  + Project (optional)
  + Version control: Choose Git
  + Access level: Public or private
* Click Create Repository.

3. Clone a Repository

To clone a repository to your local system:

git clone <repository\_url>

4. Add Remote Repository

If you have an existing local project:

git init

git remote add origin <repository\_url>

5. Push Local Code to Bitbucket

git add .

git commit -m "Initial commit"

git push -u origin main

**Basic Git Commands with Bitbucket**

1. Check the Status

git status

1. Add Files to Staging Area

git add <filename> # Add a specific file

git add . # Add all files

1. Commit Changes

git commit -m "Commit message"

1. Push Changes to Remote Repo

git push origin <branch\_name>

1. Pull Changes from Remote Repo

git pull

**Working with Branches and Pull Requests**

1. Create a New Branch

git branch <branch\_name>

1. Switch to the New Branch

git checkout <branch\_name>

1. Push Branch to Bitbucket

git push -u origin <branch\_name>

1. Create a Pull Request

* Go to Bitbucket and open the repository.
* Click Pull requests → Create pull request.
* Select the source and destination branches.
* Add a description and reviewers if needed.
* Click Create pull request.

1. Merge a Pull Request

* After review, click Merge to integrate the changes.

**Undoing Changes**

1. Unstage a File

git reset <file>

1. Discard Local Changes

git checkout -- .

1. Undo Last Commit (Soft Reset)

git reset --soft HEAD~1