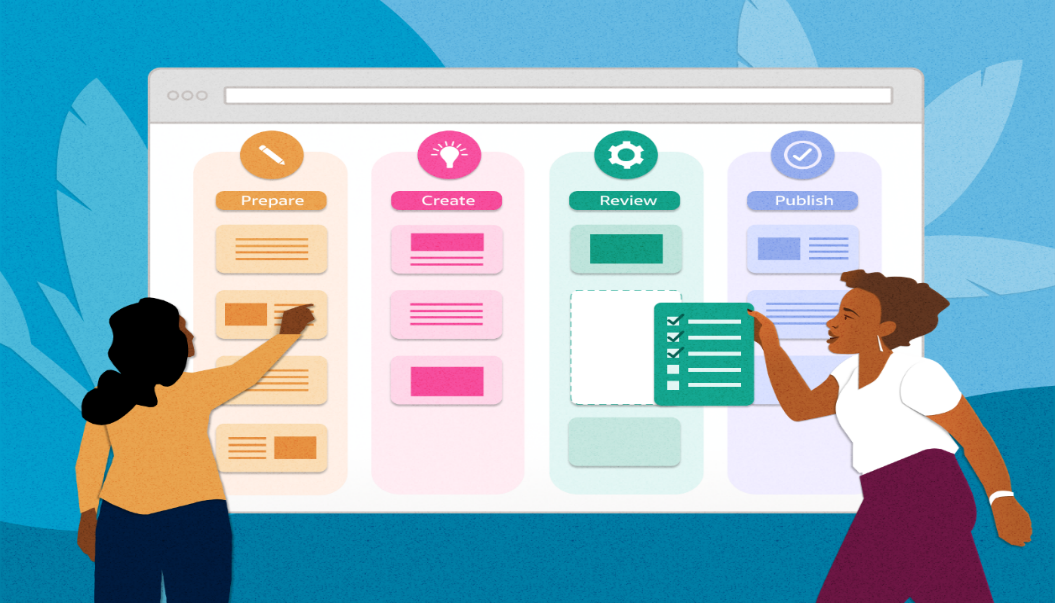
**Kanban :**

Kanban is a Japanese term that means signboard or billboard. An industrial engineer named Taiichi Ohno developed Kanban at Toyota Motor Corporation to improve manufacturing efficiency.

Although Kanban was created for manufacturing, software development shares many of the same goals, such as increasing flow and throughput. Software development teams can improve their efficiency and deliver value to users faster by using Kanban guiding principles and methods.



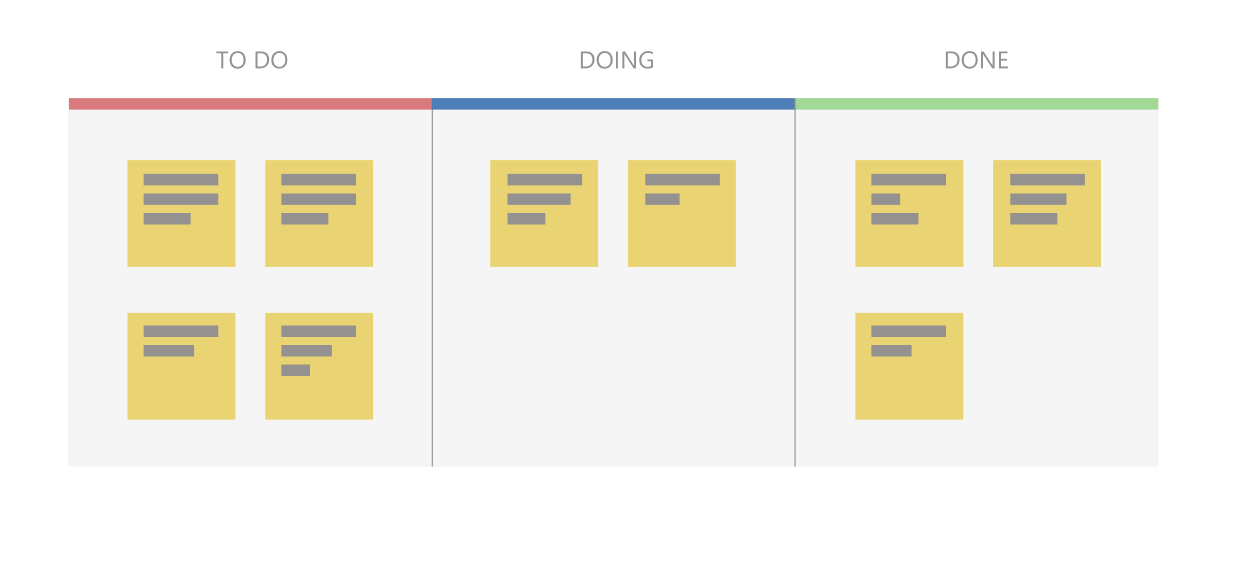
**Kanban principles :**

Adopting Kanban requires adherence to some fundamental practices that might vary from teams' previous methods.

**Visualize work**

Understanding development team status and work progress can be challenging. Work progress and current state is easier to understand when presented visually rather than as a list of work items or a document.

Visualization of work is a key principle that Kanban addresses primarily through *Kanban boards*. These boards use cards organized by progress to communicate overall status. Visualizing work as cards in different states on a board helps to easily see the big picture of where a project currently stands, as well as identify potential bottlenecks that could affect productivity.



**Use a pull model**

Historically, stakeholders requested functionality by pushing work onto development teams, often with tight deadlines. Quality suffered if teams had to take shortcuts to deliver the functionality within the timeframe.

Kanban focuses on maintaining an agreed-upon level of quality that must be met before considering work done. To support this model, stakeholders don't push work on teams that are already working at capacity. Instead, stakeholders add requests to a backlog that a team pulls into their workflow as capacity becomes available.

**Impose a WIP limit**

Teams that try to work on too many things at once can suffer from reduced productivity due to frequent and costly context switching. The team is busy, but work doesn't get done, resulting in unacceptably high lead times. Limiting the number of backlog items a team can work on at a time helps increase focus while reducing context switching. The items the team is currently working on are called *work in progress (WIP)*.

Teams decide on a *WIP limit*, or maximum number of items they can work on at one time. A well-disciplined team makes sure not to exceed their WIP limit. If teams exceed their WIP limits, they investigate the reason and work to address the root cause.

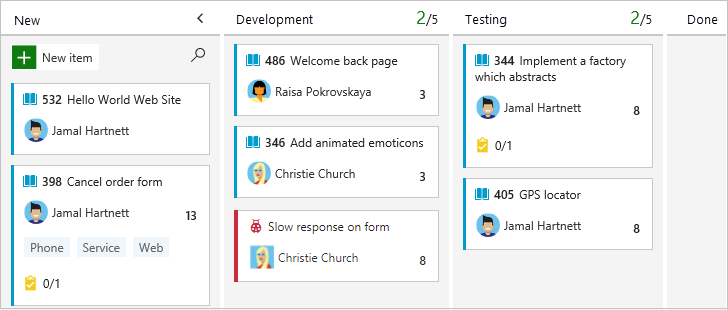
**Measure continuous improvement**

To practice continuous improvement, development teams need a way to measure effectiveness and throughput. Kanban boards provide a dynamic view of the states of work in a workflow, so teams can experiment with processes and more easily evaluate impact on workflows. Teams that embrace Kanban for continuous improvement use measurements like *lead time* and *cycle time*.

**Kanban boards**

The *Kanban board* is one of the tools teams use to implement Kanban practices. A Kanban board can be a physical board or a software application that shows cards arranged into columns. Typical column names are **To-do**, **Doing**, and **Done**, but teams can customize the names to match their workflow states. For example, a team might prefer to use **New**, **Development**, **Testing**, **UAT**, and **Done**.

Software development-based Kanban boards display cards that correspond to product backlog items. The cards include links to other items, such as tasks and test cases. Teams can customize the cards to include information relevant to their process.

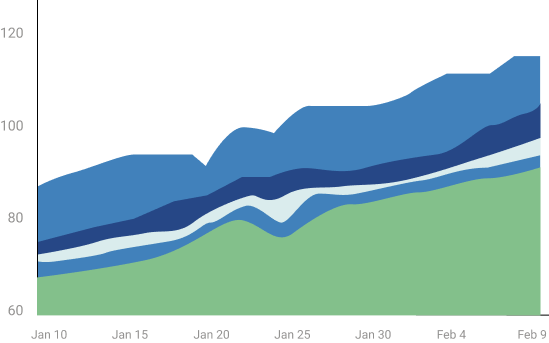


On a Kanban board, the WIP limit applies to all in-progress columns. WIP limits don't apply to the first and last columns, because those columns represent work that hasn't started or is completed. Kanban boards help teams stay within WIP limits by drawing attention to columns that exceed the limits. Teams can then determine a course of action to remove the bottleneck.

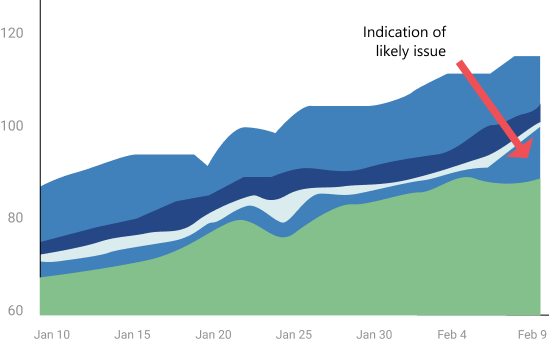
**Cumulative flow diagrams :**

A common addition to software development-based Kanban boards is a chart called a *cumulative flow diagram (CFD)*. The CFD illustrates the number of items in each state over time, typically across several weeks. The horizontal axis shows the timeline, while the vertical axis shows the number of product backlog items. Colored areas indicate the states or columns the cards are currently in.

The CFD is particularly useful for identifying trends over time, including bottlenecks and other disruptions to progress velocity. A good CFD shows a consistent upward trend while a team is working on a project. The colored areas across the chart should be roughly parallel if the team is working within their WIP limits.



A bulge in one or more of the colored areas usually indicates a bottleneck or impediment in the team's flow. In the following CFD, the completed work in green is flat, while the testing state in blue is growing, probably due to a bottleneck.



**Kanban and Scrum in Agile development :**

While broadly fitting under the umbrella of [Agile](https://learn.microsoft.com/en-us/devops/plan/what-is-agile) development, [Scrum](https://learn.microsoft.com/en-us/devops/plan/what-is-scrum) and Kanban are quite different.

* Scrum focuses on fixed length sprints, while Kanban is a continuous flow model.
* Scrum has defined roles, while Kanban doesn't define any team roles.
* Scrum uses velocity as a key metric, while Kanban uses cycle time.

Teams commonly adopt aspects of both Scrum and Kanban to help them work most effectively. Regardless of which characteristics they choose, teams can always review and adapt until they find the best fit. Teams should start simple and not lose sight of the importance of delivering value regularly to users.

**Kanban with GitHub :**

GitHub offers a Kanban experience through [project boards (classic)](https://docs.github.com/github/managing-your-work-on-github/about-project-boards). These boards help you [organize and prioritize work](https://docs.github.com/github/managing-your-work-on-github/tracking-the-progress-of-your-work-with-project-boards) for specific feature development, comprehensive roadmaps, or release checklists. You can [automate project boards (classic)](https://docs.github.com/github/managing-your-work-on-github/about-automation-for-project-boards) to sync card status with associated issues and pull requests.

**Kanban with Azure Boards :**

[Azure Boards](https://learn.microsoft.com/en-us/azure/devops/boards/boards/Kanban-overview) provides a comprehensive Kanban solution for DevOps planning. Azure Boards has deep integration across Azure DevOps, and can also be part of [Azure Boards-GitHub integration](https://learn.microsoft.com/en-us/azure/devops/boards/github).