Kanban

How the kanban methodology applies to software development

What is kanban?

Kanban is a popular framework used to implement agile and DevOps software development. It requires real-time communication of capacity and full transparency of work. Work items are represented visually on a kanban board, allowing team members to see the state of every piece of work at any time.

READ ON BELOW

Kanban articles

ARTICLE

What is a Kanban Board?

ARTICLE

Working with WIP limits for kanban

ARTICLE

Kanban vs Scrum

Discover if Kanban or Scrum

A kanban board is a physical or digital project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency(or flow).

Learn how to use work in progress limits, the 4 goals for agile teams using WIP limits, and why WIP limits are important. Get started here.

is a better framework for your agile team. Learn the key differences between the two frameworks.

ARTICLE

Kanplan: where your backlog meets kanban

Kanplan adds the backlog and backlog grooming concepts of scrum to kanban, using the backlog instead of the To Do column to plan and prioritize work.

TUTORIAL

Learn kanban with Jira Software

Step-by-step instructions on how to drive a kanban project, prioritize your work, visualize your workflow, and minimize work-in-progress with Jira Software

Try this tutorial →

PRODUCT FEATURE

From “to-do” to “done” with Jira kanban boards

The Jira Software kanban board is designed to help teams continuously improve cycle time and increase efficiency.

Get it free →

[CONTINUED]

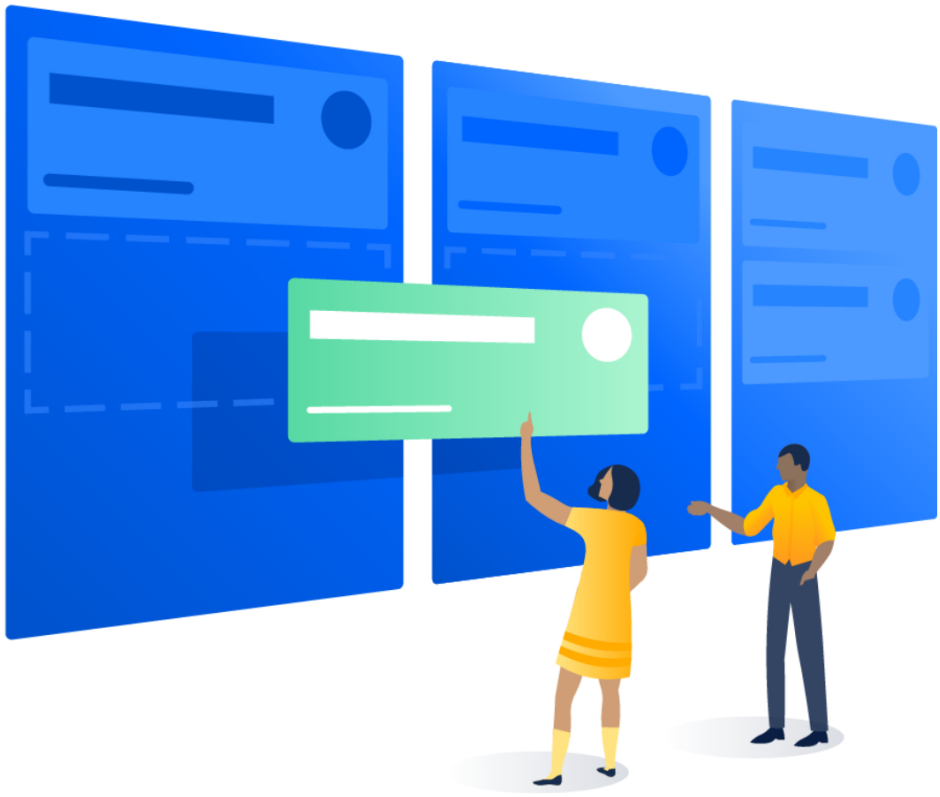
Kanban is enormously prominent among today's agile and DevOps software teams, but the kanban methodology of work dates back more than 50 years. In the late 1940s Toyota began optimizing its engineering processes based on the same model that supermarkets were using to stock their shelves. Supermarkets stock just enough product to meet consumer demand, a practice that optimizes the flow between the supermarket and the consumer. Because inventory levels match consumption patterns, the supermarket gains significant efficiency in inventory management by decreasing the amount of excess stock it must hold at any given time. Meanwhile, the supermarket can still ensure that the given product a consumer needs is always in stock.

When Toyota applied this same system to its factory floors, the goal was to better align their massive inventory levels with the actual consumption of materials. To communicate capacity levels in real-time on the factory floor (and to suppliers), workers would pass a card, or "kanban", between teams. When a bin of materials being used on the production line was emptied, a kanban was passed to the warehouse describing what material was needed, the exact amount of this material, and so on. The warehouse would have a new bin of this material waiting, which they would then send to the factory floor, and in turn send their own kanban to the supplier. The supplier would also have a bin of this particular material waiting, which it would ship to the warehouse. While the signaling technology of this process has evolved since the 1940s, this same "just in time" (or JIT) manufacturing process is still at the heart of it.

Kanban for software teams

Agile software development teams today are able to leverage these same JIT principles by

matching the amount of work in progress (WIP) to the team's capacity. This gives teams more flexible planning options, faster output, clearer focus, and transparency throughout the development cycle.



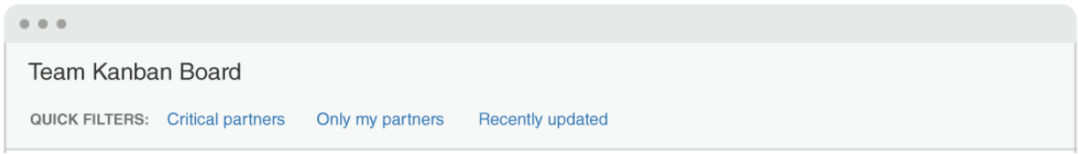
While the core principles of the framework are timeless and applicable to almost any industry, software development teams have found particular success with the agile practice. In part, this is because software teams can begin practicing with little to no overhead once they understand the basic principles. Unlike implementing kanban on a factory floor, which would involve changes to physical processes and the addition of substantial materials, the only physical things a software teams need are a board and cards, and even those can be virtual.

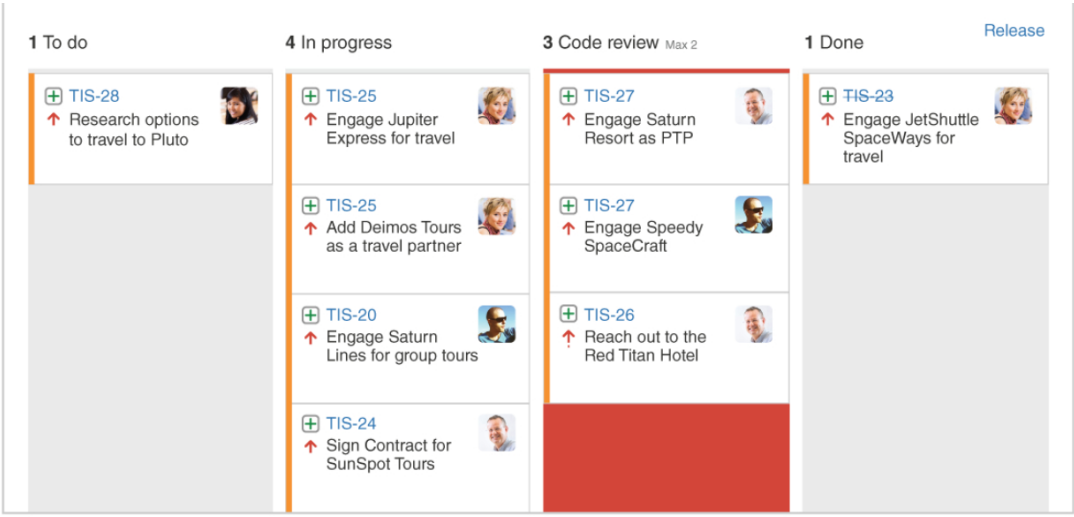
Kanban boards

The work of all kanban teams revolves around a kanban board, a tool used to visualize work and optimize the flow of the work among the team. While physical boards are popular among some teams, virtual boards are a crucial feature in any agile software development tool for their traceability, easier collaboration, and accessibility from multiple locations.

Regardless of whether a team's board is physical or digital, their function is to ensure the team's work is visualized, their workflow is standardized, and all blockers and dependencies are immediately identified and resolved. A basic kanban board has a three-step workflow: To Do, In Progress, and Done. However, depending on a team's size, structure, and objectives, the workflow can be mapped to meet the unique process of any particular team.

The kanban methodology relies upon full transparency of work and real-time communication of capacity, therefore the kanban board should be seen as the single source of truth for the team's work.





Kanban cards

In Japanese, kanban literally translates to "visual signal." For kanban teams, every work item is represented as a separate card on the board.

The main purpose of representing work as a card on the kanban board is to allow team members to track the progress of work through its workflow in a highly visual manner. Kanban cards feature critical information about that particular work item, giving the entire team full visibility into who is responsible for that item of work, a brief description of the job being done, how long that piece of work is estimated to take, and so on. Cards on virtual kanban boards will often also feature screenshots and other technical details that is valuable to the assignee. Allowing team members to see the state of every work item at any given point in time, as well as all of the associated details, ensures increased focus, full traceability, and fast identification of blockers and dependencies.

The benefits of Kanban

Kanban is one of the most popular software development methodologies adopted by agile teams today. Kanban offers several additional advantages to task planning and throughput for teams of all sizes.

Planning flexibility

A kanban team is only focused on the work that's actively in progress. Once the team completes a work item, they pluck the next work item off the top of the backlog. The product owner is free to reprioritize work in the backlog without disrupting the team, because any changes outside the current work items don't impact the team. As long as the product owner keeps the most important work items on top of the backlog, the development team is assured they are delivering maximum value back to the business. So there's no need for the fixed-length iterations you find in scrum.

> PRO TIP:
>
> Savvy product owners always engage the development team when considering changes to the backlog. For example, if user stories 1-6 are in the backlog, user story 6's estimate may be based on the completion of user stories 1-5. It's always a good practice to confirm changes with the engineering team to ensure there are no surprises.

Shortened time cycles

Cycle time is a key metric for kanban teams. Cycle time is the amount of time it takes for a unit

Cycle time is a key metric for kanban teams. Cycle time is the amount of time it takes for a unit of work to travel through the team's workflow–from the moment work starts to the moment it ships. By optimizing cycle time, the team can confidently forecast the delivery of future work.

Overlapping skill sets lead to smaller cycle times. When only one person holds a skill set, that person becomes a bottleneck in the workflow. So teams employ basic best practices like code review and mentoring help to spread knowledge. Shared skills mean that team members can take on heterogeneous work, which further optimizes cycle time. It also means that if there is a backup of work, the entire team can swarm on it to get the process flowing smoothly again. For instance, testing isn't only done by QA engineers. Developers pitch in, too.

In a kanban framework, it's the entire team's responsibility to ensure work is moving smoothly through the process.

Fewer bottlenecks

Multitasking kills efficiency. The more work items in flight at any given time, the more context switching, which hinders their path to completion. That's why a key tenet of kanban is to limit the amount of work in progress (WIP). Work-in-progress limits highlight bottlenecks and backups in the team's process due to lack of focus, people, or skill sets.

For example, a typical software team might have four workflow states: To Do, In Progress, Code Review, and Done. They could choose to set a WIP limit of 2 for the code review state. That might seem like a low limit, but there's good reason for it: developers often prefer to write new code, rather than spend time reviewing someone else's work. A low limit encourages the team to pay special attention to issues in the review state, and to review others work before raising their own code reviews. This ultimately reduces the overall cycle time.

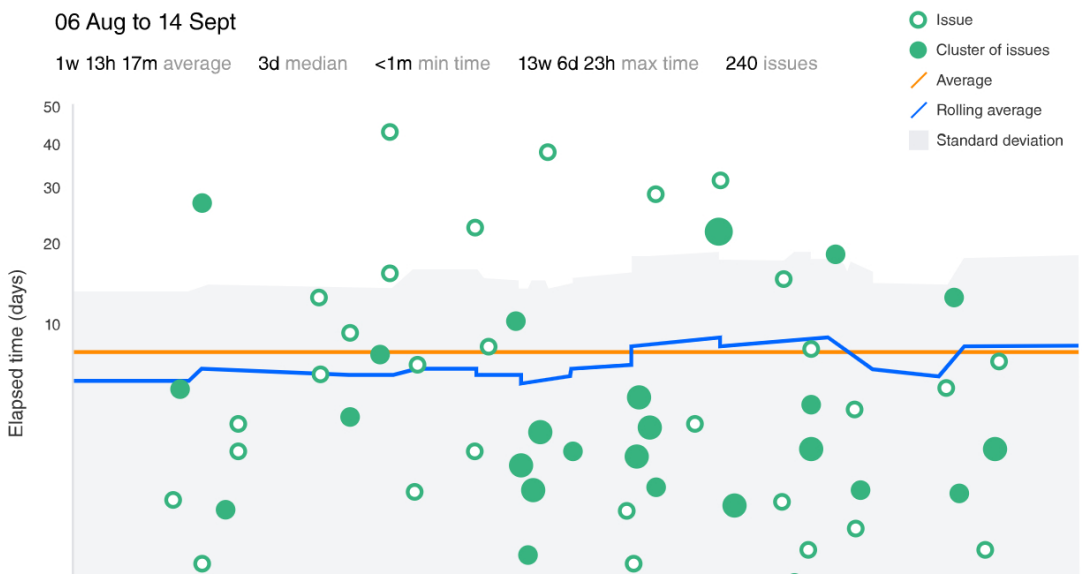
Visual metrics

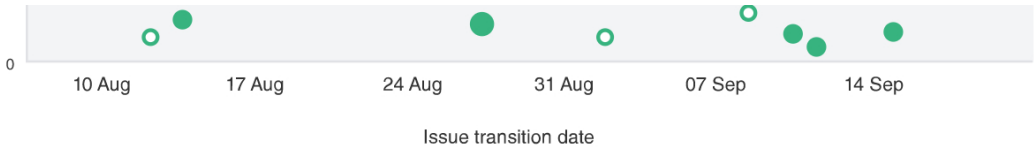
One of the core values is a strong focus on continually improving team efficiency and effectiveness with every iteration of work. Charts provide a visual mechanism for teams to ensure they're continuing to improve. When the team can see data, it's easier to spot bottlenecks in the process (and remove them). Two common reports kanban teams use are control charts and cumulative flow diagrams.

A control chart shows the cycle time for each issue as well as a rolling average for the team.

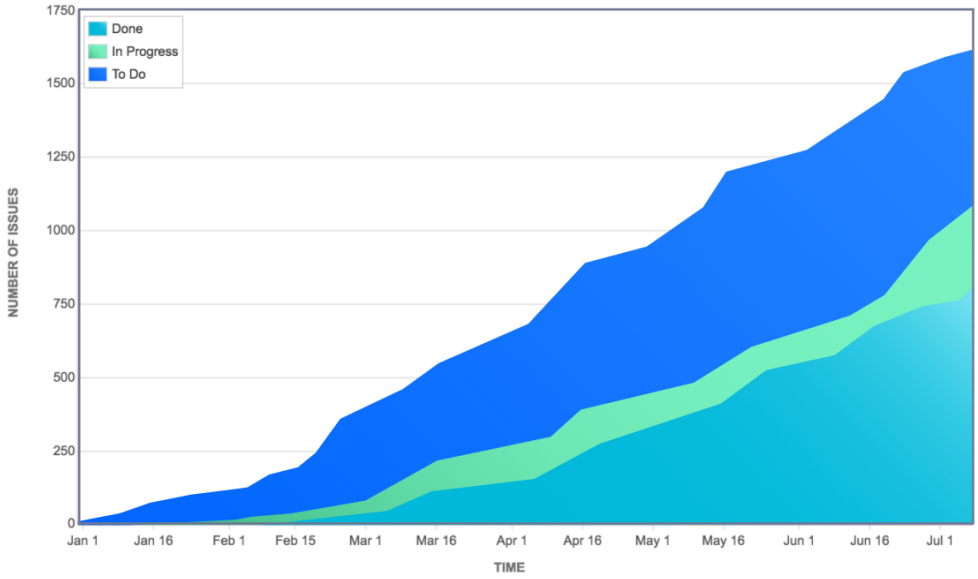
PRO TIP:

The team's goal is to reduce the amount of time an issue takes to move through the entire process. Seeing the average cycle time drop in the control chart is an indicator of success.





A cumulative flow diagram shows the number of issues in each state. The team can easily spot blockages by seeing the number of issues increase in any given state. Issues in intermediate states such as "In Progress" or "In Review" are not yet shipped to customers, and a blockage in these states can increase the likelihood of massive integration conflicts when the work does get merged upstream.



Continuous Delivery

Continuous delivery (CD) is the practice of releasing work to customers frequently. Continuous integration (CI) is the practice of automatically building and testing code incrementally throughout the day. Together they form a CI/CD pipeline that is essential for development teams (especially for DevOps teams) to ship software faster while ensuring high quality.

Kanban and CD beautifully complement each other because both techniques focus on the just-in-time (and one-at-a-time) delivery of value. The faster a team can deliver innovation to market, the more competitive their product will be in the marketplace. And kanban teams focus on precisely that: optimizing the flow of work out to customers

Scrum vs. kanban

Kanban and scrum share some of the same concepts but have very different approaches. They should not be confused with one another.

	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Key metrics	Velocity	Cycle time

Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

Some teams blend the ideals of kanban and scrum into "scrumban." They take fixed length sprints and roles from scrum and the focus on work in progress limits and cycle time from kanban. For teams just starting out with agile, however, we strongly recommend choosing one methodology or the other and running with it for a while. You can always get fancy later on.

DAN RADIGAN

Agile has had a huge impact on me both professionally and personally as I've learned the best experiences are agile, both in code and in life. You'll often find me at the intersection of technology, photography, and motorcycling.

TUTORIAL

Ready to get started?

Step-by-step instructions on how to drive a kanban project, prioritize your work, visualize your workflow, and minimize work-in-progress with Jira Software.

Read this tutorial →

UP NEXT

Agile design: process and guidelines for collaborative design

Collaborative design iterates on a product design by seeking the perspectives of your customers and developers at the outset of a project. Read more.

Read this article →

Sign up for more agile articles and tutorials.

Email

email@example.com

Subscribe