draw.io

Mermaid diagrams

A guide to Mermaid diagramming in draw.io
for Confluence and Jira Cloud

Contents

p.3 Mermaid: diagrams as code

p.4 The catch with text-to-diagram tools

p.5 Inserting a Mermaid diagram in draw.io

p.6 Diagram vs. Image: which to pick

p.7 What makes a diagram editable: the Mermaid container group

p.8 Using draw.io's AI feature to create Mermaid diagrams

p.9 Advantages of the Generate tool

p.9 Worth knowing

p.10 Cleaner layouts for complex flowcharts (ELK)

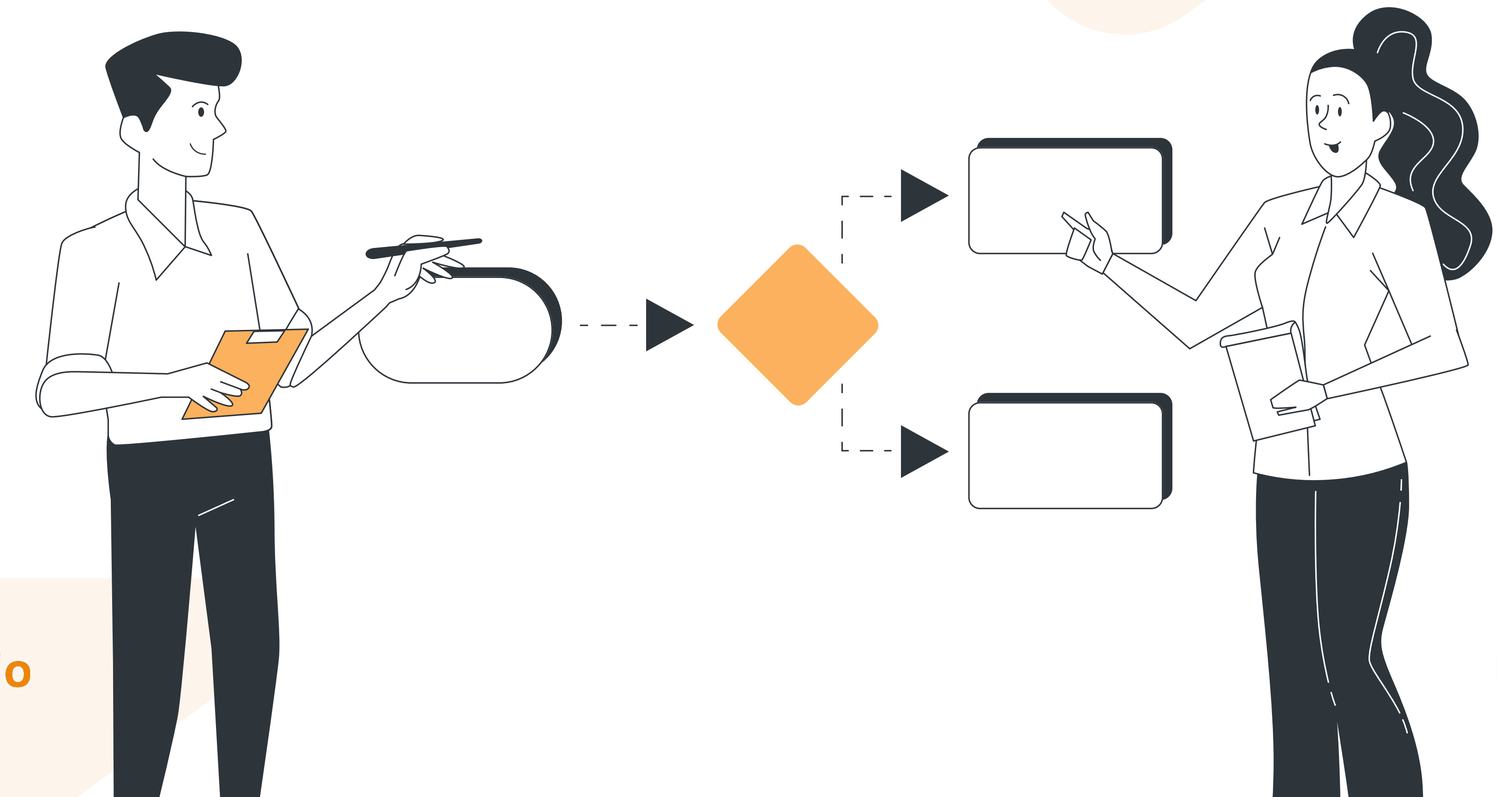
p.11 Done iterating? Convert to a standard draw.io diagram

p.12 Where your Mermaid diagrams live: Confluence and Jira Cloud

p.13 Mermaid diagram types: quick reference

p.13 Further resources

p.14 Try it yourself



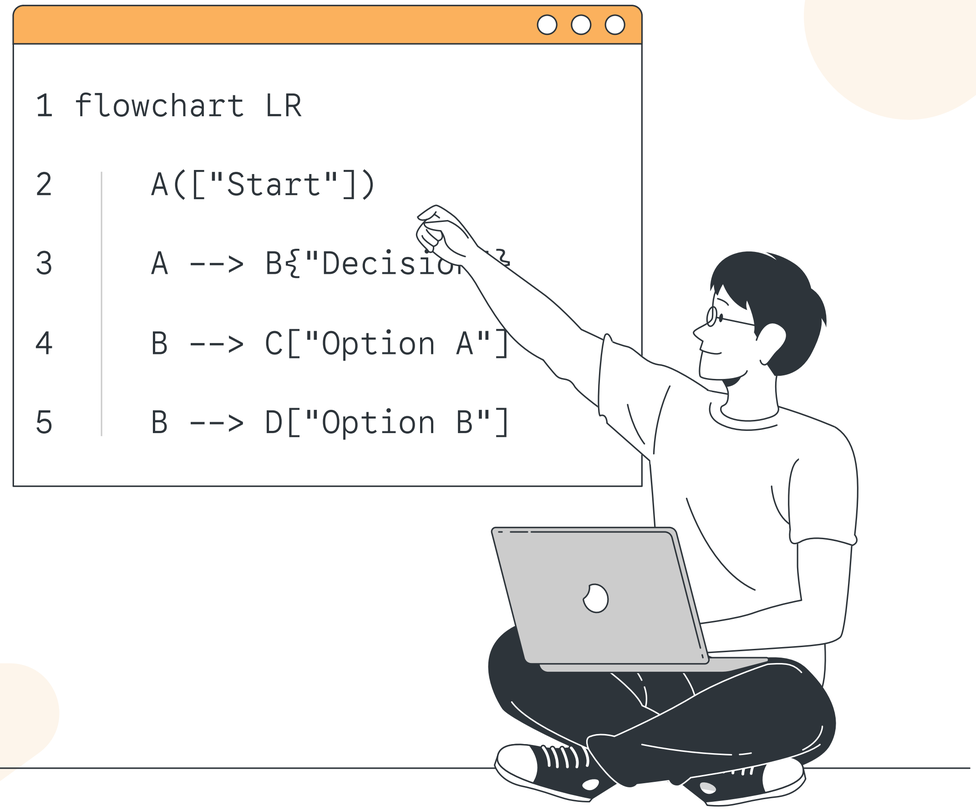
Mermaid: diagrams as code

Mermaid is a text-based syntax for describing diagrams: flowcharts, sequence diagrams, class diagrams, entity-relationship diagrams (ERD), Gantt charts, and more. Write a few lines of syntax, and Mermaid renders it as a diagram.

It's become a natural fit for engineering teams. Diagram descriptions live as plain text, so they sit comfortably in a README, a pull request, or a wiki page, and can be version-controlled and diffed like any other file. There's no need to open a separate visual tool mid-task.

Mermaid syntax has also picked up a second life recently. Many AI tools, including ChatGPT and Claude, will readily output Mermaid text when asked to describe a process or structure. Ask an AI to sketch out an architecture or a workflow, and there's a good chance you'll get Mermaid syntax back, ready to render.

But writing the syntax is only the first step. What happens once the diagram renders is where things get more interesting, and where a lot of Mermaid users run into the same handful of limitations. This whitepaper looks at those limitations and at how draw.io for Confluence and Jira Cloud handles them.



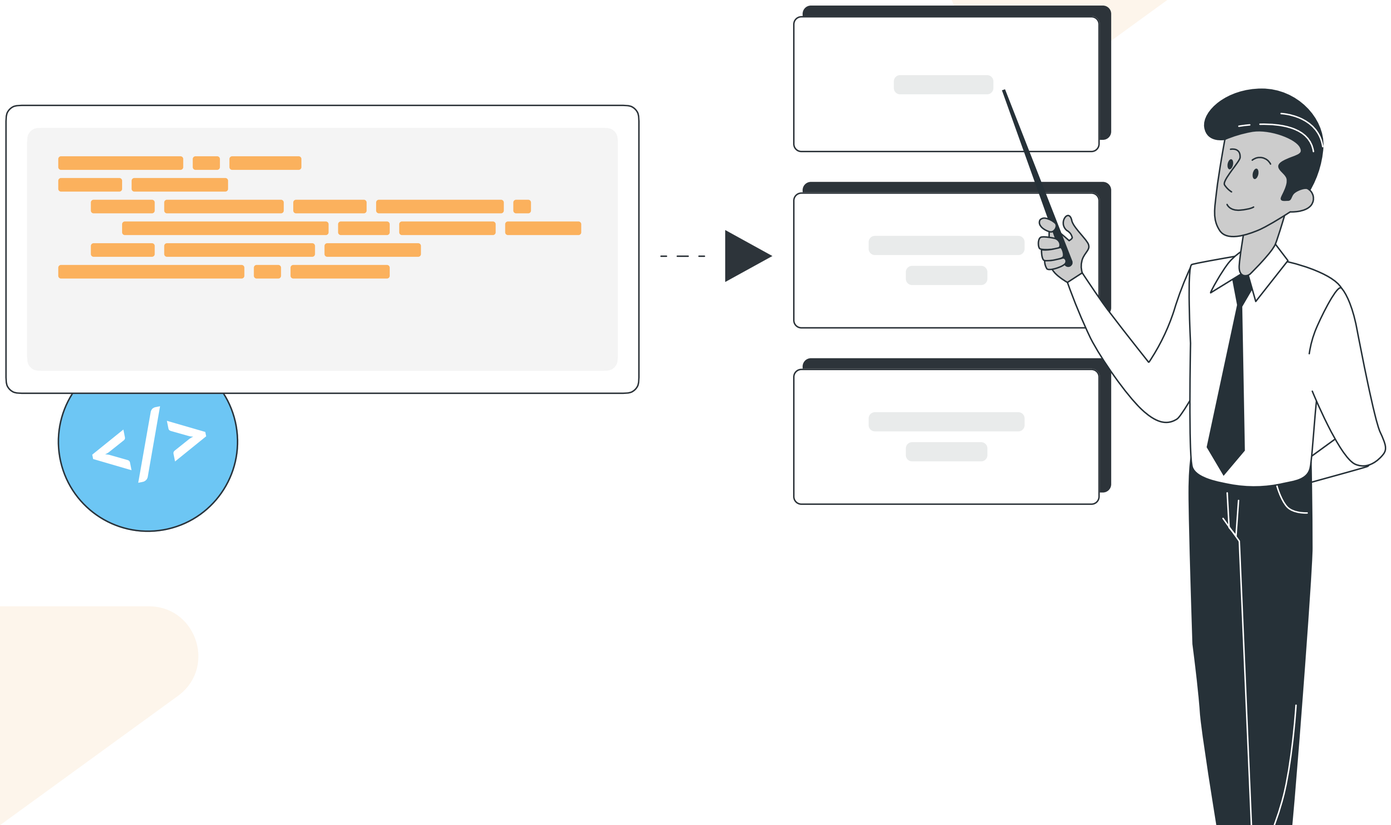
The catch with text-to-diagram tools

Text-based diagramming enables you to visualize fast. However, most Mermaid renderers give you one fixed output. So if a shape isn’t aligning correctly, a label runs too long, or you want to adjust the layout to make a relationship clearer, there's usually no direct way to fix it. Your only real lever is going back to the text, guessing what change might produce a better result, and re-rendering the whole thing from scratch.

This gets more noticeable as diagrams grow. A five-node flowchart rendered from Mermaid is usually fine as-is. A forty-node CI/CD pipeline with several conditional branches is a different story: connectors cross over each other, decision points end up far from the steps they lead to, and there's no way to just drag a shape three pixels to the left to clean it up.

There's also a collaboration gap. Mermaid diagrams commonly get embedded as static images. A teammate who wants to add a step, correct a detail, or restyle something for a presentation slide either needs to track down the original source text or recreate the diagram.

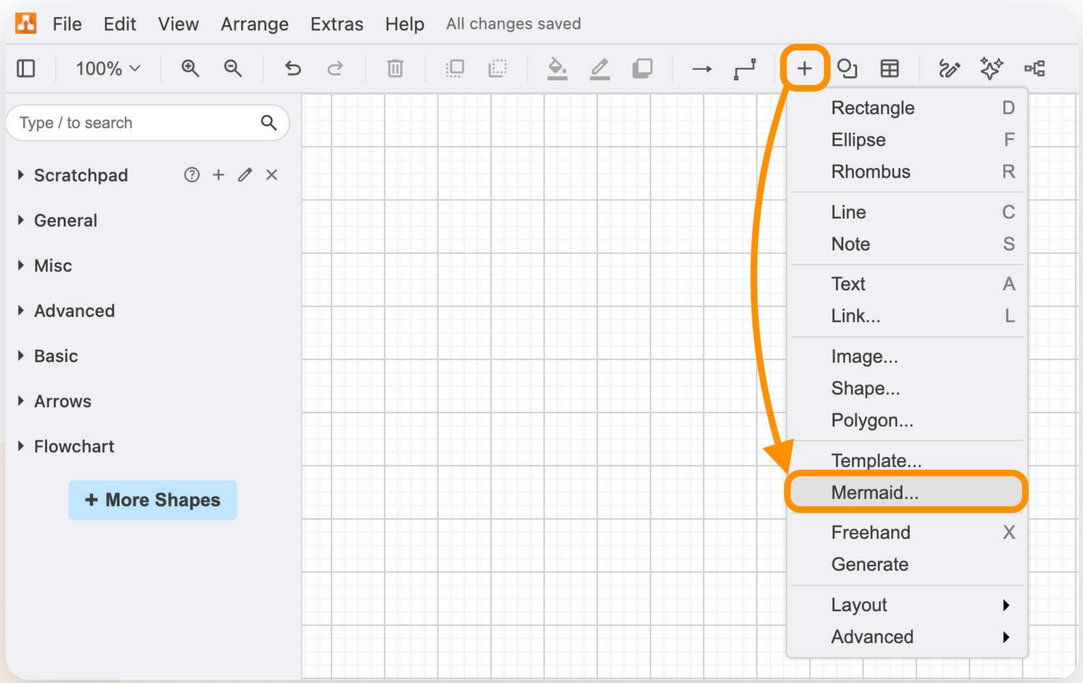
None of this is a knock on Mermaid itself. It's a genuinely good way to describe a diagram quickly. The gap is what happens after the syntax renders, and that's the part draw.io is built to close, turning the rendered diagram into something you can still reshape, restyle, and hand off, not just a fixed image.



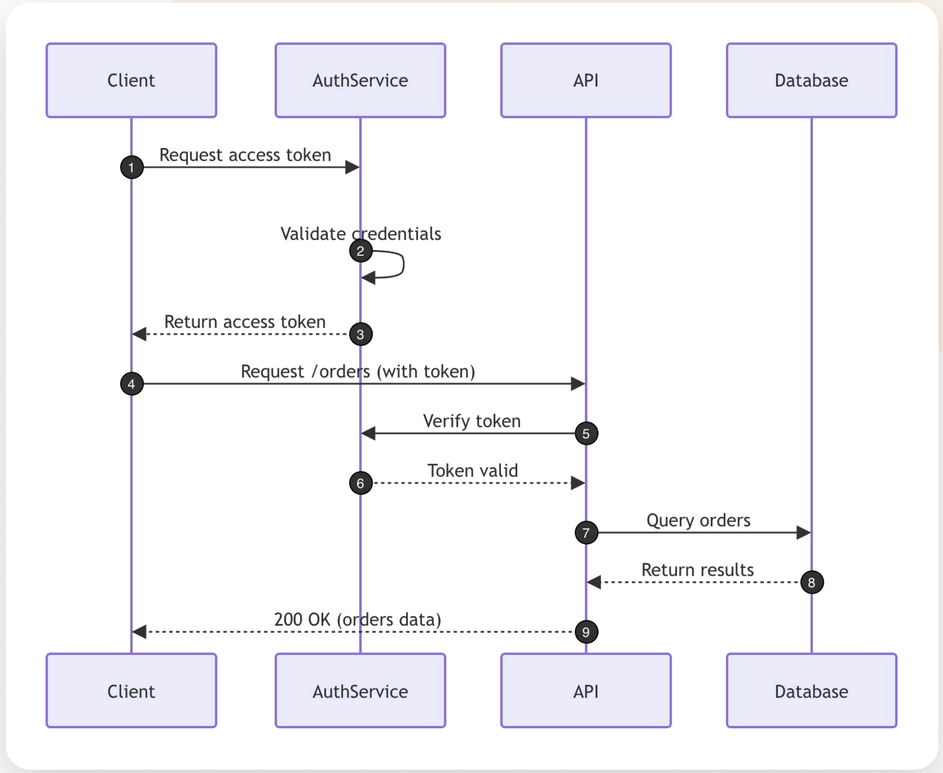
Inserting a Mermaid diagram in draw.io

To insert a Mermaid diagram in draw.io onto a Confluence page or a Jira ticket:

1. On a Confluence page, start typing */draw* and select **draw.io diagram**. If you are using Jira, in a ticket, click on the three dots (...) in the top right to go to More actions, and select **Add draw.io diagram**.
2. Open the Mermaid dialog by clicking the Plus (+) icon in the toolbar and selecting **Mermaid**.
3. Paste your Mermaid syntax into the text box and click **Insert**.



Here's a sequence diagram describing a client authenticating and requesting order data, inserted this way:



Before inserting your Mermaid diagram, there's one choice to make: whether the result should be a fully editable diagram or a fixed image - this is what we'll cover next.

Diagram vs. Image: which to pick

When you insert Mermaid syntax, a dropdown menu in the dialog lets you choose between two outputs: **Diagram** or **Image**.

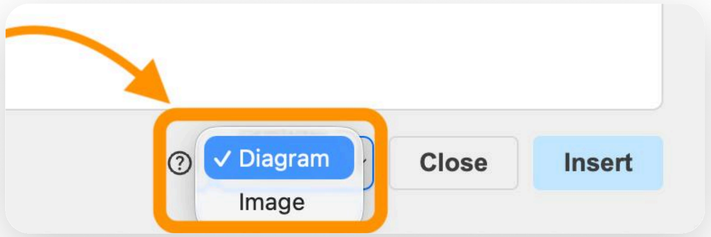
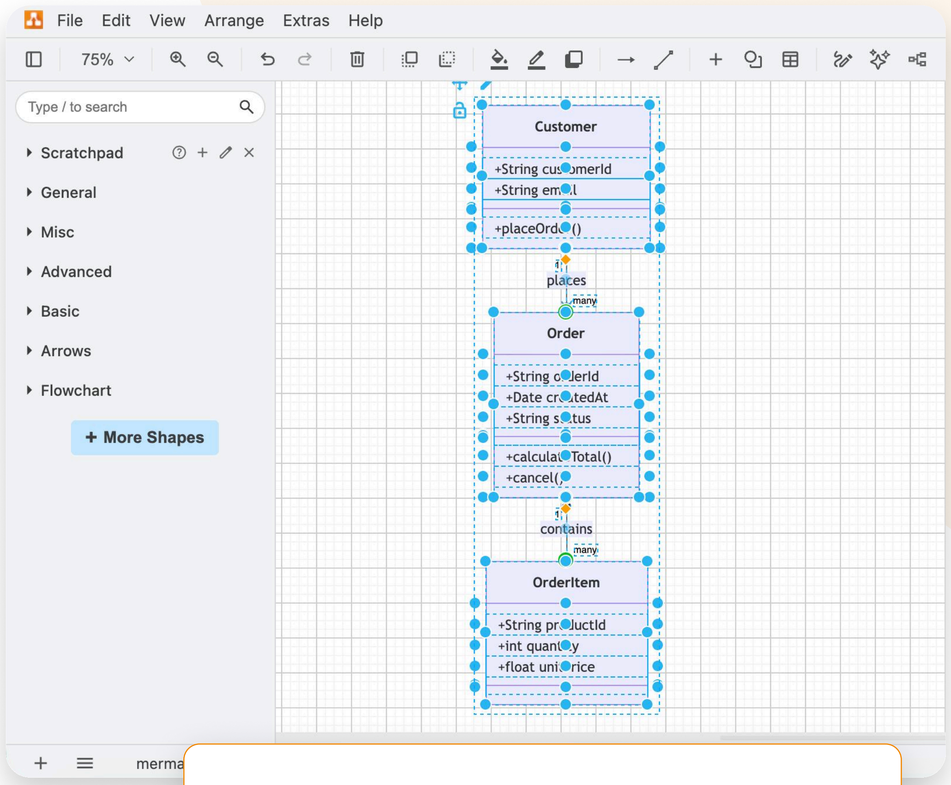


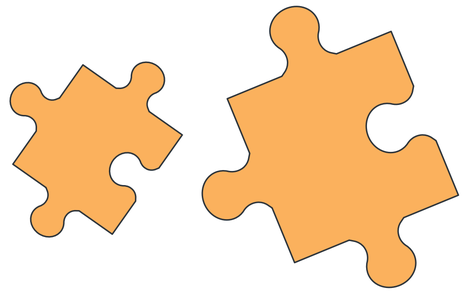
Diagram is the default. It converts the Mermaid syntax into native draw.io shapes and connectors, laid out automatically and placed inside a Mermaid container group. You can restyle it, and the Mermaid source stays attached so you can edit and regenerate it later. This is the option to pick if you expect you'll want to make future changes to the diagram again.

Image generates the diagram as a single SVG image instead. You can still double-click it to edit the underlying Mermaid text and regenerate a new version, but you can't select or restyle individual shapes within it. This is useful when you want a diagram to look exactly the same wherever it's viewed, without any risk of someone nudging a shape out of place.



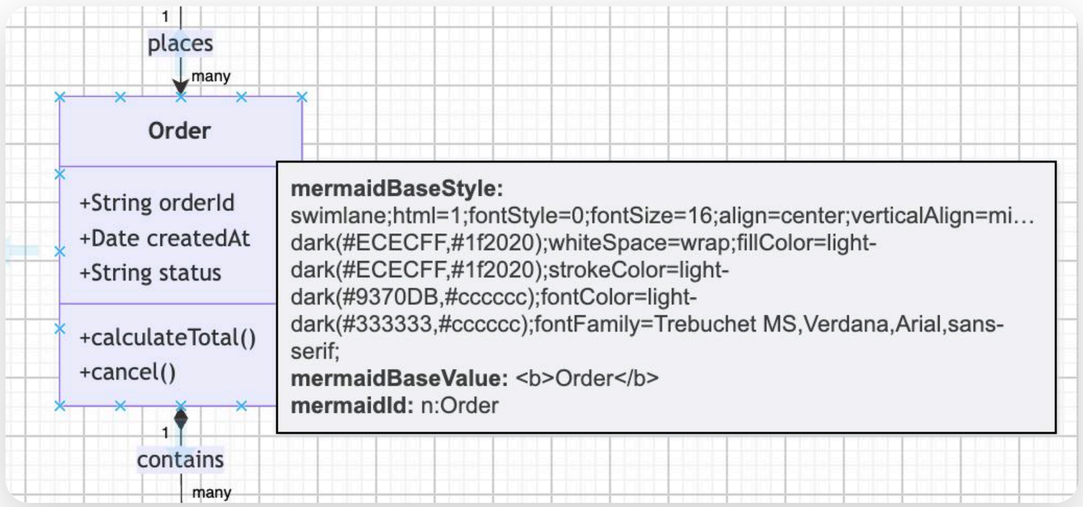
A class diagram generated with Mermaid syntax

Note: You can't switch a diagram from one option to the other after it's inserted. If you insert as an image and later decide you want editable shapes, you'll need to copy the Mermaid source and insert it again as a new diagram.



What makes a diagram editable: the Mermaid container group

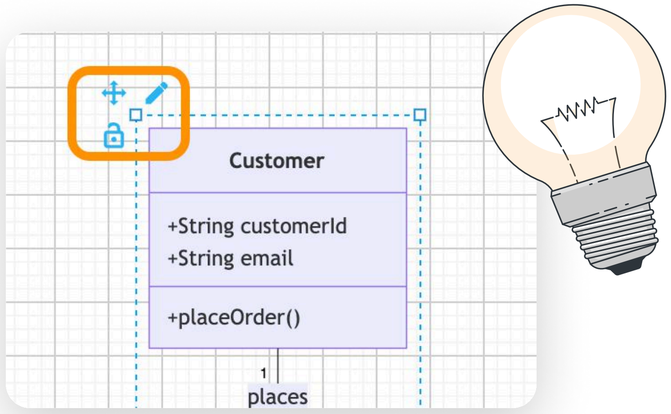
When you insert Mermaid syntax using the Diagram option, it's placed inside a **Mermaid container group**. This is a set of native draw.io shapes and connectors, generated from your syntax and automatically laid out. Hover over any shape or connector to see the Mermaid source behind it.



Taking the class diagram from the previous section as an example: once inserted, each class becomes its own shape, and you can select any of them to restyle it: changing fill colors, line colors, fonts, or connector styles through the format panel, the same way you would with any other draw.io shape.

These style changes aren't written into the Mermaid syntax itself. Instead, draw.io stores them separately and reapplies them automatically if you later edit the Mermaid source text and regenerate the diagram, matching each style to the shape it belonged to. **Note**: manually repositioning shapes doesn't get the same treatment: if you drag a shape somewhere or resize it by hand, that change is discarded on regeneration, since the layout is recalculated from scratch each time.

To go back and edit the source, select a shape in the group and click the pencil icon that appears. This reopens the Mermaid dialog, where you can revise the text and click **Apply**. If you want to lock a diagram in place to prevent accidental style or layout changes, there's a lock icon in the same location.



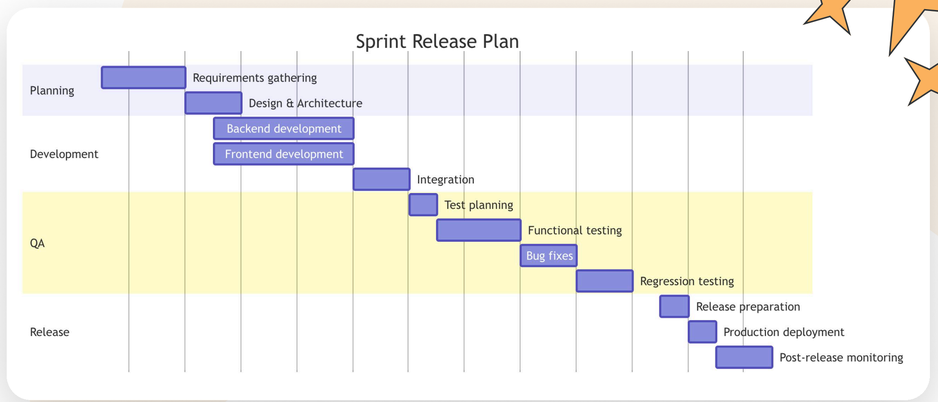
Using draw.io's AI feature to create Mermaid diagrams

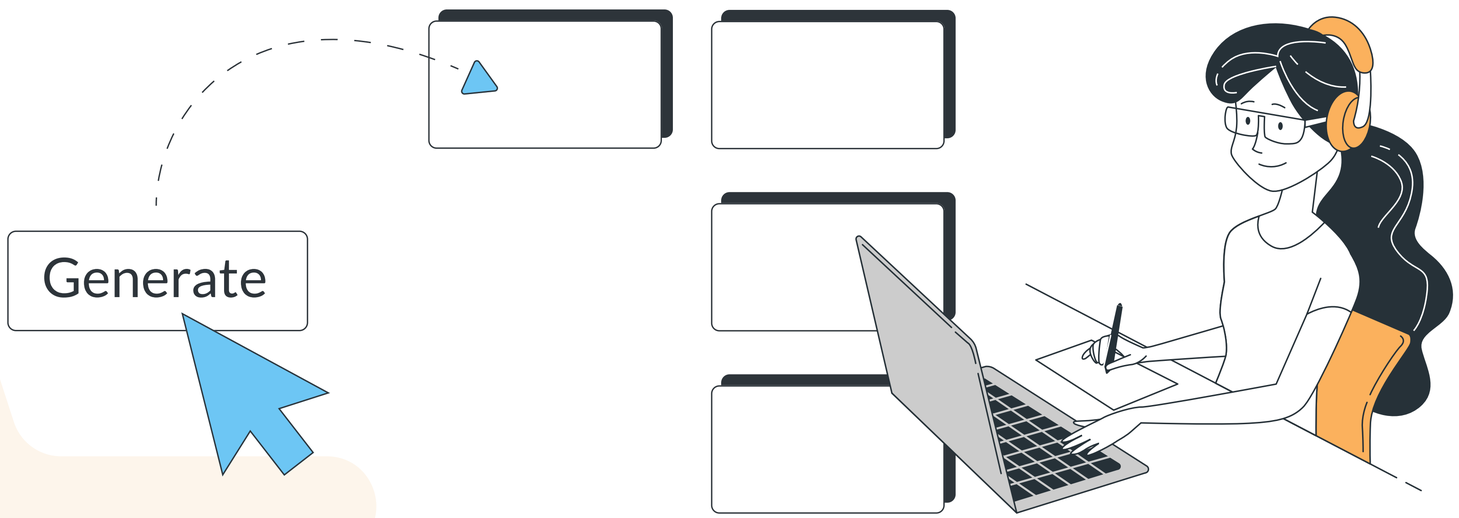
There's less need to learn Mermaid's syntax rules than there used to be. Users are increasingly turning to AI to generate code for them, Mermaid included.

The draw.io editor has its own built-in AI feature for this: **the Generate tool**, accessed via the sparkle icon ✨ in the toolbar. Instead of writing Mermaid syntax yourself, you describe the diagram you want in plain language, and the Generate tool produces it for you.

For some diagram types like Gantt charts, the result will be generated as Mermaid code. This renders the same way as any other Mermaid diagram: hover to see the source, double-click to edit it, and regenerate as needed.

Here's a Gantt chart produced using the Generate tool, based on the prompt *"Gantt chart for a sprint release plan with planning, development, QA, and release phases"*:





Advantages of the Generate tool

- **Regenerate as needed**: If you're unhappy with the initial result, you can edit the prompt and regenerate the diagram as many times as required.

- **Everything stays in the editor**. With a standalone LLM, you write a prompt, get Mermaid text back, then copy it into draw.io's Mermaid dialog yourself. The Generate tool skips that round trip: describing the diagram and getting an editable result happens in the same editor.

- **Directly editable, not just text.** The output lands in the same container-group workflow as any other Mermaid diagram: hover to see the source, double-click to edit it, restyle through the format panel.

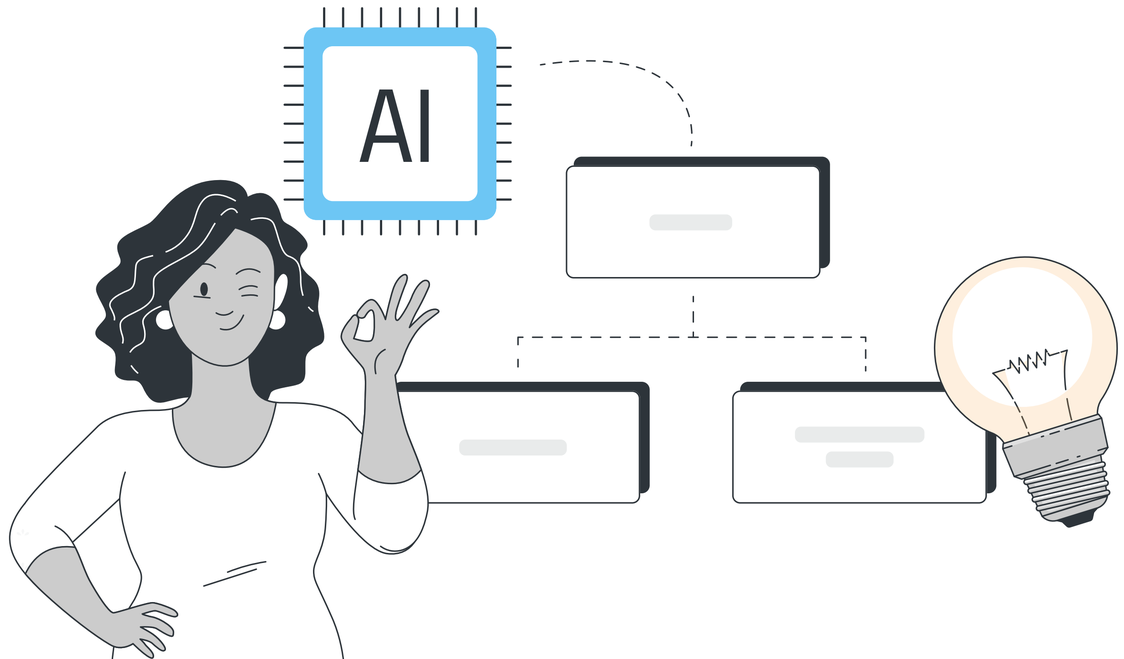
- **Existing diagrams can be included**. The plus (+) option in the Generate dialog lets you attach shapes, a page, or a file, so the AI can build on something already on the canvas rather than starting from a blank prompt.

Worth knowing

Your diagram data itself is **never** stored on draw.io's servers when using the Generate tool. It does however involve cross-domain traffic, since your prompt text and the resulting response are seen by the AI service producing the diagram. Your existing diagram data isn't included in that by default. It only gets shared if you explicitly attach shapes, a page, or the whole file to your query using the Plus (+) option in the Generate dialog, and even then, it's up to the AI service whether it factors that attachment in.

draw.io's AI Generate tool is **disabled** by default. Administrators need to explicitly enable it before anyone in their organization can use it.

If your organization works with highly sensitive data and wants to rule out this kind of data transmission entirely, the **draw.io Zero Egress app** removes cross-domain data transmission altogether, or the **draw.io Advanced app edition** includes a lockdown switch that lets administrators restrict all data egress except for Atlassian API calls.



Cleaner layouts for complex flowcharts (ELK)

By default, Mermaid's graph-based diagram types, flowchart, class, state, entity-relationship, and requirement, are laid out using **Dagre**, the same layout engine Mermaid uses internally. For most diagrams, this works well. For larger or more deeply nested graphs, connectors can end up crossing over each other or routing in ways that make the diagram harder to follow at a glance.

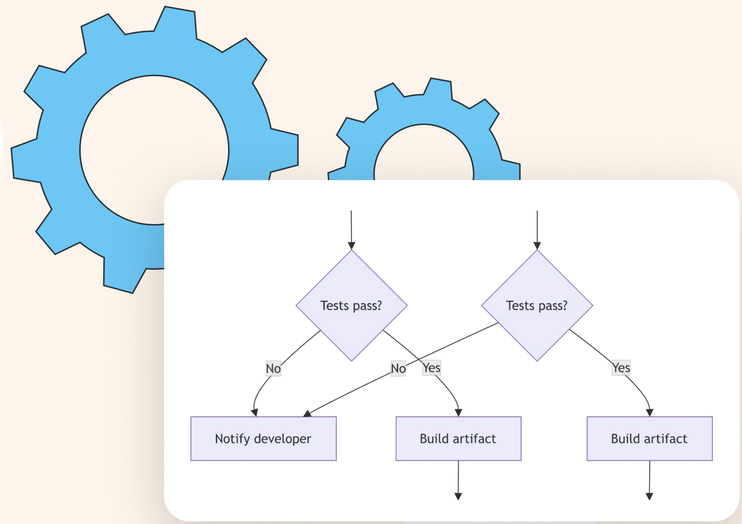
draw.io also supports **ELK** as an alternative layout engine. ELK is aimed specifically at large or complex graphs, and can produce neater orthogonal connector paths and tighter shape spacing than the default.

To use it, add an `init` directive to the top of your Mermaid source. For flowcharts specifically:

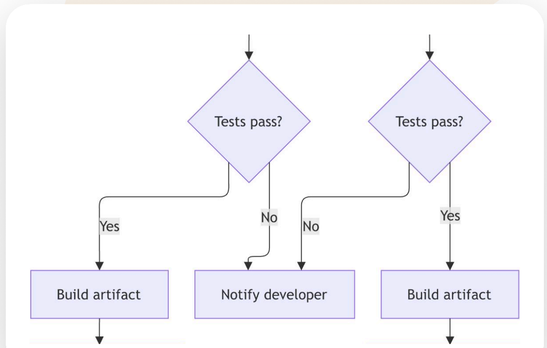
```
%%{init: {"flowchart": {"defaultRenderer": "elk"}} }%%
```

Other graph types can specify ELK using a `layout` key instead, either as an init directive or as YAML front matter at the top of the source. All other diagram types, sequence, Gantt, pie, mindmap, and so on, use their own dedicated layouts and ignore ELK entirely. If ELK isn't available or can't lay out a particular graph, draw.io falls back to Dagre automatically.

Here's the same flowchart, describing a CI/CD pipeline with several conditional branches, rendered both ways:



CI/CD pipeline, default Dagre layout



Same flowchart, with the ELK init directive added

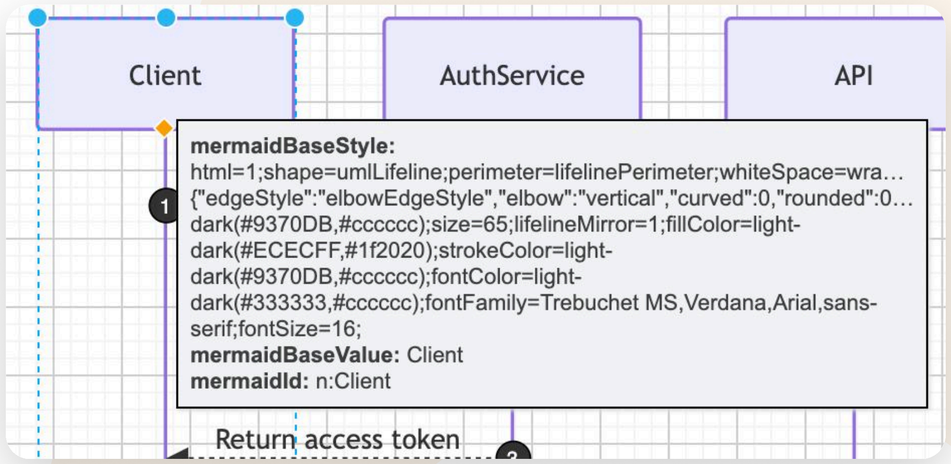
With Dagre, the two "No" branches cross each other on their way to "Notify developer." With ELK, those connections route around each other instead using distinct orthogonal paths, at the cost of a taller, narrower overall layout. Which is preferable will depend on the diagram and where it's being used.

Done iterating? Convert to a standard draw.io diagram

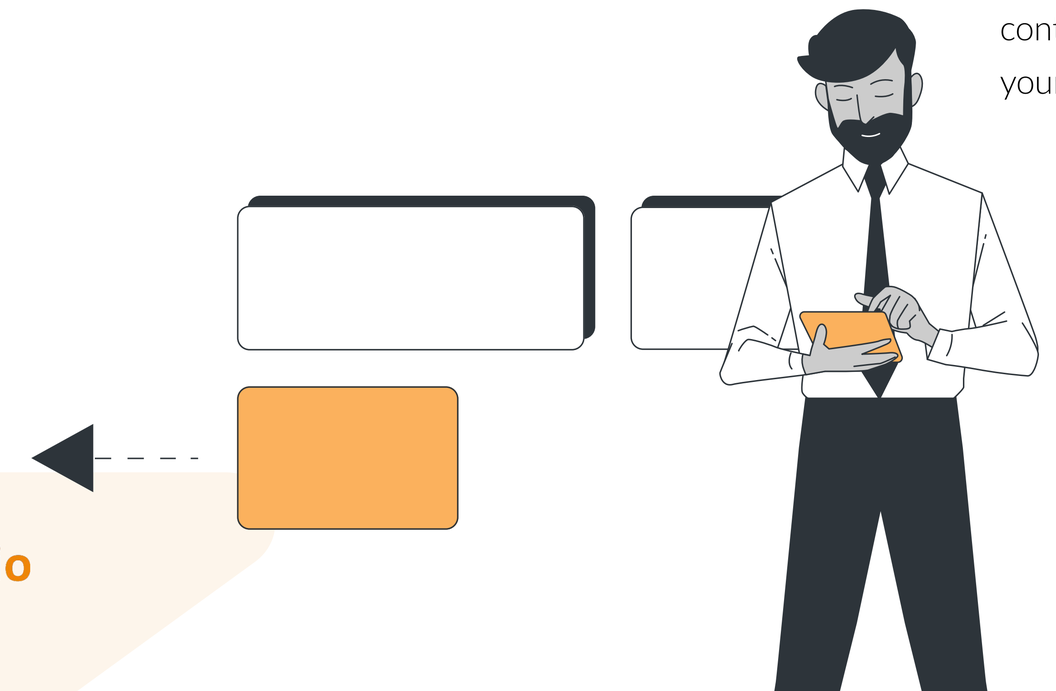
At some point, a diagram might not need to stay connected to its Mermaid source anymore. Maybe it's finished, or maybe you want to hand it off to someone who'll only ever make standard edits that don't require Mermaid code (e.g., changing shapes, colors, fonts, etc.). In that case, you can convert the Mermaid container group into a regular draw.io diagram.

To do this, right-click the container group and select **Ungroup** from the context menu.

This deletes the Mermaid source that was stored in the container group. Any styling already applied to individual shapes and connectors stays intact, and you can view this by clicking on and hovering over a shape to view the style metadata, but the diagram can no longer be reopened and regenerated from text.

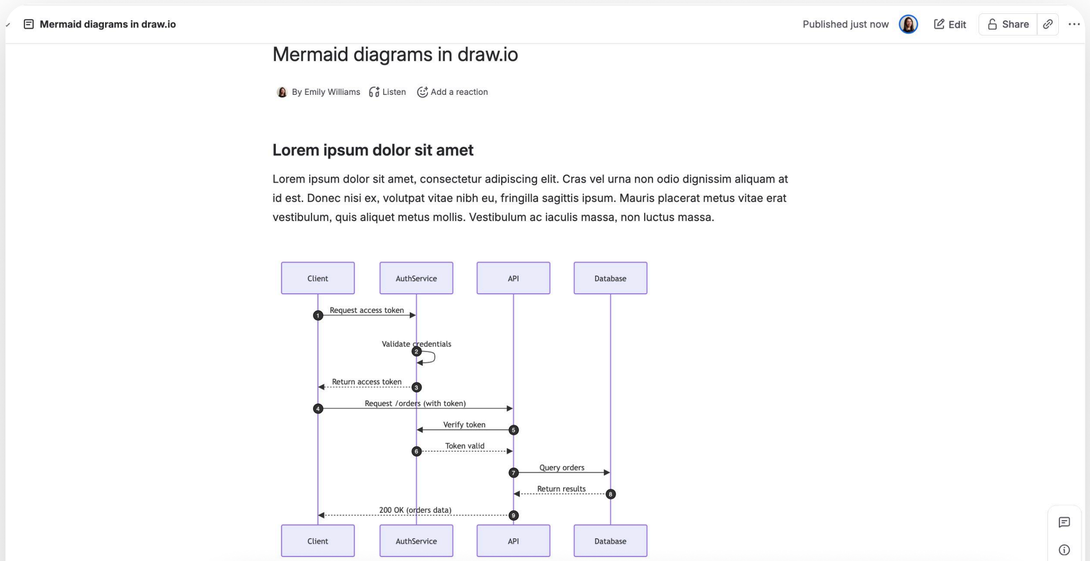


If you do this by mistake, or change your mind, *Ctrl+Z* (*Cmd+Z* on Mac) will undo it and recover the Mermaid source, as will the undo option in the toolbar. Reverting to an earlier version through **draw.io's built-in Revision History** will also restore the container group and its Mermaid source, provided you saved your diagram before ungrouping the container shape.



Where your Mermaid diagrams live: Confluence and Jira Cloud

Everything covered so far in this whitepaper happens inside your source of truth: Confluence or Jira, not a separate tool bolted on alongside them. A Mermaid diagram lives on the same page or ticket it relates to, visible to anyone who can already see that page or ticket.



A Mermaid diagram embedded on a Confluence page

Access follows Confluence and Jira's own **permission settings** directly. There's no separate login, license, or permission layer to manage on top of what's already in place for the page or ticket itself.

Edits to a diagram, including regenerating it from updated Mermaid source, are captured in the page or ticket's version history, so you can review past versions in both Confluence and Jira. Reverting to a previous version is available through Confluence's version history; Jira doesn't currently support this for tickets.

Diagram data is stored as an attachment on the page or ticket rather than on draw.io's servers. Exceptions to this include:

- **The draw.io AI Generate tool:** As mentioned before, this sends your prompt text to a third-party AI service to produce a diagram.
- **Real-time collaboration:** Data is sent securely to draw.io's server endpoints; once it's successfully returned, it's deleted from our servers, with nothing persisted.

Organizations with stricter requirements can lock this down further with the **draw.io Zero Egress app** or the **Advanced app edition's** lockdown switch.

Diagrams needed for slides, a PDF handout, external reviews, etc. can be exported to PNG, SVG, PDF, and **other formats**.

Mermaid diagram types: quick reference

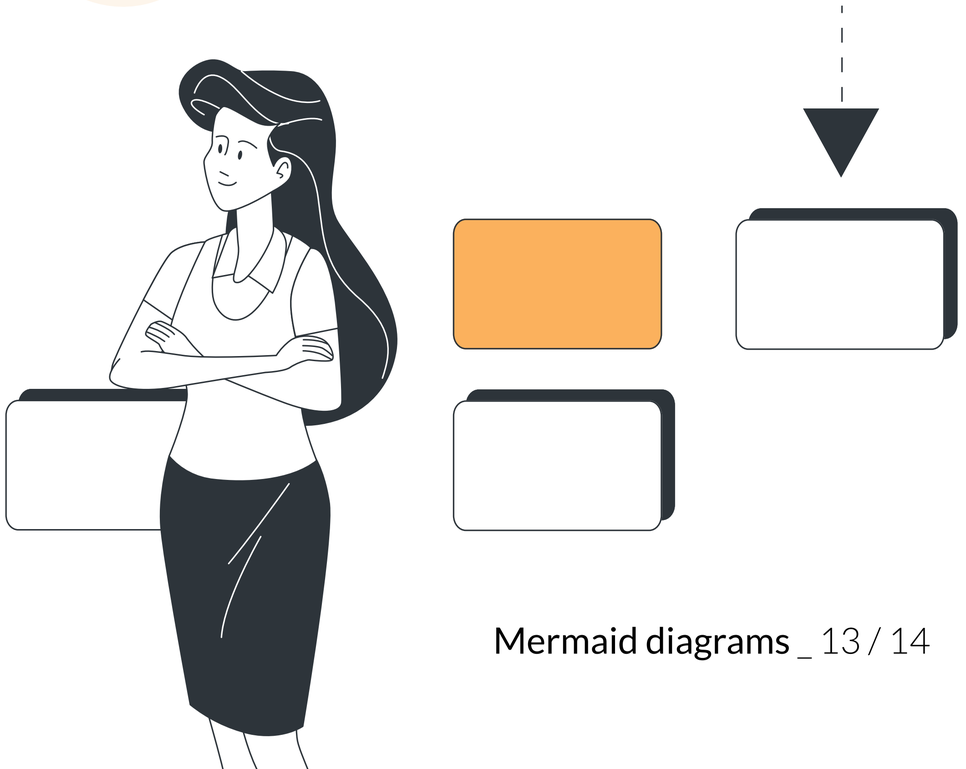
draw.io supports a wide range of Mermaid diagram types, each suited to a different kind of information.

If you need to show...	Use...
Steps, decisions, and process flow	Flowcharts
Message flow between participants over time	Sequence diagrams
Object structure and relationships	Class diagrams
Object lifecycle and transitions	State diagrams
Entities and their relationships in a data model	Entity-relationship diagrams (ERD)
Tasks, timelines, and dependencies	Gantt charts
Proportional breakdowns of a whole	Pie charts
Hierarchical ideas branching from a central topic	Mind maps
Git branching and commit history	Gitgraphs
System context and architecture	C4 diagrams

Each diagram type has its own syntax rules and supported shapes; refer to the Mermaid documentation for the complete syntax and styling options for a given type.

Further resources

- Check out Mermaid's diagram syntax page for the complete syntax and styling options for any diagram type
- Resources covering insertion, editing, and the container group in more detail:
  - Create Mermaid diagrams in draw.io for Confluence Cloud
  - Insert and edit a Mermaid diagram
  - Editable Mermaid diagrams
- Resources on the ELK layout engine:
  - Set the Mermaid layout engine (ELK)
  - Mermaid in draw.io updated to support ELK layout
- The draw.io Generate tool, for AI-assisted diagram creation beyond Mermaid



Try it yourself

Mermaid enabled engineers to write diagrams the same way they write everything else: as text, fast, without switching tools.

draw.io for Confluence and Jira Cloud picks up from there. The syntax renders in seconds, but the diagram doesn't stop being yours once it's on the page. Restyle it, fix a layout, hand it to a teammate (even one who's never touched Mermaid), or regenerate it from scratch a week later. It stays as a secure copy, exactly where your documentation already lives, ready to be customized, shared, collaborated on, exported, and more.

Start a free **30-day trial today**, or **book a demo with our team** to see it in practice.

