EXPLAINING FEATURE FLAG DEVOPS WITH DIAGRAMS

Feature-based development and release cycles are becoming more common in cloud native software services (SaaS products), where continuous development and continuous improvement are important.

Instead of the long development time needed to implement large changes in a software product, the development cycle for SaaS is shorter and more focused, even more so than the processes used by agile software development teams.

The leaner and faster development process is underpinned by feature flags and releases small updates often. It incorporates user feedback constantly through built-in feedback loops that involve active and stakeholder customers.

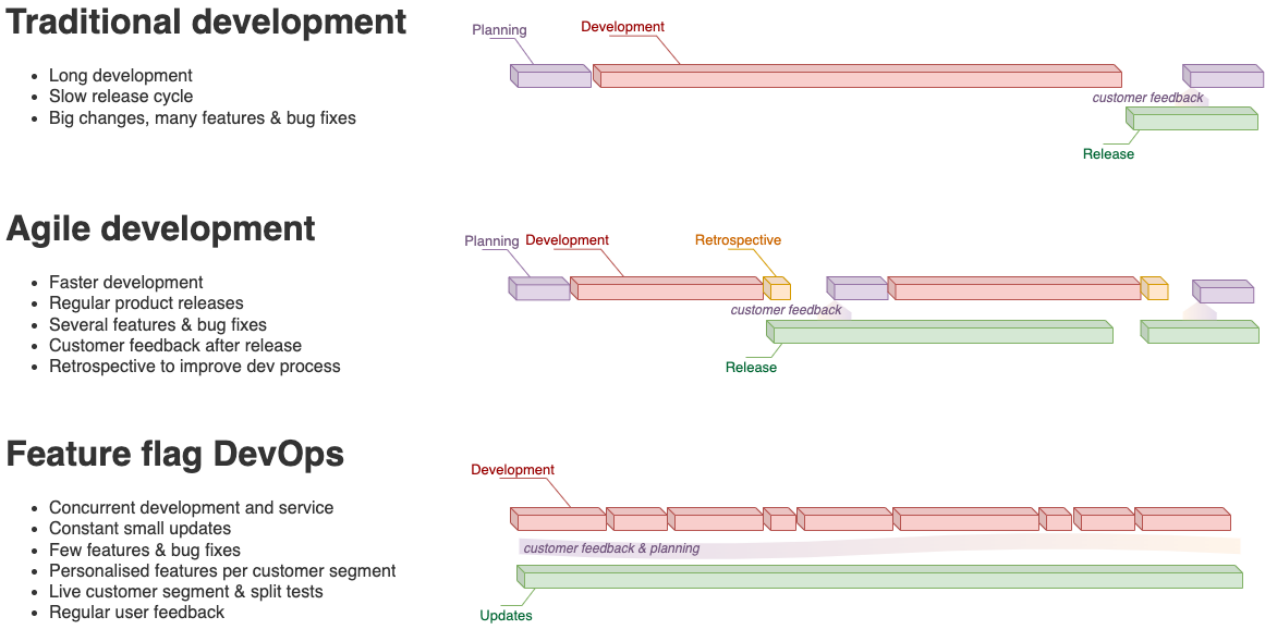

DIAGRAM 1: DIFFERENCE BETWEEN TRADITIONAL, AGILE AND FEATURE FLAG DEVELOPMENT PROCESSES

WHAT DO FEATURE FLAGS DO?

Feature flags, also known as feature toggles or feature controls, give administrators and developers fine granular control over which features are available to specific customer segments, essentially extending development and testing into the operation and deployment of the software.

With feature flags, you can more easily test updates with real users on live systems. You can also provide personalised features, enabling or disabling functionality for each customer segment to better meet their needs.

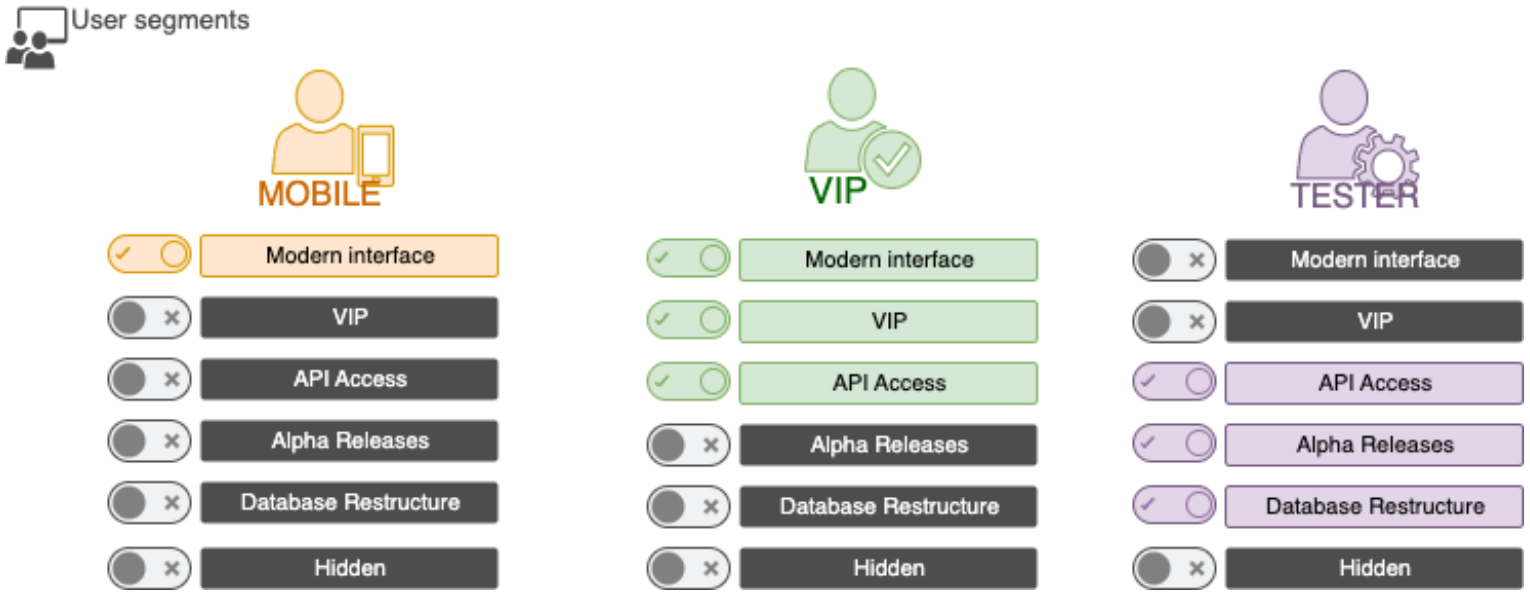

DIAGRAM 2: HOW FEATURE FLAGS TOGGLE SOFTWARE FUNCTIONALITY FOR DIFFERENT USER GROUPS

GRANULAR CONTROL AFTER DEPLOYMENT

Feature flags allow you to target users in defined segments and customise how they can see and use your SaaS product.

- **Deploy but hide** features that aren't yet ready, test dependencies and prepare intertwined systems for future updates.
- **Separate frontend and backend** by using one feature flag category for UI component visibility and another to control APIs and configurations.
- **Split test** (A/B test) two versions of a feature or interface to see which is more popular with users.
- **Block users** from accessing certain features.
- **Roll back** an unstable update easily by disabling the problematic feature flag.
- **Allow users to opt-in** to new features as they are released or opt-out and personalise which features they want to see.
- **Soft launch** a new feature to a small group and progressively expand access using multivariate feature flags for ease of split testing, gathering feedback and mitigating risks.

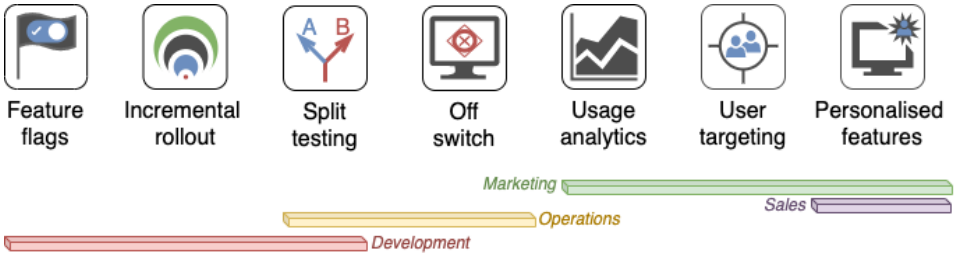

DIAGRAM 4: COMMON TEAMS AND HOW THEY CAN USE FEATURE FLAGS

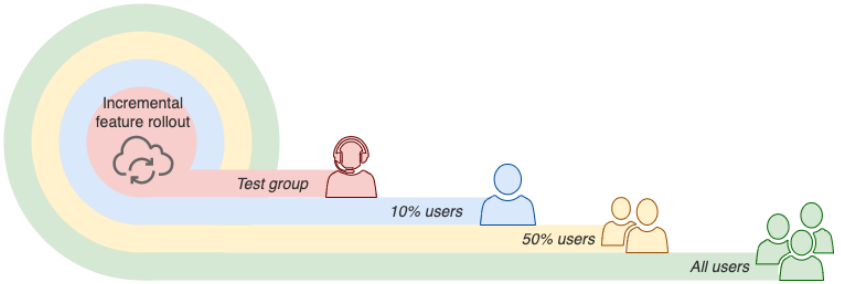

DIAGRAM 3: INCREMENTALLY DELIVER NEW FEATURES TO A PERCENTAGE OF USERS

The reasons why feature flags are useful in many situations and for many different teams are clear. However, it's a little harder to explain how the development structure must change.

GITFLOWS TO VISUALISE DEVELOPMENT STRUCTURE

In both agile and traditional software development, teams typically work on separate feature branches. Once the feature development is done, these branches are merged into a main development or nightly build branch. Before release, a candidate build needs to pass various tests, and then the software update is released to customers.

Several new features or updates to fix bugs are usually combined into the one release, as you can see in the gitflow diagram below.

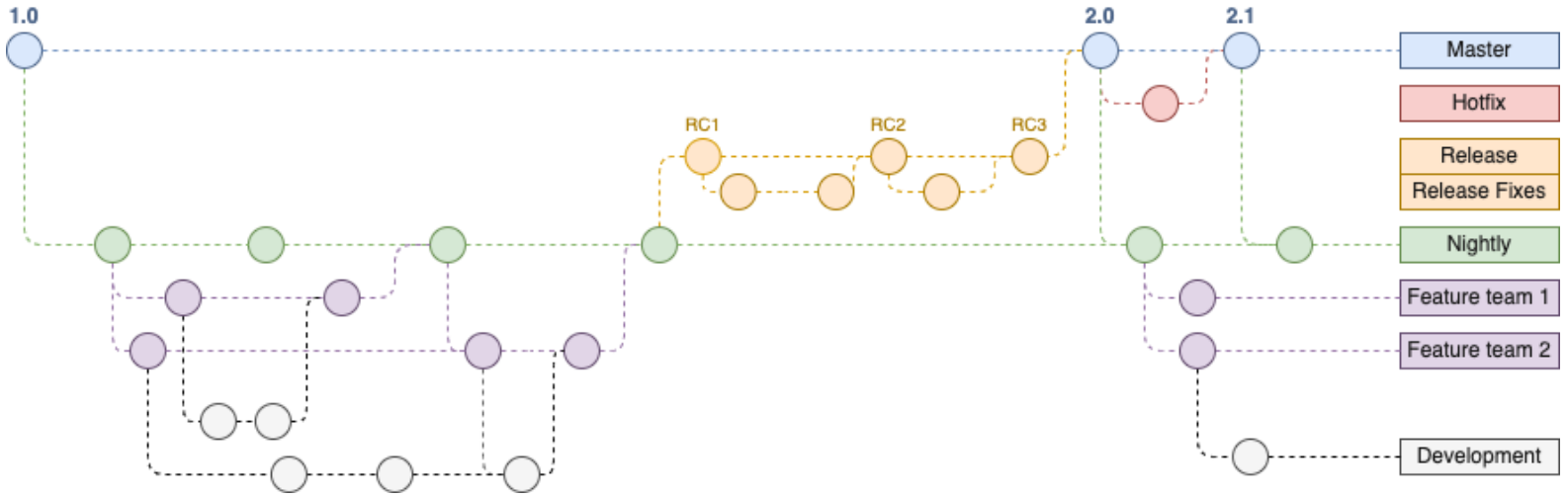

DIAGRAM 5: AN AGILE DEVELOPMENT GITFLOW USING FEATURE DEVELOPMENT BRANCHES, A NIGHTLY BUILD BRANCH AND RELEASE CANDIDATE BUILDS

SaaS and feature flag DevOps can use this git branching structure, but as each update contains a minimal number of features and bug-fixes, an extra branch just for that release adds too much overhead. Instead, development branches hang directly off the main release branch for a more streamlined development process.

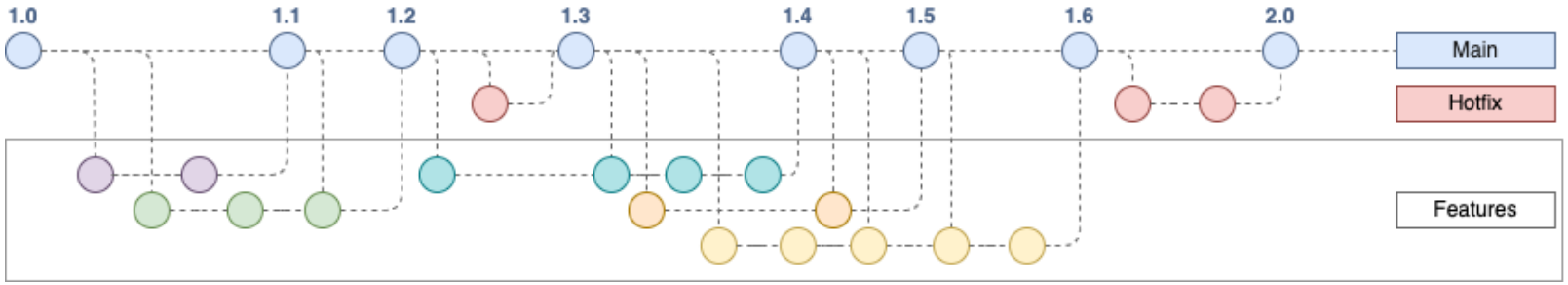

DIAGRAM 6: A STREAMLINED TRUNK-BASED SAAS AND FEATURE FLAG DEVOPS GITFLOW

Without indicating the feature flags that are applied to the deployed software, this gitflow shows only half of the story.

With diagrams.net, there are many ways to show feature flags in a diagram - tags, tooltips, and shape metadata can be used to explain how feature flags apply to each release or development branch. Tags are particularly useful for training documentation, as you can interactively display or hide shapes with specific tags in the diagram and see how the functionality is affected when those feature flags are enabled or disabled on the deployed software.

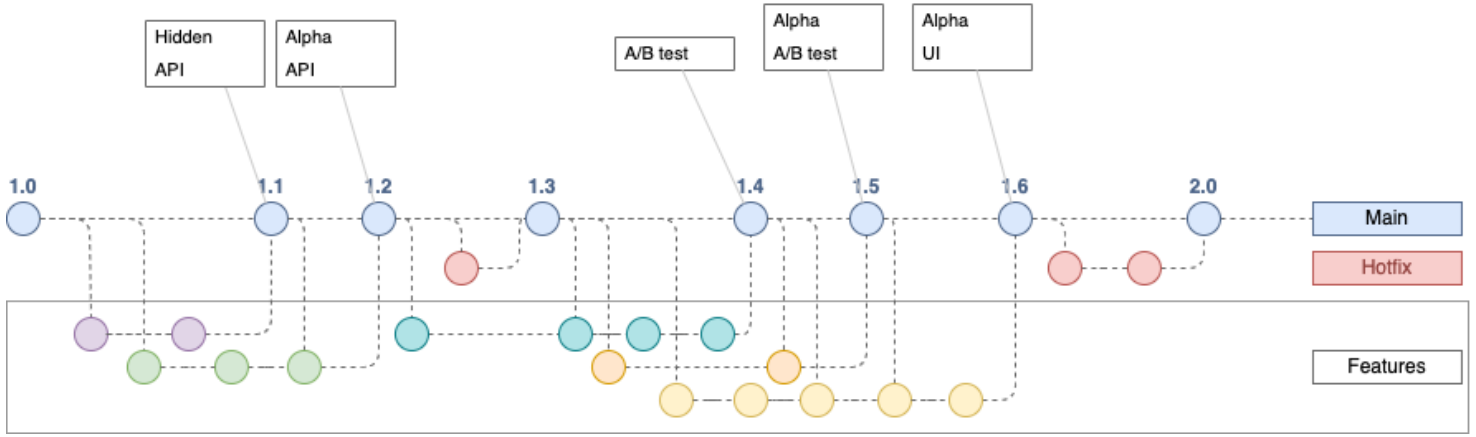

DIAGRAM 7: CALLOUTS USING LIST SHAPES ON RELEASES ARE THE CLEAREST WAY TO SHOW FEATURE FLAGS IN STATIC IMAGES

IMPLEMENTING FEATURE FLAGS IN DEPLOYMENT AND DEVELOPMENT

A feature flag system or feature toggle system works in two parts. One stores whether a feature is to be enabled or not, and the other part checks this flag configuration state whenever it is requested by the deployed system. If a feature is enabled for the customer currently using the system, allow whatever it is toggling. If it is disabled, hide it from the user.

In the code, you need to define each feature flag and wrap its affected code segments with conditional statements.

1. Define the feature flags in a single location.

   This could be in a well-structured XML file or in data structures in the code.

   Regularly delete any old feature flags that are no longer used to minimise the complexity of this file.

2. During development, wrap necessary code sections in the feature flag conditional so they can be enabled or disabled via the control interface. Not every code change will require a feature flag.

```
if (featureFlag) {
   // Run this code block if enabled
   else {
      // Run this code block if disabled
   }
}
```

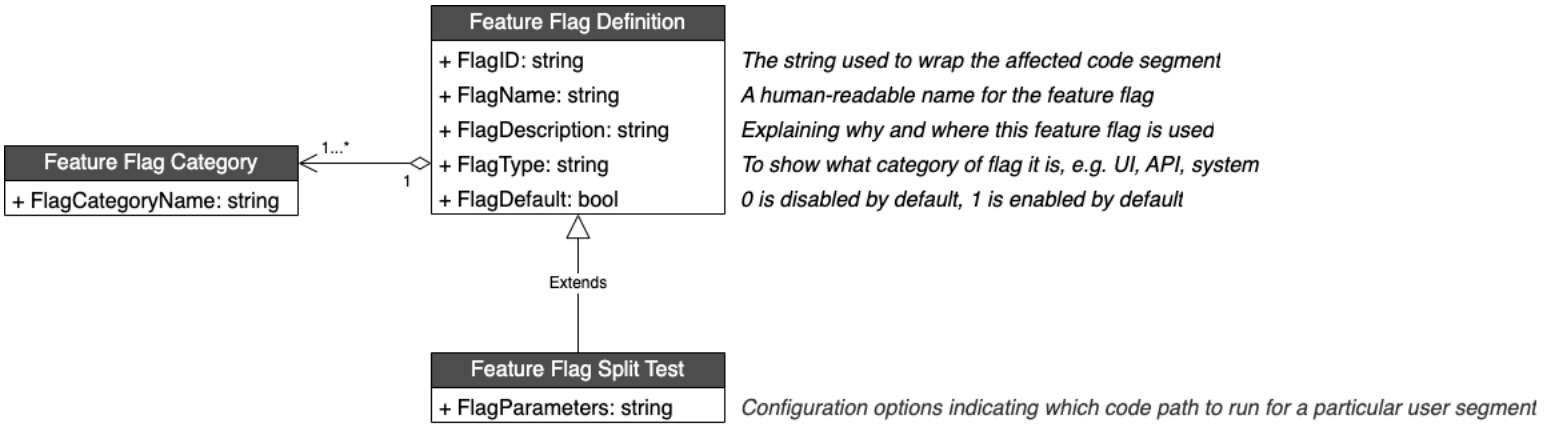

DIAGRAM 8: A UML CLASS DIAGRAM TO DEFINE THE DATA STRUCTURE IN A FEATURE FLAG SYSTEM

In the deployed system, a boolean check will decide whether a specific code segment is to be executed or not. You'll need an accessible way to control which feature flags are enabled and disabled, via a control panel or another mechanism. This front end determines which code paths are executed for specific customer segments in the running system.

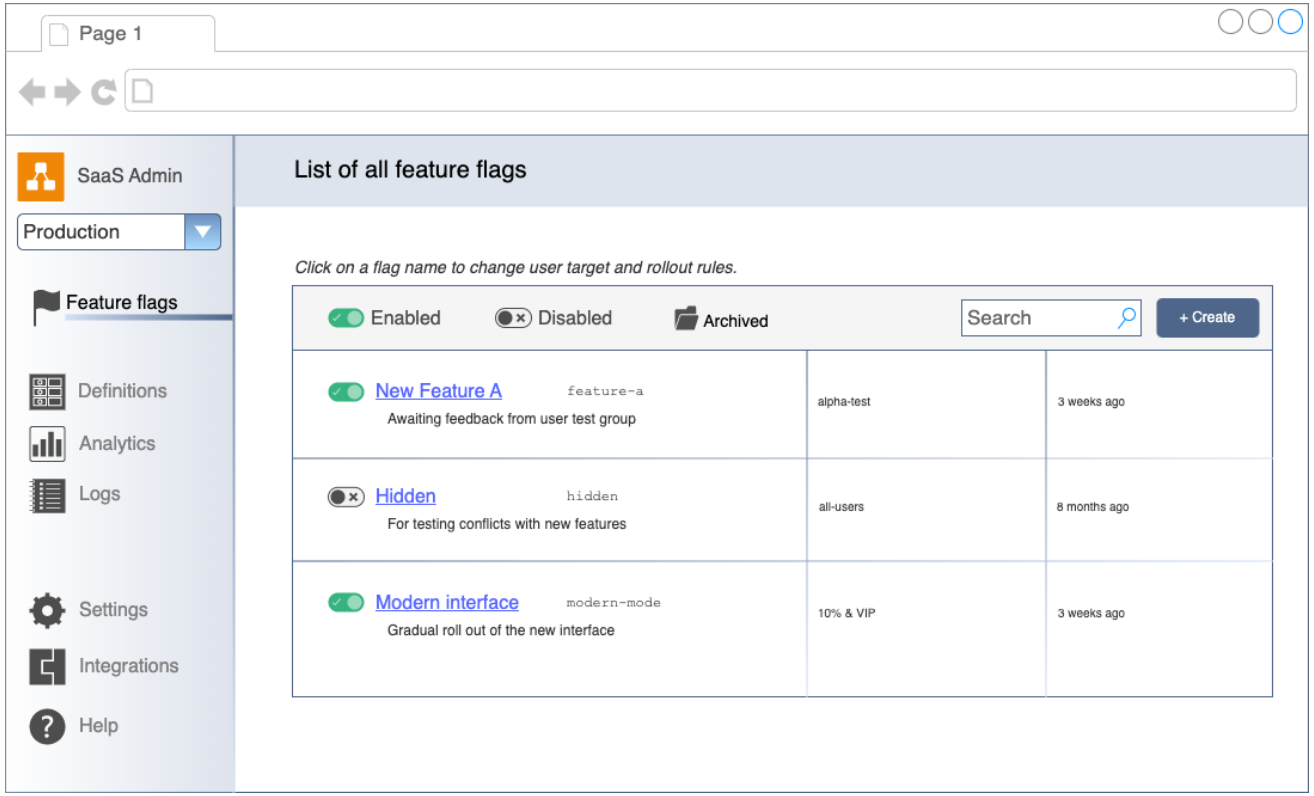

DIAGRAM 9: A MOCKUP OF A CONTROL INTERFACE FOR FEATURE FLAGS ON DEPLOYED CLOUD SOFTWARE

While you can build your own feature flag control system, there are several platforms and services available that provide this functionality. LaunchDarkly is the most commonly recommended, including by Atlassian and Microsoft.

WHY EXPLAIN USING DIAGRAMS?

Humans process visual information faster and more easily than text explanations. Software development has many difficult concepts and complex processes, therefore, it is easier to understand when visualised.

Most of the web resources covering feature flag DevOps are pure text explanations. These aren't helpful when trying to convince a time-poor and less technical audience, such as those in management, the value of a change to an existing development process. It's faster and easier to explain with the help of diagrams.

All the diagrams in this document were created with our online diagramming application at app.diagrams.net using the built-in shape libraries.

DIFFERENT TYPES OF DIAGRAMS

Teams from different departments need a variety of diagrams for various purposes - training, documentation, planning, presentations, project tracking, and so on.

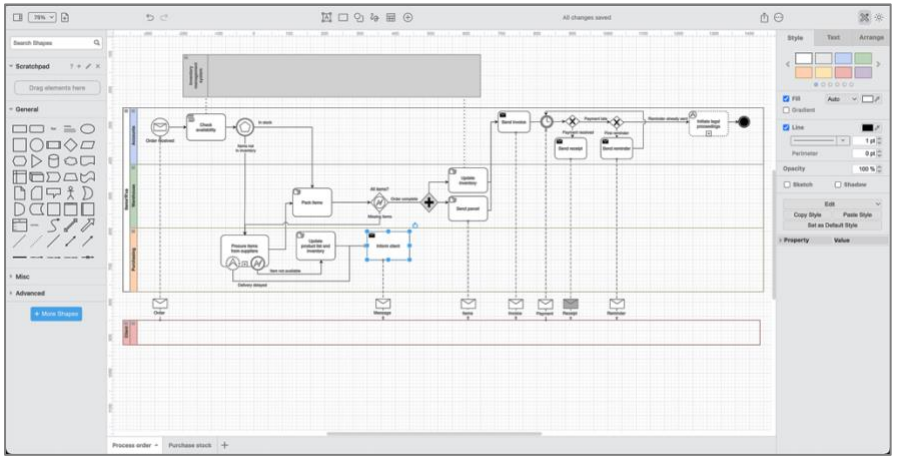

DIAGRAM 10: A BPMN DIAGRAM BEING EDITED IN DIAGRAMS.NET

With diagrams.net and our draw.io apps, you can draw an extensive range of diagrams.

From highly technical UML software specifications, cloud infrastructures, user flows, and interface mock-ups, through to whiteboard sketches, customer analyses, business processes and educational infographics, you can draw whatever you need.

Please see our gallery of example diagrams for more inspiration.

In addition to the vast shape library and built-in templates, you can create your own shapes, edit connection points on existing shapes, and draw freehand shapes.

Teams can work together on the same diagram with shared cursors for seamless remote collaboration.

You can also automatically generate diagrams from text - PlantUML, Mermaid, CSV, SQL, and more.

To diagram faster, you can customise the editor to use your preferred styles, fonts, and colours.

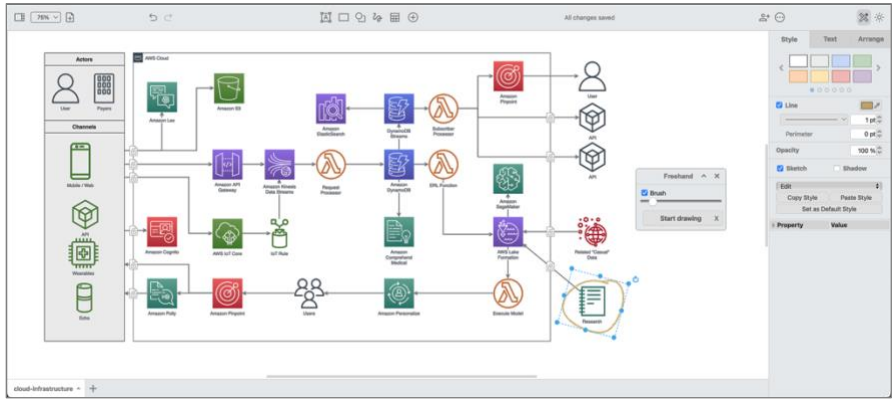

DIAGRAM 11: ANNOTATING AN AWS NETWORK DIAGRAM WITH A FREEHAND SHAPE IN A TEAM MEETING USING DIAGRAMS.NET AS AN ONLINE WHITEBOARD

START DIAGRAMMING TODAY

Anyone can use our free web application at app.diagrams.net or download the diagrams.net desktop app (Linux, Microsoft and macOS) to diagram offline.

As diagrams.net is open source, there are integrations available for many third-party platforms, in addition to our own extensions for Microsoft, Google and Atlassian products.

Your diagram data is secure - you choose which cloud storage platform or device to use to save your diagram files. No account is needed to view or work with your diagrams.

Visit diagrams.net to learn more.