Mo.net Financial Modelling Platform
Introducing CI/CD for Financial Modelling Workloads

May 2025
Revision 3

Purpose

This datasheet explores the benefits of using CI/CD with the Mo.net Financial Modelling Platform to unlock significantly enhanced development agility & quality for financial modelling workloads.

Background

Continuous Integration and Continuous Deployment (or CI/CD) is now commonplace across the IT industry, especially in Agile-based delivery environments. CI/CD allows for changes to source code and / or configurations to be made rapidly, but in a controlled & largely unattended environment. This minimises manual intervention, which is typically the source of many production issues.

The same CI/CD approach is now possible using the Mo.net Financial Modelling Platform and the integration points with either Azure DevOps or GitHub source control environments and their corresponding pipeline / automation tools. This allows model changes or assumption / parameter changes to be made and deployed without the numerous manual handoffs usually associated with such activities. This in turn makes the process of updating models or assumptions more streamlined and controlled, without compromising business-specific flexibility or governance requirements.

When implemented effectively, CI/CD drastically improves development velocity, model quality, and operational efficiency.

What is CI/CD?

Continuous Integration (CI) is the practice of automatically integrating code changes from multiple contributors into a shared repository several times a day. Each change is verified by an automated build, enabling early detection of integration issues. Build integrity can be further enhanced by automatically running unit tests as part of the automated build process.

Continuous Deployment (CD) refers to the automated delivery of validated changes to production. The goal is to reduce manual intervention, mitigate deployment risk, and provide fast feedback loops.

Key Benefits of CI/CD?

There are numerous potential benefits of adopting CI/CD concepts within the financial modelling world, however some of the main benefits are as follows:

Faster Time to Market

- Automated builds and tests speed up the integration of new features and fixes.
- Enables teams to deliver working models in hours instead of days or weeks.
- Rapid feedback loops reduce the time spent on debugging or revisiting legacy issues.

Better Model Quality

- Automated testing catches issues early in the development cycle.
- Quality gates, static code analysis, and code coverage metrics ensure models meet standards before reaching production.
- Reduced risk of bugs slipping into production improves customer satisfaction and trust.

Enhanced Productivity

- Modellers can focus on developing models instead of managing builds or manual deployments.
- Automated workflows reduce repetitive tasks, freeing up resources for other value-adding tasks.

Consistent and Reliable Deployments

- Automation removes the variability of manual processes, leading to predictable outcomes.

Improved Collaboration and Governance

- Easier collaboration through integration with version control systems like Azure DevOps or Git.
- Visibility of build status, test results, and deployment history.
- Shared dashboards and notifications help identify and resolve issues collaboratively.

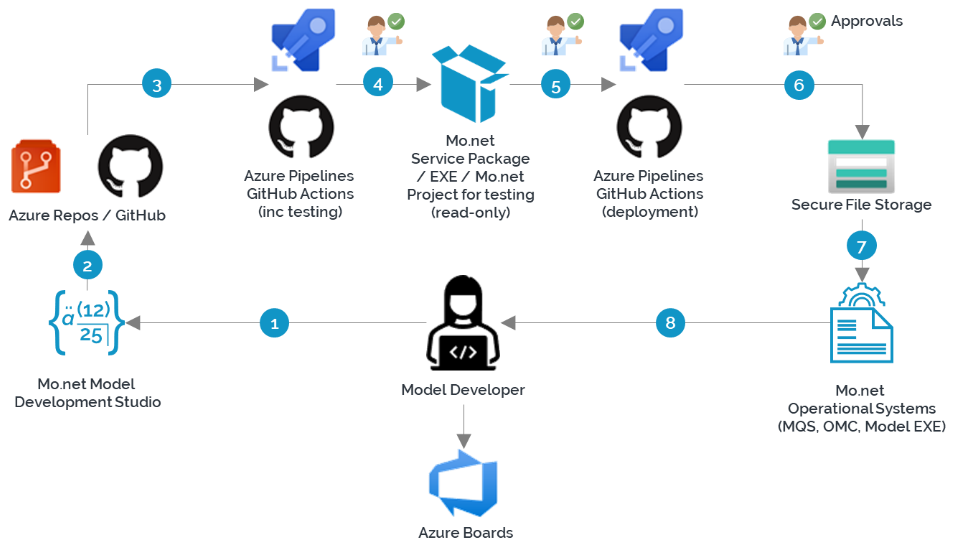
Benefits of CI/CD for Financial Modelling Workloads

In addition to the typical benefits of CI/CD listed above, there are some additional benefits when applied to financial modelling workloads. These include:

- Support for gated release processes, not typically seen in financial modelling environments, largely due to the legacy technology in place
- Rigorous Do check review / approval process
- Clarity over versions of modelling artefacts

Understanding CI/CD with the Mo.net Financial Modelling Platform

The following diagram outlines how CI/CD could be used with Mo.net to support financial modelling workloads. Each step in the process is described in more detail below.


Approvals
Azure Repos / GitHub
Azure Pipelines GitHub Actions (inc testing)
Mo.net Service Package / EXE / Mo.net Project for testing (read-only)
Azure Pipelines GitHub Actions (deployment)
Secure File Storage
Mo.net Model Development Studio
Model Developer
Mo.net Operational Systems (MQS, OMC, Model EXE)
Azure Boards
1
2
3
4
5
6
7
8

Stage	Description
1	Model developer picks a new ticket / work package from the backlog and undertakes model development using Mo.net Model Development Studio
2	Developer commits changes and pushes to Azure or GitHub repo
3	Azure Pipeline or GitHub Action notifies testing community that a new change is available for testing within the given repo
4	Testing community provides sign-off of the change and a build pipeline starts to build the required Mo.net task EXE, DLL or service package using the Mo.net Command Line Compiler
5	The generated Mo.net EXE, DLL or service package is then picked-up by a subsequent Azure Pipeline or GitHub Action. Further approvals can be supplied if required to prepare for release.
6	A separate Azure Pipeline / GitHub Action then moves the Mo.net EXE, DLL or service package to a prescribed file location ready for manual publishing via Mo.net Enterprise Service Manager / OMC, or into the landing area ready for unattended publication to a production operational system. This may optionally include a further business stakeholder (4 or 6 eyes) approval.
7	The Azure Pipeline / GitHub Action updates operational system metadata making the package / EXE available for use. Old versions of the package / EXE are optionally marked as unavailable depending on operational requirements.
8	The Azure Pipeline / GitHub Action marks the ticket / work package as live.

Build Pipelines

Build pipelines are an integral part of CI/CD. They are automated processes to build, test, and deploy code changes efficiently and reliably. A pipeline contains a set of steps (or stages) that code moves through from the commit of a change through to the deployment of the change to a live environment.

Financial Modelling Use Cases

There are already several use cases where clients have successfully leveraged CI/CD within their Mo.net modelling environments.

Enterprise Model Development

A large insurance organisation uses Azure DevOps to build and test its primary Mo.net-based valuation and reserving models. With YAML pipelines, they ensure every code commit triggers a build and a full suite of unit and integration tests. Builds are completed in under 2 minutes, and failures are reported instantly to their model development team via Microsoft Teams.

Realtime Updates to Customer-Facing Quotations Portals

A multinational bank uses Mo.net and CI/CD pipelines to deploy changes to model assumptions in minutes rather than hours, allowing them to rapidly respond to changes in market conditions. The build pipeline includes automated tests to ensure that the changes to assumptions are in line with expectations, together with a 6-eyes approval mechanism to ensure key stakeholders are part of the change management process.

Conclusion

CI/CD is no longer optional - it's essential for delivering high-quality models quickly and efficiently. By combining the Mo.net Financial Modelling Platform with Azure DevOps or GitHub, insurers can now manage complex build pipelines across diverse environments and use cases.

Contact Us

For more information regarding the Mo.net platform and the adoption of CI/CD concepts for financial modelling workloads, please get in touch:

Software Alliance Limited
30 Stamford Street, London, SE1 9LQ
Tel: +44 (0) 20 3964 2755
www.softwarealliance.net

Author: Guy Shepherd