BUILD SERVICES

Application lifecycle management

enable®

Introduction

Enable's unrivalled approach to software engineering is built on the client rapport and unwavering commitment to quality that are the bedrock of our company.

This document will take you through the processes involved in managing the lifecycle of an application, from the initial application to the post-delivery support that we provide.

This includes the initial set-up, scheduling and assignment of tasks, tracking code changes during development, supporting different versions of the application across different environments (such as development and production), user acceptance testing, deployment of the application to a given environment, roll-back procedures and flexible support offered by our client services team.

At Enable, we challenge ourselves to go beyond the modernising of ageing software and automating of inefficient processes to seize *every* opportunity to make significant improvements that will positively affect your business on a daily basis. We do this by asking the right questions to understand the fabric of your organisation and produce the most effective simple solutions.

THE JOURNEY TO A SUCCESSFUL OUTCOME

1. In the analysis phase, the application is conceived and a specification document is produced.

2. The application enters the build phase.

3. For a new application, this is set in motion by the creation of a new solution — the high-level "container" used to organize all code related to a single software product.

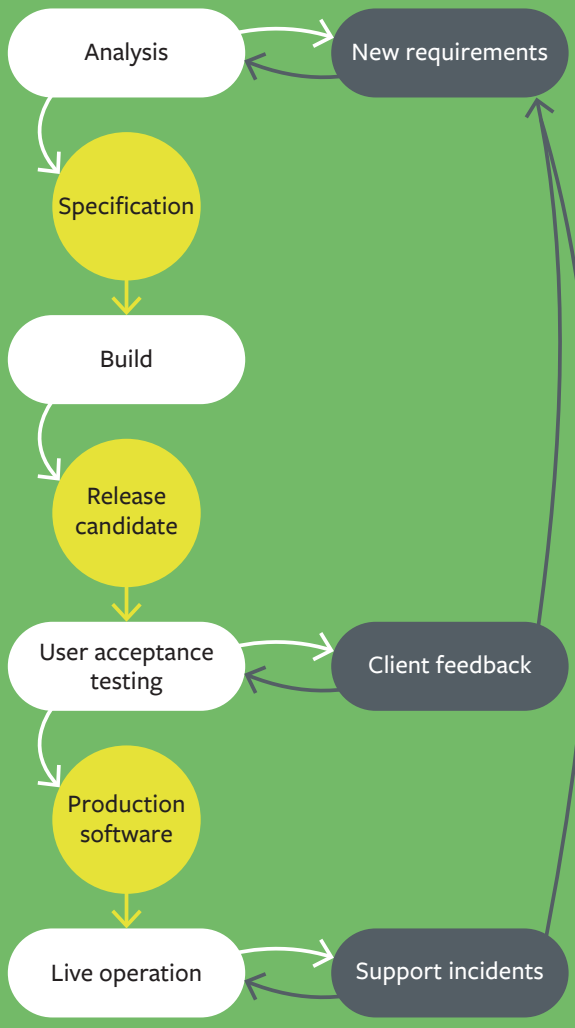
4. Our team carries out the work detailed in the specification, making use of dedicated "feature" branches for related areas of work. In-house testing is undertaken to ensure each specification point has been achieved.

5. The application is deployed to the user acceptance testing (UAT) environment, where the end-users of the software can test its performance in real-life scenarios and any concerns can be addressed.

6. Once the UAT process has concluded, the application is released to a live hosting environment.

7. The application will then be maintained and monitored regularly so that any bugs detected can be fixed without delay.

Our dynamic approach to the lifecycle of an application ensures that all clients' requirements are understood and delivered by our engineering team from its original conception.



Project governance and work item tracking

SOLUTION PREPARATION

Most projects will include a certain amount of 'lead-time' which is the period between the initiation and execution of a project. Tasks carried out during the lead-time may include:

— Visual Studio solution set-up;
— Review of branching strategies;
— Azure DevOps Server iteration schedule and sprint planning, including work item planning and tracking;
— Preparation of work areas for QA, bugs and reviews.

The requirements for the project — taken from the project scope agreed with you — are loaded into the project team's backlog iteration as requirements or user stories. The project scope is documented and captured in contractual, functional and service delivery documents.

The team now make use of custom-built solution set-up scripts, which aim to streamline the process of preparing a solution for development for the first time on an individual team member's machine. These scripts automate a variety of common tasks such as checking out the solution code from the centralized source control, installing required packages, and setting up local copies of databases.

TASK ASSIGNMENT

Each requirement or user story in the backlog will have several child tasks created for it. Typically, these will include planning/review, build and test.

Any estimated time is assigned to each task by reverse-engineering the time calculated for the corresponding requirement and the notes are reviewed alongside this to ensure allocated times are appropriate for the task.

TRACKING

Tracking progress ensures that an application is always moving in the right direction by providing a rapid feedback cycle that identifies problems early.

The current status of a task will be updated as it progresses and a typical workflow will involve a task with a status of 'Proposed' being changed to 'Active' when a team member starts working on the item.

As they progress with the item, they will regularly update the 'Completed' and 'Remaining' time assigned to the task. Having access to such a large amount of data on real task completion times allows the team to optimize the estimation process for future projects.

When an engineer commits code, they will also associate this code with the related task. This means that should another engineer be reviewing the code at a later date they can easily link it back to the original specification point.

Once complete, the item will either be marked as 'Resolved' — if a review of the item is required by another team member — or 'Closed'.

A high-level overview of the team capacity and total remaining hours (derived from the 'Remaining' time assigned to each task) gives the project lead a clear picture of whether the project is proceeding on schedule and how much work is assigned to each individual team member.

BUGS

After a team member has completed the build task associated with a requirement, another member of the team will pick up the testing task. If any bugs are discovered during the testing process a new 'Bug' work item is created by the tester and, in most cases, assigned to the team member that implemented the functionality.

Any bugs raised during testing are then fixed and the work item is marked as 'Resolved' and assigned back to the tester for them to review.

Bugs may also be raised for pre-existing issues which are not related to the current project's requirements. At any time throughout the engineering lifecycle, if any such bugs are raised, they can be added to the "Product QA" backlog of Cello – our in-house flexible ticketing system. From there they will either be picked up by the current engineering team or by an engineer assigned dedicated time to review these backlogs and resolve any issues.

QA ITEMS

Quality assurance (QA) items are raised when opportunities to improve the usability or accessibility of a new feature are identified. They are typically raised during user experience (UX) reviews between the project lead and other senior managers in the engineering phase but can also be raised by team members at any point during the engineering lifecycle.

RECOMMENDATIONS AND NEW REQUIREMENTS

Our recommendations process allows the client and Enable to jointly focus on risks and opportunities that involve the technical underpinnings of a solution. Topics that are commonly covered by this process include technical modernisation, application performance, security, refactoring and user experience design. Each recommendation will have a context as to why it has been raised and how it can be addressed.

In some cases, our team may suggest that we need to gain further insight into the client's business processes in order to understand the best way to move forward with an idea. A topic may be referred to our analysis team for one or more of the following the reasons:

— The suggestion alters the underlying design of the product;
— Significant engineering work is required;
— The business requirements are unclear;
— It hasn't been possible to locate relevant requirements in existing specification documents;
— There may be a commercial impact.

The steps our team then follows are:

— Analyse requirements & write a specification;
— Estimate and perform a commercial review;
— Schedule the project;
— Build the software.

Sufficiently minor recommendations may instead be tackled on an ad-hoc basis by an engineer outside of scheduled project time.

Change management and version control

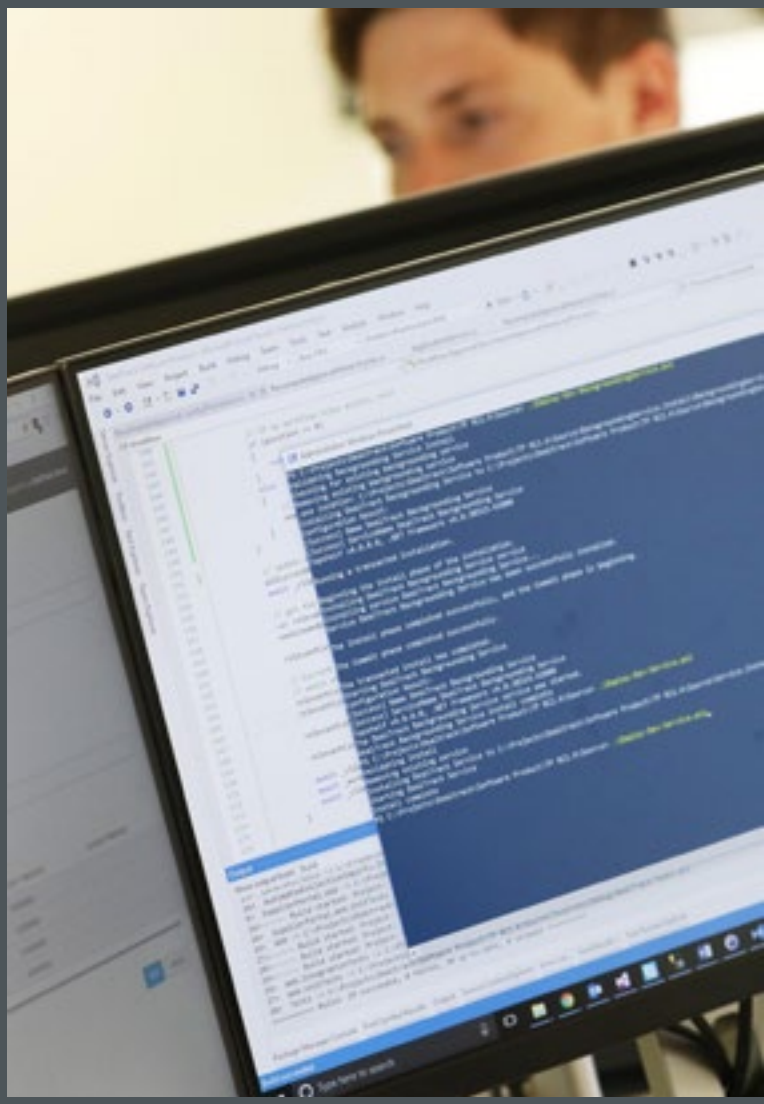
Version control allows Enable to monitor changes made by engineers and projects to have multiple streams of engineering occurring concurrently. To make this achievable, it is crucial that the source code is developed with the source control and versioning in place from the very start of a project.

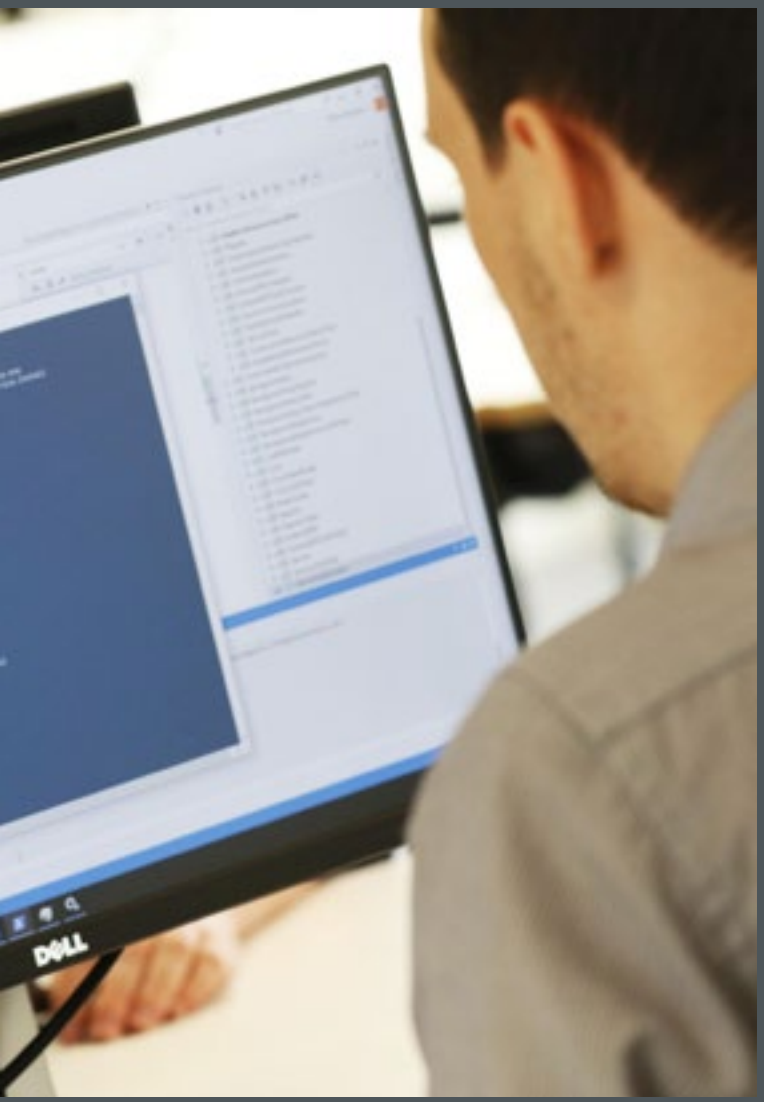
Engineering work is primarily carried out on a central master branch, with separate 'feature' branches for the implementation of single features. Upon completion of a project, a 'release' branch will be made for that phase of work; this is the version which will be used for the UAT and eventual Live release.

Version control enables engineering and support work to happen concurrently by allowing changes made in one branch to be merged into another. A support ticket fix made in a live branch can be merged into an engineering branch once any conflicting changes have been resolved to prevent a regression when the engineering branch is pushed to the live environment at a later date.

Enable uses the popular open source software Git to manage source code, which allows us to:

— Clearly track the difference between two or more versions of the code;
— Review the history of code to determine the effect of a single commit on the code base;
— Save time identifying the cause of problems;
— Organise our team's work;
— Ensure the team's work is properly integrated.





G I T

Git is an open source distributed version control system aimed at data integrity and support for distributed, non-linear workflows with the ability to handle projects of all sizes with speed and efficiency. These qualities make it an ideal choice for Enable's software engineering practices and it has become our primary source control system.

Branching is a simple and efficient process in Git so it is common that, for a phase of work, the main "master" branch will be branched a number of times into short lived engineering branches used by a single engineer or a small team. These branches may relate to the implementation of a single feature and are merged back into the master branch once the feature has been completed.

Branching and merging strategies

BRANCHING STRUCTURE

Typically, a solution will consist of a master branch for engineering work, and release branches for deployments of a project phase to production and UAT environments.

Project work – or significant bug fixes – will take place on "feature" branches off of the master branch. When the feature is complete, or the bug fixed, a code review will be carried out with another member of the team, and the feature branch will be merged back into master.

OUR APPROACH

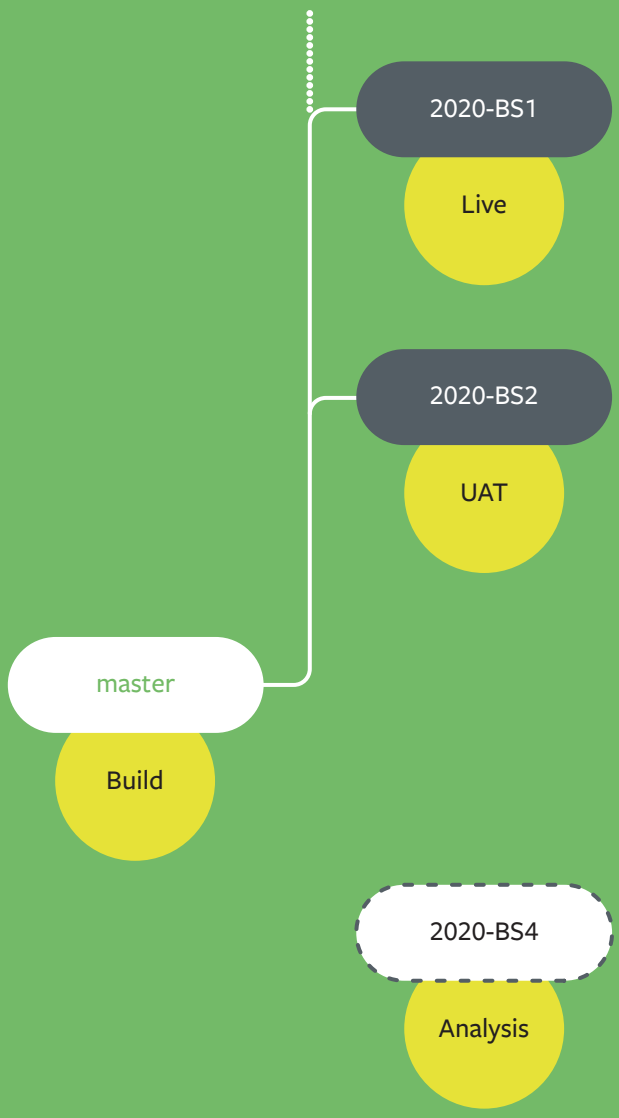
Branching allows our teams of engineers to easily collaborate inside one central code base.

For a new phase of work, all changes from the engineering project team are completed on the master branch.

While engineering work is being performed on the master branch, there are likely to be support changes made in the production branch and changes within the UAT branch.

Changes made in the production and UAT branches are actively merged down from older release branches through to newer release branches and then finally to the master branch, once the client has confirmed that the change has resolved the bug or issue. This ensures that any bugs fixed in these branches are also fixed in any future phases.

Once the engineering phase of work has been completed the lead engineer will create a new branch, named after the new release candidate's build slot (e.g. 2020-BS1), by branching from the master branch.



MERGING DURING DEPLOYMENT

When a phase of work is deployed to a new target environment — e.g. the deployment of the UAT branch to the live environment - the engineer performing the release will check that any support fixes made in the production branch have been merged from the production branch to the UAT branch.

Once the engineer is happy that all outstanding changes from the production branch have been merged into the UAT branch, it will be deployed to the live environment using the appropriate deployment steps.

Once the engineer has confirmed that this has been successful, the old production branch is deleted leaving only the UAT branch (now production) and the master branch.

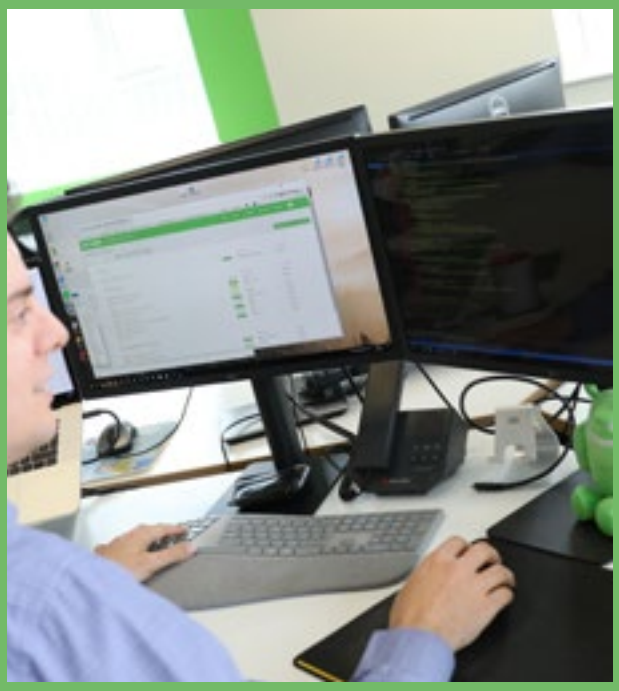
COMMIT MESSAGES

Enable has opted to take a standard approach when providing comments for a commit message to aid with merging activity.

When committing a fix or a new feature, the message is suffixed with the ID of the associated Azure DevOps work item, making it easy for future engineers to see how and why a change was implemented. For non-project work such as support tickets, bug-fixes and recommendations, the ID of the relevant item from our ticketing system, Cello, will instead be referenced.

When the change is merged into another branch, the engineer performing the merge will instead prefix the commit message with details of where the change is being merged, for example “Merge from [2020-BS1] to [master]”, followed by the details of the change.

Adding messages in this format makes it simpler for other engineers to merge these changes to other branches at a later date. This is particularly beneficial for solutions where multiple phases of engineering work are being performed simultaneously.



User acceptance testing

Once our internal testing is complete, clients are invited to do their own testing to make sure that everything is delivered to their satisfaction. This process — which occurs on test servers using realistic data in a real-life scenario — gives the client an insight into how the new system will fit with their daily workflow and gives them the opportunity to have any concerns addressed before completion of the project.

User acceptance testing (UAT) benefits both the client and the engineers as the client gains first-hand confidence that the software will add real value to their business while the engineers are assured that the client is fully satisfied. This is one of the critical software project procedures that must occur before new functionality is rolled out.

Enable proactively estimates and schedules for the UAT required by projects, which is typically calculated as a percentage of the project build time. Although the delivered project has already undergone extensive in-house testing before delivery, experience has shown us that end-user testing is invaluable for ensuring that the deliverable fits a client's requirements — client satisfaction is, after all, paramount.

The UAT phase is formalised during the delivery meeting, which takes place between the Client Services manager and client when the project is nearing completion.

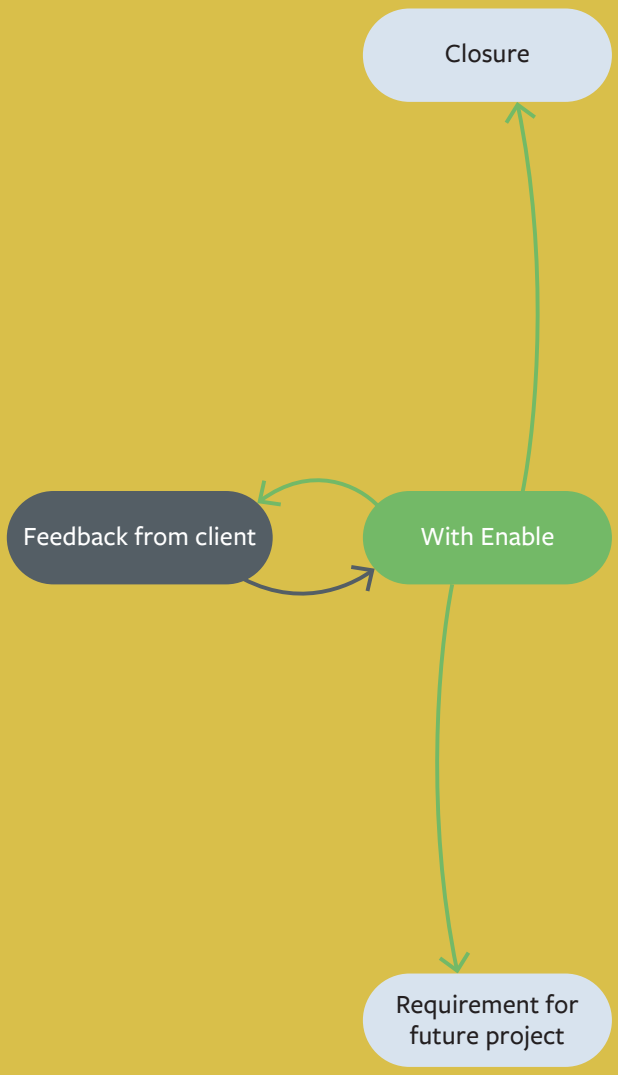
The client is guided through the UAT process covering any necessary prerequisites, the role of the Client Services Manager in the process, and how to interface with the Enable engineering team regarding reported issues. Vital information is also communicated to the client regarding weekly update emails and the release schedule for any system changes for review. Time is also allocated for the engineering team to act on any UAT feedback provided by the client and progress is continually logged, allowing efficient collaboration whereby engineers can, if required, seamlessly continue tasks started by others.

Once satisfied that the solution is ready for deployment, the client formalises this with the Client Services Manager who will then liaise with the team in order to schedule the 'go-live' date which is then shared with the client. This date is typically six weeks after the beginning of the UAT phase.

Enable uses the custom built Cello ticketing system which is designed to specifically handle the process of issue tracking during the UAT process. This software was designed to provide visibility and complete transparency when raising and tracking issues by keeping everyone in the loop about any issue.

Both Enable and the client can raise queries using this software, and all responses are made visible to both parties. A client may add and track issues through the creation of a new UAT item individually and single items are then distributed to engineers.

During the engineering process the Enable team may use Cello to raise clarifications regarding the details of the work being carried out, for example to clarify the meaning of certain specification points.



Release management and deployment

Part 1
Automated deployments

A modern software solution is no longer one monolithic application deployed to one server and there are often many applications making up a full software solution which are deployed across many servers.

We make use of a leading third-party deployment automation tool called Octopus Deploy which is used for deployment of code and performing a number of key tasks, such as the storage of important variables relating to a project.

Manually releasing to these servers and keeping track of versions across different environments is a time consuming and potentially error-prone task, so automated deployments are key to enabling this level of complex release management. The benefits of this automation include:

— Helping to reduce the risk of human errors (such as forgetting configuration changes or forgetting to run scripts) so releasing becomes repeatable in a much more reliable and predictable way;

— Taking the complexity away from deployments, meaning that almost any member of the team would be able to deploy updates to the applications;

— Providing a convenient way to deploy to a completely new environment without the painstaking process of ensuring each server is configured in the same way. This ensures consistent environments between development, UAT and live, as well as scalability;

— Targeting the same build to be released across multiple servers and environments so we are able to release to UAT and, once tested, promote the same build of the software to live (where the same deployment process is used);

— Allowing for quicker and more frequent releases to environments. For example, during work on a project using automated deployment, we deploy the latest code versions to our development environments at scheduled times throughout the day.

Due to the advantages discussed above, Enable uses automated deployments for software releases wherever possible and, as with manual releases, we operate to a formal release policy.

PRE-DEPLOYMENT CHECKS

Before requesting a release, the engineer will perform a series of checks. They will ensure:

— Downtime in the target environment has been considered;

— All necessary merges have been carried out;

— Any requests on deploying data have been respected;

— The deployment method is known;

— The Octopus Deploy configuration is complete and all Octopus Deploy components are installed on target servers;

— Any runtime dependencies have been installed on the target servers;

— All application configuration files been reviewed for settings that need to be configured in the target environment;

— The configuration has been suitably synchronised across all servers, if deploying to a web farm or load balanced web servers;

— Email redirection configuration is in place where required.

POST-DEPLOYMENT CHECKS

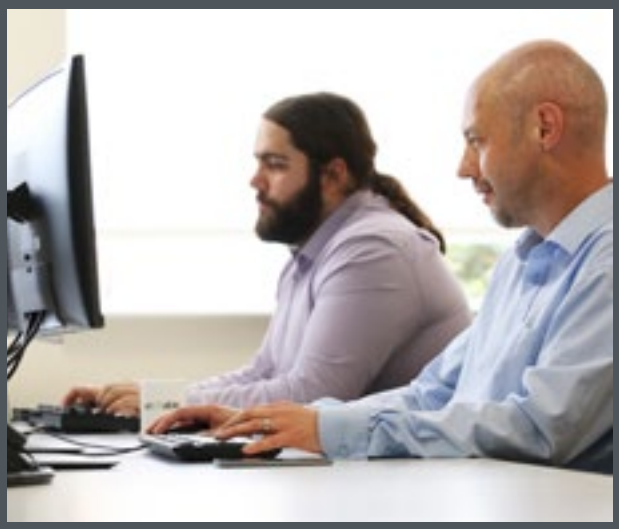
Following every deployment Enable performs the following checks:

— Testing to confirm that key areas of functionality are working, including, in the case of a deployment to fix a bug, that the bug has been fixed;

— Verify all application configuration files contain the expected setting values;

— Attempt to test file system permissions;

— Monitor application service activity, diagnostic logs and error feeds for signs of configuration issues.



Part 2
Manual deployments

Manual deployment of updates may be necessary in certain scenarios. In this eventuality, Enable observes a formal release policy dictating the steps that must be followed when deploying software updates to a production environment.

All steps are well documented and most commonly undertaken by different roles with Enable maintaining complete segregation of duties between software engineering teams and support and hosting teams. The process of manual deployment includes the following steps:

— Release preparation: deployable assets and accompanying documentation are prepared;

— Release request submission;

— Release execution: deployment of the assets contained in the release;

— Release confirmation: verifying that the deployment has been successful;

In rare cases where an issue is encountered during the release that cannot be rectified quickly, the software is rolled back to its previous state.

RELEASE PREPARATION

An engineer prepares all of the files necessary for the release and produces deployment instructions for the IT team.

Often there are multiple assets or components to each release such as website and application files, scheduled process files and database scripts.

RELEASE SUBMISSION

The release requestor submits a release request to the release executor which will include all necessary information required to successfully execute the release.

This might include manual configuration updates, or special instructions on running SQL scripts to apply database updates.

RELEASE EXECUTION

Prior to commencing with the release execution, the release executor will contact the release requestor to check whether they are available and ready to support the release process during its execution and to perform testing upon its completion.

If the release requestor is not available at this point, the release will be delayed until a time when the requestor is.

CHECKS

Before any files are released or any database scripts are run, backups of the existing files and data will be taken and only once all the backups are taken is the release executed. This may involve steps such as:

— Making the web application inaccessible and displaying a temporary "holding" page. This may not be required in cases where the release can be completed quickly and will have no impact on users currently interacting with the web application;

— Stopping any scheduled process due to be updated during the release;

— Applying the updates to the application or scheduled process using the assets included in the release;

— Applying any database updates using scripts included in the release;

— Applying any configuration updates required;

— Making the web application accessible and removing the temporary "holding" page if necessary;

— Restarting any scheduled processes that were previously stopped for the release.

RELEASE CONFIRMATION

As soon as the release has been completed, the release executor informs the release requestor that it is ready for testing. The release requestor will then test the release and, if a problem is found, will liaise with the release executor in order to figure out a solution as a matter of urgency.

Any further updates required as a result will also go through the exact same release procedure.

Once the release requestor has approved the release, a member of the IT/Hosting team will be informed and the client will be notified that the application has been updated.

Part 3
Environments & safeguards

A hosting environment can be a collection of physical or cloud devices, or a combination of both, which are all beneficial options made up of load balancers, web, database or background processing servers. Multiple environments are used during the engineering process of an application to allow for the testing of new functionality within the application before it's released to a live application. Different approaches to bug fixes are taken for each environment and these approaches vary based on the impact of any code changes which may occur.

U A T

Once a phase of work has been completed it is released to this environment where it can be tested and the client can confirm that the software is performing as expected. Code changes carried out as part of the UAT process are deployed to this environment. To minimise the number of releases, Enable carries out thorough testing of any new or changed software code resulting from a build cycle in its development environment prior to releasing to a client's UAT environment. The client is then encouraged to carry out thorough UAT, with any identified bugs being resolved in the UAT environment and subsequently tested.

The entire software build in the UAT environment, including any bug fixes, is promoted to live on a go-live date to be agreed with each individual client. It is not possible to promote bug fixes in UAT to live individually.

L I V E

The live environment hosts the main application and is the environment in which users complete the majority of their work. It will often be the most powerful of the environments.

For each bug identified in live, Enable determines whether an architectural change is required. Fixes to address problems identified in live that require a major architectural change are not promoted directly to live but are allocated into, and handled as a part of, the next build cycle. Fixes will be released into the UAT environment along with all other aspects of the build and will be tested thoroughly during UAT before being promoted to live.

A fix which does not require an architectural change, or is deemed sufficiently urgent, may be applied directly to Live; all such fixes are thoroughly tested by Enable before deployment. Our testing methodology includes test driven development, automated regression testing, manual regression testing, automated unit and integration tests, and manual end-to-end testing. Enable continuously invests in both improving testing practices and increasing the breadth of tests carried out.

We are also very receptive to working with clients to identify and create specific automated tests that can be built into our standard release procedures. When a fix is released to live for a client reported bug, we will notify the client when the fix has been released.

On completion of pre-release testing and following the release of a fix into live, Enable will carry out additional testing in live to ensure that there is no unexpected behaviour. In the unlikely event that unexpected behaviour is detected, Enable will determine what follow-up action is required in order to resolve the issue as swiftly as possible. This usually involves an immediate follow-up deployment or a roll back of the changes.

While it is standard industry practice for cloud/SaaS providers to deploy appropriately tested fixes directly to live environments without seeking permission from individual clients, the above measures work to ensure that: The number of fixes deployed directly to live is kept to a minimal level; The risk of potential problems being introduced as a result of live fixes will be significantly reduced.

D O W N T I M E

In cases where a major release candidate needs to be applied to any environment, Enable will liaise with the client in advance to agree a suitable date and time for the release to be actioned. A maintenance page can be put in place during the deployment — all of this is done to ensure minimal disruption.

There are times when a support fix will need to be deployed to an environment. The downtime required to deploy support fixes is dependent on the nature of the fix with a minor change likely to have a negligible impact on users. Deployments to UAT are typically deployed on an ad hoc basis while deployments to live can be performed at either a certain, pre-specified time each day or on an ad hoc basis. Fixes that involve changing data or with a high functional impact will often be referred to the client prior to deployment.

R O L L - B A C K

If there are any issues encountered during the release that cannot be rectified quickly, the release executor will take the following steps to revert the application to its previous state:

— Restore the database from the backup taken;
— Restore all of the website files from the backups taken and remove any temporary “holding” page;
— Restore any scheduled processes from the backups taken;
— Perform testing and notify the client.

Client services

HOW WE HELP

Our Client Services team are responsible for the delivery and support given to clients.

The team takes time to deepen their knowledge of the solution in question, using internal information systems, specifications and test versions of the software.

Working closely with and physically adjacent to Enable's software engineers, they develop an understanding of the key issues, risks and subtleties to ensure they provide the best service possible to clients.

BUILD AND DELIVERY

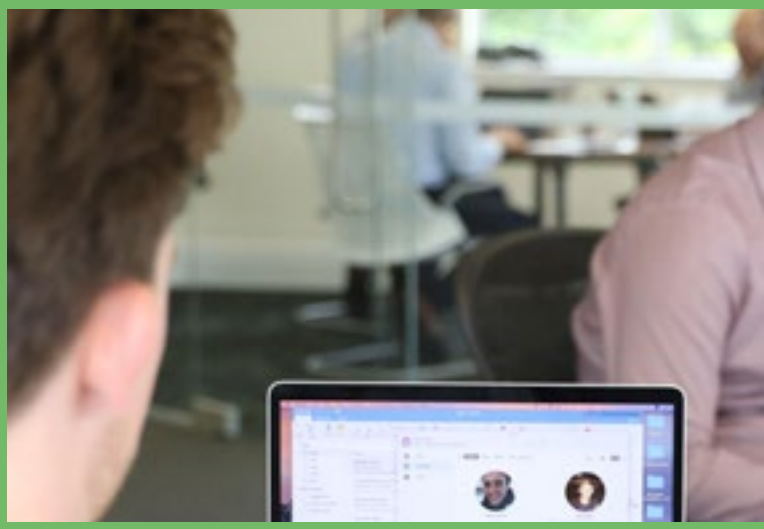
While the solution is being assembled, the team will facilitate and support any clarification questions raised by the client and engineers.

When a solution is ready to be delivered, the team will familiarise themselves with the solution and attend a presentation meeting with the client which will involve demoing the software, talking through release notes, and kick starting the user acceptance testing process.

A single member of the team will attend these meetings for smaller projects, but will be joined by a second member for larger ones. For very small projects the meeting will sometimes be held online.

DURING UAT

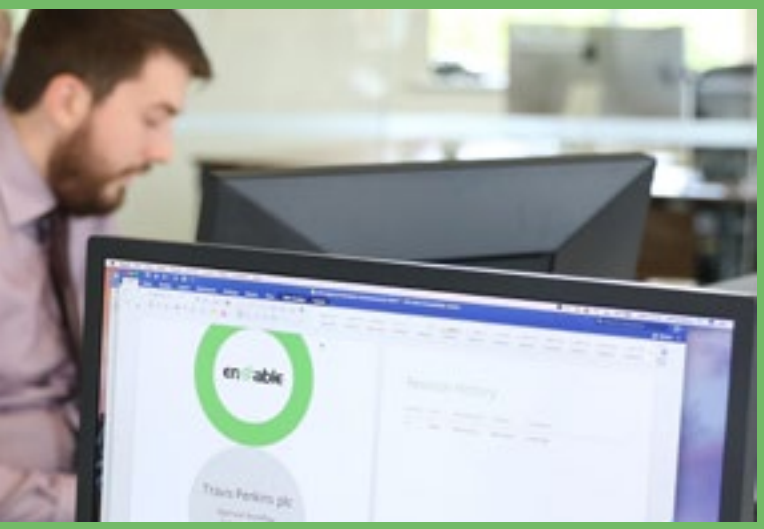
The team will provide support to the client by email, over the telephone and face to face during UAT. In a fast-paced cyclical process, the client will review the work completed and provide feedback which will then be addressed by engineers, allowing the client to conduct further reviews. After six weeks, the client should be able to verify that the individual statements in the specification documents have been fulfilled in the software solution they have received. The key word in the acronym UAT is 'acceptance' — acceptance of the precise functionality described in the specification — rather than 'testing', as the software will have already been subjected to Enable's manual and automated testing.



ONGOING HELPDESK SUPPORT

If a client encounters a bug or has a question, the team will record a ticket to track the client's request and seek a resolution.

The team will keep the client updated on the status of tickets and will be responsible for liaising with the client, attending conference calls and making suggestions when more complex situations emerge during live operation. The way in which support tickets are handled will be subject to a contractual service level agreement (SLA) with major clients receiving a quarterly support report and a review meeting to discuss the report. All members of the Client Services team will contribute to this support process.



The support service covers:

— General questions;
— Problem diagnostics and bug fixes;
— Thorough investigation work and detailed written feedback to queries;
— Quotations for changes and advice;
— Proactive software management, performance and error feed monitoring;
— Software migrations, consolidation and decommissioning.

Where the team becomes aware that a client has identified new requirements that are not covered by existing specification documents, they will refer the client to Enable's Customer Success team so that workshops can be arranged to capture these requirements in detail.

The Client Services team are available by telephone and email every business day, 52 weeks of the year between the hours of 08:00 - 18:00 (GMT), 03:00 - 20:00 (EST) and 00:00 - 17:00 (PST).

FIND US ONLINE

ENABLE.COM

HELLO@ENABLE.COM

UK OFFICE

+44 203 998 7470

10-12 THE COURTYARD
STRATFORD-UPON-AVON
CV37 9NP, UK

USA OFFICE

+1 628 251 1057

535 MISSION STREET
14TH FLOOR, SAN FRANCISCO
CA 94105, UNITED STATES