# What is Deployment?

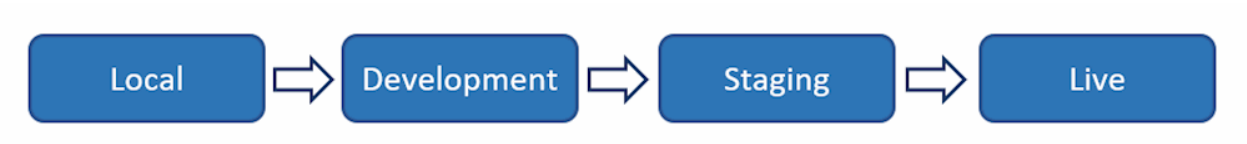
Deployment in software and web development means pushing changes or updates from one deployment environment to another. When setting up a website you will always have your live website, which is called the live environment or production environment.

If you want the ability to make changes without these affecting your live website, then you can add additional environments. These environments are called development environments or deployment environments. The additional development environments will typically be a local environment, a development environment, and a staging environment (also known as a staging site). How many environments you need is up to you and depends on the complexity of the project you are working on.

While deployment models can vary, the most common is the classic "left to right" deployment model when working with multiple deployment environments. In this model, changes are made in local, development, or staging environments (depending on the setup) and pushed from left to right through the different environments ending up in the live environment. Once this deployment process has been completed the new changes will be visible in the live environment.

## Table of content






The above image shows a very simplified and classic way of handling deployments when working with websites in a CMS. You do not necessarily need all of the above environments, but the process stays the same.

By using multiple environments you get a list of advantages - the main one being, that you can make changes without it affecting your live website. Once the changes are made, tested, and ready to be pushed live, the deployment process takes care of the rest.

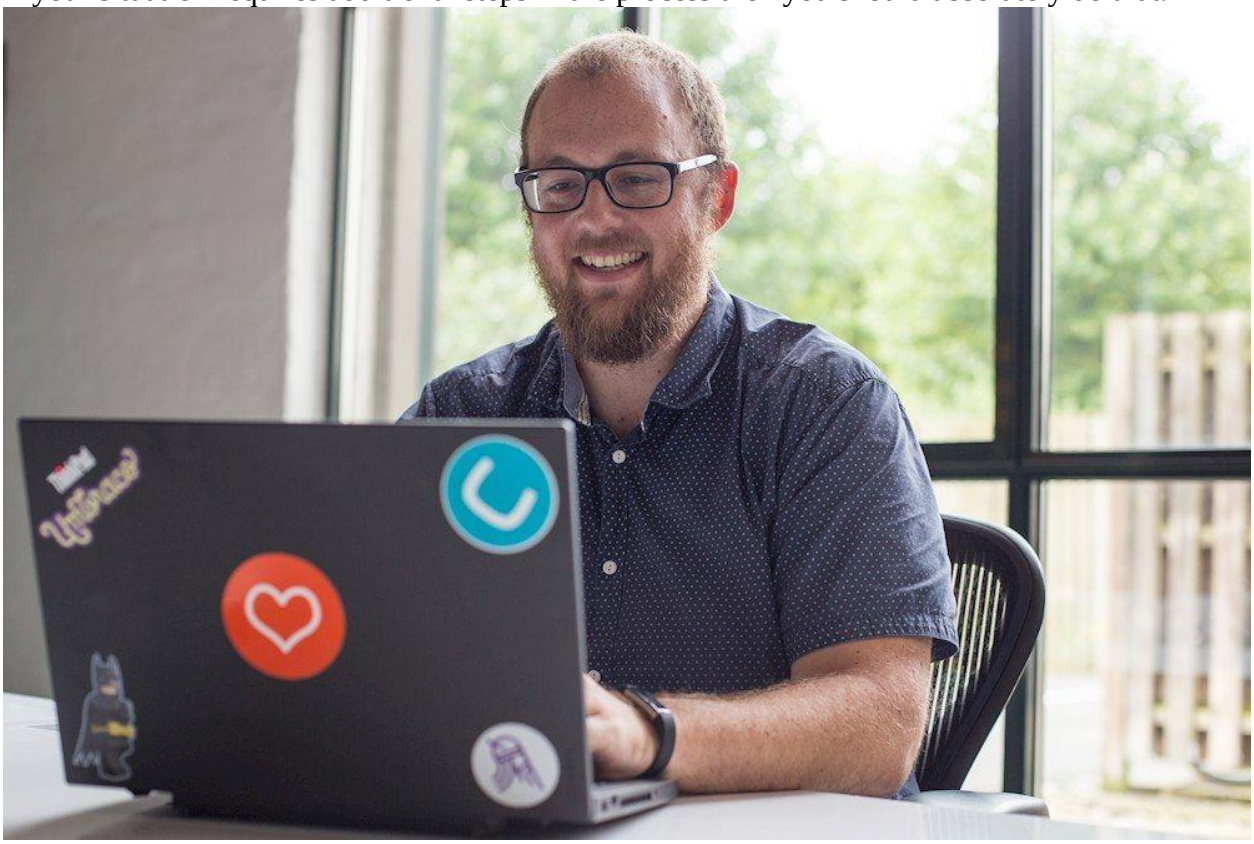
## Which steps are in the deployment process flow?

The deployment process flow consists of 5 steps: Planning, development, testing, deploying, and monitoring.

Below we'll dive into each of the 5 steps, but before we do, we'd like to add a quick note.

The deployment process flow below covers the fundamentals, which are split into 5 steps. That doesn't mean it's **the only way** to do it - there very well might be a better process for you. We've tried to keep it as simple as possible to make it cover the most important parts.

If your situation requires additional steps in the process then you should absolutely do that.



## 1. Remember to have a software deployment plan

To make sure the deployment process goes as smoothly as possible it is best to have a deployment plan that you follow - every time. By having a plan you ensure that everything is done the same way each time changes are made. This is especially helpful when multiple users are working on the same project.

A deployment plan should include rules for when to deploy from local environments to development or staging sites as well as schedules for when new changes can go to a live

environment. By having a set plan you reduce the risk of conflicts between different changes and make sure the deployment process is as smooth and easy as possible. If you're working on an open-source project it also gives you the chance to do Release Candidates and let your community test it out for any bugs you might have missed yourself.

Besides an overall plan, it's also important to plan each individual change that you're going to do. This process will be very quick for minor changes but should be much more extensive for big changes. By planning well in advance, you're much better suited to have a smooth deployment process.

## 2. The actual development

Once you have the plan in place, it's time to do the actual development. To ensure that any development can be done simultaneously and without breaking anything, it's important to only work on local or development environments. Once the development process is done, it's time to start testing and deploying the changes through your environment setup.

Not sure whether to work locally or in a development environment? Then take a look at the **deployment best practices section**.

## 3. Testing your changes

Testing your changes is crucial to ensure that no bugs make it into the final production environment. But testing cannot be completed without deploying your changes to new environments.

Once you've tested that all of your changes work on your local or development environment it's time to deploy the changes to the next environment in line. This should be done all the way up to your staging environment, where final QA testing should be done. If everything is properly tested and works in an environment resembling your live environment it's time to deploy it live.

If you discover bugs along the way in any environment, it's important to have a plan for how to handle these. Typically any change that doesn't pass the testing on the staging environment should be sent back to the development phase and - once fixed - again work its way through the environments.

## 4. Deploying changes to the live environment

Once all of the testing has been done on previous environments and any bugs have been fixed, it's time to deploy your changes to the live environment. This should be a pretty safe thing to do,

but everyone who's worked with software development knows, that something can still go wrong.

So even though it's easy to stop here, it's important to include the final step of the process: monitoring.

### 5. Monitor your changes

Once your new changes are live and real users are actively using your website or application, it's important to monitor that everything works as intended. No matter the planning put forward, there's a chance that users encounter issues or perform actions that you did not anticipate during your planning and development.

A good tip for monitoring is to plan releases for times where the least amount of users will notice and where you have development resources ready in case something needs to be fixed. That way the number of users impacted by any bug will be minimal and you'll have people ready to fix it or roll back the changes if need be.

If you do need to roll back your changes, it's important to keep calm and have a process to handle that as thoroughly as you handle deployments.

## Different types of deployment

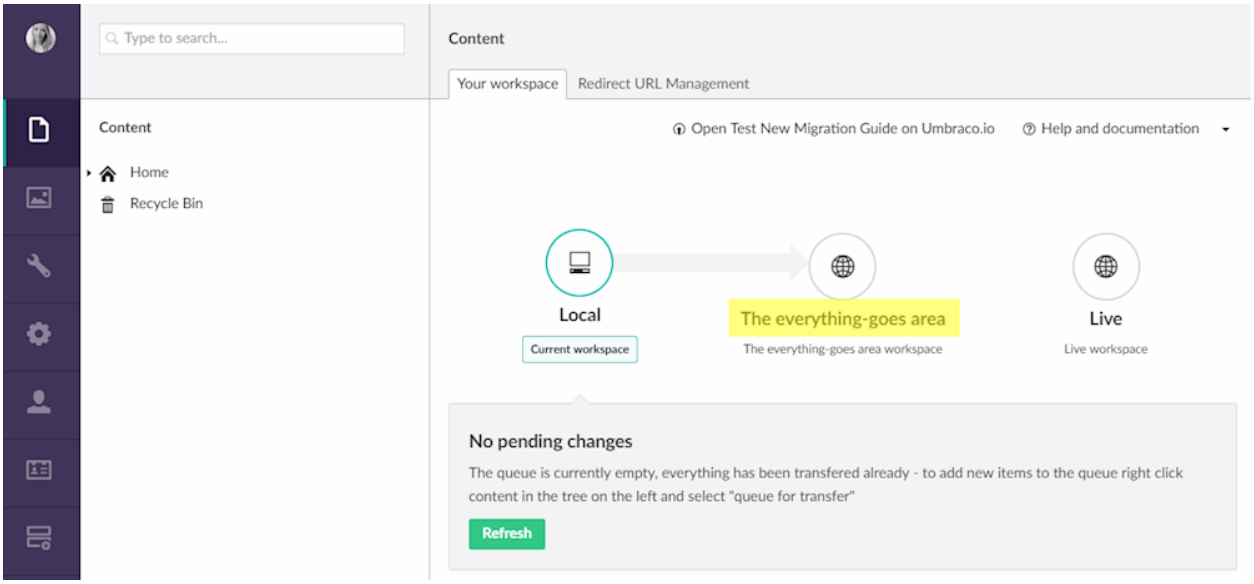
When it comes to the type of deployment it will often be split up in a two-part deployment approach. It will typically be a split between metadata and content as these have different impacts on a new environment and need to be handled differently.

### Deployment of metadata

Metadata includes changes to your code, templates, stylesheets, files, and so on. These changes will often require a validation check between environments to see if they have any unforeseen conflicts that need to be resolved. Many deployment tools will include checks for consistency and help guide you in case of conflicts.

### Deployment of content

Content such as text, images, and videos are handled differently during deployment as they are less complicated to move between environments than metadata. For that reason, you will often see that deployment tools make content deployment accessible for content editors and not only for developers. That way a content editor is not depending on a developer when it comes to pushing new content to a live environment.




*An example of multiple environments in Umbraco Cloud*

## Deployment best practices

When you're working with deployment environments it is - as previously mentioned - important to have a plan and a clear process for it in your team. To expand on that process we've gathered some best practices that are good to implement as part of your process.