HIPPO

Hippo Deployment Process

A look at the Hippo CMS deployment process

In this Tech Talk we will look at the process to deploy your Hippo CMS project to different environments. We will also discuss the options to automatically create environments and a potential approach to automate deployments.

Author: Ivo Bronsveld

HIPPO

Table of Contents

HIPPO

Introduction

This document will discuss the entire process of deploying your solution. We will take a look at the implications of this process, so you can determine the best approach to get your code, site configuration and content to the final production environment. We will also explain the possibilities to automate deployments as part of a daily build scenario.

Hippo Enterprise Subscription

Official support for any type of Hippo automation is part of the Hippo Enterprise Subscription (HES). Next to that, this subscription gives you several key advantages, such as:

- Support, priority access to the Hippo Helpdesk for support
- (Optional) Enterprise features, such as the relevance module
- Enterprise Infrastructure support, such as clustering and failover support.
- Upgrade Tooling, making (future) upgrades easier

Also, the HES provides access to the Hippo Professional Services department. Hippo Professional Services consists of highly trained engineers, experienced enterprise (infra) architects and solution consultants. Having these people at your disposal make any implementation a breeze.

HIPPO

The Installation and Deployment Process

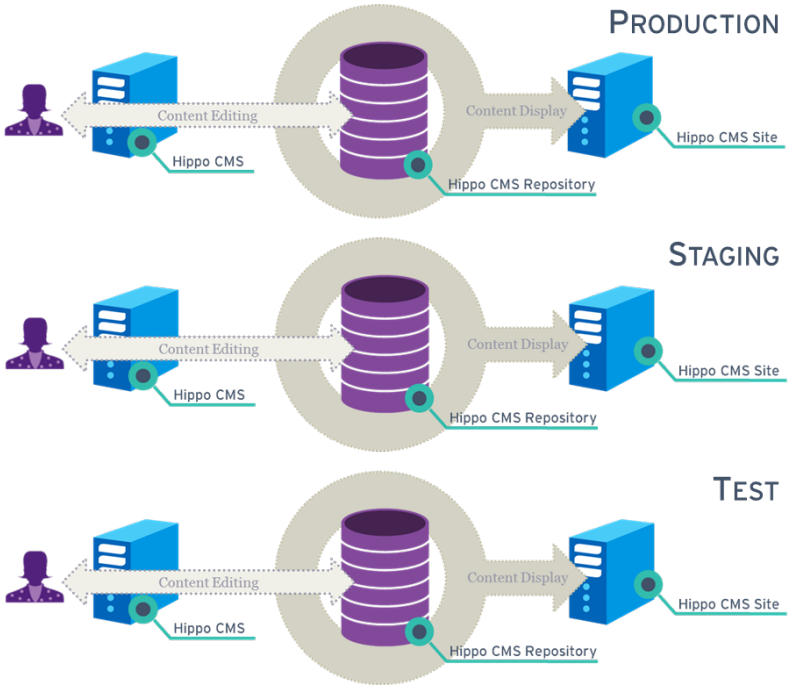
The three C's: Content, Configuration and Code

A Hippo CMS application will consist of three distinct parts:

- Code
  A developer will create code, such as JSP (templates) or Components (Java classes) that contain the business and application logic. Changes to this will always require a deployment to get it to another environment.
- Configuration
  Hippo will need to be configured so it knows when to execute what code. For instance: www.example.com/news uses the "News Overiew" template. Changes to this will not require a deployment to get it to another environment, but it is recommended in some cases to do this through a structured process.
- Content
  Obviously, Hippo CMS contains content. The content is a huge part of any application and will be very important part of the deployment process. Changes to content will not require a deployment.

Working with Content, the Hippo way...

In order to explain the role content plays in the deployment process, we need to take a look at the lifecycle of content on the different environments. Every environment (Test, Staging and Production) will have its own repository, CMS and site.

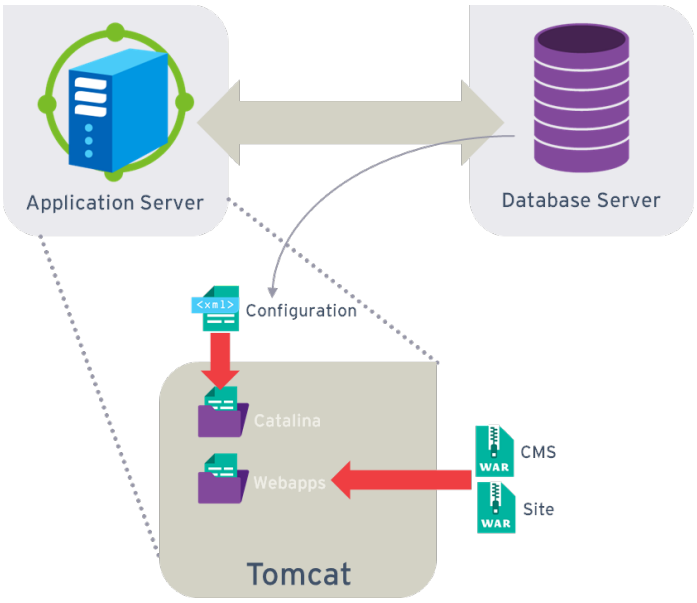


This means that content will be present (and created) on any of those environments at any moment in time. During a deployment content is not typically overwritten. It is very important to keep this in mind during the deployment.

HIPPO

Technical Overview of the application

A Hippo installation can contain many different parts, but all share the application server and the database server.

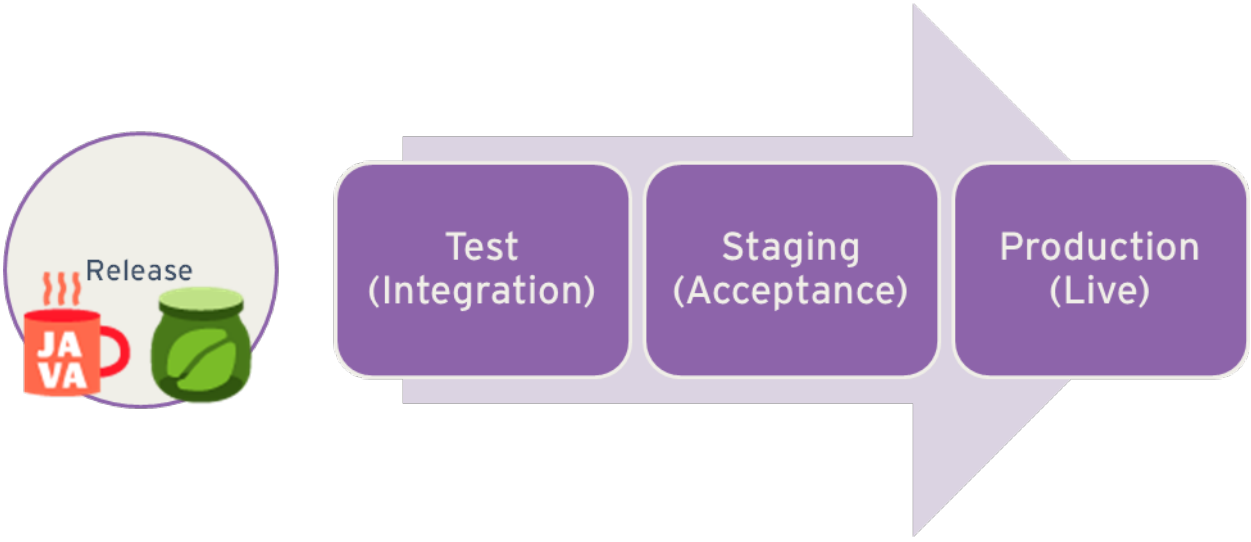


The application server will typically contain a tomcat installation which will contain the Hippo CMS and Site application (as WAR files). The configuration (containing things like database connections) is also stored here. The database server will store all the content and site configuration.

Next to these two war files, there will be several shared files as well that are used by both the installations. The focus will be on the two primary WAR files, as most of the application logic is contained in those.

HIPPO

The DTAP process



With any (Hippo CMS) project, the best practice will be to deploy the application to a test environment first, staging environment second and finally to the production environment.

Development Environment

Developers typically will develop the solution on a local machine. This will be done through use of the Hippo development tools and your normal IDE. Once the solution is good to go to the test environment, the developer build something called a distribution.

This distribution (or release) typically contains:

- Code (compiled in the correct war files)
- Configuration

And, like mentioned in Working with Content, the Hippo way..., content is generally not considered to be part of the deployment and will not be packaged.

Test Environment

The Test Environment can be used to test the validity of the solution on a technical level. In some cases, this deployment is part of a nightly build process. The Automated Deployments chapter contains some pointers on how the nightly deployment can be arranged.

Once the solution is determined to be technically sound, a deployment to staging will be done.

HIPPO

Staging Environment

With Hippo, this environment is typically used for two specific uses:

1. Final test before go live
   Does the deployment work without issues when going to production?
2. User Acceptance Testing
   Is everything implemented correctly from a functional standpoint?

A staging environment is not used for the creation of content. As mentioned in Working with Content, the Hippo way... content is normally created on the production environment. However, before deploying to the staging environment, it is common to copy the existing production environment (content / configuration) to staging to assure everything works when going to production.

If the release will replace an existing solution, there might be a need to change existing content or content types on the environment. In order to make sure these content types do not get overwritten or content replaced, we use Updater Scripts. More information on this topic will be available on the Community Website[1].

Production Environment

The production environment is the end-point of your deployment. The deployment from Staging to Production would generally be exactly the same as the deployment from Test to Staging.

[1] http://www.onehippo.org/7_8/library/concepts/upgrade/update-existing-content-items.html

HIPPO

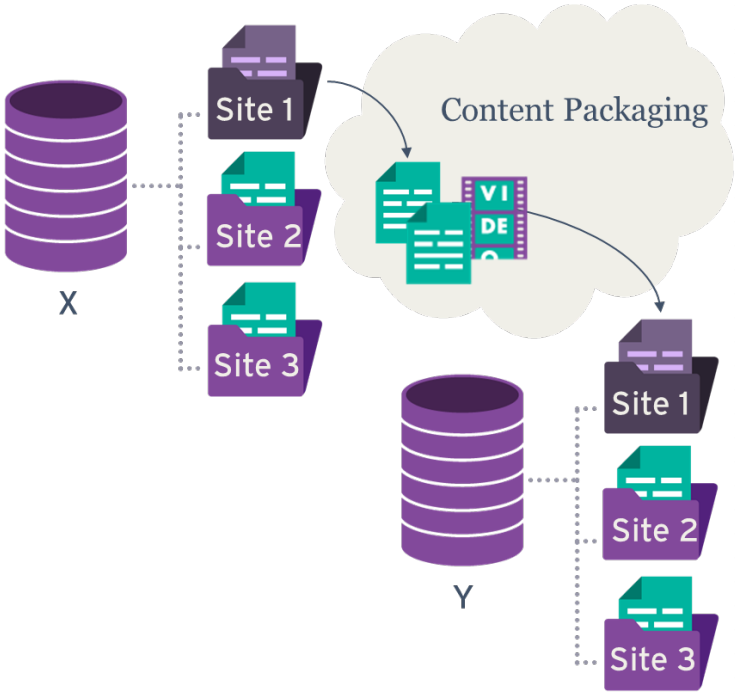
Migrating Content between Environments

Even though content is generally not part of deployments, sometimes there is a need to deploy content from staging to production. In the next release of Hippo a new feature will be introduced to facilitate this: Content Packaging

What is Content Packaging?

Content Packaging will allow you to 'bundle' certain documents and folders so you can deploy them from one environment to the next. This can be great in many different scenarios, like:

- Separating the editing and display environment.
- Move content from Production to Acceptance
- Move content into development environments



The content packaging solution will also allow you to do this automatically, reducing the effort of manually importing and exporting content.

Manual XML Import and Export

The XML Import and Export solution can be used to migrate content between different environments as well. This is done through the console and can be great for smaller pieces of content or configuration stored in the repository.

Automating Deployments

First we will take a look at automating the deployment process.

Before we start, we will assume the distribution has been built according to the instructions[2] specified.

The specifics will depend on the exact infrastructure and server configuration, but the five steps in the process will be rather similar for any type of configuration.

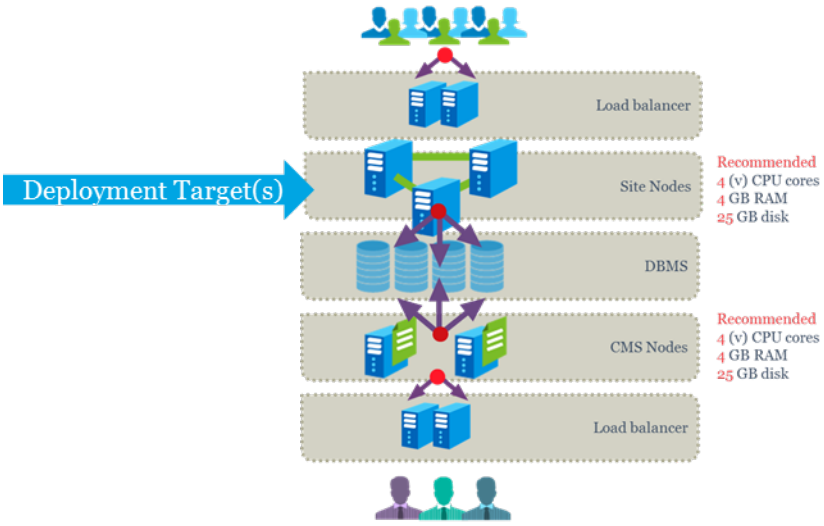
Next, when looking to automate the deployment (especially to multiple servers), there are a number of things to consider.

Deployment Steps

1. Retrieve war files
2. Stop Tomcat
3. Delete existing war folder
4. Move war to folder
5. Start tomcat

- Will you push the deployment to multiple locations from a central location?
- Will you pull the deployment from a generic location?
- What is the trigger for the deployment?

Most of our customers attach a deployment script to a build server, creating an automated deployment to a test server.



[2] http://www.onehippo.org/7_8/library/deployment/create-and-deploy-a-project-distribution.html

HIPPO

Example Script

```
#!/bin/bash

cd /tmp

wget http://www.mysite.com/deploy/myproject-distribution.tgz

tar -zxvf myproject-distribution.tgz

sudo service tomcat stop

sudo rm -rf /opt/cms/tomcat/webapps/site* /opt/cms/tomcat/webapps/cms*

sudo rm -rf /opt/cms/tomcat/shared/lib/*

sudo mv webapps/site.war webapps/cms.war /opt/cms/tomcat/webapps/

sudo mv shared/lib/* /opt/cms/tomcat/shared/lib

sudo service tomcat start

rm -rf /tmp/myproject-distribution.tgz /tmp/webapps /tmp/shared
```

Automating (server) Installations

In some cases, you would like to be able to create server installations on the fly. This is extremely important in O.E.M. installations, but can also help when trying to quickly scale.

Creating (new) server installations is also a possibility. Of course, this depends largely on the number of servers and infrastructure required. Typically, this involves a lot of non-Hippo technology, like an application server, a database server and load balancers.

The discussion of this is out of the scope of this document, but we can help you find the right solution. Contact us at helpdesk@onehippo.com

Additional Reading

The Hippo Community website contains much more information about deployments, including:

Deployment Scenarios

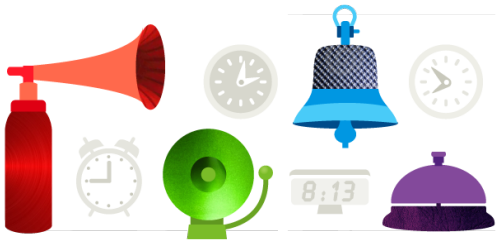
http://www.onehippo.org/7_8/library/deployment/supported-deployments.html

How to create a project distribution?

http://www.onehippo.org/7_8/library/deployment/create-and-deploy-a-project-distribution.html

Configure Apache HTTPD

http://www.onehippo.org/7_8/library/deployment/configuring/configure-apache-httpd-web-server-for-sites.html



System Administration Training

Also, when dealing with Hippo CMS deployments, check out our Hippo Sys Admin course. In this half day course, you'll learn all about how to install and maintain Hippo CMS and get insight into Hippo infrastructure best practices.

More information can be found on:

http://www.onehippo.com/en/services/education/hippo-system-administration-training.html