magnolia®

Architectural Blueprints: Moving your content management into the cloud

4. Magnolia with RabbitMQ activation

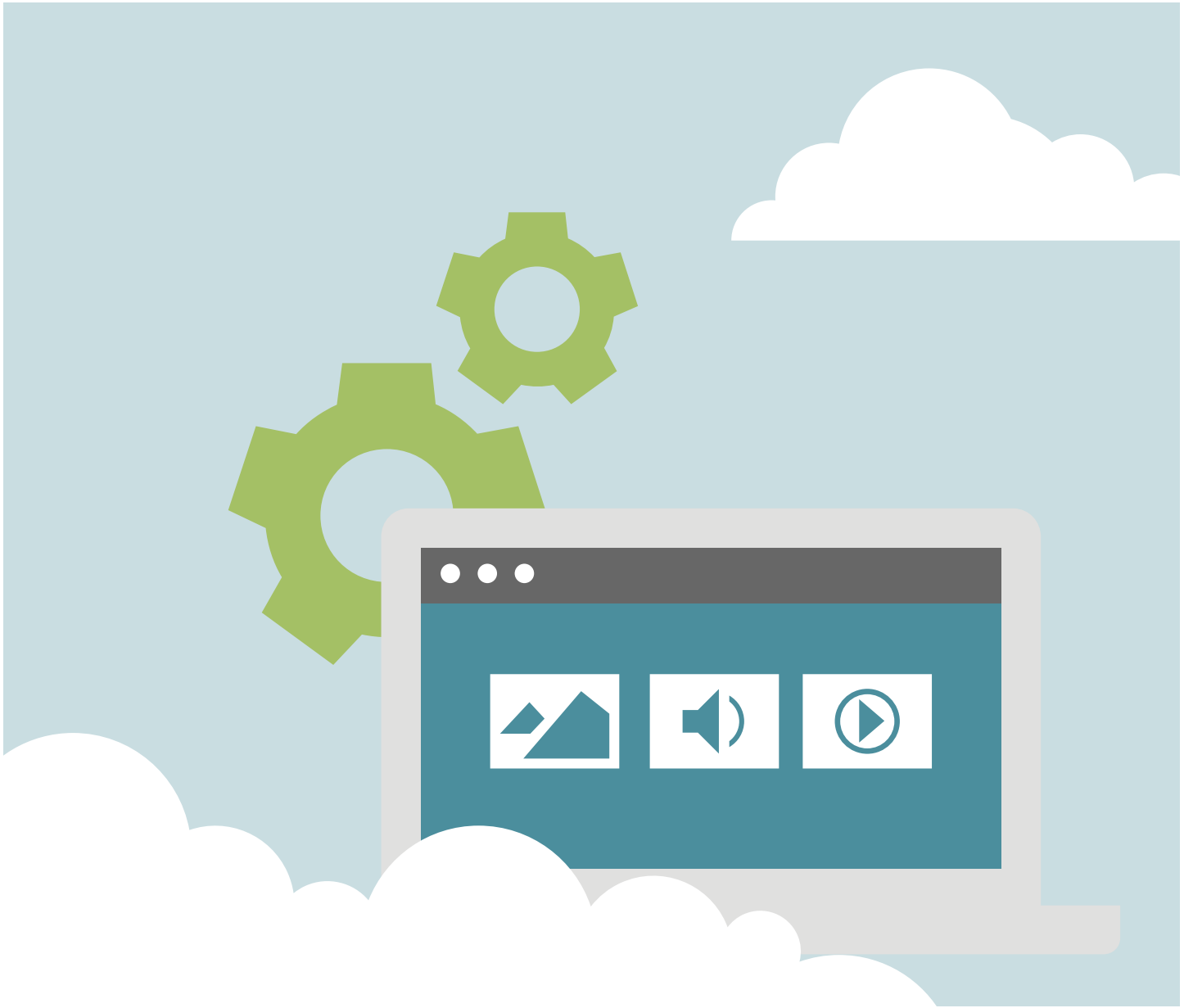
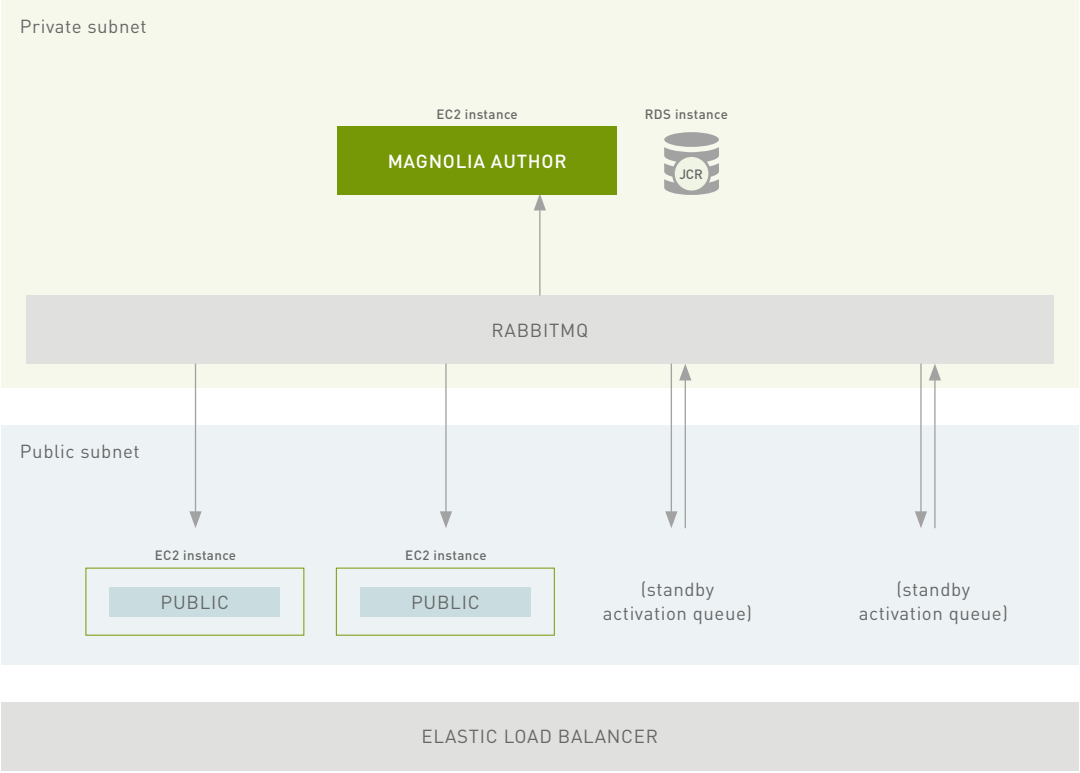


Table of contents

Architectural Blueprint 4: Magnolia with RabbitMQ activation

Enterprises today increasingly want to tap the benefits of moving their digital ecosystems to the cloud. A key piece in that ecosystem is their content management system (CMS).



Using Amazon Web Services (AWS) and Magnolia CMS as an example, we offer four architectural blueprints for optimizing the deployment to ensure seamless integration. We discuss the advantages and disadvantages of each approach and offer recommendations to achieve autoscaling for high-performance and high-traffic scenarios.

The four blueprints cover:

1. Standard CMS deployment
2. CMS deployment with content source target
3. CMS deployment with JCR clustering
4. CMS deployment with RabbitMQ

This white paper, the final of the four blueprints, looks at the general principles of deploying a CMS with RabbitMQ. While using AWS and Magnolia as a case study, the lessons learnt can be applied to other enterprise-level CMS deployments in the cloud.

You may have noticed that there are certain common problems in the architectural blueprints we have presented: content synchronization on a new Magnolia instance and registering a subscription for a new Magnolia public instance, for example.

Publishing content in Magnolia uses "transactional activation". The Magnolia author instance ensures that all subscribed public instances receive and save the published content. If one of the public subscribers fails to publish the content, the content publication is rolled back across all public subscribers.

Transactional activation guarantees that all public subscribers are kept in sync. Transactional activation comes at a cost: the time to publish content is proportional to the number of public subscribers; with more public instances, more time will be taken to ensure a successful publication.

Publishing content with RabbitMQ activation is an alternative to transactional activation. It uses RabbitMQ, an open-source messaging broker, to deliver activation messages from the Magnolia author to Magnolia public subscribers. RabbitMQ allows a looser coupling between the Magnolia author and public instances: publication with RabbitMQ activation will not be transactional, but the time to publish content will depend on how many public instances are running. With RabbitMQ activation, public instances could get out of sync, but also makes it easier to start and stop Magnolia public instances.

RabbitMQ activation allows many Magnolia public instances to be connected to a single Magnolia author without increasing the time to publish content.

There are other benefits to RabbitMQ. Since the Magnolia author and public instances are decoupled, there is no need for a publication freeze to prevent publications while a new Magnolia public is starting. There is no need to register the new public instance with the Magnolia author either; the public instance is registered with RabbitMQ, not the Magnolia author.

To provide a robust, scalable delivery mechanism, RabbitMQ can be set up to provide high availability queues with federation and clustering to ensure that the Magnolia author can always send activation messages for distribution to Magnolia public instances.

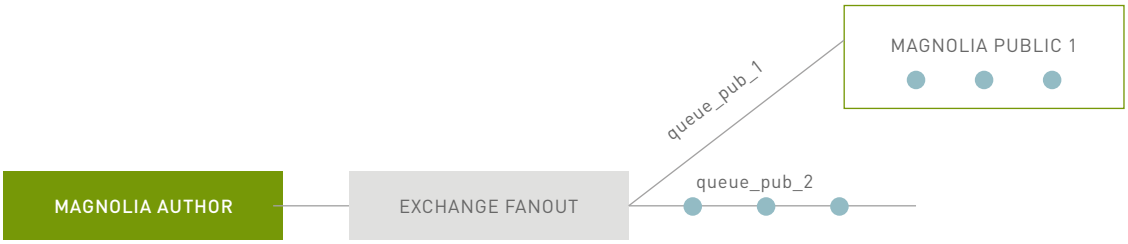
Content synchronization and RabbitMQ activation

In other blueprints, we recommended a hybrid approach to content synchronization: use the Magnolia Backup module to restore most of your JCR content, followed up by the Magnolia Synchronization module to synchronize any content changed since the last backup.

You can still use the Synchronization module with RabbitMQ activation, but RabbitMQ activation gives you another way to update recently changed content.

RabbitMQ is a message broker, distributing messages to clients through queues. Messages stay in RabbitMQ queues until they are delivered to clients. RabbitMQ queues that don't have a client will save messages until they are delivered to a client.

RabbitMQ activation uses RabbitMQ to deliver activation messages from a Magnolia author to Magnolia public instances. Each Magnolia public instance is attached to a queue, waiting for activation messages. With RabbitMQ, you can create "standby" activation queues without a Magnolia public instance client. Activation messages will be saved in the standby queue until a public instance claims the standby queue and RabbitMQ will deliver the activation messages to the public instance.



Activation messages are relatively small and RabbitMQ is capable of managing many queues and messages; standby activation queues could store many hundreds or thousands of messages in a RabbitMQ broker using minimal system resources.

Content synchronization coordination

There is some practical limit to how many messages can be stored in RabbitMQ queues. At some point, the standby activation queues must be flushed.

Standby activation queues can be used instead of the Magnolia Synchronization module. You will need to coordinate the backup with the standby activation queues.

When a backup is made, the standby activation queues should be flushed, since any publications waiting in the standby queues will be contained in the backup.

There are two ways you can make a scheduled backup of a Magnolia instance:

- By setting up a scheduled job within Magnolia (note that Magnolia must be running to make the backup)
- By setting an AWS Lambda function to run at a scheduled time (note that Magnolia doesn't have to be running when the backup is made)

Setting up a scheduled job in Magnolia to launch a backup is easy; see the Magnolia Scheduler module:
https://documentation.magnolia-cms.com/display/DOCS56/Scheduler+module

A scheduled AWS Lambda function could launch a Magnolia backup via the Magnolia REST API and flush the RabbitMQ standby activation queues. The coordination between backup and flushing queues doesn't even have to be precise: with RabbitMQ activation, a Magnolia public instance will discard any activation messages it has already received.

Handling out-of-sync public instances

When using transactional activation, publishing content can fail, but Magnolia guarantees all the public instances will be in sync. Transactional activation can fail for many different reasons: a subscribed Magnolia instance might not be actually running or is unable to process the publication within a set time; the Magnolia instance's JCR repository is corrupted; there is a time difference between the Magnolia author and Magnolia public instance; and many other reasons. Transactional activation prevents the public instances from getting out sync, but may also prevent any publications until a failing Magnolia public instance is repaired or taken out of service and its subscription is deactivated. Unfortunately, transactional activation doesn't provide a convenient hook for AWS services for detecting ailing Magnolia subscribers and correcting them.

RabbitMQ activation provides a feedback channel for activations: Magnolia subscribers can report whether a publication succeeded or failed on RabbitMQ acknowledgement. That acknowledgement can be monitored by the RabbitMQ monitoring app in the Magnolia author. The monitoring app shows what activations succeeded and failed for each Magnolia public instance, how long an activation message stayed in queue and how long it took for the public instance took to process it.

The RabbitMQ monitoring app can help you see whether Magnolia instances are in sync or not, but the underlying notification mechanism—the activation acknowledgement queue—can be an integration hook for managing Magnolia public instances with AWS services.

Here's how: activation notifications from RabbitMQ activations are also stored with Magnolia in a separate JCR workspace. Those notifications note whether an activation succeeded or failed, how long it waited for delivery in RabbitMQ and how long it took the Magnolia public instance to process the activation once RabbitMQ delivered it. All this information could be used to identify out-of-sync or ailing Magnolia public instances.

There are at least two ways you could tie in activation acknowledgements to AWS services:

Within Magnolia: use the Magnolia Observation module to watch the RabbitMQ activation notification workspace.

When the workspace is updated with notification that shows a problem with a Magnolia public instance (a failed activation or a slow activation, for example), post a notification on an AWS SNS topic noting a problem with the Magnolia instance. AWS Lambda functions could then do further notifications to notify Magnolia sysadmins or terminate the Magnolia instance and replace it.

Outside Magnolia: build an external client to monitor the activation acknowledgement queue and post a notification on an AWS SNS topic noting a problem with the Magnolia instance, letting other AWS services (like a Lambda function) kick in and handle the problem.

Recommendations

Recommended when running five or more Magnolia public subscribers

RabbitMQ activation allows publications to many public instances.

Recommended for high frequency of publication

Publishing is an expensive operation for the Magnolia author. Using RabbitMQ greatly reduces the load on the Magnolia author in publishing to many public instances.

Recommended for large numbers of Magnolia Admin Central users

Magnolia can support up to 30 to 50 simultaneous users of Admin Central. RabbitMQ reduces the performance load when those authors publish content.

AWS services used

- AWS Cloudwatch events (for autoscaling notifications)
- AWS Lambda functions (for autoscaling coordination)
- AWS SNS notifications (for invoking Lambda functions)
- AWS SSM agent (for executing commands on remote EC2 instances and restoring backups)

Autoscaling recipe with RabbitMQ activation

In this recipe, the logic of selection of an available standby activation queue resides in a Lambda function.

Step 1. AWS Lambda function sets up the new public instance:

- Select an available standby activation queue.
- Launch Magnolia.
- Wait for Magnolia to become available; configure the instance's RabbitMQ client configuration to use the selected standby activation queue.

Step 2. Once the new Magnolia public instance is up and running, add it to the load balancer.

Variation: autoscaling recipe without AWS Lambda

The logic of selecting an available standby activation queue could be included in a Magnolia module, so no Lambda function would be needed.

The module could query RabbitMQ, find an unused standby activation queue, and update its RabbitMQ client configuration directly.

This variation avoids waiting for Magnolia to start up before changing the RabbitMQ configuration.

Step 1. On Magnolia starting up:

- Select an available standby activation queue and update its RabbitMQ client configuration directly.

Step 2. In a Lambda function handling the autoscaling notification:

- Add the new Magnolia public instance to the load balancer.

Tooling

The following Magnolia features and extensions will help in building this blueprint:

- **Magnolia RabbitMQ modules:** publish content with RabbitMQ
- **Magnolia Property REST API:** for adjusting the RabbitMQ client configuration
- **Magnolia Node REST API:** for adjusting the RabbitMQ client configuration
- **Magnolia Backup module:** back up and restore a JCR repository
- **Magnolia Synchronization module:** synchronize content between a Magnolia author and public instance
- **Magnolia Observation module:** get notifications when the contents of a JCR workspace is changed
- **Magnolia Services auto-license module:** installs your Magnolia license and avoids the license prompt on first start-up
- **Magnolia Scheduler module:** execute commands at specified times

More about Magnolia modules:
https://documentation.magnolia-cms.com/display/DOCS56/Modules

More about Magnolia Extensions:
https://wiki.magnolia-cms.com/display/EX/Magnolia+Extensions

magnolia®

Contact us

Switzerland (HQ)
Magnolia International Ltd.
Oslo-Strasse 2
4142 Münchenstein (Basel)
Switzerland

+41 61 228 90 00
info@magnolia-cms.com

North America
Magnolia Americas, Inc.
168 SE 1st Street
Suite 1007
Miami, FL 33131
United States of America

(305) 267-3033
info-us@magnolia-cms.com

Spain
Magnolia España Software and Computer Applications S.L.
Paseo de la Castellana 153, Bajo
28046 Madrid
España

+34 662 63 43 36
info-es@magnolia-cms.com

United Kingdom
Magnolia Software UK Ltd.
9 Devonshire Square, 3rd Floor
London EC2M 4YF
United Kingdom

+44 7554 041 782

Czech Republic
Magnolia Software & Services CZ s.r.o.
Chobot 1578
76701 Kromeriz
Česká Republika

+420 571 118 715
info-cz@magnolia-cms.com

Singapore
Magnolia SIngapore
7 Temasek Boulevard
Suntec Tower One, Level 44-01
038987 Singapore

+65 64 30 6778

Vietnam
Magnolia
Etown 1 Building
Unit 7.10
364 Cong Hoa Street
Tan Binh District
Ho Chi Minh City, Vietnam

+84 28 3810 6465
vietnam@magnolia-cms.com