Neos Replicator

Release dev-master

Flownative

Aug 27, 2018

Contents

This documentation covering version dev-master has been rendered at: Aug 27, 2018

The Neos Replicator allows to replicate content from one site to another, to provide staging and synchronisation.

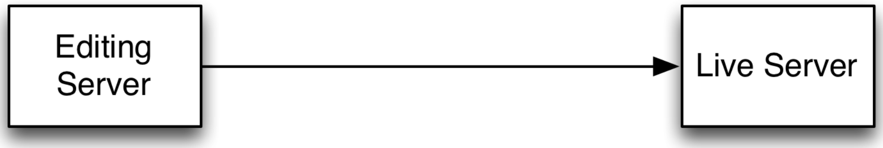
Note: This documentation serves as a specification for the package to be built. None of this is currently implemented!

CHAPTER 1

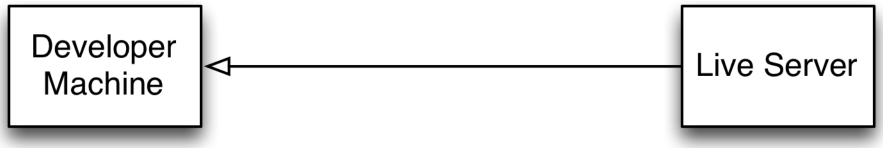
Overview

1.1 Use-cases

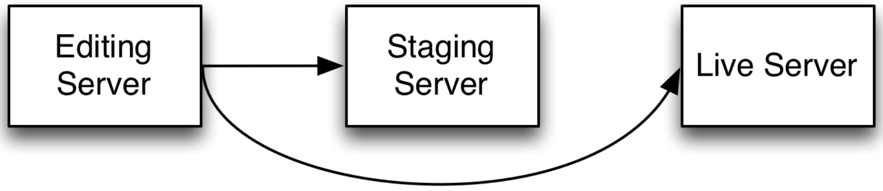
The simplest use-case for this is a simple replication from one server to another. This can be used to have an editing server that is not reachable from the internet. Content is then pushed to a live server (that has no editors, a read-only database connection, . . .):



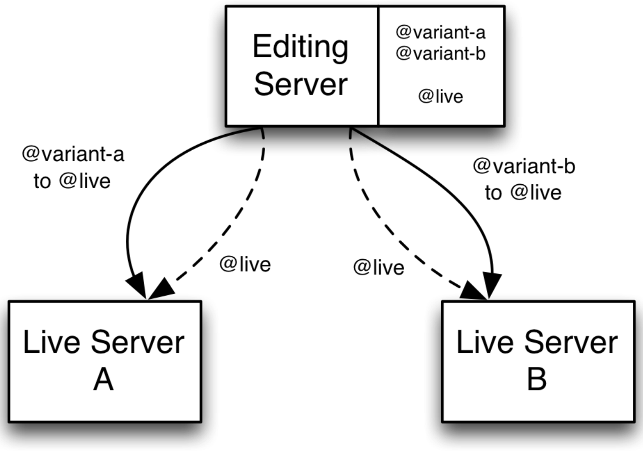
As simple but working in the other direction: developers can pull live content to their machine, so they can work with content beyond *lorem ipsum*:



Only a bit more advanced: add a staging machine to the first example and use it to have your legal department or management double check things before pushing to the live server:



One way to do A/B testing could be to replicate different content to two live servers. After drawing conclusions from the test, replicate the "winning" version to both live servers. This could be done by using three workspaces, two for the variants and a final workspace. The variant workspaces are replicated to the live workspace on one of the servers each. After the winning variant has been published to live on the editing server, that is is pushed to both servers, again into the live workspace, overriding the previous content.



CHAPTER 2

Installation

The systems need to be set up so far, that Neos can render a “no homepage has been defined”. That means a site package must exist and the configuration and setup of the database must be done.

The package then needs to be available in all Neos instances that will be involved with the replication. It can be installed by

```
$ composer require flownative/neos-replicator
```

CHAPTER 3

Configuration

The configuration is split in two areas of responsibility. There is the setup of available targets to be used in replication and there is the configuration of the replication itself.

3.1 Endpoint configuration

This is done in *Settings.yaml* as follows:

```yaml
Flownative:
  Neos:
    Replicator:
      # a key used to authenticate against the replication API, make it long and␣
↪random
      apiKey: 'long and random key for this system'

      # any number of nodes available for replication
      nodes:
        # the keys can be assigned at will, the special __self__ node is reserved␣
↪and always defined
        'staging':
          # a human-readable name for the endpoint, optional
          name: 'ame.com Staging Server'
          # a human-readable description for the endpoint, optional
          description: 'The staging server'
          # the base URL to use for replication, SSL is highly recommended!
          baseUrl: 'https://staging.acme.com/'
          # a key used to authenticate against the replication API, make it long␣
↪and random
          apiKey: 'use a random string here'
```

You can have as many endpoints as you need, and you can define them all at every system or put only the needed ones on every setup, depending on the replication you actually need to run.

3.2 Replication configuration

Replications define what is replicated between systems:

- which data (content, users, assets, . . .)
- the source and target workspace
- endpoints to replicate from/to
- filters to exclude or include items
- . . .

This is done in *Settings.yaml* as follows:

```
      # any number of replication configurations
      replications:
        'to-staging':
          # a human-readable name for the endpoint, optional
          name: 'ame.com Editing to Staging'
          # a human-readable description for the endpoint, optional
          description: 'Deploys from editing to staging'
          # how replication is triggered: publish, manual
          # optional, defaults to manual
          trigger: 'publish'
          # what data is to be replicated: nodes, users, assets
          # optional, defaults to nodes
          types: ['nodes']
          # workspaces to replicate, examples:
          # 'live', 'variant-a:live', '*', 'user-*'
          # keep in mind that base workspaces will be replicated as well, even if␣
↪not included in the list
          workspaces: ['live']
          # source (one of the nodes)
          # optional (defaults to __self__)
```

CHAPTER 4

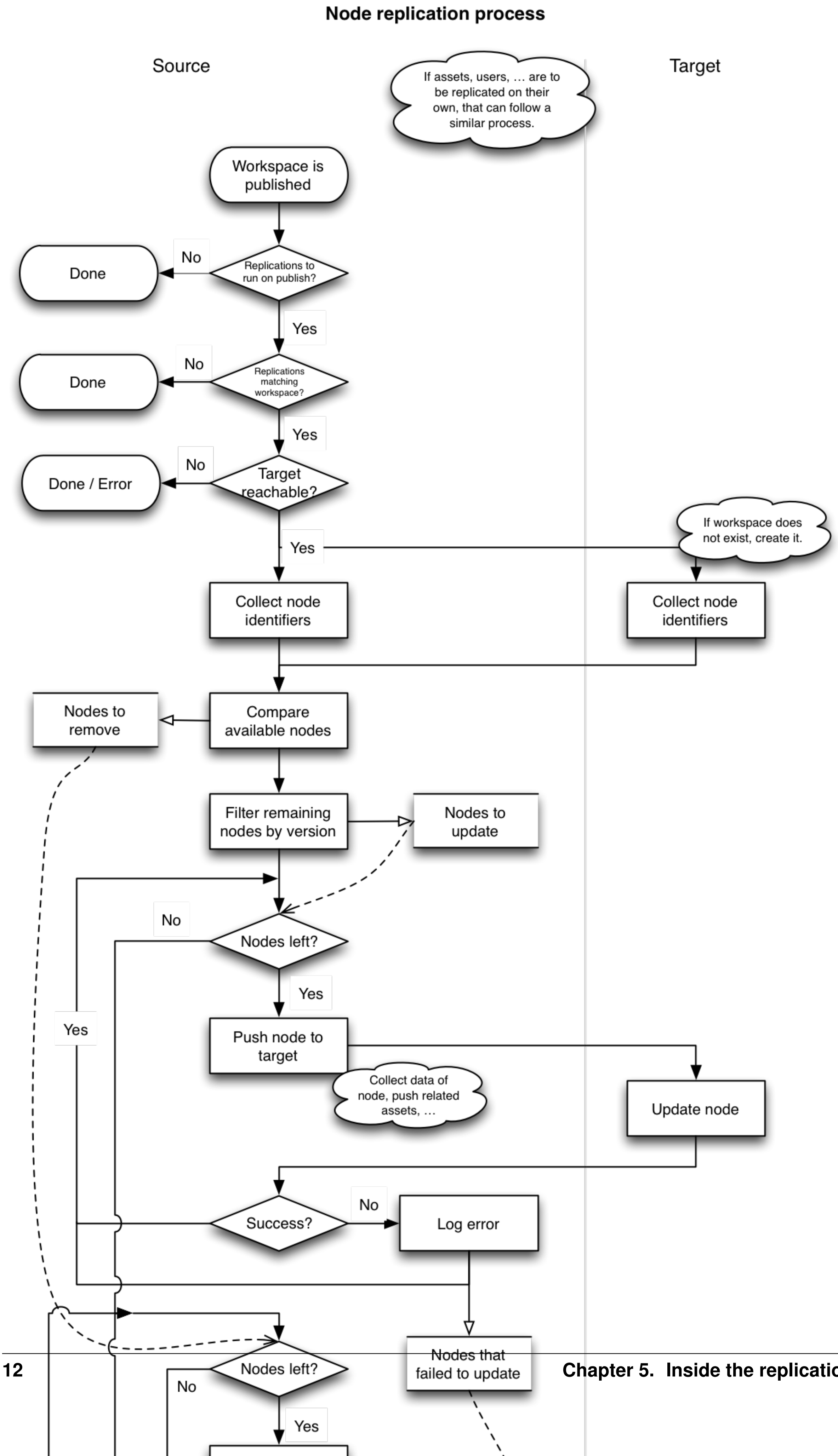
Things to keep in mind

- **Connected items are replicated** even if not explicitly enabled in some cases. If you just enable the replication of nodes, assets used in these nodes will be transferred as well.
- **Settings are never replicated**. Since Neos relies on configuration in YAML and we consider that configuration part of your code, it is assumed you deploy it as such.
- **Packages are never replicated**. Your site package, any add-ons your site uses and the Neos system itself are-again-just code and deployed as such. That means any content relying on specific code being available will only work as expected if that code is available on the target.
- **ContentObject use on nodes is not supported**. If a node has a *ContentObject* attached, this will be ingored.

CHAPTER 5

Inside the replication

Here is what happens:


Node replication process
Source
If assets, users, … are to be replicated on their own, that can follow a similar process.
Target
Workspace is published
Done
No
Replications to run on publish?
Yes
Done
No
Replications matching workspace?
Yes
Done / Error
No
Target reachable?
If workspace does not exist, create it.
Yes
Collect node identifiers
Collect node identifiers
Nodes to remove
Compare available nodes
Filter remaining nodes by version
Nodes to update
No
Nodes left?
Yes
Yes
Push node to target
Collect data of node, push related assets, …
Update node
No
Success?
Log error
Nodes left?
No
Yes
Nodes that failed to update