Nanbando Documentation

Release 0.8.0

Johannes Wachter

Mar 03, 2017

Contents

Nanbando is a simple application to automate website backups. It provides an elegant way to extend and configure the backup parts. Nanbando has built-in support for various storage's and provides easy to use sync and fetch operations. It was built with modularity, extensibility and simplicity in mind.

CHAPTER 1

Features

- Backup/Restore different systems
- Extensibility over events, plugins, commands or presets
- Reuse components of your application to improve your backup
- Connectivity over ssh-connections or remote-storages
- Different level of configuration

CHAPTER 2

Requirements

- PHP: ^5.6 || ^7.0
- ext-xml
- ext-curl
- ext-mbstring
- ext-zip

CHAPTER 3

Contents

Installation

To install the application simply download the executable and move it to the global bin folder.

```
wget http://nanbando.github.io/core/nanbando.phar
wget http://nanbando.github.io/core/nanbando.phar.pubkey
chmod +x nanbando.phar
mv nanbando.phar /usr/local/bin/nanbando
mv nanbando.phar.pubkey /usr/local/bin/nanbando.pubkey
```

After first installation you can update the application with a built-in command.

```
nanbando self-update
```

Note: The executable is signed with a OpenSSL private key. This ensures the origin of the build.

Check the configuration state of your application by using the command `nanbando check`.

Usage

Before we can start to create backup-projects we have to configure the local and remote storage. This global configuration will be shared for all projects of the executing user. The configuration will be written in the file `~/.nanbando.yml`.

```
nanbando:
    storage:
        local_directory: "%home%/nanbando/local"
        remote_service: filesystem.remote
```

```
oneup_flysystem:
    adapters:
        remote:
            local:
                directory: "%home%/nanbando/remote"

    filesystems:
        remote:
            adapter: remote
            alias: filesystem.remote
            plugins:
                - filesystem.list_files
```

Note: In the configuration you can use the parameter `%home%` which points to the home directory of the current user.

The application contains a simple directory backup plugin which we will use in this simple usage example. To start a new backup goto the root directory of your website and create a file named `nanbando.json` which contains the configuration and later also the dependencies for this backup-project.

```
{
    "name": "application",
    "backup": {
        "data": {
            "plugin": "directory",
            "parameter": {
                "directory": "path/to/data/directory"
            }
        }
    },
    "require": {
    }
}
```

After you have created this file you can run following command to configure the local installation with the given configuration. If you have added requirements to the configuration the application will install them into the folder `.nanbando`.

Note: For readonly filesystems you can overwrite the folder `.nanando` by setting the environment variable `NANBANDO_DIR`.

```
php nanbando.phar reconfigure
php nanbando.phar backup
```

The second command will create a new backup zip in the local folder `~/nanbando/local/application/<date>_<environment>_<label>.zip`. The environment and the label are optional and will be omitted if not exists (e.g. without environment `<date>_<label>.zip` or without label and environment `<date>.zip`).

Note: The environment can be set via the config file `"environment"` or env-variable `NANBANDO_ENVIRONMENT`. Example: `NANBANDO_ENVIRONMENT=prod php nanbando.phar backup`.

After this steps you can do following steps:

- `php nanbando.phar restore` - restore a local backup

- `php nanbando.phar push` - push backups to remote storage
- `php nanbando.phar fetch` - fetch a backup on a different machine to restore it there

Connectivity

Nanbando has two options to communicate with other servers.

1. Remotes: Storage to push and fetch backups.
2. Servers: Uses SSH connection to execute commands on a remote server.

Remotes

The remote-storage enhances the user to push backups to or fetch backups from a secure place. Nanbando uses for that the library flysystem which implements a really easy to use abstraction of different remote-storages.

Nanbando brings a lot of build in adapters:

- Amazon S3
- Dropbox
- Azure
- (S)FTP

This adapters can be configured in the *global configuration*.

You can then simply push your backups with `php nanbando.phar push` and fetch with `php nanbando.phar fetch`.

Servers

Servers enables nanbando to connect over ssh with other servers to execute different commands there or download backups directly to your local machine.

You can append the options `--server` to the following commands to execute the specified command there.

```
php nanbando.phar backup --server <server-name>
php nanbando.phar information <backup-name> --server <server-name>
php nanbando.phar get <server-name> <backup-name>
```

The last command will download the specified backup directly in your local directory to restore it locally. Servers can be configured in your *global configuration* or *local project configuration*. It depends if you want to share the configuration or keep it secret.

Configuration

The configuration is devided into two parts - global (optional) and project configuration.

Warning: After changing configuration please run command `reconfigure` to be sure that the configuration will be used for recreating the symfony container.

Global configuration

The global congfiguration is placed in the user home directory. This will be used for all projects used by the user. Put this configuration into `~/.nanbando.yml`.

```
nanbando:
    storage:
        local_directory: "%home%/nanbando"
        remote_service: filesystem.remote

oneup_flysystem:
    adapters:
        remote:
            local:
                directory: "%home%/nanbando/remote"

    filesystems:
        remote:
            adapter: remote
            alias: filesystem.remote
            plugins:
                - filesystem.list_files
```

Note: The configuration documentation for the `oneup_flysystem` can be found on github OneupFlysystemBundle.

For nanbando you have to define the local directory, where the backup command can place the backup archives, and the remote filesystem-service which can be configured in the `oneup_flysystem` extension.

By default the `local_directory` will be set to `%home%/nanbando` and the `remote_service` will be `null`. This leads to local backups will work out of the box but all commands (`fetch`, `push`) which needs the remote-storage will be disabled.

Local project configuration

The local configuration contains the name, backup configuration and the additional *Plugins*.

```
{
    "name": "application",
    "parameters": {
        "directory": "path/to/data/directory"
    },
    "servers": {
        "production": {
            "ssh": {
                "host": "<ip-address>",
                "username": "nanbando",
                "password": "<your-password|true>"
            },
            "directory": "test-data",
            "executable": "../Development/nanbando/bin/nanbando"
        }
    },
    "backup": {
        "data": {
            "plugin": "directory",
            "parameter": {
                "directory": "%directory%"
            }
        }
    },
    "require": {
    }
}
```

The `backup` section can contain as much parts as needed. Each plugin can provide its own `parameter` structure.

Note: The section `parameters` can be used to define global parameters which can be used in the plugin configuration. To import files place them in the `imports` array. This can be used to reuse the symfony-application parameter.

Server Configuration

You can specify the servers-configuration in the local project or global configuration. It depends if you want to share the configuration or keep it secret.

Currently nanbando is able to connected over ssh to the remote server. As authentication method `username & password` or `rsakey file` is available.

```
nanbando:
    servers:
        production:
            ssh:
                host: <ip-address>
                username: nanbando
                password: <your-password|true>
                rsakey:
                    file: <path>
                    password: <your-password|true>
            directory: /var/www
            executable: nanbando
```

As an example this configuration is from the "Global configuration" - but the same as json is also available in "Local project configuration".

The password is optional in the configuration you will be asked for it when nanbando needs it.

Note: You can also use environment variables to configure different values for ssh-connections. Use this variable names: `NANBANDO_SSH_USERNAME`, `NANBANDO_SSH_PASSWORD`, `NANBANDO_SSH_RSAKEY_FILE` and `NANBANDO_SSH_RSAKEY_PASSWORD`.

Plugins

Nanabando was written with extensibility (see *Extending*) in mind. To keep the core as small as possible only one plugin is included in the application. But nanbando also provides optional plugins which can be installed by each backup-project.

Usage

You can use a plugin by adding it to the `nanbando.json` file. There you can configure it like other composer projects in the require section of the file.

```
{
    "name": "application",
    "backup": {
        "su_standard": {
            "plugin": "mysql",
            "parameter": {
                "username": "root",
                "database": "your_database"
            }
        }
    },
    "require": {
        "nanbando/mysql": "^0.1"
    }
}
```

To install this plugin run the `reconfigure` command. It will install the plugin with the embedded-composer and reconfigure the local application. After that you can run the `backup` command to backup the database and `restore` to restore the database.

Available Plugins

This list of plugins is currently quite small but there should be more plugins soonish.

- `nanbando/mysql` - This plugin backups your mysql-database with the `mysqldump` command
- `nabando/jackrabbit` - This plugin will backups your jackrabbit data by exporting into xml
- `nabando/sulu` - This plugin provides presets and auto-detection for sulu applications

Cookbook

The Cookbook covers some applications and how nanbando can be used to backup the data of this applications.

How to backup a Sulu application?

Note: To simplify the backup configuration for sulu-applications you can use the Sulu Plugin.

You can use the following configuration to backup the application using jackrabbit as phpcr backend.

```
{
    "name": "test-application",
    "imports": [
        "app/config/parameters.yml"
    ],
    "parameters": {
        "jackrabbit_uri": "http://localhost:8080/server/"
    },
    "backup": {
        "uploads": {
            "plugin": "directory",
            "parameter": {
                "directory": "var/uploads"
            }
        },
        "database": {
            "plugin": "mysql",
            "parameter": {
                "username": "%database_user%",
                "password": "%database_password%",
                "database": "%database_name%"
            }
        },
        "cmf": {
            "plugin": "jackrabbit",
            "parameter": {
                "jackrabbit_uri": "%jackrabbit_uri%",
                "workspace": "%phpcr_workspace%",
                "path": "/cmf"
            }
        },
        "versions": {
            "plugin": "jackrabbit",
            "parameter": {
                "jackrabbit_uri": "%jackrabbit_uri%",
                "workspace": "%phpcr_workspace%",
                "path": "/jcr:versions"
            }
        },
        "cmf_live": {
            "plugin": "jackrabbit",
            "parameter": {
                "jackrabbit_uri": "%jackrabbit_uri%",
                "workspace": "%phpcr_workspace%_live",
                "path": "/cmf"
            }
        }
    },
    "require": {
        "nanbando/mysql": "^0.1",
        "nanbando/jackrabbit": "^0.1"
    }
}
```

Note: This configuration is optimized for Sulu (minimal) version ^*1.3* with the drafting feature. If you want to backup earlier versions you can omit the backup section *cmf_Live*. For the standard edition you have to adapt the path to the uploads directory.

If you use mysql as data storage for phpcr you can remove the `cmf`, `cmf_live` and `versions` part of the backup.

```
{
    "name": "test-application",
    "imports": [
        "app/config/parameters.yml"
    ],
    "backup": {
        "uploads": {
            "plugin": "directory",
            "parameter": {
                "directory": "uploads"
            }
        },
        "database": {
            "plugin": "mysql",
            "parameter": {
                "username": "%database_user%",
                "password": "%database_password%",
                "database": "%database_name%"
            }
        }
    },
    "require": {
        "nanbando/mysql": "^0.1"
    }
}
```

Extending

Nanbando makes you easy to hook into the application. You can use one of the following possibilities to extend nanbando.

Bundle

All begins with a bundle. The application uses Symfony Bundles to build the environment. The bundles will be discovered by puli. And loaded by composer. So you can guess which parts are mandatory to hook into nanbando.

You can take a look into a already existing nanbando-bundle like Mysql Plugin.

Composer

Create a `composer.json` file and register the repository on packagist.

Puli

Puli uses a simple configuration file in json form so create a basic `puli.json` file with following content.

```
{
    "version": "1.0",
    "name": "<name>",
    "bindings": {
        "<uuid>": {
            "_class": "Puli\\Discovery\\Binding\\ClassBinding",
            "class": "<bundle-class>",
            "type": "nanbando/bundle"
        }
    }
}
```

Bundle Class

A Symfony Bundle is simply a structured set of files within a directory that implement a single feature.

```php
<?php

namespace Acme\TestBundle;

use Symfony\Component\HttpKernel\Bundle\Bundle;

class AcmeTestBundle extends Bundle
{
}
```

In nanbando the bundle can contain a plugin (for backup tasks) or any other extension like event-listener or commands.

Plugins

Plugins are symfony-services tagged with `<tag name="nanbando.plugin" alias="{alias}"/>` inside a *Bundle*. The alias can be used in the Local-*Configuration*.

```xml
 <service id="plugins.mysql" class="Nanbando\Plugin\Mysql\MysqlPlugin">
    <argument type="service" id="output"/>
    <argument type="service" id="temporary_files"/>

    <tag name="nanbando.plugin" alias="mysql"/>
</service>
```

Events

Nanbando issues events which can be listened to by using the standard symfony event dispatcher. You can register a listener in your dependency injection configuration as follows:

```xml
<service id="nanbando_mysql.event_listener.backup" class=
↪"Nanbando\Plugin\Mysql\EventListener\BackupListener">
    <tag name="kernel.event_listener" event="<event_name>" method="methodToCall" />
</service>
```

Backup

The backup fires the event `nanbando.pre_backup` before the process starts and `nanbando.post_backup` after the backup is finished.

The main event is `nanbando.backup` which does the magic and backup the data.

Restore

The backup fires the event `nanbando.pre_restore` before the process starts and `nanbando.post_restore` after the backup is finished.

The main event is `nanbando.restore` which does the magic and restores the data.

Presets

Presets are an easy way to integrate your application (e.g. Sulu Plugin) into the nanbando system. Presets are backup-configurations for specific applications and versions.

Inside a bundle the extension is able to prepend presets for different applications, versions and options.

```
<?php

namespace Nanbando\Plugin\Sulu\DependencyInjection;

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Extension\Extension;
use Symfony\Component\DependencyInjection\Extension\PrependExtensionInterface;

/**
 * Integrates sulu presets into nanbando.
 */
class NanbandoSuluExtension extends Extension implements PrependExtensionInterface
{
    /**
     * {@inheritdoc}
     */
    public function prepend(ContainerBuilder $container)
    {

    $container->prependExtensionConfig(
        'nanbando',
        [
            'presets' => [
                [
                    'application' => 'sulu',
                    'version' => '*',
                    'backup' => [
                        'database' => [
                            'plugin' => 'mysql',
                            'parameter' => [
                                'username' => '%database_user%',
                                'password' => '%database_password%',
                                'database' => '%database_name%',
                            ],
                        ],
                    ],
                ],
            ]
        );
    }

    /**
     * {@inheritdoc}
     */
    public function load(array $configs, ContainerBuilder $container)
    {
    }
}
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search