

# Ningyou Manual

Christian K lker

2019-12-22

## Contents

<b>1</b>	<b>Abstract</b>	<b>3</b>
1.1	Changes . . . . .	3
1.1.1	0.1.1 . . . . .	3
1.1.2	0.1.0 . . . . .	3
<b>2</b>	<b>WARNING</b>	<b>3</b>
<b>3</b>	<b>Introduction</b>	<b>4</b>
3.1	Features . . . . .	4
<b>4</b>	<b>Download Ningyou</b>	<b>4</b>
4.1	Git tar Release . . . . .	4
4.2	Production Repository . . . . .	5
4.3	Transitional Repository . . . . .	5
4.4	Development Repository . . . . .	5
<b>5</b>	<b>Installation</b>	<b>5</b>
5.1	Installation From A Tar Release With <code>ningyou-install</code> . . . . .	5
5.2	Installation From A Tar Release Or Git Repository The Perl Standard Way . . . . .	5
<b>6</b>	<b>Bootstrap - Initial Setup</b>	<b>6</b>
6.1	Simple Setup . . . . .	6
6.2	Setup With Non Standard Worktree . . . . .	6
6.3	Setup With Existing Worktree . . . . .	6
6.3.1	Main Configuration With Worktree As Root Repository (Not Recommended) . . . . .	6
6.3.2	Main Configuration With Worktree As User Repository (Recommended) . . . . .	7
6.4	Setup With Existing Non Standard Worktree . . . . .	7
6.5	Bootstrap Examples . . . . .	8
<b>7</b>	<b>Setup Host Configuration</b>	<b>8</b>

<b>8</b>	<b>Configuration</b>	<b>9</b>
<b>9</b>	<b>Ningyou Classes</b>	<b>10</b>
9.1	Adding A New Ningyou Class . . . . .	10
<b>10</b>	<b>Ningyou Modules</b>	<b>10</b>
10.1	Adding A Ningyou Module . . . . .	10
<b>11</b>	<b>Applying Changes To The System</b>	<b>12</b>
11.1	Understanding What Would Be Applied . . . . .	12
<b>12</b>	<b>Dependencies</b>	<b>13</b>
12.1	Development Dependencies . . . . .	13
12.2	Build Dependencies . . . . .	13
12.3	Runtime Dependencies . . . . .	13
12.3.1	Dependencies For Provider CPAN . . . . .	13
12.4	Runtime Recommendations . . . . .	13
<b>13</b>	<b>Syntax</b>	<b>13</b>
13.1	Examples . . . . .	14
<b>14</b>	<b>Configuration Explained</b>	<b>14</b>
<b>15</b>	<b>Advanced Configuration</b>	<b>18</b>
15.1	Package . . . . .	18
15.2	Cpan . . . . .	19
15.2.1	Non Standard Perl Locations . . . . .	21
<b>16</b>	<b>Provider</b>	<b>22</b>
16.1	Cpan . . . . .	22
16.2	Directory . . . . .	22
16.3	File . . . . .	23
16.4	Git . . . . .	23
16.5	Link . . . . .	23
16.6	Nop . . . . .	24
16.7	Package . . . . .	24
16.8	Rsync . . . . .	24
16.9	Version . . . . .	25
16.10	Limitations . . . . .	25
<b>17</b>	<b>Possible Future Improvements</b>	<b>25</b>
<b>18</b>	<b>Development</b>	<b>25</b>
18.1	Profiling . . . . .	25
<b>19</b>	<b>DISCLAIMER OF WARRANTY</b>	<b>26</b>
<b>20</b>	<b>Author</b>	<b>26</b>

21 Copyright And License	26
--------------------------	----

22 More Information About GNU General Public License	27
--	----

# 1 Abstract

This documents describes the installation, set up, configuration and usage of the deployment tool **Ningyou** in detail. Some simple parts are omitted for didactically reasons and can be found in INSTALL.md and USAGE.md, because these documents are usually read first.

## 1.1 Changes

Version	Date	Author	Notes
0.1.1	2019-12-22	Christian Külker	add versions in YAML front matter
0.1.0	2019-12-12	Christian Külker	initial release

### 1.1.1 0.1.1

- add versions in YAML front matter

### 1.1.2 0.1.0

- initial release

# 2 WARNING

```
//=====\\
|| This software is in ALPHA state, not tested and contains many bugs.  ||
|| You are encouraged to help and report them. However be aware, that    ||
|| this software is intended to run as root and as such it can DAMAGE    ||
|| your system. You may experience the LOSS OF DATA. You are using the  ||
|| software at your own risk! Please read also this document.           ||
\\=====\\
```

## 3 Introduction

Deploy frameworks are usually one of two kinds: deterministic or object oriented. The feature of object oriented frameworks is that dependencies can be inherited. The drawback is often that it is very hard to predict the outcome and correctness of the deployment.

**Ningyou** tries to merge the best out of this two worlds: a) it is group oriented and dependency based with an easy configuration similar to existing tools. It produces exactly the same actions from the same configuration on the same machine architecture. Say, it is predictable. On top of it **Ningyou** can provide from this a shell script that can be use on a similar second machine without **Ningyou**, or just look at it to understand what will be done in a predictable way.

### 3.1 Features

- Debian package deployment
- Git repository deploy support
- CPAN module deploy support
- Simple configuration files
- Template::Toolkit language support in configuration files
- Deployment of directories, files, links
- Deployment of directory content via rsync

## 4 Download Ningyou

To download the software the following methods are available:

- git tar release
- git production repository: branch master
- git transitional repository: branch staging
- git development repository: branch develop

### 4.1 Git tar Release

```
wget https://github.com/ckuelker/ningyou/archive/v0.1.0.tar.gz
tar xvzf v0.1.0.tar.gz
```

This will create the `ningyou-0.1.0` directory.

## 4.2 Production Repository

This is intended for developers as well as users, who would like to use the latest stable release and prefer to update the worktree via git.

```
git clone --single-branch --branch master https://github.com/ckuelker/ningyou.git
```

Or

```
git clone https://github.com/ckuelker/ningyou.git
```

## 4.3 Transitional Repository

Only need for maintenance of **Ningyou**.

```
git clone --single-branch --branch staging https://github.com/ckuelker/ningyou.git
```

## 4.4 Development Repository

Needed for developers.

```
git clone --single-branch --branch develop https://github.com/ckuelker/ningyou.git
```

# 5 Installation

There are different types of installation methods:

- installing via ningyou-install (recommended)
- installing via Makefile.PL

## 5.1 Installation From A Tar Release With **ningyou-install**

```
ningyou-0.1.0/bin/ningyou-install
```

See also INSTALL.md for the complete procedure.

## 5.2 Installation From A Tar Release Or Git Repository The Perl Standard Way

```
perl Makefile.PL  
make test  
make install
```

This will install the **ningyou** executable and libraries into default locations.

## 6 Bootstrap - Initial Setup

### 6.1 Simple Setup

To kick start the installation configuration **Ningyou** can help you. As root execute

```
cd /srv
ningyou bootstrap
```

This will create `~/.gitconfig` if not present or use it if present. It will create the global **Ningyou** configuration `~/.ningyou.ini` in the home directory. This file will reference the **worktree** for **Ningyou** with the main configuration of software to install. You can decide where this **worktree** should be installed. Without parameters it will be created in the current directory under the name **deploy**. So the above will create `/srv/deploy` as a **worktree** git repository.

### 6.2 Setup With Non Standard Worktree

Usually the **worktree** is called **deploy** however that is not mandatory. To name it differently (for example `my-ningyou-worktree`) a parameter can be added to bootstrap like so:

```
ningyou bootstrap /srv/my-ningyou-worktree
```

It is recommended to use a full qualified path, not a relative path.

### 6.3 Setup With Existing Worktree

If there is already one host deployed by **Ningyou** the repository (named `deploy.git` for example) can be reused. While **Ningyou** is usually executed as **root**, the **worktree** repository does not need to be used as **root**, as **Ningyou** do not commit to the repository. The following to examples show the difference.

#### 6.3.1 Main Configuration With Worktree As Root Repository (Not Recommended)

As root:

```
cd /srv
git clone GIT_URL_TO_YOUR_NINGYOU_REPOSITORY/deploy.git
ningyou --main-configuration-only bootstrap
```

The above command only creates `/root/.ningyou.ini` pointing to `/srv/deploy`.

### 6.3.2 Main Configuration With Worktree As User Repository (Recommended)

As root:

```
cd /srv/  
mkdir deploy  
chown $USER.$USER deploy
```

As user:

```
cd /srv  
git clone GIT_URL_TO_YOUR_NINGYOU_REPOSITORY/deploy.git
```

As root:

```
ningyou --main-configuration-only bootstrap  
mv ~/.ningyou.ini /home/USER/  
ln -s /home/USER/.ningyou.ini /root/.ningyou.ini
```

The above `bootstrap` command creates `root/.ningyou.ini` pointing to `/srv/deploy`, but it creates not a git repository. The git repository is cloned by the user and managed by the user (recommended). To use `ningyou` by a user and `root` `ningyou` needs to have access to the same configuration.

Now `root` moves the configuration `.ningyou.ini` to the user directory and links it to `root/.ningyou.ini`, so that the user has the same information as `root`. In that way `root` and the user can execute `ningyou` with exactly the same information. In this example the file is owned by `root`, so that only `root` can change it. If `root` changes the configuration it will be automatically available to the user.

## 6.4 Setup With Existing Non Standard Worktree

Lets assume a non standard `worktree` at `/root/system-administration` exists and should be used:

```
ningyou --main-configuration-only bootstrap /root/system-administration
```

This will create `~/.ningyou.ini` with a `worktree` pointing to `/root/system-administration`

A non standard `worktree` can be use also in the exact same way as the previous section explained:

As root:

```
cd /srv/  
mkdir my-ningyou-worktree  
chown $USER.$USER my-ningyou-worktree
```

As user:

```

cd /srv
git clone GIT_URL_TO_YOUR_NINGYOU_REPOSITORY/my-ningyou-worktree.git

As root:

ningyou --main-configuration-only bootstrap /srv/my-ningyou-worktree
mv ~/.ningyou.ini /home/USER/
ln -s /home/USER/.ningyou.ini /root/.ningyou.ini

```

## 6.5 Bootstrap Examples

Initialization with `ningyou bootstrap`:

```

ningyou bootstrap                : all
ningyou bootstrap --main-configuration-only : ~/.ningyou.ini only
ningyou bootstrap PATH           : all with worktree PATH
ningyou bootstrap --main-configuration-only PATH : ~/.ningyou.ini only but
                                                reference worktree PATH

```

## 7 Setup Host Configuration

This section is assuming a working tree is in `/srv/deploy`. The `bootstrap` action creates the host configuration automatically for the current host. The configuration can be found in `/srv/deploy/host.domain.tld.ini`.

If the `worktree` git repository is reused on a different host, the host configuration for that specific host has to be created in the `worktree`. Assuming the host is called `h.example.com`, the configuration should be in `/srv/deploy/h.example.com.ini`

A simple `h.example.com.ini` may look like this:

```

[version]
project=0.1.0
configuration=0.1.0
file=0.1.0

[global]
ningyou=1

[debian-gnu-linux-10-buster-amd64-x86_64]

[h.example.com]

```

The above file assumes there is a global `ningyou module`. This should have been created during the `bootstrap` process.



## 8 Configuration

A configuration for the current operating system is needed. The description of the current installed OS is derived from `facter's lsbdistdescription`. The description of Debian GNU/Linux 8.11 (jessie) gives `debian-gnu-linux-8.11-jessie` as OS description. For Debian Buster it will be `debian-gnu-linux-10-buster-amd64-x86_64`.

In case the host is called `host.domain.tld`, there are three default classes for configuration: `[global]`, `host.domain.tld` and for example `debian-gnu-linux-10-buster-amd64-x86_64`. These sections can be found in `host.domain.tld.ini` and correspond to the directories with the same name.

OS specific configuration should go into `debian-gnu-linux-10-buster-amd64-x86_64` and host specific should go into `host.domain.tld`.

Under each directory there is - or need to be created - a `modules` directory where **Ningyou** *modules* live. Each *module* has its own directory, for example the `zsh` *module* directory tree might look like this:

```
zsh
  files
    zshrc
  manifests
    zsh.ini
```

The `zsh.ini` is the module configuration. This configuration can control weather and what files from the `files` directory are distributed. This is the `zsh` content:

```
[version:zsh]
; Ningyou Project version - changed by Ningyou
project=0.1.0
; Ningyou Configuration Space version - changed by Ningyou
configuration=0.1.0
; version of this file - change this when you update the file
file=0.1.0
```

```
[package:zsh]
ensure=latest
```

```
[file:/root/.zshrc]
source=ningyou:///global/modules/zsh/files/zshrc
mode=640
owner=root
group=root
ensure=latest
require=global:package:zsh
checksum=80afb055812d5449dc3c25e317f52654
```

As of **Ningyou** 0.1.0 the `[version:zsh]` section is optional. If present it will be printed if the `--verbose` flag is used. The section `[package:zsh]` together with the attribute `ensure` and its value `latest` makes sure that the latest `zsh` package is installed.

The next section `[file:/root/.zshrc]` makes sure that the `zsh` configuration `global/modules/zsh/files/zshrc` will be copied to `/root/.zsh`.

## 9 Ningyou Classes

As of 0.1.0 **Ningyou** classes are supported. They are basically artificial groups.

### 9.1 Adding A New Ningyou Class

Edit `~/.ningyou.ini` and add this section

```
[class]
server=0
client=1
```

This adds the `server` class and enables it, while it also adds a `client` class that is not enabled. Before using `ningyou` a matching directory structure should be added to the worktree.

```
mkdir -p /srv/deploy/server/modules
mkdir -p /srv/deploy/client/modules
```

## 10 Ningyou Modules

### 10.1 Adding A Ningyou Module

To add a new **Ningyou** module a decision has to be made:

1. Is the software specific to the host?
2. Is the software specific to the distribution?
3. Is the software specific to a class?
4. Is the software not specific.

In relation to the answers of the above questions a location for the new module has to be chosen:

1. `host.domain.tld` - example: `h.example.com`
2. `distribution` - example: `debian-gnu-linux-10-buster-amd64-x86_64`
3. `group` - example: `server`
4. `global`

Lets assume for this example the software **screen** is not host or distribution specific and should be installed on all hosts. Change to the directory:

```
cd /srv/deploy/global/modules
```

Create the module boiler plate:

```
ningyou module screen
setting up module [screen]
we are in cwd [/srv/deploy/global/modules]
the current working directory is OK
create module [screen] ...
make directory [screen] ...
[PASS] mkdir -p screen
make directory [screen/files] ...
[PASS] mkdir -p screen/files
make directory [screen/manifests] ...
[PASS] mkdir -p screen/manifests
created file [screen/manifests/screen.ini], please edit
```

This will create the files and directories:

```
screen
  files
  manifests
    screen.ini
```

Then edit `screen/manifests/screen.ini` and change it like:

```
[version:screen]
; Ningyou Project version - changed by Ningyou
project=0.1.0
; Ningyou Configuration Space version - changed by Ningyou
configuration=0.1.0
; version of this file - change this when you update the file
file=0.1.0

;[nop:screen]
; the 'nop' provider provides a 'no operation' - nothing
; can be used to check (via debug) if configuration section is actually used
; it should be disabled by commenting out, because it will be always pending
;debug=Message for module [screen] debug

[package:screen]
ensure=latest
```

The `[nop:screen]` is only for development purpose.

## 11 Applying Changes To The System

### 11.1 Understanding What Would Be Applied

Lets assume the following status:

```
ningyou status
```

```
# Ningyou v0.1.0 at h.example.com with /srv/deploy/h.example.com.ini
# Status modules(s) all in /srv/deploy
class:module                                     enabled status
=====
global:ningyou                                   [YES]    [DONE]
global:tree                                       [YES]    [TODO]
global:vim                                        [YES]    [DONE]
```

It already could be guessed that changes of the system would effect the global tree module as it has the status [TODO]. Executing the `script` action show what and how changes would be applied in detail:

```
ningyou script
```

```
#!/bin/bash
# +-----+
# | Ningyou script                                     |
# | |                                                 |
# | | This script was created with the Ningyou script action |
# | |                                                 |
# | | Date: 2019-12-12                                   |
# | |                                                 |
# +-----+
#
# Ningyou project version: 0.1.0
# Ningyou script version: 0.1.0
# Worktree: /srv/deploy
# Configuration: /srv/deploy/h.example.com.ini
# Command Line (approx): /usr/local/bin/ningyou script all
#
# Script commands for h.example.com:
aptitude --assume-yes install tree
```

So this will install the `tree` package.

## 12 Dependencies

### 12.1 Development Dependencies

libdevel-nytpref-perl

### 12.2 Build Dependencies

make

### 12.3 Runtime Dependencies

```
libapt-pkg-perl          # comment=Deploy::Ningyou::Provider::Package
libmodule-pluggable-perl
perl-doc
factor                  # maybe replaced by Sys::Facter in the future
libfile-touch-perl
libtemplate-perl
libconfig-ini-perl
libcapture-tiny-perl
libfile-dircompare-perl
libgraph-perl
liblist-compare-perl
git
```

#### 12.3.1 Dependencies For Provider CPAN

cpanminus

### 12.4 Runtime Recommendations

```
libmodule-runtime-perl (libmodule-pluggable-perl)
libmodule-require-perl (libmodule-pluggable-perl)
```

## 13 Syntax

**Ningyou** can be invoked from the command line

ningyou [OPTIONS] <ARGUMENT> [<SCOPE>]

OPTIONS:

```

ningyou [--main-configuration-only] bootstrap [PATH_WORKTREE]
ningyou [--help|--man|--version]
ningyou [--verbose] list|show|script|apply [<SCOPE>]
ningyou module <NAME>

```

SCOPE:

```
|all|<NAME_OF_MODULE>|<NAME_OF_MODULE> <NAME_OF_MODULE> ...
```

## 13.1 Examples

```

ningyou --version                : print Ningyou project version number

ningyou help                    : show brief help message (same as --help)
ningyou --help                  : show brief help message
ningyou man                     : show man page (same as --man)
ningyou --man                   : show man page

ningyou bootstrap               : install in `cwd` a ~/.ningyou.ini
                                initialize ~/.ningou.ini and worktree

ningyou list                    : list enabled modules

ningyou status                  : print brief information about modules
ningyou --verbose status        : print information about all section
ningyou --verbose status global:zsh : print information about zsh module only

ningyou script                  : print bash script
ningyou --verbose script        : print bash script with explanations

ningyou apply                   : execute commands (see ningyou script)
ningyou --verbose apply         : execute commands with explanations

```

## 14 Configuration Explained

The configuration space resides in the *worktree*. The *worktree* is set when executing `ningyou bootstrap` and it's value is recorded in `~/.ningyou.ini`.

```

; +-----+
; | ningyou.ini => ~/.ningyou.ini |
; |                               |
; | Main configuration for Ningyou |
; |                               |
; | Version: 0.1.0 (Change also inline: [version] file=) |
; |                               |

```

```

; | Changes:
; |
; | 0.1.0 2019-03-28 Christian Kuelker <c@c8i.org>
; |     - initial release
; |
; +-----+
;
[version]
; Ningyou Project version - changed by Ningyou
project=0.1.0
; Ningyou Configuration Space version - changed by Ningyou
configuration=0.1.0
; version of this file - change this when you update the file
file=0.1.0

[global]
    worktree=/srv/deploy

[system]
    fqhn=h.example.com

[os]
    distribution=debian-gnu-linux-10-buster-amd64-x86_64
    ; package manager cache time to live, default 3600 = 1h
    pm_cache_ttl=3600

```

If the *worktree* lives for example in */srv/deploy* than a minimal working tree with a **global:zsh** *module* on the host *h.example.com* with a Debian **Stretch** operating system would be:

```

/srv/deploy
  debian-gnu-linux-10-buster-amd64-x86_64
  global
    modules
      zsh
        files
          zshrc
        manifests
          zsh.ini
    h.example.com
    h.example.com.ini

```

The directories *h.example.com*, *debian-gnu-linux-10-buster-amd64-x86\_64* and *global/modules* are created by *ningyou bootstrap*.

The configuration for the **global:zsh** *module* can be created like this:

```
cd /srv/deploy/global/modules
```

ningyou module zsh

This will create the files and directories: `zsh/files` and `zsh/manifests/zsh.ini`.

The content of `zsh.ini` looks like this:

```
; +-----+
; | modules/zsh/manifests/zsh.ini |
; |                               |
; | Configuration for a Ningyou module. |
; |                               |
; | Version: 0.1.0 (Change also inline: [version] file=) |
; |                               |
; | Changes: |
; |                               |
; | 0.1.0 2019-04-19 Christian Kuelker <c@c8i.org> |
; |     - initial release |
; |                               |
; +-----+
;
[version:zsh]
; Ningyou Project version - changed by Ningyou
project=0.1.0
; Ningyou Configuration Space version - changed by Ningyou
configuration=0.1.0
; version of this file - change this when you update the file
file=0.1.0

[nop:zsh]
; the 'nop' provider provides a 'no operation' - nothing
; can be used to check (via debug) if configuration section is actually used
debug=NOP zsh

; [package:zsh]
```

To enable this *module* configuration aka to install the `zsh` package: fist uncomment `; [package:zsh]` like so `[package:zsh]` and add the line to the `[global]` section of `h.example.com.ini`.

`zsh=1`

Comment out the `[nop:zsh]` and `debug=NOP zsh`

The `h.example.com.ini` configuration is simple:

```
; +-----+
; | h.example.com.ini |
; | |
; | Configuration for one host |
; | |
```



```

; | Version: 0.1.1 (Change also inline: [version] file=)
; |
; | Changes:
; |
; | 0.1.1 2019-04-19 Christian Kuelker <c@c8i.org>
; |     - enable global:zsh
; |
; | 0.1.0 2019-04-11 Christian Kuelker <c@c8i.org>
; |     - initial release
; |
; +-----+
[version]
; Ningyou Project version - changed by Ningyou
project=0.1.0
; Ningyou Configuration Space version - changed by Ningyou
configuration=0.1.0
; version of this file - change this when you update the file
file=0.1.1

[global]
; distribution independent modules
; active modules = 1
; inactive modules = 0
;     zsh=1
[debian-gnu-linux-10-buster-amd64-x86_64]
; distribution debian-gnu-linux-10-buster-amd64-x86_64 dependent modules

[h.example.com]
; host h.example.com dependent modules

As zsh needs a configuration file ~/.zshrc that needs to be added a zsh con-
figuration file to /srv/deploy/gobal/modules/zsh/files/ and a file provider
to the configuration. The complete configuration looks like this:

; +-----+
; | modules/zsh/manifests/zsh.ini
; |
; | Configuration for a Ningyou module.
; |
; | Version: 0.1.1 (Change also inline: [version] file=)
; |
; | Changes:
; |
; | 0.1.1 2019-04-19 Christian Kuelker <c@c8i.org>
; |     - add package:zsh
; |     - add file:/root/.zshrc
; |

```

```

; | 0.1.0 2019-04-19 Christian Kuelker <c@c8i.org> |
; | - initial release |
; | |
; +-----+
;
[version:zsh]
; Ningyou Project version - changed by Ningyou
project=0.1.0
; Ningyou Configuration Space version - changed by Ningyou
configuration=0.1.0
; version of this file - change this when you update the file
file=0.1.1

;[nop:zsh]
; the 'nop' provider provides a 'no operation' - nothing
; can be used to check (via debug) if configuration section is actually used
;debug=NOP zsh

[package:zsh]
[file:/root/.zshrc]
    source=ningyou:///global/modules/zsh/files/zshrc
    mode=640
    owner=root
    group=root
    ensure=latest
    require=global:package:zsh
    checksum=80afb055812d5449dc3c25e317f52653

```

## 15 Advanced Configuration

### 15.1 Package

The main attribute for packages is **ensure=**, with either the value ‘present’ or ‘latest’, while ‘missing’ is also possible. Additional attributes ‘version’ and ‘source’ are possible and mutual exclusive. For ‘version’ the packages of that version needs to be in the package repository. The same functionality can be archived by ‘ensure=present’ if the latest version is the desired version. If the latest version is not the desired version and if this package is not in the repository it can be provided by the ‘source’ attribute. However this is not recommended and should be considered a method of last resort. The reason is that the installation is not done via **aptitude** it is via **dpkg** that do not check for dependencies nor records its installation.

## 15.2 Cpan

```
[% TAR='Dist-Zilla-Plugin-PerlTidy-0.21.tar.gz' %]
```

```
[cpan:Dist::Zilla::Plugin::PerlTidy]
source=ningyou:///global/modules/devel/files/[% TAR %]
ensure=latest
require=global:package:libpath-iterator-rule-perl
environment=/srv/env/perl
download=https://cpan.metacpan.org/authors/id/F/FA/FAYLAND/[% TAR %]
```

This will download the source from the address provided with the ‘download’ attribute, if not already downloaded, and store it under the path provided by the attribute ‘source’. If ‘source’ by it self is a URL it will be downloaded and installed each time and not stored. In the above example [% TAR %] is a Template::Toolkit expression.

```
perl-5.24.1
  bin
  lib
    i486-linux-gnu-thread-multi
    perl5
      Dist
        Zilla
          App
            Command
              perltidy.pm
            Plugin
              PerlTidy.pm
          x86_64-linux-gnu-thread-multi
            auto
              Dist
                Zilla
                  Plugin
                    PerlTidy
              perllocal.pod
      man
        man3
          Dist::Zilla::App::Command::perltidy.3pm
          Dist::Zilla::Plugin::PerlTidy.3pm
```

The ‘source’ attribute can be a **Ningyou** URL, module name, distribution file, local file path, HTTP URL or git repository URL. The following will work as expected:

```
source=Plack 1)a)
source=Plack/Request.pm 1)a)
```

```

source=MIYAGAWA/Plack-1.0000.tar.gz                1)b)
source=/path/to/Plack-1.0000.tar.gz                2)b)
source=ningyou:///global/modules/NINGYOU_MODULE/file/Plack-1.0000.tar.gz 2)b)
source=ningyou:///~/Plack-1.0000.tar.gz            2)b)
source=http://cpan.metacpan.org/authors/id/M/MI/MIYAGAWA/Plack-0.9990.tar.gz 1)b)
source=git://github.com/plack/Plack.git            3)b)

```

- 1) will be downloaded from CPAN.
- 2) will be fetched from local file system. The 'download' attribute can be used to store the archive at the source place
- 3) will be cloned from git
- a) will install the latest version and update current version if 'ensure=latest' otherwise not
- b) will install the latest version and update current version even if 'ensure=present'

Additionally, the notation “~” and “@” can be used to specify version for a given module. “~” specifies the version requirement in the CPAN::Meta::Spec format, while “@” pins the exact version, and is a shortcut for “~”== VERSION“.

```

source=Plack~1.0000                                # 1.0000 or later
source=Plack~">= 1.0000, < 2.0000"                 # latest of 1.xxxx
source=Plack@0.9990                                # specific version. same as Plack~"== 0.9990"

```

The version query including specific version or range will be sent to MetaCPAN to search for previous releases. The query will search for BackPAN archives by default, unless the “-dev” option is specified, in which case, archived versions will be filtered out.

For a git repository, a branch, tag, or commit SHA can be specified to build. The default is “master”

```

source=git://github.com/plack/Plack.git@1.0000      # tag
source=git://github.com/plack/Plack.git@devel       # branch

```

In case the latest module is wanted use this configuration:

```

[cpan:Dist::Zilla::Plugin::PerlTidy]
source=Dist::Zilla::Plugin::PerlTidy
ensure=latest

```

In case the specific version is wanted:

```

[cpan:Dist::Zilla::Plugin::PerlTidy]
source=Dist::Zilla::Plugin::PerlTidy@0.20
ensure=present

```

However sometimes the specific version is not possible to fetch, in this case:

```

[cpan:Dist::Zilla::Plugin::PerlTidy]
source=ningyou:///global/modules/devel/files/Dist-Zilla-Plugin-PerlTidy-0.20.tar.gz

```

```
ensure=present
```

```
download=https://cpan.metacpan.org/authors/id/B/BI/BINARY/Dist-Zilla-Plugin-PerlTidy-0.20.ta
```

Limitation: When changing the ‘source’ and ‘download’ to newer versions while ‘ensure=present’ remains, **Ningyou** will not update the package, because one old version is still present. If you to update to a specific version you have to define specific ‘source’ and ‘download’ and set ‘ensure=latest’, this will update the package to the version specified in the source file, but not higher, even though there might be a newer version on CPAN.

If ‘ensure=missing’ the CPAN module will be uninstalled. This works only for some sources. For files it do not work. The following works:

```
[cpan:Dist::Zilla::Plugin::PerlTidy]
source=Dist::Zilla::Plugin::PerlTidy
ensure=missing
```

Make sure that your environment sets the PERL\_LOCAL\_LIB\_ROOT variable.

### 15.2.1 Non Standard Perl Locations

It is possible to give an ‘environment’ attribute. The environment specified in this file will be sourced in before installation, update or remove. By this more than one Perl distribution can be managed.

```
environment=/srv/env/perl-5.24.1
```

The file /srv/env/perl-5.24.1 contains:

```
path_add_before(){
    if [ -d "$1" ] && [[ ":$PATH:" != *":$1:*" ]]; then
        path=($1 $path)
    fi
}
DIR=/srv/perl-5.24.1
for d in bin lib man; do
    if [ ! -d $DIR/$d ]; then mkdir -p $DIR/$d; fi
done
```

```
PL=lib/perl5
if [ ! -d $DIR/$PL ]; then mkdir -p $DIR/$PL; fi
```

```
PV=lib/i486-linux-gnu-thread-multi
if [ ! -d $DIR/$PV ]; then mkdir -p $DIR/$PV; fi
```

```
export PERL_MB_OPT="--install_base $DIR"
export PERL_MM_OPT="INSTALL_BASE=$DIR"
export PERL5LIB="$DIR/$PL:$DIR/$PV:$PERL5LIB"
```

```
path_add_before $DIR/bin
export PERL_LOCAL_LIB_ROOT=$DIR
```

## 16 Provider

This section describes the module configuration space for each *provider*. This information is needed if one would write one's own *module* configuration or change existing.

### 16.1 Cpan

SECTION

[cpan:PERL\_MODULE]

MANDATORY ATTRIBUTES

ensure=latest|present|missing  
source=PATH\_TO\_TAR\_ARCHIVE|...

OPTIONAL ATTRIBUTES

download=URL  
environment=<PATH\_TO\_SOURCE\_IN\_ENVIRONMENT>  
require=CLASS:PROVIDER:DESTINATION

### 16.2 Directory

SECTION

[directory:/path/to/directory]

MANDATORY ATTRIBUTES

ensure=present|missing|purged

OPTIONALLY ATTRIBUTES

owner=USER	REM:default root
group=GROUP	REM:default root
mode=0755 755	REM:default 0700
require=CLASS:PROVIDER:DESTINATION	

WARNING: The 'ensure' *attribute* with the value 'purged' removes the directory recursively. You have been warned.

## 16.3 File

### SECTION

[file:/path/to/file]

### MANDATORY ATTRIBUTES

ensure=latest|present|missing REM:for latest checksum is mandatory

### OPTIONALLY ATTRIBUTES

REM:if source do not exists an empty file will be created  
source=/tmp/FILE - absolute  
source=ningyou://zshrc - module  
source=ningyou:///global/modules/zsh/files/zshrc - worktree  
source=~/.zshrc - home  
owner=USER REM:default root  
group=GROUP REM:default root  
mode=0644|644 REM:default 0600  
require=CLASS:PROVIDER:DESTINATION  
checksum=md5 REM:for checksum source and latest  
is mandatory

## 16.4 Git

The check for unclean repositories will not trigger on unchecked files on purpose:  
adding files to non managed git repositories is possible.

### SECTION

[git:/path/to/directory]

### MANDATORY ATTRIBUTES

source=git repository location  
ensure=present|latest|missing

### OPTIONALLY ATTRIBUTES

comment=test  
group=GROUP REM:default root (recursive)  
mode=0750|750 REM:default 0750 dir only  
owner=USER REM:default root (recursive)  
require=CLASS:PROVIDER:DESTINATION

## 16.5 Link

### SECTION

[link:/path/to/link]

MANDATORY ATTRIBUTES  
type=symbolic|hard REM:default symbolic  
ensure=present|missing REM:default present  
source=path/to/(file|directory)

OPTIONALLY ATTRIBUTES  
comment=text  
require=CLASS:PROVIDER:DESTINATION

## 16.6 Nop

SECTION  
[nop:<MODULE>]

MANDATORY ATTRIBUTES  
debug=<MESSAGE> | define a debug message

## 16.7 Package

SECTION  
[package:PACKAGE\_NAME]

MANDATORY ATTRIBUTES  
ensure=present|missing|latest REM:default present

OPTIONALLY ATTRIBUTES  
source=path/to/package  
require=CLASS:PROVIDER:DESTINATION

## 16.8 Rsync

SECTION  
[rsync:/path/to/directory]

MANDATORY ATTRIBUTES  
source=/path/to/DIRECTORY

OPTIONALLY ATTRIBUTES  
comment=some text about rsync  
dry=1  
group=GROUP REM:default Ningyou group  
itemize=1



```

mode=0755|755                REM:default from system
owner=USER                    REM:default Ningyou user
purge=1
require=global:package:vim
require=CLASS:PROVIDER:DESTINATION

```

## 16.9 Version

### SECTION

```
[version:<MODULE>]           | provider head
```

### MANDATORY ATTRIBUTES

```

project=<NUMBER>              | project version number handled by Ningyou
configuration=<NUMBER>        | configuration version number handled by Ningyou
file=><NUMBER>                  | file version number handled by user

```

## 16.10 Limitations

- As of now **Ningyou** only supports git repositories for its working tree and do not interact via git with its working tree. There is the git provider for Ningyou. There are no expectations to use it for working tree itself. Even though this seems a limitation it is a feature for now.
- Users and home directories are not handled as meta data in **Ningyou**, but as a workaround this feature can be used on a per module basis with `Template::Toolkit`.
- While most changes are applied in the first invocation of **ningyou**, there are corner cases in which a configuration triggers other actions in a second run. Ideas and pull requests are welcome.

## 17 Possible Future Improvements

- Write `Provider::Tar`

## 18 Development

### 18.1 Profiling

The New York Times profiler is an easy start to understand the bottle necks of **Ningyou**.

```
aptitude install libdevel-nytprof-perl>
mkdir /tmp/pf
cd /tmp/pf
NINGYOU_DEBUG=/tmp/ningyou.debug perl -d:NYTProf ningyou --verbose apply
BROWSER=firefox nytprofhtml --open
firefox /tmp/pf/nytprof/index.html
```

## 19 DISCLAIMER OF WARRANTY

BECAUSE THIS SOFTWARE IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE SOFTWARE, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE SOFTWARE AS PERMITTED BY THE ABOVE LICENSE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE SOFTWARE TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## 20 Author

Christian Külker c@c8i.org

## 21 Copyright And License

This software is Copyright (c) 2013, 2014, 2019 by Christian Külker.

This is free software, licensed under:

The GNU General Public License, Version 2, June 1991

## 22 More Information About GNU General Public License

This software is copyright (C) 2013 by Christian Külker

This software is copyright (C) 2014 by Christian Külker

This software is copyright (C) 2019 by Christian Külker

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA