niteoweb.fabfile Documentation

Release 1.0

NiteoWeb Ltd.

March 10, 2014

This package contains a selection of Fabric command we at NiteoWeb use all the time. By sharing them online we hope to save someone some time researching how certain Plone-oriented tasks are performed with Fabric. Commands contain some hardcoded internal stuff, so they are not really usable out-of-the-box, more so as a point of reference. If there is interest, we'll rewrite them to be more reusable.

- Source code @ GitHub
- Releases @ PyPI
- Sphinx docs @ ReadTheDocs

CHAPTER 1

Table of Contents

1.1 Headquarters (HQ) server

Nothing here yet ...

1.2 Projects server

A Projects server is a server that runs your Plone projects. This is a prerequisite to have before you can run any commands from the *Project* group of commands.

1.2.1 Sample fabfile

Below is a `fabfile.py.in` buildout template that uses commands from *Server* group to set up a Projects server (based on Ubuntu 10.04).

```
import os
from fabric.api import env
from niteoweb.fabfile.server import *

env.path = os.getcwd()
env.hosts = ['${ips:server}']
env.hostname = '${config:hostname}'
env.shortname = '${config:shortname}'
env.temp_root_pass = '${pass:temp_root}'
env.server_ip = '${ips:server}'
env.hq_ip = '${ips:hq}'
env.bacula_ip = '${ips:bacula}'
env.office1_ip = '${ips:office1}'
env.office2_ip = '${ips:office2}'

env.email = 'maintenance@company.com'
env.admins = ['bob', 'jane', ]

env.rules = [
    # allow SSH access from our offices
    'ufw allow from %(office1_ip)s to any port ssh' % env,
    'ufw allow from %(office2_ip)s to any port ssh' % env,

    # allow access to Bacula File Deamon from our backup server
    'ufw allow from %(bacula_ip)s to any port bacula-fd' % env,

    # allow access to Munin from HQ server
    'ufw allow from %(hq_ip)s to any port munin' % env,
```

```
    # allow HTTP from everywhere
    ufw allow http
    ufw allow https
]

def deploy():
    """The highest-level meta-command for deploying Projects
    server. Use this command only on a vanilla Ubuntu 10.04 server."""

    with settings(user='root', password=env.temp_root_pass):
        create_admin_accounts(default_password='secret123')

    create_projects_group()

    # security
    harden_sshd()
    install_ufw()
    disable_root_login()

    # bootstrap server
    set_hostname()
    set_system_time()
    install_unattended_upgrades()
    raid_monitoring()
    install_rkhunter()

    # install software stack
    install_system_libs()
    install_nginx()
    install_sendmail()

    # install python
    install_python_26()
    install_python_24()
    configure_egg_cache()

    # monitoring, backup, etc.
    install_munin_node()
    install_bacula_client()
    configure_hetzner_backup()
```

1.2.2 Sample buildout.cfg

This `fabfile.py` template has a dependency on the *niteoweb.fabfile* package and also expects to find certain buildout values and config files in certain directories. Here's a sample `buildout.cfg` that you can use to prepare an environment for using this `fabfile.py.in`. Save the `fabfile.py.in` in `etc/` directory in your buildout.

```
[buildout]
unzip = true
newest = false
extensions = buildout.dumppickedversions
prefer-final = true

parts =
    fabric
    fabfile
    bacula-fd-conf
    bacula-master-conf
    duplicity-sh
```

```
# Configuration constants
[config]
# domain on which this server runs
hostname = zulu.company.com

# server's name
shortname = zulu

# Ports of services running on this server
# (besides Nginx running on port 80 and 443)
[ports]
ssh = 22
munin = 4949
bacula = 9102

# Various IPs needed for deployment
[ips]
server = ?.?.?.?
hq = ?.?.?.?
bacula = ?.?.?.?
office1 = ?.?.?.?
office2 = ?.?.?.?

# Passwords
[pass]
bacula = strong_password_here
duplicity = strong_password_here
hetzner_ftp_user = whatever_hetzner_gives_you
hetzner_ftp_pass = whatever_hetzner_gives_you
temp_root = root_password_that_hetzner_gives_you_for_a_new_server
# temp_root password is changed and disabled later on in deployment

# Prepare Fabric
[fabfile]
recipe = collective.recipe.template
input = ${buildout:directory}/etc/fabfile.py.in
output = ${buildout:directory}/fabfile.py

[fabric]
recipe = zc.recipe.egg
eggs =
    Fabric
    niteoweb.fabfile

# Generate config files from templates in ./etc
[bacula-fd-conf]
recipe = collective.recipe.template
input = ${buildout:directory}/etc/bacula-fd.conf.in
output = ${buildout:directory}/etc/bacula-fd.conf

[bacula-master-conf]
recipe = collective.recipe.template
input = ${buildout:directory}/etc/bacula-master.conf.in
output = ${buildout:directory}/etc/bacula-master.conf

[duplicity-sh]
recipe = collective.recipe.template
input = ${buildout:directory}/etc/duplicity.sh.in
output = ${buildout:directory}/etc/duplicity.sh
```

1.2.3 Config files

Samples of config files that you need to put inside `etc/` directory in your buildout:

- `bacula-fd.conf.in`.
- `bacula-master.conf.in`.
- `duplicity.sh.in`.
- `duplicityfilelist.conf`.
- `nginx.conf`.

1.3 Bacula server

Nothing here yet ...

1.4 IPsec server

This is how to setup an IPsec server in your office so you can remotely access your internal LAN when you are on the road and also have all traffic encrypted when sitting in a cafe and using a public network.

1.4.1 Prerequisities

You router needs to forward ports 500 and 4500 to your IPsec server.

1.4.2 Sample fabfile

Below is a `fabfile.py.in` buildout template that uses commands from *Server* group to set up an IPsec server (based on Ubuntu 10.04).

```
import os
from fabric.api import env
from fabric.api import settings
from fabric.api import sudo

from niteoweb.fabfile.server import *

env.path = os.getcwd()
env.hosts = ['${ips:server}']
env.server_ip = '${ips:server}'
env.shortname = '${config:shortname}'
env.hostname = '${config:hostname}'
env.temp_root_pass = '${pass:temp_root}'

env.email = 'maintenance@company.com'
env.admins = ['bob', 'jane', ]

def deploy():
    """The highest-level meta-command for deploying Plone to the server.
    Use this command only on a fresh and clean server."""

    with settings(user='root', password=env.temp_root_pass):
        create_admin_accounts(default_password='secret123')

    # security
    harden_sshd()
```

```
    disable_root_login()

    # bootstrap server
    set_hostname()
    set_system_time()
    install_unattended_upgrades()
    install_sendmail()
    install_rkhunter()

    # install software stack
    install_ipsec()
```

1.4.3 Sample buildout.cfg

This `fabfile.py` template has a dependency on the *niteoweb.fabfile* package and also expects to find certain buildout values and config files in certain directories. Here's a sample `buildout.cfg` that you can use to prepare an environment for using this `fabfile.py.in`. Save the `fabfile.py.in` in `etc/` directory in your buildout.

```
[buildout]
unzip = true
newest = false
extensions = buildout.dumppickedversions
prefer-final = true

parts =
    fabfile
    fabric
    racoon.conf
    psk.txt

[config]
# Project shortname
shortname = ipsec

# Main domain on which this project runs on
hostname = ipsec.company.com

# Various IPs needed for deployment
[ips]
server = ?.?.?.?

[pass]
# Temporary root password assigned to us by hosting provider
temp_root = some_password_here
ipsec = strong_password_here

# Prepare Fabric
[fabric]
recipe = zc.recipe.egg
eggs =
    Fabric
    niteoweb.fabfile

[fabfile]
recipe = collective.recipe.template
input = ${buildout:directory}/etc/fabfile.py.in
output = ${buildout:directory}/fabfile.py

# Generate config files from templates in ./etc
[racoon.conf]
```

```
recipe = collective.recipe.template
input = ${buildout:directory}/etc/racoon.conf.in
output = ${buildout:directory}/etc/racoon.conf

[psk.txt]
recipe = collective.recipe.template
input = ${buildout:directory}/etc/psk.txt.in
output = ${buildout:directory}/etc/psk.txt
```

1.4.4 Config files

Samples of config files that you need to put inside `etc/` directory in your buildout:

- `racoon.conf.in`.
- `psk.txt.in`.

1.4.5 Client configuration

Configuring a client to use this IPsec server is fairly easy. For iOS, go to Settings -> Network -> VPN and add a new IPsec VPN with the following settings:

- Description: whatever
- Server: Public IP of your router behind which the IPsec server sits
- Account: a Linux user on the machine that is in the `sudo` group
- Group name: `sudo` (it's specified in `racoon.conf`)
- Secret: secret set for group `sudo` in `psk.txt`

1.5 Available commands and how to use them

Commands are separated into two grops:

- Project: commands for setting up and managing Plone/Python projects
- Server: commands for setting up and managing servers

1.5.1 Project commands

`niteoweb.fabfile.project.`**`configure_nginx`**(*shortname=None*)
Upload Nginx configuration for this site to /etc/nginx/sites-available and enable it so it gets included in the main nginx.conf.

`niteoweb.fabfile.project.`**`download_code`**(*shortname=None*, *prod_user=None*, *svn_params=None*, *svn_url=None*, *svn_repo=None*, *svn_dir=None*)
Pull project code from code repository.

`niteoweb.fabfile.project.`**`download_data`**()
Download Zope's Data.fs from the server.

`niteoweb.fabfile.project.`**`enable_nginx_config`**(*shortname=None*)
Make a link from sites-available/ to sites-enabled/ and reload Nginx.

`niteoweb.fabfile.project.`**`prepare_buildout`**(*prod_user=None*, *python_version=None*, *production_cfg=None*)
Prepare zc.buildout environment so we can use `bin/buildout -c production.cfg` to build a production environment.

niteoweb.fabfile.project.**run_buildout**(*prod_user=None*, *production_cfg=None*)
Run `bin/buildout -c production.cfg` in production user's home folder on the production server.

niteoweb.fabfile.project.**start_supervisord**(*prod_user=None*)
Start *supervisord* process monitor which in turn starts Zope and optionally others (Varnish, HAProxy, etc.).

niteoweb.fabfile.project.**supervisorctl**(**cmd*)
Runs an arbitrary supervisorctl command.

niteoweb.fabfile.project.**upload_blobs**(*prod_user=None*, *path=None*)
Upload BLOB part of Zope's data to the server.

niteoweb.fabfile.project.**upload_data**(*prod_user=None*)
Upload Zope's data to the server.

niteoweb.fabfile.project.**upload_nginx_config**(*shortname=None*, *nginx_conf=None*)
Upload Nginx configuration to /etc/nginx/sites-available.

niteoweb.fabfile.project.**upload_sphinx**(*hq_ip=None*, *sphinx_dir=None*, *path=None*)
Uploads HTML files generated by Sphinx.

niteoweb.fabfile.project.**upload_zodb**(*prod_user=None*, *path=None*)
Upload ZODB part of Zope's data to the server.

1.5.2 Server commands

niteoweb.fabfile.server.**add_to_bacula_master**(*shortname=None*, *path=None*, *bacula_host_string=None*)
Add this server's Bacula client configuration to Bacula master.

niteoweb.fabfile.server.**configure_bacula_client**(*path=None*)
Upload configuration for Bacula File Deamon (client) and restart it.

niteoweb.fabfile.server.**configure_bacula_master**(*path=None*)
Upload configuration files for Bacula Master.

niteoweb.fabfile.server.**configure_egg_cache**()
Configure a system-wide egg-cache so we have a local cache of eggs that we use in order to add speed and reduncancy to zc.buildout.

niteoweb.fabfile.server.**configure_hetzner_backup**(*duplicityfilelist=None*, *duplicitysh=None*)
Hetzner gives us 100GB of backup storage. Let's use it with Duplicity to backup the whole disk.

niteoweb.fabfile.server.**configure_nginx**(*nginx_conf=None*)
Upload Nginx configuration and restart Nginx so this configuration takes effect.

niteoweb.fabfile.server.**configure_postgres**()
Upload Postgres configuration from `etc/` and restart the server.

niteoweb.fabfile.server.**configure_racoon**(*racoonconf=None*, *psktxt=None*)
Upload racoon configuration files and restart the service.

niteoweb.fabfile.server.**configure_ufw**(*rules=None*)
Configure Uncomplicated Firewall.

niteoweb.fabfile.server.**create_admin_account**(*admin*, *default_password=None*)
Create an account for an admin to use to access the server.

niteoweb.fabfile.server.**create_admin_accounts**(*admins=None*, *default_password=None*)
Create admin accounts, so admins can access the server.

niteoweb.fabfile.server.**create_project_user**(*prod_user*)
Add a user for a single project so the entire project can run under this user.

niteoweb.fabfile.server.**create_projects_group**()
Create a group that will hold all project users -> users that are dedicated for running one project.

niteoweb.fabfile.server.**disable_root_login**()
Disable *root* login for even more security. Access to *root* account is now possible by first connecting with your dedicated maintenance account and then running `sudo su -`.

niteoweb.fabfile.server.**generate_selfsigned_ssl**(*hostname=None*)
Generate self-signed SSL certificates and provide them to Nginx.

niteoweb.fabfile.server.**harden_sshd**()
Security harden sshd.

niteoweb.fabfile.server.**initialize_postgres**()
Initialize the main database.

niteoweb.fabfile.server.**install_bacula_client**()
Install and configure Bacula backup client, which listens for instructions from Bacula master and backups critical data when told to do so.

niteoweb.fabfile.server.**install_bacula_master**()
Install and configure Bacula Master.

niteoweb.fabfile.server.**install_ipsec**(*racoonconf=None*, *psktxt=None*, *server_ip=None*)
Install and configure IPsec server.

niteoweb.fabfile.server.**install_java**()
Install java from webupd8 repository.

niteoweb.fabfile.server.**install_munin_node**(*add_to_master=True*)
Install and configure Munin node, which gathers system information and sends it to Munin master.

niteoweb.fabfile.server.**install_mysql**(*default_password=None*)
Install MySQL database server.

niteoweb.fabfile.server.**install_nginx**(*nginx_conf=None*)
Install and configure Nginx webserver.

niteoweb.fabfile.server.**install_php**()
Install FastCGI interface for running PHP scripts via Nginx.

niteoweb.fabfile.server.**install_postgres**()
Install and configure Postgresql database server.

niteoweb.fabfile.server.**install_python_24**()
Install Python 2.4 and tools for it.

niteoweb.fabfile.server.**install_python_26**()
Install Python 2.6 and tools for it.

niteoweb.fabfile.server.**install_python_27**()
Install Python 2.7 and tools for it.

niteoweb.fabfile.server.**install_rkhunter**(*email=None*)
Install and configure RootKit Hunter.

niteoweb.fabfile.server.**install_sendmail**(*email=None*)
Prepare a localhost SMTP server for sending out system notifications to admins.

niteoweb.fabfile.server.**install_system_libs**(*additional_libs=None*)
Install a bunch of stuff we need for normal operation such as `gcc`, `rsync`, `vim`, `libpng`, etc.

niteoweb.fabfile.server.**install_ufw**(*rules=None*)
Install and configure Uncomplicated Firewall.

niteoweb.fabfile.server.**install_unattended_upgrades**(*email=None*)
Configure Ubuntu to automatically install security updates.

niteoweb.fabfile.server.**normalize_rackspace**()
docstring for normalize_rackspace

niteoweb.fabfile.server.**raid_monitoring**(*email=None*)
Configure monitoring of our RAID-1 field. If anything goes wrong, send an email!

niteoweb.fabfile.server.**set_hostname**(*server_ip=None*, *hostname=None*)
Set server's hostname.

niteoweb.fabfile.server.**set_system_time**(*timezone=None*)
Set timezone and install `ntp` to keep time accurate.

CHAPTER 2

TODO

- evangelize to others and teach them how to use these commands
- make commands less NiteoWeb specific and more generalized

CHAPTER 3

Credits

- Initial release by Nejc Zupan, NiteoWeb Ltd.

CHAPTER 4

Changelog

4.1 2.2.2 (2012-02-09)

- Use `--force` when purging old Duplicity backups so it also purges old incomplete backups. [zupo]
- Instructions on how to setup iOS or OS X to connect to IPsec server. [zupo]

4.2 2.2.1 (2012-01-25)

- Fixed GitHub's URLs to point to github.com/niteoweb. [zupo]

4.3 2.2 (2012-01-25)

- Fabric step for installing *IPsec*. [zupo]
- Use sudo when configuring rkhunter. [zupo]
- Moved config files in `docs` to `docs/etc/` folder so they don't mix with Sphinx files. [zupo]

4.4 2.1.3 (2011-12-23)

- Run *bootstrap* and *buildout* with `prod_user`, not with *root*. [zupo]

4.5 2.1.2 (2011-12-23)

- Use `prod_user` from `opts` and not from `env`. [zupo]

4.6 2.1.1 (2011-12-23)

- Minor runtime fix for `supervisorctl` command. [zupo]
- Moved `cmd` coomand to `__init__.py` so it's available both in `server.py` and `project.py`. [zupo]
- Update RKHunter's files properties DB every time you run apt-get install, this prevents warnings every time a new version of some package is installed. [zupo]

4.7 2.1 (2011-11-15)

- Lots of minor bugfixes. [zupo]
- You can now specify python version that is used for bootstraping buildout. [zupo]
- Added *gitk* to list of libraries to install. [zupo]
- Added buildout.cfg to test how sphinx docs are generated. [zupo]
- Enabled choosing filename for 'production' buildout configuration. [zupo]
- The configure_egg_cache() command is now more resilient to multiple runs. [zupo]
- Added instructions and examples on how to use niteoweb.fabfile for setting up a new server for running Plone projects. [zupo]
- Added commands for installing and configuring a server that will run Plone projects. [zupo]
- Added Sphinx documentation. [zupo]

4.8 2.0.2 (2011-11-13)

- You can now specify python version that is used for bootstraping buildout. [zupo]

4.9 2.0.1 (2011-11-13)

- HISTORY.txt missing from release. [zupo]

4.10 2.0 (2011-11-13)

- Use niteoweb.fabfile.err instead of _verify_opts. [zupo]
- Breaks backwards compatibility with commands in project.py [zupo]

4.11 0.1.2 (2011-10-21)

- Added many new commands for setting up servers. [zupo]

4.12 0.1.1 (2011-08-28)

- Packaging fixes. [zupo]

4.13 0.1 (2011-08-28)

- Initial release. [zupo]

CHAPTER 5

License

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

n