I decided for my new company 'Mynij' to operate by myself my own SlapOS infrastructure and ERP and to use whatever exists that is convenient and cheap to bootstrap my presence on the Web while maintaining my accounting and document database.

Step 1: web presence

I went to Gandi (www.gandi.net) to create my domain name:www.mynij.com.

While doing so, I found that Gandi was proposing also to host my web site and my emails. I decided to try both.

Webmail at Gandi is powered by Roundcube. This is fine for me since this is the same as the one I use everyday at Nexedi.

About web site creation, I know that Gandi has contributed to a tool called Silex (www.silex.me). However it seems that their free templates are now powered by a proprietary software called Basekit. I found Basekit templating system very easy to use. I could get some basic home page in quite short time, last weekend, without thinking too much. It is sad though that this is no longer Free Software.

I also requested a free blog powered by Dotclear: http://blog.mynij.com/.

Step 2: dedicated server

I need a dedicated server to run various experiments, and also to run Mynij's core IT system using ERP5 Free Software. I found that Online.net has a nice Xeon server with 6 cores (12 virtual cores), 500 GB SSD and 32 GB RAM for less than 30€ / month: the Dedibox Classic 2016. That is perfect for me.

I installed **Debian 8** (not Ubuntu) on the server with a RAID 1 configuration that merges the two 250 GB SSD disks as a single disk with some redundancy. Everything was easy and fully automated. Online.net even provides a virtual KVM to remotely control the server. This virtual KVM only requires a modern HTML5 web browser. No Java, no flash is needed.

Note: I use Debian 8 here because some SlapOS software release, such as SlapOS CDN (see bellow) are not yet tested with Ubuntu. This could change in the future.

Step 3: installing SlapOS

I decided to create my own mini-cloud with automated orchestration using SlapOS. SlapOS has the advantage to be extremely stable and to consume very few resources thanks to nanocontainers. I actually care about stability very much: this is why I do not trust traditional containers (ex. LXC, Docker, Kubernetes, etc.) which do not enforce strict consistency between binaries and Linux private kernel API and thus lead to various unexpected or strange crashes. The lack of stability of traditional containers is actually not LXC's fault but rather the fact that most GNU/Linux distributions contain a lot of binaries that directly call the private API of Linux kernel, which itself is not consistent from one version to the next. With SlapOS, this problem is gone: SlapOS recompiles everything to match the runtime environment.

You can find documentation about SlapOS at the following URL:http://community.slapos.org/wiki/osoe-Lecture.SlapOS.Extended

IPv6 Configuration

Before starting SlapOS installation, I had to configure IPv6. IPv6 is required for SlapOS. Usually, one uses a Free IPv6 service such as Grandenet or a commercial service such as VIFIB since most hosting providers still do not support IPv6. Luckily, Online (or OVH) support IPv6 at no extra cost.

In order to use IPv6 with Online's dedibox, one needs to access the IPv6 network console:

https://console.online.net/fr/network/

If no IPv6 range was already allocated to the server, you will then need to request one IPv6 range for your server. Once it is allocated, your range is identified by so-called **DUID** number that can later be used to configure IPv6 networking on your server.

You will also need to configure IPv6 networking on your server. Online provides a documentation here:

http://documentation.online.net/fr/serveur-dedie/reseau/prefixe_ipv6

What I did not understand the first time I tried is that you need to configure three files:

```
/etc/dhcp/dhclient6.conf
/etc/systemd/system/dhclient.service
/etc/network/interfaces
```

If you only configure dhcp, then your server will not be allocated any IPv6 address. And if you only configure interfaces, your server will not know which default gateway it should use for IPv6.

Most values are self-explanatory. The only values one should should care are IPV6ADDRESS and PREFIXLEN. Just to make things clear, here is my own */etc/network/interfaces* file (with XXXX to keep my own privacy):

```
iface eth0 inet6 static
   address 2001:XXXX:XXXX::1
   netmask 64
```

Beware of the file */etc/systemd/system/dhclient.service*. On Debian 8, *dhclient* is located at */sbin/dhclient* and not */usr/sbin/dhclient*.

Note: you must replace 2001:XXXX:XXXX::1 with an address part of the range that has been allocated to you by Online and that can be found in the console (https://console.online.net/fr/network/)

SlapOS Package

Next step consists on installing SlapOS. I just followed the documentation:

http://community.slapos.org/wiki/osoe-Lecture.SlapOS.Extended/developer-Installing.SlapOS.Package

I typed the following command lines for Debian 8:

```
sudo su
echo "deb http://download.opensuse.org/repositories/home:/VIFIBnexedi/Debian_8.0/ ./" | tee /etc/apt/sources.list.d/slapos.list
wget -O- "http://download.opensuse.org/repositories/home:/VIFIBnexedi/Debian_8.0/Release.key" | apt-key add -
apt-get update
apt-get install slapos-node
```

That's all.

Step 4: configuring SlapOS

There are many different ways to configure SlapOS. Usually, people connect to a SlapOS Master service such as the one provided by VIFIB. But since I wanted to create a fully autonomous system, at zero cost, I decided to use a less known approach to configure a SlapOS node: slapproxy.

I typed the following command line:

```
slapos configure local --interface-name eth0
```

This configured my Dedibox server as a SlapOS node and a SlapOS master running at the same time on the same host.

Note: if the command line does seem to progress, just type "enter" and it will surely progress and end.

In order to be sure that everything is done, type the following command:

```
slapos node
```

You should see two services running: **slapproxy** and **watchdog**. If this is not the case, try again after one minute. If this is still not the case, then something must have gone wrong and you will end up learning SlapOS source code, which is fairly small (only a few thousand lines for a full cloud and devops system that does much more than some monster projects).

Commenting slapproxy configuration

You will now need to change the file:

```
/etc/opt/slapos/slapos-proxy.cfg
```

and add a sharp sign (#) at the first column of the last 11 lines, like this:

```
#[multimaster/https://slap.vifib.com]
#key = key file path coming from your slapos master account
#cert = certificate file path coming from your slapos master account
#software_release_list =
#  http://git.erp5.org/gitweb/slapos.git/blob_plain/HEAD:/software/apache-frontend/software.cfg

#[multimaster/http://imaginary-slapos-master.com]
# No certificate here: it is http.
#software_release_list =
#  http://mywebsite.me/my_software_release.cfg
#  /some/arbitrary/local/unix/path
```

The purpose of this sharp sign (#) is to comment out some lines on slapproxy that are present to forward CDN service requests to VIFIB's SlapOS Master. Nothing is wrong here. We just do not want in this tutorial to depend on anyone, so this is the simplest way to go.

Step 5: installing ERP5, Webrunner and CDN

From this step, things are going to become quite easy (in appearance).

I decided to install a couple of software that I am going need:

- **ERP5** to run everything in Mynij (accounting, invoicing, project, CRM, documents, etc.)
- **Webrunner** Web IDE to develop software
- **CDN** to provide accelerated or secured HTTP/HTTPS access to ERP5 or Webrunner

Installation of ERP5 requires to type the following command line:

```
slapos supply https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/erp5/software.cfg local_computer
```

Installation of Webrunner requires to type the following command line:

```
slapos supply https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/slaprunner/software.cfg local_computer
```

Installation of CDN requires to type the following command line:

```
slapos supply https://lab.nexedi.com/nexedi/slapos/raw/1.0.45/software/apache-frontend/software.cfg local_computer
```

Typing those commands will initiate the build process by SlapOS in background. You can track this process by having a look at log file:

```
tail -f /opt/slapos/log/slapos-node-software.log
```

This process can take a few hours if SlapOS has to recompile a software. It can take a few minutes if a precompiled binary already exists in a public cache.

You can also trigger the build process manually by running:

```
slapos node software --now
```

Note: if build is still running, a warning message will be displayed when you try to launch a second build in parallel. This is intentional so that not all server resources get mobilised for compilation.

Step 6: launching ERP5

One can know that the installation is finished whenever the result of the command line:

```
slapos node software --now
```

looks like this:

```
2017-02-14 19:39:26 slapos[11333] INFO Processing software releases...
2017-02-14 19:39:26 slapos[11333] INFO Finished software releases.
```

What SlapOS did is to recompile the complete ERP5 stack, including all of its dependencies down to glibc. You can get an idea of all dependencies by having a look at this file:

https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/stack/erp5/buildout.cfg

What SlapOS can also do is to generate all kinds of templates that are mutually dependent. This is called sometimes service orchestration. You can get an idea of the kind of template files that are used by having a look here:

https://lab.nexedi.com/nexedi/slapos/tree/master/stack/erp5

It is thus possible to create an ERP5 instance with all its service (more than 10) by typing one line:

```
slapos request erp5 https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/erp5/software.cfg
```

This will trigger the configuration of an ERP5 service. SlapOS will configure everything in background. You can track the configuration process by having a look at the log file:

```
tail -f /opt/slapos/log/slapos-node-instance.log
```

You can also trigger the build process manually by running:

```
slapos node instance --now
```

Note: if configuration is still running, a warning message will be displayed. This is intentional to ensure that not all system resources are mobilised by orchestration.

Checking configuration completion

One can know that the configuration is finished if the result of the command line:

```
slapos request erp5 https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/erp5/software.cfg
```

displays various connection parameters such as:

```
2017-02-21 20:01:58 slapos[14507] INFO Requesting erp5 as instance of https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/erp5/software.cfg...
2017-02-21 20:01:58 slapos[14507] INFO Instance requested.
State is : started.
2017-02-21 20:01:58 slapos[14507] INFO Connection parameters of instance are:
2017-02-21 20:01:58 slapos[14507] INFO {'_': {"hosts-dict": {"erp5-cloudooo": "10.0.22.183", "erp5-smtp": "127.0.0.2", "erp5-catalog-0": "10.0.90.164", "erp5-memcached-volatile": "10.0.198.177", "erp5-memcached-persistent": "10.0.159.155"}, "site-i
2017-02-21 20:01:58 slapos[14507] INFO You can rerun the command to get up-to-date information.
```

If you reach this state, please write down carefully the value of parameter*family-default-v6* (*https://[2001:XXXX:XXXX::XXXX]:2151* in my case) and the value of*inituser-password.*

Step 7: adding a CDN for ERP5

Let us now add a CDN service by typing:

```
slapos request cdn https://lab.nexedi.com/nexedi/slapos/raw/1.0.45/software/apache-frontend/software.cfg
```

This will run an apache server, an nginx server and a traffic server that can be used to protect or accelerate access to the various services running on this host.

Once the service is running, typing again:

```
slapos request cdn https://lab.nexedi.com/nexedi/slapos/raw/1.0.45/software/apache-frontend/software.cfg
```

will return connection parameters like this once it is running:

```
2017-02-21 20:04:41 slapos[16879] INFO Requesting cdn as instance of https://lab.nexedi.com/nexedi/slapos/raw/1.0.45/software/apache-frontend/software.cfg...
2017-02-21 20:04:41 slapos[16879] INFO Instance requested.
State is : started.
2017-02-21 20:04:41 slapos[16879] INFO Connection parameters of instance are:
```

```
2017-02-21 20:04:41 slapos[16879] INFO {'accepted-slave-amount': '5',
 'domain': 'None',
 'monitor-base-url': None,
 'monitor-password': 'XXXXX',
 'monitor-setup-url': 'https://monitor.app.officejs.com/#page=settings_configurator&url=/public/feeds',
 'monitor-url': '/public/feeds',
 'monitor-user': 'admin',
 'rejected-slave-amount': '0',
 'rejected-slave-list': '[]',
 'slave-amount': '5'}
2017-02-21 20:04:41 slapos[16879] INFO You can rerun the command to get up-to-date information.
```

You will then need to find out what is the address of the running CDN. Type the command:

```
slapos node
```

in order to have a look at all services running on your server under SlapOS control. Search for a service called *frontend_apache-on-watch*.

```
slappart9:frontend_apache-on-watch            RUNNING   pid 19300, uptime 0:27:36
slappart9:frontend_nginx-on-watch             RUNNING   pid 19303, uptime 0:27:36
```

This tells you that your CDN has been allocated to partition 9. You can now have a look at the configuration file:

```
more /srv/slapgrid/slappart9/etc/apache_frontend.conf
```

you may need to replace slappart9 by the appropriate value that you could read with *slapos node* previously.

Search in this file a few lines that looks like this:

```
Listen 10.0.203.217:26011
Listen 10.0.203.217:26012
Listen 10.0.203.217:8080
Listen 10.0.203.217:4443
```

This tells you that your CDN is listening on the internal IP address 10.0.203.217. We will need this in the next step.

Firewall configuration

In order for this CDN to be accessible from the outside world, we need to configure the firewall of the server by typing a command that looks like:

```
iptables -t nat -A PREROUTING -p tcp -d IPV4ADDRESS --dport 443 -j DNAT --to-destination CDNADDRESS:4443
```

where IPV4ADDRESS is the host public IPv4 address and CDNADDRESS is the internal IPv4 address of the CDN service.

In the case of Mynij, I typed this command line:

```
iptables -t nat -A PREROUTING -p tcp -d 163.172.72.196 --dport 443 -j DNAT --to-destination 10.0.203.217:4443
```

This command redirects port 443 on the host publish IPv4 address (163.172.72.196) to port 4443 on the CDN service listening to 10.0.203.217

You will of course need to replace those values with your own.

CDN Slave

We can now add a CDN slave by typing something like:

```
slapos request cdn_erp5 https://lab.nexedi.com/nexedi/slapos/raw/1.0.45/software/apache-frontend/software.cfg --slave --parameters url=https://[2001:XXXX:XXXX::XXXX]:2151/ type=zope custom_domain=erp5.mynij.com server-alias=erp5.mynij.cc
```

and replacing the XXXX values with your owns.

This command creates so-called slave service, something quite unique in SlapOS. It is useful whenever one service depends on another service which is shared by many services. In our case, the *cdn* service has a slave called *cdn_erp5*. We will see bellow how to add a second slave to *cdn*.

DNS Update

Finally, you need to declare the erp5 subdomain in Gandi DNS entries through the Gandi DNS administration pannel. In my case, I declared this:

```
@ 10800 IN A 217.70.184.38
one 10800 IN A 163.172.72.196
blog 10800 IN CNAME blogs.vip.gandi.net.
erp5 10800 IN CNAME one.mynij.com.
imap 10800 IN CNAME access.mail.gandi.net.
pop 10800 IN CNAME access.mail.gandi.net.
runner 10800 IN CNAME one.mynij.com.
smtp 10800 IN CNAME relay.mail.gandi.net.
webmail 10800 IN CNAME webmail.gandi.net.
www 10800 IN CNAME gandi.ws.
@ 10800 IN MX 50 fb.mail.gandi.net.
@ 10800 IN MX 10 spool.mail.gandi.net.
@ 10800 IN TXT "v=spf1 include:_mailcust.gandi.net ?all"
```

Then I validated the new zone. In this configuration, 217.70.184.38 is provided by Gandi, 163.172.72.196 is the public IPv4 of my host and one is the name I have to my server. The only two entries I added are **erp5** and **runner**.

After a few minutes (or hours), I typed: https://erp5.mynij.com/ and the following message was displayed.

```
Zope Quick Start
```

Everything should now be fine to keep on.

Step 8: configuring ERP5

I decided to follow the tutorial: https://www.erp5.com/erp5-HowTo.Request.Erp5.Instance.On.Slap.Os.Webrunner

But instead of starting from the first step, I started from the sentence "Once you have an URL, you can create your ERP5 site." until the sentence "refresh the page from time to time to see when your instance is ready".

This gave me a basic ERP5 instance, ready for the next step: automated configuration.

I moved to the following URL:

```
https://erp5.mynij.com/erp5/portal_configurator/
```

This displays the automated configurator of ERP5, a tool that is able to create a complete configuration of ERP5 based on a few questions. I clicked on the "Configure" button of the "Small and Medium Business" icon.

After some questions and some minutes, I can now start running my company with ERP5. ERP5 provides:

- CRM
- Accounting
- Document management
- Project management
- etc.

Step 9: add a Webrunner

I would also like to add a Webrunner Web IDE in order to do some personal development. For this, I type:

```
slapos request runner https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/slaprunner/software.cfg
```

After some minutes, I type again the same command:

```
slapos request runner_erp5 https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/slaprunner/software.cfg
```

and I see the following result:

```
2017-02-14 20:48:43 slapos[27670] INFO Requesting runner_erp5 as instance of https://lab.nexedi.com/nexedi/slapos/raw/1.0.48/software/slaprunner/software.cfg...
2017-02-14 20:48:43 slapos[27670] INFO Instance requested.
State is : started.
2017-02-14 20:48:43 slapos[27670] INFO Connection parameters of instance are:
2017-02-14 20:48:43 slapos[27670] INFO {'backend-url': 'https://[2001:XXXX:XXXX::XXXX]:50005',
 'git-private-url': 'https://[2001:XXXX:XXXX::XXXX]:9686/git/',
 'git-public-url': 'https://[2001:XXXX:XXXX::XXXX]:9686/git-public/',
 'init-password': 'YYYYYY',
 'init-user': 'admin',
 'monitor-base-url': None,
 'monitor-setup-url': 'https://monitor.app.officejs.com/#page=settings_configurator&url=/public/feeds&username=admin&password=YYYYYY',
 'public-url': '/public/',
 'ssh-command': 'ssh slapuser10@2001:XXXX:XXXX::XXXX -p 22222',
 'url': 'https://',
 'webdav-url': '/share/'}
2017-02-14 20:48:43 slapos[27670] INFO You can rerun the command to get up-to-date information.
```

I can now request a CDN slave instance for this Webrunner by copying its URL (https://[2001:XXXX:XXXX::XXXX]:50005) to the relevant parameter and using a dedicated domain:

```
slapos request cdn_runner https://lab.nexedi.com/nexedi/slapos/raw/1.0.45/software/apache-frontend/software.cfg --slave --parameters url=https://[2001:XXXX:XXXX::XXXX]:50005/ custom_domain=runner.mynij.com server-alias=runner.mynij.com
```

I can now access the Webrunner through the URL: https://runner.mynij.com/

A complete tutorial about Webrunner can be found here: http://community.slapos.org/wiki/slapos-Wiki.Home/developer-Lecture.Web.Runner.Extended

Notes

Many options for SlapOS frontend are described in https://lab.nexedi.com/nexedi/slapos/tree/1.0.48/software/apache-frontend

You can find the latest stable versions of ERP5, CDN and Webrunner by going tohttps://lab.nexedi.com/nexedi/slapos/blob/master/software/slaprunner/software.cfg and looking at latest tags