FUNC
Fedora Unified Network Controller

Luca Foppiano
<lfoppiano@byte-code.com>

Summary

The big picture

Solutions

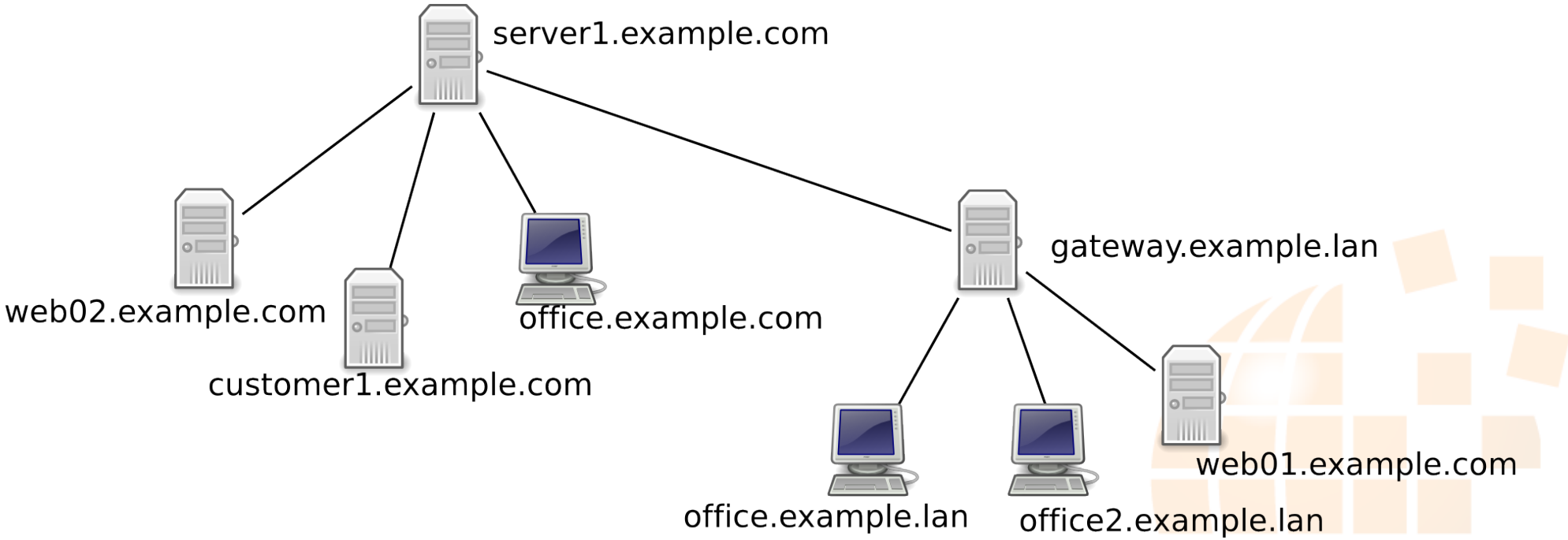
Features

What about future?

Related projects

The big picture

- *“turn off all testing virtual machines”*
- *“restart all crashed web services”*
- *“update all machines with operation suffix in the name (eg. web*.example.*, customer*, etc)”*



Solutions?

- Manual solution
- SSH or Telnet (as your risk)
- Func (https://fedorahosted.org/func)

Manual solution

- Obsolete
- Expensive
- Impossible on world distributed network
- High risk
- Need to trust unskilled people

SSH

- Secure
- Bash powered
- Problems with multi-hop
- Requires manual “public key” exchange
- Security issues (one machine has the control of whole networks without any filter)

Func

- Provides Python APIs (and/or CLI command) to manage huge number of machines
  - Fedora Unified Network Controller
  - A Red Hat and Fedora Project
  - Written in Python
  - Secure (https based connection)
  - Module-based architecture
  - Easy to expand by writing new modules
  - Security model guarantee by ACLs
  - Web interface based on TurboGear (FuncWeb)

Func: quick start

- Two components: certmaster (51235/tcp) and minion (51234/tcp)
- Status or the art: 1 Certmaster, N minions (Proxy module will be available after Google SoC)
- Certmaster needs to sign minions by certificate generation (automatically performed using autosign): certmaster-ca tool.
- Minion needs only to know who is certmaster

```
[root@a~]# certmaster-ca --list
[root@a~]# certmaster-ca --sign hostname.domain.x
[root@a~]# certmaster-ca --clear hostname.domain.x
```

Func: starting

- Open a shell on certmaster host

Python API

```
>>> from func.overlord import client
>>> client1 = client.Client("*.lan")
>>> client1.service.restart("httpd")
>>> client1.command.run("df -h")
```

CLI

```
[root@a~]# func "*.com" call service restart "httpd"
[root@a~]# func "*.lan" call command run "df -h"
```

What about modules?

- Func based on modules architecture
- A module support new stuff
- 20 modules (libVirt, jboss, info, process, command, iptables, nagios, etc)
- Writing a new module is simple.
- When you write a module, it works on both CLI and PyScripting, no modification on func are needed.

How to write new module

- func-create-module
- By hand

```
import func_module
class NewModule(func_module.FuncModule):
    version = "1.0"
    api_version = "0.1"
    description = "new module"

    def __init__(self):
        pass

    def anAction(self, arg1, arg2):
        pass
```

Advanced features

- Async mode
  - Only on python API (implementation is coming ;-))
  - Useful on long time required commands (eg. Yum update)
- Multiplexer: possibility to launch more than one process
- Globbing
- Grouping

Globbing

Python API

```
>>> from func.overlord import client
>>> glob1 = client.Client("customer*; office.example.lan")
>>> glob1.yumcmd.update();
```

CLI

```
[root@a~]# func "*.example.org;*.lan" run yumcmd update
[root@a~]# func "web*.domain.it;virt*" run
```

Grouping

```
[root@a~]# cat /etc/func/groups
[webservers]
host = office.example.lan, customer01.example.com

[jbossas]
host = *.example.lan
```

Python API & CLI Example

```
>>> from func.overlord import client
>>> client.Client(@webservers).service.restart(“httpd”);
```

```
[root@a~]# func “@webservers” run service restart “httpd”
```

Future ideas

- Modules module
- Google Summer of Code:
  - Proxy module
  - System-config-* module
  - User/groups manipulation
- Package on other distributions (Debian, Suse, Ubuntu, etc.)

Related projects

- Symbolic (http://www.opensymbolic.org)
- Puppet (http://reductivelabs.com/trac/puppet)
- Puppet-team (http://projects.byte-code.com/trac/puppet-team)
- Cobbler (http://cobbler.et.redhat.com/)

Thanks ;-)

Questions and answers?