http://projects.unbit.it/uwsgi/

uWSGI: MacGyver swiss army knife

(Roberto De Ioris EuroPython 2011)

http://projects.unbit.it/uwsgi/

uWSGI: MacGyver swiss army knife

(Roberto De Ioris EuroPython 2011)

uvvuesgiai not iuuisghi nor microuisghi

The first uWSGI bug:

The first uWSGI bug:

a fu!@#$g name...

uWSGI

http://projects.unbit.it/uwsgi/

What's that ?

What's that ?

... later

uWSGI

http://projects.unbit.it/uwsgi/

What it is not:

What it is not:

Not (only) a WSGI server/gateway

What it is not:

Not (only) a WSGI server/gateway

Not a communication protocol

http://projects.unbit.it/uwsgi/

uWSGI is an application server

uWSGI

http://projects.unbit.it/uwsgi/

uWSGI is an application server

or better...

http://projects.unbit.it/uwsgi/

uWSGI is an application server

or better...

an application container

History

- 2008 the RugD project (dead)
- january 2009 the first tests
- april 2009 first official release (500 lines)
- may 2009 first external helps
- ... the first cavy, Marco Beri
- ... the first bugs
- ... the first flames
- summer 2009, Cherokee :)
- benchmarks
- june 2010 Nginx :)
- benchmarks (again)
- Lighttpd :((sdeng...)
- 2011 grow grow grow
- june 2011 0.9.8 release (40.000 lines)
- june 2011 official doc on django site
- june 2011 official debian packages
- june 2011 0.9.8.1 at europython

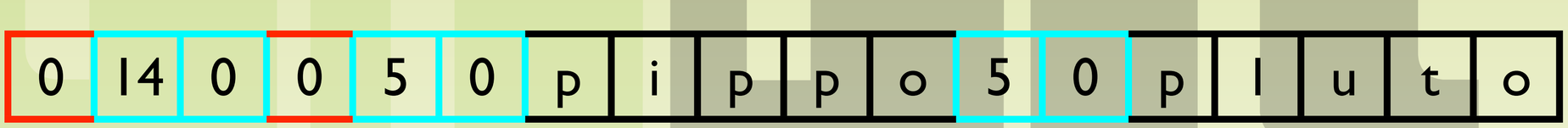
The uwsgi protocol

```
struct uwsgi_header {
  uint8_t modifier1;
  uint16_t pktsize;
  uint8_t modifier2;
};

struct uwsgi_string {
  uint16_t size;
  char string[size];
};
```

The uwsgi protocol: a dictionary

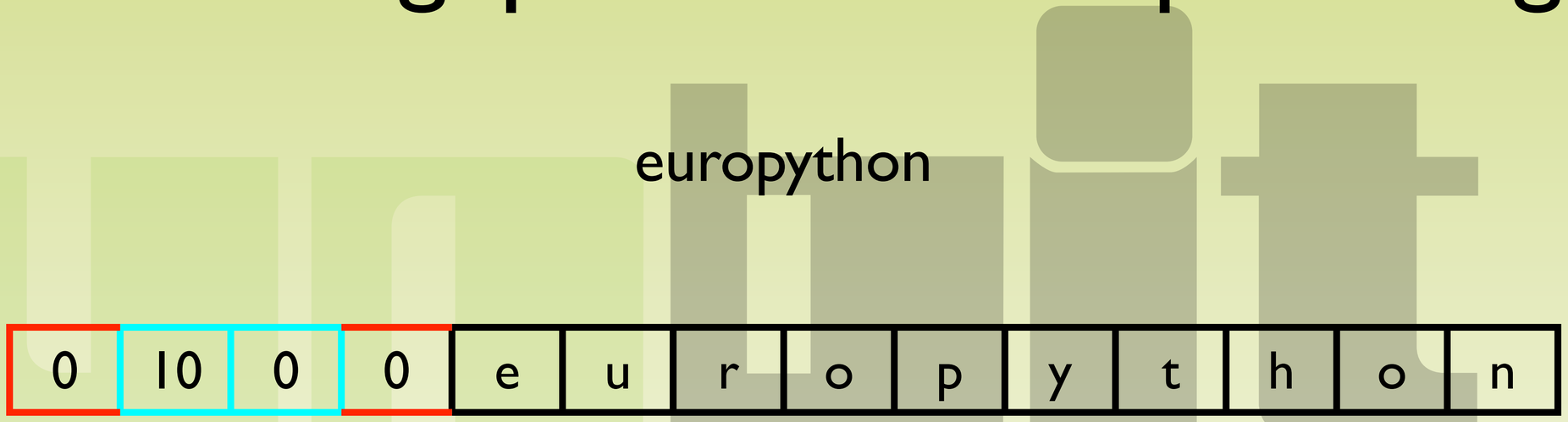
pippo = pluto



in nginx:

```
uwsgi_param pippo pluto;
```

The uwsgi protocol: a simple string



in nginx:

```
uwsgi_string europython;
```

http://projects.unbit.it/uwsgi/

uWSGI: dynamically bloated

Master
Workers
Spooler
Gateways
Daemons
Generic plugins
Request plugins

uWSGI

http://projects.unbit.it/uwsgi/

Architecture:

uWSGI

http://projects.unbit.it/uwsgi/

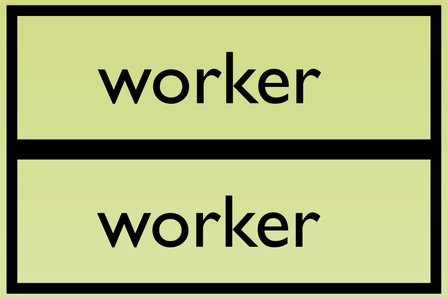
Architecture:

worker

uWSGI

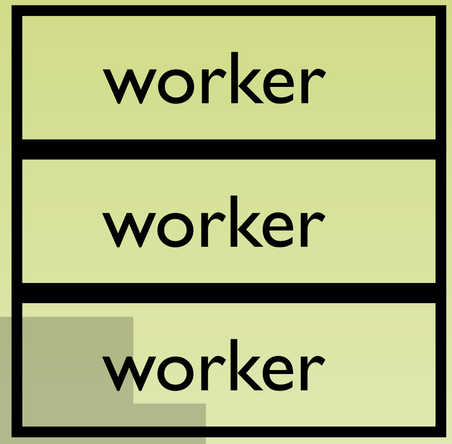
http://projects.unbit.it/uwsgi/

Architecture:



http://projects.unbit.it/uwsgi/

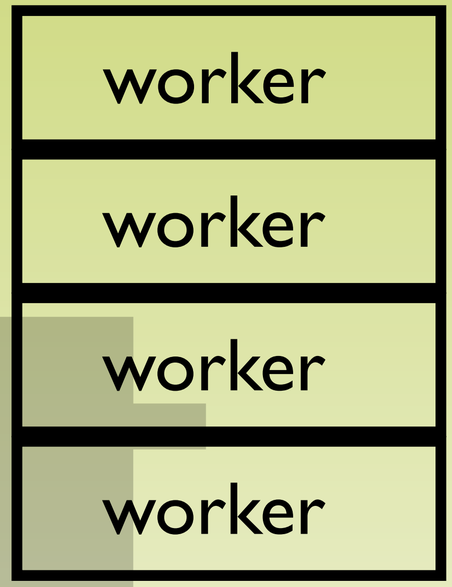
Architecture:



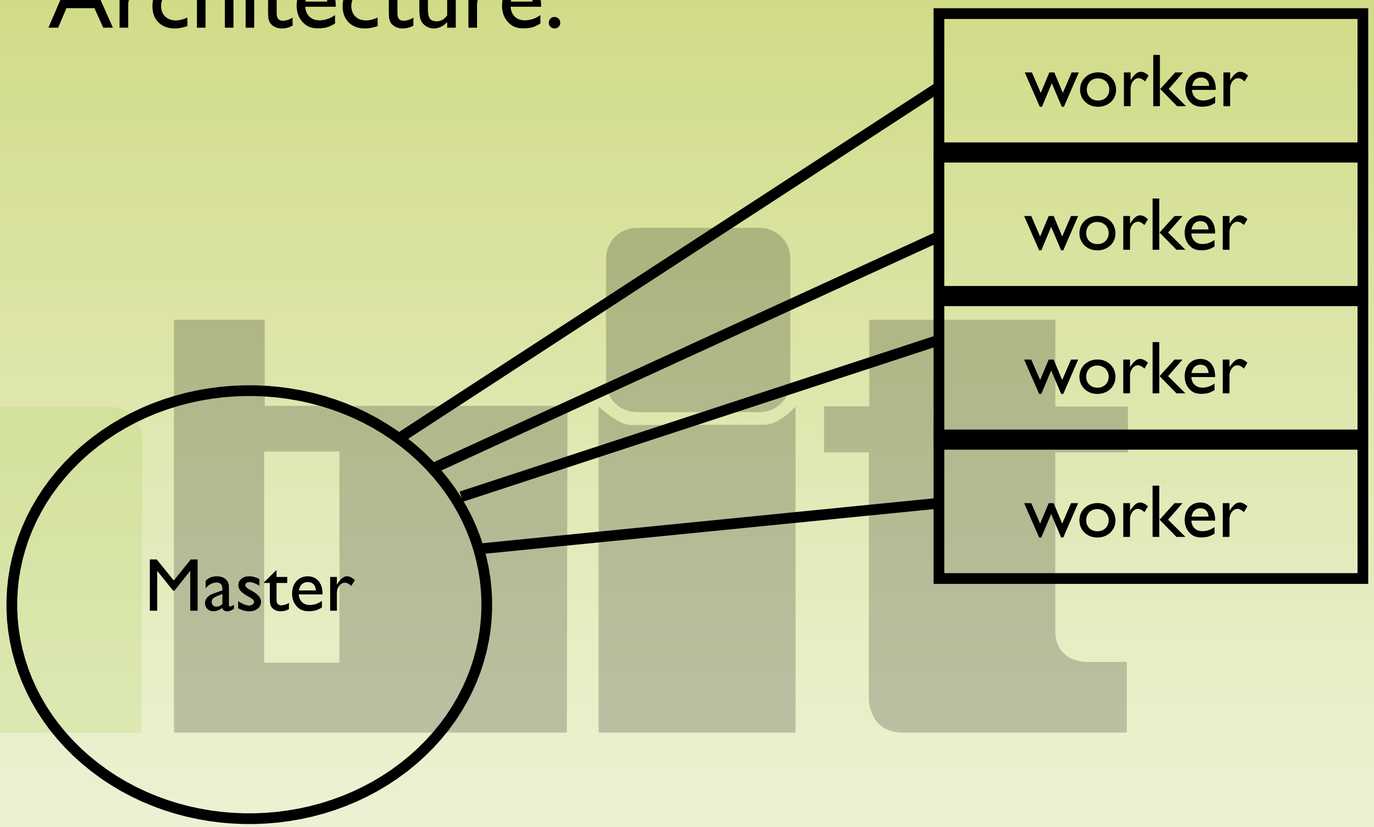
uWSGI

http://projects.unbit.it/uwsgi/

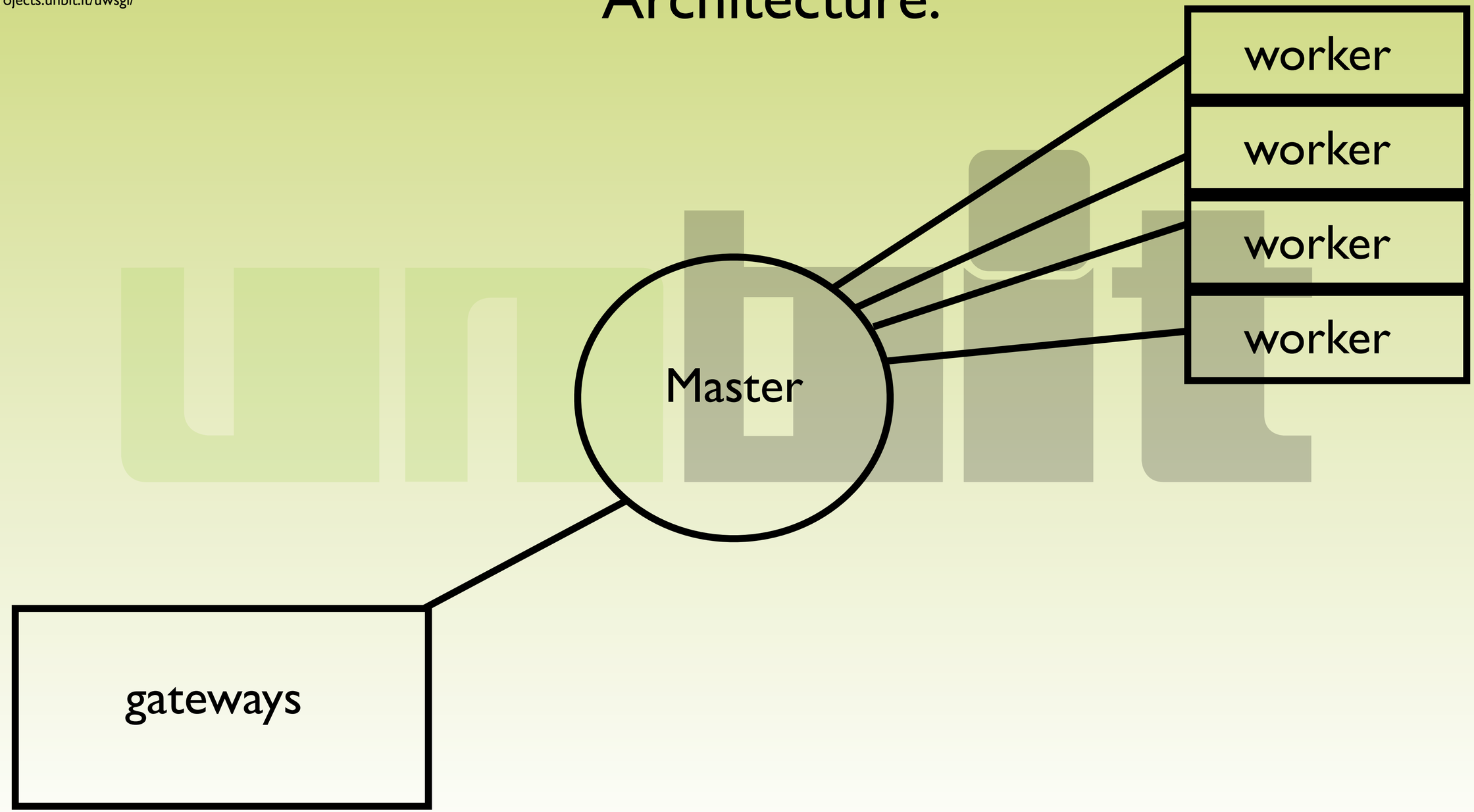
Architecture:



Architecture:

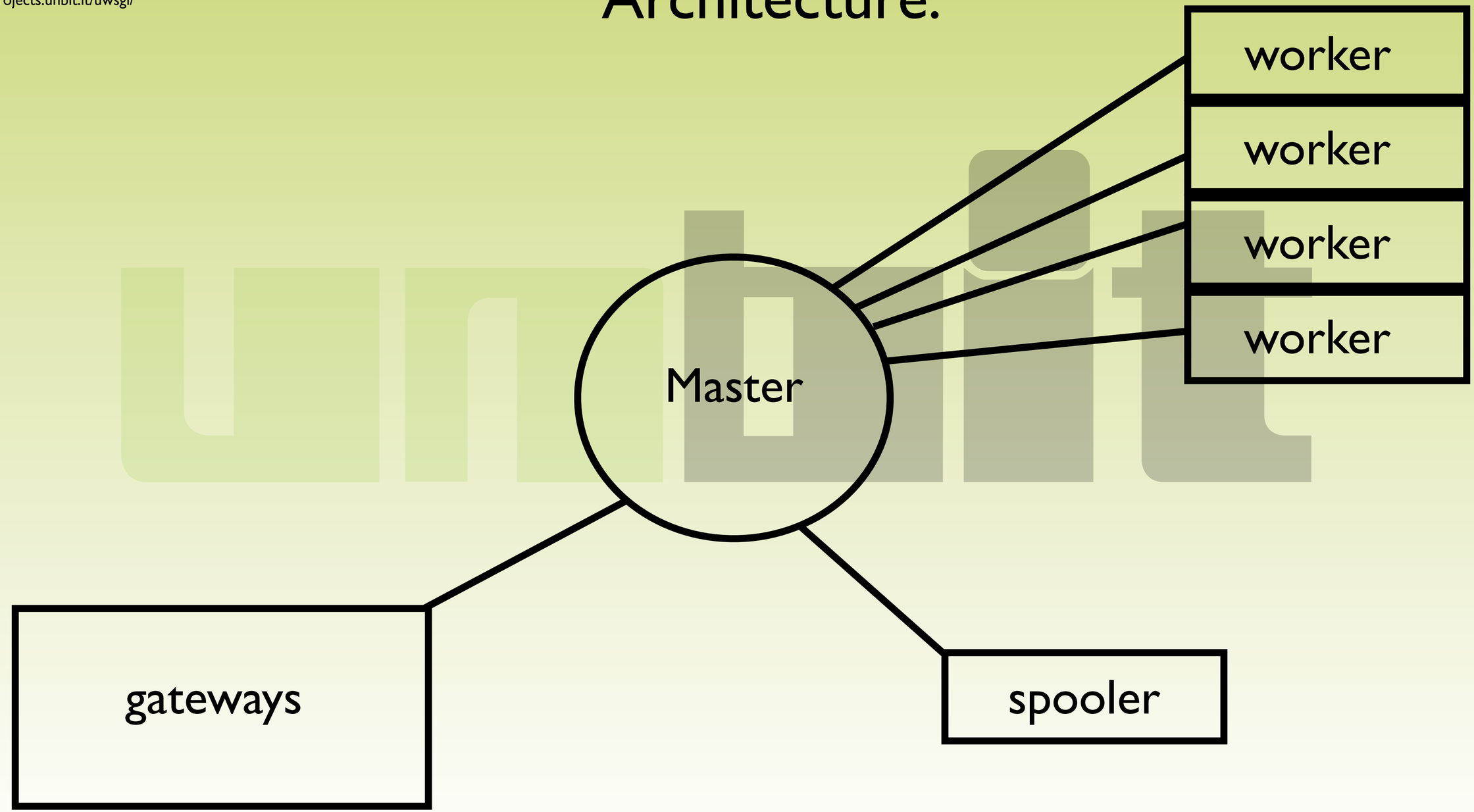


Architecture:



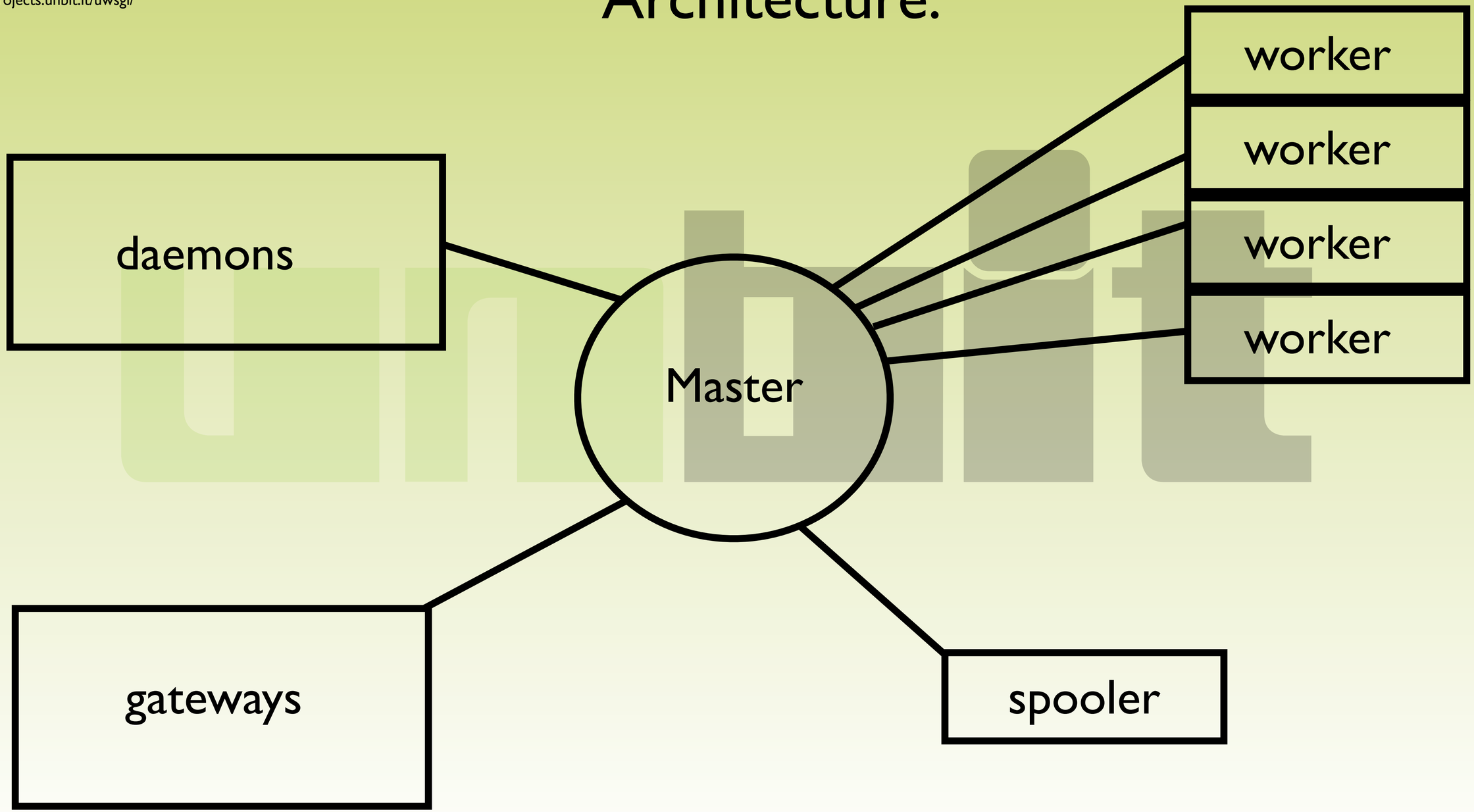
http://projects.unbit.it/uwsgi/

Architecture:



http://projects.unbit.it/uwsgi/

Architecture:



Sysadmin porn

- unlimited support for admin's taste
- config via pipe
- logging everywhere
- jailing
- proxying
- resource monitoring and limiting
- no fork bombing
- dynamic configurations
- harakiri
- various spells

The first app (WSGI)

Hello world

http://projects.unbit.it/uwsgi/

The first app (WSGI)

Hello world

```
# myapp.py

def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return "<h1>Hello World</h1>"
```

The first app (WSGI)

Hello world

```
# myapp.py

def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return "<h1>Hello World</h1>"
```

```
# uwsgi --socket 127.0.0.1:3031 --module myapp
```

The first app (WSGI)

Hello world

```
# myapp.py

def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return "<h1>Hello World</h1>"
```

```
# uwsgi --socket 127.0.0.1:3031 --module myapp
```

```
<Location />
  uWSGISocket 127.0.0.1:3031
  SetHandler uwsgi-handler
</Location>
```

http://projects.unbit.it/uwsgi/

The first app (WSGI)

Hello world

```
# myapp.py

def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return "<h1>Hello World</h1>"
```

```
# uwsgi --socket 127.0.0.1:3031 --module myapp
```

```
<Location />
  uWSGISocket 127.0.0.1:3031
  SetHandler uwsgi-handler
</Location>
```

```
server {
    server_name ep2011.europython.eu;
    location {
        include uwsgi_params;
        uwsgi_pass 127.0.0.1:3031;
    }
}
```

http://projects.unbit.it/uwsgi/

nginx uwsgi_params content

```
uwsgi_param  QUERY_STRING       $query_string;
uwsgi_param  REQUEST_METHOD     $request_method;
uwsgi_param  CONTENT_TYPE       $content_type;
uwsgi_param  CONTENT_LENGTH     $content_length;

uwsgi_param  REQUEST_URI        $request_uri;
uwsgi_param  PATH_INFO          $document_uri;
uwsgi_param  DOCUMENT_ROOT      $document_root;
uwsgi_param  SERVER_PROTOCOL    $server_protocol;

uwsgi_param  REMOTE_ADDR        $remote_addr;
uwsgi_param  REMOTE_PORT        $remote_port;
uwsgi_param  SERVER_PORT        $server_port;
uwsgi_param  SERVER_NAME        $server_name;
```

Problem one: concurrency

Multiprocess ?
Multithread ?

```
# uwsgi --socket 192.168.0.1:4040 --module myapp --processes 2

# uwsgi --socket 192.168.0.1:4040 --module myapp --workers 2

# uwsgi --socket 192.168.0.1:4040 --module myapp --threads 8

# uwsgi --socket 192.168.0.1:4040 --module myapp --threads 8 --processes 2

# uwsgi --socket 192.168.0.1:4040 --module myapp --workers 2 --enable-threads
```

Async/non-blocking

```
# uwsgi --async 4000 --module myapp:async


def async(env, start_response):
    headers = [('Content-Type', 'text/html')]
    start_response('200 OK', headers)

    yield "<h1>One</h1>"
    yield "<h2>Two</h2>"
    yield "<h3>Three</h3>"
```

http://projects.unbit.it/uwsgi/

Async/non-blocking

```
# uwsgi --async 4000 --module myapp:async


def async(env, start_response):
    headers = [('Content-Type', 'text/html')]
    start_response('200 OK', headers)

    yield "<h1>One</h1>"
    yield "<h2>Two</h2>"
    yield "<h3>Three</h3>"
```

```
# uwsgi --async 4000 --module myapp --callable async2

import uwsgi

def async2(env, start_response):
    headers = [('Content-Type', 'text/html')]
    start_response('200 OK', headers)

    yield "<h1>One</h1>"
    yield uwsgi.async_sleep(10)
    yield "<h2>Two</h2>"
    uwsgi.async_sleep(20)
    yield "<h3>Three</h3>"
```

http://projects.unbit.it/uwsgi/

Green threads/coroutine: uGreen and Greenlet

```
# uwsgi --async 4000 --ugreen --module myapp --callable async2

import uwsgi

def hello(who):
    uwsgi.async_sleep(10)
    print(who)
    uwsgi.suspend()

def async2(env, start_response):
    headers = [('Content-Type', 'text/html')]
    start_response('200 OK', headers)

    yield "<h1>One</h1>"
    hello("Serena")
    yield "<h2>Two</h2>"
    hello("Alessandro")
    yield "<h3>Three</h3>"
```

http://projects.unbit.it/uwsgi/

uWSGI api

uWSGI api

stats

```
uwsgi.total_requests()
uwsgi.workers()
uwsgi.reload()
...
```

uWSGI api

stats

```
uwsgi.total_requests()
uwsgi.workers()
uwsgi.reload()
...
```

fd/socket api

```
uwsgi.wait_fd_read(fd[,timeout])
uwsgi.wait_fd_write(fd[,timeout])
...
```

http://projects.unbit.it/uwsgi/

uWSGI api

fd/socket api

```
uwsgi.wait_fd_read(fd[,timeout])
uwsgi.wait_fd_write(fd[,timeout])
...
```

stats

```
uwsgi.total_requests()
uwsgi.workers()
uwsgi.reload()
...
```

rpc

```
uwsgi.register_rpc("hello", hello_world)
uwsgi.rpc("192.168.0.1:3031", "hello")
...
```

http://projects.unbit.it/uwsgi/

uWSGI api

fd/socket api

```
uwsgi.wait_fd_read(fd[,timeout])
uwsgi.wait_fd_write(fd[,timeout])
...
```

stats

```
uwsgi.total_requests()
uwsgi.workers()
uwsgi.reload()
...
```

rpc

```
uwsgi.register_rpc("hello", hello_world)
uwsgi.rpc("192.168.0.1:3031", "hello")
...
```

caching

```
uwsgi.cache_set("pippo","pluto")
uwsgi.cache_get("pippo")
...
```

http://projects.unbit.it/uwsgi/

uWSGI api

fd/socket api

```
uwsgi.wait_fd_read(fd[,timeout])
uwsgi.wait_fd_write(fd[,timeout])
...
```

stats

```
uwsgi.total_requests()
uwsgi.workers()
uwsgi.reload()
...
```

rpc

```
uwsgi.register_rpc("hello", hello_world)
uwsgi.rpc("192.168.0.1:3031", "hello")
...
```

caching

```
uwsgi.cache_set("pippo","pluto")
uwsgi.cache_get("pippo")
...
```

queue

```
uwsgi.queue_push("ciao")
uwsgi.queue_pop()
uwsgi.queue_pull()
...
```

http://projects.unbit.it/uwsgi/

The Spooler

```
# uwsgi --spooler myqueue --module myapp --socket /tmp/uwsgi.sock

import uwsgi
import time

def manage_spool_request(vars)
    print(vars)
    time.sleep(40)


uwsgi.spooler = manage_spool_request


def application(e, sr)
    hh = [('Content-Type', 'text/plain')]
    sr('200 OK', hh)

    uwsgi.spool({'func': 'topogigio'})

    uwsgi.spool(func=topolino,gigio=topo)

    return "tasks enqueued"
```

http://projects.unbit.it/uwsgi/

The uWSGI-Signal subsystem

```
import uwsgi

def i_am_the_signal_handler(num):
    print("I am the signal handler %d" % num)

# register a signal handler
uwsgi.register_signal(1, '', i_am_the_signal_handler)

# raise a signal
uwsgi.signal(1)
```

http://projects.unbit.it/uwsgi/

The uWSGI-Signal subsystem

```
import uwsgi

def i_am_the_signal_handler(num):
    print("I am the signal handler %d" % num)

# register a signal handler
uwsgi.register_signal(1, '', i_am_the_signal_handler)

# raise a signal
uwsgi.signal(1)


# add a timer, raise signal 1 every 3 seconds
uwsgi.add_timer(1, 3)

# another timer, raise signal 2 every 20 seconds
uwsgi.add_timer(2, 20)

# a red black timer

uwsgi.add_rb_timer(3, 10)
```

http://projects.unbit.it/uwsgi/

The uWSGI-Signal subsystem

```
import uwsgi

def i_am_the_signal_handler(num):
    print("I am the signal handler %d" % num)

# register a signal handler
uwsgi.register_signal(1, '', i_am_the_signal_handler)

# raise a signal
uwsgi.signal(1)


# add a timer, raise signal 1 every 3 seconds
uwsgi.add_timer(1, 3)

# another timer, raise signal 2 every 20 seconds
uwsgi.add_timer(2, 20)

# a red black timer

uwsgi.add_rb_timer(3, 10)

# monitor /tmp directory and raise signal 4 at each mod

uwsgi.add_file_monitor(4, "/tmp")
```

http://projects.unbit.it/uwsgi/

The uWSGI-Signal subsystem

```
import uwsgi

def i_am_the_signal_handler(num):
    print("I am the signal handler %d" % num)

# register a signal handler
uwsgi.register_signal(1, '', i_am_the_signal_handler)

# raise a signal
uwsgi.signal(1)


# add a timer, raise signal 1 every 3 seconds
uwsgi.add_timer(1, 3)

# another timer, raise signal 2 every 20 seconds
uwsgi.add_timer(2, 20)

# a red black timer

uwsgi.add_rb_timer(3, 10)

# monitor /tmp directory and raise signal 4 at each mod

uwsgi.add_file_monitor(4, "/tmp")


# add a cron-like signal

# raise signal 5 every hour
uwsgi.add_cron(5, 59, -1, -1 , -1, -1)
```

http://projects.unbit.it/uwsgi/

config, config and config

```
# uwsgi --socket @foobar --wsgi-file pinax.wsgi --chdir /var/apps/pinax --master --processes 4
```

http://projects.unbit.it/uwsgi/

config, config and config

```
# uwsgi --socket @foobar --wsgi-file pinax.wsgi --chdir /var/apps/pinax --master --processes 4
```

--xml

```
<uwsgi>
<socket>@foobar</socket>
<wsgi-file>pinax.wsgi</wsgi-file>
<chdir>/var/apps/pinax</chdir>
<master/>
<processes>4</processes>
</uwsgi>
```

http://projects.unbit.it/uwsgi/

config, config and config

```
# uwsgi --socket @foobar --wsgi-file pinax.wsgi --chdir /var/apps/pinax --master --processes 4
```

--xml

```
<uwsgi>
<socket>@foobar</socket>
<wsgi-file>pinax.wsgi</wsgi-file>
<chdir>/var/apps/pinax</chdir>
<master/>
<processes>4</processes>
</uwsgi>
```

--ini

```
[ini]
socket = @foobar
wsgi-file = pinax.wsgi
chdir = /var/apps/pinax
master = true
processes = 4
```

http://projects.unbit.it/uwsgi/

config, config and config

```
# uwsgi --socket @foobar --wsgi-file pinax.wsgi --chdir /var/apps/pinax --master --processes 4
```

--xml

```
<uwsgi>
<socket>@foobar</socket>
<wsgi-file>pinax.wsgi</wsgi-file>
<chdir>/var/apps/pinax</chdir>
<master/>
<processes>4</processes>
</uwsgi>
```

--ini

```
[ini]
socket = @foobar
wsgi-file = pinax.wsgi
chdir = /var/apps/pinax
master = true
processes = 4
```

--yaml

```
uwsgi:
  socket: @foobar
  wsgi-file: pinax.wsgi
  chdir: /var/apps/pinax
  master: 1
  processes: 4
```

http://projects.unbit.it/uwsgi/

config, config and config

```
# uwsgi --socket @foobar --wsgi-file pinax.wsgi --chdir /var/apps/pinax --master --processes 4
```

--xml

```
<uwsgi>
<socket>@foobar</socket>
<wsgi-file>pinax.wsgi</wsgi-file>
<chdir>/var/apps/pinax</chdir>
<master/>
<processes>4</processes>
</uwsgi>
```

--ini

```
[ini]
socket = @foobar
wsgi-file = pinax.wsgi
chdir = /var/apps/pinax
master = true
processes = 4
```

--yaml

```
uwsgi:
  socket: @foobar
  wsgi-file: pinax.wsgi
  chdir: /var/apps/pinax
  master: 1
  processes: 4
```

--sqlite3

```
|socket|@foobar|
|wsgi-file|pinax.wsgi|
|chdir|/var/apps/pinax|
|master|1|
|processes|4|
```

http://projects.unbit.it/uwsgi/

config, config and config

```
# uwsgi --socket @foobar --wsgi-file pinax.wsgi --chdir /var/apps/pinax --master --processes 4
```

--xml

```
<uwsgi>
<socket>@foobar</socket>
<wsgi-file>pinax.wsgi</wsgi-file>
<chdir>/var/apps/pinax</chdir>
<master/>
<processes>4</processes>
</uwsgi>
```

--ini

```
[ini]
socket = @foobar
wsgi-file = pinax.wsgi
chdir = /var/apps/pinax
master = true
processes = 4
```

--yaml

```
uwsgi:
  socket: @foobar
  wsgi-file: pinax.wsgi
  chdir: /var/apps/pinax
  master: 1
  processes: 4
```

--sqlite3

```
|socket|@foobar|
|wsgi-file|pinax.wsgi|
|chdir|/var/apps/pinax|
|master|1|
|processes|4|
```

--json

```
{'socket':'@foobar', 'wsgi-
file': 'pinax.wsgi',
'chdir': '/var/apps/pinax',
'master': 1, 'processes':
4}
```

uWSGI

http://projects.unbit.it/uwsgi/

Going massive

uWSGI

http://projects.unbit.it/uwsgi/

Going massive

Virtualhosting + multiple interpreter ?

http://projects.unbit.it/uwsgi/

Going massive

Virtualhosting + multiple interpreter ?

Emperor

```
# uwsgi --emperor /etc/uwsgi/vassals

# uwsgi --emperor /etc/uwsgi/vassals/*/config.ini

# uwsgi --emperor /var/apps/*/*/config.*
```

http://projects.unbit.it/uwsgi/

Going massive

Virtualhosting + multiple interpreter ?

Emperor

```
# uwsgi --emperor /etc/uwsgi/vassals

# uwsgi --emperor /etc/uwsgi/vassals/*/config.ini

# uwsgi --emperor /var/apps/*/*/config.*
```

FastRouter and HTTP router/proxy

```
# uwsgi --fastrouter /tmp/uwsgi.sock --fastrouter-use-cache
# uwsgi --fastrouter 192.168.173.1:5050 --fastrouter-user-base /tmp/sockets/
# uwsgi --fastrouter :5051 --fastrouter-subscription-server 192.168.173.20:1717

# uwsgi --master --subscribe-to 192.168.173.20:1717:unbit.it --module myapp -p 4

# uwsgi --http :8080 --module werkzeug.testapp:test_app

# uwsgi --http :9090 --http-to /tmp/uwsgi.sock

# uwsgi --http :8000 --http-subscription-server 192.168.0.1:2230
```

http://projects.unbit.it/uwsgi/

Going massive

Virtualhosting + multiple interpreter ?

Emperor

```
# uwsgi --emperor /etc/uwsgi/vassals

# uwsgi --emperor /etc/uwsgi/vassals/*/config.ini

# uwsgi --emperor /var/apps/*/*/config.*
```

FastRouter and HTTP router/proxy

```
# uwsgi --fastrouter /tmp/uwsgi.sock --fastrouter-use-cache
# uwsgi --fastrouter 192.168.173.1:5050 --fastrouter-user-base /tmp/sockets/
# uwsgi --fastrouter :5051 --fastrouter-subscription-server 192.168.173.20:1717

# uwsgi --master --subscribe-to 192.168.173.20:1717:unbit.it --module myapp -p 4

# uwsgi --http :8080 --module werkzeug.testapp:test_app

# uwsgi --http :9090 --http-to /tmp/uwsgi.sock

# uwsgi --http :8000 --http-subscription-server 192.168.0.1:2230
```

Clustering

```
# uwsgi --cluster 225.1.1.1:1717 --master --processes 4 --module myapp --socket 192.168.173.*:3031

# uwsgi --cluster 225.1.1.1:1717
```

http://projects.unbit.it/uwsgi/

Multiprotocol and Mongrel2

```
# uwsgi --socket :8080 --protocol http --module myapp
# uwsgi --socket :8080 --protocol fastcgi --module myapp

# uwsgi --zmq tcp://192.168.0.1:9999,tcp://192.168.0.1:9998 --module myapp
```

I have no time....

Erlang integration
Zerg mode
Snapshotting
uwsgicc

uWSGI

http://projects.unbit.it/uwsgi/

In the Cloud

DotCloud

AppHosted

******Cloud (will be announced...stay tuned...)

Q/A