Driving VMware with Python: vixpy

Andrew Neitsch

2015-11-27

github.com/andrewdotn/vixpy

Outline

Testing services

- Unit tests are great for individual modules of code

Testing services

- Unit tests are great for individual modules of code
- End-to-end testing requires deploying the code ...

Testing services

- Unit tests are great for individual modules of code
- End-to-end testing requires deploying the code ... to multiple machines ...

Testing services

- Unit tests are great for individual modules of code
- End-to-end testing requires deploying the code … to multiple machines …
- setting up prerequisites like databases …

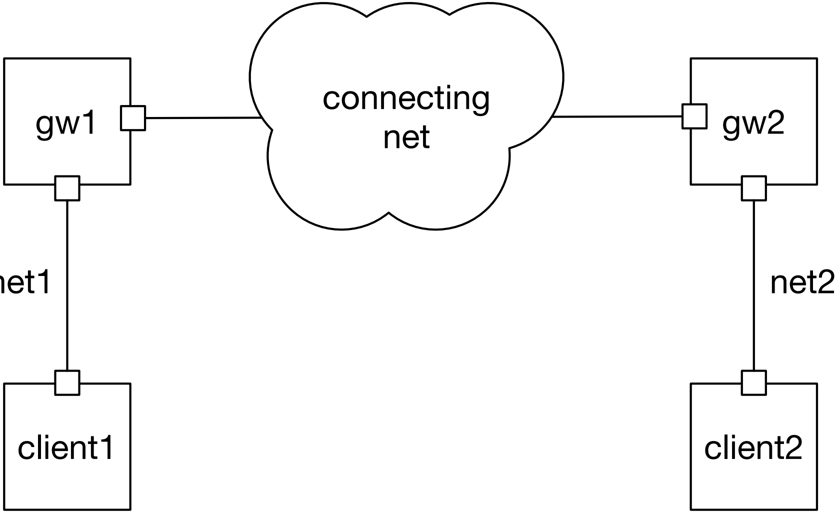
Testing services

- Unit tests are great for individual modules of code
- End-to-end testing requires deploying the code … to multiple machines …
- setting up prerequisites like databases …
- and configuration management …

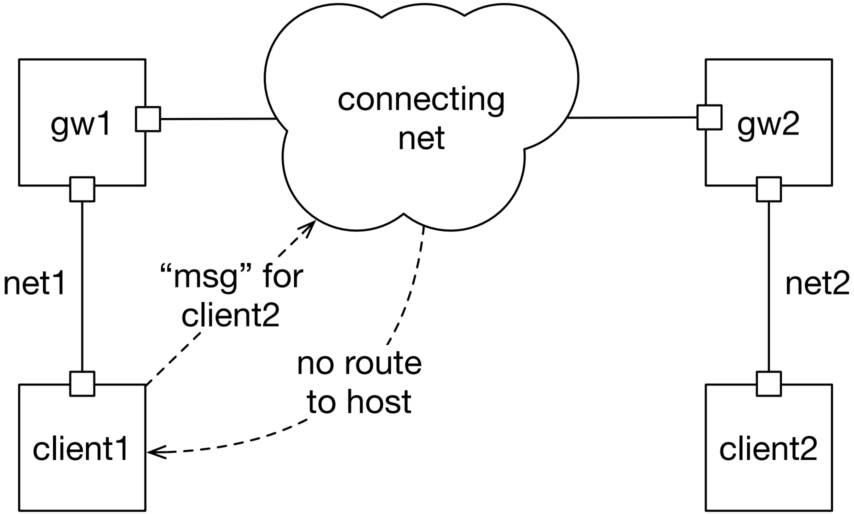
Testing services

- Unit tests are great for individual modules of code
- End-to-end testing requires deploying the code ... to multiple machines ...
- setting up prerequisites like databases ...
- and configuration management ...
- and sometimes with specialized networking needs too.

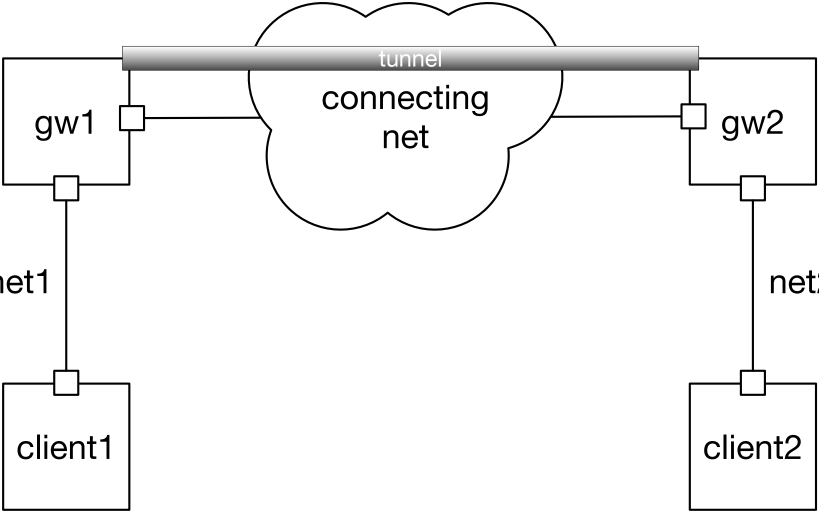
Example: OpenVPN



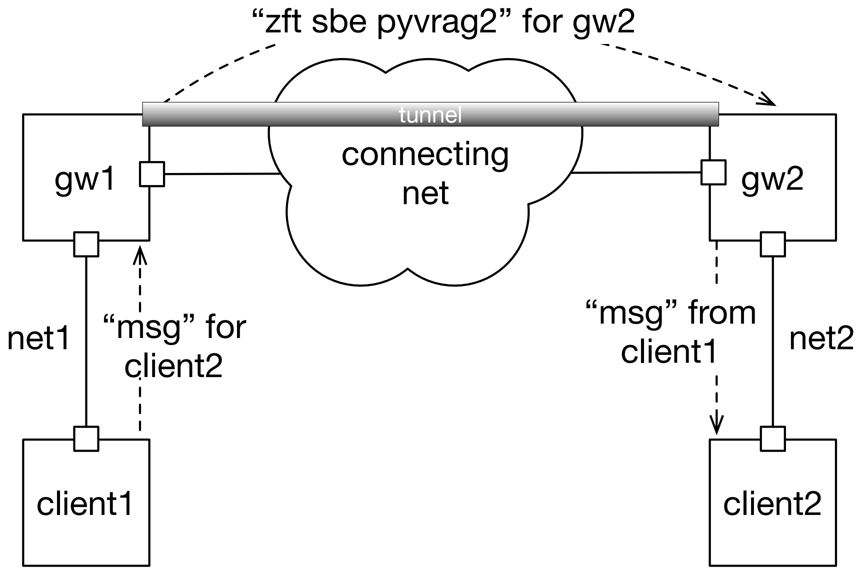
Example: OpenVPN



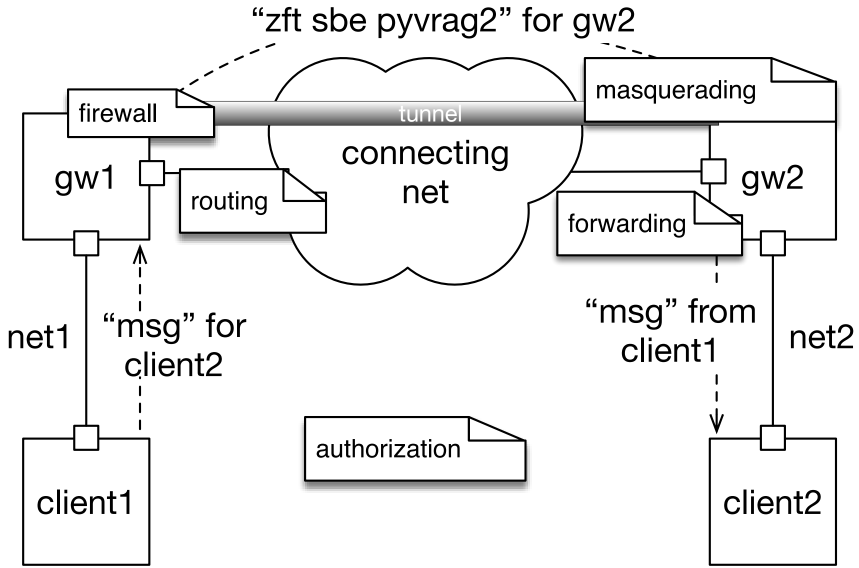
Example: OpenVPN



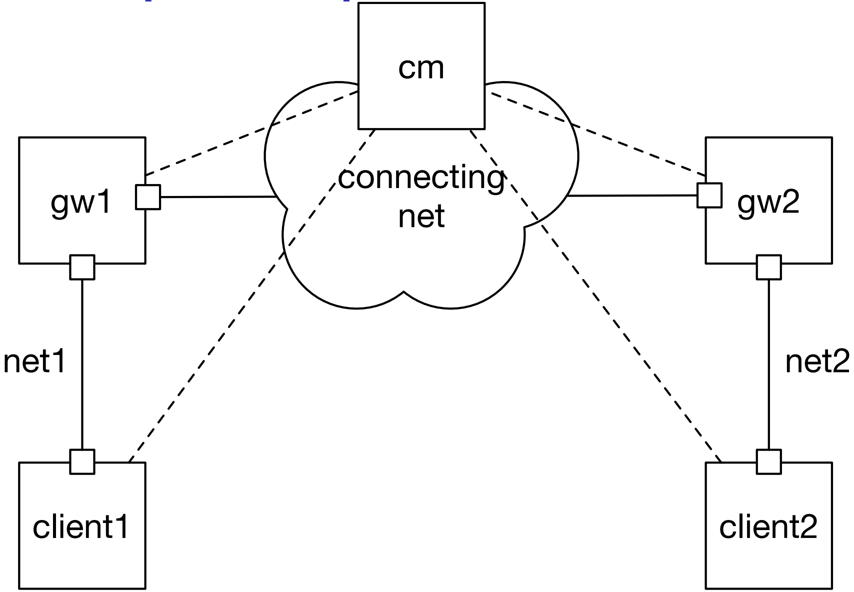
Example: OpenVPN



Example: OpenVPN



Example: OpenVPN



Example: OpenVPN

Testing this requires at least 4 VMs
and three networks

Typical solution: Vagrant

Vagrantfile:

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/centos-7.1"

 %w[client1 gw1 gw2 client2 cm].each do |name|
    config.vm.define name
  end
end

$ vagrant up

...
```

Typical solution: Vagrant

Advantages:

- declarative specification file
- `vagrant` command-line tool
- support for 20+ cloud providers

Typical solution: Vagrant

Disadvantages for automatic service testing:

- declarative specification file
- `vagrant` command-line tool
- support for 20+ cloud providers

Typical solution: Vagrant

Disadvantages for automatic service testing:

- declarative specification file
  - Tricky to configure dynamically
- `vagrant` command-line tool
- support for 20+ cloud providers

Typical solution: Vagrant

Disadvantages for automatic service testing:

- declarative specification file
- `vagrant` command-line tool
  - no API for interacting with VMs
- support for 20+ cloud providers

Typical solution: Vagrant

Disadvantages for automatic service testing:

- declarative specification file
- `vagrant` command-line tool
- support for 20+ cloud providers
  - lowest-common-denominator support
  - no snapshots
  - no cloning

Typical solution: vagrant

- really, really slow
- making a mistake means starting over
- can't actually test OpenVPN due to networking issues

VMware Fusion/Workstation

- Really fast
- Linked clones
- Snapshots

VMware Fusion/Workstation

- Really fast
- Linked clones
- Snapshots

Can we drive it programatically to get something faster and easier to use?

vmrun

VMware Fusion.app/Contents/Library/vmrun

```
start stop reset suspend pause unpause listSnapshots
snapshot deleteSnapshot revertToSnapshot runProgramInGuest
fileExistsInGuest directoryExistsInGuest
setSharedFolderState addSharedFolder removeSharedFolder
enableSharedFolders disableSharedFolders
listProcessesInGuest killProcessInGuest runScriptInGuest
deleteFileInGuest createDirectoryInGuest
deleteDirectoryInGuest listDirectoryInGuest
renameFileInGuest captureScreen writeVariable readVariable
getGuestIPAddress list upgradevm installTools
checkToolsState deleteVM clone
```

VIX

`VMware Fusion.app/Contents/Public`

`vix.h` and shared library

vmware.com/support/developer/vix-api

```
jobHandle = VixVM_PowerOn(vmHandle,
                          VMPOWEROPTIONS,
                          VIX_INVALID_HANDLE,
                          NULL, // *callbackProc,
                          NULL); // *clientData);
err = VixJob_Wait(jobHandle, VIX_PROPERTY_NONE);
if (VIX_FAILED(err)) {
    printf("PowerOn failed\n");
    goto fail;
}
```

Cython

- Python code gets compiled into a Python C module
- Can do whatever C or Python can do
- Didn't have to read the manual
- Though there is an O'Reilly book

Example: Python to C

```
def hi(x):
    return x + 42
```

Python to C and back

```
$ cython foo.pyx
$ wc -l foo.c
1737
$ cat foo.c
...
/* "foo.pyx":2
 * def hi(x):
 *     return x + 42             # <<<<<<<<<<<<<<
 */
  __Pyx_XDECREF(__pyx_r);
 __pyx_t_1 = __Pyx_PyInt_AddObjC(__pyx_v_x, __pyx_int_42, 42, 0);
  __Pyx_GOTREF(__pyx_t_1);
...
$ gcc -shared foo.c -o foo.so -lpython \
    -I/System/Library/Frameworks/Python.framework/Headers
$ python -c 'import foo; print foo.hi(42)'
84
```

Example: C to Python

```
$ cat foo.pyx
cdef extern:
    int printf(const char* s, ...)

def blah(x):
    printf("hi there, x is at %p\n", <void*>x)
$ cython ... && gcc ...
$ python -c 'import foo; foo.blah(42)'
hi there, x is at 0x7fe183605be8
```

Example: C to Python

```
$ cat foo.pyx
cdef extern:
    int printf(const char* s, ...)

def blah(x):
    printf("hi there, x is at %p\n", <void*>x)
$ cython ... && gcc ...
$ python -c 'import foo; foo.blah(42)'
hi there, x is at 0x7fe183605be8
```

cdefs aren't exported, idea seems to be to use them to create a pythonic interface

Cython for VIX

It just worked!

Cython for VIX

It just worked!

the code

Cython for VIX

Time to

- clone a VM
- run a command
- delete the VM

vagrant 69 seconds

Cython for VIX

Time to

- clone a VM
- run a command
- delete the VM

vagrant	69 seconds
vixpy	10 seconds

VixJob_Wait()

called with Python interpreter lock held

no point in spawning other threads—
they'll just block

Async callbacks

```
VixVM_PowerOn(vmHandle,
              VMPOWEROPTIONS,
              VIX_INVALID_HANDLE,
              NULL, // *callbackProc
              NULL); // *clientData
```

cython Demos/callback/cheese.pyx

Async callback trouble

```
78304 Segmentation fault: 11  python test.py
make: *** [all] Error 139
...
```

Async callback trouble

```
78304 Segmentation fault: 11  python test.py
make: *** [all] Error 139
...
lldb> expr *__pyx_t_1->ob_type
...
```

Async callback trouble

```
78304 Segmentation fault: 11  python test.py
make: *** [all] Error 139
...
lldb> expr *__pyx_t_1->ob_type
...
CFLAGS=-g ./configure && make
...
```

Async callback trouble

```
78304 Segmentation fault: 11  python test.py
make: *** [all] Error 139
...
lldb> expr *__pyx_t_1->ob_type
...
CFLAGS=-g ./configure && make
...
Fatal Python error: take_gil: NULL tstate
```

Eventually found `nogil`

```
cdef VixJob_Wait() nogil
...
with nogil:
    retCode = VixJob_Wait(...)
```

If you're careful not to use Python objects—

Enables standard threads to run concurrently!

Results

Time to clone five test VMs, run a command in each, and then destroy them:

Results

Time to clone five test VMs, run a command in each, and then destroy them:

vagrant	5 minutes 45 seconds

Results

Time to clone five test VMs, run a command in each, and then destroy them:

vagrant	5 minutes 45 seconds
vixpy	16 seconds

Results

Time to clone five test VMs, run a command in each, and then destroy them:

vagrant	5 minutes 45 seconds
vixpy	16 seconds

∼20x faster

Conclusions

CytHon is fantastic

- For calling C libraries
- For rewriting hot spots in C

Conclusions

Cython is fantastic

- For calling C libraries
- For rewriting hot spots in C

VIX is promising

Conclusions

Cython is fantastic

- For calling C libraries
- For rewriting hot spots in C

VIX is promising—future work:

- Clean up snapshots

Conclusions

Cython is fantastic

- For calling C libraries
- For rewriting hot spots in C

VIX is promising—future work:

- Clean up snapshots
- Allow showing GUI

Conclusions

Cython is fantastic

- For calling C libraries
- For rewriting hot spots in C

VIX is promising—future work:

- Clean up snapshots
- Allow showing GUI
- Flesh out APi

Conclusions

Cython is fantastic

- For calling C libraries
- For rewriting hot spots in C

VIX is promising—future work:

- Clean up snapshots
- Allow showing GUI
- Flesh out APi
- Iterate on my OpenVPN setup

Questions?

Call for talks

“w/o interesting talks, there’s not a ton of point in ‘meeting up’ ”

Call for talks

“w/o interesting talks, there’s not a ton of point in ‘meeting up’ ”

1. Pick something you find interesting
2. Talk about it
3. Include suggestions for stuff to hack on

Suggested exercises

- Get the code running on your machine
- Use Cython to call a useful C library
- Use vixpy to launch a multi-tier application