facebook

facebook

Python @ Facebook

Angelo Failla
Production Engineer @ Facebook Dublin
October 11th, 2014 -- PyCon Ireland 2014

Agenda

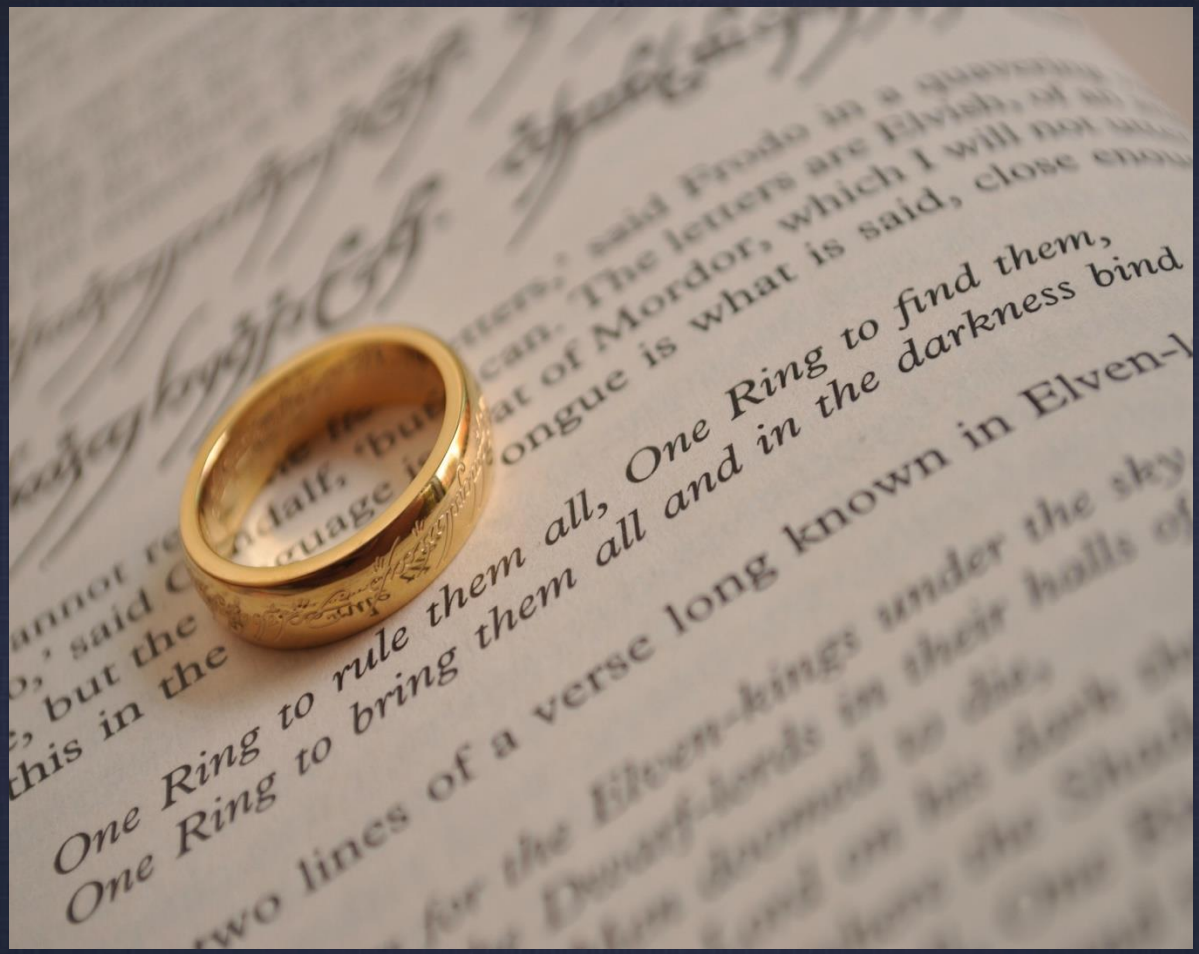
1. How do we use Python at Facebook?
2. Description of a production system based on Python
3. Questions

How do we use Python at Facebook?


facebook
INFRASTRUCTURE

- Centralized repo for back-end systems
- A LOT of code.
- All Python code hosted there
- 3rd most used language (PHP/Hack => C++ => Python)
- Still a LOT of code :P

ONE BUILD TOOL

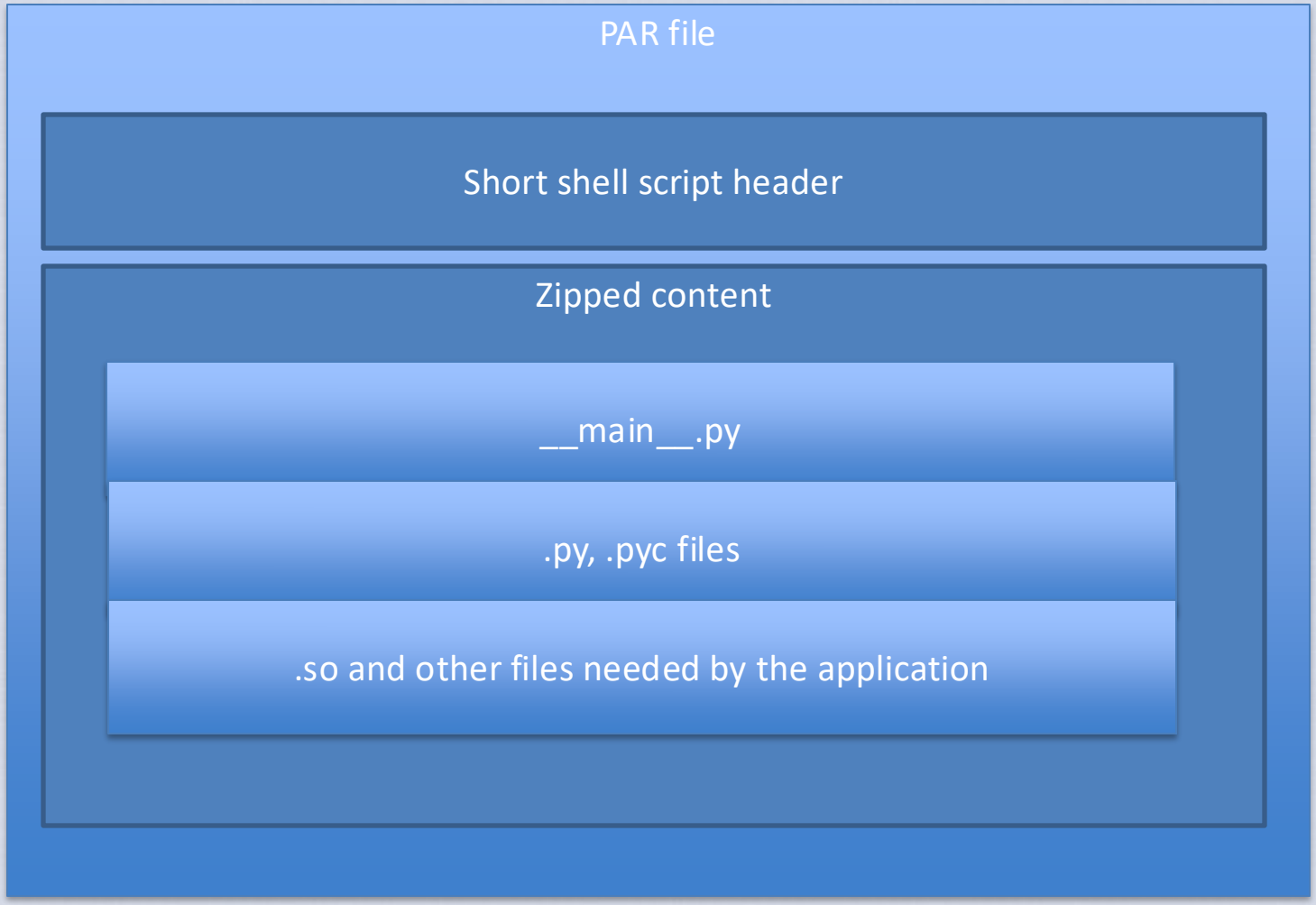


TO RULE THEM ALL

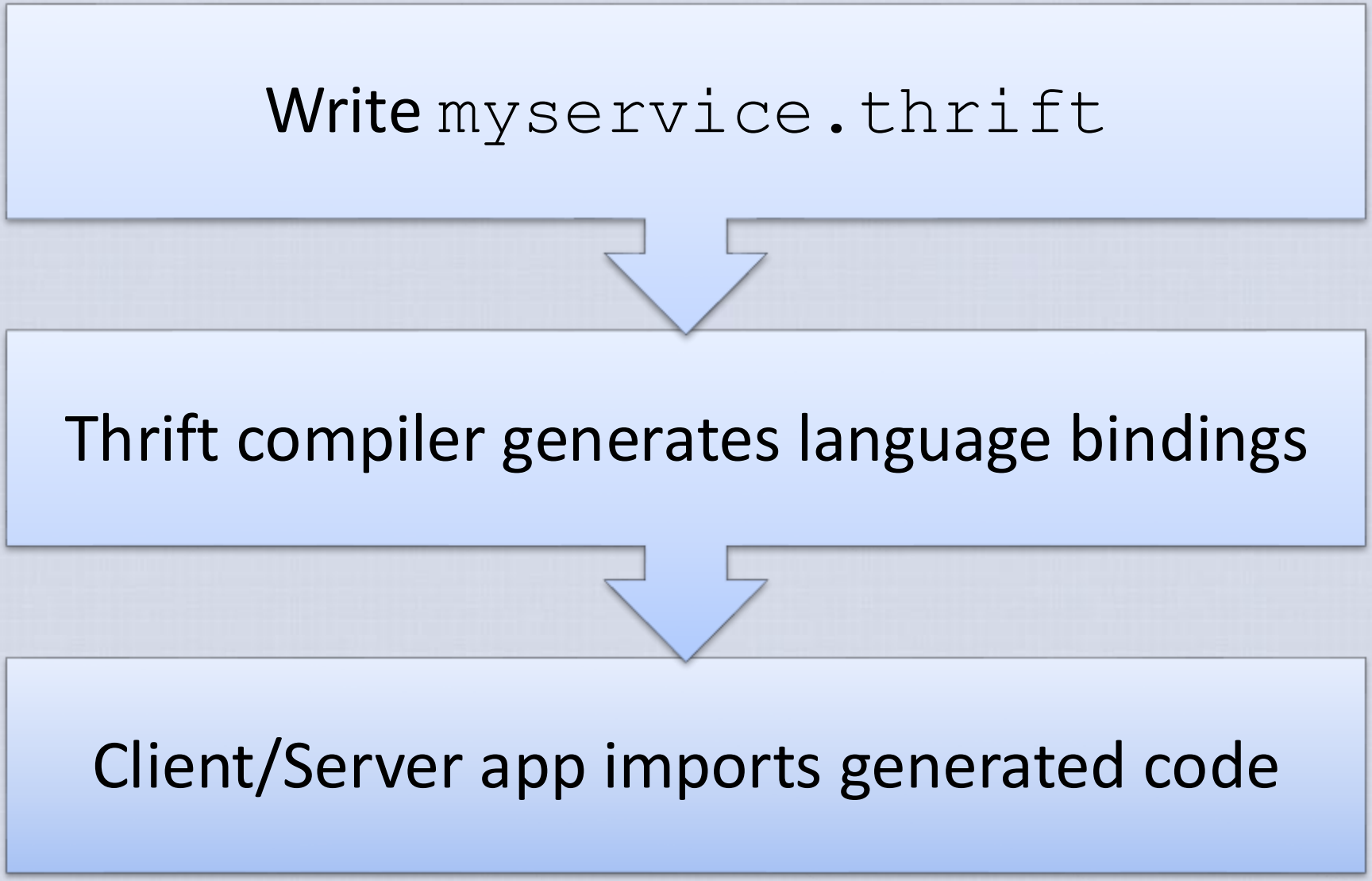
Everything is built from HEAD
C++ binaries are statically linked

http://cdn.instapop.com/assets/memes/Skeptical%20Baby/4302/thumb.jpeg?1383838705

How about the same in Python? We use .par files



Thrift (https://thrift.apache.org/) in a nutshell



```
# cat
~/repo/thrift_example/myservice.thrift

namespace py thrift_example.myservice

service Calculator {
  i32 add(1: i32 num1, 2: i32 num2)
}
```

```
$ thrift --gen py myservice.thrift

$ tree gen-py/
gen-py/
├── __init__.py
└── myservice
    ├── Calculator.py
    ├── Calculator-remote
    ├── constants.py
    ├── __init__.py
    └── ttypes.py
```

```
# cat ~/fbcode/thrift_example/server.py
# import lines removed, see: http://tinyurl.com/p6a7cuw

 1: class CalculatorHandler(Calculator.Iface):
 2:   def add(self, num1, num2):
 3:     return num1 + num2
 4:
 5: handler = CalculatorHandler()
 6: socket = TSocket.TServerSocket(port=9090)
 7: tfactory = TTransport.TBufferedTransportFactory()
 8: pfactory = TBinaryProtocol.TBinaryProtocolFactory()
 9: server = TServer.TSimpleServer(
10:   handler, socket, tfactory, pfactory)
11: server.serve()
```

```
# cat ~/fbcode/thrift_example/client.py
# import lines removed, see: http://tinyurl.com/p6zz9ex

 1: socket = TSocket.TSocket('localhost', 9090)
 2: transport = TTransport.TBufferedTransport(socket)
 3: protocol = TBinaryProtocol.TBinaryProtocol(transport)
 4: client = Calculator.Client(protocol)
 5: transport.open()
 6:
 7: sum = client.add(1, 2)
 8: print('1 + 2 = %d' % sum)
 9:
10: transport.close()
```

```
# cat ~/repo/thrift_example/TARGETS
thrift_library(
    name = "myservice_thrift",
    languages = ["ruby", "php", "cpp", "python"],
    thrift_srcs = {"myservice.thrift": ["Calculator"]},
)

python_binary(
    name = "myservice_server",
    main_module = "thrift_example.server",
    srcs = ["server.py"],
    deps = [":myservice_thrift-py"],
)

python_binary(
    name = "myservice_client",
    main_module = "thrift_example.client",
    srcs = ["client.py"],
    deps = [":myservice_thrift-py"],
)
```

```
pallotron@dev:~/repo $ fbconfig thrift_example/
Configuring 20 targets
Writing 208/208 rules to makefile [100%]
Done

pallotron@dev:~/repo $ fbmake opt
 thrift/compiler/thriftl.ll
 common/memory/JEMalloc.cpp
 […]
Linking _build/opt/thrift/compiler/thrift...
 _build/opt/thrift/compiler/thrift
 thrift_example/myservice.thrift
 _build/opt/thrift_example/myservice_thrift-Calculator-pyremote.lpar
 _build/opt/thrift_example/myservice_client.lpar
 _build/opt/thrift_example/myservice_server.lpar
 _build/opt/thrift_example/myservice_server.par
 _build/opt/thrift_example/myservice_thrift-Calculator-pyremote.par
 _build/opt/thrift_example/myservice_client.par
```


BitTorrent™

Python as a DSL

Enter `configerator`...

Thrift schema:

```
struct Job {
  1: string name,
  2: Scheduling
       scheduling,
}
struct Scheduling {
  1: int priority,
}
```

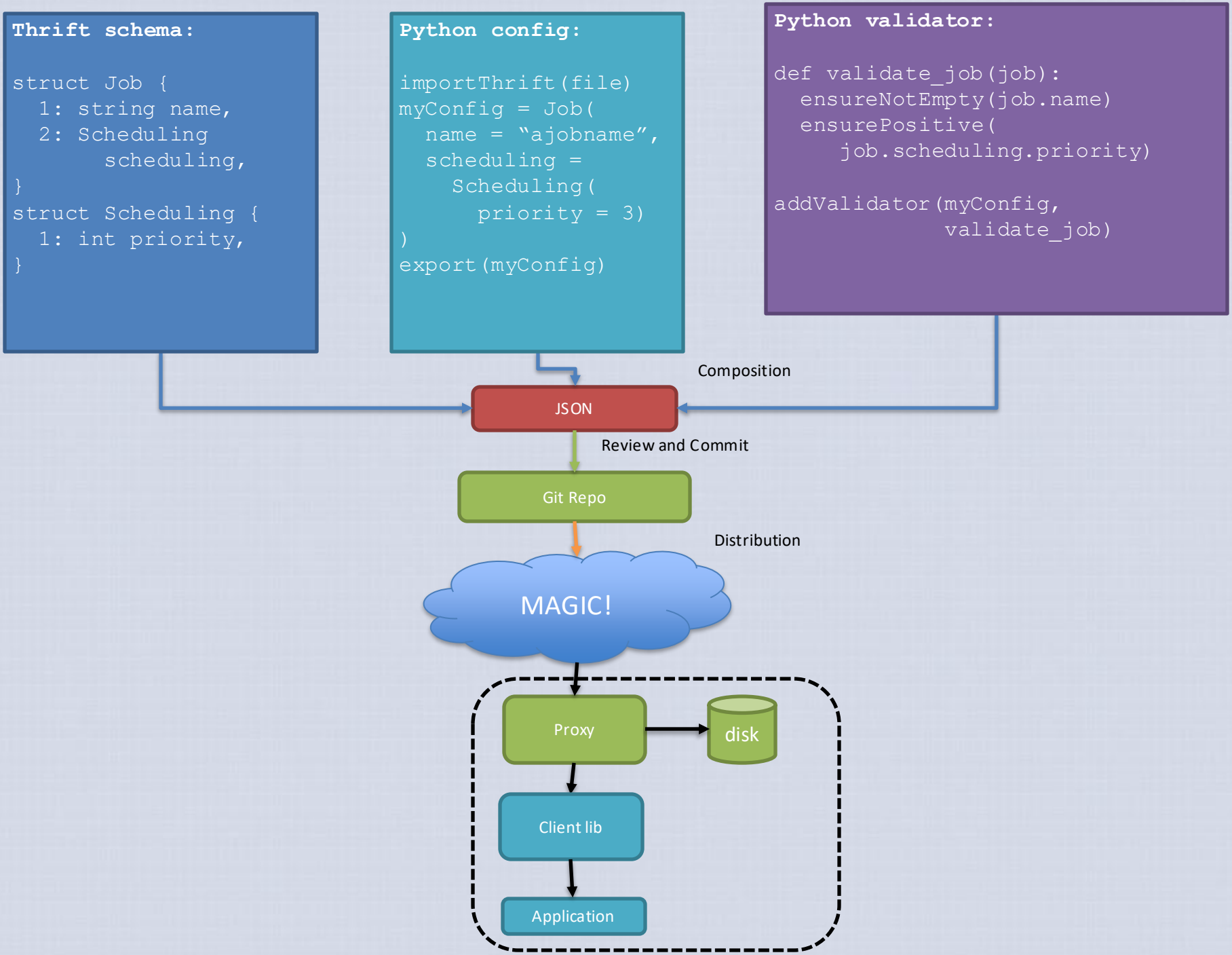
Python config:

```
importThrift(file)
myConfig = Job(
  name = "ajobname",
  scheduling =
    Scheduling(
      priority = 3)
)
export(myConfig)
```

Python validator:

```
def validate_job(job):
  ensureNotEmpty(job.name)
  ensurePositive(
     job.scheduling.priority)

addValidator(myConfig,
             validate_job)
```



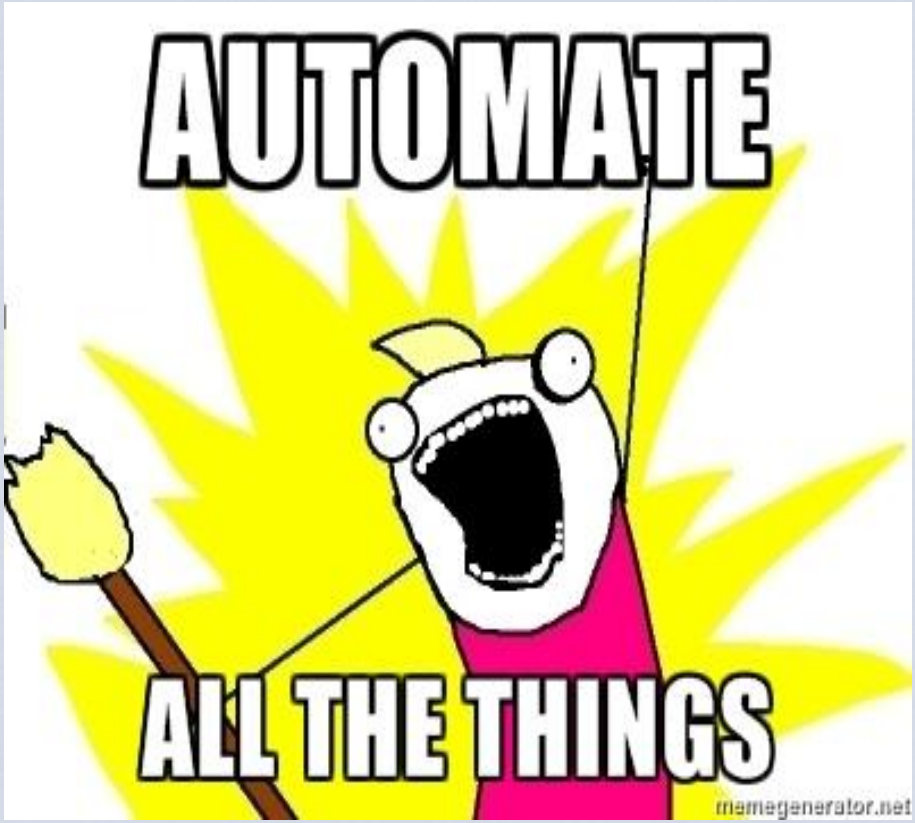
```
 1: def consume():
 2:
 3:  handle = dict()
 4:  ctc = ConfigeratorThriftClient()
 5:  fname = "example/config"
 6:
 7:   def on_update(name):
 8:      handle['config'] = ctc.getJSONConfigContent(name)
 9:
10:   ctc.startMonitoringConfig(fname)
11:   ctc.setUpdateCallback(on_update)
12:   on_update(fname)
13:
14:   return handle
```

Benefits of using `configerator`

Code review on configuration changes

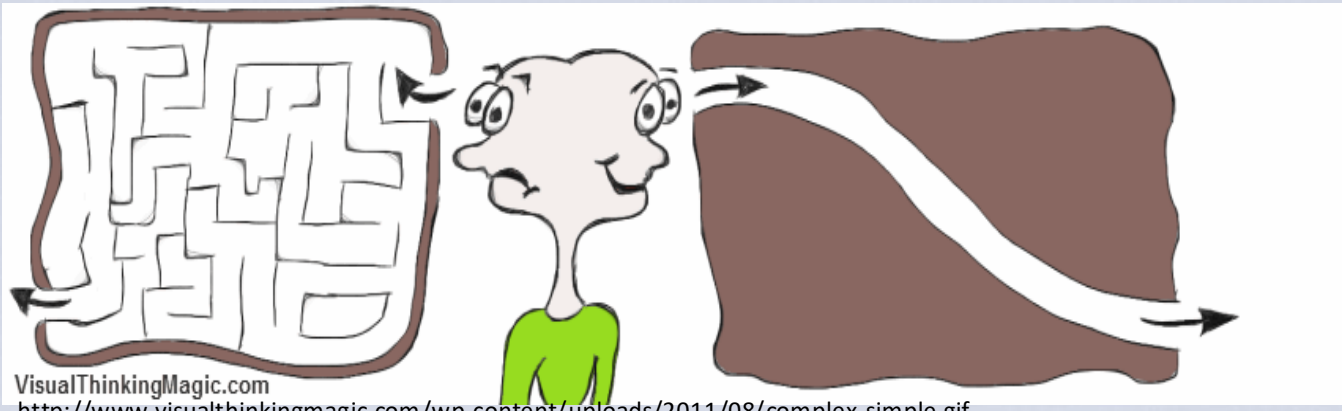
http://www.jasonawesome.com/wp-content/uploads/2010/06/sally-code-review-300x256.png

Automated validation



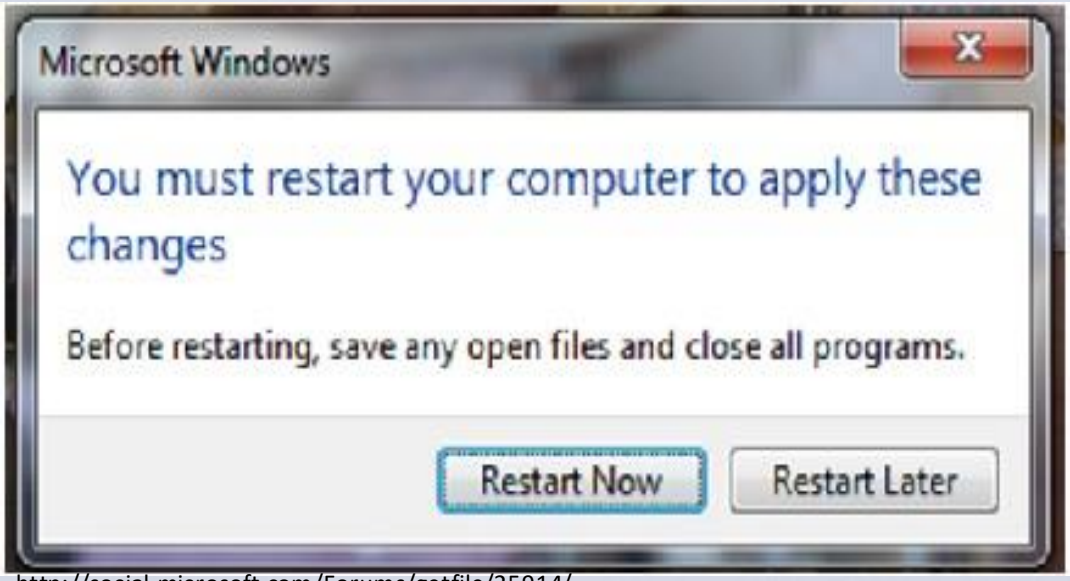
http://sounddesignlive.com/wp-content/uploads/2013/08/sound-design-live-automate_all_the_things.jpeg

Clear representation of complex config



http://www.visualthinkingmagic.com/wp-content/uploads/2011/08/complex-simple.gif

No service restart required to pick up changes

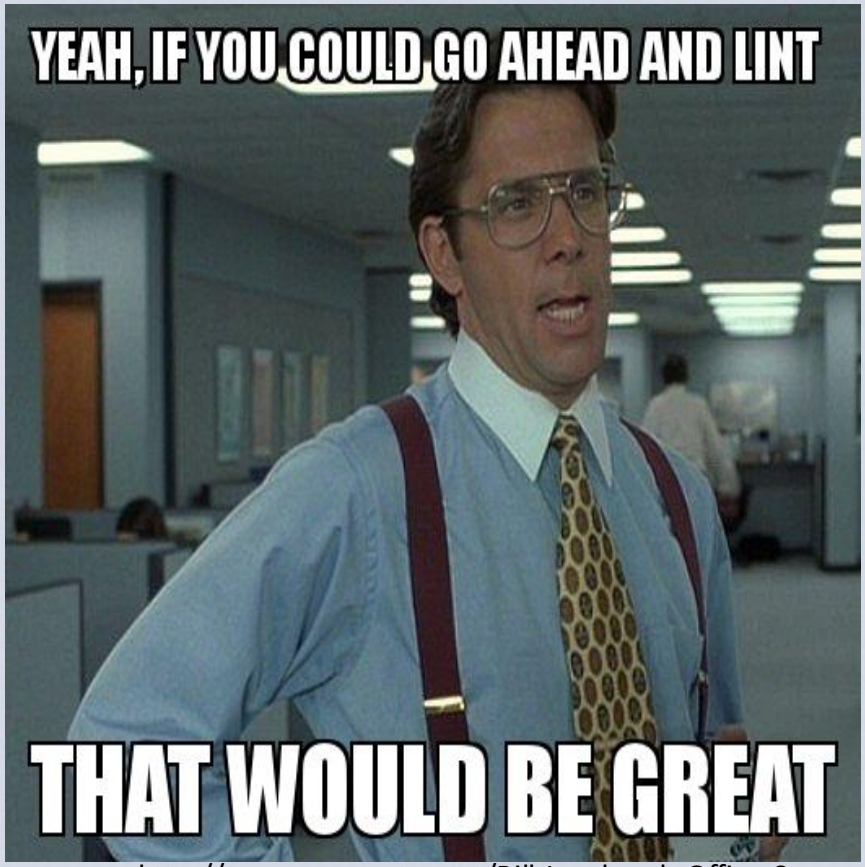


http://social.microsoft.com/Forums/getfile/25914/

Code review at FB

- Code *is* peer reviewed
- We use Phabricator
- Originally developed @ FB by Evan Priestley
- Go check www.phabricator.org
- Supports git/svn/hg

- **(Some) PEPs enforced and push blocking**
- **Common code dir for accelerated development**
- **Unit tests results are clearly visible in our code review tool**



http://memegenerator.net/Bill-Lumbergh-Office-Space

Suggestions for reviewers:

- Reviewing code is not writing code
- Nitpicks are not enough
- Stop typing and have a conversation
- Build trust
- Find people to emulate

Making diffs easier to be reviewed:

- One and only one thesis
- Do No Harm
- Mix and match reviewers
- Write good test plans
- You are not your code

Production Python at Facebook

PROJECT KEANU

- Automated load tests of regions/clusters/hosts
- Implements feedback control loop
- No simulated traffic
- Goal is to find out “breaking point” == real capacity
- Runs daily
  - Peek time
  - When no incidents
  - Skips drained clusters

KEANU

Questions?

facebook