Python Charming at Cisco

Sergiu Daniluk

Technical Consulting Engineer – ACI Solutions

19 March 2019

What we will cover

- Introduction
- Python, what is it?
- Why (not) Python?
- Projects
- Lessons we learned
- Ask me Anything!

A little about
me

- On planet Earth since 1990
- With Cisco since 2015
  - Data Center Switching
  - ACI Solutions
- Previously:
  - Network & Security Engineer (RO)
- I like Python

A little about Cisco TAC

- World's Best Support Organization
  - According to TSIA
  - 10 years in a row!
- Customer Service & Advocacy
- Engineers gonna Engineer
- Grassroots innovation

Python?

```
def fibonacci(n):
    a, b = 0, 1
    for i in range(n):
        yield a
        a, b = b, a + b
```

```
for i in range(1,101):
    fizz = "Fizz"*(1 - i**2%3)
    buzz = "Buzz"*(1 - i**4%5)
    print(fizz+buzz or i)
```

- High-level, General purpose
- Friendly & Easy to Learn
- Open-source & Cross-Platform
- “Batteries Included”
- Extensive and mature ecosystem

- Too OO / not OO enough
- Whitespace = syntax
- Duck typing
- Slow
- No ’real’ async

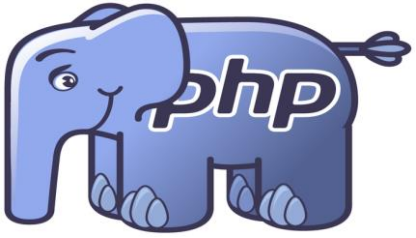
But why?

- Automate the boring stuff
  - and the repetitive stuff
- Forge your own swords
- Glue things that don't fit together
- Meet in the middle
- Interacting with APIs

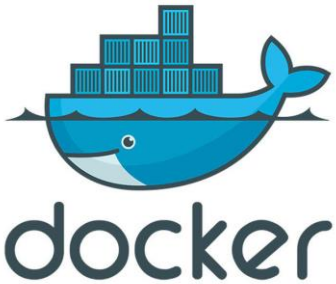
```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Why not ...

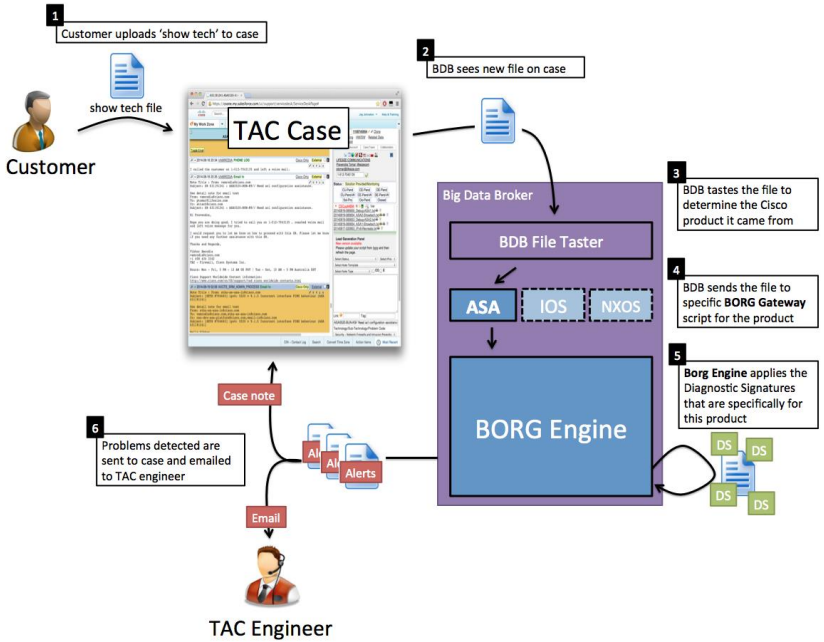


The Ecosystem



What we have built

BDB/Borg



- General Purpose execution framework
- Backend in Go, runs scripts in Python
- 15000+ scripts performing automated analysis for all platforms
- Pioneer award winner in 2016

SquareWheels

```
from squarewheels4.nxos import MDSShowTechDetail
show_tech = MDSShowTechDetail('show_tech_det.txt')
```

```
sh_ver = show_tech['show version'].text
print(sh_ver[:85]) # first 85 characters
```

```
Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
```

```
sh_hard = show_tech["show hardware"]
sh_hard['switch_hardware']['model_number']
```

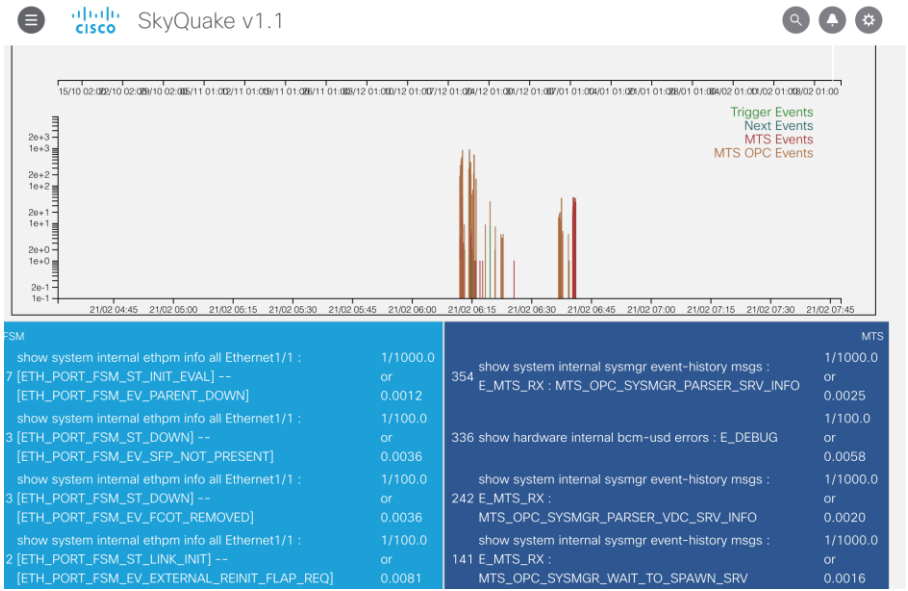
```
'DS-C9513'
```

```
sh_int = show_tech['show interface'].data
for interface, i_data in sh_int.items():
    if i_data['i_errors'] > 0:
        print(interface)
```

```
fc4/1
fc5/1
```

- Python library to extract from support dumps and parse text to data
- Performance critical parts in C
- Written by Kris Vandecruys

SkyQuake



- Anomaly detection on events
- HTML / TypeScript on the front-end
- Leverages other in-house libraries

show hard int cpu-mac inband stats

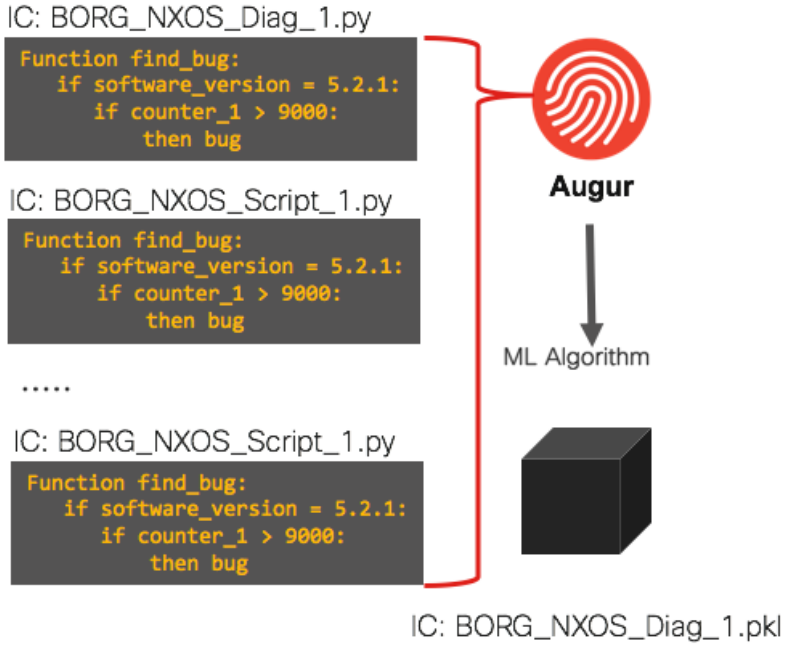
```
qdisc stats:
----------------+---------
Tx queue depth . 10000
qlen ........... 0
packets ........ 2828653891
bytes .......... 2838030451241
drops .......... 8102
```

show diagnostic result …

```
Error code ------------------> DIAG TEST UNTESTED
Total run count -------------> 88187
Last test execution time ----> Tue Jul 25 09:53:12 2017
First test failure time -----> Thu Mar 20 17:46:10 2014
Last test failure time ------> Wed Jul 19 10:36:14 2017
Last test pass time ---------> Tue Jun 27 10:17:46 2017
Total failure count ---------> 27
Consecutive failure count ---> 3
Last failure reason ---------> Skipped testing, port cfg
                               state is unknown
Next Execution time ---------> Tue Jul 25 10:13:12 2017
```

- Anomaly detection on cli output
- Python Pre- and Post-Processing
- Deep Learning: Python Keras on TF
- API frontend: Flask

Augur



- Pure Python
- Prediction of issue detection
- Early stages of development

What we have learned

The Good

- Hackathons are an effective way to get people engaged, increase the skill level and produce results
- Python is versatile, thriving and people have fun automating the boring stuff
- Sharing your work is almost always a Good Thing

The Bad

- You can’t do everything with Python
- A good mix of skills requires time, energy and money
- Versioning and packaging ecosystem has a long way to go
- Working with different skills, you need to rewrite old code frequently

The Ugly

- Programming is just step 1: Don’t neglect:
  - Testing
  - Security
  - User experience
  - Lifecycle
  - Documentation
- “Holy Wars”
- Lots of ‘glue’ code due to heterogeneity of the organization

Ask me anything!


CISCO