

A network diagram on the left side of the slide features several circular nodes of varying sizes and colors (red, orange, and maroon) connected by thin grey lines. The nodes are arranged in a non-linear fashion, with some larger nodes acting as central hubs.

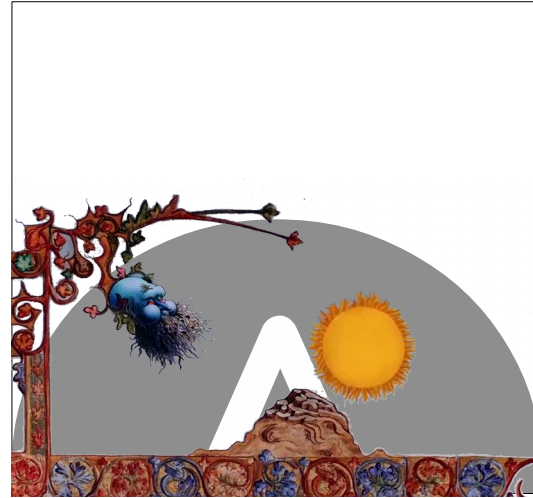
Rehabilitating Pickle

Alex Willmer
Developer, CGI
EuroPython 2018

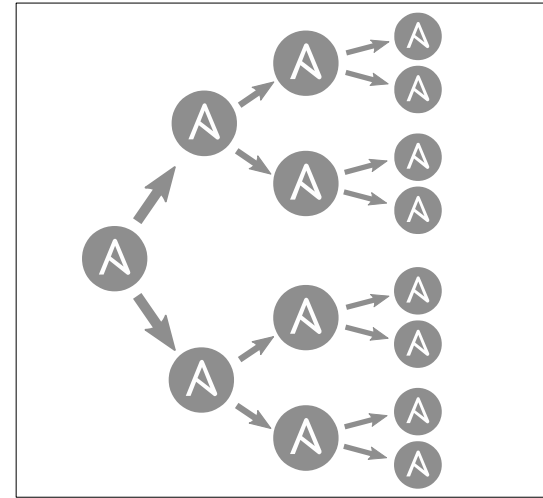
Previously at CGI



DevOps with Ansible
There is much rejoicing
(Yayyy)



Playbooks grow, get slow
Summer becomes Winter



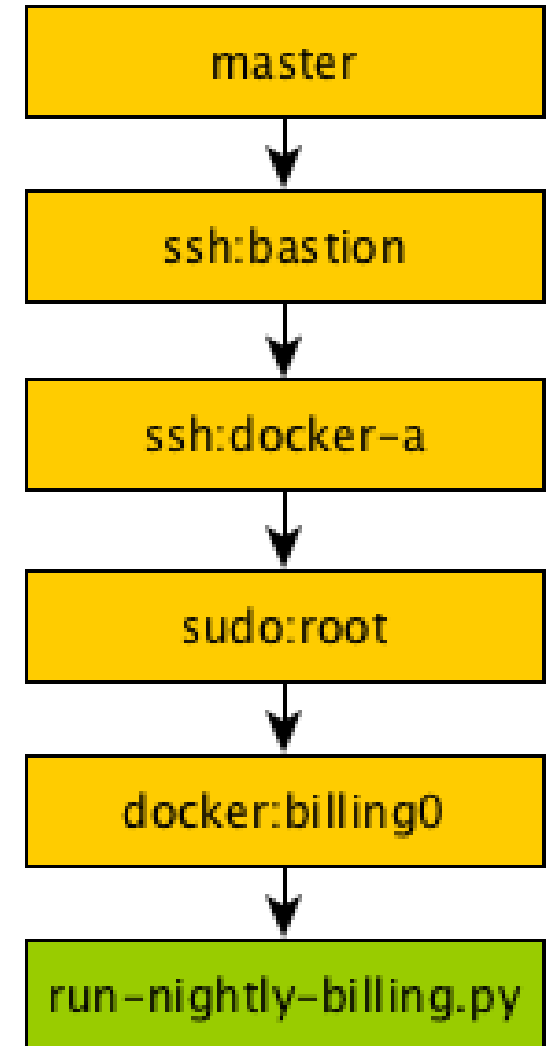
Mitogen for Ansible appears

als
with mitogen **my playbook runtime went from 45 minutes to just under 3 minutes.** Awesome work!”
The runtime was reduced from **1.5 hours on 4 servers to under 3 minutes.** Thanks!”
h, performance improvement using Mitogen is *huge* (as mentioned before, running with Mitogen enables tasks to complete or take a few seconds). Without Mitogen, the same task takes 19m49! **I’m not even deploying without Mitogen anymore :)”**
Works like a charm, thank you for your quick response and for trying it out. **He is not kidding about the speed increase!** I don’t know what kind of dark magic @dmw_83 has used, but his Mitogen strategy took Clojars’ Ansible runs from **45 minutes to 2 minutes.** I still can’t quite believe it.”

Ansible is fast again ...

Mitogen

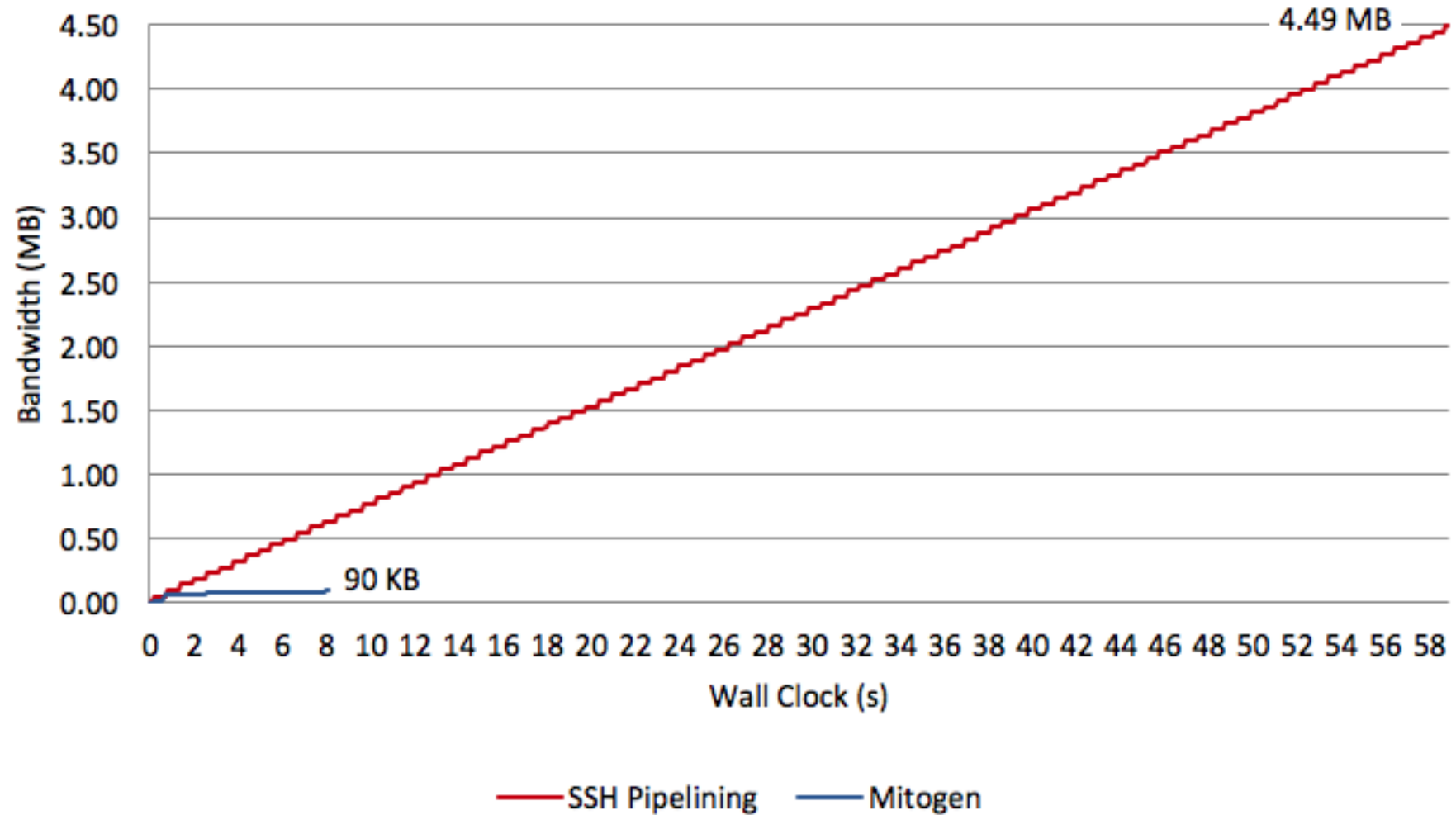
```
bastion = router.ssh(  
    hostname='jump-box.mycorp.com')  
  
docker_host = router.ssh(  
    via=bastion_host,  
    hostname='docker-a.prod.mycorp.com')  
  
sudo_account = router.sudo(  
    via=docker_host,  
    username='azure_diamond', password='hunter2')  
  
internal_box = router.docker(  
    via=sudo_account,  
    container='billing0')  
  
internal_box.call(os.system, './run-nightly-billing.py')
```



Mitogen for Ansible

```
# ansible.cfg
[defaults]
strategy_plugins = <...>/ansible_mitogen/plugins/strategy
strategy = mitogen_linear
```

Mitogen for Ansible



Pickle

```
>>> pickle.dumps((1., 'text', [2,], {3: None}),
...               protocol=0)
b'(\F1.0\nVtext\np0\n(lp1\nL2L\na(dp2\nL3L\nNstp3\n.'
```



```
>>> pickle.loads(_)
```

```
(1., 'text', [2,], {3: None})
```

Why Pickle?

- Works everywhere Python does
- Preserves nearly all Python types, & custom classes
- Handles recursive data structures
- Produces compact serializations

**DON'T USE
PICKLE**

Why not Pickle?

Warning: *The pickle module is not secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.*

[Python documentation » 12.1 pickle — Python object serialization](#)

pickle.load() executes arbitrary code

```
>>> pickle.loads(b"cos\nsystem\n"
...              b"(S'echo hello world'\ntR.")
hello world
0
```

Demo

Restricting globals

```
WHITELIST = {"set", "frozenset"}
```

```
class RestrictedUnpickler(pickle.Unpickler):  
    def find_class(self, module, name):  
        if module == "builtins" and name in WHITELIST:  
            return getattr(builtins, name)  
        raise pickle.UnpicklingError("Denied!")
```

Other attacks

Denial of service

- Protocol downgrade
- Billion Laughs (pickle bombs)
- Unhandled exceptions

Weird machines

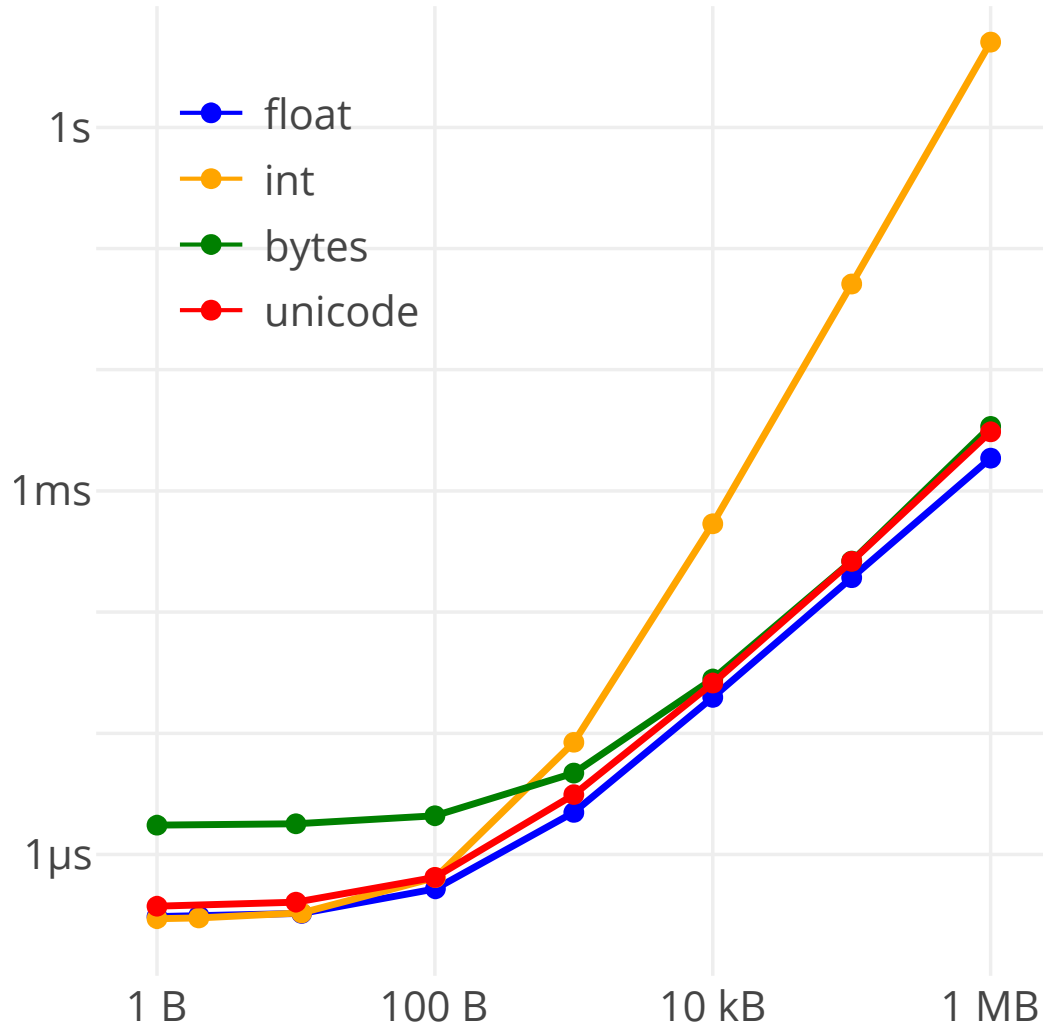
- Unused opcodes
- Parser abuse
- Stack corruption

Data Exposure

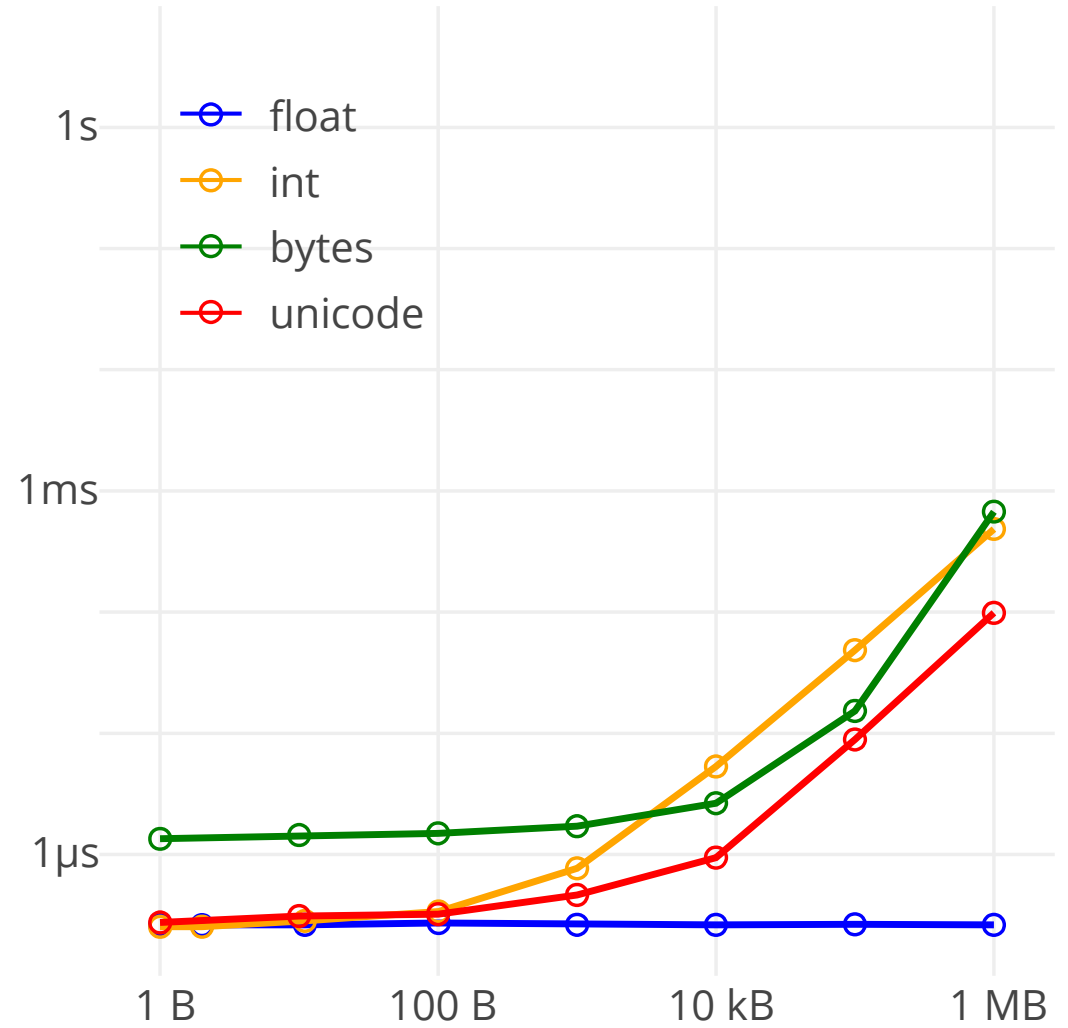
- Object traversal (gadget chains)

Protocol downgrade

Protocol 0



Protocol 1

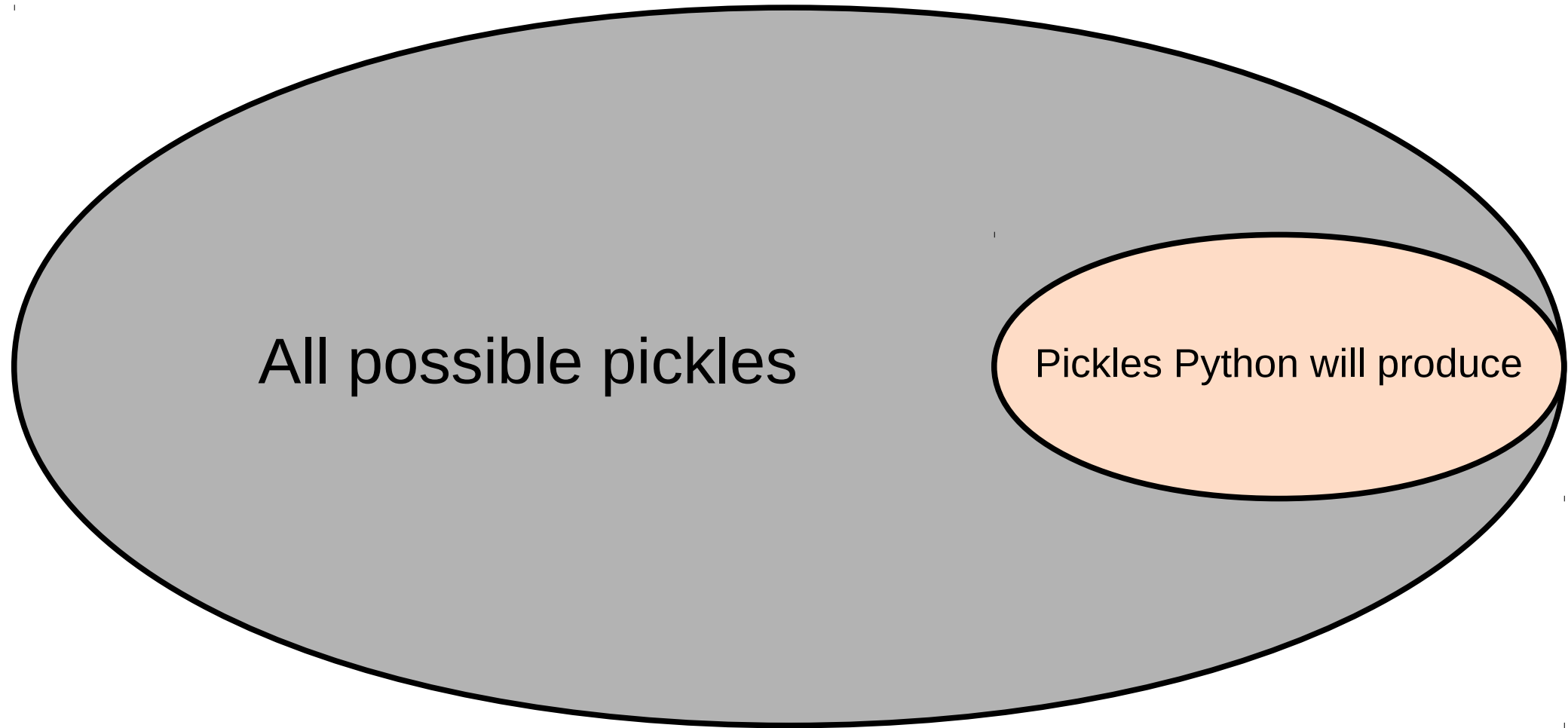


Billion laughs / pickle bombs

```
>>> a = ['lol', 'lol', 'lol', 'lol', 'lol',
...      'lol', 'lol', 'lol', 'lol', 'lol']
>>> b = [a, a, a, a, a, a, a, a, a, a]
>>> c = [b, b, b, b, b, b, b, b, b, b]
...
>>> laughs = [i, i, i, i, i, i, i, i, i, i]
>>> with open('billion-laughs.pkl', 'wb') as f:
...     pickle.dump(laughs, f)
```

Demo

Weird machines



Other considerations

Don't use pickle for long term storage

“Its automatic, magical behavior shackles you to the internals of your classes in non-obvious ways. You can't even easily tell which classes are baked forever into your pickles. Once a pickle breaks, figuring out why and where and how to fix it is an utter nightmare.”

Don't use Pickle – use Camel, Eevee

Thank you

Slides, more info

- github.com/moreati/pickle-fuzz

Mitogen, zero install distributed programs (by David Wilson)

- mitogen.readthedocs.io

Pikara, pickle analyzer (by Latacora)

- github.com/latacora/pikara

Sour pickles, real world pickle attacks (by Sensepost)

- sensepost.com/blog/2011/blackhat-2011-presentation/
- youtube.com/watch?v=HsZWFMKsM08

Don't use pickle – use camel (by Eevee)

- eev.ee/release/2015/10/15/dont-use-pickle-use-camel/

CGI are hiring



Alex Willmer

Developer, CGI

alex.willmer@cgi.com



@moreati