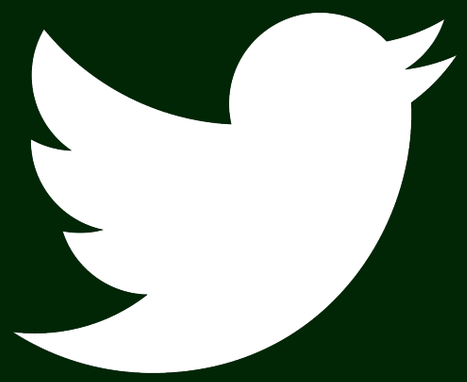
A Deep Dive Into Requests

H1



@lukasaoz

@lukasa

Metaswitch Networks

PROJECT CALICO

Requests Core Contributor

urllib3 Core Contributor

Maintainer of 10+ others

You Know This

```
r = requests.get('http://www.google.com/')
print r.content
```

Let's Learn Something New

Sessions

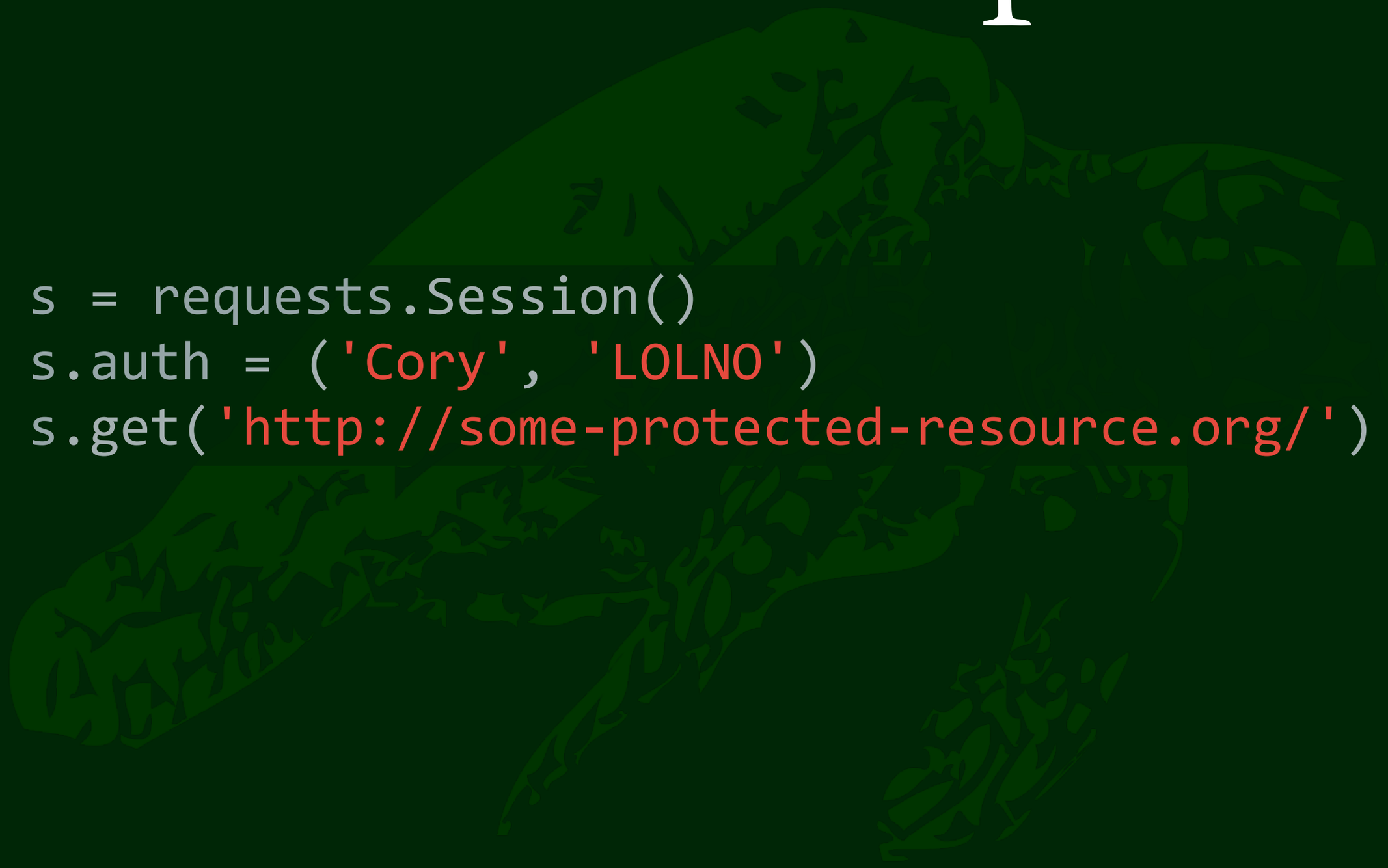
Sessions

- Requests' Persistence Layer
- Manage cookies
- Avoid boilerplate
- Connection Pooling!

Cookie Management

```
>>> import requests
>>> s = requests.Session()
>>> s.get('http://www.google.com/')
>>> len(s.cookies)
2
```

Avoid Boilerplate



```
>>> s = requests.Session()
>>> s.auth = ('Cory', 'LOLNO')
>>> s.get('http://some-protected-resource.org/')
```

Connection Pooling

```
# Two TCP Connections
>>> requests.get('http://www.google.com/')
>>> requests.get('http://www.google.com/')
```

```
# Re-use the same TCP connection
>>> s = requests.Session()
>>> s.get('http://www.google.com/')
>>> s.get('http://www.google.com/')
```

Transport Adapters

- Change the way requests manages connections.
- Useful for applying connection-scope changes.
- Think: SSL version, protocol etc.

Streaming

```
# Does not download response body
>>> r = requests.get(url, stream=True)

# Now, iterate over the body, in chunks...
>>> [x for x in r.iter_content(1024)]
# or lines...
>>> [x for x in r.iter_lines()]

# Downloads all at once
>>> r.content
```

SSL Client Certs

```
>>> r = requests.get(url, cert=(
        'cert.crt', ‘cert.key’
    ))
```

Unusual Auth

- OAuth 1 and 2: requests_oauthlib
- Kerberos: requests_kerberos
- NTLM: requests_ntlm

Related Projects

- requests-toolbelt: Utilities
- trequests, treq, grequests, async-requests
- requests-ftp
- urllib3, hyper

Thanks!

@lukasaoz

@lukasa