Create Your Computing Infrastructure

. . . with Python!

Ed Leafe
Rackspace
PyTexas
August 2013

About Me

ed.leafe@rackspace.com

ed@openstack.org

Google+: +EdLeafe

Twitter: @EdLeafe

About Me

Python Developer for Rackspace

About Me

Worked on OpenStack since its inception

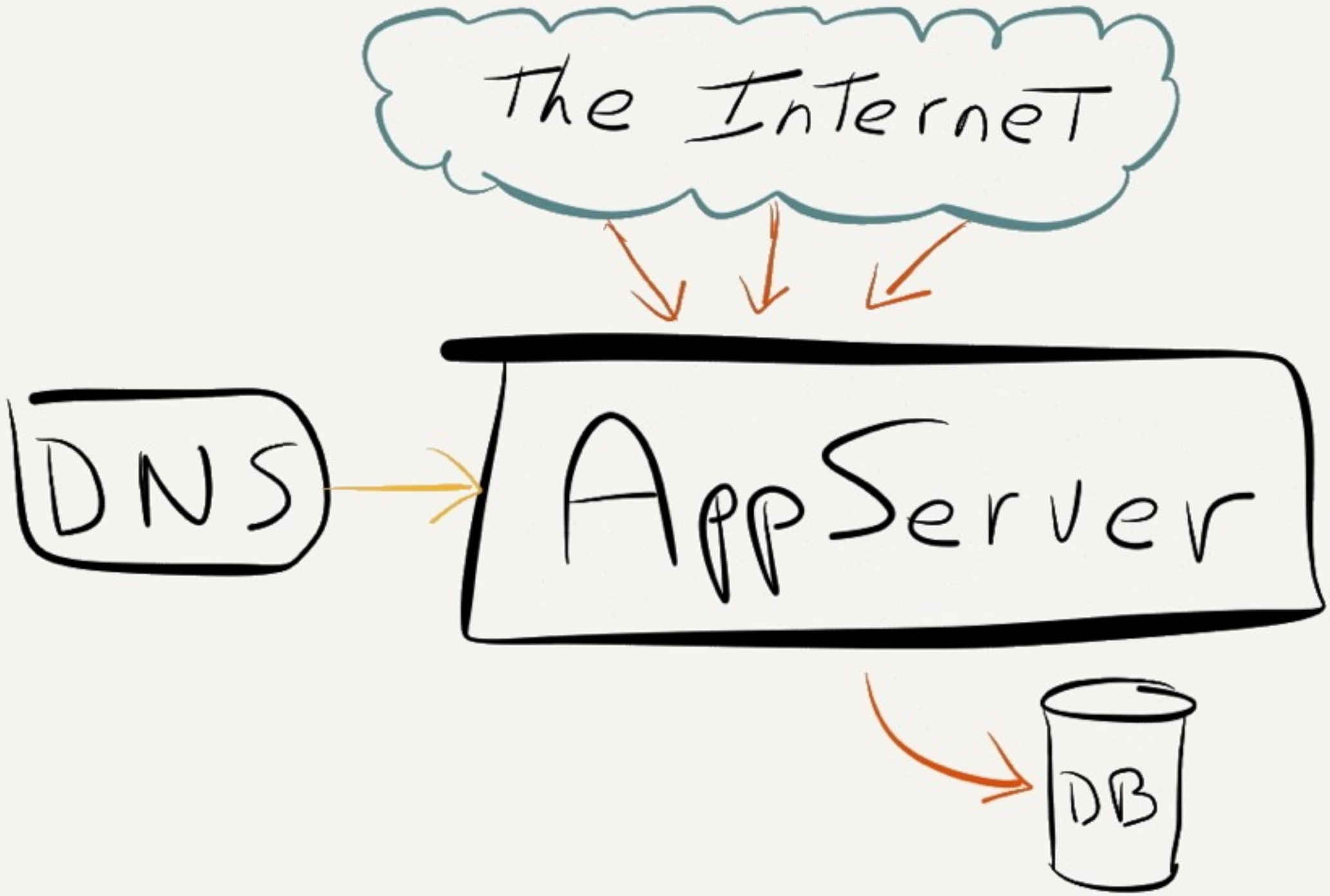
About Me

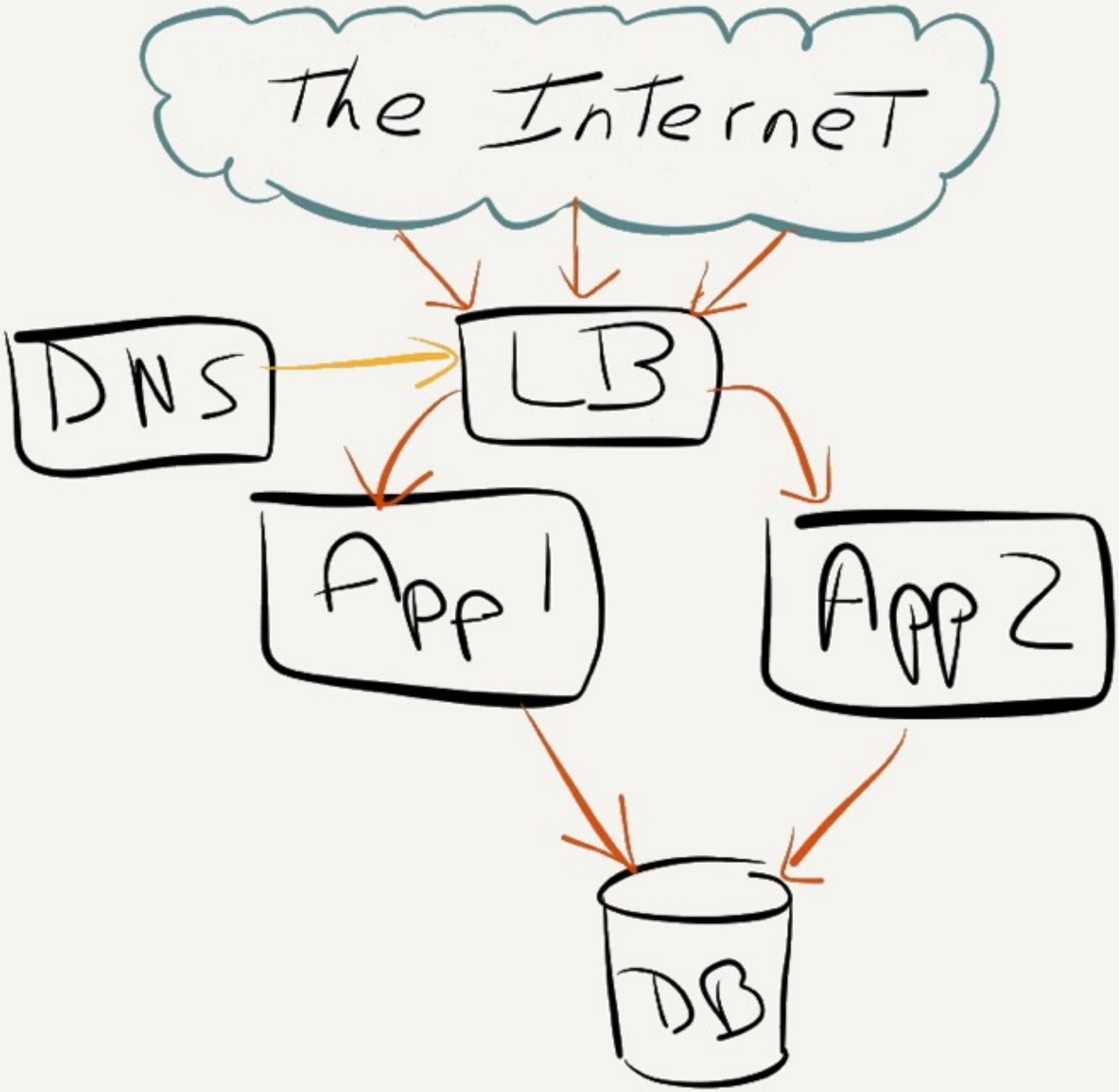
Creator of **pyrax**

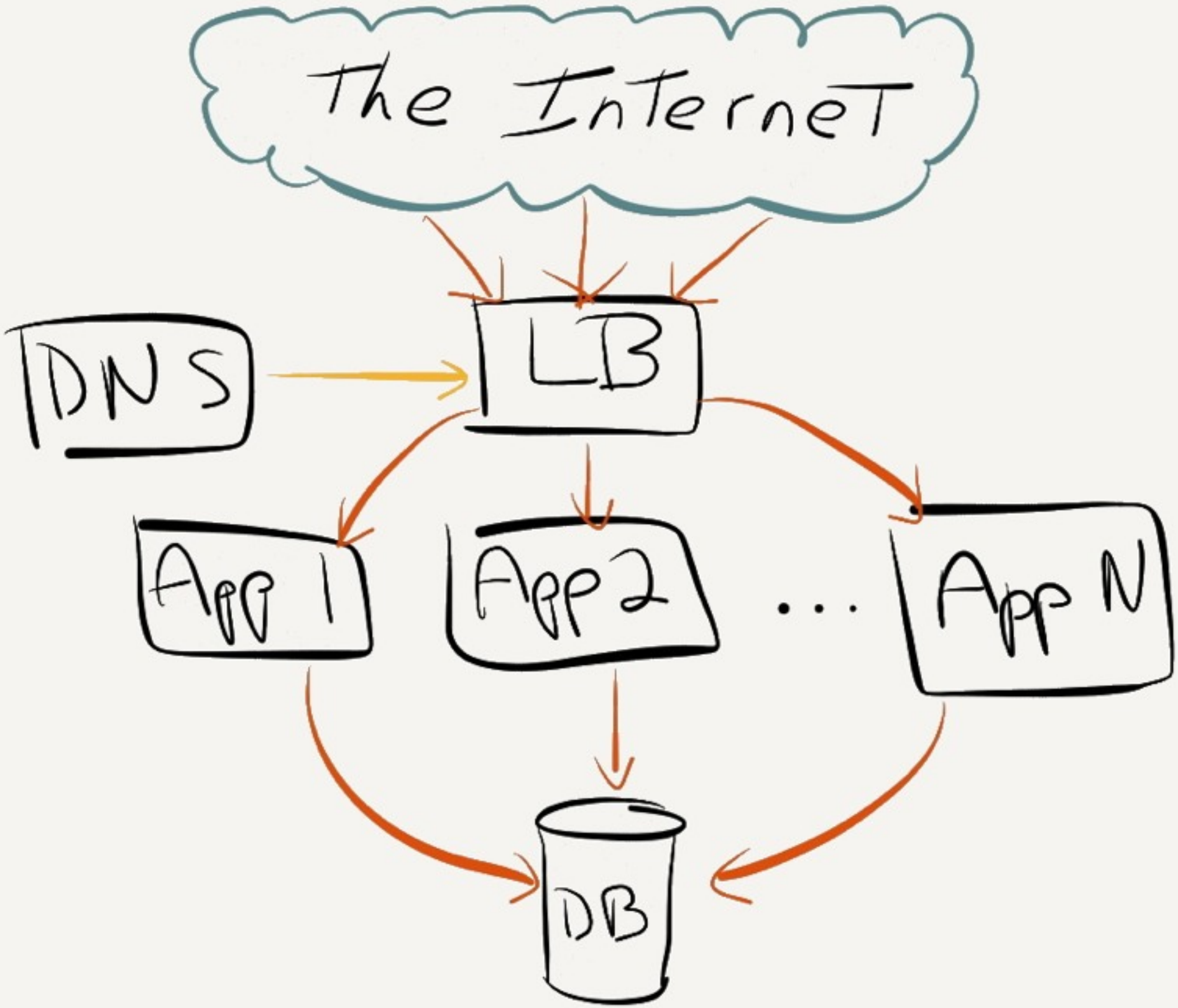
The Python SDK for
OpenStack Clouds

You just wrote an awesome app!

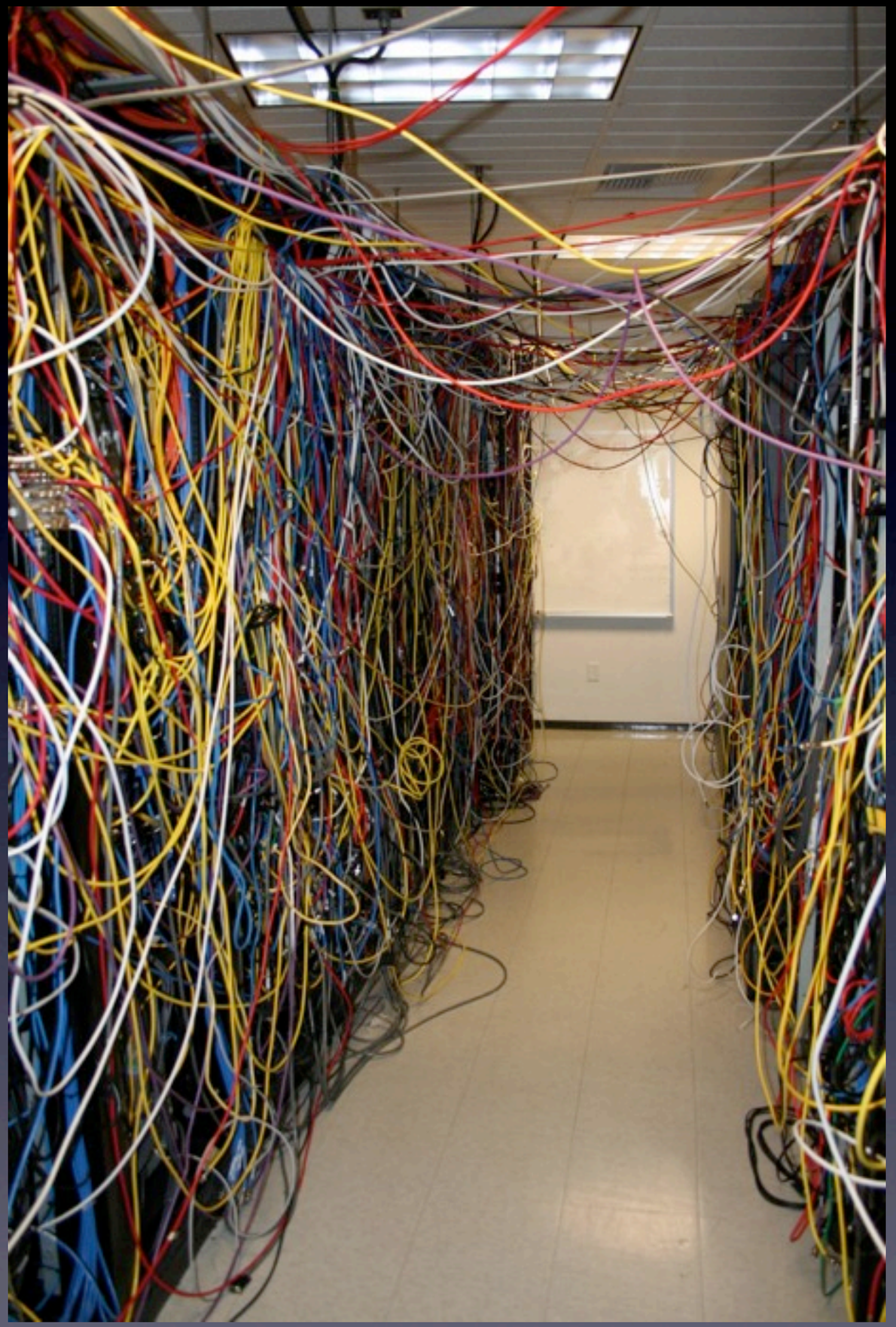
Server Infrastructure


The Internet
DNS
AppServer
DB


the Internet
DNS
LB
App 1
App 2
DB


the Internet
DNS
LB
App 1
App 2
...
App N
DB

Hardware



Enter…The Cloud

Enter…The Cloud

- Provision and configure hardware in advance

Enter…The Cloud

- Provision and configure hardware in advance
- Resources can now be allocated dynamically

Enter…The Cloud

- Provision and configure hardware in advance
- Resources can now be allocated dynamically
- Resources can now be allocated

  programmatically!

APIs

APIs

Provide general purpose, low-level access

APIs

Flexible and powerful…

APIs

Flexible and powerful… but tedious!

APIs

- Documented at:
  - http://docs.rackspace.com/
  - http://api.openstack.org/api-ref.html

Jump right in!

Let's get a list of our Cloud Servers!

```
import urllib2
uri = "https://dfw.servers.api.rscloud.com/"
      "v2/753591/servers/detail"
resp = urllib2.urlopen(uri)
print resp.read()
```

Oh, yeah...

Oh, yeah...

```
HTTP Error 401: Unauthorized
```

So let's handle auth

```
uri = "http://identity.api.rackspacecloud.com/v2.0/tokens"
credentials = {"auth": {"passwordCredentials":
        {"username": "pytexas",
        "password": "topsecret"}}}
req = urllib2.Request(uri, data=json.dumps(credentials)
req.add_header("Content-Type", "application/json")
```

```
try:
    raw_resp = urllib2.urlopen(req)
except urllib2.HTTPError as e:
    errcode = e.getcode()
    if errcode == 401:
        # Invalid authorization
        print "Invalid credentials"
    else:
        print "Authentication Error: %s" % e
resp = json.load(raw_resp)
```

Not done yet...

```
token_info = resp.get("access").get("token")
token = token_info.get("id")
expires = token_info.get("expires")
tenant_info = token_info.get("tenant")
tenant_id = tenant_info.get("id")
```

Still not finished...

```
access = resp.get("access")
svc_catalog = access.get("serviceCatalog")
# Let's see what's in this
print svc_catalog
```

```
[{u'endpoints': [{u'publicURL': u'https://monitoring.api.rackspacecloud.com/v1.0/753591',
    u'tenantId': u'753591'}],
  u'name': u'cloudMonitoring',
  u'type': u'rax:monitor'},
 {u'endpoints': [{u'publicURL': u'https://ord.loadbalancers.api.rackspacecloud.com/v1.0/753591',
    u'region': u'ORD',
    u'tenantId': u'753591'},
   {u'publicURL': u'https://dfw.loadbalancers.api.rackspacecloud.com/v1.0/753591',
    u'region': u'DFW',
    u'tenantId': u'753591'}],
  u'name': u'cloudLoadBalancers',
  u'type': u'rax:load-balancer'},
 {u'endpoints': [{u'publicURL': u'https://servers.api.rackspacecloud.com/v1.0/753591',
    u'tenantId': u'753591',
    u'versionId': u'1.0',
    u'versionInfo': u'https://servers.api.rackspacecloud.com/v1.0',
    u'versionList': u'https://servers.api.rackspacecloud.com/'}],
  u'name': u'cloudServers',
  u'type': u'compute'},
 {u'endpoints': [{u'publicURL': u'https://cdn1.clouddrive.com/v1/MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d',
    u'region': u'DFW',
    u'tenantId': u'MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d'},
   {u'publicURL': u'https://cdn2.clouddrive.com/v1/MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d',
    u'region': u'ORD',
    u'tenantId': u'MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d'}],
  u'name': u'cloudFilesCDN',
  u'type': u'rax:object-cdn'},
 {u'endpoints': [{u'publicURL': u'https://dfw.blockstorage.api.rackspacecloud.com/v1/753591',
    u'region': u'DFW',
    u'tenantId': u'753591'},
   {u'publicURL': u'https://ord.blockstorage.api.rackspacecloud.com/v1/753591',
    u'region': u'ORD',
    u'tenantId': u'753591'}],
  u'name': u'cloudBlockStorage',
  u'type': u'volume'},
 {u'endpoints': [{u'publicURL': u'https://dfw.databases.api.rackspacecloud.com/v1.0/753591',
    u'region': u'DFW',
    u'tenantId': u'753591'},
   {u'publicURL': u'https://ord.databases.api.rackspacecloud.com/v1.0/753591',
    u'region': u'ORD',
    u'tenantId': u'753591'}],
  u'name': u'cloudDatabases',
  u'type': u'rax:database'},
 {u'endpoints': [{u'publicURL': u'https://dfw.servers.api.rackspacecloud.com/v2/753591',
    u'region': u'DFW',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://dfw.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://dfw.servers.api.rackspacecloud.com/'},
   {u'publicURL': u'https://ord.servers.api.rackspacecloud.com/v2/753591',
    u'region': u'ORD',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://ord.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://ord.servers.api.rackspacecloud.com/'}],
  u'name': u'cloudServersOpenStack',
  u'type': u'compute'},
 {u'endpoints': [{u'internalURL': u'https://snet-storage101.dfw1.clouddrive.com/v1/MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d',
    u'publicURL': u'https://storage101.dfw1.clouddrive.com/v1/MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d',
    u'region': u'DFW',
    u'tenantId': u'MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d'},
   {u'internalURL': u'https://snet-storage101.ord1.clouddrive.com/v1/MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d',
    u'publicURL': u'https://storage101.ord1.clouddrive.com/v1/MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d',
    u'region': u'ORD',
    u'tenantId': u'MossoCloudFS_844909d3-431e-4919-8211-beca1e850a5d'}],
  u'name': u'cloudFiles',
  u'type': u'object-store'},
 {u'endpoints': [{u'publicURL': u'https://dns.api.rackspacecloud.com/v1.0/753591',
    u'tenantId': u'753591'}],
```

Simple!

The Service Catalog

- All of the available services and their endpoints.
- For services that are region-specific, holds the endpoints for each region.

Compute Service Catalog


```
{u'endpoints': [{u'publicURL': u'https://dfw.servers.api.
      rackspacecloud.com/v2/753591',
    u'region': u'DFW',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://dfw.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://dfw.servers.api.rackspacecloud.com/'},
   {u'publicURL': u'https://ord.servers.api.rackspacecloud.com/
      v2/753591',
    u'region': u'ORD',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://ord.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://ord.servers.api.rackspacecloud.com/'}],
  u'name': u'cloudServersOpenStack',
  u'type': u'compute'}
```

Compute Service Catalog


```
{u'endpoints': [{u'publicURL': u'https://dfw.servers.api.
      rackspacecloud.com/v2/753591',
    u'region': u'DFW',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://dfw.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://dfw.servers.api.rackspacecloud.com/'},
   {u'publicURL': u'https://ord.servers.api.rackspacecloud.com/
      v2/753591',
    u'region': u'ORD',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://ord.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://ord.servers.api.rackspacecloud.com/'}],
  u'name': u'cloudServersOpenStack',
  u'type': u'compute'}
```

Compute Service Catalog


```
{u'endpoints': [{u'publicURL': u'https://dfw.servers.api.
      rackspacecloud.com/v2/753591',
    u'region': u'DFW',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://dfw.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://dfw.servers.api.rackspacecloud.com/'},
   {u'publicURL': u'https://ord.servers.api.rackspacecloud.com/
      v2/753591',
    u'region': u'ORD',
    u'tenantId': u'753591',
    u'versionId': u'2',
    u'versionInfo': u'https://ord.servers.api.rackspacecloud.com/v2',
    u'versionList': u'https://ord.servers.api.rackspacecloud.com/'}],
u'name': u'cloudServersOpenStack',
u'type': u'compute'}
```

Find URI by Region

```
access = resp["access"]
svc_cat = access["serviceCatalog"]
compute_svc = svc_cat["compute"]
eps = compute_svc["endpoints"]
reg_ep = [ep for ep in eps
          if ep["region"] == region][0]
base_uri = reg_ep["public_url"]
list_uri = "%s/servers/detail" % base_uri
```

Create the Request

```
import urllib2
req = urllib2.Request(list_uri)
```

Add the Auth Headers

```
req.add_header("X-Auth-Token", token)
req.add_header("X-Auth-Project-Id",
               tenant_id)
```

Add the HTTP Headers

```
req.add_header("Accept", "application/json")
req.add_header("Content-Type",
        "application/json")
req.add_header("User-Agent", "myapp/1.1")
```

Make the Request

```
raw_resp = urllib2.urlopen(req)
resp = json.load(raw_resp)
print resp
```

```
{u'servers': [{u'OS-DCF:diskConfig': u'AUTO',
   u'OS-EXT-STS:power_state': 1,
   u'OS-EXT-STS:task_state': None,
   u'OS-EXT-STS:vm_state': u'active',
   u'accessIPv4': u'166.78.8.247',
   u'accessIPv6': u'2001:4800:7810:0512:8ca7:b42c:ff04:b4ae',
   u'addresses': {u'private': [{u'addr': u'10.181.18.164', u'version': 4}],
    u'public': [{u'addr': u'2001:4800:7810:0512:8ca7:b42c:ff04:b4ae',
      u'version': 6},
     {u'addr': u'166.78.8.247', u'version': 4}]},
   u'created': u'2013-01-24T15:13:23Z',
   u'flavor': {u'id': u'2',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/flavors/2',
      u'rel': u'bookmark'}]},
   u'hostId': u'fcaa2858a7255b1bbba8cf5e3a1550f0f1bea780164a4412709e5d4d',
   u'id': u'19b394eb-17ce-4221-9989-824be1bd2236',
   u'image': {u'id': u'09a4e770-26f7-4377-8893-2b21a0b76ba8',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/images/09a4e770-26f7-4377-8893-2b21a0b76ba8',
      u'rel': u'bookmark'}]},
   u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/v2/728829/servers/19b394eb-17ce-4221-9989-824be1bd2236',
     u'rel': u'self'},
    {u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/servers/19b394eb-17ce-4221-9989-824be1bd2236',
     u'rel': u'bookmark'}],
   u'metadata': {},
   u'name': u'ibacktest',
   u'progress': 100,
   u'rax-bandwidth:bandwidth': [],
   u'status': u'ACTIVE',
   u'tenant_id': u'728829',
   u'updated': u'2013-01-24T15:17:55Z',
   u'user_id': u'235799'},
  {u'OS-DCF:diskConfig': u'AUTO',
   u'OS-EXT-STS:power_state': 1,
   u'OS-EXT-STS:task_state': None,
   u'OS-EXT-STS:vm_state': u'active',
   u'accessIPv4': u'198.61.217.238',
   u'accessIPv6': u'2001:4800:780e:0510:8ca7:b42c:ff04:a78e',
   u'addresses': {u'private': [{u'addr': u'10.180.24.4', u'version': 4}],
    u'public': [{u'addr': u'2001:4800:780e:0510:8ca7:b42c:ff04:a78e',
      u'version': 6},
     {u'addr': u'198.61.217.238', u'version': 4}]},
   u'created': u'2012-12-03T17:23:16Z',
   u'flavor': {u'id': u'2',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/flavors/2',
      u'rel': u'bookmark'}]},
   u'hostId': u'ec8d624af62a9cb511daba3c106d27142b3cff0eaf3aa61cda89f1fa',
   u'id': u'5e16de51-724c-4885-8849-65719bb455fd',
   u'image': {u'id': u'5cebb13a-f783-4f8c-8058-c4182c724ccd',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/images/5cebb13a-f783-4f8c-8058-c4182c724ccd',
      u'rel': u'bookmark'}]},
   u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/v2/728829/servers/5e16de51-724c-4885-8849-65719bb455fd',
     u'rel': u'self'},
    {u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/servers/5e16de51-724c-4885-8849-65719bb455fd',
     u'rel': u'bookmark'}],
   u'metadata': {},
   u'name': u'BmwIqtGb',
   u'progress': 100,
   u'rax-bandwidth:bandwidth': [],
   u'status': u'ACTIVE',
   u'tenant_id': u'728829',
   u'updated': u'2013-01-19T16:47:03Z',
   u'user_id': u'235799'}]}
```

```
{u'servers': [{u'OS-DCF:diskConfig': u'AUTO',
   u'OS-EXT-STS:power_state': 1,
   u'OS-EXT-STS:task_state': None,
   u'OS-EXT-STS:vm_state': u'active',
   u'accessIPv4': u'166.78.8.247',
   u'accessIPv6': u'2001:4800:7810:0512:8ca7:b42c:ff04:b4ae',
   u'addresses': {u'private': [{u'addr': u'10.181.18.164', u'version': 4}],
    u'public': [{u'addr': u'2001:4800:7810:0512:8ca7:b42c:ff04:b4ae',
      u'version': 6},
     {u'addr': u'166.78.8.247', u'version': 4}]},
   u'created': u'2013-01-24T15:13:23Z',
   u'flavor': {u'id': u'2',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/flavors/2',
      u'rel': u'bookmark'}]},
   u'hostId': u'fcaa2858a7255b1bbba8cf5e3a1550f0f1bea780164a4412709e5d4d',
   u'id': u'19b394eb-17ce-4221-9989-824be1bd2236',
   u'image': {u'id': u'09a4e770-26f7-4377-8893-2b21a0b76ba8',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/images/
09a4e770-26f7-4377-8893-2b21a0b76ba8',
      u'rel': u'bookmark'}]},
   u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/v2/728829/servers/
19b394eb-17ce-4221-9989-824be1bd2236',
     u'rel': u'self'},
    {u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/servers/
19b394eb-17ce-4221-9989-824be1bd2236',
     u'rel': u'bookmark'}],
   u'metadata': {},
   u'name': u'ibacktest',
   u'progress': 100,
   u'rax-bandwidth:bandwidth': [],
   u'status': u'ACTIVE',
   u'tenant_id': u'728829',
   u'updated': u'2013-01-24T15:17:55Z',
   u'user_id': u'235799',
   ...
}]}
```

Fun, huh?

Just One Server:


```
{u'OS-DCF:diskConfig': u'AUTO',
   u'OS-EXT-STS:power_state': 1,
   u'OS-EXT-STS:task_state': None,
   u'OS-EXT-STS:vm_state': u'active',
   u'accessIPv4': u'166.78.8.247',
   u'accessIPv6': u'2001:4800:7810:0512:8ca7:b42c:ff04:b4ae',
   u'addresses': {u'private': [{u'addr': u'10.181.18.164', u'version': 4}],
    u'public': [{u'addr': u'2001:4800:7810:0512:8ca7:b42c:ff04:b4ae',
      u'version': 6},
     {u'addr': u'166.78.8.247', u'version': 4}]},
   u'created': u'2013-01-24T15:13:23Z',
   u'flavor': {u'id': u'2',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/flavors/2',
      u'rel': u'bookmark'}]},
   u'hostId': u'fcaa2858a7255b1bbba8cf5e3a1550f0f1bea780164a4412709e5d4d',
   u'id': u'19b394eb-17ce-4221-9989-824be1bd2236',
   u'image': {u'id': u'09a4e770-26f7-4377-8893-2b21a0b76ba8',
    u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/images/
09a4e770-26f7-4377-8893-2b21a0b76ba8',
      u'rel': u'bookmark'}]},
   u'links': [{u'href': u'https://dfw.servers.api.rackspacecloud.com/v2/728829/servers/
19b394eb-17ce-4221-9989-824be1bd2236',
     u'rel': u'self'},
    {u'href': u'https://dfw.servers.api.rackspacecloud.com/728829/servers/
19b394eb-17ce-4221-9989-824be1bd2236',
     u'rel': u'bookmark'}],
   u'metadata': {},
   u'name': u'NM0978FR',
   u'progress': 100,
   u'rax-bandwidth:bandwidth': [],
   u'status': u'ACTIVE',
   u'tenant_id': u'728829',
   u'updated': u'2013-01-24T15:17:55Z',
   u'user_id': u'235799'}
```

Wouldn't Python objects be better to work with?

Each call requires:

- Retrieve authentication credentials from cache
    - Re-auth when no token, or when token expires
- Determine endpoint for selected region from service catalog
- Form URI from endpoint and API spec
- Add Headers:
    - Auth headers: "X-Auth-Token" & "X-Auth-Project-Id"
    - "Content-Type: application/json"
    - "Accept: application/json"
    - "User-agent: xxx"
- Transmit request
- Handle all errors
- Retry if possible

...and then

- Parse results
  - Headers
  - JSON body
- Extract list of resource data
- Convert to native objects
- Return entity (or list of entities) with appropriate info

SDKs to the Rescue!

pyrax

The Python SDK for OpenStack-based Clouds

pyrax

Open source

https://github.com/rackspace/pyrax

pyrax

Supported by Rackspace

So let's see some code!

Authentication

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import pyrax
import pyrax.utils as utils
import pyrax.exceptions as exc

# If you have keyring configured...
pyrax.keyring_auth()
# If you have a credential file...
pyrax.set_credential_file("/path/to/file")
# Or just set directly
pyrax.set_credentials(username, password)
```

Credential File

For Keystone authentication:

```
[keystone]
username = myusername
password = 1234567890abcdef
tenant_id = abcdef1234567890
```

For Rackspace authentication:

```
[rackspace_cloud]
username = myusername
api_key = 1234567890abcdef
```

Configuration File

```
[default]
identity_type = rackspace
keyring_username = leaferax
region = SYD
custom_user_agent =
debug = False
verify_ssl = False

[devstack]
identity_type = keystone
region = RegionOne
custom_user_agent =
debug = True
auth_endpoint = http://166.78.147.147:5000/v2.0/
tenant_name = demo
tenant_id = d774d0b56b60464c80e3430062aae65f
keyring_username = demo
```

Shortcuts

```
cs = pyrax.cloudservers
cf = pyrax.cloudfiles
clb = pyrax.cloud_loadbalancers
cdb = pyrax.cloud_databases
cnw = pyrax.cloud_networks
dns = pyrax.cloud_dns
cmn = pyrax.cloud_monitoring
```

Create the Database

```
# NOTE: The flavor ID and size are hardcoded
# for demo purposes.
db_instance = cdb.create("DemoDB", flavor=1, volume=2)
utils.wait_for_build(db_instance, verbose=True)
db = db_instance.create_database("demodb")
db_user = db_instance.create_user("demouser",
        "topsecret", db)
```

Create the Database

```
# NOTE: The flavor ID and size are hardcoded
# for demo purposes.
db_instance = cdb.create("DemoDB", flavor=1, volume=2)
utils.wait_for_build(db_instance, verbose=True)
db = db_instance.create_database("demodb")
db_user = db_instance.create_user("demouser",
        "topsecret", db)
```

Create the Database

```
# NOTE: The flavor ID and size are hardcoded
# for demo purposes.
db_instance = cdb.create("DemoDB", flavor=1, volume=2)
utils.wait_for_build(db_instance, verbose=True)
db = db_instance.create_database("demodb")
db_user = db_instance.create_user("demouser",
        "topsecret", db)
```

Create the Database

```
# NOTE: The flavor ID and size are hardcoded
# for demo purposes.
db_instance = cdb.create("DemoDB", flavor=1, volume=2)
utils.wait_for_build(db_instance, verbose=True)
db = db_instance.create_database("demodb")
db_user = db_instance.create_user("demouser",
        "topsecret", db)
```

Create an isolated network

```
new_network_name = "isolated"
new_network_cidr = "192.168.0.0/24"
new_net = cnw.create(new_network_name,
        cidr=new_network_cidr)
```

Create an isolated network

```
new_network_name = "isolated"
new_network_cidr = "192.168.0.0/24"
new_net = cnw.create(new_network_name,
        cidr=new_network_cidr)
```

Create an isolated network

```
new_network_name = "isolated"
new_network_cidr = "192.168.0.0/24"
new_net = cnw.create(new_network_name,
        cidr=new_network_cidr)
```

Define a Public Key

```
# Store the public key
keyfile = os.expanduser("~/.ssh/id_rsa.pub")
with open(keyfile, "r") as kf:
    pub = kf.read()
    key = cs.keypairs.create("macbook", pub)
```

Create the App Servers

```
# This is hardcoded for demo purposes.
app_image = "fb61c42c-b65c-45f0-b1b8-20bd5e47de32"
# Create two servers with only ServiceNet
networks = [{"net-id": cnw.SERVICE_NET_ID}]
server1 = cs.servers.create("PyTexas_Srv1",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
server2 = cs.servers.create("PyTexas_Srv2",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
utils.wait_for_build(server1, verbose=True)
utils.wait_for_build(server2, verbose=True)
```

Create the App Servers

```
# This is hardcoded for demo purposes.
app_image = "fb61c42c-b65c-45f0-b1b8-20bd5e47de32"
# Create two servers with only ServiceNet
networks = [{"net-id": cnw.SERVICE_NET_ID}]
server1 = cs.servers.create("PyTexas_Srv1",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
server2 = cs.servers.create("PyTexas_Srv2",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
utils.wait_for_build(server1, verbose=True)
utils.wait_for_build(server2, verbose=True)
```

Create the App Servers

```
# This is hardcoded for demo purposes.
app_image = "fb61c42c-b65c-45f0-b1b8-20bd5e47de32"
# Create two servers with only ServiceNet
networks = [{"net-id": cnw.SERVICE_NET_ID}]
server1 = cs.servers.create("PyTexas_Srv1",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
server2 = cs.servers.create("PyTexas_Srv2",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
utils.wait_for_build(server1, verbose=True)
utils.wait_for_build(server2, verbose=True)
```

Create the App Servers

```
# This is hardcoded for demo purposes.
app_image = "fb61c42c-b65c-45f0-b1b8-20bd5e47de32"
# Create two servers with only ServiceNet
networks = [{"net-id": cnw.SERVICE_NET_ID}]
server1 = cs.servers.create("PyTexas_Srv1",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
server2 = cs.servers.create("PyTexas_Srv2",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
utils.wait_for_build(server1, verbose=True)
utils.wait_for_build(server2, verbose=True)
```

Create the App Servers

```
# This is hardcoded for demo purposes.
app_image = "fb61c42c-b65c-45f0-b1b8-20bd5e47de32"
# Create two servers with only ServiceNet
networks = [{"net-id": cnw.SERVICE_NET_ID}]
server1 = cs.servers.create("PyTexas_Srv1",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
server2 = cs.servers.create("PyTexas_Srv2",
        image=app_image, flavor=2, key_name="macbook",
        nics=networks)
utils.wait_for_build(server1, verbose=True)
utils.wait_for_build(server2, verbose=True)
```

Define the LB Nodes & Virtual IP

```
# Get the server IPs
ip1 = server1.addresses["private"][0]["addr"]
ip2 = server2.addresses["private"][0]["addr"]
# Define the nodes
node1 = clb.Node(address=ip1, port=80,
        weight=1, condition="ENABLED")
node2 = clb.Node(address=ip2, port=80,
        weight=1, condition="ENABLED")
# Create the Virtual IP
vip = clb.VirtualIP(type="PUBLIC")
```

Define the LB Nodes & Virtual IP

```
# Get the server IPs
ip1 = server1.addresses["private"][0]["addr"]
ip2 = server2.addresses["private"][0]["addr"]
# Define the nodes
node1 = clb.Node(address=ip1, port=80,
        weight=1, condition="ENABLED")
node2 = clb.Node(address=ip2, port=80,
        weight=1, condition="ENABLED")
# Create the Virtual IP
vip = clb.VirtualIP(type="PUBLIC")
```

Define the LB Nodes & Virtual IP

```
# Get the server IPs
ip1 = server1.addresses["private"][0]["addr"]
ip2 = server2.addresses["private"][0]["addr"]
# Define the nodes
node1 = clb.Node(address=ip1, port=80,
        weight=1, condition="ENABLED")
node2 = clb.Node(address=ip2, port=80,
        weight=1, condition="ENABLED")
# Create the Virtual IP
vip = clb.VirtualIP(type="PUBLIC")
```

Create the Load Balancer

```
# Create the Load Balancer
lb = clb.create("PyTexas_LB", port=80, protocol="HTTP",
        nodes=[node1, node2], virtual_ips=vip,
        algorithm="WEIGHTED_ROUND_ROBIN")
utils.wait_for_build(lb, verbose=True)
lb_ip = lb.virtual_ips[0].address
```

Create the Load Balancer

```
# Create the Load Balancer
lb = clb.create("PyTexas_LB", port=80, protocol="HTTP",
        nodes=[node1, node2], virtual_ips=vip,
        algorithm="WEIGHTED_ROUND_ROBIN")
utils.wait_for_build(lb, verbose=True)
lb_ip = lb.virtual_ips[0].address
```

Create the Load Balancer

```
# Create the Load Balancer
lb = clb.create("PyTexas_LB", port=80, protocol="HTTP",
        nodes=[node1, node2], virtual_ips=vip,
        algorithm="WEIGHTED_ROUND_ROBIN")
utils.wait_for_build(lb, verbose=True)
lb_ip = lb.virtual_ips[0].address
```

Configure DNS

```
domain_name = "pyraxdemo.com"
dom = dns.create(name=domain_name,
        emailAddress="ed.leafe@rackspace.com",
        ttl=900, comment="PyTexas Demo")
a_rec = {"type": "A",
        "name": domain_name,
        "data": lb_ip,
        "ttl": 900}
recs = dom.add_record(a_rec)
```

Configure DNS

```
domain_name = "pyraxdemo.com"
dom = dns.create(name=domain_name,
        emailAddress="ed.leafe@rackspace.com",
        ttl=900, comment="PyTexas Demo")
a_rec = {"type": "A",
        "name": domain_name,
        "data": lb_ip,
        "ttl": 900}
recs = dom.add_record(a_rec)
```

Configure DNS

```
domain_name = "pyraxdemo.com"
dom = dns.create(name=domain_name,
        emailAddress="ed.leafe@rackspace.com",
        ttl=900, comment="PyTexas Demo")
a_rec = {"type": "A",
        "name": domain_name,
        "data": lb_ip,
        "ttl": 900}
recs = dom.add_record(a_rec)
```

That's it!

Adding Another Node

```
server3 = cs.servers.create("PyTexas_Srv3",
        image=image_id, flavor=2, nics=networks)
utils.waitfor_build(server3, verbose=True)
ip3 = server3.addresses["private"][0]["addr"]
node3 = clb.Node(address=ip3, port=80,
        weight=1, condition="ENABLED")
lb.add_nodes(node3)
```

Adding Another Node

```
server3 = cs.servers.create("PyTexas_Srv3",
        image=image_id, flavor=2, nics=networks)
utils.waitfor_build(server3, verbose=True)
ip3 = server3.addresses["private"][0]["addr"]
node3 = clb.Node(address=ip3, port=80,
        weight=1, condition="ENABLED")
lb.add_nodes(node3)
```

Adding Another Node

```
server3 = cs.servers.create("PyTexas_Srv3",
        image=image_id, flavor=2, nics=networks)
utils.waitfor_build(server3, verbose=True)
ip3 = server3.addresses["private"][0]["addr"]
node3 = clb.Node(address=ip3, port=80,
        weight=3, condition="ENABLED")
lb.add_nodes(node3)
```

Adding Another Node

```
server3 = cs.servers.create("PyTexas_Srv3",
        image=image_id, flavor=2, nics=networks)
utils.waitfor_build(server3, verbose=True)
ip3 = server3.addresses["private"][0]["addr"]
node3 = clb.Node(address=ip3, port=80,
        weight=1, condition="ENABLED")
lb.add_nodes(node3)
```

Session Materials

https://github.com/EdLeafe/pytexas2013

Q & A

ed.leafe@rackspace.com

ed@openstack.org

Google+: +EdLeafe

Twitter: @EdLeafe