Using Service Discovery to build dynamic python applications

EuroPython 2016
Service discovery
query a distributed catalog for a given service
Service discovery **register** & **query**

- I'm looking for api_x!
  - api_x @ host1:port & host2:port
  - api_x @ host2:port
  - api_y @ host2:port
  - mysql_x @ host8:port
  - memcache @ host4:port & host6:port
  - memcache @ host4:port
  - memcache @ host6:port
  - mongodb_x @ host3:port
  - mongodb_y @ host4:port

- I'm looking for a memcache server!
  - memcache @ host4:port & host6:port

- I'm looking for the mysql_x instance!
  - mysql_x @ host8:port

- I'm hosting api_x
- I'm hosting api_x
- I'm hosting api_x
- I'm a memcache server
- I'm a memcache server
- I'm hosting mysql_x
ZooKeeper (Apache)

Facts
- since 2007
- Java
- ZAB “consensus”

Features
- Mature
- Features
- Hadoop
- Service discovery
- Maintenance
- Python C binding
- Not datacenter aware
etcd (CoreOS)

Facts
- since 2013
- Go
- Raft consensus

Adoption
- Fast
- Simple
- HTTP API

- Health checking
- Service discovery
- Not datacenter aware
Consul (HashiCorp)

Facts
- since 2013
- Go
- Raft consensus

- Health checking
- Service discovery
- Datacenter aware
- Web UI
- HTTP API
- DNS API
Notes on Zookeeper & etcd

Service discovery = abusing the K/V store!

K/V store is like a filesystem where you can store data.

- /
  - api_x
    - providers
      - host1:port
      - host2:port
  - memcache
    - providers
      - host4:port
      - host6:port
## Python client libraries

<table>
<thead>
<tr>
<th>Zookeeper</th>
<th>kazoo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>zc.zk</td>
</tr>
<tr>
<td></td>
<td>pookeeper</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>etcd</th>
<th>python-etcd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aioetcd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Consul</th>
<th>python-consul</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>consulate</td>
</tr>
</tbody>
</table>
Python clients reliability

can you trust your engine?
zc.zk client reliability (kazoo)

```python
>>> import zc.zk

>>> zk = zc.zk.ZooKeeper('yazd:2181,dunno:2181', wait=False)

>>> zk.state
'CONNECTED'

# zookeeper server up

>>> zk.properties('/ep2016/color')
zc.zk.Properties(/ep2016/color)

# zookeeper server down

>>> zk.properties('/ep2016/color')
SessionExpiredError:
```

- Multiple hosts
- Autoreconnect
- Connection state
- Rich exceptions
- Don’t fail on connect
etcd client reliability

```python
>>> import etcd

>>> etc = etcd.Client(host='localhost', port=4001, allow_reconnect=True)

# etcd server up
>>> etc.leader
{u'clientURLs': [u'http://localhost:2379', u'http://localhost:4001'], ...

# etcd server down
>>> etc.leader
EtcdException: Cannot get leader data: Connection to etcd failed due to MaxRetryError ("HTTPConnectionPool(host='127.0.0.1', port=4001): Max retries exceeded with url: /v2/stats/self (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f9beaac3cd0>: Failed to establish a new connection: [Errno 111] Connection refused',),")",)

- Multiple hosts
- Autoreconnect
- Connection state
- Rich exceptions
- Don’t fail on connect
```
>> import consul

>>> cons = consul.Consul(host='localhost', port=8500)

# consul server up
>>> cons.status.leader()
'u'172.17.15.2:8300'

# consul server down
>>> cons.status.leader()
ConnectionError: HTTPConnectionPool(host='127.0.0.1', port=8500): Max retries exceeded with url: /v1/status/leader (Caused by NewConnectionError('<requests.packages.urllib3.connection.HTTPConnection object at 0x7f9beaafa290>: Failed to establish a new connection: [Errno 111] Connection refused',))

- Multiple hosts
- Autoreconnect
- Connection state
- Rich exceptions
- Don’t fail on connect
Service registration

health checking and deregistration!
def register(client):
    while True:
        if client.state == 'CONNECTED':
            try:
                client.create('/ep2016/providers', ephemeral=False, makepath=True)
            except NodeExistsError:
                pass
            try:
                client.register('/ep2016/providers', ('yazd', 5000))
                break
            except NodeExistsError:
                print('waiting for registration...')
                sleep(0.5)
        else:
            print('zookeeper host is down, reconnecting...')
            sleep(0.5)

>>> zk = zc.zk.ZooKeeper('yazd:2181,dunno:2181', session_timeout=5, wait=True)
>>> register(zk)
def register(client):
    while True:
        if client.state == 'CONNECTED':
            try:
                client.create('/ep2016/providers', ephemeral=False, makepath=True)
            except NodeExistsError:
                pass
            try:
                client.register('/ep2016/providers', ('yazd', 5000))  # implicit ephemeral znode
                break
            except NodeExistsError:
                print('waiting for registration...')
                sleep(0.5)
        else:
            print('zookeeper host is down, reconnecting...')
            sleep(0.5)
# session_timeout = failure detection latency
>>> zk = zc.zk.ZooKeeper('yazd:2181,dunno:2181', session_timeout=5, wait=True)
>>> register(zk)
etcd service registration

```python
def register(client):
    while True:
        try:
            client.read('/ep2016/providers')
        except (etcd.EtcdKeyNotFound, KeyError):
            client.write('/ep2016/providers', None, dir=True)
        except etcd.EtcdException:
            print('etcd host is down, reconnecting...')
            continue
        try:
            client.write('/ep2016/providers/yazd:5000', 'yazd:5000', dir=False, ttl=5)
        except etcd.EtcdAlreadyExist:
            pass
        except etcd.EtcdException:
            print('etcd host is down, reconnecting...')
        return

>>> etc = etcd.Client(host='localhost', port=4001, allow_reconnect=True)
>>> register(etc)
```
etcd health checking

```python
class HealthPinger(threading.Thread):
    stop = False

def __init__(self):
    threading.Thread.__init__(self)
    self.client = etcd.Client(host='localhost', port=4001, allow_reconnect=True)

def run(self):
    while HealthPinger.stop is False:  # infinite loop registration before TTL expires
        self.register()
        sleep(TTL - 1)

def register(self):
    try:
        self.client.read('/ep2016/providers')
    except (etcd.EtcdKeyNotFound, KeyError):
        self.client.write('/ep2016/providers', None, dir=True)
    except etcd.EtcdException:
        print('etcd host is down, reconnecting...')
        return

    try:
        self.client.write('/ep2016/providers/yazd:5000', 'yazd:5000', dir=False, ttl=5)  # ttl = failure detection latency
    except etcd.EtcdAlreadyExist:
        pass
    except etcd.EtcdException:
        print('etcd host is down, reconnecting...')
        return

>>> register_thread = HealthPinger().start()
```
def register(client):
    while True:
        try:
            client.agent.service.register('ep2016', address='yazd', port=5002)
            break  # integrated service registration <3
        except (ConnectionError, consul.ConsulException):
            print('consul host is down, reconnecting...')
            sleep(0.5)

>>> cons = consul.Consul(host='localhost', port=8500)
>>> register(cons)
def register(client):
    # create a HTTP health check for our web service which the consul server will run every 2 seconds

    while True:
        try:
            client.agent.service.register('ep2016', address='yazd', port=5002, check=check_http)
            break
        except (ConnectionError, consul.ConsulException):
            print('consul host is down, reconnecting...')

        sleep(0.5)

>>> cons = consul.Consul(host='localhost', port=8500)
>>> register(cons)
Discover all the things!
Querying the catalog for a service

# Zookeeper
addresses = zk.children('/ep2016/providers')
for address in sorted(addresses):
    host, port = address.split(':')

# etcd
children = etc.read('/ep2016/providers', recursive=True).children
for child in children:
    host, port = child.value.split(':')

# Consul
index, services = cons.health.service('ep2016', passing=True)
for service_info in services:
    service = service_info['Service']
Live demo!

(hopefully)
Thanks

source code: github.com/ultrabug/ep2016
@ultrabug