KeystoneClient ライブラリ

日本 OpenStack ユーザ会

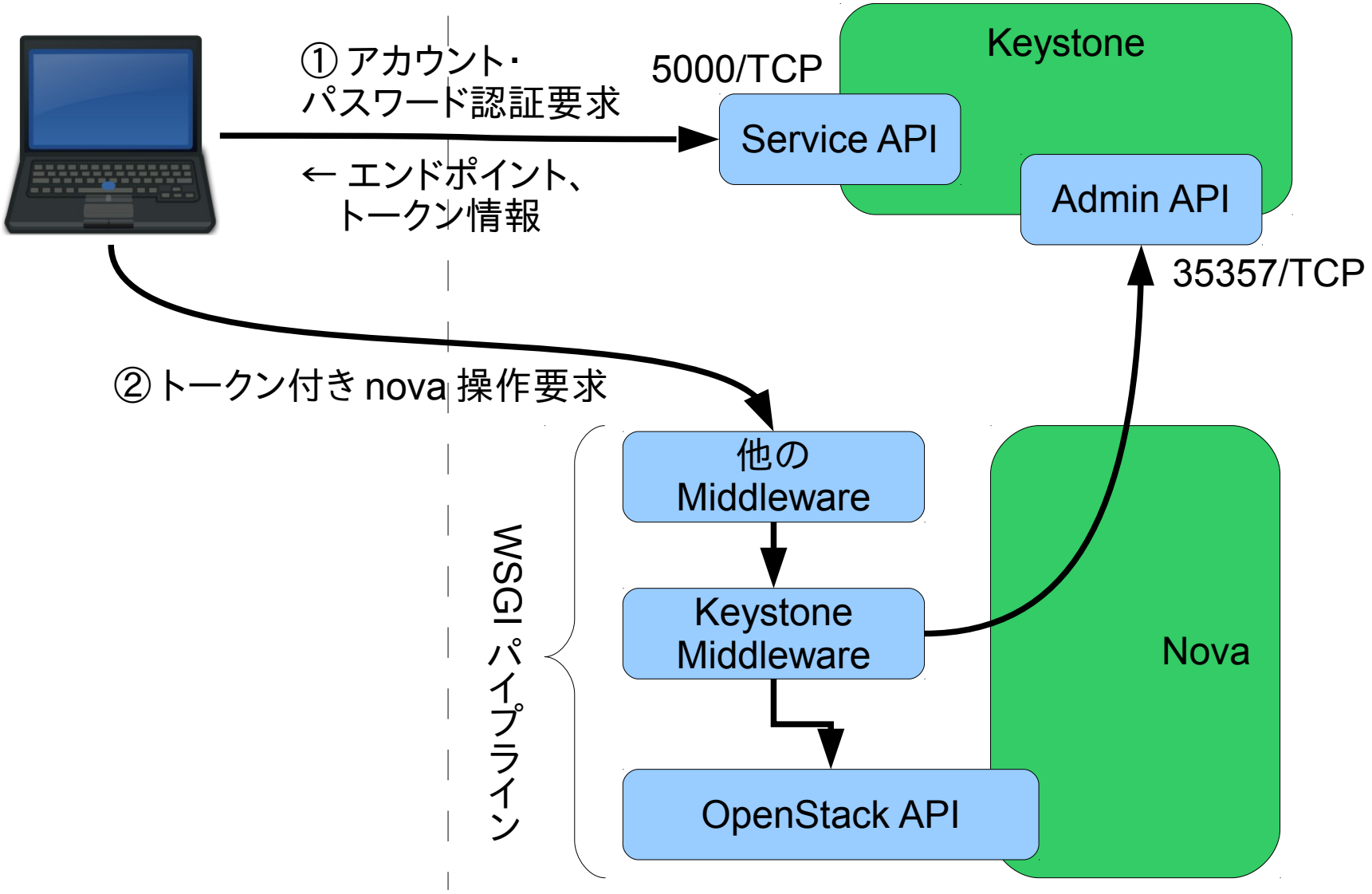
吉山 晃 <akirayoshiyama@gmail.com>

OpenStack のクライアントライブラリ

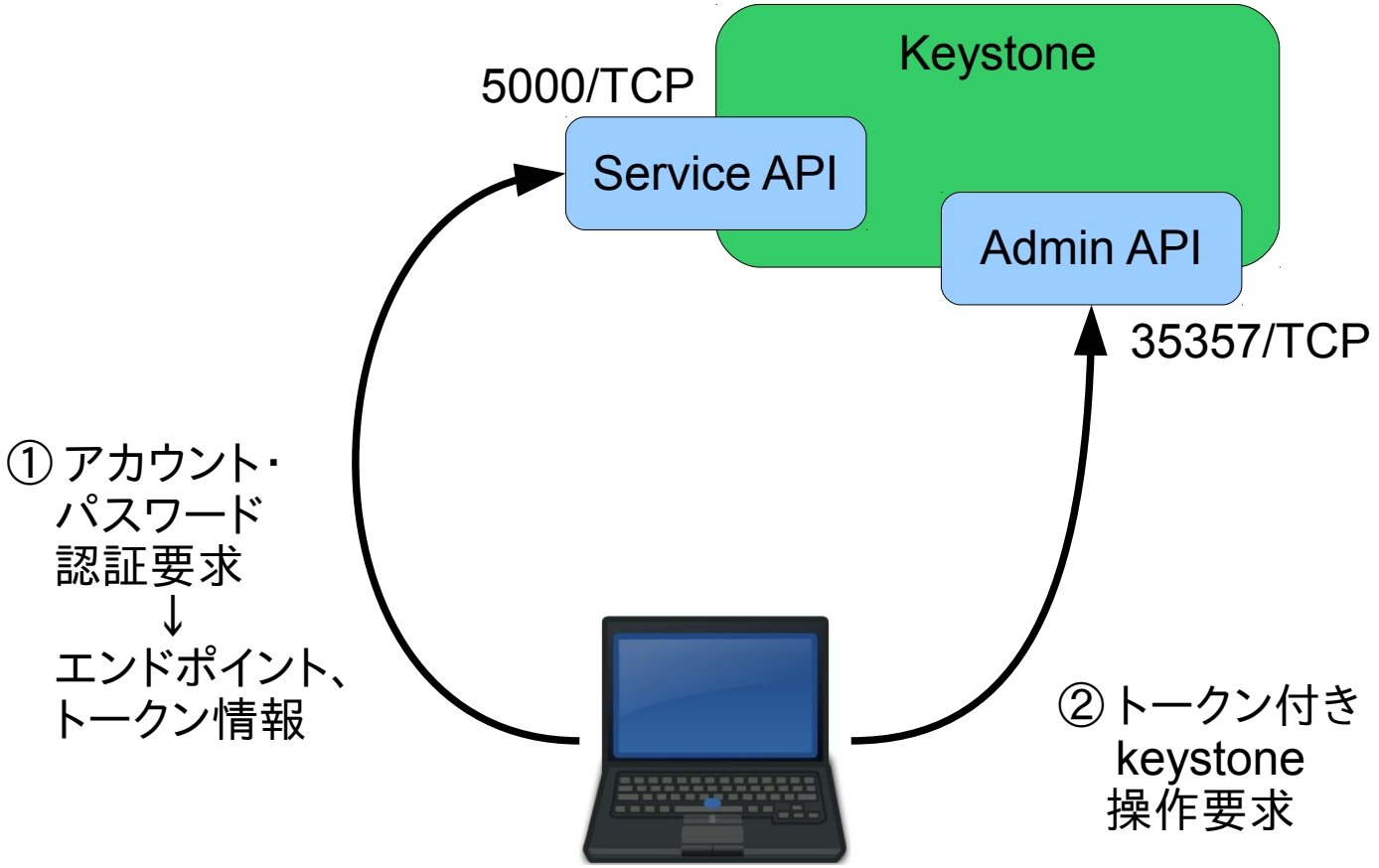
- python-novaclient 2012.1
- python-keystoneclient 2012.1
- python-quantumclient 2012.1
- python-swiftclient（開発中）
- python-glanceclient（開発中）

今日は python-keystoneclient
（以後 keystoneclient ）を紹介します。

Keystone 認証（nova の場合）



Keystone 自体の認証



keystoneclient の使い方①

1.keystoneclient をインポート

```
from keystoneclient.v2_0 import client
```

2.コントロール用インスタンスの作成

- Service API からの作成

```
keystone = client.Client(
    username=USER,  password=PASS,
    tenant_name=TENANT_NAME,
    tenant_name=KEYSTONE_URL)
```

- Admin API からの作成

```
keystone = client.Client(
    endpoint=ENDPOINT, token=TOKEN)
```

keystoneclient の使い方②

操作対象	コード例
テナント	`keystone.tenants.list()`
ユーザ	`keystone.users.list()`
ロール	`keystone.roles.list()`
エンドポイント	`keystone.endpoints.list()`
トークン	`keystone.tokens.list()`

keystoneclient の使い方③

操作種別	コード例
作成	keystone.tenants.**create**()
一覧	keystone.tenants.**list**()
詳細取得	keystone.tenants.**get**()
更新	keystone.tenants.**update**()
削除	keystone.tenants.**delete**()

サンプルプログラム

YAML 形式のデータを読み込み、

- ロール
- テナント
- ユーザ
- サービス
- エンドポイント

を自動的に登録する

サンプルデータ (YAML)

```
roles:
  - name: admin
  - name: memberRole
tenants:
  - name: service
    desc: Service Tenant
users:
  - name: nova
    tenant_roles:
      - tenant: service
        role: admin
services:
  - name: nova
    type: compute
    desc: Nova Compute Service
    urls:
      - region: RegionOne
        public:   http://localhost:8774/v2/$(tenant_id)s
        internal: http://localhost:8774/v2/$(tenant_id)s
        admin:    http://localhost:8774/v2/$(tenant_id)s
```

サンプルプログラム

```
# keystoneclient ライブラリをインポート
from keystoneclient.v2_0 import client

# YAML ライブラリをインポート
import yaml

#  前スライドの YAML データを Python  の辞書型データとしてロード
fixture = yaml.load(open("fixture.yaml").read())

# keystoneclient インスタンスを作成
keystone = client.Client(
    endpoint=endpoint, token=token)
```

サンプルプログラム(続き)

```
#  ロールデータのインポート
roles = {}
for _role in fixtures.get('roles'):
      name = _role['name']
      # create()  の戻り値(辞書)を roles[]  に保存
      roles[name] = keystone.roles.create(name)

#  テナントデータのインポート
tenants = {}
for _tenant in fixtures.get('tenants'):
      name = _tenant['name']
      # create()  の戻り値(辞書)を tenants[]  に保存
      tenants[name] = keystone.tenants.create(
            name,
            description=_tenant.get('desc'))
```

サンプルプログラム(続き)

```
# ユーザデータのインポート
users = {}
for _user in fixtures.get('users'):
    name = _user['name']
    primary_tenant = tenants.get(
        _user['tenant_roles'][0]['tenant'])
    # create() の戻り値(辞書)を tenants[] に保存
    users[name] = keystone.users.create(
        name,
        _user.get('pass'),
        _user.get('email'),
        tenant_id=primary_tenant.id)
    # ユーザのテナント毎のロールを登録
    for tenant_role in _user['tenant_roles']:
        role_name = tenant_role['role']
        tenant_name = tenant_role['tenant']
        keystone.roles.add_user_role(
            users[name].id,
            roles[role_name].id,
            tenants[tenant_name].id)
```

サンプルプログラム（続き）

```
# サービスデータのインポート
services = {}
for service in fixtures.get('services'):
    name = service['name']
    services[name] = keystone.services.create(
                                name,
                                service['type'],
                                service.get('desc', ""))
    for url in service['urls']:
        keystone.endpoints.create(
            url['region'],
            services[name].id,
            url['public'],
            url['admin'],
            url['internal'])
```

まとめ

- keystoneclient は便利
  - API は整理されており使いやすい
  - < クライアントインスタンス >.< クラス >.create() の戻り値に登録されたエントリの UUID が含まれる
    →サンプルプログラムを応用すれば、
    　各種データを自動的に登録可能
  - Keystone の RESTful API に変更があっても keystoneclient が変更を吸収してくれる（はず）


おぷ☆すた
Open ★ Stack