CHINA OpenStack Days

IT大咖说 知识分享平台

CHINA OpenStack Days

EasyStack open cloud computing

OpenStack 基础公共库的现在与未来

郭长波

EasyStack 社区总监

个人简介

EasyStack 创始工程师，开源社区负责人 .

OpenStack 公共基础库项目 Oslo PTL.

2017 年当选为 OpenStack 基金会个人独立董事 .

社区活跃技术贡献者 ,2012 年至今提交近 1000 commits 和 45000+ LOCs，

中国 OpenStack 用户组组织者和北京 OpenStack meetup 组织者

多次参加国内 meetup 和 OpenStack 峰会并进行演讲，

巴塞罗那 , 波士顿和悉尼峰会 Upstream Developement Track Chair

2017 OpenStack Days China 主要组织者之一

内容大纲

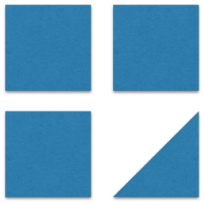
- 基础库 oslo 概览
- oslo 关键组件分析
- 近期版本功能更新

oslo 概览 – 为什么需要公共基础库

OpenStack 有多少项目?

59 个项目 988 个代码仓库

相同功能 –>需要基础库



- BARBICAN
- CHEF OPENSTACK
- CINDER
- CLOUDKITTY
- COMMUNITY APP CATALOG
- CONGRESS
- DESIGNATE
- DOCUMENTATION
- DRAGONFLOW
- EC2-API
- FREEEZER
- GLANCE
- HEAT
- NOVA
- OPENSTACK CHARMS
- OPENSTACK-ANSIBLE
- OSLO
- PACKAGING-DEB
- PACKAGING-RPM
- PUPPET OPENSTACK
- QUALITY ASSURANCE
- RALLY
- REFSTACK
- RELEASE MANAGEMENT
- REQUIREMENTS
- SAHARA

oslo 概览 – 历史

官方名称

OpenStack Common Libraries

项目使命

To produce a set of python libraries containing code shared by OpenStack projects. The APIs provided by these libraries should be high quality , stable, consistent, documented and generally applicable.

项目历史

2011-07 oslo-incubator	2013-04 oslo.messaging	2016-06 Farewell oslo-incubator
2012-01 oslo.config	new separated libraries	.....
2012 -12 oslo.db	.....	2017-05 castellan

oslo 概览 – 应用分类

OpenStack 开发者：

是理解各个项目底层实现的关键，开发中经常用到

如数据库访问，消息队列，服务方式运行，常用简单方法等

OpenStack 运维人员：

理解常用的公共库原理，更好的运维云平台

配置项管理，权限访问，请求时间收集，运行时状态查看等

Python 开发者：

学习常用标准库的典型用法 ,Restful API, 数据库等常见功能实现，

应用到非 OpenStack 开发，快速搭建分布式 Python 程序

oslo 概览 – 项目浏览

1 automaton
2 castellan
3 cookiecutter
4 debtcollector
5 futurist
6 mox3
7oslo-cookiecutter
8 oslo.cache
9 oslo.concurrency
10 oslo.config
11 oslo.context
12 oslo.db
13 oslo.i18n
14 oslo.log
15 oslo.messaging
16 oslo.middleware
17 oslo.policy
18 oslo.privsep
19 oslo.reports
20 oslo.rootwrap
21 oslo.serialization
22 oslo.service
23 oslo.tools
24 oslo.utils
25 oslo.versionedobjects
26 oslo.version
27 oslo.vmware
28 oslosphinx
29 oslotest
30 osprofiler
31 pbr
32 pylockfile
33 stevedore
34 taskflow
35 tooz

https://wiki.openstack.org/wiki/Oslo

内容大纲

- 基础库 oslo 概览
- **oslo 关键组件分析**
- 近期版本功能更新

oslo.config

用途

从命令行或文件分析配置项，支持定义多种配置项类型，自带类型校验功能，
支持运行时改变配置项值方便测试，支持从源代码生成包含所有配置项的样例文件

常用类型

StrOpt , BoolOpt , IntOpt , PortOpt ,ListOpt , DictOpt , IPOpt , HostnameOpt

－引用已有的配置项值： $name${group.name}
－约束条件： choices , required , secret , mutable

```python
from oslo_config import cfg

opts = [
    cfg.StrOpt('bind_host', default='0.0.0.0'),
    cfg.PortOpt('bind_port', default=9292),
]

CONF = cfg.CONF
CONF.register_opts(opts)

def start(server, app):
    server.start(app, CONF.bind_port, CONF.bind_host)
```

```
1 [DEFAULT]
2 bing_host= 127.0.0.1
3 bind_port=9999
```

oslo.db(1/2)

用途

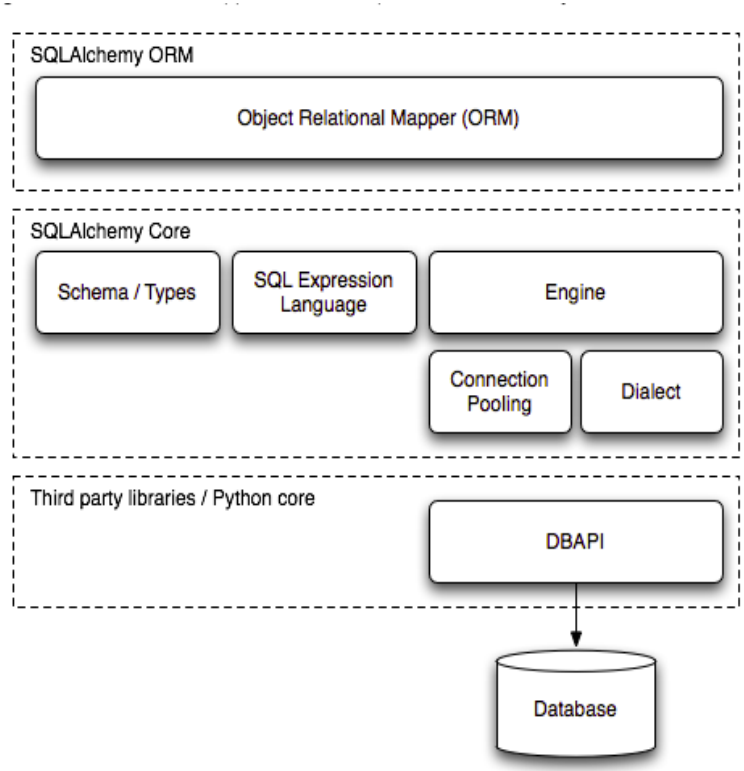
提供连接操作数据库接口，及各种易用方法，如创建更新表结构，创建查询删除记录；
深度使用 Python 数据库模块 SQLAlchemy，支持多种数据库类型

特性

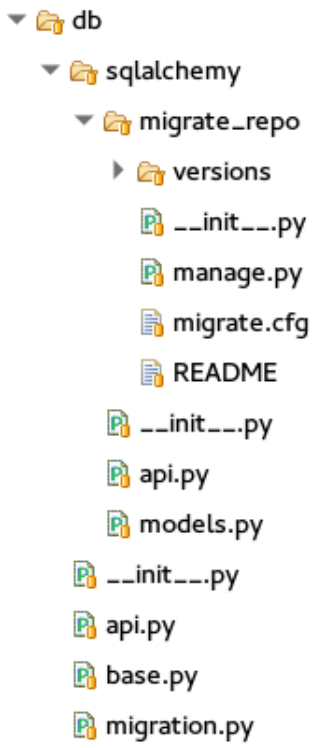
支持 MySQL, PostgreSQL, Sqlite, DB2
支持 多种 MySQL DBAPI driver

PyMySQL vs MySQL-python

mysql+pymysql://



oslo.db(2/2)




```
[database]

#
# From oslo.db
#

# DEPRECATED: The file name to use with SQLite. (string value)
# Deprecated group/name - [DEFAULT]/sqlite_db
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: Should use config option connection or slave_connection to connect the
# database.
#sqlite_db = oslo.sqlite

# If True, SQLite uses synchronous mode. (boolean value)
# Deprecated group/name - [DEFAULT]/sqlite_synchronous
#sqlite_synchronous = true

# The back end to use for the database. (string value)
# Deprecated group/name - [DEFAULT]/db_backend
#backend = sqlalchemy

# The SQLAlchemy connection string to use to connect to the database. (string
# value)
# Deprecated group/name - [DEFAULT]/sql_connection
# Deprecated group/name - [DATABASE]/sql_connection
# Deprecated group/name - [sql]/connection
#connection = <None>
```

oslo.messaging(1/2)

用途

希望定义用户友好易于理解的接口，为组件提供 RPC 和 notification 功能，
支持多种消息传输机制，通过配置项选择不同 backend.

支持 driver

包括 amqp, fake, Kafka, kombu, pika, rabbit, zmq 曾经支持 qpid

基础概念

Transport: 抽象不同 backend 通过统一的方式供上层调用

Executors: 控制接收到消息调度处理方式 , 包括 blocking, eventlet, threading

Target: 封装了关于消息将发往哪和 RPC server 应该监听什么消息的所有信息

RPC Server: 开放一些 endpoints , 每个 endpoint 包含若干可以被远程调用的方法

RPC client: 负责发送方法到远程 RPC server, 接收远程调用方法结果

oslo.messagi

```python
import eventlet
eventlet.monkey_patch()

from oslo_config import cfg
import oslo_messaging
import time

class ServerControlEndpoint(object):

    target = oslo_messaging.Target(namespace='control',
                                   version='2.0')

    def __init__(self, server):
        self.server = server

    def stop(self, ctx):
        if self.server:
            self.server.stop()

class TestEndpoint(object):

    def test(self, ctx, arg):
        return arg

transport = oslo_messaging.get_rpc_transport(cfg.CONF)
target = oslo_messaging.Target(topic='test', server='server1')
endpoints = [
    ServerControlEndpoint(None),
    TestEndpoint(),
]
server = oslo_messaging.get_rpc_server(transport, target, endpoints,
                                       executor='eventlet')
```

```python
transport = messaging.get_transport(cfg.CONF)
target = messaging.Target(topic='test', version='2.0')
client = messaging.RPCClient(transport, target)
client.call(ctxt, 'test', arg=arg)
```

```python
server.stop()
server.wait()
```

内容大纲

- 基础库 oslo 概览
- oslo 关键组件分析
- **近期版本功能更新**

oslo.policy 更新

关于 oslo.policy

- 根据 policy.json 规则检验 API 授权
- 静态读取配置文件 policy.json

现阶段问题

- 部署者必须定义所有规则
- 不能在代码里嵌入默认规则
- 没有方法获得需要设置哪些规则
- 没有办法生成规则帮助文字

改进方式

- 增加注册 policy 规则机制
- 添加自动生成 policy 规则功能
- 允许 Policy 规则传入帮助文字及生成

```
{
    "context_is_admin":  "role:admin",
    "admin_or_owner":  "is_admin:True or project_id:%(project_id)s",
    "default": "rule:admin_or_owner",

    "cells_scheduler_filter:TargetCellFilter": "is_admin:True",

    "compute:create": "rule:admin_or_owner",
    "compute:create:attach_network": "rule:admin_or_owner",
    "compute:create:attach_volume": "rule:admin_or_owner",
    "compute:create:forced_host": "is_admin:True",

    "compute:get": "rule:admin_or_owner",
    "compute:get_all": "rule:admin_or_owner",
    "compute:get_all_tenants": "is_admin:True",

    "compute:update": "rule:admin_or_owner",

    "compute:get_instance_metadata": "rule:admin_or_owner",
    "compute:get_all_instance_metadata": "rule:admin_or_owner",
    "compute:get_all_instance_system_metadata": "rule:admin_or_owner",
    "compute:update_instance_metadata": "rule:admin_or_owner",
    "compute:delete_instance_metadata": "rule:admin_or_owner",

    "compute:get_diagnostics": "rule:admin_or_owner",
    "compute:get_instance_diagnostics": "rule:admin_or_owner",

    "compute:start": "rule:admin_or_owner",
    "compute:stop": "rule:admin_or_owner",
```

oslo.config 更新

Mutable Config options

- 允许运行时改变配置项的值
- oslo.config , oslo.service，配置项共同起作用
- 典型应用 logging 配置项 debug=True

Machine Readable Sample Config

- 目前可以生成 ini 格式的 sample 配置文件
- 部署工具需要机器可读格式 (yaml, json) sample 配置文件

开发中新功能

- 配置项 validator
- plugin 方式从非 ini 文件获得配置项，典型应用 etcd.

其它更新

Global request id

- 每个操作在日志文件对应一个 Request id
- 目前一个操作在不同组件对应不同 request id
- 一个操作各个组件使用同一个 request id 便于运维
- 多个组件修改： oslo.context, oslo.log, nova/neutron-client

接收 Castellan

- generic key manager interface for OpenStack
- Barbican 与 Oslo 团队共同维护
- Barbican 是 Castellan driver 之一
- Vault 支持正在开发之中

未来

THANK YOU