# **第一周：****熟悉开发环境**

### **注意：本课程需要你已经具备基本的 Python 和 Flask 基础，如果对这两者还很陌生，请先学习本课详情中推荐的先学课程。 另外，由于需要在环境中提交代码，请开通“我的代码库”。**

推荐先学以下免费课程：

* [《Python快速教程》](https://www.shiyanlou.com/courses/214)
* [《Flask Web框架》](https://www.shiyanlou.com/courses/29)
* [《Git Community Book》](https://www.shiyanlou.com/courses/4)
* [《Bootstrap入门教程》](https://www.shiyanlou.com/courses/50)

## **一、实验简介**

本节实验为实验楼训练营课程《仿 OpenStack 开发云计算管理软件》第1节，希望与大家一起完成下面几项任务：

1. 云平台项目功能模块的分析与设计
2. 搭建开发环境
3. 基于 Flask 的基本框架和 Bootstrap 的极简页面

## **二、项目目标**

在6周的时间里，我们将在实验楼中开发一个简单的云计算管理软件，我们称之为为LouCloud，当然你也可以为其任意命名，因为这个项目将由你独立完成。

LouCloud 将具备[OpenStack](https://www.openstack.org/)的基本功能，具备简单的服务器和虚拟机管理功能，可以用于在局域网内构建小型私有云。

LouCloud 的基本功能如下：

1. 用户管理：具备管理员和普通用户两种用户角色，整个系统中将只有一个管理员，管理员用户在系统初始化时创建，普通用户由管理员创建。
2. 服务器管理：具备添加，删除服务器到资源池的功能，同时支持多台服务器的管理。
3. 镜像管理：镜像的添加和删除功能。镜像用来创建虚拟机的磁盘，是一个已经装好系统同时做了一定配置的特殊磁盘。
4. 虚拟机模板管理：虚拟机模板定义了基于此模板创建的虚拟机的基本配置信息，比如虚拟机拥有多少个 CPU，多少内存以及使用哪个镜像。
5. 虚拟机管理：虚拟机的创建，启动，关闭，重启，删除操作。
6. 配置管理：系统的一些配置信息，比如新用户默认设置等。

下面几节内容将对这些功能模块进行一一分析。

### **2.1 用户管理**

#### **2.1.1 用户角色**

包含两种角色，每种角色具备不同的管理权限：

* 管理员：系统唯一，初始化时创建。具备用户管理，服务器管理，镜像管理，模板管理，配置管理功能。
* 普通用户：由管理员添加，具备虚拟机管理功能。

#### **2.1.2 添加用户**

管理员操作，输入用户基本信息创建新用户。

#### **2.1.3 删除用户**

管理员操作。

#### **2.1.4 更新用户**

更新用户基本信息。

#### **2.1.5 用户更新配置**

用户可以修改密码，用户名等配置。

### **2.2 服务器管理**

服务器代表的是物理资源，LouCloud 统一管理若干台服务器构成的物理资源池，用户按需创建虚拟机，虚拟机根据调度策略选择在物理资源池中的其中一台服务器上运行。

#### **2.2.1 添加服务器**

输入服务器的 IP 地址，将服务器加入到 LouCloud 系统管理的资源池。

#### **2.2.2 删除服务器**

删除指定的服务器，前提是服务器上没有运行的虚拟机。

### **2.3 镜像管理**

#### **2.3.1 添加镜像**

输入镜像的路径及名称，添加到 LouCloud 系统中。

#### **2.3.2 删除镜像**

在 LouCloud 系统中删除镜像记录，但不会在服务器上删除。

### **2.4 模板管理**

#### **2.4.1 添加模板**

输入模板的基本配置，添加到 LouCloud 系统中。

#### **2.4.2 删除模板**

在 LouCloud 系统中删除模板。

#### **2.4.3 更新模板**

更新模板的基本配置。

### **2.5 虚拟机管理**

### **2.5.1 创建**

选择模板，输入虚拟机名称，创建虚拟机。

### **2.5.2 删除**

删除虚拟机。

### **2.5.4 开机**

启动虚拟机。

### **2.5.5 关机**

关闭虚拟机。

### **2.5.6 重启**

重启虚拟机。

### **2.5.7 连接虚拟机**

通过 Web 桌面的方式连接虚拟机。

### **2.6 配置管理**

添加系统所需的配置信息，比如用户虚拟机配额等。

综上，这就是我们6星期内要完成的产品需求，如果有任何不清楚的地方，请随时点击实验文档下方的课程问答发起提问。

## **三、云平台架构设计**

### **3.1 相关知识点**

这些技术都会应用在项目的架构设计中，你完成实验内容的基础上花更多的时间阅读我们推荐的技术文章，这样才能更透彻的学习到更多细节。

* 操作系统：[Linux 基本操作](https://www.shiyanlou.com/courses/1)
* 编程语言：[Python 语言开发](https://www.shiyanlou.com/courses/214)
* 开发框架：[Flask Web 框架](https://www.shiyanlou.com/courses/29)，[MVC 开发模式](https://zh.wikipedia.org/wiki/MVC)
* 虚拟化技术： [KVM/QEMU](http://blog.csdn.net/zwan0518/article/details/10613481)，[Libvirt API 接口开发](https://www.ibm.com/developerworks/cn/linux/l-libvirt/)
* 云计算技术：[基础设施即服务（IaaS）基本概念](http://www.zdnet.com.cn/wiki-IaaS)
* 数据库：[MySQL 数据库设计](http://blog.csdn.net/taijianyu/article/details/5945490)及 [SQLAlchemy 接口开发](http://www.jb51.net/article/49789.htm)
* 开发流程：[Git 基本操作及代码库使用](https://www.shiyanlou.com/courses/4)
* 安装部署：Virtualenv，Apache，WSGI等
* 前端开发：[Bootstrap 3.0](https://www.shiyanlou.com/courses/50)，[HTML](https://www.shiyanlou.com/courses/19)，[Javascript](https://www.shiyanlou.com/courses/21)

### **3.2 技术架构**

整个项目的知识点比较集中，Web 开发 + 虚拟化开发。底层为 MySQL 数据库与 QEMU 虚拟化，中间层的接口分别为 Flask-SQLAlchemy 及 Libvirt API。上层为 Flask Web 开发框架。

项目在实验楼的 Linux 环境下进行开发，实验环境标配的是 Ubuntu 14.04 64位 的环境。项目选用 Python 语言，Web 开发框架选用 Flask 。虚拟机操作使用 Libvirt API ，数据库采用 MySQL ，代码管理使用 Git ，采用 WSGI 方式部署到 Apache 上，前端基于 Bootstrap 开发。

## **四、开发环境搭建**

LouCloud 使用 Python 语言的 Flask Web 开发框架。

Flask 是一个使用 Python 开发的微型开发框架，基于 Werkzeug WSGI 工具箱和 Jinja2 模板引擎。 Flask 也被称为 “microframework” ，因为它具有极简的核心，然而却可以通过扩展的方式增加其他功能。 Flask 默认不支持数据库连接，不支持用户登陆认证等功能。但是，我们可以通过各种扩展获得对这些功能的支持，比如：数据库 ORM 、表单验证、文件上传、开放式身份验证技术。这使得 Flask 成为具有高度扩展性和灵活性的 Python Web 框架。

### **4.1 安装 virtualenv**

sudo apt-get update

sudo apt-get install python-virtualenv

Virtualenv 用于创建独立的 Python 开发环境，不同版本的Python开发环境可以相互独立，互不影响，它能够：

1. 在没有 Linux 管理员权限的情况下安装各种 Python 模块；
2. 不同应用可以使用不同版本的 Python 模块 ；
3. 模块升级不影响其他应用；

后续 LouCloud 中所需的 Python 模块都将以 Virtualenv 的方式安装。

### **4.2 安装虚拟化相关组件**

虚拟化技术简单来说就是将一台物理服务器虚拟出虚拟 CPU ，内存等资源，将操作系统运行在这些虚拟资源构成的资源池上。虚拟化技术将在第二节实验中详细介绍。KVM ，Xen ，VMware ESXi 为目前最广泛使用的虚拟化软件，这类虚拟化软件通常需要 CPU 上的硬件支持虚拟化技术。由于实验楼的开发环境为虚拟机，所以无法直接启动需要硬件虚拟化技术支持的 KVM，退而求其次我们选择 QEMU 虚拟化。

Libvirt API 是一套用于多种虚拟化平台统一管理的接口，详细文档：[http://libvirt.org](http://libvirt.org/)，我们基于 Libvirt 开发虚拟机的管理模块，virsh 是 Libvirt 的一套命令行接口，我们可以使用各种 virsh 命令来管理虚拟机的生命周期。

python-libvirt 是 Python 版本的 Libvirt API。

# 安装虚拟化组件

sudo apt-get install qemu libvirt-bin python-libvirt

# 启动 libvirt-bin 服务

sudo service libvirt-bin start

# 查看 libvirt-bin 服务状态

sudo service libvirt-bin status

# 查看当前虚拟机列表，具体可使用virsh --help 帮助命令

sudo virsh list

### **4.3 MySQL 相关组件**

由于实验环境中已经安装了 MySQL 服务器，所以这一步操作可以省略。

sudo apt-get install mysql-server

实验楼环境中的 MySQL 管理员账户为 root ，密码为空，这个账户我们后续会用到。

### **4.4 Flask 代码框架**

#### **LouCloud v0.1的代码文件结构**

|-- README.md

|-- loucloud

| |-- \_\_init\_\_.py

| |-- app.py

| |-- config.py

| |-- extension.py

| |-- static

| |-- templates

| `-- user

| |-- \_\_init\_\_.py

| `-- views.py

`-- manage.py

其中：

1. manage.py 提供了测试和初始化命令；
2. loucloud 模块为核心代码；
3. loucloud/user 模块提供用户登陆认证及用户管理等基本操作；
4. loucloud/static 与 loucloud/templates 用来存放界面实现所需的静态文件以及 jinja2 模板；
5. loucloud/app.py loucloud/**init**.py 及loucloud/config.py 用于初始化和部署 Flask 应用；
6. loucloud/extension.py 用来初始化和配置 Flask 的扩展，例如 Flask-Login 等；

#### **创建必要的代码目录**

cd /home/shiyanlou/Code/shiyanlou\_cs354

touch manage.py

mkdir loucloud

cd loucloud

mkdir user static templates

touch \_\_init\_\_.py config.py app.py extension.py

cd user

touch \_\_init\_\_.py views.py

cd ../../

#### **提交代码**

注意如果一次实验没有完成，实验楼的环境是不保存的，需要手动按照下述步骤将你的代码提交到我的代码库，以防止数据丢失。

注意，虽然本次创建的文件都为空，但仍然可以提交。

cd /home/shiyanlou/Code/shiyanlou\_cs354

# 查看当前文件修改的列表

git status

# commit

git add \*

git commit -m 'first commit of loucloud'

# push到git.shiyanlou.com远程仓库

git push origin master

### **4.5 Flask 及相关扩展**

1. Flask：Flask 框架基础包
2. Flask-SQLAlchemy：在 Flask 中使用的 SQLALchemy ORM，用于数据库操作；
3. Flask-WTF：页面表单扩展；
4. Flask-Cache：缓存扩展；
5. Flask-Login：用户登陆认证及会话管理组件；
6. Flask-Script：Flask 的脚本支持，例如 manage.py 这类启动和管理脚本；

### **创建 requirement.txt 文件**

该文件中包含开发环境所需要的各种 Python 模块。

touch requirement.txt

vim requirement.txt

输入以下内容，每个包占一行

flask

flask-sqlalchemy

flask-wtf

flask-cache

flask-login

flask-script

#### **虚拟环境初始化与进入**

cd /home/shiyanlou/Code/shiyanlou\_cs354

# 初始化虚拟环境venv

virtualenv venv

# 进入virtualenv venv

source venv/bin/activate

退出虚拟环境只需要输入deactivate命令即可。

deactivate

#### **安装 Python 模块**

进入到 Virtualenv 虚拟环境后执行下面的命令即可安装 requirement.txt 文件中的 Python 模块到 Virtualenv 虚拟环境的路径中。

pip install -r requirement.txt

### **4.6 其他 Python 模块**

下述 Python 模块分别具备不同的功能，将他们安装在我们的代码环境中最大的作用是避免我们把时间浪费在重复制作轮子上。

1. paramiko：SSH 链接管理组件；
2. mysql-python：MySQL 链接管理；
3. libvirt：Libvirt 虚拟化管理组件

由于 mysql-python 模块依赖于 libmysqlclient-dev，python-dev 软件包，所以需要先通过 apt-get 安装这些包再执行后续操作。

sudo apt-get install libmysqlclient-dev python-dev

**请自己将上述 Python 模块 paramiko 和 mysql-python 添加到requirement.txt文件中，然后按照上述步骤进入到virtualenv中执行安装。**

libvirt 比较特殊，pip 安装会失败，所以我们在4.2中选择直接安装到系统中，在这里我们需要把系统中的包拷贝到当前virtualenv环境中。

cp /usr/lib/python2.7/dist-packages/libvirt\* /home/shiyanlou/Code/shiyanlou\_cs354/venv/lib/python2.7/site-packages

测试下依赖的 Python 模块是否都已经起作用：

cd /home/shiyanlou/Code/shiyanlou\_cs354

source venv/bin/activate

python

>> import libvirt

>> import flask

>> import paramiko

有的时候 pip 没有把 paramiko 的依赖包安装全，import paramiko时会报错，需要手动执行pip install pycrypto ecdsa后才可以。

如果上述命令都没有错误即证明所需 Python 模块已安装成功。

### **4.7 Bootstrap**

Bootstrap 是 LouCloud 项目使用的前端框架，主要原因是简单易用，可以让不懂 js ，css 的后端程序员快速写出简单的页面。

Bootstrap 无需安装，仅仅在页面中引入下述的链接即可，我们本节实验没有用到，后续将再次回顾这一内容。

<!-- Bootstrap 核心 CSS 文件 -->

<link rel="stylesheet" href="http://labfile.oss.aliyuncs.com/bootstrap/3.3.2/css/bootstrap.min.css">

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->

<script src="http://labfile.oss.aliyuncs.com/jquery/1.11.1/jquery.min.js"></script>

<!-- Bootstrap 核心 JavaScript 文件 -->

<script src="http://labfile.oss.aliyuncs.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>

### **4.8 添加 .gitignore**

由于 venv 目录包含所有依赖的软件包，接近29M，所以我们在提交代码时无需提交该目录，创建 .gitignore 文件可以达到这个目的，这个文件中可以使用通配符，文件名及目录名来告诉 git 忽略一些文件或文件夹。

cd /home/shiyanlou/Code/shiyanlou\_cs354

touch .gitignore

echo "venv" >> .gitignore

**注意：由于非会员的实验楼环境不会保存，每次开始实验后先执行一次下述虚拟环境初始化操作：**

virtualenv venv

source venv/bin/activate

pip install -r requirement.txt

cp /usr/lib/python2.7/dist-packages/libvirt\* /home/shiyanlou/Code/shiyanlou\_cs354/venv/lib/python2.7/site-packages

deactivate

### **4.9 提交代码**

参考4.4中的代码提交步骤将修改的代码进行保存。

## **五、最小的可运行代码**

现在我们需要将 Flask 程序运行起来，需要按照先前学过的 Flask 知识对项目进行最小运行代码的添加。

Tip：为了快速输入代码，可以使用[实验楼的Web IDE功能](https://www.shiyanlou.com/questions/968)。

**注意：本节所提供的程序仅供参考，仍需你根据自己的 Flask 基础进行编写，可以无需完全按照下述代码框架实现。如果你对下述代码任何一行有疑问，可以在实验楼问答中提出，在开课期间，我们会在24小时内及时回复。**

### **5.1 loucloud/config.py**

用来初始化Flask 应用的配置信息，代码如下，主要是Flask各个插件的配置：

# -\*- coding: utf-8 -\*-

import os

class DefaultConfig(object):

PROJECT = "loucloud"

DEBUG = False

TESTING = False

# http://flask.pocoo.org/docs/quickstart/#sessions

SECRET\_KEY = 'your secret key'

# Flask-Sqlalchemy: http://packages.python.org/Flask-SQLAlchemy/config.html

SQLALCHEMY\_ECHO = True

# MYSQL for production.

#SQLALCHEMY\_DATABASE\_URI = 'mysql://username:password@server/db?charset=utf8'

# Flask-cache: http://pythonhosted.org/Flask-Cache/

CACHE\_TYPE = 'simple'

CACHE\_DEFAULT\_TIMEOUT = 60

### **5.2 loucloud/extension.py**

初始化flask-sqlalchmey，flask-login及flask-cache三个插件：

# -\*- coding: utf-8 -\*-

from flask.ext.sqlalchemy import SQLAlchemy

db = SQLAlchemy()

from flask.ext.cache import Cache

cache = Cache()

from flask.ext.login import LoginManager

login\_manager = LoginManager()

### **5.3 loucloud/app.py**

# -\*- coding: utf-8 -\*-

import os

from flask import Flask, request, render\_template

from .config import DefaultConfig

from .extension import db, cache, login\_manager

from .user import user

# For import \*

\_\_all\_\_ = ['create\_app']

DEFAULT\_BLUEPRINTS = (

user,

)

def create\_app(config=None, app\_name=None, blueprints=None):

"""Create a Flask app."""

if app\_name is None:

app\_name = DefaultConfig.PROJECT

if blueprints is None:

blueprints = DEFAULT\_BLUEPRINTS

app = Flask(app\_name)

configure\_app(app, config)

configure\_extension(app)

configure\_blueprints(app, blueprints)

return app

def configure\_app(app, config=None):

"""Different ways of configurations."""

# http://flask.pocoo.org/docs/api/#configuration

app.config.from\_object(DefaultConfig)

def configure\_extension(app):

# flask-sqlalchemy

db.init\_app(app)

# flask-cache

cache.init\_app(app)

# flask-login

#login\_manager.login\_view = 'user.index'

#login\_manager.setup\_app(app)

def configure\_blueprints(app, blueprints):

"""Configure blueprints in views."""

for blueprint in blueprints:

app.register\_blueprint(blueprint)

上述程序中由于用户表在数据库中尚未建立，所以 login\_manager 代码暂时注释。

### **5.4 loucloud/\_\_init\_\_.py**

将app创建函数导出来，供外部引用。

# -\*- coding: utf-8 -\*-

from app import create\_app

### **5.5 loucloud/user/**

user模块将用来实现用户管理功能，不是本周的任务，本周只实现一个简单的测试页面即可。

user目录下有两个文件，相信学过Flask后你应该很熟悉这两个文件的功能。其中需要提到的是 Blueprint。Flask 使用 Blueprint 让应用实现模块化，在Flask中可以具备多个Blueprint，每个Blueprint注册到一个URL下比如 “/”、“/sample”等，详细可见 Flask 官方文档。

#### **loucloud/user/views.py**

# -\*- coding: utf-8 -\*-

from flask import Blueprint

from flask import current\_app as APP

user = Blueprint('user', \_\_name\_\_, url\_prefix='/')

@user.route('/')

def index():

return "Hello, LouCloud!"

#### **loucloud/user/\_\_init\_\_.py**

# -\*- coding: utf-8 -\*-

from .views import user

### **5.6 manage.py**

使用 Flask-Script 实现基本的运行命令。

# -\*- coding: utf-8 -\*-

from flask.ext.script import Manager

from loucloud import create\_app

app = create\_app()

manager = Manager(app)

@manager.command

def run():

"""Run in local machine."""

app.run()

if \_\_name\_\_ == "\_\_main\_\_":

manager.run()

### **5.7 运行代码**

cd /home/shiyanlou/Code/shiyanlou\_cs354

source venv/bin/activate

python manage.py run

打开实验环境桌面上的浏览器，访问http://localhost:5000，将看到Hello, LouCloud！字样，证明程序已经正常运行。这个时候不要忘记截图保存。

### **5.8 提交代码**

是否还记得如何提交代码。

提交后，我们将使用git tag给版本做个标记v0.1，表明我们的第一个最简版本已经完成。

cd /home/shiyanlou/Code/shiyanlou\_cs354

git tag

git tag -a v0.1 -m "LouCloud v0.1"

git push origin –-tags

## **六、本节总结及下节预告**

本节实验简单介绍了开发环境的搭建以及 Flask 框架的使用。下一节将主要介绍虚拟化技术以及如何使用 Libvirt 进行虚拟化开发。

## **七、实验报告**