DevStack是一系列用于快速部署一个完整openstack环境的可扩展脚本，其中openstack是基于gitmaster分支上的最新版本。它是一个交互式的开发环境和许多openstack项目功能测试的基础。

在<https://git.openstack.org/cgit/openstack-dev/devstack>上可找到资源。

Warning：

在安装期间，DevStack将对你的系统做出大量更改。只能在服务器或者专门用于部署openstack的虚拟机上运行DevStack。

快速入门

## 安装linux

首先安装一个纯净的低版本linux系统。DevStack已经开始支持Ubuntu 16.04/17.04, Fedora 24/25, CentOS/RHEL 7， 以及其它Debian 和OpenSUSE系统。

如果你没有特殊偏好，那么Ubuntu 16.04将是最好的选择。

## 添加一个stack用户

DevStack需要在一个非管理员但拥有sudo权限的用户下运行。

你可以用以下命令来快速创建一个独立的stack用户来运行DevStack:

$ sudo useradd -s /bin/bash -d /opt/stack -m stack

既然该用户将大量更改你的系统，它就需要以下命令来赋予stack用户sudo权限：

$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

$ sudo su – stack

## 下载DevStack

$ git clone https://git.openstack.org/openstack-dev/devstack

$ cd devstack

DevStack repo包含一个openstack的安装脚本和一些配置文件模板。

## 创建一个local.conf，写入下面命令，设置四个密码：

[[local|localrc]]

ADMIN\_PASSWORD=secret

DATABASE\_PASSWORD=$ADMIN\_PASSWORD

RABBIT\_PASSWORD=$ADMIN\_PASSWORD

SERVICE\_PASSWORD=$ADMIN\_PASSWORD

以上就是开始一个DevStack的最低配置。

## 开始安装

./stack.sh

这个过程大概 需要15至20分钟，这很大程度上取决于你的网速。在此过程中，许多git树和包会被安装。

## 安装结果：

至此，DevStack已经帮你安装了keystone，glance，nova, cinder, neutron和horizon。动态IP可以让用户连接到外网。

你可以通过openstack web 上的接口来管理 vm, networks volumes和image。

你可以在shell中使用source openrc命令和openstack的命令工具来管理DevStack。

你可以使用cd /opt/stack/tempest命令并且运行已经配置好的 tempest tests来让你的DevStack生效。

## 更多：

你也可以了解更多DevStack的配置，以便按你的需求定制DevStack，包括更改默认网络设置。

如果有特殊安装需求，需要阅读guides；开启devstack plugins可以让openstack支持一些基础openstack中没有的服务、特性和配置；the big picture帮助我们为DevStack贡献代码。

Configuration

## [local.conf](https://docs.openstack.org/devstack/latest/configuration.html#id3)[¶](https://docs.openstack.org/devstack/latest/configuration.html#local-conf)

通过local.conf文件来修改DevStack的配置。

在devstack/samples下有一个例子。

local.conf文件的头部和一般的INI文件的头部很像，但是用的是这样的括号（[[ … ]]），并且内部字段用管道（|）来分割。注意，括号和字段之间，以及管道和字段之间没有空格：

'[[' <phase> '|' <config-file-name> ']]'

这里的<phase>是一个stack.sh里面定义的phase名字的集合，<config-gile-name>是配置文件名。文件名会在stack.sh中使用，因此所有的环境变量都是可用的而且都可能被使用。强烈建议在项目配置文件头部定义变量（看下面NOVA\_CONF的例子）.如果配置文件的路径不存在，则跳过。

定义的phases有：

* **local**—在stackrc被sourced（shell）之前，从local.conf中引用localrc。
* **post-config**—在二层服务被配置之后，启动之前运行。
* **extra**—在服务开始之后，extra.d中的所有文件被执行之前运行。
* **post-extra—**extra.d中的所有文件被执行之后运行。

这个文件将在序列中严格执行；meta-sections可能被多次定义，但以最后一次为准。

[[post-config|$NOVA\_CONF]]

[DEFAULT]

use\_syslog = True

[osapi\_v3]

enabled = False

一个特殊的meta-section local|localrc 用于提供默认的localr文件（实际上是.localrc.auto）。它允许将对DevStack的所有用户设置包含在一个文件中。如果localrc存在，那么它会被作为向后兼容的文件保护起来。

[[local|localrc]]

IPV4\_ADDRS\_SAFE\_TO\_USE=10.254.1.0/24

ADMIN\_PASSWORD=speciale

LOGFILE=$DEST/logs/stack.sh.log

注意，Q\_PLUGIN\_CONF\_FILE比较特别，前面需要加一个/。

[[post-config|/$Q\_PLUGIN\_CONF\_FILE]]

同样需要注意的是，localrc作为一个shell脚本被sourced，所以它需要尊徐shell的标准，尤其是等号两边没有空格。

## [openrc](https://docs.openstack.org/devstack/latest/configuration.html#id4)[¶](https://docs.openstack.org/devstack/latest/configuration.html#openrc)

openrc 配置登录凭证适用于openstack命令行工具。为了获取HOST\_IP和SERVICE\_HOST（用于 denpoint），openrc在开始时source stackrc（这个过程会依次source local.conf中的localrc）。下面是他们的默认值。

**OS\_PROJECT\_NAME (OS\_TENANT\_NAME)**

Keystone已经将团队项目作为拥有资源的实体，并将其标准化。但一些地方的引用依旧是旧版本的团队租户。同时Project\_name更习惯被叫做project\_id。OS\_TENANT\_NAME仍然支持和兼容旧版的工具。

OS\_PROJECT\_NAME=demo

**OS\_USERNAME**

除了拥有实体，openstack将实体的执行者叫做用户。

OS\_USERNAME=demo

**OS\_PASSWORD**

Keystone的认证需要提供一个密码。在环境变量中设置密码的注意事项，在DevStack中基本上是可接受的折中。

HOST\_IP=127.0.0.1

SERVICE\_HOST=$HOST\_IP

**OS\_AUTH\_URL**

Keystone认证之后会返回一个令牌和一个服务列表。这个列表包含了用户访问如nova，swift，glance，keystone等服务的url。

OS\_AUTH\_URL=http://$SERVICE\_HOST:5000/v2.0

**KEYSTONECLIENT\_DEBUG, NOVACLIENT\_DEBUG**

设置在客户端可以debug，默认是不能的。

*# export KEYSTONECLIENT\_DEBUG=1*

*# export NOVACLIENT\_DEBUG=1*

[Minimal Configuration](https://docs.openstack.org/devstack/latest/configuration.html#id5)[¶](https://docs.openstack.org/devstack/latest/configuration.html#minimal-configuration)

在没有设置local.conf中的localrc时，stack.sh还是可以运行的，但最好还是设置一些变量。下面就是一个最小配置的例子。