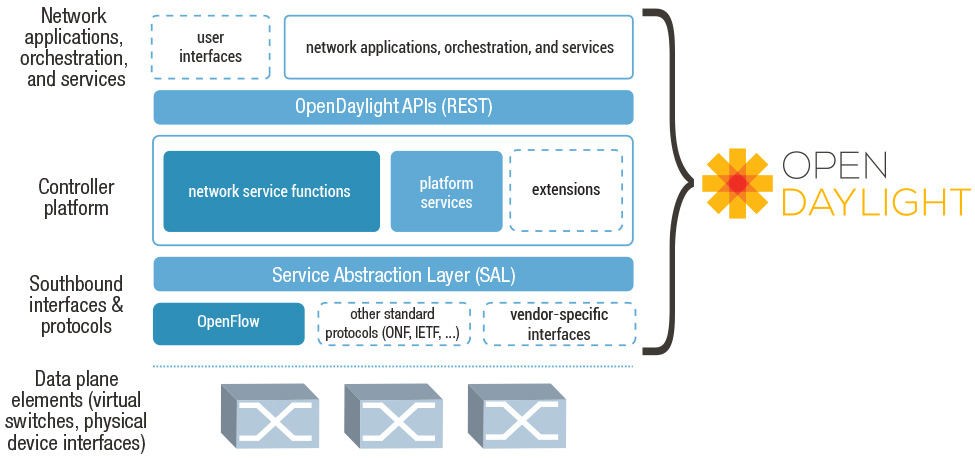
# 基础知识

## 1.1 OpenDaylight

OpenDaylight是一套以社区为主导的开源框架，旨在推动创新实施以及软件定义网络（简称SDN）透明化。面对SDN型网络，大家需要合适的工具帮助自己管理基础设施，这正是OpenDaylight的专长。作为项目核心，OpenDaylight拥有一套模块化、可插拔且极为灵活的控制器，这使其能够被部署在任何支持Java的平台之上。这款控制器中还包含一套模块合集，能够执行需要快速完成的网络任务。



### 1.1.1 OpenDayLight的诞生背景

随着互联网的普及，用户数量不断攀升，网络不堪重负。移动终端发展势如破竹，智能手机不断更新换代，各种手机软件层出不穷，手机控随时随地上网，导致流量需求与日俱增，负荷过度的网络无法满足用户需求。网络体系庞大，架构臃肿，不够灵活，不能适应不断涌现出的新业务需求，服务质量得不到保证。网络体系复杂，网络操作需要与其他IT操作的集成与协作，导致网络部署困难。网络更新麻烦，动手操作过多，网络管理员分身乏术。改良已经无法解决现有的网络问题，网络改革势在必行，于是SDN应运而生。  
 SDN是美国斯坦福大学clean slate研究组提出的一种新型网络架构。传统网络采用分布式策略工作，由设备制定转发策略，而SDN架构中设备不运行任何协议，转发表由控制器下发给设备，实现数据平台与控制平台的分离。SDN的核心思想就是控制与转发分离，将软件应用到网络控制中，并起到主导作用，而不是由固定模式的协议控制网络。SDN的目的是提高网络的可控性与可编程性，可以根据用户需求灵活地提供不同质量等级的服务。

### 1.1.2 OpenDayLight诞生的利益分析

SDN的提出立刻就在业界引起了轩然大波，尤其是一直被网络设备商压制的网络用户，将其视为摆脱网络设备商牵制，翻身做主人的机会，于是2011年一个以网络用户为主导的非营利性组织ONF就此诞生了。ONF宗旨是制定SDN统一标准，推动SDN产业化。ONF的工作重点是制定唯一的南向接口标准openflow，制定硬件行为转发标准，并且推出了一系列openflow 协议，其中较为稳定的是openflow1.0和openflow1.3版本。ONF从用户的角度制定协议，必然可以维护用户的利益，但是其间也出一些问题。  
 网络设备的研发十分复杂，是一个系统化工程，需要结合方方面面考虑，需要丰富的实战经验，而这些正是网络用户所缺乏的，因此直接导致openflow协议过于理想化，只能在实验及简单网络环境中应用，无法实现大规模商用。这种情况下ONF不得不接受网络设备商的参与，2013以设备商和软件商为主导的另一SDN组织ODL腾空出世，网络设备商出于自身利益出发，也加入到SDN大军中。并不是网络设备商都不计较利益，不计得失地贡献自己的技术，设备商也有自己的考量，越来越多的人看好SDN，这是一股不可逆转的趋势，与其坐等网络用户摆脱自己，不如化被动为主动积极参与其中，众多设备商联手研发出统一的控制框架，其中可以嵌入一些服务与应用模块，各大设备商都争相在大框架中融入更多的自己的技术，因为贡献越多意味着影响越大，在ODL中争得一席之地，才能为以后的发展留下生机。换句话说，置之死地而后生，贡献出自己的核心技术，这些技术随着SDN的推广被推向世界，说不定柳暗花明又一村呢！无论各自的目的是什么，ODL与ONF有共同的目的，推动SDN和网络功能虚拟化发展，打造统一开放的SDN平台，推动SDN产业化。  
 由此看来，ODL是SDN大环境下的必然产物，不仅仅得到网络用户的认可，还受到网络设备商的鼎力支持，注定在SDN发展中脱颖而出，成为SDN的灵魂产物之一。

### 1.1.3 OpenDayLight控制器

ODL拥有一套模块化、可插拔灵活地控制平台作为核心，这个控制平台基于Java开发，理论上可以运行在任何支持Java的平台上，其官方文档推荐的最佳运行环境是最新的Linux（Ubuntu 12.04+）及JVM1.7+。  
 ODL控制器采用OSGI框架，SGI框架是面向Java的动态模型系统，它实现了一个优雅、完整和动态的组件模型，应用程序（Bundle）无需重新引导可以被远程安装、启动、升级和卸载，通过OSGI捆绑可以灵活地加载代码与功能，实现功能隔离，解决了功能模块可扩展问题，同时方便功能模块的加载与协同工作。  
ODL控制平台引入了SAL，SAL北向连接功能模块，以插件的形式为之提供底层设备服务，南向连接多种协议，屏蔽不同协议的差异性，为上层功能模块提供一致性服务，使得上层模块与下层模块之间的调用相互隔离。SAL可自动适配底层不同设备，使开发者专注于业务应用的开发。  
 此外，ODL控制平台采用了Infinispan技术，Inﬁnispan是一个高扩展性、高可靠性、键值存储的分布式数据网格平台，选用Infinispan来实现数据的存储、查找及监听，用开源网格平台实现controller的集群。

### 1.1.4 OpenDayLight控制器设计原则

ODL在设计的时候遵循了六个基本的架构原则（以下来自opendaylight官方文档）：

1、运行时模块化和扩展化（Runtime Modularity and Extensibility）：支持在控制器运行时进行服务的安装、删除和更新。

2、多协议的南向支持（Multiprotocol Southbound）：南向支持多种协议。

3、服务抽象层（Service Abstraction Layer）：南向多种协议对上提供统一的北向服务接口。MD-SAL（Model Driven Service Abstraction Layer）是opendaylight 的一个主要feature。

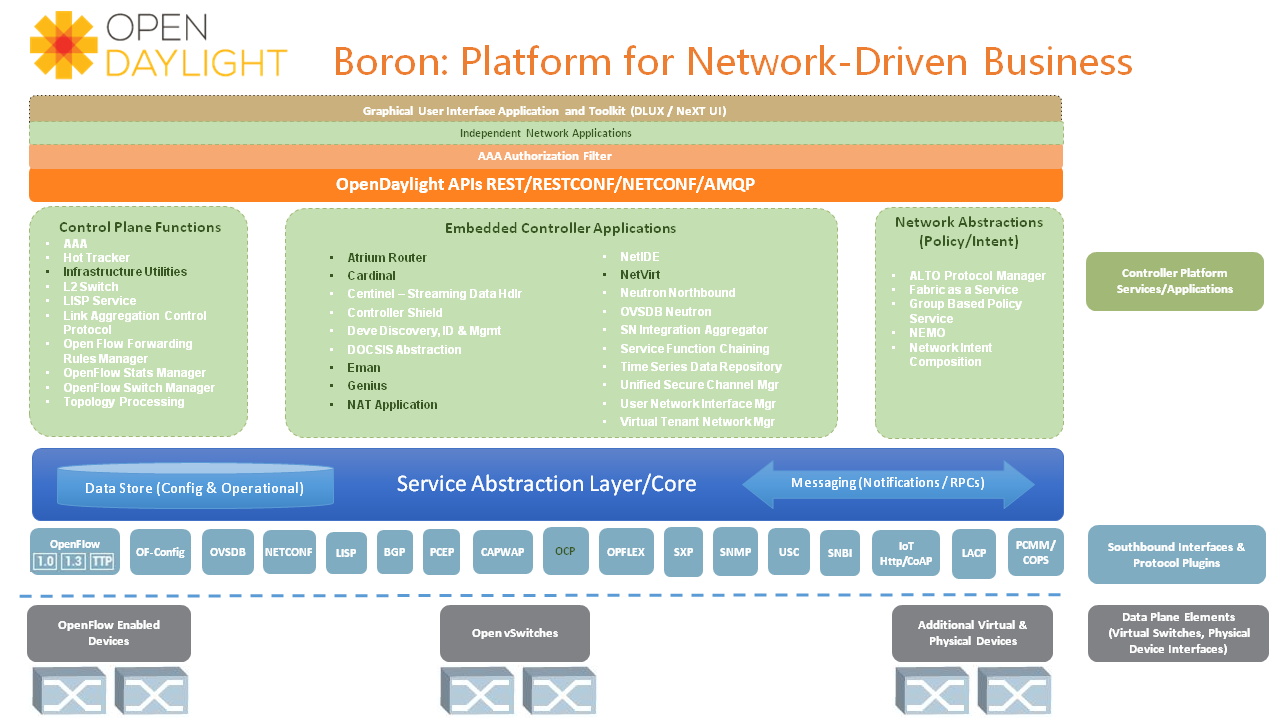
4、开放的可扩展北向API（Open Extensible Northbound API）：提供可扩展的应用API，通过REST或者函数调用方式。两者提供的功能要一致。

5、支持多租户、切片（Support for Multitenancy/Slicing）：允许网络在逻辑上（或物理上）划分成不同的切片或租户。控制器的部分功能和模块可以管理指定切片。控制器根据所管理的分片来呈现不同的控制观测面。

6、一致性聚合（Consistent Clustering）：提供细粒度复制的聚合和确保网络一致性的横向扩展（scale-out）。

### 1.1.5 OpenDayLight控制器架构

如下图所示，ODL控制器主要包括开放的北向API，控制器平面，以及南向接口和协议插件。北向API有OSGI和REST两类，同一地址空间应用使用OSGI类，而不同地址空间的应用则使用REST类。OSGI是有状态的连接，有注册机制，而rest是无状态链接。上层应用程序利用这些北向API获得网络智能信息、运行算法进行分析并且设计部署新的网络策略。  
  
 控制器平台包括一系列功能模块，可动态组合提供不同服务。其中主要包括拓扑管理、转发管理、主机监测、交换机管理等模块。服务抽象层SAL是控制器模块化的核心，自动适配底层不同的设备，使开发者专注于业务应用的开发。SAL北向连接功能模块，以插件的形式为之提供底层设备服务。南向连接多种协议插件，屏蔽不同协议的差异性，为北向功能模块提供一致性服务，SAL起到中间调度作用。  
 南向接口支持多种不同协议，如openflow1.0、openflow1.3、BEG-LS等。底层支持混合模式交换机和经典openflow交换机。



## 1.2MAVEN

Maven是一个项目管理和综合工具。[Maven](http://www.yiibai.com/maven)提供了开发人员构建一个完整的生命周期框架。开发团队可以自动完成项目的基础工具建设，Maven使用标准的目录结构和默认构建生命周期。

Maven这个个项目管理和构建自动化工具，越来越多的开发人员使用它来管理项目中的jar包。但是对于我们程序员来说，我们最关心的是它的项目构建功能。所以这里我们介绍的就是怎样用 maven 来满足我们项目的日常需要。Maven 使用惯例优于配置的原则 。它要求在没有定制之前，所有的项目都有如下的结构：



一个 maven 项目在默认情况下会产生 JAR 文件，另外 ，编译后 的 classes 会放在 basedir/target/classes 下面， JAR 文件会放在 ${basedir}/target 下面。

概括地说，Maven简化和标准化项目建设过程。处理编译，分配，文档，团队协作和其他任务的无缝连接。 Maven增加可重用性并负责建立相关的任务。

# 环境软硬件要求

下面是开发环境的要求:

操作系统: Windows 7 64位

计算机内存: 2G或2G以上

接下来几章主要是介绍如何在Windows 7上一步步搭建OpenDayLight的开发环境:先准备一个JDK的环境，再配置Maven，配置Eclipse并应用Maven，再从github上下载源码，使用maven编译，最后运行测试。

# JDK配置

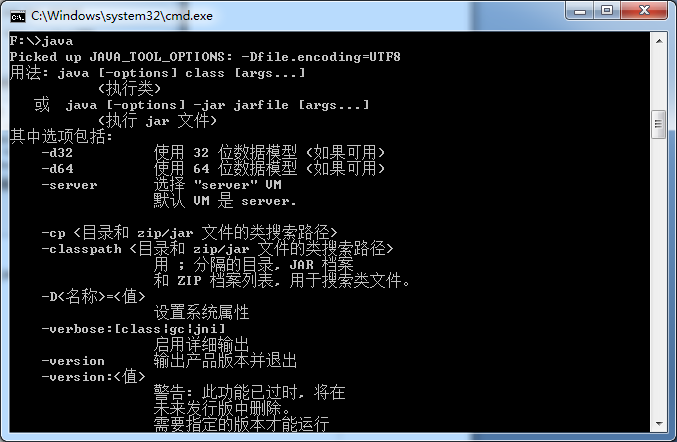
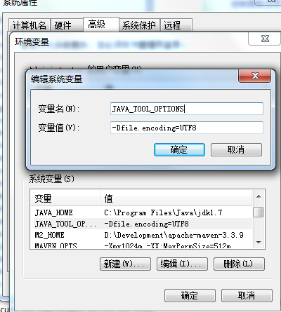
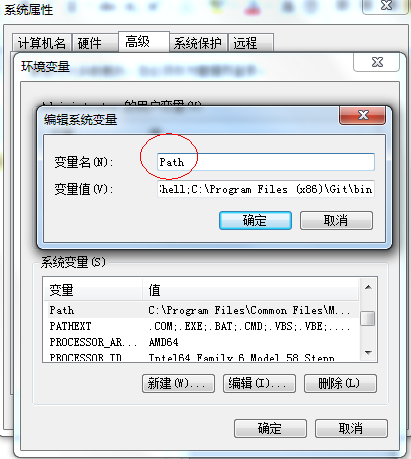
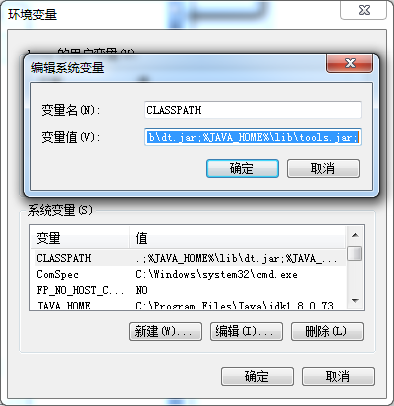
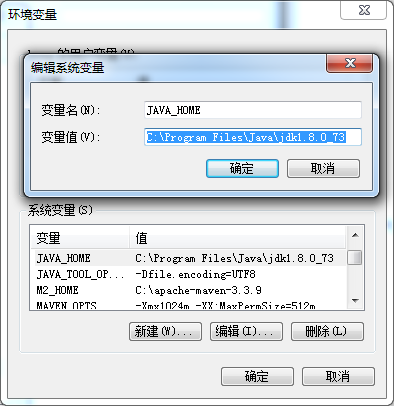
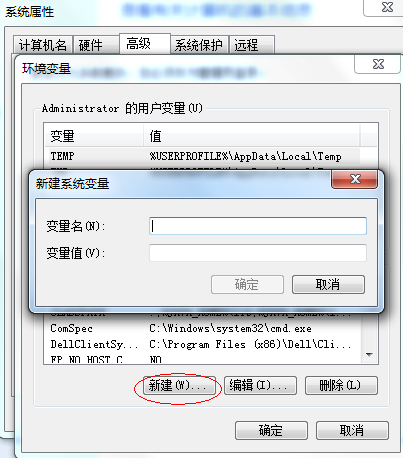
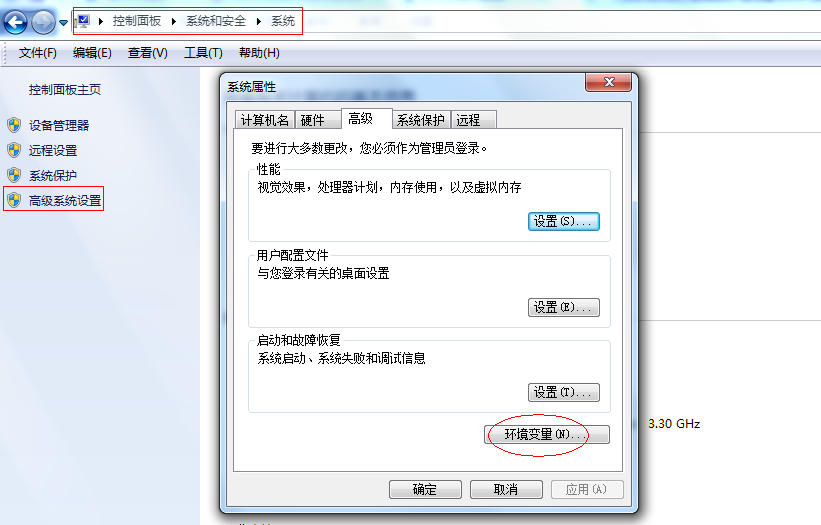
1、采用版本：jdk-8u131-windows-x64

2、安装包已放于附件中。

3、找到jdk-8u131-windows-x64.exe并下载

4、安装，双击jdk-8u131-windows-x64.exe该文件下一步下一步即可，安装目录为：C:\Program Files\Java\jdk1.8.0\_131

5、配置环境变量JAVA\_HOME，从控制面板->系统和安全->系统，选择左侧的高级系统设置  
  
6、点击“环境变量...”，进入配置，点击最下方的“新建(N)...”，  
  
7、新建系统环境变量，变量名为JAVA\_HOME，变量值是C:\Program Files\Java\jdk1.8.0\_131  
  
8、配置环境变量CLASSPATH，同上，新建系统环境变量为CLASSPATH，值为：  
.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;  
  
9、修改path环境变量，添加%JAVA\_HOME%\bin  
  
10、新增JAVA\_TOOL\_OPTIONS环境变量：  
变量名：JAVA\_TOOL\_OPTIONS  
变量值：-Dfile.encoding=UTF8  
  
11、从cmd打开dos窗口，输入javac和java确定是否配置成功，出现下面界面说明jdk是ok的。



# 4、Maven安装配置

1、安装版本：apache-maven-3.5.0

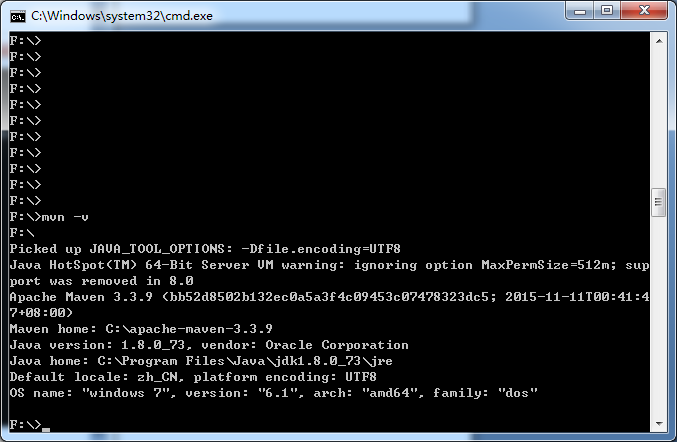
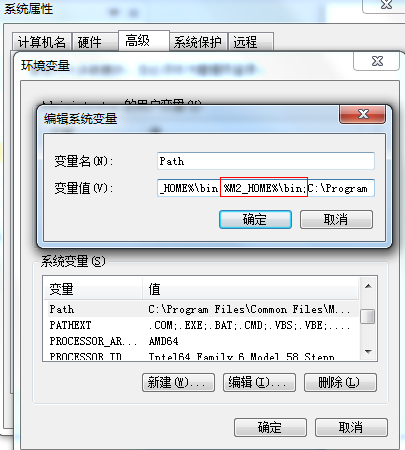
2、安装包已经放于附件中。

3、找到apache-maven-3.5.0-bin.tar.gz，解压放在

C:\apache-maven-3.5.0

4、配置环境变量，新增M2\_HOME和MAVEN\_OPTS系统环境变量  
M2\_HOME：C:\apache-maven-3.5.0  
MAVEN\_OPTS：-Xms256m -Xmx1024m  
PATH：%M2\_HOME%\bin;

5、在path里加入maven的bin路径  
  
6、运行cmd，打开dos命令窗口，输入mvn -v命令，出现下面界面说明ok。



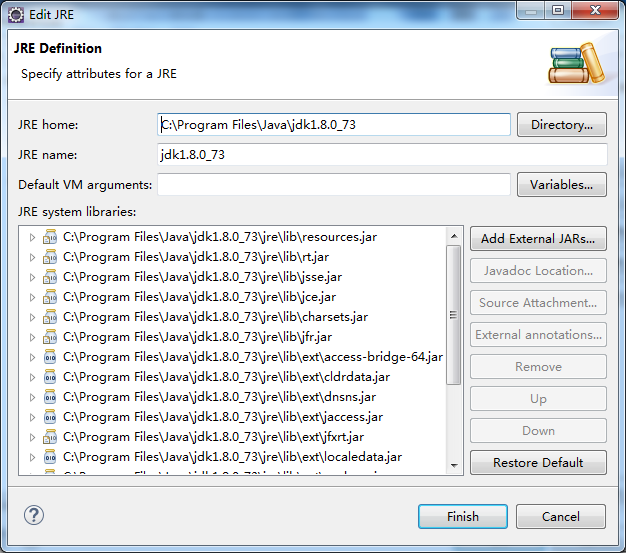
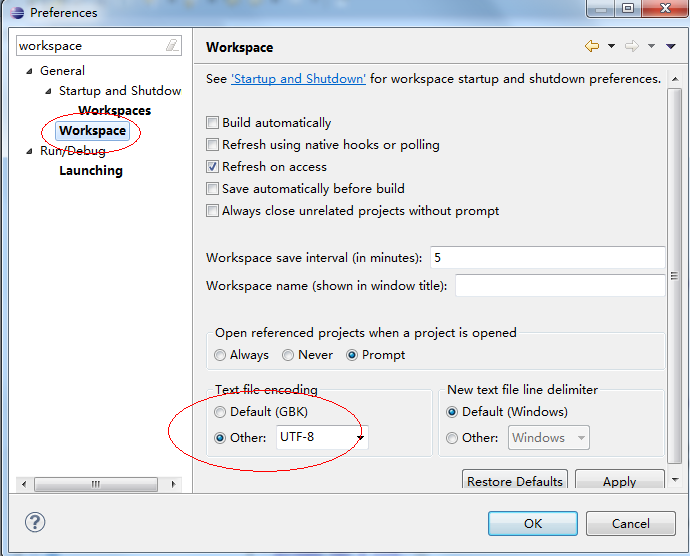
# 5、Eclipse安装配置

1、安装版本：eclipse-jee-mars-2-win32-x86\_64

2、安装包已放于附件中。

3、解压到本地目录下（任意目录都可以），运行eclipse.exe，启动eclipse

4、配置eclipse，有时候Windows默认为GBK，需要设置工作区编码为UTF-8：  
从Window=>Preferences=>General=>Workspace  
  
5、设置JDK环境，从Window=>Preferences，左侧输入jdk，选择Installed JREs，默认是jre，这里必须选择jdk，如下图所示



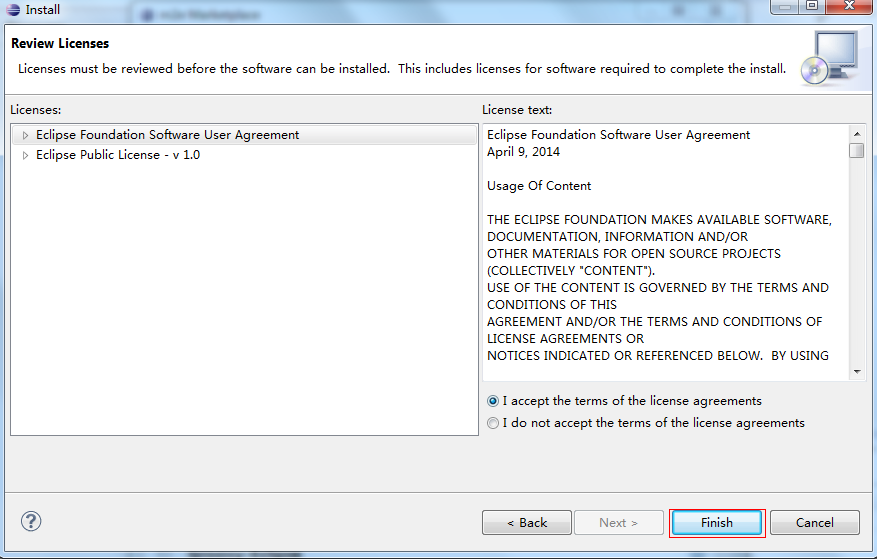
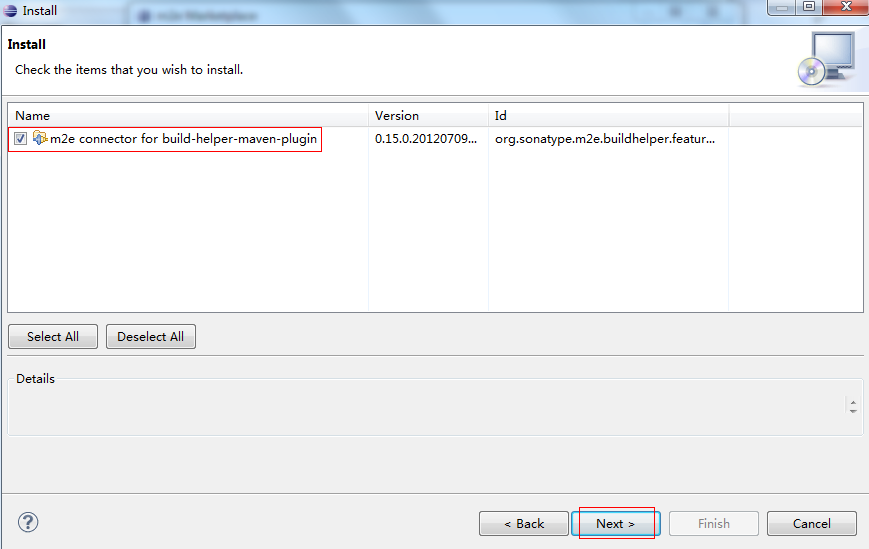
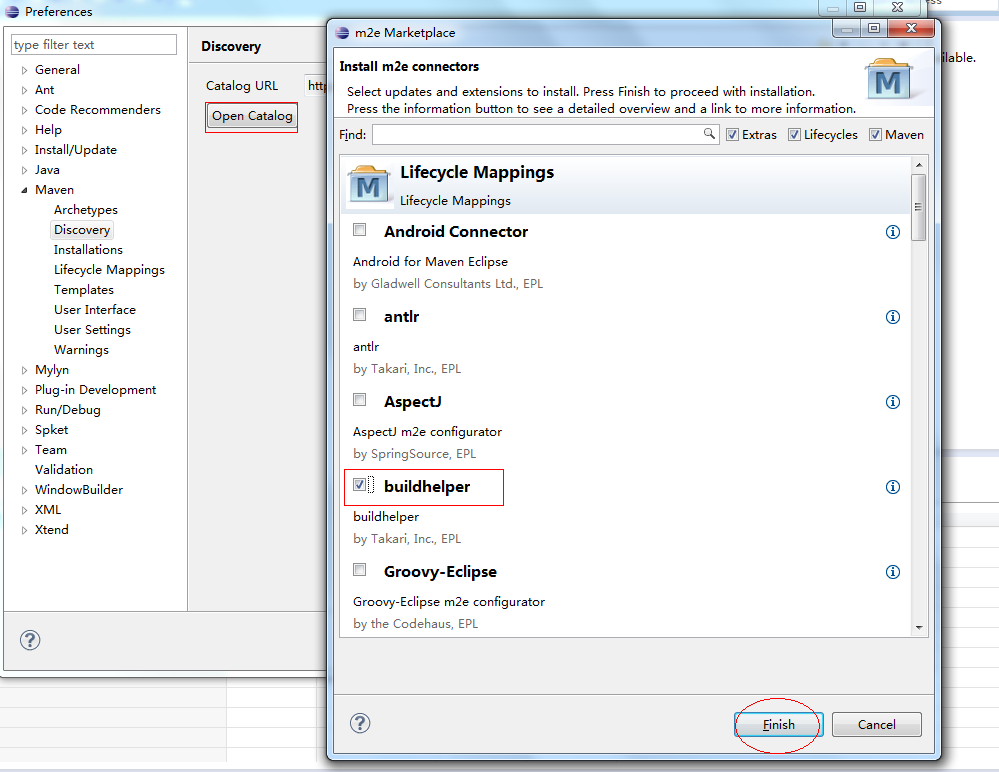
注意 Default VM arguments 要加上-Xms256m -Xmx1024m，否则编译时，会出现这样的错误：

Error occurred during initialization of VM

Initial heap size set to a larger value than the maximum heap size

在Eclipse里选：Window->Preference->Installed JREs->Edit(选中jre)，

在Default VM Arguments里输入-Xms256m -Xmx1024m，表示最小内存256M，最大1G，然后运行就可以了。  
6、安装build-helper-maven：  
a)Windows => Preferences => Maven =>Discovery=>Open Catalog =>buildhelper  
  
  
  
7、安装成功后，会提示重启eclipse。



# 6、Git安装配置

1、安装版本: Git-2.9.2-64-bit

2、安装包已放于附件中

3、双击Git-2.9.2-64-bit.exe进行安装

4、安装成功后，打开桌面上的Git Bash快捷方式

5、现在可以使用git命令下载github上的源码了

# 7、下载源码

下载odl源码有两种方式，一种是从gerrit上下载，一种是从github上下载。

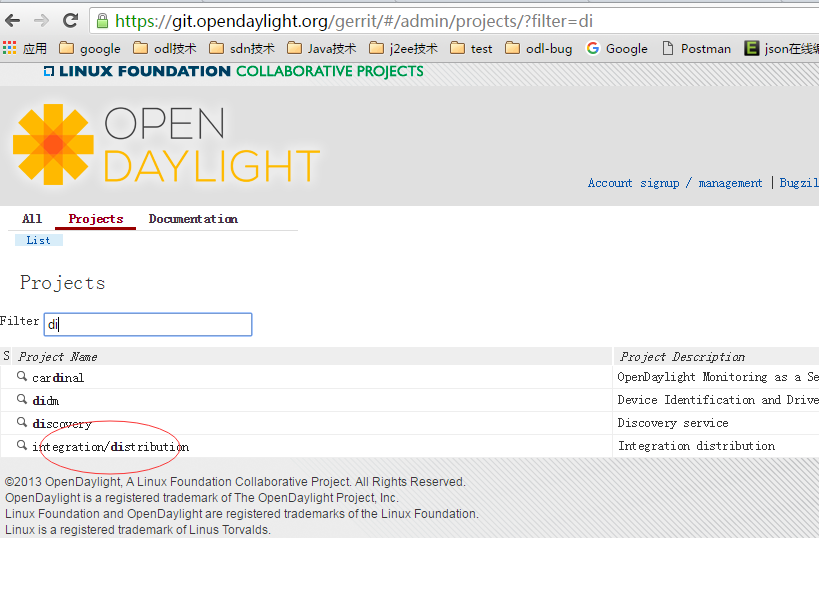
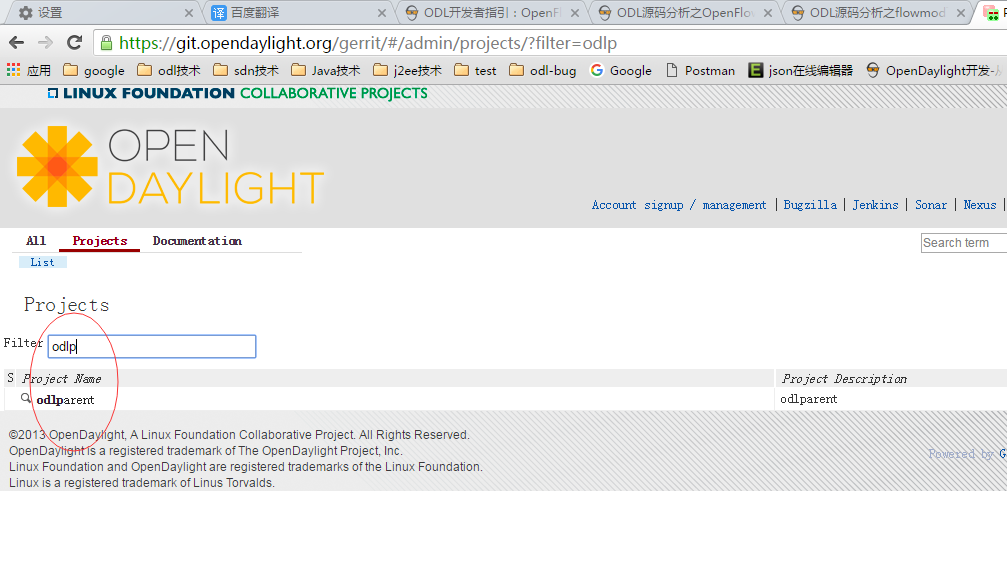
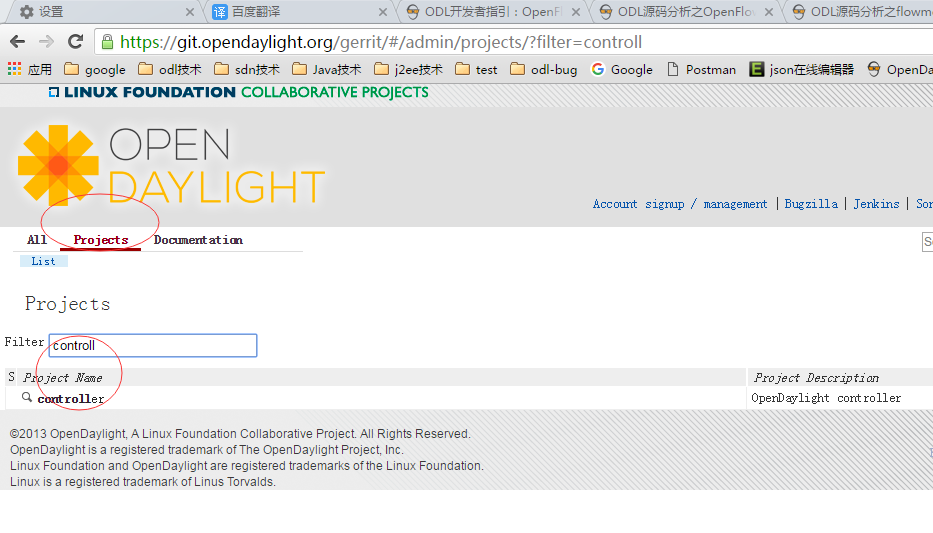
Controller是控制器源码，不包含其它组件；odlparent是odl-parent是所有opendaylight项目的父项目。Distribution为发行版，包含所有组件。

## 7.1默认版本下载

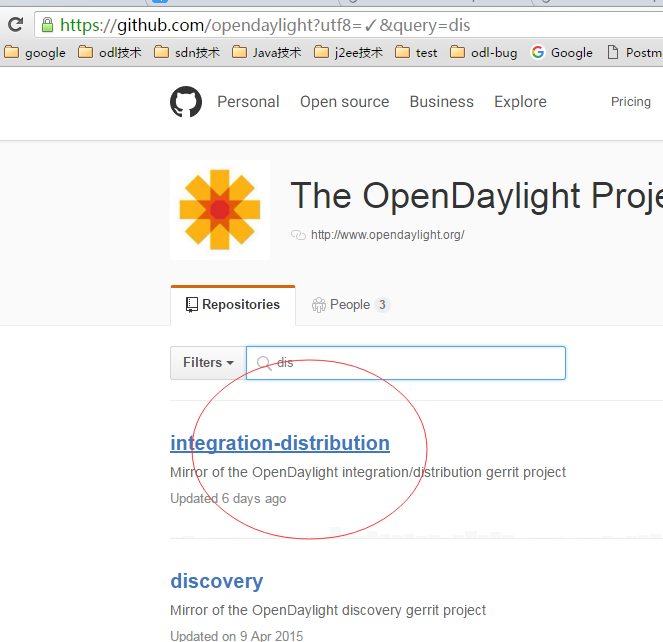
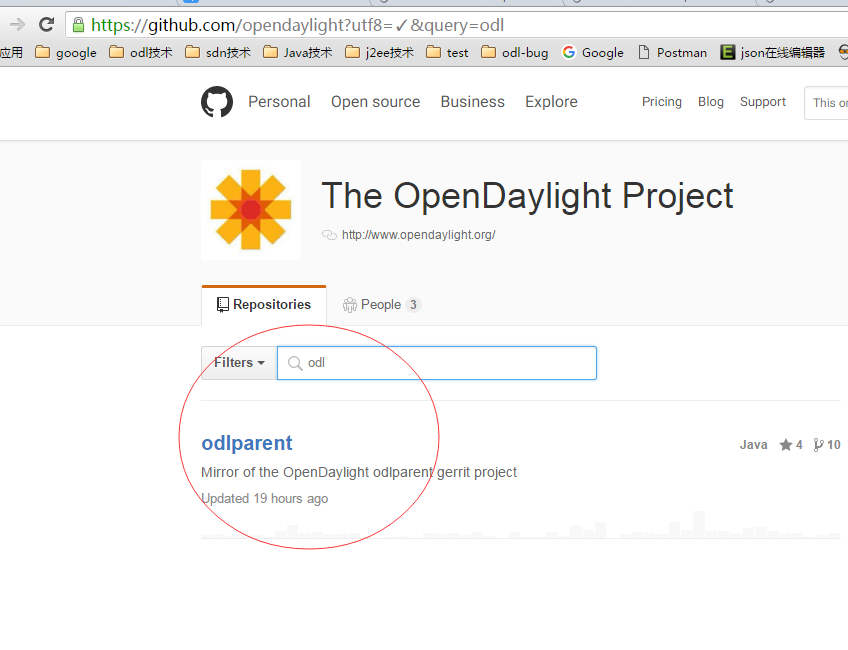
1、Gerrit方式—网页下载

gerrit是odl控制器源代码官方存储位置，在浏览器中输入[https://git.opendaylight.org](https://git.opendaylight.org/" \t "_blank)，点击Projects按

下载odl之后还需要下载odlparent源码  
  
继续下载distribution源码



2、github方式—网页下载  
odl控制器源码同时也会放在github上托管，浏览器输入https://github.com/opendaylight，找到controller  
  
同理，下载controller之后需要下载odlparent源码。  
  
继续下载distribution源码



3、github命令行方式

打开Git Bash，如下载然后输入发行版本

git clone <https://git.opendaylight.org/gerrit/p/integration/distribution.git>

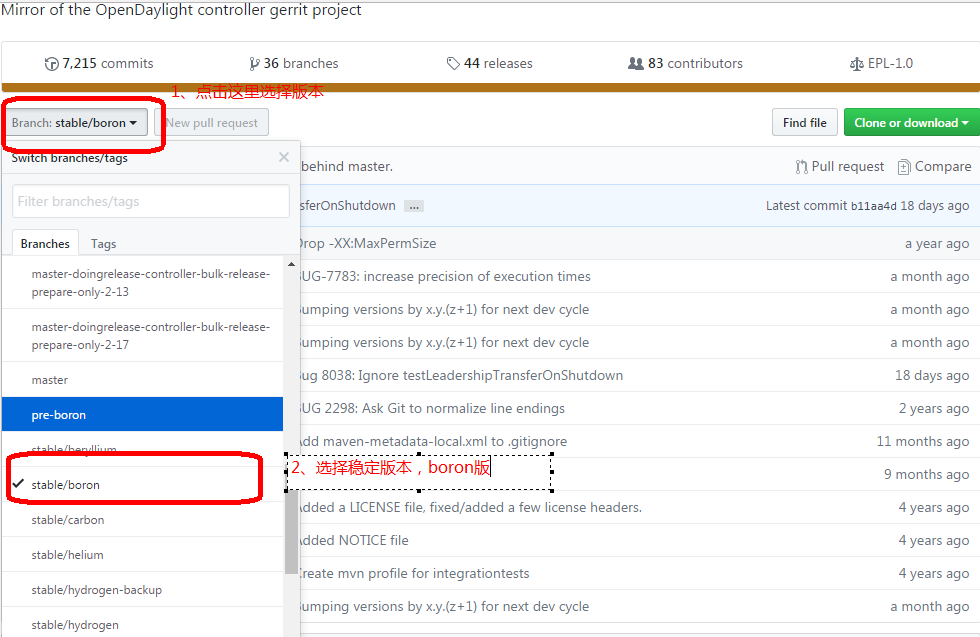


其它包类似。

## 7.2指定版本下载

1、网页指定版本

使用网页下载和上述方法一致，但在下载前要选择版本:



2、命令行指定版本

上一节下载后，切换到目录下，执行

git checkout stable/boron

stable/ 后面跟版本名称，就可以指定版本了。

# 编译源码

## 8.1编译前准备

1、准备仓库

在安装好maven之后，maven会在制定目录创建一个repository本地仓库，以后编译出来的jar包会自动安装到该仓库中，比如默认的maven本地仓库为C:\Users\byroot\.m2\repository。

把setting.zip压缩包的文件拷到C:\Users\byroot\.m2 目录下。

这里以distribution为例，编译好后运行。

2、下载源码并指定版本

a.下载distribution源代码

如果在第7章已经下载了，直接解压了就可了，不需要执行以下步骤:

打开Git Bash，然后输入git clone <https://git.opendaylight.org/gerrit/p/integration/distribution.git>

b.指定版本:

git checkout stable/boron

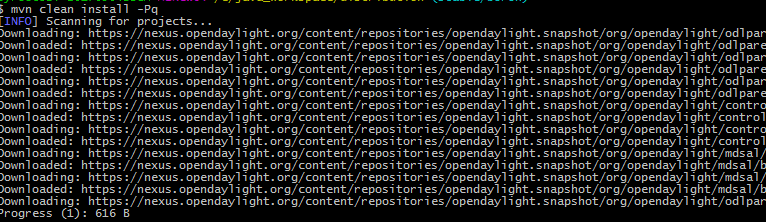
## 8.2命令行编译

编译的源码的方式可以分为两种，一种直接在命令行窗口下敲命令执行。另外一种是先配置好eclipse的编译命令，其实就是把第一种方法的编译命令放到eclipse的build配置中。

下面介绍第一种编译方法:

1、打开cmd，切换到distribution目录下

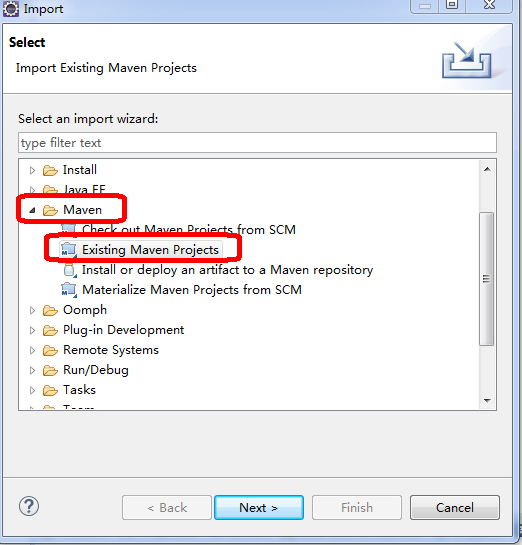
2、输入mvn clean install –Pq，就可以开始编译了。注意：编译时会从网上的库下载文件，有时会下载不成功，多运行几次命令就可以了。



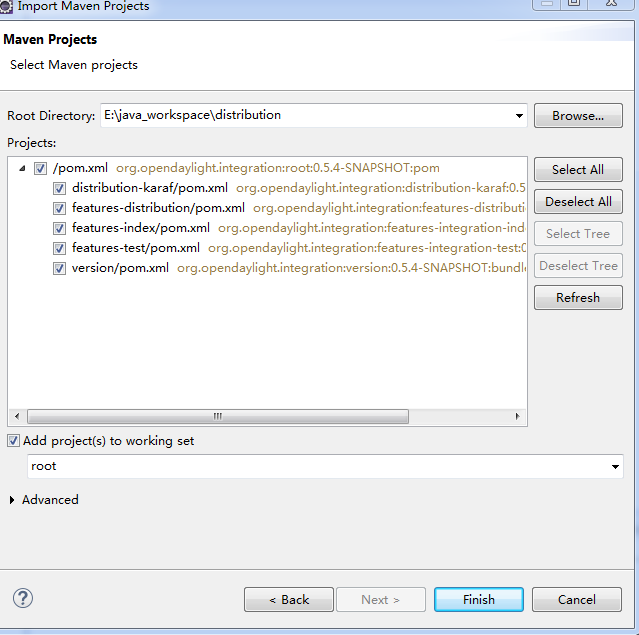
## 8.3 Eclipse编译

1、导入项目

File->Import



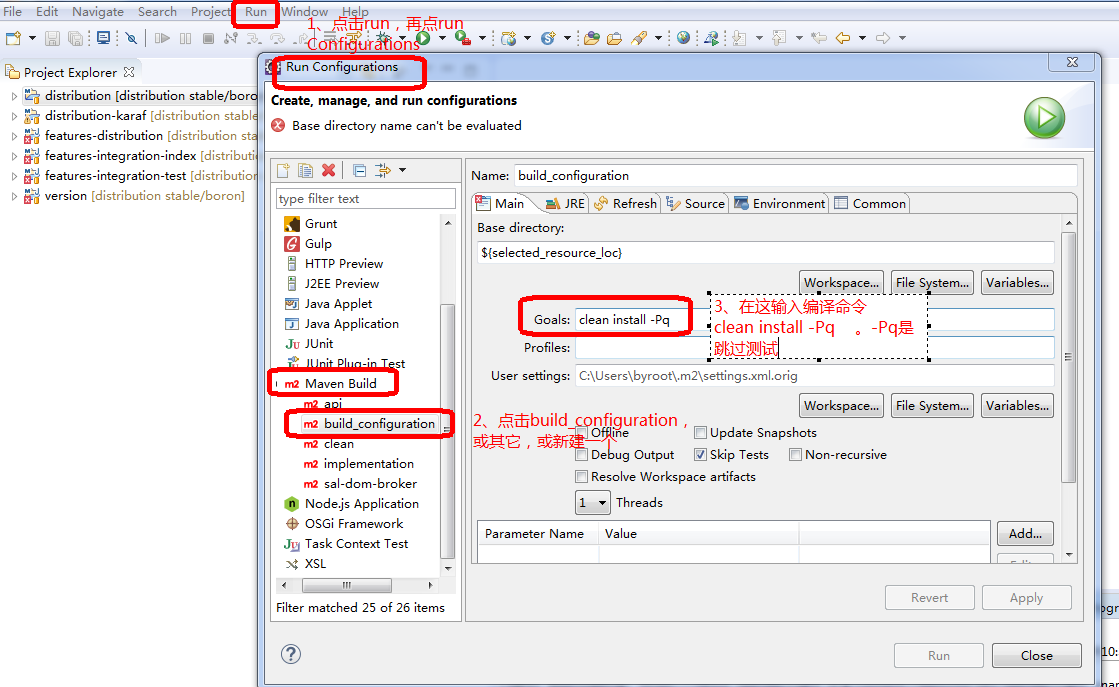
点maven再点Existing Maven Projects。



把项目文件夹导进来。

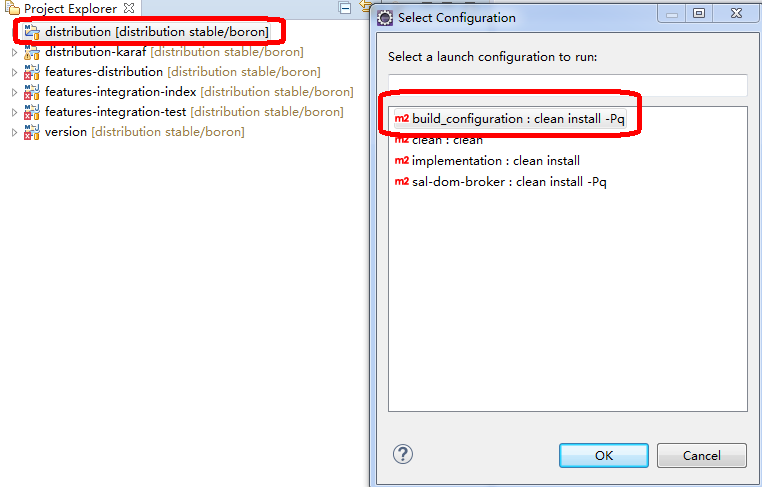
2、编译

配置编译项



编译:

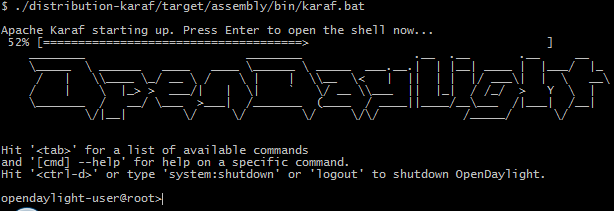
右键点主项目，点run as ，再点 Maven build



Console会出现编译过程，等待编译完成即可。注意：有时有些文件会下不来，但多执行几次就能编译成功了。

## 8.4运行测试

1、运行./distribution-karaf/target/assembly/bin/karaf.bat



出现上图就表明运行成功了。

2、安装组件

输入下面命令，安装相应的组件:

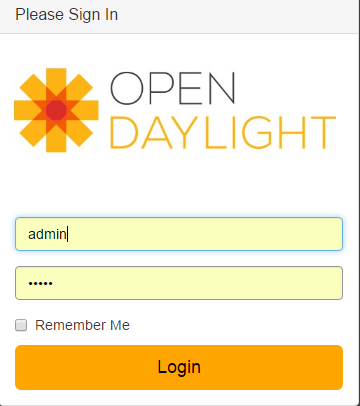
feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-core odl-dlux-node odl-dlux-yangui odl-dlux-yangvisualizer odl-dlux-yangman

3、用浏览器登录界面

打开chrmoe浏览器，输入

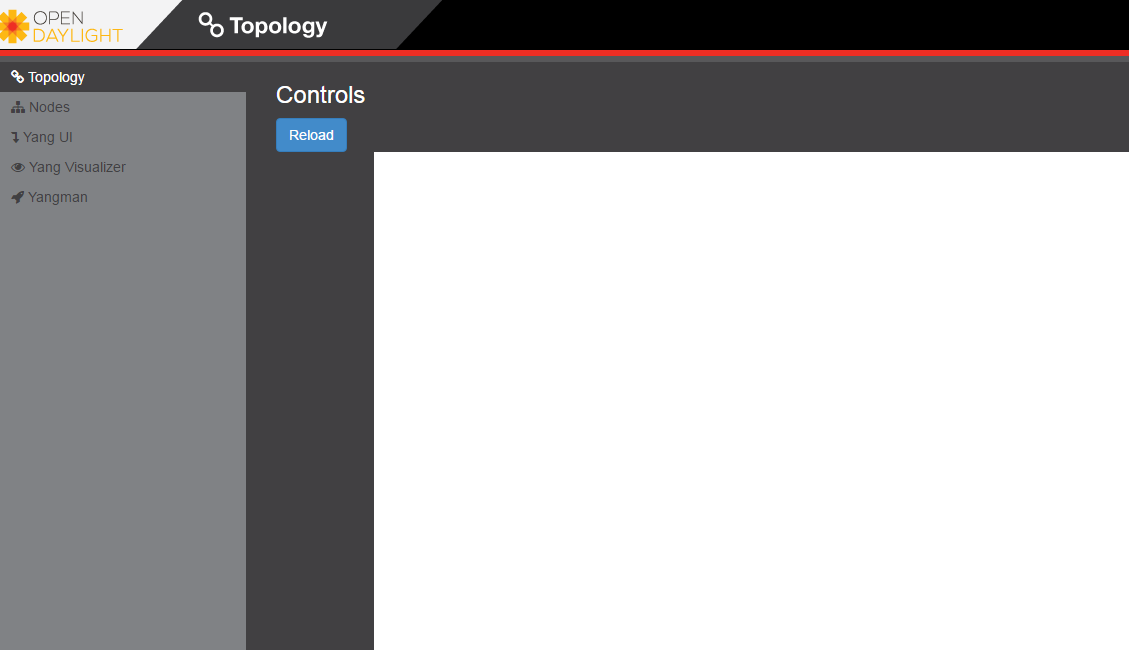
<http://192.168.10.57:8080/index.html#/login>

IP改为本地IP



用户名和密码都为admin.

登录后出现下图的界面:



4、mininet连接ODL

a.运行虚拟机

解压mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip 压缩包，运行ovf文件，等待虚拟机完全导入完成。

b.连接ODL

虚拟机开机后，输入用户名和密码 mininet 、mininet。

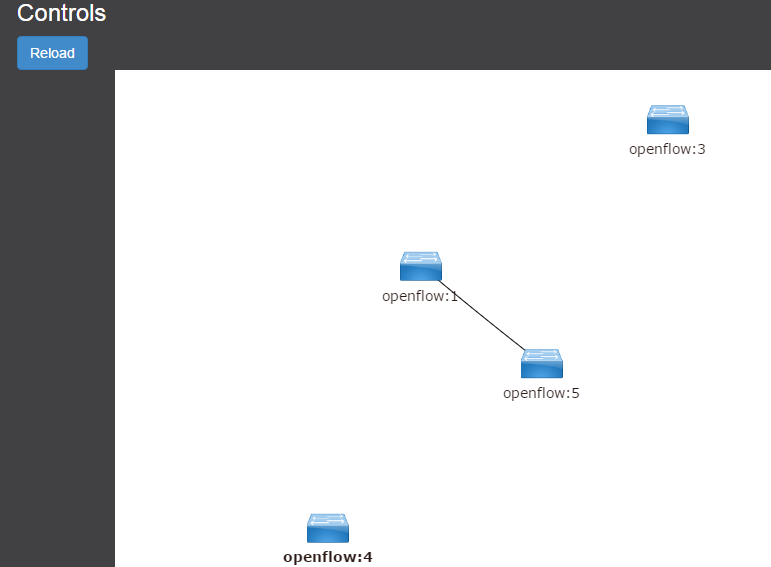
输入：

sudo mn --controller=remote,ip=192.168.0.101 --topo tree,3

IP为装有OpenDayLight的主机的IP，注意mininet虚拟机要和装OpenDayLight的主机的网段要一样。

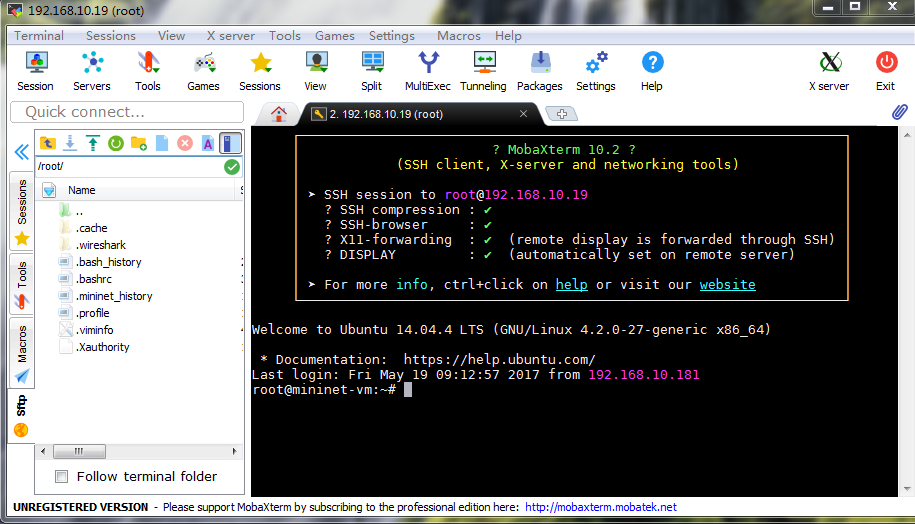
c.登录OpenDayLight

登录OpenDayLight的界面后，能看到类似的拓扑



6、可视化操作界面连接ODL

a.打开MobaXterm，并连接mininet虚拟机:



b.设置环境变量

输入

export DISPALY=192.168.10.181:0.0

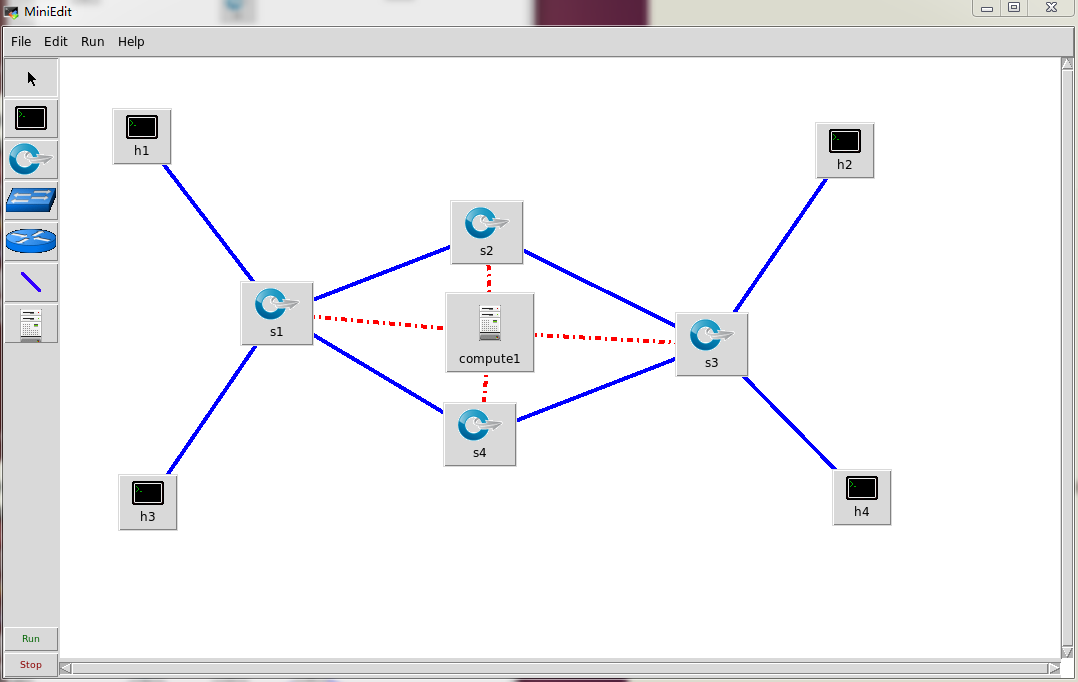
注意IP为你正在使用的Windows7的IP。

c.打开mininet

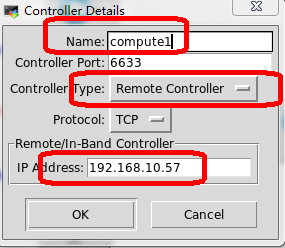
/home/mininet/mininet/examples/miniedit.py

d.画拓扑

如图，画出类似的拓扑



右键点控制器，再点properties，进入控制器的设置



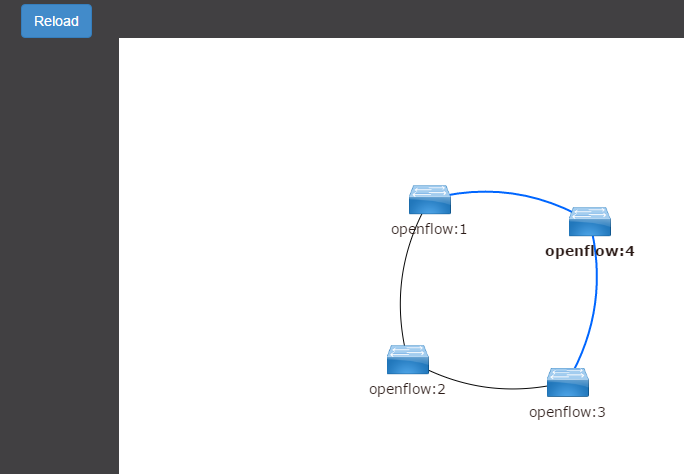
Name填上合适的名字，Controller Type选择Remote Controller，IP Address填上控制器的IP，至于端口一般都是66633.

e.运行

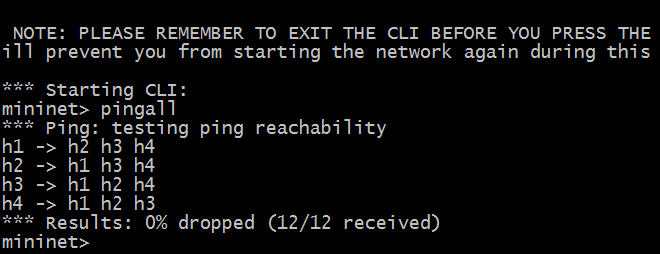
点击菜单栏 run ->run

f.ODL界面查看

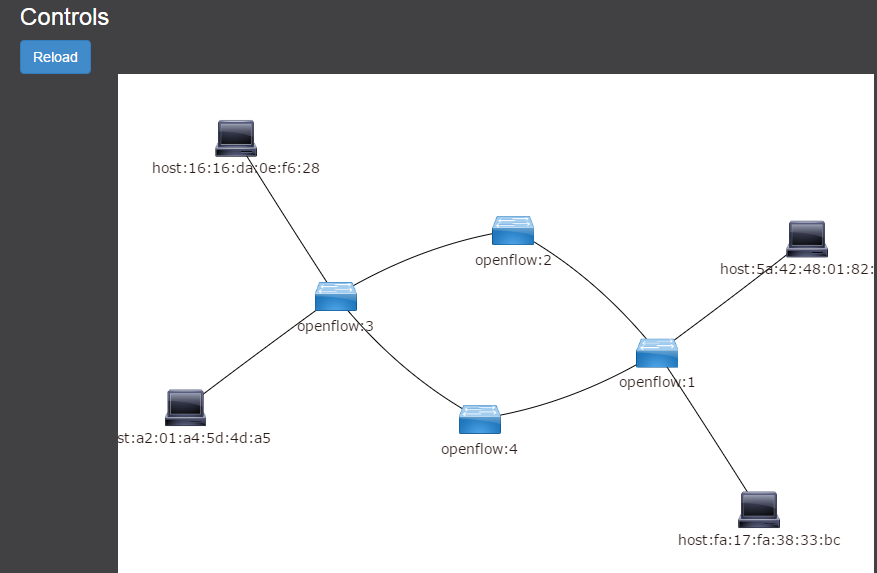
登录OpenDayLight的界面后，能看到类似的拓扑



更进一步，可以输入pingall等命令，发现整个网络了：



查看ODL的拓扑：



类似的要想使用哪台虚拟机，就在命令前面加上虚拟机名称就可以了，如

h1 ping

另外，右键点虚拟机，再点terminal就可以进入该虚拟机的命令行窗口了。

