第一章 开发环境配置

开发环境是我们日常工作的一个环境，不论你现在的公司是否会强制你使用统一的开发环境，对于希望通过阅读本书，跟着本书一起写项目的同学来说，我希望大家有统一的环境，这不仅有利于代码的一致性，同时也是为了减少大家在开发中遇到很多不必要的问题。为了能够保证我们接下来的开发环境保持一致性，我们这一章介绍一个能够在所有系统都保持一致性开发环境的工具Vagrant，通过该工具使得我们保持一致的Go环境、项目目录等。这一章第一小节介绍Vagrant的一些基础知识、第二小节介绍如何安装Vagrant以及如何配置，接着第三小节介绍如何使用Vagrant，对Vagrant的常用命令进行详细的介绍，第四小节基于Vagrant的环境进行Go环境的安装，最后会根据这些介绍提出一些练习供大家深入的理解。

1. Vagrant介绍
2. Vagrant安装配置
3. Vagrant使用入门
4. Go环境安装配置
5. 总结
6. 每章一练

links

- 目录
- 下一节: Vagrant介绍

1.1 Vagrant的介绍

虚拟开发环境

平常我们经常会遇到这样的问题：在开发机上面开发完毕程序，放到正式环境之后会出现各种奇怪的问题：描述符少了、nginx配置不正确、MySQL编码不对、php缺少模块、glibc版本太低等。

所以我们就需要虚拟开发环境，我们虚拟和正式环境一样的虚拟开发环境，而随着个人开发机硬件的升级，我们可以很容易的在本机跑虚拟机，例如VMware、VirtualBox等。因此使用虚拟化开发环境，在本机可以运行自己喜欢的OS（Windows、Ubuntu、Mac等），开发的程序运行在虚拟机中，这样迁移到生产环境可以避免环境不一致导致的莫名错误。

虚拟开发环境特别适合团队中开发环境、测试环境、正式环境不同的场合，这样就可以使得整个团队保持一致的环境，我写这一章的初衷就是为了让大家和我的开发环境保持一致，让读者和我们整个大团队保持一致的开发环境。

Vagrant

Vagrant就是为了方便的实现虚拟化环境而设计的，使用Ruby开发，基于VirtualBox等虚拟机管理软件的接口，提供了一个可配置、轻量级的便携式虚拟开发环境。使用Vagrant可以很方便的就建立起来一个虚拟环境，而且可以模拟多台虚拟机，这样我们平时还可以在开发机模拟分布式系统。

Vagrant还会创建一些共享文件夹，用来给你在主机和虚拟机之间共享代码用。这样就使得我们可以在主机上写程序，然后在虚拟机中运行。如此一来团队之间就可以共享相同的开发环境，就不会再出现类似“只有你的环境才会出现的bug”这样的事情。

团队新员工加入，常常会遇到花一天甚至更多时间来从头搭建完整的开发环境，而有了Vagrant，只需要直接将已经打包好的package（里面包括开发工具，代码库，配置好的服务器等）拿过来就可以工作了，这对于提升工作效率非常有帮助。

Vagrant不仅可以用来作为个人的虚拟开发环境工具，而且特别适合团队使用，它使得我们虚拟化环境变得如此的简单，只要一个简单的命令就可以开启虚拟之路。

links

- 目录
- 上一节: 开发环境配置
- 下一节: Vagrant安装配置

1.2 Vagrant安装配置

实际上Vagrant只是一个让你可以方便设置你想要的虚拟机的便携式工具，它底层支持VirtualBox、VMware甚至AWS作为虚拟机系统，本书中我们将使用VirtualBox来进行说明，所以第一步需要先安装Vagrant和VirtualBox。

VirtualBox安装

VirtualBox是Oracle开源的虚拟化系统，它支持多个平台，所以你可以到官方网站：https://www.virtualbox.org/wiki/Downloads/ 下载适合你平台的VirtualBox最新版本并安装，它的安装过程都很傻瓜化，一步一步执行就可以完成安装了。

Vagrant安装

最新版本的Vagrant已经无法通过`gem`命令来安装，因为依赖库太多了，所以目前无法使用`gem`来安装，目前网络上面很多教程还是类似这样的命令，那些都是错误的。目前唯一安装的办法就是到官方网站下载打包好的安装包：http://www.vagrantup.com/downloads.html 他的安装过程和VirtualBox的安装一样都是傻瓜化安装，一步一步执行就可以完成安装。

> *尽量下载最新的程序，因为VirtualBox经常升级，升级后有些接口会变化，老的Vagrant可能无法使用。*

要想检测安装是否成功，可以打开终端命令行工具，输入`vagrant`，看看程序是不是已经可以运行了。如果不行，请检查一下$PATH里面是否包含`vagrant`所在的路径。

Vagrant配置

当我们安装好VirtualBox和Vagrant后，我们要开始考虑在VM上使用什么操作系统了，一个打包好的操作系统在Vagrant中称为JBox，即Box是一个打包好的操作系统环境，目前网络上什么都有，所以你不用自

己去制作操作系统或者制作Box：vagrantbox.es上面有大家熟知的大多数操作系统，你只需要下载就可以了，下载主要是为了安装的时候快速，当然Vagrant也支持在线安装。

建立开发环境目录

我的开发机是Mac，所以我建立了如下的开发环境目录，读者可以根据自己的系统不同建立一个目录就可以：

```
/Users/astaxie/vagrant
```

下载box

前面讲了box是一个操作系统环境，实际上它是一个zip包，包含了Vagrant的配置信息和VirtualBox的虚拟机镜像文件.我们这一次的实战使用官方提供了一个box:Ubuntu lucid 64
http://files.vagrantup.com/lucid64.box

当然你也可以选一个自己团队在用的系统，例如CentOS、Debian等，我们可以通过上面说的地址下载开源爱好者们制作好的box。当然你自己做一个也行，下一节我会讲述如何自己制作包。

添加box

添加box的命令如下：

```
vagrant box add base 远端的box地址或者本地的box文件名
```

`vagrant box add` 是添加box的命令

`base` 是box的名称，可以是任意的字符串，`base` 是默认名称，主要用来标识一下你添加的box，后面的命令都是基于这个标识来操作的。

例子：

```
vagrant box add base http://files.vagrantup.com/lucid64.box
vagrant box add base https://dl.dropbox.com/u/7225008/Vagrant/CentOS-6.3-
x86_64-minimal.box
vagrant box add base CentOS-6.3-x86_64-minimal.box
vagrant box add "CentOS 6.3 x86_64 minimal" CentOS-6.3-x86_64-minimal.box
```

我在开发机上面是这样操作的，首先进入我们的开发环境目录`/Users/astaxie/vagrant`，执行如下的命令

```
vagrant box add base lucid64.box
```

安装过程的信息：

```
Downloading or copying the box...
Extracting box...te: 47.5M/s, Estimated time remaining: --:--:--)
Successfully added box 'base' with provider 'virtualbox'!
```

box中的镜像文件被放到了：`/Users/astaxie/.vagrant.d/boxes/`，如果在window系统中应该是放到了：
`C:\Users\当前用户名\.vagrant.d\boxes\`目录下。

通过`vagrant box add`这样的方式安装远程的box，可能很慢，所以建议大家先下载box到本地再执行这样的操作。

初始化

初始化的命令如下：

```
vagrant init
```

如果你添加的box名称不是base，那么需要在初始化的时候指定名称，例如

```
vagrant init "CentOS 6.3 x86_64 minimal"
```

初始化过程的信息：

```
A `Vagrantfile` has been placed in this directory.
You are now ready to `vagrant up` your first virtual environment!
Please read the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

这样就会在当前目录生成一个 `Vagrantfile` 的文件，里面有很多配置信息，后面我们会详细讲解每一项的含义，但是默认的配置就可以开箱即用。

启动虚拟机

启动虚拟机的命令如下：

```
vagrant up
```

启动过程的信息:

```
Bringing machine 'default' up with 'virtualbox' provider...
[default] Importing base box 'base'...
[default] Matching MAC address for NAT networking...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
[default] Mounting shared folders...
[default] -- /vagrant
```

连接到虚拟机

上面已经启动了虚拟机，之后我们就可以通过ssh来连接到虚拟机了。比如在我的开发机中可以像这样来连接：

```
vagrant ssh
```

连接到虚拟机后的信息如下：

```
Linux lucid64 2.6.32-38-server #83-Ubuntu SMP Wed Jan 4 11:26:59 UTC 2012
x86_64 GNU/Linux
Ubuntu 10.04.4 LTS

Welcome to the Ubuntu Server!
 * Documentation:  http://www.ubuntu.com/server/doc
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 07:31:39 2012 from 10.0.2.2
```

这样我们就可以像连接到一台服务器一样进行操作了。

window机器不支持这样的命令，必须使用第三方客户端来进行连接，例如putty、Xshell4等.

putty为例：

主机地址: 127.0.0.1

端口: 2222

用户名: vagrant

密码: vagrant

系统信息

进入系统之后我们可以看一下系统的基础信息：

```
vagrant@lucid64:/vagrant$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/lucid64-root
                       78G  945M   73G   2% /
none                  179M  176K  179M   1% /dev
none                  184M     0  184M   0% /dev/shm
none                  184M   64K  184M   1% /var/run
none                  184M     0  184M   0% /var/lock
none                  184M     0  184M   0% /lib/init/rw
none                   78G  945M   73G   2% /var/lib/ureadahead/debugfs
/dev/sda1             228M   17M  199M   8% /boot
/vagrant              298G   76G  222G  26% /vagrant
```

`/vagrant` 这个目录是自动映射的，被映射到 `/Users/astaxie/vagrant` ，这样就方便我们以后在开发机中进行开发，在虚拟机中进行运行效果测试了。

Vagrantfile配置文件详解

在我们的开发目录下有一个文件 `Vagrantfile` ，里面包含有大量的配置信息，主要包括三个方面的配置，虚拟机的配置、SSH配置、Vagrant的一些基础配置。Vagrant是使用Ruby开发的，所以它的配置语法也是Ruby的，但是我们没有学过Ruby的人还是可以跟着它的注释知道怎么配置一些基本项的配置。

1. box设置

```
  config.vm.box = "base"
```

上面这配置展示了Vagrant要去启用那个box作为系统，也就是上面我们输入 `vagrant init Box名称` 时所指定的box，如果没有输入box名称的話，那么默认就是 `base` ，VirtualBox提供了VBoxManage这个命令行工具，可以让我们设定VM，用 `modifyvm` 这个命令让我们可以设定VM的名称和内存大小等等，这里说的名称指的是在VirtualBox中显示的名称，我们也可以在Vagrantfile中进行设定，在Vagrantfile中加入如下这行就可以设定了：

```
 config.vm.provider "virtualbox" do |v|
  v.customize ["modifyvm", :id, "--name", "astaxie", "--memory",
"512"]
 end
```

这行设置的意思是调用VBoxManage的 `modifyvm` 的命令，设置VM的名称为 `astaxie` ，内存为512MB。你可以类似的通过定制其它VM属性来定制你自己的VM。

2. 网络设置

Vagrant有两种方式来进行网络连接，一种是host-only(主机模式)，意思是主机和虚拟机之间的网络互访，而不是虚拟机访问internet的技术，也就是只有你一個人自High，其他人访问不到你的虚拟机。另一种是Bridge(桥接模式)，该模式下的VM就像是局域网中的一台独立的主机，也就是说需要VM到你的路由器要IP，这样的话局域网里面其他机器就可以访问它了，一般我们设置虚拟机都是自high为主，所以我们的设置一般如下：

```
 config.vm.network :private_network, ip: "11.11.11.11"
```

这里我们虚拟机设置为hostonly，并且指定了一个IP，IP的话建议最好不要用 `192.168..` 这个网段，因为很有可能和你局域网里面的其它机器IP冲突，所以最好使用类似 `11.11..` 这样的IP地址。

3. hostname设置

`hostname` 的设置非常简单，Vagrantfile中加入下面这行就可以了：

```
 config.vm.hostname = "go-app"
```

设置 `hostname` 非常重要，因为当我们有很多台虚拟服务器的时候，都是依靠 `hostname` 来做识别的，例如Puppet或是Chef，都是通过 `hostname` 来做识别的，既然设置那么简单，所以我们就别偷懒，设置一个。

4. 同步目录

我们上面介绍过 `/vagrant` 目录默认就是当前的开发目录，这是在虚拟机开启的时候默认挂载同步的。我们还可以通过配置来设置额外的同步目录：

```
config.vm.synced_folder "/Users/astaxie/data", "/vagrant_data"
```

上面这个设定，第一个参数是主机的目录，第二个参数是虚拟机挂载的目录

5. 端口转发

```
config.vm.network :forwarded_port, guest: 80, host: 8080
```

上面这句配置可厉害了，这一行的意思是把对host机器上8080端口的访问请求forward到虚拟机的80端口的服务上，例如你在你的虚拟机上使用nginx跑了一个Go应用，那么你在host机器上的浏览器中打开`http://localhost:8080`时，Vagrant就会把这个请求转发到VM里面跑在80端口的nginx服务上，因此我们可以通过这个设置来帮助我们去设定host和VM之间，或是VM和VM之间的信息交互。

修改完Vagrantfile的配置后，记得要用`vagrant reload`命令来重启VM之后才能使用VM更新后的配置

links

- 目录
- 上一节: Vagrant的介绍
- 下一节: Vagrant使用入门

3 Vgrant使用入门

前面我们已经学会了如何安装并配置Vagrant，而且也已经按照默认的方式开启了，那么这一小节就给大家介绍一下Vagrant的高级应用。

Vagrant常用命令

前面讲了Vagrant的几个命令：

- `vagrant box add` 添加box的操作
- `vagrant init` 初始化box的操作
- `vagrant up` 启动虚拟机的操作
- `vagrant ssh` 登录虚拟机的操作

Vagrant还包括如下一些操作：

- `vagrant box list`

  显示当前已经添加的box列表

```
$ vagrant box list
base (virtualbox)
```

- `vagrant box remove`

  删除相应的box

```
$ vagrant box remove base virtualbox
Removing box 'base' with provider 'virtualbox'...
```

- `vagrant destroy`

  停止当前正在运行的虚拟机并销毁所有创建的资源

```
$ vagrant destroy
Are you sure you want to destroy the 'default' VM? [y/N] y
[default] Destroying VM and associated drives...
```

- `vagrant halt`

  关机

```
$ vagrant halt
[default] Attempting graceful shutdown of VM...
```

- `vagrant package`

  打包命令，可以把当前的运行的虚拟机环境进行打包

```
$ vagrant package
[default] Attempting graceful shutdown of VM...
[default] Clearing any previously set forwarded ports...
[default] Creating temporary directory for export...
[default] Exporting VM...
[default] Compressing package to: /Users/astaxie/vagrant/package.box
```

- `vagrant plugin`

  用于安装卸载插件

- `vagrant provision`

  通常情况下Box只做最基本的设置，而不是设置好所有的环境，因此Vagrant通常使用Chef或者Puppet来做进一步的环境搭建。那么Chef或者Puppet称为provisioning，而该命令就是指定开启相应的provisioning。按照Vagrant作者的说法，所谓的provisioning就是"The problem of installing software on a booted system"的意思。除了Chef和Puppet这些主流的配置管理工具之外，我们还可以使用Shell来编写安装脚本。

  例如： `vagrant provision --provision-with chef`

- `vagrant reload`

  重新启动虚拟机，主要用于重新载入配置文件

```
$ vagrant reload
[default] Attempting graceful shutdown of VM...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
```

```
[default] -- 22 => 2222 (adapter 1)
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
[default] Setting hostname...
[default] Mounting shared folders...
[default] -- /vagrant
```

- `vagrant resume`

  恢复前面被挂起的状态

```
$vagrant resume
[default] Resuming suspended VM...
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
```

- `vagrant ssh-config`

  输出用于ssh连接的一些信息

```
$vagrant ssh-config
Host default
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile "/Users/astaxie/.vagrant.d/insecure_private_key"
  IdentitiesOnly yes
  LogLevel FATAL
```

- `vagrant status`

  获取当前虚拟机的状态

```
$vagrant status
Current machine states:
```

```
default                   running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
```

- `vagrant suspend`

  挂起当前的虚拟机

```
$ vagrant suspend
[default] Saving VM state and suspending execution...
```

模拟打造多机器的分布式系统

前面这些单主机单虚拟机主要是用来自己做开发机，从这部分开始的内容主要将向大家介绍如何在单机上通过虚拟机来打造分布式造集群系统。这种多机器模式特别适合以下几种人：

1. 快速建立产品网络的多机器环境，例如web服务器、db服务器
2. 建立一个分布式系统，学习他们是如何交互的
3. 测试API和其他组件的通信
4. 容灾模拟，网络断网、机器死机、连接超时等情况

Vagrant支持单机模拟多台机器，而且支持一个配置文件Vagrntfile就可以跑分布式系统。

现在我们来建立多台VM跑起來，並且让他们之间能够相通信，假设一台是应用服务器、一台是DB服务器，那么这个结构在Vagrant中非常简单，其实和单台的配置差不多，你只需要通过 `config.vm.define` 来定义不同的角色就可以了，现在我们打开配置文件进行如下设置：

```
Vagrant.configure("2") do |config|
  config.vm.define :web do |web|
    web.vm.provider "virtualbox" do |v|
          v.customize ["modifyvm", :id, "--name", "web", "--memory", "512"]
    end
    web.vm.box = "base"
    web.vm.hostname = "web"
    web.vm.network :private_network, ip: "11.11.1.1"
  end
```

```
  config.vm.define :db do |db|
    db.vm.provider "virtualbox" do |v|
          v.customize ["modifyvm", :id, "--name", "db", "--memory", "512"]
    end
    db.vm.box = "base"
    db.vm.hostname = "db"
    db.vm.network :private_network, ip: "11.11.1.2"
  end
end
```

这里的设置和前面我们单机设置配置类似，只是我们使用了 `:web` 以及 `:db` 分别做了两个VM的设置，并且给每个VM设置了不同的 `hostname` 和IP，设置好之后再使用 `vagrant up` 将虚拟机跑起来：

```
$ vagrant up
Bringing machine 'web' up with 'virtualbox' provider...
Bringing machine 'db' up with 'virtualbox' provider...
[web] Setting the name of the VM...
[web] Clearing any previously set forwarded ports...
[web] Creating shared folders metadata...
[web] Clearing any previously set network interfaces...
[web] Preparing network interfaces based on configuration...
[web] Forwarding ports...
[web] -- 22 => 2222 (adapter 1)
[web] Running any VM customizations...
[web] Booting VM...
[web] Waiting for VM to boot. This can take a few minutes.
[web] VM booted and ready for use!
[web] Setting hostname...
[web] Configuring and enabling network interfaces...
[web] Mounting shared folders...
[web] -- /vagrant
[db] Setting the name of the VM...
[db] Clearing any previously set forwarded ports...
[db] Fixed port collision for 22 => 2222. Now on port 2200.
[db] Creating shared folders metadata...
[db] Clearing any previously set network interfaces...
[db] Preparing network interfaces based on configuration...
[db] Forwarding ports...
[db] -- 22 => 2200 (adapter 1)
[db] Running any VM customizations...
[db] Booting VM...
```

```
[db] Waiting for VM to boot. This can take a few minutes.
[db] VM booted and ready for use!
[db] Setting hostname...
[db] Configuring and enabling network interfaces...
[db] Mounting shared folders...
[db] -- /vagrant
```

看到上面的信息输出后，我们就可以通过`vagrant ssh`登录虚拟机了，但是这次和上次使用的不一样了，这次我们需要指定相应的角色，用来告诉ssh你期望连接的是哪一台：

```
$ vagrant ssh web
vagrant@web:~$

$ vagrant ssh db
vagrant@db:~$
```

是不是很酷！现在接下来我们再来验证一下虚拟机之间的通信，让我们先使用ssh登录web虚拟机，然后在web虚拟机上使用ssh登录db虚拟机(默认密码是`vagrant`)：

```
$ vagrant ssh web
Linux web 2.6.32-38-server #83-Ubuntu SMP Wed Jan 4 11:26:59 UTC 2012 x86_64
GNU/Linux
Ubuntu 10.04.4 LTS

Welcome to the Ubuntu Server!
 * Documentation:  http://www.ubuntu.com/server/doc
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Thu Aug  8 18:55:44 2013 from 10.0.2.2
vagrant@web:~$ ssh 11.11.1.2
The authenticity of host '11.11.1.2 (11.11.1.2)' can't be established.
RSA key fingerprint is e7:8f:07:57:69:08:6e:fa:82:bc:1c:f6:53:3f:12:9e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '11.11.1.2' (RSA) to the list of known hosts.
vagrant@11.11.1.2's password:
Linux db 2.6.32-38-server #83-Ubuntu SMP Wed Jan 4 11:26:59 UTC 2012 x86_64
GNU/Linux
Ubuntu 10.04.4 LTS
```

```
Welcome to the Ubuntu Server!
 * Documentation:  http://www.ubuntu.com/server/doc
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Thu Aug  8 18:58:50 2013 from 10.0.2.2
vagrant@db:~$
```

通过上面的信息我们可以看到虚拟机之间通信是畅通的，所以现在开始你伟大的架构设计吧，你想设计怎么样的架构都可以，唯一限制你的就是你主机的硬件配置了。

links

- 目录
- 上一节: Vagrant安装配置
- 下一节: Go环境安装配置

4 Go环境安装配置

前面我们已经安装好了虚拟环境，那么接下来就需要在虚拟环境中安装Go环境了，因此你首先要运行`vagrant up`把虚拟机开起来，然后通过`vagrant ssh`登录到系统中。

Ubuntu环境设置

我们需要修改配置文件`/etc/default/locale`设置为：

```
LANG="en_US.UTF-8"
LANGUAGE="en_US:en"
```

如果我们通过`vagrant ssh`登录终端，按tab键出现`bash: warning: setlocale: LC_CTYPE: cannot change locale (zh_CN.UTF-8)`的警告错误，解决方法如下：

编辑 /etc/profile，

```
sudo vi /etc/profile
```

在文件尾部添加一句：

```
export LC_ALL=C
```

通过上面的修改之后，退出系统，然后需要通过`vagrant reload`重启虚拟机.

安装

在Linux下安装Go很简单，只需要两个步骤：下载、解压。

首先下载Go的Linux编译好的包：

```
wget https://storage.googleapis.com/golang/go1.4.2.linux-amd64.tar.gz
```

然后解压就可以了：

```
sudo tar -C /usr/local -xzf go1.4.2.linux-amd64.tar.gz
```

我们现在就可以测试:

```
/usr/local/go/bin/go
```

如果出现如下的界面，那么说明已经安装成功了。

环境变量设置

上面我们已经成功安装了Go，那么接下来我们配置一些Go开发需要的信息：`GOPATH`的设置(关于GOPATH的概念请到https://github.com/astaxie/build-web-application-with-golang/blob/master/ebook/01.2.md)。

上面我们已经知道Vagrant启动之后，会默认把Vagrant这个目录挂载到系统的`/vagrant`目录，因此我们设置`GOPATH`到该目录：

```
$ cd
$ mkdir /vagrant/gopath/
$ vim .bashrc
```

切换到用户目录，打开bashrc进行设置，在最末尾增加如下两行：

```
export GOPATH=/vagrant/gopath
export PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
```

设置完毕之后，执行`source`命令使其生效：

```
$ source .bashrc
```

这个时候`GOPATH`设置成功，同时Go命令都已经加入了`PATH`，你在命令行下面执行如下：

```
$ go env
```

就会成功显示如下信息：

```
GOARCH="amd64"
GOBIN=""
GOCHAR="6"
GOEXE=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/vagrant/gopath/"
GORACE=""
GOROOT="/usr/local/go"
GOTOOLDIR="/usr/local/go/pkg/tool/linux_amd64"
CC="gcc"
GOGCCFLAGS="-g -O2 -fPIC -m64 -pthread"
CGO_ENABLED="1"
```

至此所有的Go环境配置完成，你和你的小伙伴们一定被如此简单的配置惊呆了吧。

links

- 目录
- 上一节: Vagrant使用入门
- 下一节: 总结

5 总结

这一章我们主要学习了如何使用Vgrant搭建虚拟环境，Vagrant是虚拟环境的核武器，第一小节详细的介绍了Vagrant的功能，然后介绍了如何安装配置Vagrant，接着介绍了Vagrant的高级应用，如何单机打造多服务器的过程，在搞定虚拟环境之后我们开始了Go的安装配置，安装就是解压，配置就是两句话，都是轻松搞定。但是我们利用了Vagrant的文件同步，可以在主机上修改，在虚拟机上面编译运行，通过这一章我们都了解了如何配置虚拟环境配置Go，使得我们的开发环境和参数都保持一致，为我们后面的实战代码开发做好了铺垫。

links

- 目录
- 上一节: Go环境安装配置
- 下一节: 每章一练

6 每章一练

1. 使用Vagrant搭建一个虚拟环境，并且设置网络为public
2. 启动Vagrant，并通过ssh客户端连接
3. Vagrant搭建一个分布式的系统，两台主web应用服务器，一台redis服务器，一台DB服务器
4. 参考gox库在Go环境下实现交叉编译
5. Go下载一个开源库：`go get github.com/astaxie/beego`，查看源码所在位置，了解其原理
6. 在Vagrant目录下新建一个文件`hello.go`，在host机器编辑此文件，然后在虚拟机中运行，了解开发步骤

links

- 目录
- 上一节: 总结
- 下一节: 第二章 开发工具配置