

# 在AWS上部署免费的Shiny应用

张丹



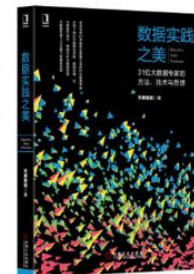
# 作者介绍

《R的极客理想》系列图书

**张丹**，《R的极客理想》系列图书作者，民生银行金融大数据分析师，前况客创始人兼CTO。

12年IT编程背景，4年量化投资经验，金融大数据专家。精通R, Java, Nodejs 编程语言，曾获得10项SUN及IBM技术认证。丰富的互联网应用开发架构经验，掌握大数据处理、数据挖掘机器学习等核心技术，熟悉金融二级市场、交易规则和投研体系。

博客<http://fens.me>，Alexa全球排名70k。



Shiny是R语言中一个**神级**的应用，唯一的缺点就是不支持并发。

我们很多时候都是做本地Shiny应用，用于展示各种报表的效果。但有时候也需要把报表上传到互联网上，其他人也能看到。这样就需要一个互联网的解决方案，刚好AWS有了免费的服务器支持。简直是完美！！

1. Shiny是什么？
2. 本地开发一个Shiny小应用
3. 申请AWS免费服务器
4. 在AWS上安装R语言环境
5. 在AWS上安装Shiny Server
6. 在AWS上部署自己的Shiny应用
7. 番外篇

# 1. Shiny是什么？

Shiny是RStudio公司开发的，一个用于R语言的Web应用程序框架，可以轻松开发交互式web应用，不需要了解HTML, CSS, JS等前端知识。

Shiny构建出应用的惊艳程度，远远超过了说明的文字。一定要学学，下面是一个Shiny小程序的截图。

# 1. Shiny是什么？

在AWS上部署免费的Shiny应用

6

## Here is a Shiny app

Shiny apps are easy to write. No web development skills are required.

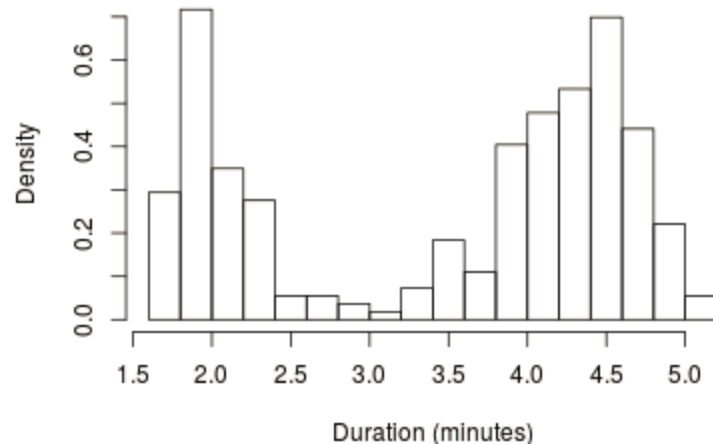
Number of bins in histogram (approximate):

20

☐ Show individual observations

☐ Show density estimate

Geyser eruption duration



ui.R

server.R

```
shinyUI(bootstrapPage(  
  selectInput(inputId = "n_breaks",  
    label = "Number of bins in histogram (approximate):",  
    choices = c(10, 20, 35, 50),  
    selected = 20),  
  
  checkboxInput(inputId = "individual_obs",  
    label = strong("Show individual observations"),  
    value = FALSE),  
  
  checkboxInput(inputId = "density",  
    label = strong("Show density estimate"),  
    value = FALSE),  
  
  plotOutput(outputId = "main_plot", height = "300px"),  
  
  # Display this only if the density is shown  
  conditionalPanel(condition = "input.density == true",  
    sliderInput(inputId = "bw_adjust",  
      label = "Bandwidth adjustment:",  
      min = 0.2, max = 2, value = 1, step = 0.2)  
  )  
))
```

# 1. Shiny是什么？

Shiny的主页：<http://shiny.rstudio.com/>

我们安装Shiny可以直接从CRAN获取，通过一行R程序就可以安装了。

```
~ R  
> install.packages("shiny")
```

1. Shiny是什么？
2. 本地开发一个Shiny小应用
3. 申请AWS免费服务器
4. 在AWS上安装R语言环境
5. 在AWS上安装Shiny Server
6. 在AWS上部署自己的Shiny应用



## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

9

下面我们用Shiny开发一个小应用的实例，主要是为介绍Shiny的用法，包括网页的界面UI和后端程序，数据源使用R语言自带的一个数据集。

数据集是faithful，统计的是美国黄石国家公园的泉水(Old Faithful geyser)喷发的持续时间和喷发等待时间。

## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

10

数据集包括2列

- eruptions为喷发持续时间
- waiting为喷发的等待时间。

```
> head(faithful, 20)
```

|    | eruptions | waiting |
|----|-----------|---------|
| 1  | 3.600     | 79      |
| 2  | 1.800     | 54      |
| 3  | 3.333     | 74      |
| 4  | 2.283     | 62      |
| 5  | 4.533     | 85      |
| 6  | 2.883     | 55      |
| 7  | 4.700     | 88      |
| 8  | 3.600     | 85      |
| 9  | 1.950     | 51      |
| 10 | 4.350     | 85      |
| 11 | 1.833     | 54      |
| 12 | 3.917     | 84      |
| 13 | 4.200     | 78      |
| 14 | 1.750     | 47      |
| 15 | 4.700     | 83      |
| 16 | 2.167     | 52      |
| 17 | 1.750     | 62      |
| 18 | 4.800     | 84      |
| 19 | 1.600     | 52      |
| 20 | 4.250     | 79      |

## 2. 本地开发一个Shiny小应用

Shiny应用，分为定义了客户端程序`ui.R`，和服务器端程序`server.R`，这2个文件默认要求放同一个目录中。

另外，我们还需要一个启动文件`run.R`，用于启动Shiny的应用。当然，如果在RStudio中开发，就不需要`run.R`的文件，直接点Shiny应用的启动按钮就行了。

## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

12

目录结构如下：

```
D:\workspace\dash
|—run.R
|—demo1
|   |—server.R
|   |—ui.R
```

## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

13

编辑文件：server.R

```
library(shiny)

shinyServer(function(input, output) {

  # 输出到UI的main_plot
  output$main_plot <- renderPlot({

    # 直方图
    hist(faithful$eruptions,
         probability = TRUE,
         breaks = as.numeric(input$n_breaks),
         xlab = "持续时间",
         main = "喷发持续时间")

    # 是否显示individual_obs
    if (input$individual_obs) {
      rug(faithful$eruptions)
    }

    # 是否显示conditionalPanel
    if (input$density) {
      dens <- density(faithful$eruptions, adjust = input$bw_adjust)
      lines(dens, col = "blue")
    }
  })
})
```

## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

14

编辑文件：ui.R

```
library(shiny)
shinyUI(bootstrapPage(
  #
  headerPanel("第一个Shiny应用"),
  #
  # 左侧布局
  sidebarPanel(
    #
    # 下拉框
    selectInput(inputId = "n_breaks", label = "直方图中的分隔数", choices = c(10, 20, 35, 50), selected =
    #
    # 单选框
    checkboxInput(inputId = "individual_obs", label = strong("实际观察点"), value = FALSE),
    #
    # 单选框
    checkboxInput(inputId = "density", label = strong("密度估计曲线"), value = FALSE)
  ),
  #
  # 主布局
  mainPanel(
    plotOutput(outputId = "main_plot", height = "300px"),
    #
    conditionalPanel(condition = "input.density == true",
      sliderInput(inputId = "bw_adjust", label = "带宽调整", min = 0.2, max = 2, value =
```

## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

15

编辑文件：run.R

```
library("shiny")  
runApp("./demo1", host='127.0.0.1', port=3840)
```

## 2. 本地开发一个Shiny小应用

启动Shiny应用时，本地的3840端口，就被打开了。

```
~ D:\workspace\dash>R -f run.r
R version 3.2.3 (2015-12-10) — "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R
'license()' 'licence()'
R.
'contributors()'
'citation()' RR
'demo()' 'help()'
'help.start()' HTML
'q()' R.

> library("shiny")
> runApp("./demo1", host='127.0.0.1', port=3840)

Listening on http://127.0.0.1:3840
```

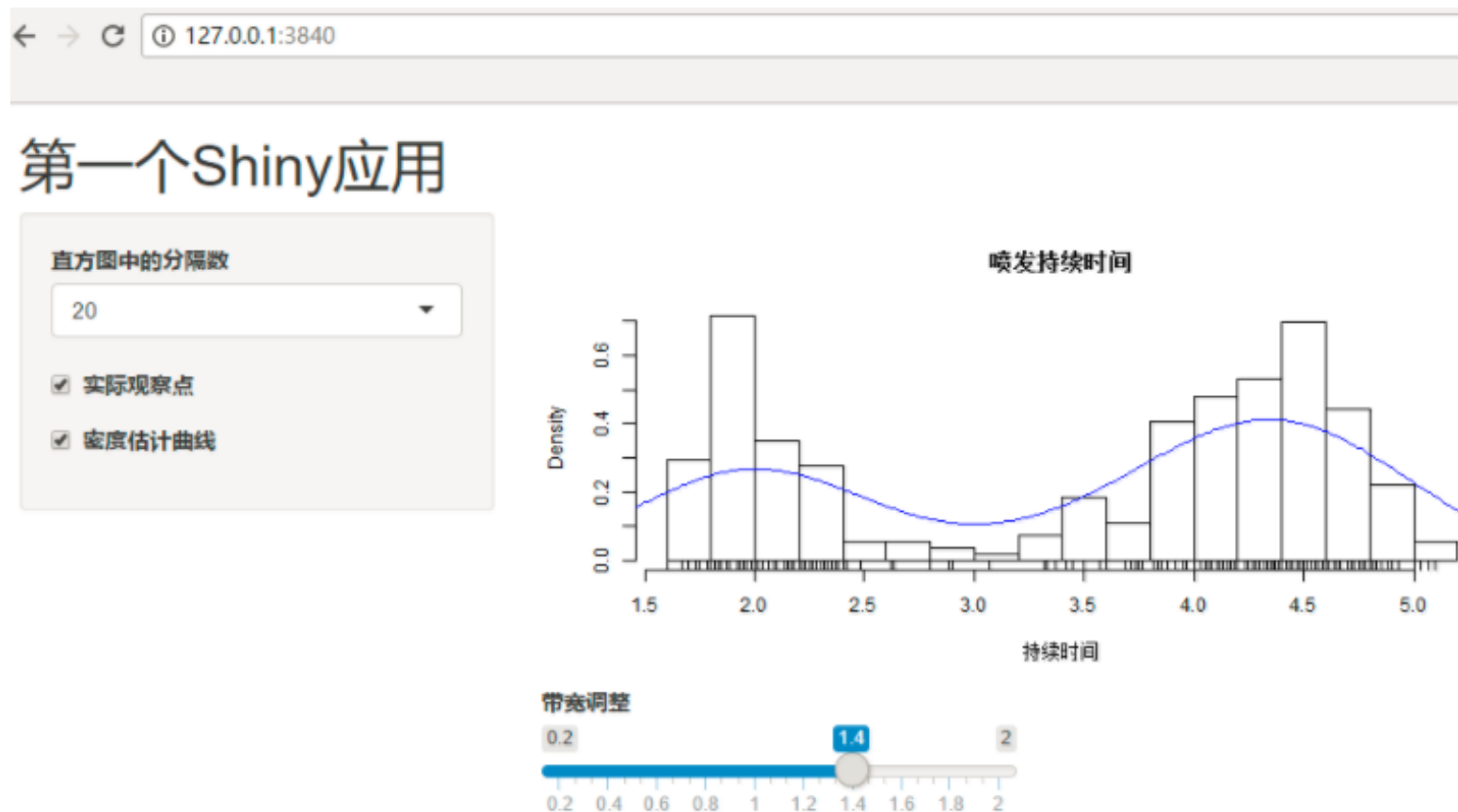


## 2. 本地开发一个Shiny小应用

在AWS上部署免费的Shiny应用

17

我们可以用浏览器，来访问本地的服务 `http://127.0.0.1:3840`。



1. Shiny是什么？
2. 本地开发一个Shiny小应用
3. 申请AWS免费服务器
4. 在AWS上安装R语言环境
5. 在AWS上安装Shiny Server
6. 在AWS上部署自己的Shiny应用

### 3.申请AWS免费服务器

AWS是Amazon提供的一个云服务平台，软件开发人员可以轻松购买计算、存储、数据库和其他基于 Internet 的服务来支持其应用程序。

**免费**的套餐，让互联网用户可以免费的使用他的资源，包括了服务器，数据库，CDN，负载均衡等服务。

首先，你需要**注册**一个AWS账号，然后登录进去，选择地区，申请免费的服务器。目前免费开放的区域不包括中国区，我选择了一个日本**东京**的服务器，Ubuntu Linux 64bit。

# 3.申请AWS免费服务器

1. 选择 AMI

2. 选择实例类型

3. 配置实例

4. 添加存储

5. 添加标签

6. 配置安全组

7. 审核

步骤 1: 选择一个Amazon 系统映像(AMI)

AWS Marketplace

社区 AMI

☐ 仅免费套餐

Amazon Linux

符合条件的免费套

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repository includes Docker, PHP, MySQL, PostgreSQL, and other packages.

根设备类型: ebs 虚拟化类型: hvm

SUSE Linux

符合条件的免费套

SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Services modules enabled.

根设备类型: ebs 虚拟化类型: hvm

Red Hat

符合条件的免费套

Red Hat Enterprise Linux version 7.3 (HVM), EBS General Purpose (SSD) Volume Type.

根设备类型: ebs 虚拟化类型: hvm

Ubuntu

符合条件的免费套

Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

根设备类型: ebs 虚拟化类型: hvm

Microsoft Windows

符合条件的免费套

Microsoft Windows 2016 Datacenter edition. [English]

根设备类型: ebs 虚拟化类型: hvm

美国东部 (弗吉尼亚北部)

美国东部 (俄亥俄)

美国西部 (加利福尼亚北部)

美国西部 (俄勒冈)

加拿大 (中部)

欧洲 (爱尔兰)

欧洲 (法兰克福)

欧洲 (伦敦)

亚太区域 (新加坡)

亚太区域 (悉尼)

亚太区域 (首尔)

亚太区域 (东京)

亚太区域 (孟买)

南美洲 (圣保罗)

64 位

选择

64 位

# 3.申请AWS免费服务器

在AWS上部署免费的Shiny应用

21

免费的资源，有一些限制，只能1核心CPU，1G内存，最大30G存储等。

服务 资源组

bsspirit 东京 支持

1. 选择AMI2. 选择实例类型3. 配置实例4. 添加存储5. 添加标签6. 配置安全组7. 审核

步骤 7: 核查实例启动

请查看您实例启动的详细信息。您可返回对每个部分进行编辑更改。单击启动，以为您的实例分配一个密钥对并完成启动过程。

提高实例的安全性。您的安全组 Tokyo1 向世界开放。

您的实例可以从任何 IP 地址访问。我们建议您更新您的安全组规则，只允许从已知的 IP 地址访问。

您也可以在安全组中打开其他端口，以便于访问您正在运行的应用程序或服务，例如 Web 服务器的 HTTP (80)。 [编辑安全组](#)

AMI 详细信息

编辑AMI

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-785c491f

符合条件的免费套餐

Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

根设备类型: ebs 虚拟化类型: hvm

实例类型

编辑实例类型

| 实例类型     | ECU | vCPU | 内存 (GiB) | 实例存储 (GB) | 可用的优化 EBS | 网络性能            |
|----------|-----|------|----------|-----------|-----------|-----------------|
| t2.micro | 变量  | 1    | 1        | 仅限于 EBS   | -         | Low to Moderate |

安全组

编辑安全组

| 安全组 ID      | 名称     | 描述                                                    |
|-------------|--------|-------------------------------------------------------|
| sg-1a329b7c | Tokyo1 | launch-wizard-1 created 2017-07-05T21:37:49.911+08:00 |

所有选定的安全组入站规则

| 类型 | 协议 | 端口范围 | 来源 |
|----|----|------|----|
|----|----|------|----|

# 3.申请AWS免费服务器

在AWS上部署免费的Shiny应用

22

大概等3分钟，服务器启动完成，然后就可以通过SSH进行访问了。

The screenshot displays the AWS Management Console interface for an EC2 instance. The left sidebar shows the navigation menu with categories like EC2 控制面板, 事件, 标签, 报告, 限制, 实例, 竞价请求, 预留实例, 专用主机, 映像, AMI, 捆绑任务, ELASTIC BLOCK STORE, 卷, 快照, 网络与安全, 安全组, 弹性 IP, 置放群组, 密钥对, 网络接口, 负载均衡, 负载均衡器, 目标群组, and AUTO SCALING. The main content area shows a list of EC2 instances with columns for Name, 实例 ID, 实例类型, 可用区, 实例状态, 状态检查, 警报 状态, 公有 DNS (IPv4), IPv4 公有 IP, and IPv6 IP. Two instances are listed: 'NewOne' and 'Tokyo1'. The 'NewOne' instance is highlighted with a red box, showing its details in the '实例: i-052ccf7fcbcd1a121 (NewOne)' section. The details include the public DNS and IPv4 address, which are also highlighted with a red box. The instance is a t2.micro type in the ap-northeast-1 region, running on the ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20170619.1 (ami-785c491f) AMI. The instance was started on 2017年7月6日 UTC+8上午11:26:31 (不到 1 小时).

| Name   | 实例 ID               | 实例类型     | 可用区             | 实例状态    | 状态检查        | 警报 状态 | 公有 DNS (IPv4)             | IPv4 公有 IP   | IPv6 IP |
|--------|---------------------|----------|-----------------|---------|-------------|-------|---------------------------|--------------|---------|
| NewOne | i-052ccf7fcbcd1a121 | t2.micro | ap-northeast-1a | running | 2/2 的检查已... | 无     | ec2-52-69-175-78.ap-no... | 52.69.175.78 | -       |
| Tokyo1 | i-09c704c5b8e902326 | t2.micro | ap-northeast-1a | running | 2/2 的检查已... | 无     | ec2-52-68-222-11.ap-no... | 52.68.222.11 | -       |

实例: i-052ccf7fcbcd1a121 (NewOne) 公有 DNS: ec2-52-69-175-78.ap-northeast-1.compute.amazonaws.com

| 属性     | 值                                                                                |
|--------|----------------------------------------------------------------------------------|
| 实例 ID  | i-052ccf7fcbcd1a121                                                              |
| 实例状态   | running                                                                          |
| 实例类型   | t2.micro                                                                         |
| 弹性 IP  | -                                                                                |
| 可用区    | ap-northeast-1a                                                                  |
| 安全组    | Tokyo1 查看规则                                                                      |
| 计划的事件  | 无计划事件                                                                            |
| AMI ID | ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20170619.1 (ami-785c491f) |
| 平台     | -                                                                                |
| IAM 角色 | -                                                                                |
| 密钥对名称  | Tokyo1                                                                           |
| 所有者    | 115839633000                                                                     |
| 启动时间   | 2017年7月6日 UTC+8上午11:26:31 (不到 1 小时)                                              |
| 终止保护   | False                                                                            |
| 生命周期   | normal                                                                           |
| 监控     | 基本                                                                               |

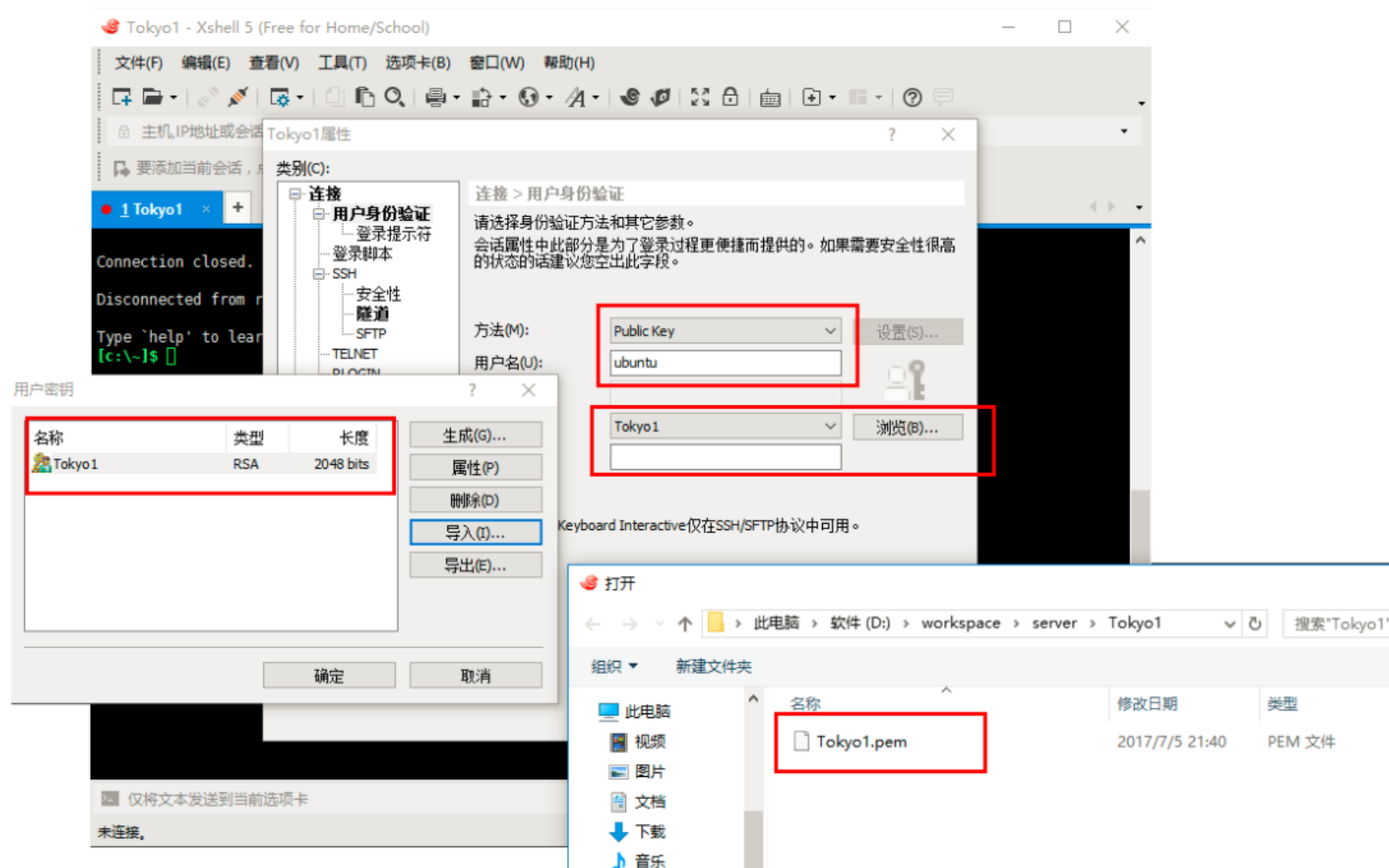
| 属性            | 值                                                     |
|---------------|-------------------------------------------------------|
| 公有 DNS (IPv4) | ec2-52-69-175-78.ap-northeast-1.compute.amazonaws.com |
| IPv4 公有 IP    | 52.69.175.78                                          |
| IPv6 IP       | -                                                     |
| 私有 DNS        | ip-172-31-31-236.ap-northeast-1.compute.internal      |
| 私有 IP         | 172.31.31.236                                         |
| 辅助私有 IP       | -                                                     |
| VPC ID        | vpc-c7c486a3                                          |
| 子网 ID         | subnet-7b4f520d                                       |
| 网络接口          | eth0                                                  |
| 源/目标检查        | True                                                  |
| EBS 优化        | False                                                 |
| 根设备类型         | ebs                                                   |
| 根设备           | /dev/sda1                                             |
| 块存储设备         | /dev/sda1                                             |

# 3.申请AWS免费服务器

在AWS上部署免费的Shiny应用

23

AWS为了保证安全性，建议使用秘钥访问，所以你需要创建一个秘钥对，下载一个xx.pem的私钥，然后配到Putty或XShell等用于远程登录的客户端里。



1. Shiny是什么？
2. 本地开发一个Shiny小应用
3. 申请AWS免费服务器
4. 在AWS上安装R语言环境
5. 在AWS上安装Shiny Server
6. 在AWS上部署自己的Shiny应用



## 4. 安装R语言环境

登录后，我们就可以安装R语言的环境了。安装过程比较简单，默认版本的R语言环境，直接使用是apt-get命令就是最方便的。

服务器所使用的系统环境

- Linux Ubuntu 16.04.2 LTS 64-bit
- R: 3.2.3 x86\_64-pc-linux-gnu (64-bit)

## 4. 安装R语言环境

我们先更新apt的软件源，安装必备的系统软件，包括r-base和git，以及的Shiny应用的依赖库libcurl4-openssl-dev，libxml2-dev。如果你忘了装了，后面再装也都不影响。

```
~ sudo apt-get update
~ sudo apt-get install r-base
~ sudo apt-get install git
~ sudo apt-get install libcurl4-openssl-dev
~ sudo apt-get install libxml2-dev
```

## 4. 安装R语言环境

接下来，让我们安装R语言的依赖包。

```
~ sudo -i # 切换到root用户
~ R      # 启动R语言环境
```

这里有一个小技巧，就是在R语言的环境中安装第三方R包，并切换到root用户。

我们需要预装的包，主要就是shiny，当然如果你还有依赖其他的包，需要一块安装。

安装时，R会让我们选择软件源，如果用https协议的镜像列表，你需要配置一下curl进行下载。你依然可以选择用http协议的镜像列表，选61之后，会出现http的镜像列表。

```
# 安装包
> install.packages("shiny")

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
— Please select a CRAN mirror for use in this session —
HTTPS CRAN mirror
```

1. Shiny是什么？
2. 本地开发一个Shiny小应用
3. 申请AWS免费服务器
4. 在AWS上安装R语言环境
5. 在AWS上安装Shiny Server
6. 在AWS上部署自己的Shiny应用

## 5.安装Shiny Server

在开发时，我们其实只是用到了shiny的R语言第三方包，可以在本地的开发环境，运行Shiny的程序。那么，如果把一个Shiny放到公司内网或外网给其他人用呢？这时就是需要Shiny Server了。

顺利安装完R的依赖包，接下来就是要安装Shiny Server了。Shiny Server是一个单独的软件，目前还不支持通过apt-get或R本身进行安装，需要下载安装。

## 5.安装Shiny Server

在AWS上部署免费的Shiny应用

31

Shiny Server提供一个稳定的Shiny应用在线的运行环境，Shiny Server分成开源版本和企业版本。开源版本，提供了基本的Shiny功能，数据、可视化、运行环境，对于个人来说已经足够用了，而且非常友好。企业版本，提供安全和管理功能添加到基本的开源版本中。**RStudio公司出品，必属精品！！**

Shiny Server是一个单独的软件，我们需要下载进行安装，下载地址：  
<https://www.rstudio.com/products/shiny/shiny-server/>

# 5.安装Shiny Server

在AWS上部署免费的Shiny应用

32

在Ubuntu的环境中，我们可以通过下面的命令，进行下载和安装。

```
~ sudo apt-get install gdebi-core
~ wget https://download3.rstudio.org/ubuntu-12.04/x86_64/shiny-server-1.5.3.838-amd64.deb
~ sudo gdebi shiny-server-1.5.3.838-amd64.deb
```

```
Reading package lists... D ● shiny-server.service - ShinyServer
```

```
Building dependency tree      Loaded: loaded (/etc/systemd/system/shiny-server.service; enabled; vendor preset: enabled)
```

```
Reading state information.    Active: active (running) since Thu 2017-07-06 07:40:24 UTC; 6ms ago
```

```
Reading state information.    Process: 25322 ExecStartPost=/bin/sleep 3 (code=exited, status=0/SUCCESS)
```

```
Reading state information.    Main PID: 25325 (shiny-server)
```

```
Tasks: 12
```

```
Memory: 35.4M
```

```
CPU: 411ms
```

```
CGroup: /system.slice/shiny-server.service
```

```
└─25321 /bin/bash -c /opt/shiny-server/bin/shiny-server --pidfile=/var/run/shiny-server.p
```

```
└─25325 /opt/shiny-server/ext/node/bin/shiny-server /opt/shiny-server/lib/main.js --pidfi
```



运行完安装的命令，默认情况Shiny Server会被直接启动起来，其中3838的端口被打开。

## 检查启动端口

```
~ netstat -nlpt
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)

```

| Proto | Recv-Q | Send-Q | Local Address | Foreign Address | State  | PID/Program name |
|-------|--------|--------|---------------|-----------------|--------|------------------|
| tcp   | 0      | 0      | 0.0.0.0:22    | 0.0.0.0:*       | LISTEN | -                |
| tcp   | 0      | 0      | 0.0.0.0:3838  | 0.0.0.0:*       | LISTEN | -                |
| tcp6  | 0      | 0      | :::22         | :::*            | LISTEN | -                |

# 5.安装Shiny Server

在AWS上部署免费的Shiny应用

34

我们可以通过浏览器，直接基于IP和端口进行Shiny Server的访问了。



The screenshot shows a web browser window with the address bar displaying "52.69.175.78:3838". The main content area has a blue header with the text "Welcome to Shiny Server!". Below this is a gray box with the message: "If you're seeing this page, that means Shiny Server is installed and running. **Congratulations!**".

**What's Next?**

Now you're ready to setup Shiny — if you haven't already — and start deploying your Shiny applications.

If you see a Shiny application running on the right side of this page, then Shiny is configured properly on your server and already running an example. Bravo! You can see this application on your server at </sample-apps/hello/>.

If you see a gray box or an error message, then there's a bit more work to do to get Shiny running fully. You can continue with [the installation instructions](#) or use [the Admin Guide](#) for more information. If you're seeing an error message in the panel to the right, you can use it to help diagnose what may be wrong. If you think Shiny is installed and setup properly and things still aren't working, you can look in the [Shiny Server log](#) which may have more information about

**It's Alive!**

Number of bins:

1 6 11 16 21 26 31 36 41 46 50

**Histogram of x**

Frequency

15 25

The histogram shows the frequency distribution of data 'x'. The x-axis represents the number of bins, ranging from 1 to 50. The y-axis represents the frequency, ranging from 15 to 25. The distribution is roughly bell-shaped, centered around 30 bins.

## 5.安装Shiny Server

在AWS上部署免费的Shiny应用

35

打开的页面是默认的Shiny Server的网页，如果和上面的截图一样，说明你的Shiny Server安装成功了。

提醒一下，AWS的EC2的主机，一定要配置网络访问策略，打开3838端口，允许外部访问，不然一直都是无法访问此网站的错误。

1. Shiny是什么？
2. 本地开发一个Shiny小应用
3. 申请AWS免费服务器
4. 在AWS上安装R语言环境
5. 在AWS上安装Shiny Server
6. 在AWS上部署自己的Shiny应用

## 6. 部署自己的Shiny应用

最后一步，把我们自己开发的Shiny应用，部署到AWS的EC2上面。代码上传的过程，我们可以基于github来完成。

1. 在github上面，新建一个项目，名为shiny-demo。
2. 把本地开发的代码，上传到github的shiny-demo项目中。
3. 在AWS的EC2上，从github的shiny-demo项目中，下载代码。
4. 在AWS的EC2上，修改Shiny Server的配置，加载项目代码。
5. 在AWS的EC2上，重启Shiny Server，发现错误。
6. 在AWS的EC2上，查看日志修复错误。
7. 在浏览器上访问，自己的Shiny应用。

## 6. 部署自己的Shiny应用

在AWS上部署免费的Shiny应用

38

6.1. 在github上面创建项目，名为shiny-demo。

github操作过程省略。项目地址 <https://github.com/bsspirit/shiny-demo>。

## 6. 部署自己的Shiny应用

6.2 把本地开发的代码，上传到github的shiny-demo项目中。

切换到本地开发的环境。

```
~ cd d:\workspace\dash
~ git init
~ git add .
~ git commit -m 'init'
~ git remote add origin https://github.com/bsspirit/shiny-demo.git
~ git push -u origin master
```

## 6.3 在AWS的EC2上，从github的shiny-demo项目中，下载代码。

```
# 下载github项目
~ git clone https://github.com/bsspirit/shiny-demo
Cloning into 'shiny-demo'...
remote: Counting objects: 12, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 12 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (12/12), done.
Checking connectivity... done.

# 查看项目文件
~ cd shiny-demo
~ tree
.
|___ demo1
|   |___ server.R
|   |___ ui.R
|___ README.md
|___ run.R
```



## 6. 部署自己的Shiny应用

6.4 在AWS的EC2上，修改Shiny Server的配置，加载项目代码。

编辑shiny-server的配置文件shiny-server.conf。

```
~ sudo vi /etc/shiny-server/shiny-server.conf
```

```
# 新增加指向github的代码位置
location /demo1 {
    app_dir /home/ubuntu/shiny-demo/demo1;
    log_dir /var/log/shiny-server/demo1;
}
```

## 6. 部署自己的Shiny应用

在AWS上部署免费的Shiny应用

42

6.5 在AWS的EC2上，重启Shiny Server，发现错误。

重启Shiny Server，虽然只是重启，但经常出现错误。

```
~ sudo service shiny-server restart
```

# 对于Ubuntu 15.04+的系统，推荐用下面的命令。

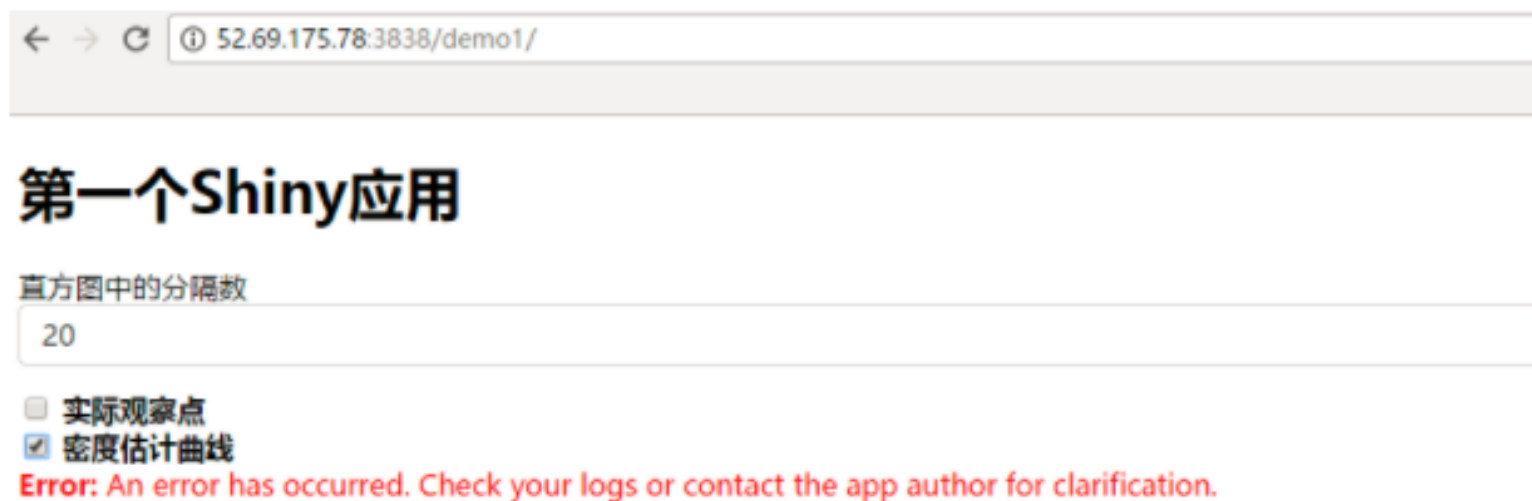
```
~ sudo systemctl restart shiny-server
```

## 6. 部署自己的Shiny应用

在AWS上部署免费的Shiny应用

43

重启后，就可以在浏览器上，访问自己的Shiny应用了。



← → ↻ ⓘ 52.69.175.78:3838/demo1/

### 第一个Shiny应用

直方图中的分隔数

20

☐ 实际观察点

☒ 密度估计曲线

Error: An error has occurred. Check your logs or contact the app author for clarification.

## 6. 部署自己的Shiny应用

6.6 在AWS的EC2上，查看日志，修复错误。

主要的调试的方法，就是检查Shiny Server的日志。日志在设置上，有一个很大的坑，我也是在挠头搞了3个小时后才发现的。

由于Shiny Server为了保证性能，所以非敏感性的错误日志被设置了**自动清除**，每当你出现了错误，要去看日志定位问题的时候，这个日志就刚好被自动清除了。坑很大！！都开始怀疑人生了。

所以，你在调试时需要修改一个参数，保证日志不会被自动清除。

## 6. 部署自己的Shiny应用

在AWS上部署免费的Shiny应用

45

发现问题，检查日志，我们对对应日志的解决问题。

很多情况下，诡异的错误都是缺少第三方包造成的，当你程序中使用了第三包的时候，一直要记得在Shiny的服务器上面安装好。记得用root用户！！

## 6. 部署自己的Shiny应用

在AWS上部署免费的Shiny应用

46

总结一下，我们利用免费的AWS的EC2服务器资源，发布了自己的Shiny应用，是多么的开心啊！

这样以后就可以大胆地去开发自己喜欢的Shiny应用了，当然不只是Shiny应用，你可以利用AWS的免费资源，做更多的事情。**老司机，都明白的！！**

原本是准备把一个基于赌场原型的Shiny应用放到互联网，考虑服务器位置和选型的问题，无意中发现了AWS的免费资源，这样就有了这样的一篇Shiny与AWS结合的文章。

要不要分析一下**赌场**的模型呢。

把中国金融二级市场，在游戏里重新实现。

参与者可以扮演下面的角色：

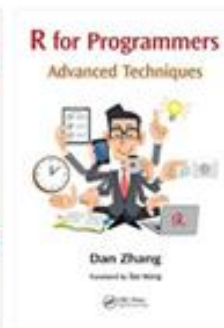
中央银行，商业银行，交易所，券商，证监会，银监会，**散户**，私募，做市商，  
配资机构，保险机构....

使用市场的规则，用模型改进交易策略，让游戏平衡，增加博弈的维度。





新书《R的极客理想-量化投资篇》，将在2017年8月由机械工业出版社出版。



# Thank you!

张丹

在AWS上部署免费的Shiny应用