

Introduction: Topics for Week 1-3

FE8828 Programming Web Applications in Finance

Week 1

What's Internet? What's Web?

Launch into the Cloud

Catch-up with R

Dr. Yang Ye <Email:yy@runchee.com>

Nov 02, 2017

■ Week 1:

- *What's Internet? What's Web?*
- *Launch into the Cloud: AWS*
- *R catch-up: R Markdown and Shiny layout*

■ Week 2:

- *Data, visualization, and web: part 1/2*

■ Week 3:

- *Data, visualization, and web: part 2/2*

Introduction: Topics for Week 4-6

■ Week 4:

- *Finance applications*

■ Week 5:

- *Bitcoin and Blockchain*

■ Week 6:

- *Further topics of Blockchain*
- *Presentation of Assignment: Web application.*

■ Week 7:

- *Presentation of Assignment: Blockchain.*

Introduction: Assingments

■ Week 1: A static website. A front page, a about page and a description page. (due by Week 2)

■ Week 2: A dynamic website that can do CRUD. (due by Week 3)

■ Week 3: A data-driven website that does data analytics. (due by Week 4)

■ Week 4: A finance-data application website. (due by Week 6)

■ Week 5: Reading on Blockchain. (due by Week 7) Working on web assignment. (due by Week 6)

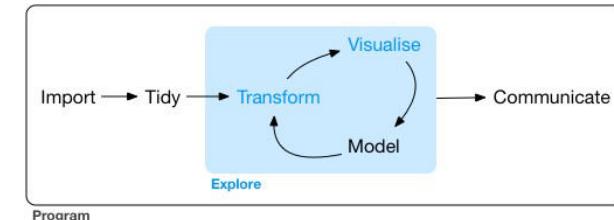
■ Week 6: Reading on Blockchain. (due by Week 7)

Introduction: Objective

1. Know the way of Internet: the network, the cloud and the application.
2. Build real-world data-driven reports and dashboard, data visualization.
3. Latest Internet technology in cryptocurrency and payment system like Bitcoin and Blockchain.

Where does this course stands on?

- It's about "Data science" Data -> Model -> Application



- We deep dive into R.

- *R is a system that has been designed to process data. Later in career you will expose Python, Julia, Hadoop.*
- *The principle will be the same.*

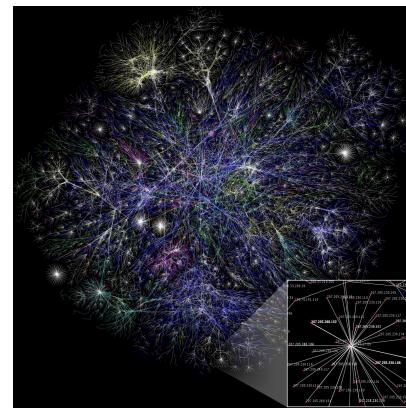
Keep calm and code on



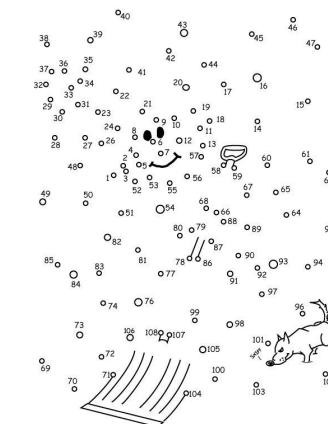
Lecture 01: What's Internet? What's Web?

- Network
- Internet
- HTTP/HTML/Web

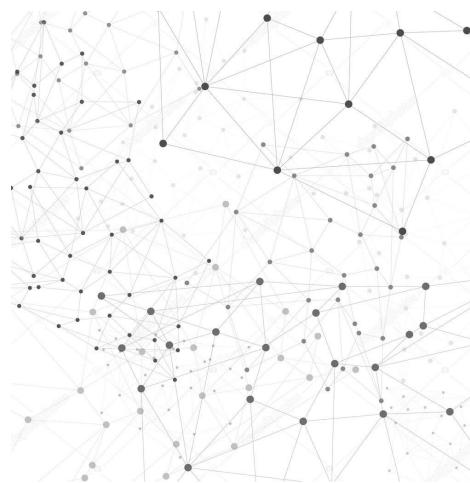
Network



Network is to connect the dots

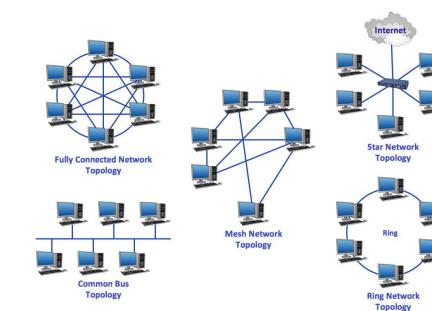


Network is to connect the dots devices



Building network

- There are many ways to connect the devices: Network topology



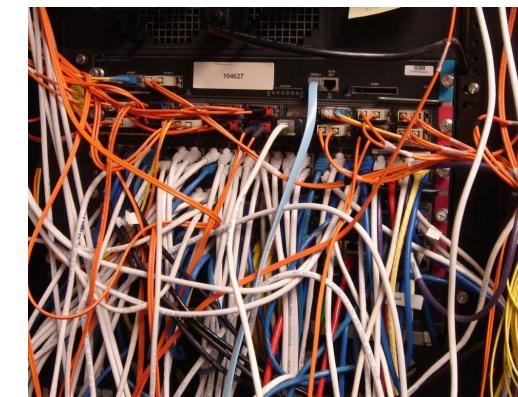
- Which network topology is our home Wi-Fi?

Building network

- Different network topology takes different way of communications.
- Fully-connected network is the most costly and robust. Ring is the cheapest but vulnerable.

Network talks

After laying down the wires, what runs inside?



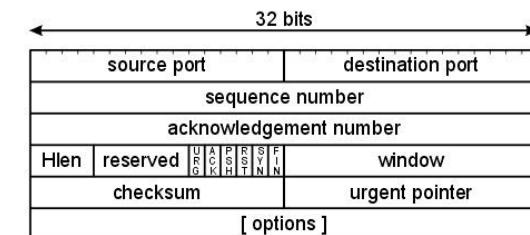
Network talks

- Information turns to packet
- Protocol designs the packet and process
- Infrastructure is to route the packets to the destination.

Packet

- Information turns to packet

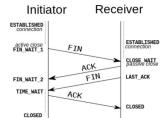
TCP header format



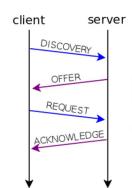
Protocol

2. Protocol designs the packet and process

TCP Session

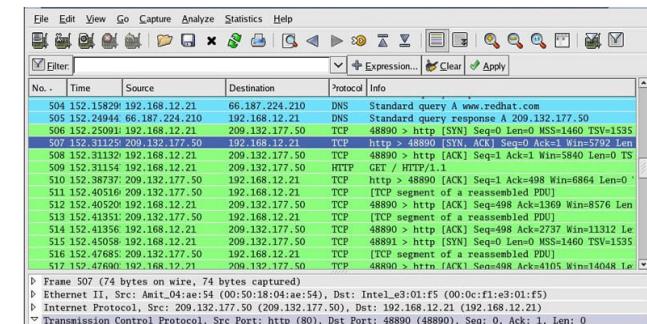


DHCP Session



Routing/Gateway

3. Infrastructure helps to transmit and route the packets to the destination.

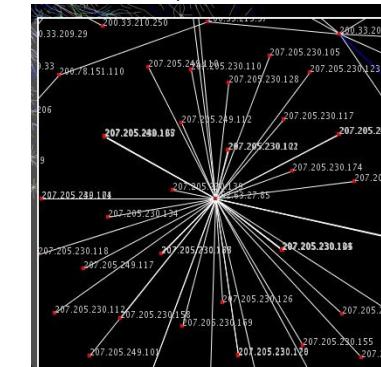


TCP/IP and Internet

The Defense Advanced Research Projects Agency (DARPA) created the TCP/IP model in the 1970s to build ARPANET. ARPANET is a wide area network that preceded the internet.

What does TCP/IP gives?

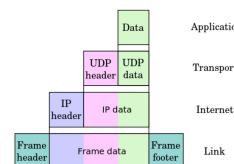
I. IP (Internet Protocol, IP address)



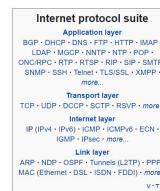
2. TCP (Transmission Control Protocol) TCP provides reliable, ordered, and error-checked delivery of a stream

TCP/IP

- Four Layers



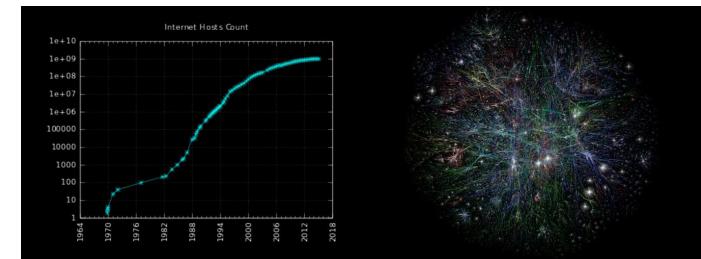
- Application layer runs many protocols



What happens after plugging cable, turning on Wi-Fi?

- Every network device has a hardware address
- DHCP client, a component of the operating system sends out a DHCP request and receives an offer from a DHCP server running on the router.
- Device accepts the IP address and uses it to label for itself. Router also knows where to send the packet.

This is why and how Internet growed in size



When it wants to visit someone on the network? 1/2

- Domain Name System:
 - We don't use 123.456.789.012 but www.google.com.
 - DNS is the directory service for internet.
 - Your device also receives one or more DNS server addresses so the computer knows where to send DNS requests.



Demo with nslookup.

When it wants to visit someone on the network? 2/2

- Then, your device creates and send the packet “request”. Wait for response.
- Router and gateway will relay the packets to the receiver.



27 of 82

FE8828 Programming Web Applications in Finance (60)

10/30/2017, 12:05 AM
file:///D/Dropbox/Docs/MFE/FE8828/notes/lec01.html#(60)

HTTP Request/Response

- Request

```
GET /index.html HTTP/1.1
Host: www.example.com
```

- Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
<head>
    <title>An Example Page</title>
</head>
<body>
    Hello World, this is a very simple HTML document.
</body>
</html>
```

HTTP/HTML

- Initiated by Tim Berners-Lee at CERN in 1989.
- HTML was also invented by Tim, “HTML tags”

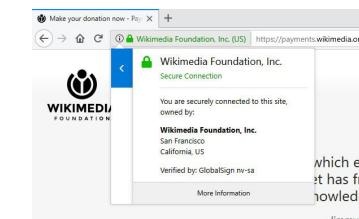
29 of 82

FE8828 Programming Web Applications in Finance (60)

10/30/2017, 12:05 AM
file:///D/Dropbox/Docs/MFE/FE8828/notes/lec01.html#(60)

HTTP/HTML

- Hypertext Transfer Protocol (HTTP) but obviously, it does file, music, anything now.
- It's a clear text protocol. That's why we need to use HTTPS (HTTP on SSL) to secure the communication.



In browser, you can see the green lock in address bar. App doesn't show it, we just trust that they are using HTTPS.

Web

When you have HTML and URL (Uniform Resource Locator), Web is born.

- Website: <https://en.wikipedia.org/>
- Document: https://en.wikipedia.org/wiki/World_Wide_Web
- Resource: https://en.wikipedia.org/wiki/World_Wide_Web#/media/File:Web_Index.svg

How this resource is used in the document.

```
<div class="thumbinner" style="width:302px;">
  <a href="/wiki/File:Web_Index.svg" class="image">
    
  </a>
</div>
```

Web Browser

First generation



Web application

- Static v.s. Dynamic
- Dynamic website display content based on user input.
- Supported by HTML/CSS/JavaScript. HTML 5, CSS 3 and JavaScript 7 .
- App also uses HTML/CSS/JS.



Web application

- Why it is important?



- Needless to say. It is not 1995 anymore.
- Easy to develop
- Easy to deploy
- Runs fast

To recap

Why the Internet succeeded?

- Information flows by packet.
 - IP protocol sets the address for the device
 - TCP protocol transmit the packet reliably
 - UDP, for real-time transmission which can accept failure.
- Local device just needs to send the packet
- Transmission is done by the network devices
- Efficient and scalable

To recap

Web application

- Browser => HTML/CSS/JavaScript => HTTP => TCP => IP => Network physical.
- We will write in R, which subsequently writes HTML/CSS/JavaScript.

Content

1. Sign-up for AWS Account
2. Setup AWS for EC2
3. Launch EC2
4. Running R

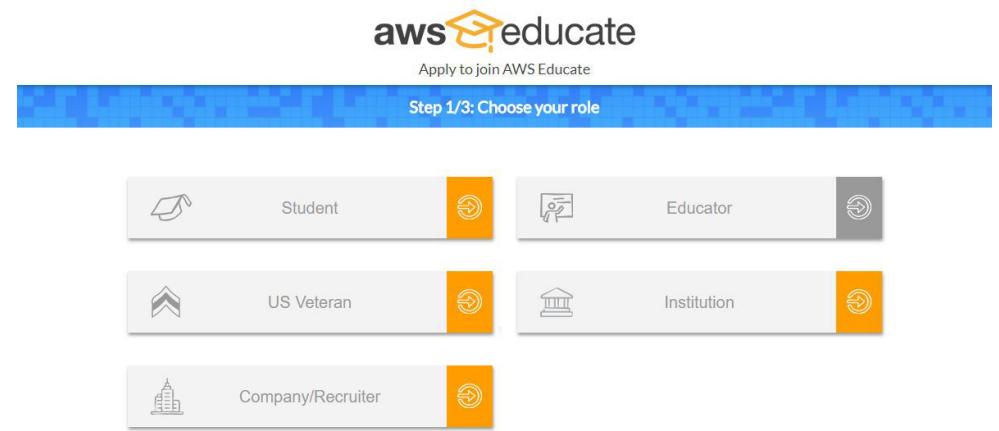
Launch in the cloud: AWS

Yang Ye
MFE FF8828
2017

Sign-up for AWS

1. Sign-up with an AWS account at <https://aws.amazon.com>
- Students will need to either create a new AWS account or use one that they've already created. Their AWS account needs to be fully activated by completing phone verification steps and adding a valid credit card.
2. Sign-up AWS Educate account at
<https://aws.amazon.com/education/awseducate/>
- They will need to select the AWS Account option and enter their 12 digit AWS Account ID number when they apply to AWS Educate.
3. (Depends) Because NTU is not (yet) listed as a institute, you may need to apply credit from AWS Educate separately
<https://aws.amazon.com/education/awseducate/contact-us/>
4. Claim credit from AWS according to the instruction of AWS

Register for AWS Educate account



<https://www.awseducate.com/registration>

Apply AWS Educate Credit

Name: Yang Ye
Account: 457734848212

Regarding* Account and Billing Support
 Service Limit Increase
 Technical Support
Unavailable under the Basic Support Plan

Service*

Category*

Subject*

Description Guidance

AWS Educate provides educators and students with grant-based access to AWS, training, and content, while also providing educators and students with a forum for collaboration. If you would like additional information about AWS Educate, please visit <https://aws.amazon.com/education/awseducate/>.

Description*

<https://aws.amazon.com/education/awseducate/contact-us/>

2. Setup AWS

After you login to AWS

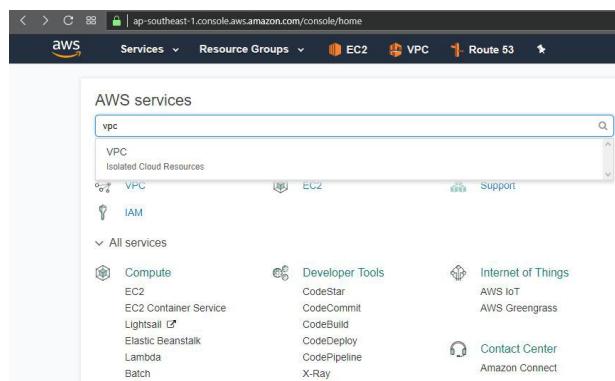
1. Change your region to Singapore.
2. Add shortcuts in the navigation bar
 - a. EC2
 - b. VPC



<https://aws.amazon.com/education/awseducate/contact-us/>

2. Create VPC Network

Search for VPC



2.1 VPC: Click “Start VPC Wizard”

A screenshot of the VPC Dashboard. At the top right, there are buttons for "Start VPC Wizard" and "Launch EC2 Instances". Below these buttons, a note says "Note: Your Instances will launch in the Asia Pacific (Singapore) region". On the left, there's a sidebar with "Virtual Private Cloud" sections for "Your VPCs", "Subnets", "Route Tables", and "Internet Gateways". On the right, a summary shows "2 VPCs", "0 Egress-only Internet Gateways", "3 Subnets", "2 Route Tables", "3 Internet Gateways", "2 Network ACLs", and "0 Elastic IPs".

2.2 VPC two steps - follow defaults, add a name

Step 1: Select a VPC Configuration

VPC with a Single Public Subnet

Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.

Creates:

A /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet.

Select

Step 2: VPC with a Single Public Subnet

IPv4 CIDR block*: 10.0.0.0/16 (65535 IP addresses available)

IPv6 CIDR block: No IPv6 CIDR Block Amazon provided IPv6 CIDR block

VPC name: mte

Public subnet's IPv4 CIDR*: 10.0.0.0/24 (251 IP addresses available)

Availability Zone*: No Preference

Subnet name: Public subnet

You can add more subnets after AWS creates the VPC.

Service endpoints

Add Endpoint

Enable DNS hostnames*: Yes No

Hardware tenancy*: Default

2.3 Subnet: Change settings. Tick auto-assign public IP4 address

Create Subnet

Subnet Actions

Search Subnet

Name: Public subnet

Subnet Actions

Delete Subnet

Edit IPv6 CIDRs

Create Flow Log

Modify auto-assign IP settings

Virtual Private Cloud

Your VPCs

Subnets

Route Tables

Public subnet subnet-834b62ca available vpi

Modify auto-assign IP settings

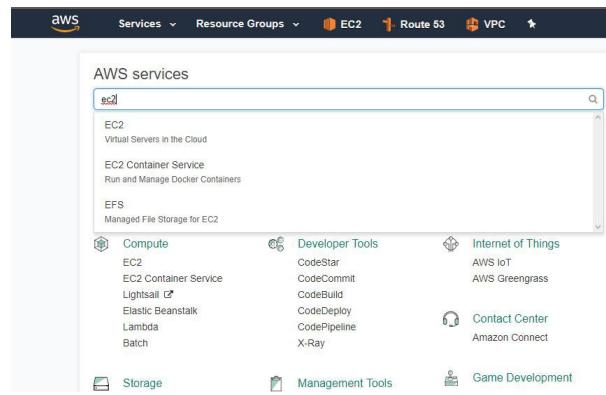
Enable auto-assign public IPv4 or IPv6 addresses to automatically request an IP address for instances launched into this subnet.

Auto-assign IPs Enable auto-assign public IPv4 address

Note: You can override the auto-assign IP settings for each individual instance at launch time for IPv4 or IPv6. Regardless of how you've configured the auto-assign public IP feature, you can assign a public IP address to an instance that has a single, new network interface with a device index of eth0.

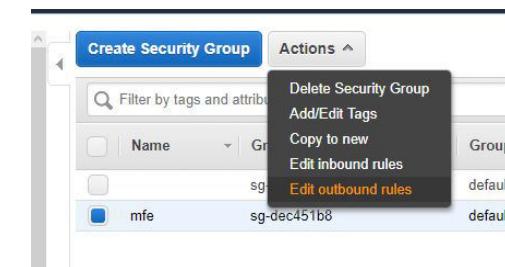
Cancel Save

3. Go to EC2



3.1 In VPC wizard, it created new security group.

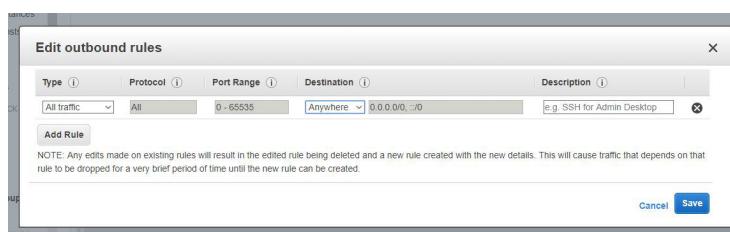
Selected the newly created security group.
Edit Inbound rules and edit outbound rules.



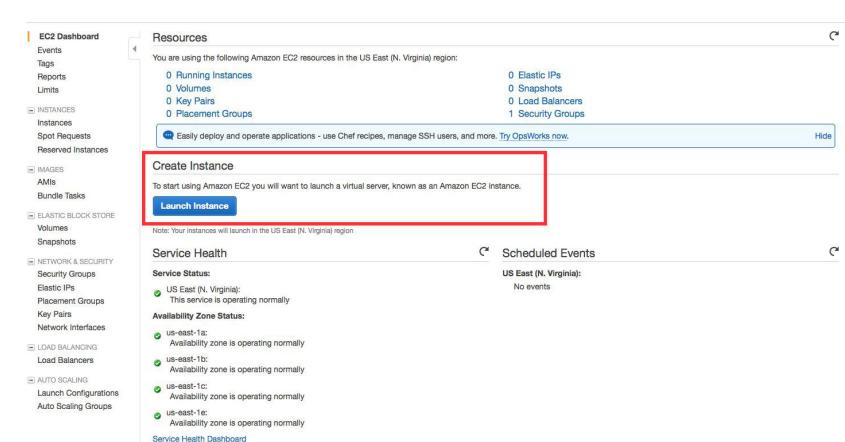
3.2 Inbound/Outbound: choose All traffic/Anywhere

An inbound firewall protects the network against incoming traffic from the internet or other network segments, namely disallowed connection from outside.

Outbound rules determines what application can connect to the outside.



Ready to Launch



EC2 Step 1: Community AMI: search for “RStudio”

ami-a13b59c2

This is created by http://www.louisaslett.com/RStudio_AMI/

EC2 Step 2: Choose instance type. You can enjoy having one instance of t2.micro during 12-month free-tier.

- Upgrade to higher instance type when you have received your AWS educate credit.
- T2.medium/t2.large should be good enough. There are more expensive ones.

EC2 Step 3: Make sure Auto-assign Public IP is ticked. Click “Next”.

EC2 Step 4: Add Storage, use the default setting.

- One AMI image can be launched into many instances so its content is static.
- The author of AMI image configures a persistent storage so it saves any changes we did.
- Most importantly, we don't need to do anything now.

EC Step 6: Click “6. Configure Security Group”

Select an existing security group. Your previous changes are loaded.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: Create a new security group Select an existing security group

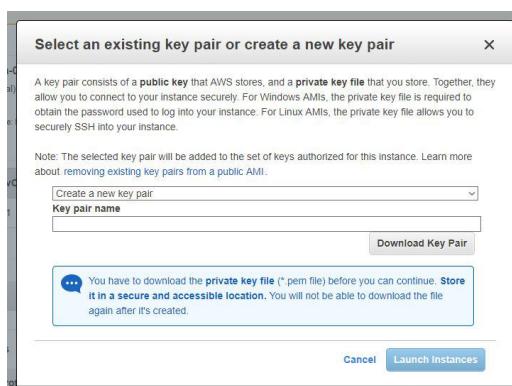
Security Group ID	Name	Description	Actions
sg-dec451b8	default	default VPC security group	Copy to new

Inbound rules for sg-dec451b8 (Selected security groups: sg-dec451b8)

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	0.0.0.0/0	

haha!!!

One last thing: Key pair
Create if you didn't have an existing one or lost the previous download.



EC 2 Step 7: Review and Launch

Click the blue button.

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details

AMI: [AMI Studio-1.0.153_R-3.4.1_Julia-0.8.0_ubuntu16.04-LTS-64bit - ami-e13b59c2](#)

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

Security Group ID	Name	Description
sg-dec451b8	default	default VPC security group

All selected security groups inbound rules

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	0.0.0.0/0	

[Cancel](#) [Previous](#) [Launch](#)

If you know SSH

You can connect to the server by

- Windows: Download Git for Windows from <https://git-scm.com/download/>.

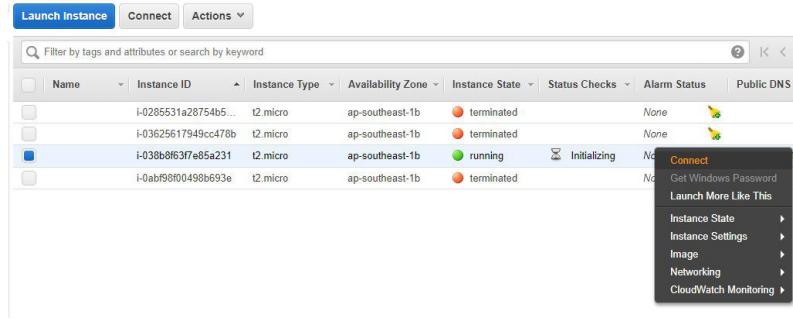
`ssh -i 'c:\Users\yourusername\Downloads\MyKeyPair.pem' ubuntu@{IP_Address}`

- Mac: skip download Git. Go straight

`ssh -i ~/Downloads/MyKeyPair.pem ubuntu@{IP_Address}`

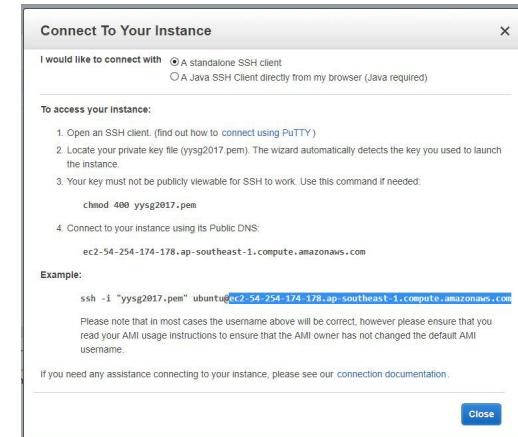
Control the instance.

- Connect gives information.
- Stop but not to terminate, which deletes all data from EBS.
- A running instance charges.
- A stopped instance will charge a small fee for the storage. Our configured size is 10G, within the free-tier for new account within 12-month.
- Use AWS calculator



Access it

- Copy this address



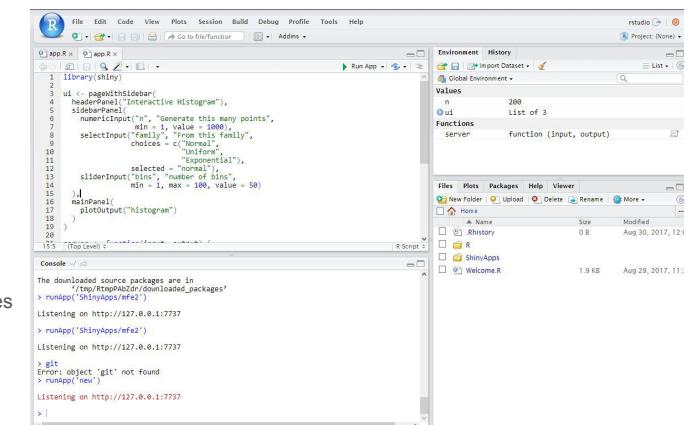
<http://ec2-54-254-174-178.ap-southeast-1.compute.amazonaws.com/>

Initial password: rstudio/rstudio.

The top screenshot shows the RStudio interface with a Shiny app running in the browser tab. The bottom screenshot shows the 'Sign in to RStudio' dialog box, where 'rstudio' is entered into the Username field.

RStudio Setup

- Install packages
 - tidyverse
 - tidyquant
 - install.packages
- Tools -> Shell
 - passwd
- Shiny
 - Directory ShinyApps
 - Create sub-directories
- Files
 - Manage upload



Shiny Server

/home/rstudio/ShinyApps/new/

/ShinyApps/new/

http://ec2-54-254-174-178.ap-southeast-1.compute.amazonaws.com/shiny/rstudio/new/

For this course

- AWS setup is optional in this course.
- If you finishing working on it, leave the instance as Stopped. Be aware of how much you spend on AWS.
- It's cool to have something running in the cloud. You can show people to impress.
- You can run R Studio and application on your laptop
- You can submit your assignment with me by Dropbox/Google drive.
- Please organize your assignments into different directories.

AWS Recap

- AWS (Amazon Web Services) is an utility company for the internet like electricity/water.
- We can launch many computing/storage resources as we need.
 - We use Spot instance which is expensive.
 - AWS provides other pricing plan for long-term usage.
- Cool! We got something in the cloud.

Disclaimer:

- 1.
2. I am not working for Amazon and I don't get paid by this.
3. I am not owning Amazon shares directly and indirectly.
4. I don't plan to long AMZN during the course of this course.

FE8828 Programming Web Applications in Finance (60)

file:///D:/Dropbox/Docs/MFE/FE8828/notes/lec01.html#(60)

Lecture 03: R Markdown and Shiny (layout)

Introduction

- **Markdown** is a format that is easy to read and can be converted to other formats, HTML, PDF, Word, Slides.
- R Studio extends it further to create R notebook, interactive document and web application, which is **R Markdown**.
- **Shiny** is a web programming framework in R. We use it extensively in this course. We begin with the layout part.

Markup and Markdown

- Document stores information.
- Web is a superset of interlinked documents.
- HTML is a markup language, built for machines.
- Markdown is for humans to write doc, with minimal added to decorate it, created by John Gruber in collaboration with Aaron Swartz in 2004.

A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. - John Gruber



Markdown doc example

title	subtitle	author	date
My first bitcoin	and how I bought a pizza!	Despicable me	Jul 9, 2010

How I bought it

I found someone was selling 10000 on ebay for \$30. I think that's

- cool
- fun
- hacker

How I used it

I forgot to bring my wallet the other day. I was hungry so I used the bitcoins to buy some pizza.



Markdown example

```
---
title: My first bitcoin
subtitle: and how I bought a pizza!
author: "Gru"
date: "Jul 9, 2010"
---

# How I bought it

I found someone was selling _10000_ on ebay for __$30__.
I think that's

- cool
- fun
- hacker

# How I used it

I forgot to bring my wallet the other day.
So I used **the bitcoins** to buy some pizza.

![Pizza](..../notes/imgs/bitcoin-pizza.png)
```

Markdown: Header and Code

Headers

More hashtag, deeper level.

```
# Header1
## Header2
### Header3
```

Code

Give four spaces before it

```
    if (a > 0) {
        print(a)
    }
```

```
if (a > 0) {
    print(a)
}
```

Markdown: List

```
* First paragraph.
Continued.

* Second paragraph. With a code block, which must be indented
eight spaces:

{ code }
```



- First paragraph. Continued.
- Second paragraph. With a code block, which must be indented eight spaces:

Markdown: Multi-level lists

Put four more spaces for each level.

```
* fruits
  + apples
    - macintosh
    - red delicious
  + pears
* vegetables
  + broccoli
  + chard
```

- fruits
 - apples
 - macintosh
 - red delicious
 - pears
- vegetables
 - broccoli
 - chard

Markdown: Ordered Lists

Put 4 more spaces for each level.

```
#. Chapter 1
  #. Section 1.1
  #. Section 1.2
#. Chapter 2
#. Chapter 3
```



1. Chapter 1
 1. Section 1.1
 2. Section 1.2
2. Chapter 2
3. Chapter 3

Table

Tables	Are	Cool
col 3 is	right-aligned	\$1600
col 2 is	centered	\$12
zebra stripes	are neat	\$1



Tables	Are	Cool
col 3 is	right-aligned	\$1600
col 2 is	centered	\$12
zebra stripes	are neat	\$1

Markdown: Inline formatting

Emphasis

To emphasize some text, surround it with *s or _, like this:

```
This text is \_emphasized with underscores\_, and this
is \*emphasized with asterisks\*.
Double * or _ produces strong emphasis:

This is \*\*strong emphasis\*\* and \_\_with underscores\_\_.
```



This text is emphasized with underscores, and this is emphasized with asterisks.
Double * or _ produces strong emphasis.

This is **strong emphasis** and **with underscores**. A * or _ character surrounded by spaces, or backslash-escaped, will not trigger emphasis.

- If the verbatim text includes a backtick, use two backticks.
- Use \ to turn off \~, \^.

Markdown: Inline formatting

Strikethrough

This ~~is deleted text.~~ This is deleted text.

Superscripts and subscripts

H²O is a liquid. 2¹⁰ is 1024. H₂O is a liquid. 2¹⁰ is 1024.

Verbatim. inline code

Use backtick `. What is the difference between `>>` and `>>`? What is the difference between >>= and >>?



Note:

Markdown: Links

```
<http://google.com>
```



<http://google.com>

Images

A link immediately preceded by a ! will be treated as an image. The link text will be used as the image's alt text:

```
! [Pizza] (imgs/bitcoin-pizza.png)
```




Pizza

Formula

MathJax. Use LaTeX syntax. There are many online references.

Inline with text

```
$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Centered

```
$$\sum_{i=1}^n x_i$$
```

$$\sum_{i=1}^n X_i$$

R Markdown

Reference in R Studio

- R Markdown Cheat Sheet: Help > Cheatsheets > R Markdown Cheat Sheet,
- R Markdown Reference Guide: Help > Cheatsheets > R Markdown Reference Guide.

Create it via File > New File > R Markdown.

- Document
- Presentation
- Shiny

R Markdown Document example

```
---
title: "Data Analysis Report"
author: "Yang Ye"
date: "October 23, 2017"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## Report
```{r cars}
summary(cars)
```

## Including Plots
```{r pressure, echo=FALSE}
plot(pressure)
```

```

R Markdown Document Output

In the header, you can change the output to other types:

- html_document
- pdf_document
- word_document
- Ctrl+Shift+K or “Knitr”

Code block for R Markdown

- R Markdown is a extension to Markdown that you can execute code among the code. If you name the file as **.Rmd** and **knit** in R Studio.

```
```{r Calculate_7}
a <- 3
b <- 4
print(a + b)
```

```

```
## [1] 7
```

- Calculate_7 is the chunk name. It's optional to give a chunk name. If included, each code chunk needs a distinct name. It's usually best to give each code chunk a name, for easier debug.
- R code can also be inline. For example, to generate a random number everytime, include this `runif(1, 0, 1)`, 0.8516547.

- echo is to decide whether to display code, default is FALSE.
- result is to decide whether to display result, default is show, set to “hide” to hide.

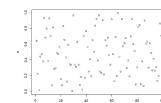
■ include is to hide both code and result, default is FALSE.

```
```{r cars, echo = TRUE}
a <- runif(100, 0, 1)
```

```{r plot}
plot(a)
```

```

```
a <- runif(100, 0, 1)
```



R Markdown example: Table

```
```{r kable}
knitr::kable(
 mtcars[1:5,],
 caption = "A knitr kable."
)
```

```

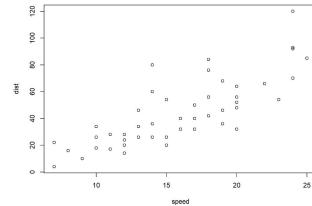
A knitr kable.

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |

R Markdown example: plot

```
```{r plot1, echo = FALSE}
a <- filter(cars, speed > 4)
plot(a)
```

```



R Shiny

- To start, use R Studio.
- File > New File > Shiny Web App...
- Choose single file
- Give a name and folder
- Ctrl+Shift+S or “Run App”

UI First

I removed everything in functions `server` and `ui`. This is the minimal Shiny.

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  )

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Sidebar Layout

Let's add a minimal sidebarLayout

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

fluidPage

- fluidPage means to place the controls from left-right, top-down order.
- fluidPage function can take any number of input parameter.

```
fluidPage(
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  ))
```

Add some things

- titlePanel("Hello Shiny!"),
h1("Introduction to Layout"), h2("Sidebar Layout")

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  fluidPage(
    titlePanel("Hello Shiny!"),
    sidebarLayout(
      sidebarPanel(
        h1("Introduction to Layout"),
        h2("Sidebar Layout"),
        a("A link to Google", href="http://www.google.com"),
        tags$ul("About",
          tags$li("Who are we"),
          tags$li("What we do")
        ),
        tags$ol("Steps",
          tags$li("Write"),
          tags$li("Run")
        )
      ),
      mainPanel(
        img(src = "p19-Hero-Image-796x398.jpg")
      )
    )
  )
)
```

```
# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Sidebar layout with bar on the right

```
fluidPage(
  sidebarLayout(position = "right",
    sidebarPanel(),
    mainPanel()
  )
)
```

More tags

Each tag is a function.

```
h1("A header")
p("some text as a paragraph")
a("A link to Google", href="http://www.google.com")
img(src = "p19-Hero-Image-796x398.jpg", width = "100%")
tags$ul("title", tags$li("Item 1"), tags$li("Item 2"))
tags$ol("Step", tags$li("Item 1"), tags$li("Item 2"))
```

Note:

- For image, you need to create a sub-directory `www` together with the R source file. Place the file under it.
- `tags` is a list of functions. To avoid name conflict, I prefer to use `tags$img()`, even `img()` is available to use.

Panels

`titlePanel()` and `wellPanel()`

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  fluidPage(
    titlePanel("Hello Shiny!"),
    sidebarLayout(
      sidebarPanel(
        h1("Well 1"),
        wellPanel(
          h2("Well 1.1"),
          actionButton("goButton", "Go!")
        ),
        h1("Well 2"),
        wellPanel(
          h2("Well 2.1"),
          actionButton("goButton2", "Go!")
        )
      ),
      mainPanel(
      )
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) { }
```

```
# Run the application
shinyApp(ui = ui, server = server)
```

Navlist panel

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  fluidPage(
    titlePanel("Hello Shiny!"),
    navlistPanel(
      "Header A",
      tabPanel("Section 1",
        h1("Section 1")),
      tabPanel("Section 2",
        h1("Section 2")),
      "Header B",
      tabPanel("Section 3",
        h1("Section 3")),
      "-----",
      tabPanel("Component 5")
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) { }

# Run the application
shinyApp(ui = ui, server = server)
```

tabPanel

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  fluidPage(
    titlePanel("Hello Shiny!"),
    tabsetPanel(
      tabPanel("Plot", h1("plot")),
      tabPanel("Summary", h1("summary")),
      tabPanel("Image", img(src = "p19-Hero-Image-796x398.jpg"))
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

navBar

```
library(shiny)

ui <- fluidPage(
  navbarPage(title = "Runchee Technology",
    fluidPage(
      navbarPanel("Product",
        titlePanel("Hello!"),
        "One more thing!"),
      tabPanel("About us",
        titlePanel("Hello!"),
        "Exordinary people"),
      navbarMenu(title = "Contact Us",
        tabPanel("Address", "3/4 platform"),
        tabPanel("Phone", "+123.456")
      )
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Column-based layout

- Caveat: There is fluidRow, but no fluidColumn.
- Column counts always add up to $12 = 4 + 6 + 2$; otherwise, it will appear in the next line.

```
library(shiny)

ui <- fluidPage(
  fluidPage(
    fluidPage(
      titlePanel("Hello Shiny!"),
      fluidRow(
        column(4,
          wellPanel(
            dateInput("date", "How's weather today?")
          )
        ),
        column(6,
          h3("Plot"),
          wellPanel(plotOutput("distPlot"))
        ),
        column(2, h3("Extra"),
          wellPanel(plotOutput("extraPlot"))
        )
      )
    )
  )
)
```

```
# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Composition layout: Top and Down

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  fluidPage(
    fluidPage(
      title = "Diamonds Explorer",
      fluidRow(
        column(12,
          img(src = "p19-Hero-Image-796x398.jpg", width = "100%")
        )
      ),
      hr(),
      fluidRow(
        column(3,
          h4("Diamonds Explorer"),
          sliderInput('sampleSize', 'Sample Size',
                     min=1, max=nrow(diamonds), value=min(1000, nrow(diamonds)),
                     step=500, round=0),
          br(),
          checkboxInput('jitter', 'Jitter'),
          checkboxInput('smooth', 'Smooth')
        ),
        column(4, offset = 1,
               selectInput('x', 'X', names(diamonds)),
               selectInput('y', 'Y', names(diamonds), names(diamonds)[[2]]),
               selectInput('color', 'Color', c('None', names(diamonds)))
        ),
        column(4,
               selectInput('facet_row', 'Facet Row', c('None','.', names(diamonds))),
```

78 of 82

FE8828 Programming Web Applications in Finance (60)

10/30/2017, 12:05 AM

file:///D/Dropbox/Docs/MFE/FE8828/notes/lec01.html#(60)

79 of 82

FE8828 Programming Web Applications in Finance (60)

10/30/2017, 12:05 AM

file:///D/Dropbox/Docs/MFE/FE8828/notes/lec01.html#(60)

```
            )
          )
        )
      )
    )
  )
}

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

R Markdown can also contain Shiny

```
---
title: "MFE FE8828 Assignment 1"
date: 2017-11-03
output: html_document
runtime: shiny
---

```{r setup, include = FALSE}
```

```{r, echo = FALSE}
wellPanel("Inputs",
 numericInput("fav_num", "What's your favorite number?", 3))
```
```

Inputs

What's your favorite number?

This is interactive document.

Assignments

- (Optional) Setup AWS and run EC2.
- Create a website with Shiny using navBar layout
 - You are starting a company to offer.
 - Decide what you want to do
 - Create three pages. Name the pages depending on what you want to do. e.g. Product, About Us and Contact Us
 - Use different layouts for the pages: sideBar, column-based layout, Navlist.
 - Be creative!

Hello!

One more thing!



New breakthrough

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

To be Released

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Talk to Us!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut