FE8828 Programming Web Applications in Finance

Week 1

What's Internet? What's Web?

Launch into the Cloud

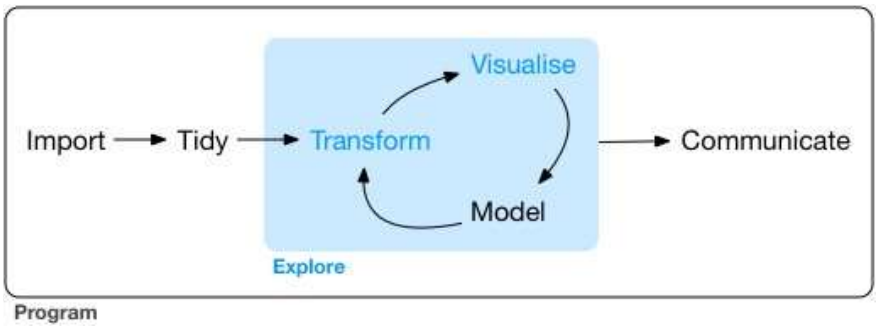
R Markdown and Shiny layout

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

Oct 29, 2018

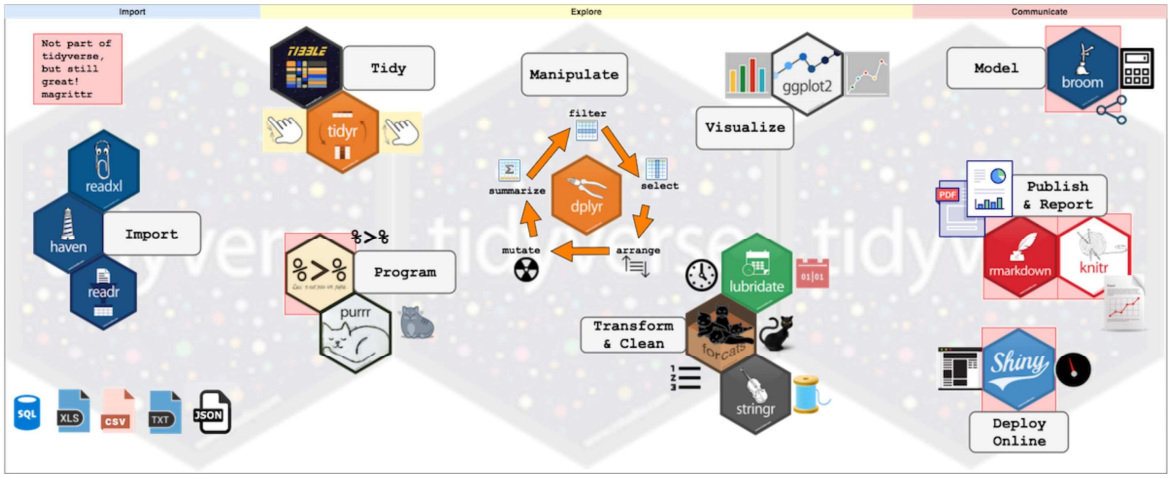
Where does this course stands on?

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



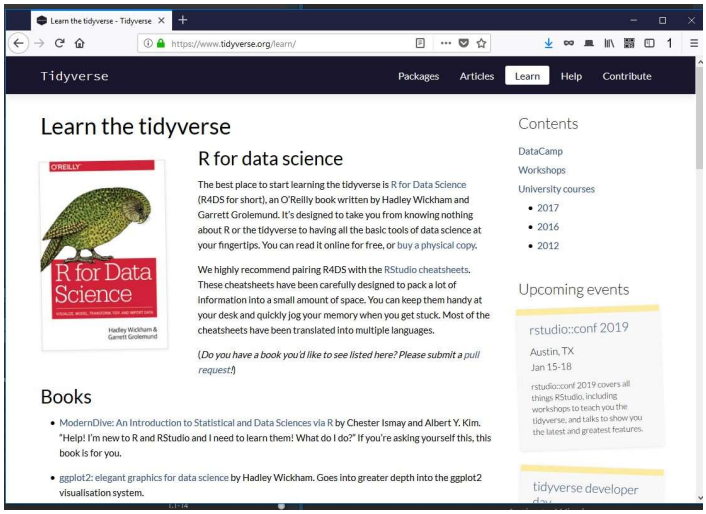
Where does this course stands on? (updated in 2018)

New version in 2018.



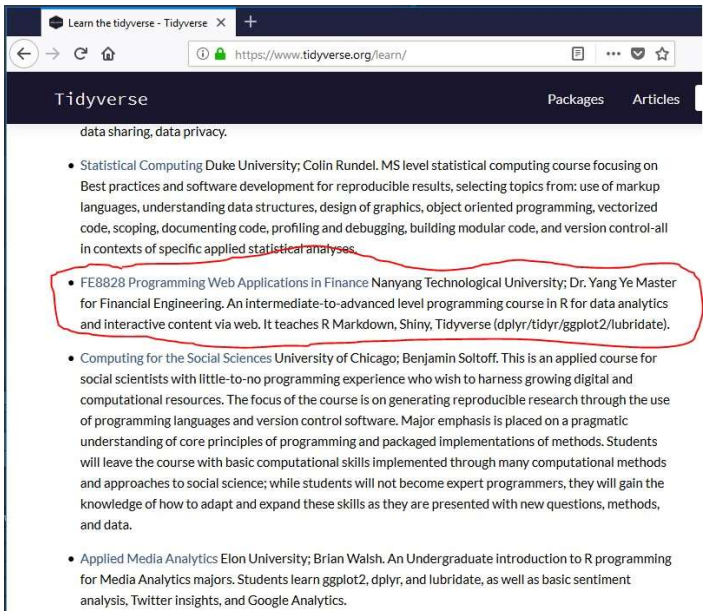
Where does this course stands on?

- TidyVerse with a finance flavour
- Reference book



Where does this course stands on?

- Our course



Objective

1. Know the way of Internet: the network, the cloud and the application.
2. Use data manipulation and data visualization to do exploratory data analysis.
3. To do option valuation, and trading strategy performance analysis.
4. Build real-world data-driven reports and dashboard, data visualization.
5. Latest technology in cryptocurrency and payment system like Bitcoin and Blockchain.

What does it take?

- Programming is our tool
  - *Intermidiate level R*
  - *R is a system that has been designed to process data.*
  - *Use R in other MFE courses*
  - *Later in career you will learn Python, Julia, Hadoop. The principle and idea is the same. Since you have practiced the full pipeline, easier to master the idea.*
- Take a habit of good analyst - Use notebook-style research, reproducible
- Take a mind of data exploration
- Take a mind of analysis - answer is not fixed but open-ended. You need to draw conclusion and make suggestion.
- Take a mind of strategy thinking

Course Outline: Week 1-3

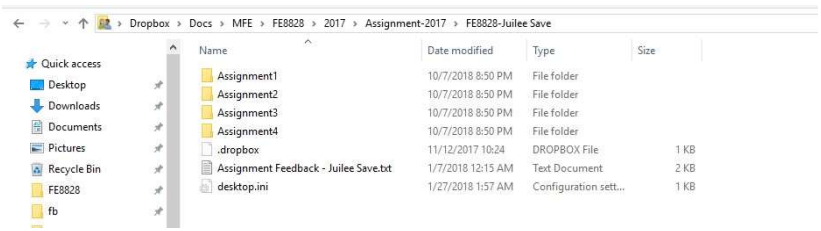
- Week 1:
  - *What's Internet? What's Web?*
  - *Launch into the Cloud: AWS*
  - *R Markdown and Shiny/1: layout*
- Week 2:
  - *Intermediate R Programming*
  - *Shiny/2: R Web Framework*
  - *Data Manipulation and Exploratory Data Analysis/1*
- Week 3:
  - *Data Manipulation and Exploratory Data Analysis/2*

Course Outline: Week 4-6

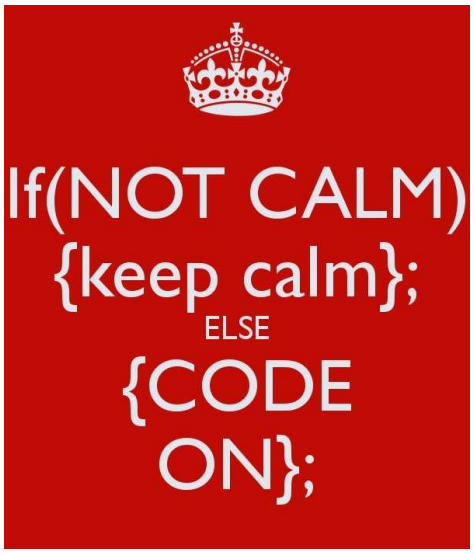
- Week 4
  - *Data Visualization*
  - *Shiny/3: Advanced*
- Week 5
  - *Build Applications*
- Week 6:
  - *Further topics on Blockchain*

Assingments

- Week 1: A static website. A front page, a about page and a description page. (due by week 2)
- Week 2: Shiny application. (due by week 3)
- Week 3: Data analysis (due by week 4)
- Week 4: Data visualization (due by week 5)
- Week 6: Group project
- You can submit your assignment with me by Dropbox/Google drive. My account: leafyoung@gmail.com (Google/Dropbox)
- Please organize your assignments into directories.



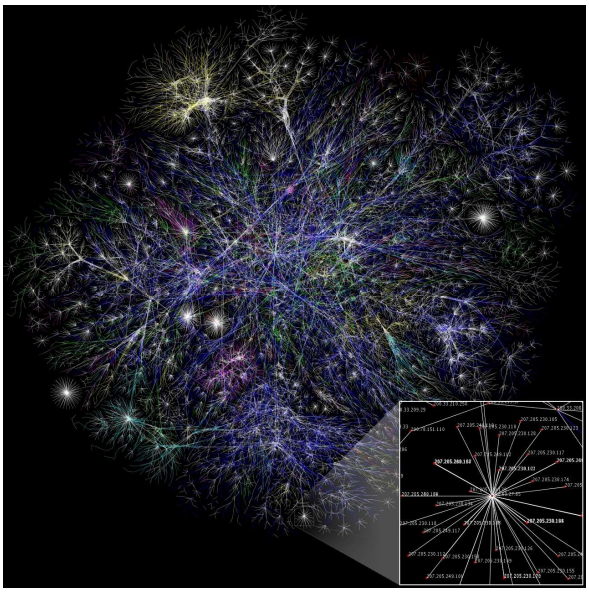
Keep calm and code on



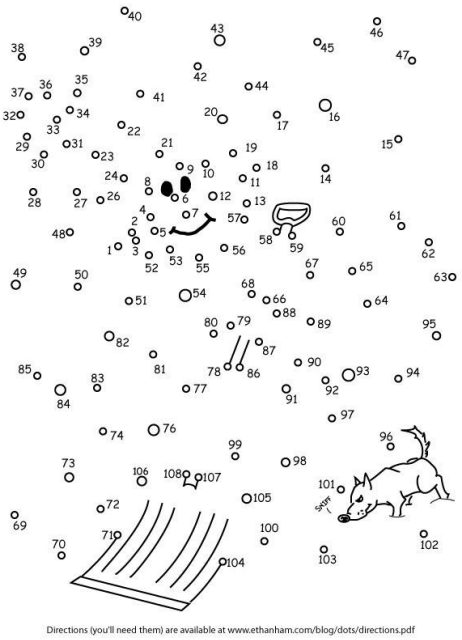
Lecture 1: What's Internet? What's Web?

- Network
- Internet
- HTTP/HTML/Web

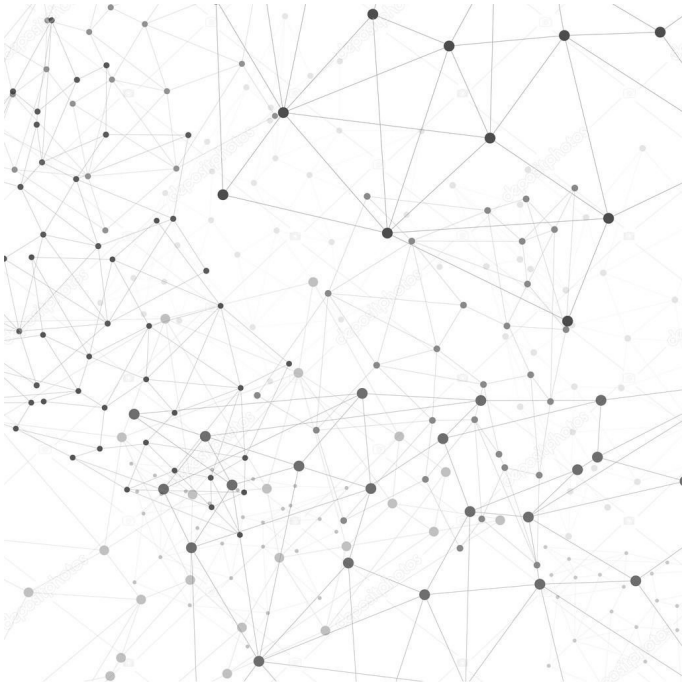
Network



Network is to connect the dots

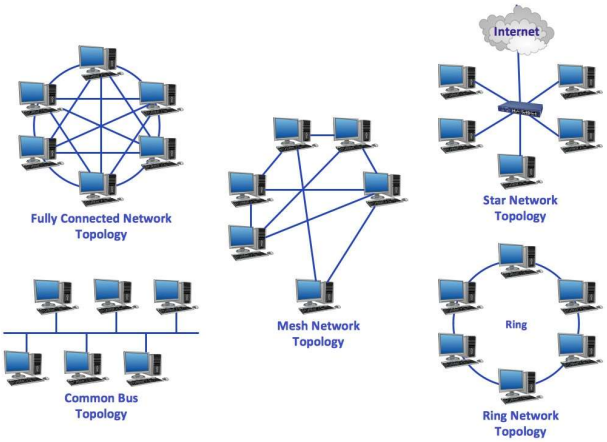


Network is to connect the devices



Building network

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



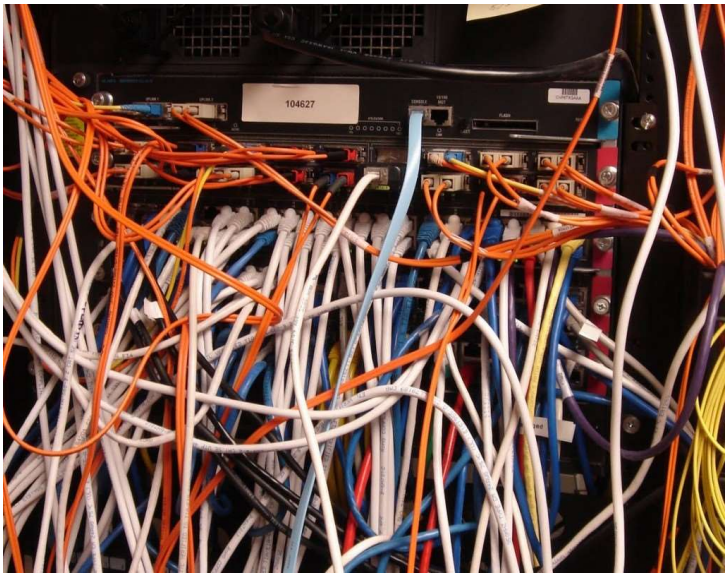
- Which network topology is our home Wi-Fi?

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

What runs inside the cables?

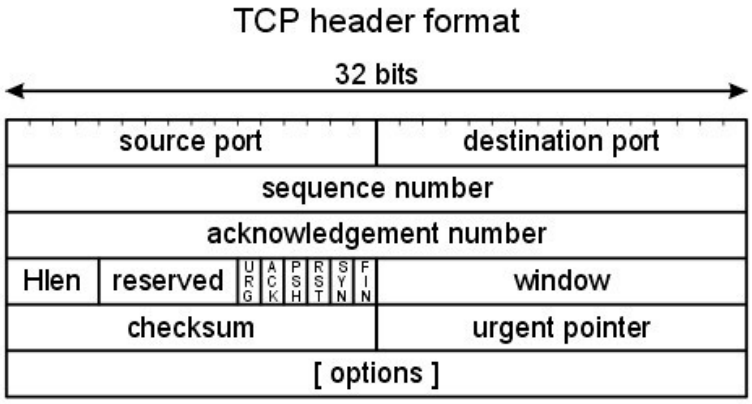


Network talks with Network Protocol

1. Information turns to *packet* according to protocol speicification
2. Protocol also specifies the process
3. Infrastructure is to route the *packets* to the destination.

Packet

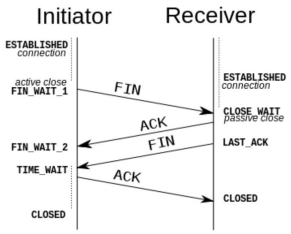
1. Information turns to packet



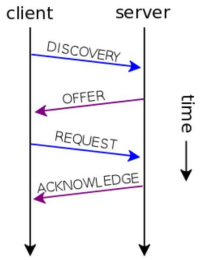
Protocol

2. Protocol designs the packet and process

TCP Session

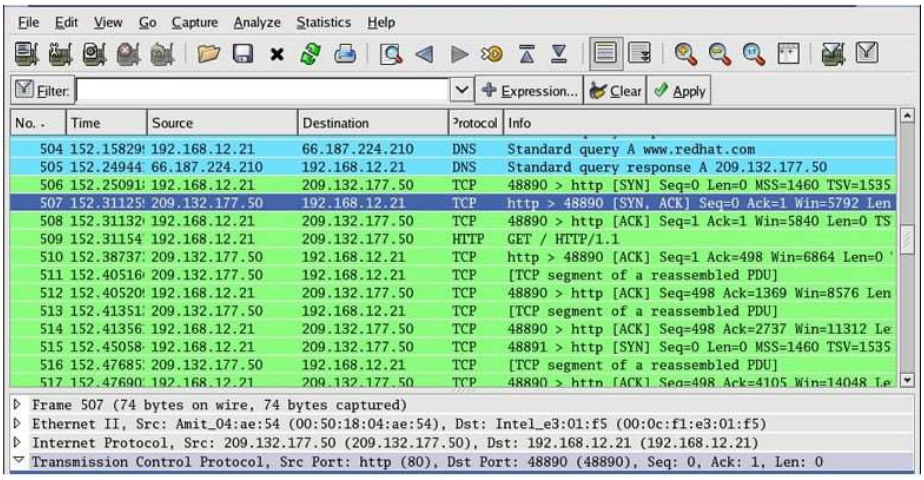


DHCP Session



Routing/Gateway

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

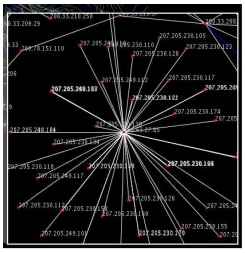


TCP/IP and Internet

- 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?

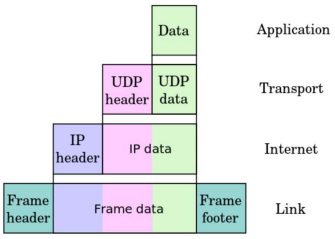
A family of protocols but what's most famous/"fundamental" is IP and TCP. 1. IP (Internet Protocol) - IP address.



2. TCP (Transmission Control Protocol) / UDP (User Datagram Protocol)
    - *TCP provides reliable, ordered, and error-checked delivery of a stream*
    - *UDP provides real-time transmission which can accept failure.*

TCP/IP

- Four Layers



- Application layer runs many protocols

Internet protocol suite

Application layer
BGP · DHCP · DNS · FTP · HTTP · IMAP · LDAP · MGCP · NNTP · NTP · POP · ONC/RPC · RTP · RTSP · RIP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP · *more...*

Transport layer
TCP · UDP · DCCP · SCTP · RSVP · *more...*

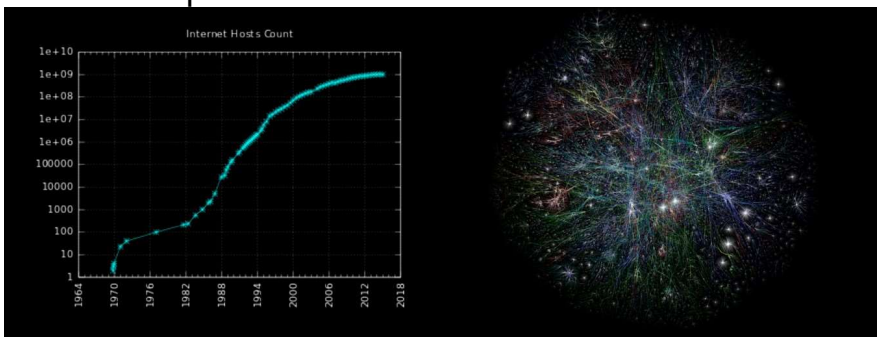
Internet layer
IP (IPv4 · IPv6) · ICMP · ICMPv6 · ECN · IGMP · IPsec · *more...*

Link layer
ARP · NDP · OSPF · Tunnels (L2TP) · PPP · MAC (Ethernet · DSL · ISDN · FDDI) · *more...*

V · T · E

This is why and how Internet growed in size

A well-defined network protocles.



IETF (Internet Engineering Task Force) maintains and still gets new protocol approved.

I E T F

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

- We have talked about what runs inside the cable?
- Every network device has a hardware address
- Dynamic Host Configuration Protocol (DHCP) protocol:
  - *DHCP client, a component of the operating system sends out a DHCP request and recieves 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.

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.*
  - *DNS server would return the actual IP address of the domain name.*
  - *One kind of attack to Internet is to hijack/brings down Root Domain Servers for Global (8 of them) or a country's root DNS.*
  - *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.

HTTP/HTML

- Now we shall have an idea of how Internet works, let's move on to Web.
- Initiated by Tim Berners-Lee at CERN (where big collison happens in "The Large Hadron Collider") in 1989.
- HTML was also invented by Tim, "HTML tags"

HTTP Request/Response

- Request

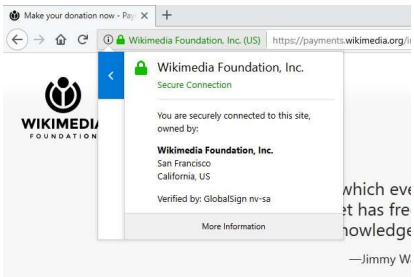
```
GET /index.html HTTP/1.1
Host: www.example.com
```

- 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

- Hypertext Transfer Protocol (HTTP) but obviously, it does file, music, anything else 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. This is about digital certificate, associated with crytography and authentication. Another topic.



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">
    <img alt="" src="//upload.wikimedia.org/wikipedia/commons/thumb/6/65/Web_Index.svg/300px-Web
        width="300" height="154" class="thumbimage"  />
    </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 deploy: no copy needed*
  - *Runs fast: every browser is optimized*
  - *Easy to develop: less effort than App and cross-platform.*

To recap

Why the Internet succeeded?

- Information flows by packet.
  - *IP protocol sets the address for the device*
  - *TCP protocol transmits the packet reliably*
  - *UDP protocol does real-time transmission which can accept failure.*
- Local device just needs to send the packet
- Nework routers/gateways does the route/transmission to the destination.
- Scalable and Efficient

To recap

Web application

- Browser => HTML/CSS/JavaScript => HTTP => TCP => IP => Network physical.
- We will write in R Shiny, which subsequently output HTML/CSS/JavaScript to be run in browser.

Lecture 2: Amazon Web Services: Launch into the Cloud

- Sign-up for AWS Account
- Setup AWS for EC2
- Launch EC2
- Running R

Lecture 2: Amazon Web Services Launch into the Cloud

Yang Ye
MFE FF8828
2018

Disclaimer

Disclaimer:

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

Content

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

Amazon Web Services

History:

While Amazon standardized its internal infrastructure for all teams, it found the opportunity of open it up to the public and make computing infrastructure available to all people. Amazon thus becomes the new utility company for the Internet age, supplying crucial infrastructure like electricity and water.

We can almost attribute the inventor of “Cloud”, “Cloud services”, “Cloud computing” to Amazon.

Now the “cloud” market has many competitor, with AWS still holds about 33% (Apr 2018)

Sign-up for AWS as a student

1. Sign-up with an AWS account at https://aws.amazon.com
- 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 applying to AWS Educate.
3. (Depends) Because NTU is not (yet) listed as a institutude, you may need to apply credit from AWS Educate separatedly https://aws.amazon.com/education/awseducate/contact-us/
4. Claim credit from AWS according according to the instruction of AWS

AWS Free Tier for New Account

AWS Free Tier (12 Month Introductory Period):

These free tier offers are only available to new AWS customers, and are available for 12 months following your AWS sign-up date. When your 12 month free usage term expires or if your application use exceeds the tiers, you simply pay standard, pay-as-you-go service rates (see each service page for full pricing details). Restrictions apply; see offer terms for more details.

Elastic Compute Cloud (EC2)

- 750 hours of Amazon EC2 Linux t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month*
- 750 hours of Amazon EC2 Microsoft Windows Server† t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month*
- 750 hours of an Elastic Load Balancer shared between Classic and Application load balancers, 15 GB data processing for Classic load balancers, and 15 LCUs for Application load balancers*
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with EBS Magnetic) and 1 GB of snapshot storage*
- 500 MB-month of Amazon EC2 Container Registry storage for new customers*

Amazon Simple Storage Service (S3)

- 5 GB of Amazon S3 standard storage, 20,000 Get Requests, and 2,000 Put Requests*

Data Transfer

- 15 GB of data transfer out and 1GB of regional data transfer aggregated across all AWS services*

Amazon Data Pipeline

- 3 low frequency preconditions running on AWS per month*
- 5 low frequency activities running on AWS per month*

Amazon ElastiCache

- 750 hours of Amazon ElastiCache cache.t2micro Node usage - enough hours to run continuously each month.*

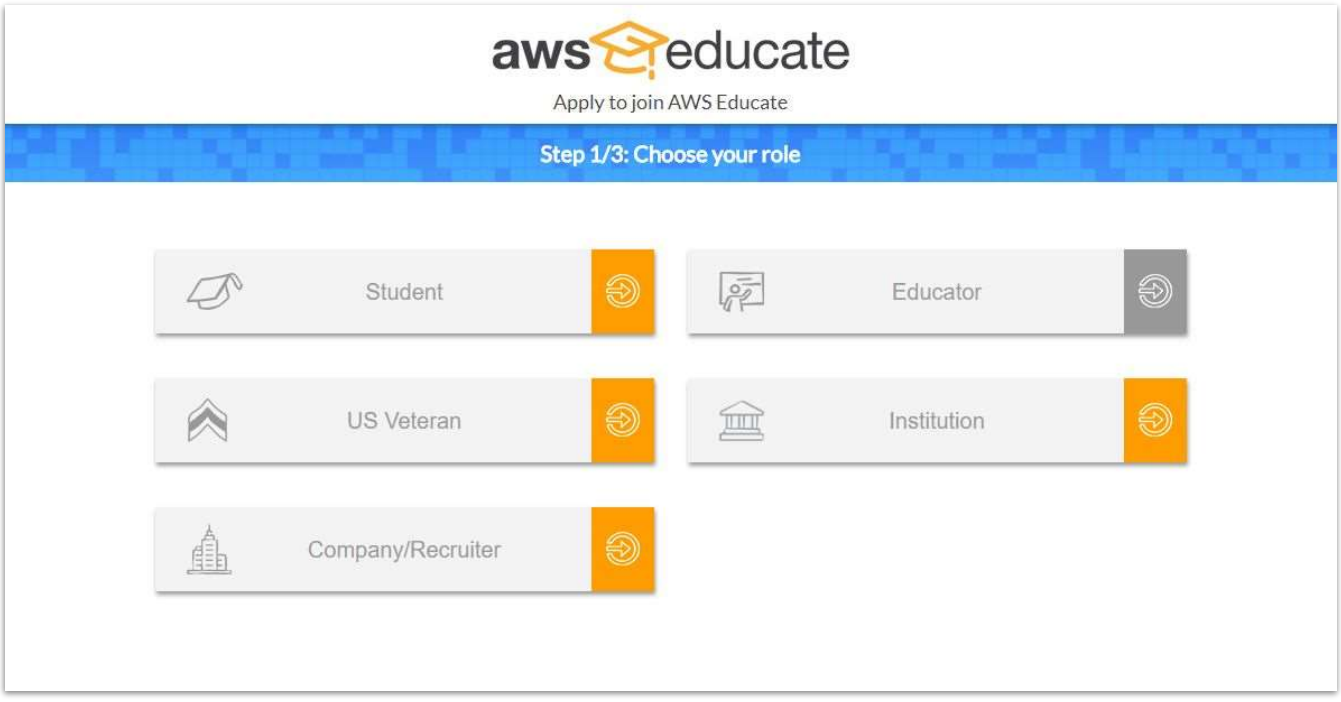
Amazon CloudFront

- 50 GB Data Transfer Out, 2,000,000 HTTP and HTTPS Requests of Amazon CloudFront*

Amazon API Gateway

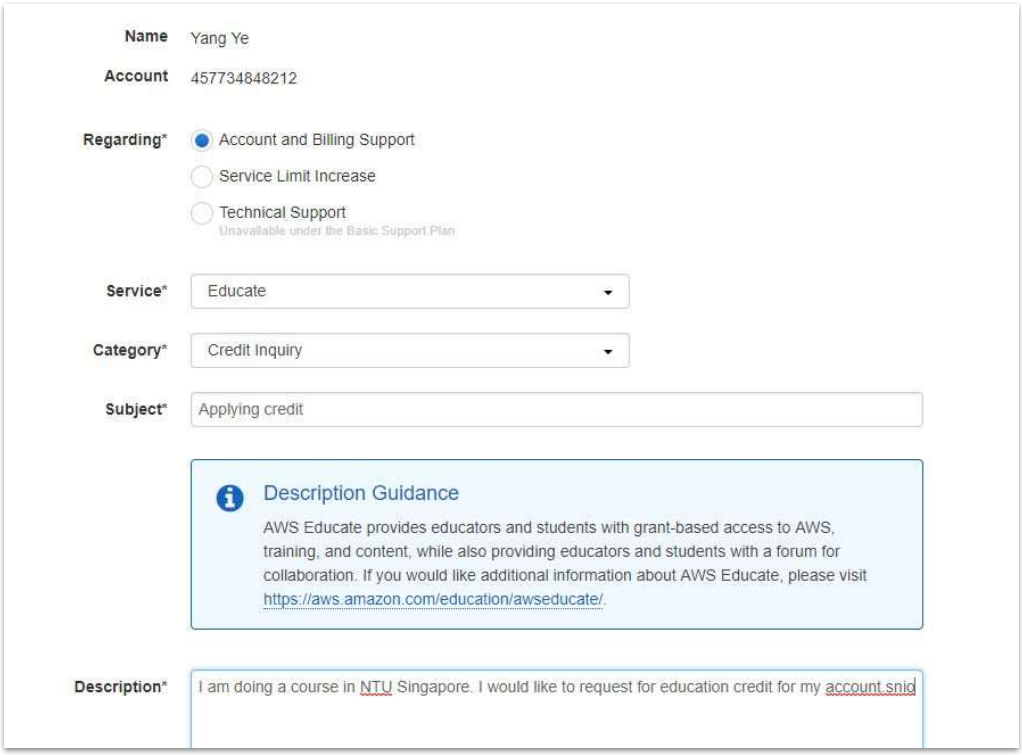
- 1 Million API Calls per month*

Register for AWS Educate account



https://www.awseducate.com/registration

Apply AWS Educate Credit



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

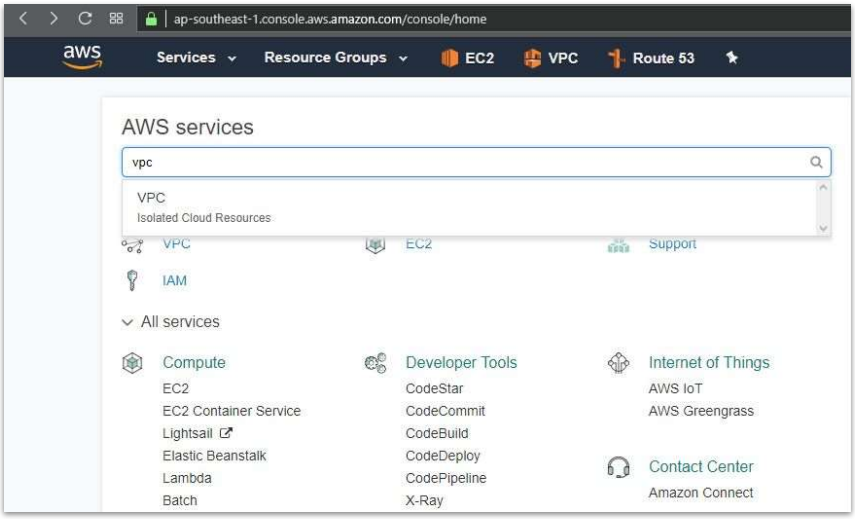
2. Setup AWS

After you login to AWS

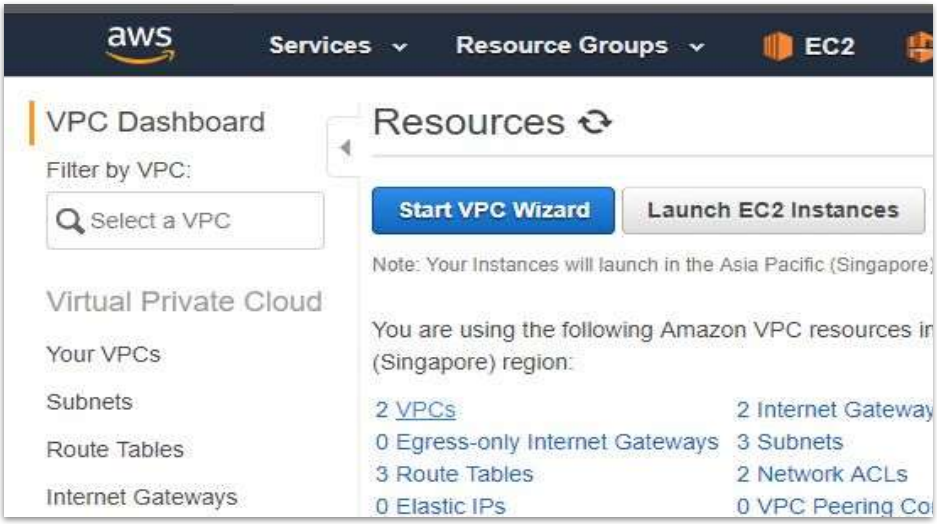
1. Change your region to **Singapore**.
2. Add shortcuts in the navigation bar
   a. **EC2**
   b. **VPC**

2. Create VPC Network

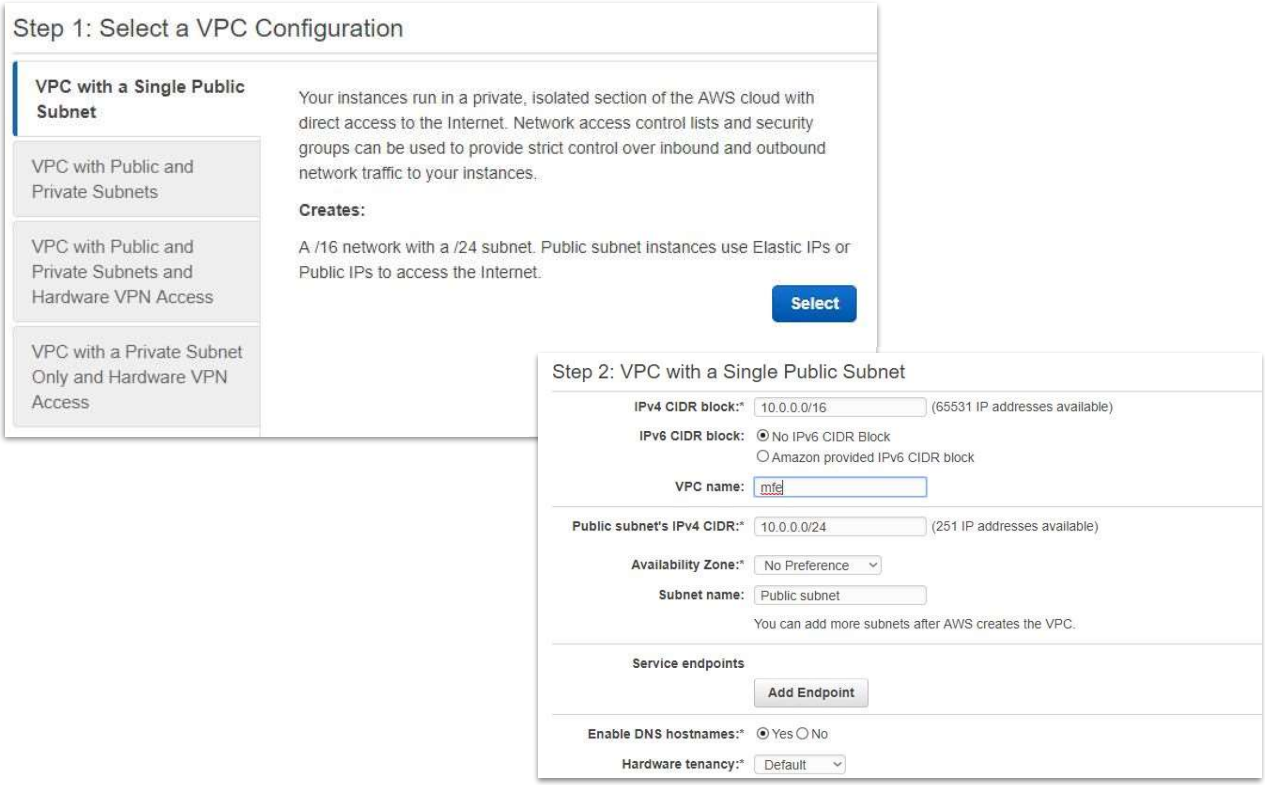
Search for VPC



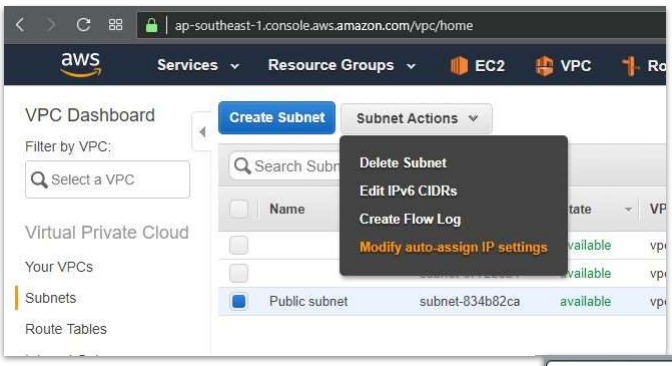
2.1 VPC: Click “Start VPC Wizard”



2.2 VPC two steps - follow defaults, add a name

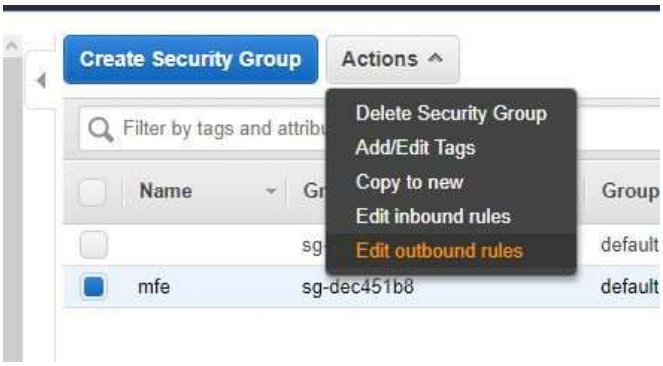


2.3 Subnet: Change settings. Tick auto-assign public IP4 address



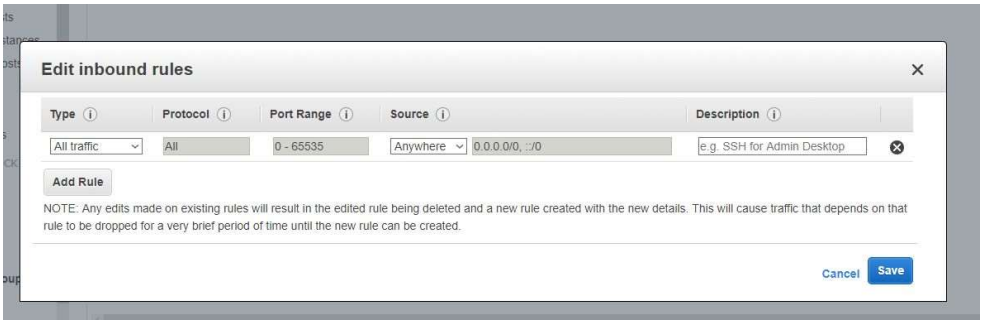
2.4 In VPC wizard, it created new security group.

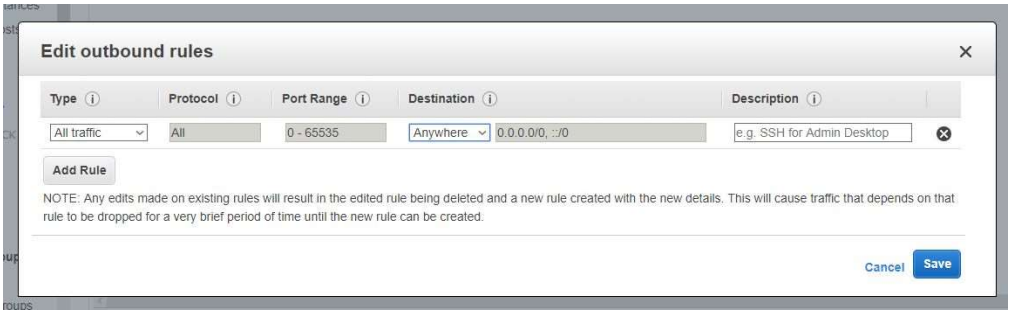
Selected the newly created security group.
Edit Inbound rules and edit outbound rules.



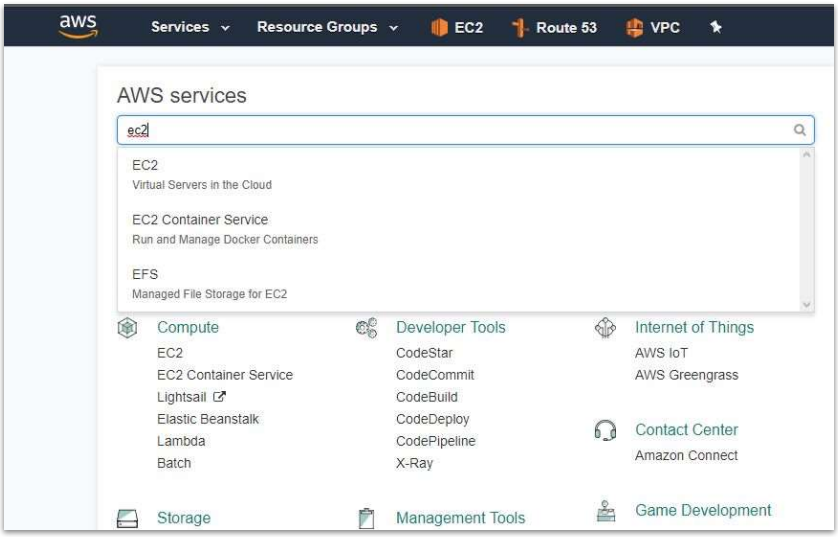
2.4 Inbound/Outbound: choose All traffic/Anywhere

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.

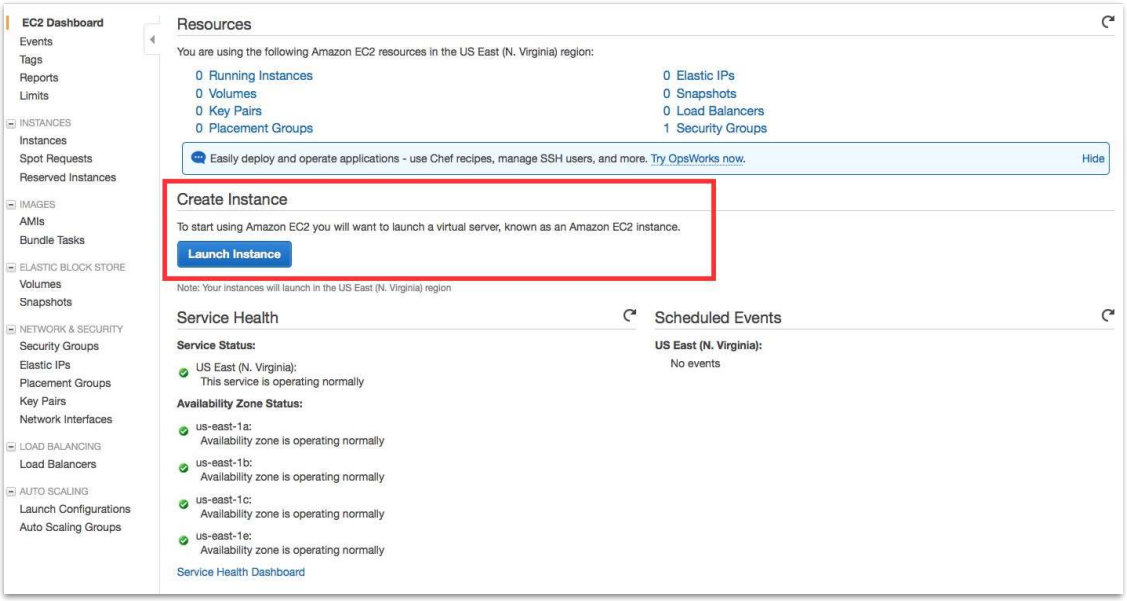




3. Go to EC2

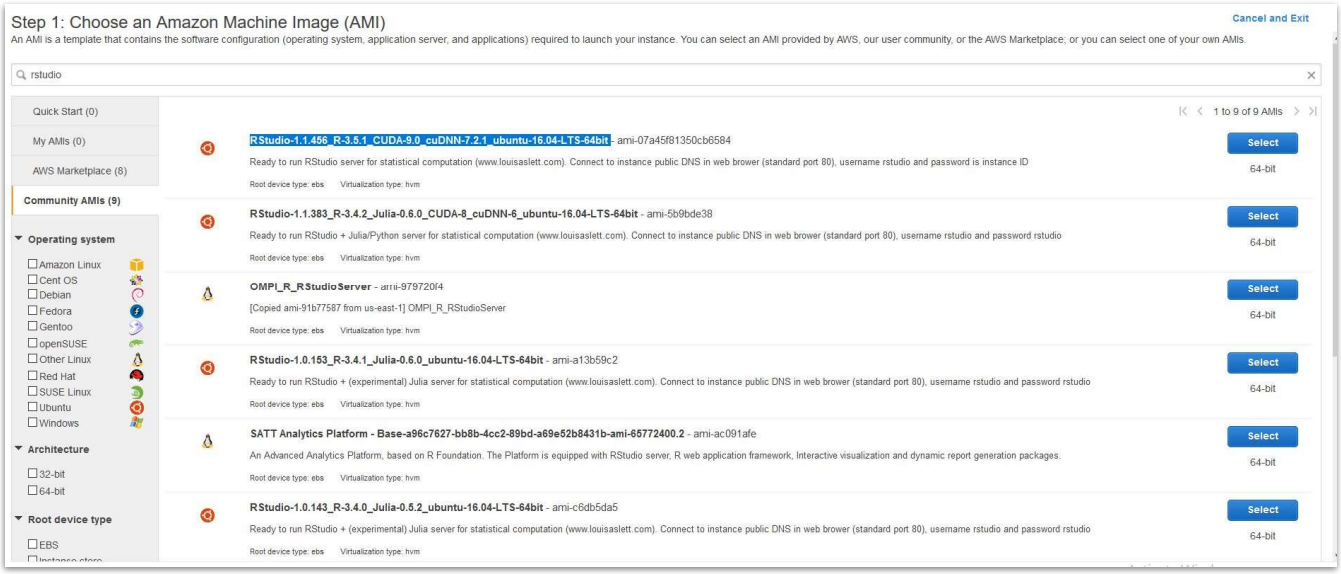


3.1 Ready to Launch



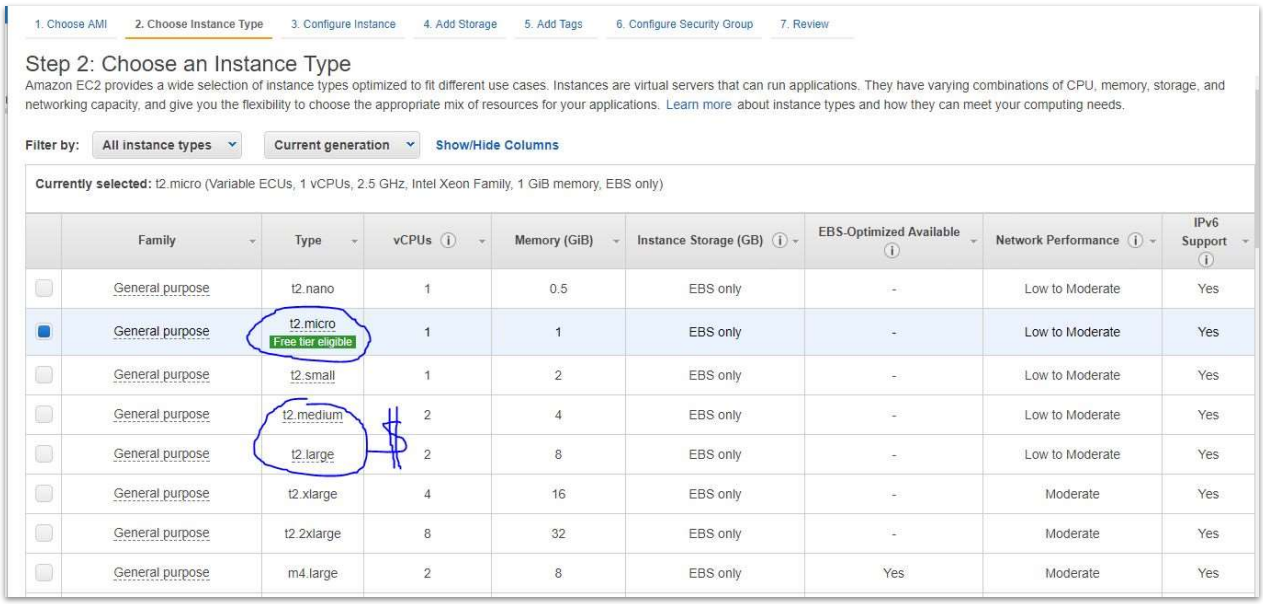
EC2 Step 1: Community AMI: search for “RStudio”

RStudio-1.1.456_R-3.5.1_CUDA-9.0_cuDNN-7.2.1_ubuntu-16.04-LTS-64bit
Id: ami-07a45f81350cb6584 in Singapore region
created by http://www.louisaslett.com/RStudio_AMI/
Newer than below

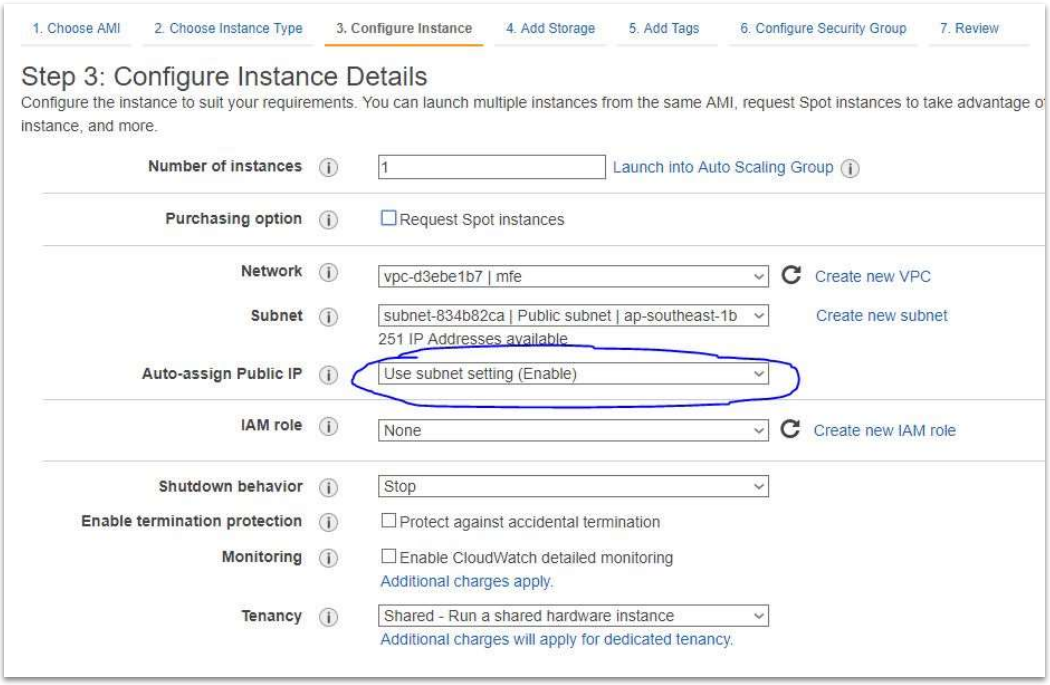


EC2 Step 2: Choose instance type. You can enjoy having one instance of **t2.micro** during 12-month free-tier.

- 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.



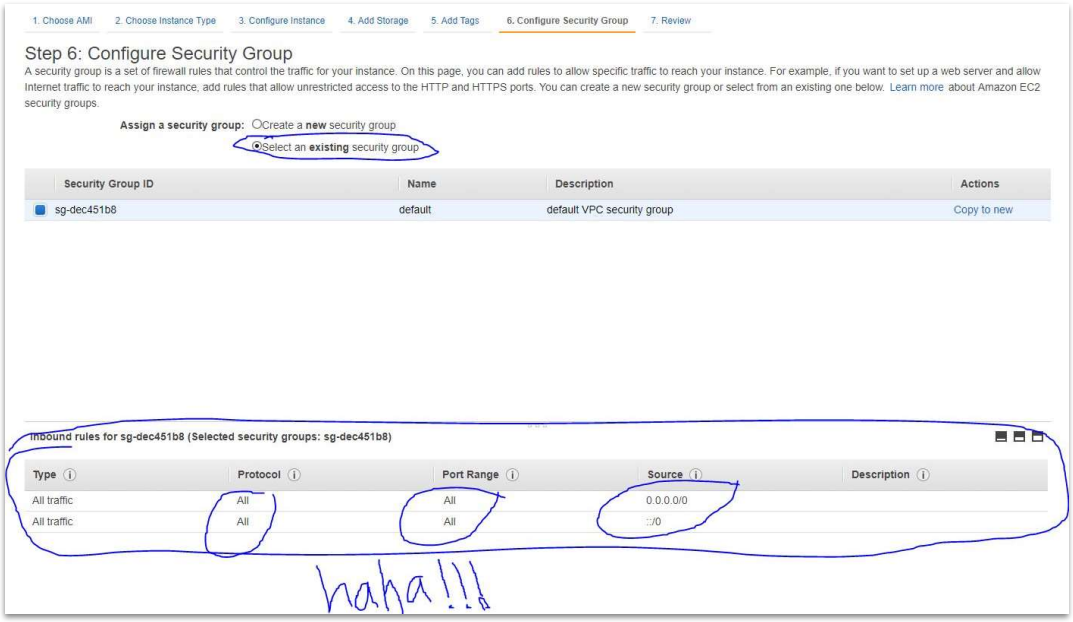
EC2 Step 3: Make sure Auto-assign Public IP is ticked. Click “Next”.



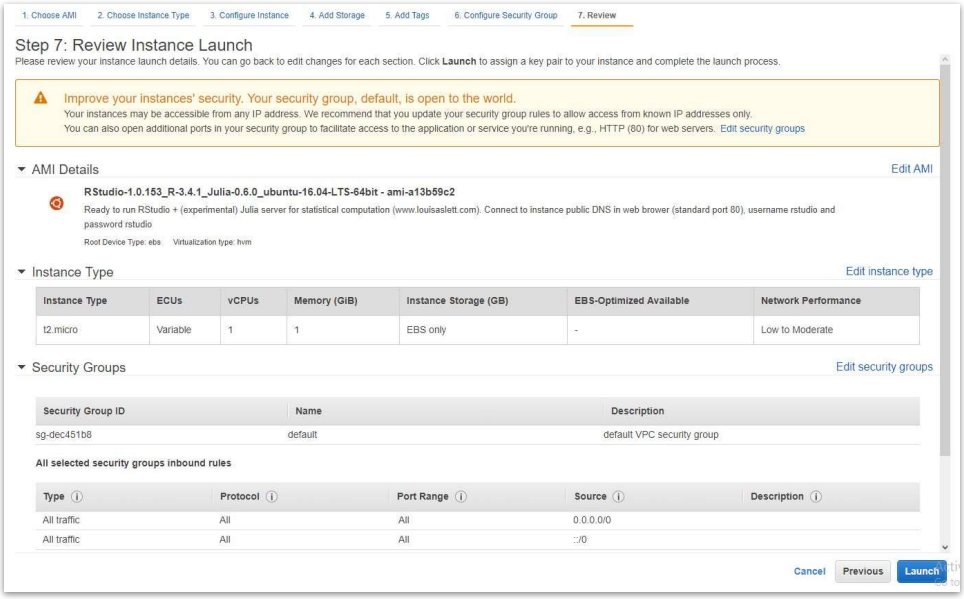
EC 2 Step 4: Add Storage, use the default setting.

- 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.

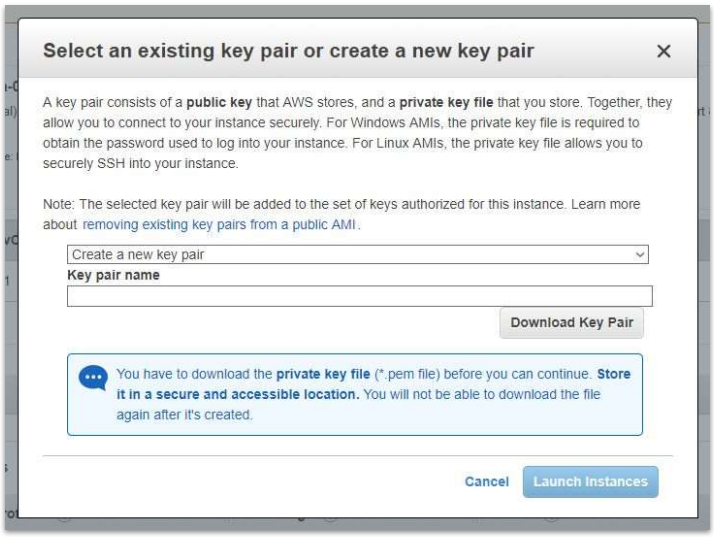
EC Step 6: Click “6. Configure Security Group” Select an existing security group. Your previous changes are loaded.



EC 2 Step 7: Review and Launch
Click the blue button.



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



If you know SSH

You can connect to the server by

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

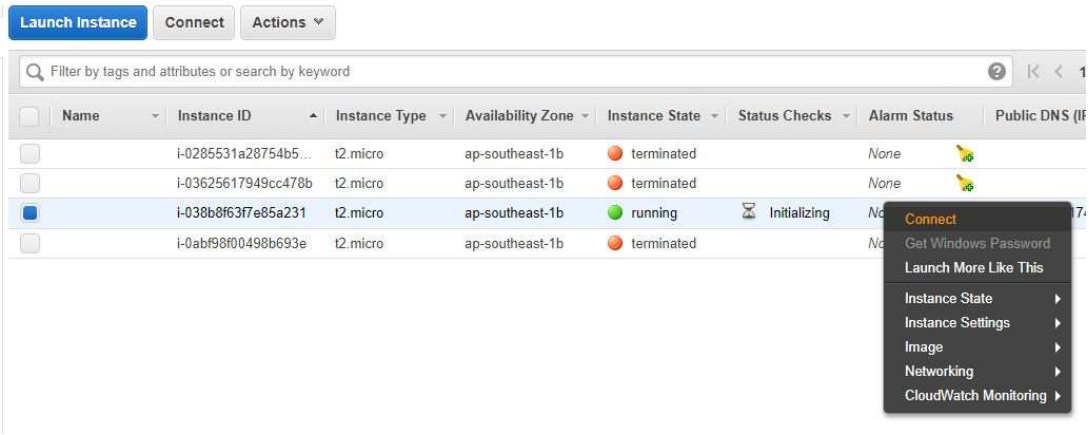
ssh -i 'c:\Users\yourusername\Downloads\MyKeyPair.pem' ubuntu@{IP_Address}

- Mac: skip download Git. Go straight

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

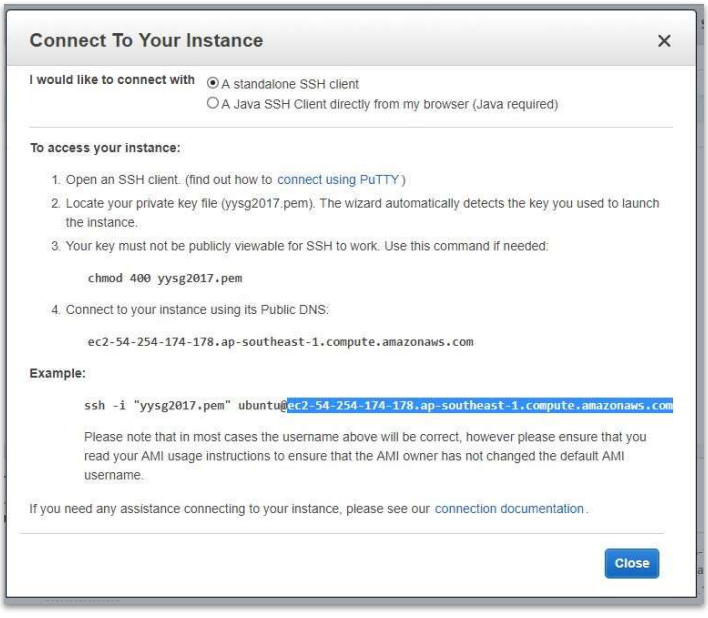
Control the instance.

- 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

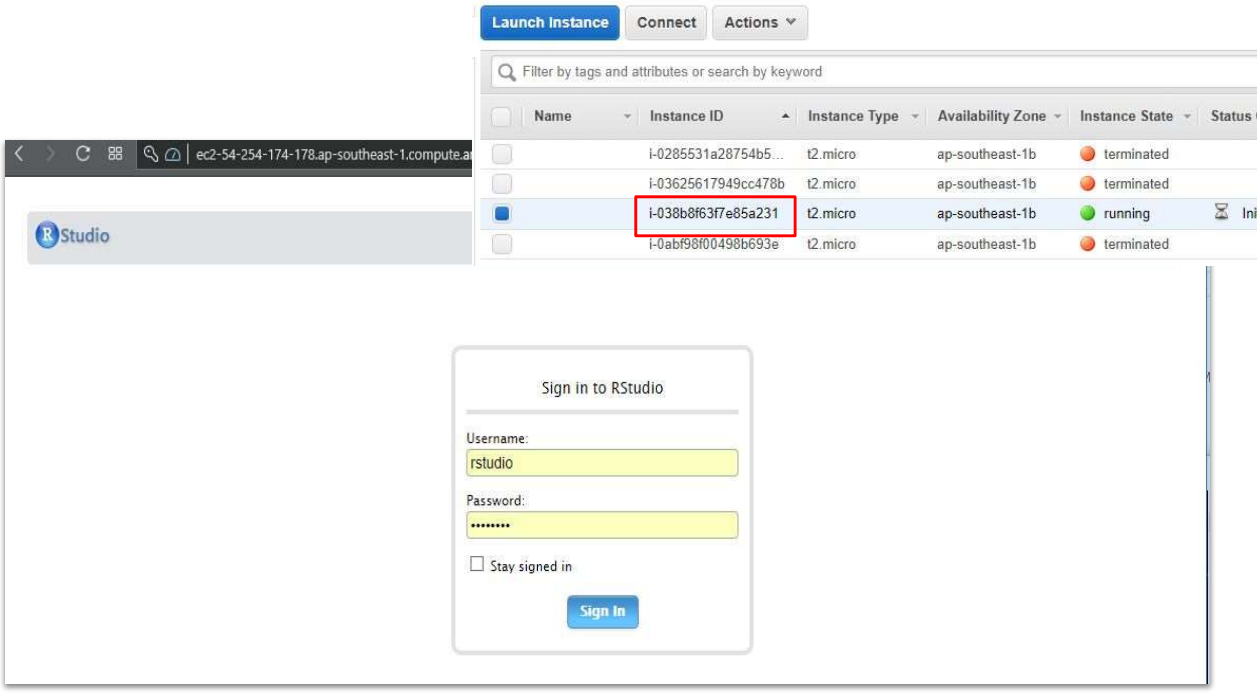
- Copy this address



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

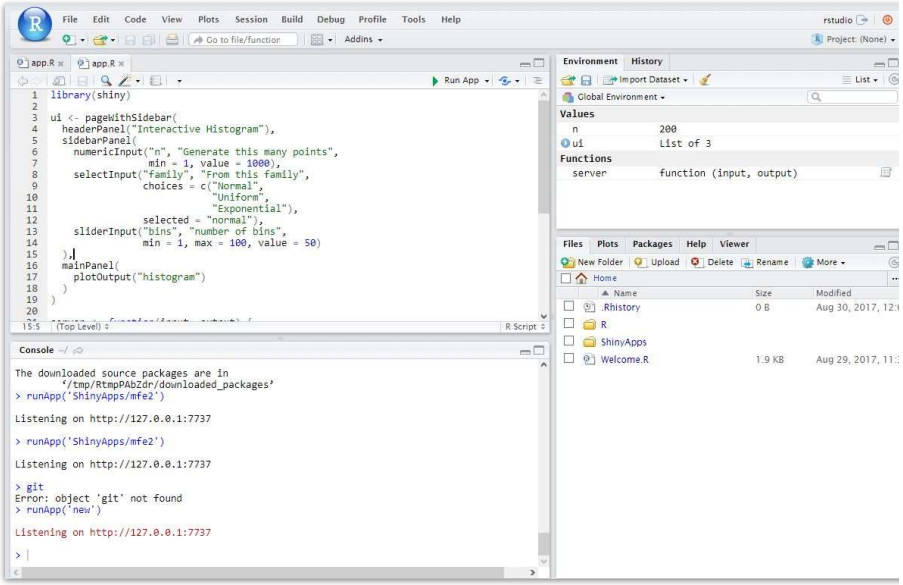
This would change every time when you launch it.

Initial password: rstudio/Instance ID.

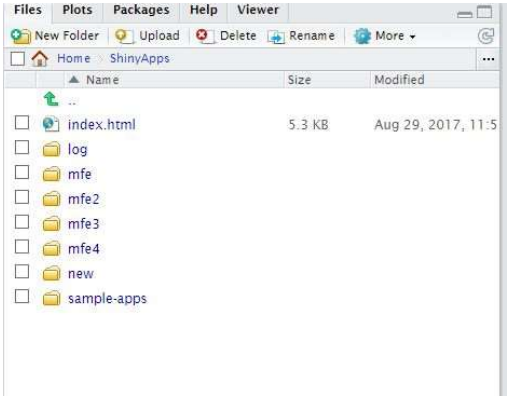


RStudio Setup

- 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://ec2-13-229-181-28.ap-southeast-1.compute.amazonaws.com/shiny/rstudio/sample-apps/hello/

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.
- It's cool to have something running in the cloud. You can show people to impress.

- AWS setup is optional in this course.
- You can run R Studio and application on your laptop
- If you finishing working on it, leave the instance as Stopped. Be aware of how much you spend on AWS.

Lecture 3: R Markdown and Shiny/1: 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 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 Output 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: 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
```

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

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.

Markdown: Inline formatting

Strikthrough

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

Superscripts and subscripts

H~2~O is a liquid. 2^10^ is 1024. H_2O is a liquid. 2^{10} is 1024.

Verbatim. inline code

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

Note:

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

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

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

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

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, 2018"
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
```

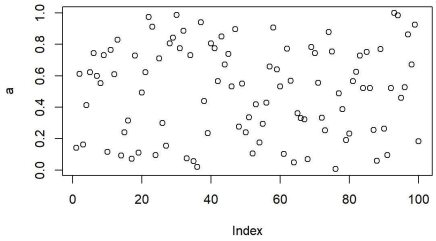
- Calculate_7 is the chunk name. It's optional to give a chunk name. If included, each code chunk needs a distinct name. It's usually best to give each code chunk a name, for easier debug.
- R code can also be inline. For example, to generate a random number everytime, include this `runif(1, 0, 1)`, 0.5523454.

Chunk options

- *echo* is to decide whether to display code, default is FALSE.
- *results* is to decide whether to display result, default is "markup", 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)
```
```



R Markdown example: Table

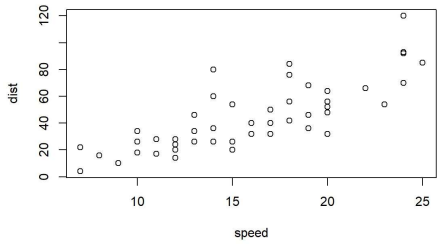
```
```{r kable}
knitr::kable(
  mtcars[1:5, ],
  caption = "A knitr kable."
)
```
```

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 Markdown: Practice

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. (shiny-1-empty.R)

```
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 (shiny-2-sidebar.R)

```
library(shiny)

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

  fluidPage(sidebarLayout(
    sidebarPanel("This is a panel on the side"),
    mainPanel("This is the main panel")
  )),

  fluidPage(sidebarLayout(
    sidebarPanel("This is a panel on the side"),
    mainPanel("This is the main panel")
  ))
)

# 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 parameters.

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

Add some items

- `titlePanel("Hello Shiny!"),h1("Introduction to Layout"), h2("Sidebar Layout")` (shiny-3-sidebar-min.R)

```
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"),
      ),
      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

Sidebar with more tags (shiny-3-sidebar.R)

```
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)
```

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()` (shiny-4-wellPanel.R)

```
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 (shiny-5-navPanel.R)

```
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 (shiny-6-tabPanel.R)

```
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 (shiny-7-navbar.R)

```
library(shiny)

ui <- fluidPage(
  fluidPage(
    navbarPage(title = "Runchee Technology",
               tabPanel("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 (shiny-8-column.R)

- 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 (shiny-10-composite.R)

```
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))),
               selectInput('facet_col', 'Facet Column', c(None='.', names(diamonds)))
        )
```



```
      )
    )
  )
)

# 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 (shiny-mfe-example.Rmd)

````
---
title: "MFE FE8828 Assignment 1"
date: 2018-11-03
output: html_document
runtime: shiny
---

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

# Use echo = TRUE for assignment is an exception, so code is visible.
```{r, echo = TRUE}
wellPanel("Inputs",
          numericInput("fav_num", "What's your favorite number?", 3))
```
````

Inputs

What's your favorite number? 3

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!*

Runchee Technology Product About Us Contact Us

Hello!

One more thing!

New breakthrough

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

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