

# **FE8828 Programming Web Applications in Finance**

**Week I**

**What's Internet? What's Web?**

**Launch into the Cloud**

**Catch-up with R**

Dr. Yang Ye <Email:[yy@runchee.com](mailto:yy@runchee.com)>

Nov 02, 2017

# Introduction: Topics for Week 1-3

## ■ 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 applicaiton.*
- Week 7:
  - *Presentation of Assignment: Blockchain.*

# Introduction: Assingments

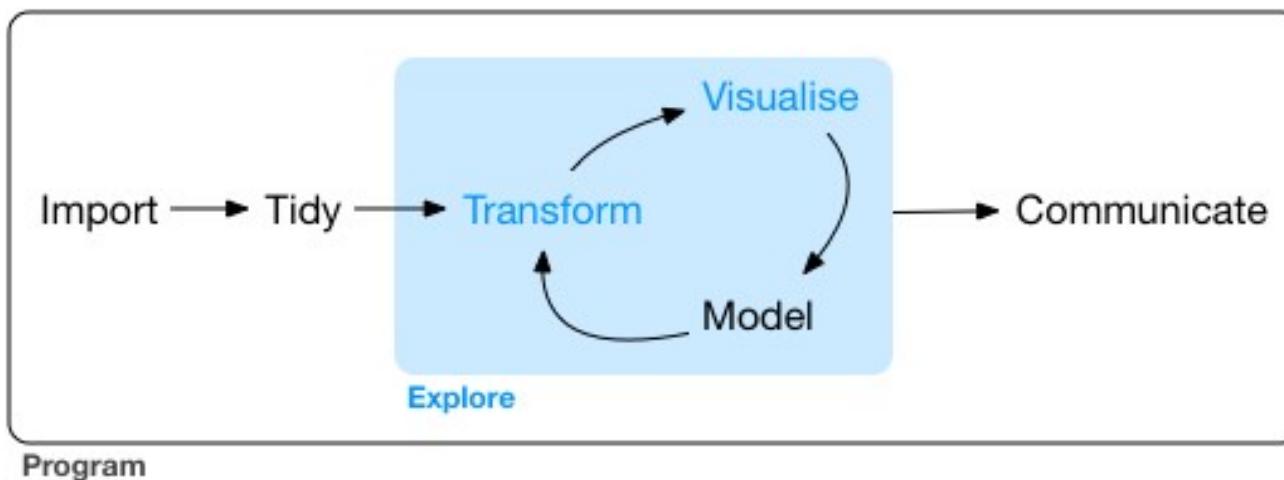
- Week 1: A static website. A front page, a about page and a description page. (due by Week 2)
- Week 2: A dynamic website that can do CRUD. (due by Week 3)
- Week 3: A data-driven website that does data analytics. (due by Week 4)
- Week 4: A finance-data application website. (due by Week 6)
- Week 5: 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.

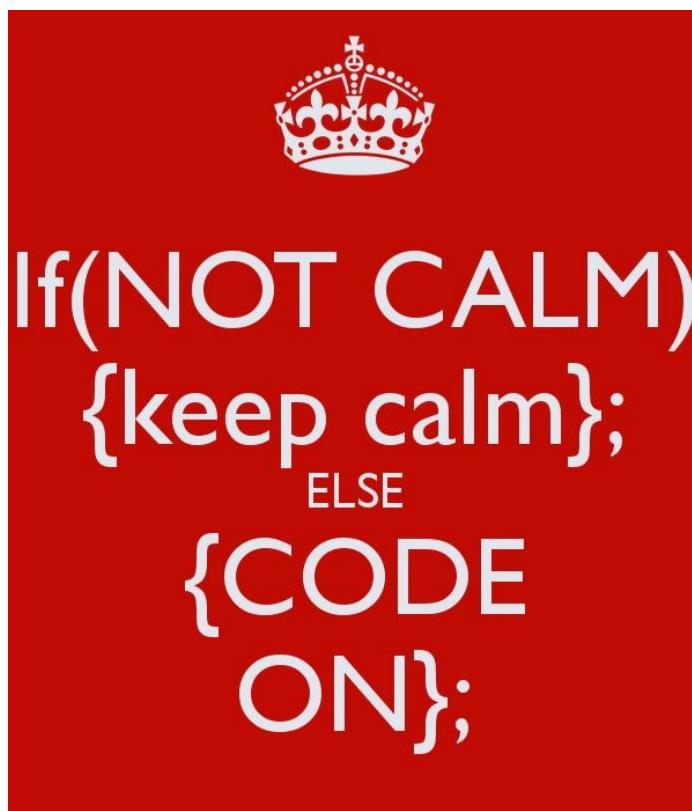
# Where does this course stands on?

- 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.*

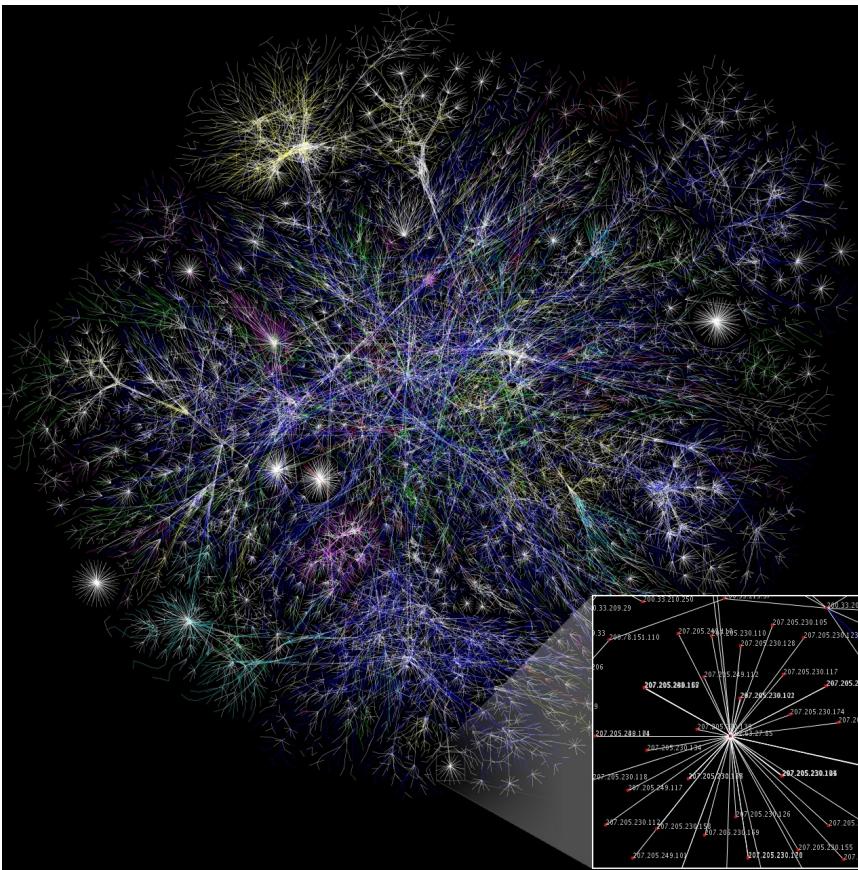
# Keep calm and code on



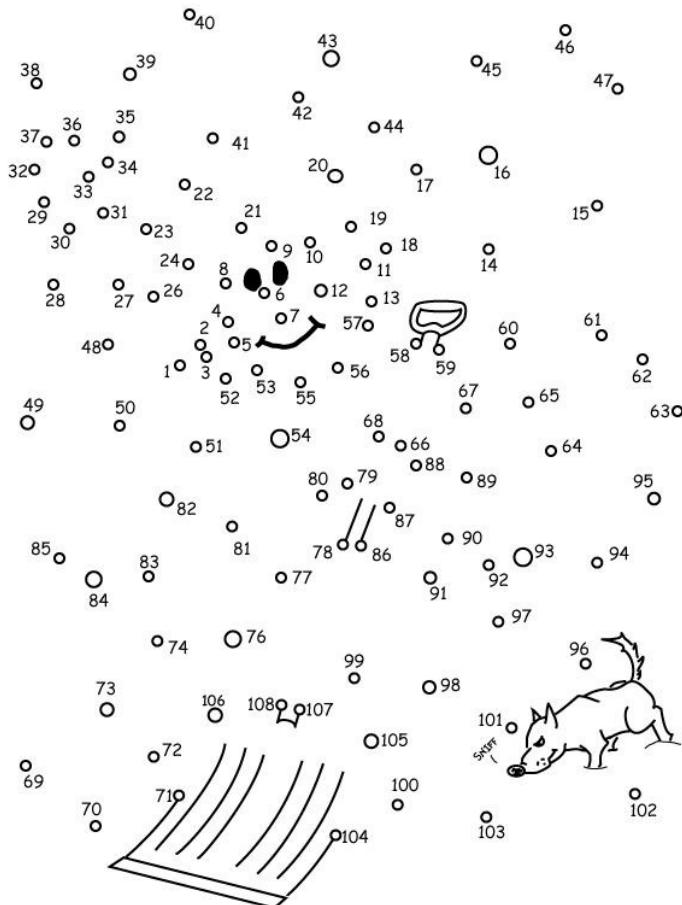
# Lecture 01: What's Internet? What's Web?

- Network
- Internet
- HTTP/HTML/Web

# Network

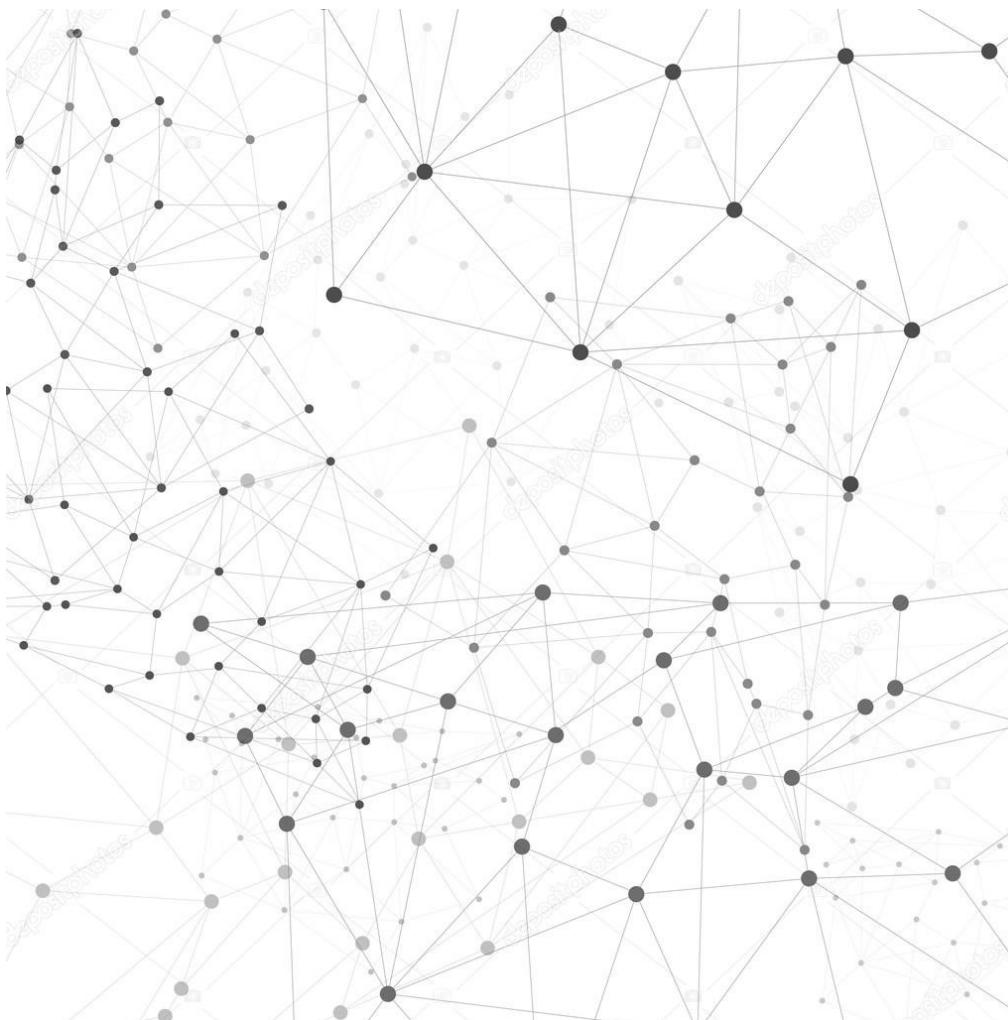


# Network is to connect the dots



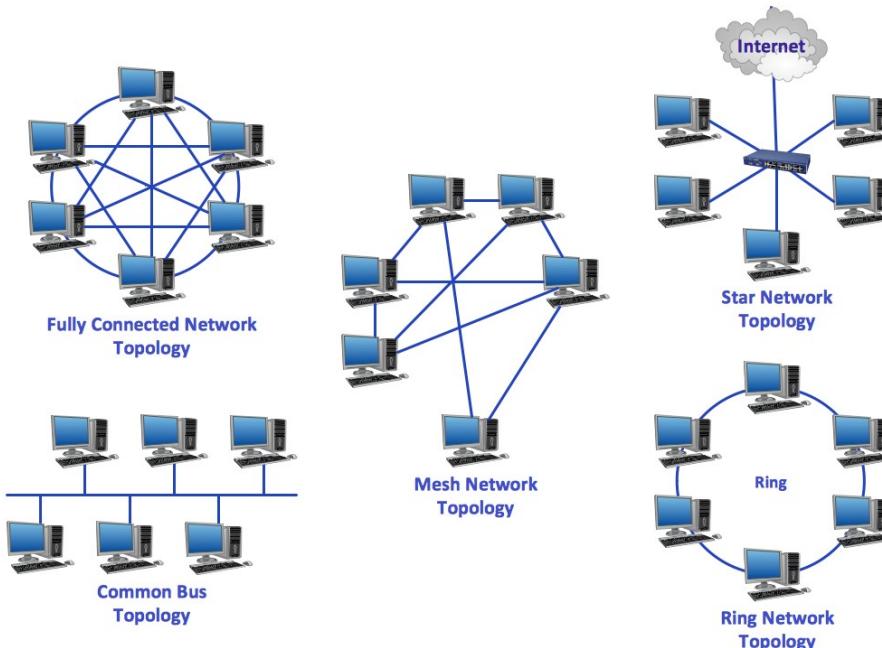
Directions (you'll need them) are available at [www.ethanham.com/blog/dots/directions.pdf](http://www.ethanham.com/blog/dots/directions.pdf)

# Network is to connect the dots devices



# Building network

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



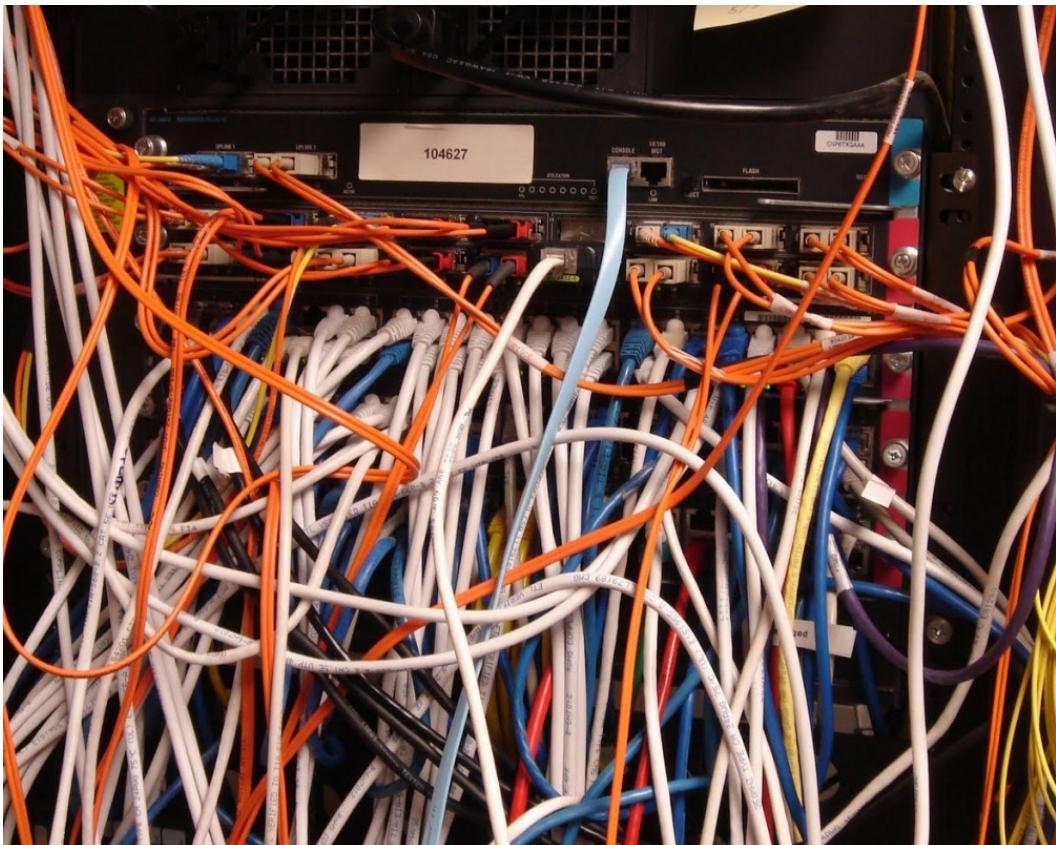
- Which network topology is our home Wi-Fi?

# 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?

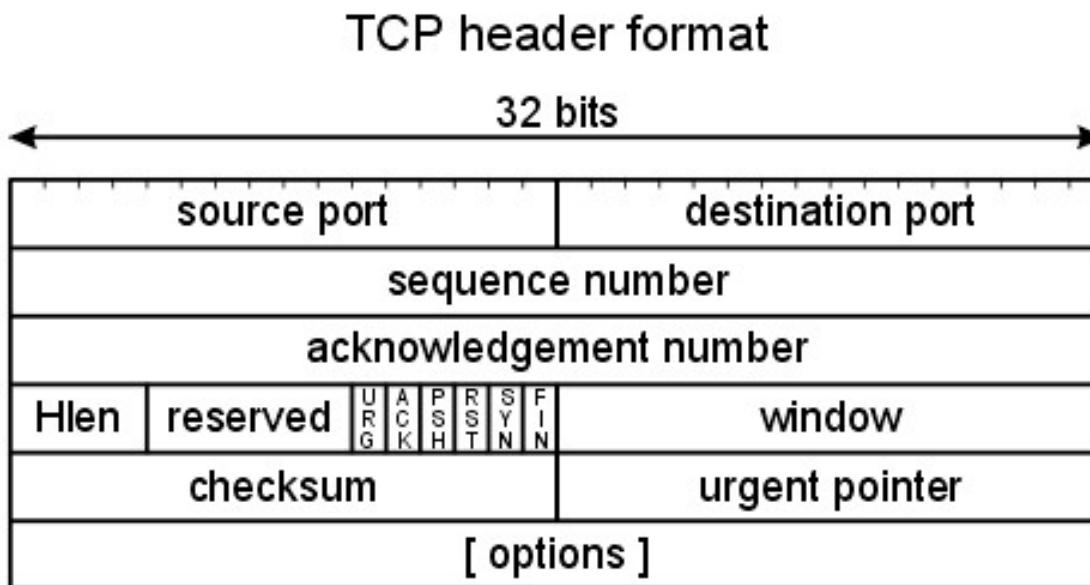


# Network talks

1. Information turns to *packet*
2. Protocol designs the packet and process
3. Infrastructure is to route the packets to the destination.

# Packet

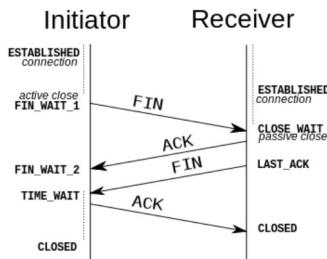
## I. Information turns to packet



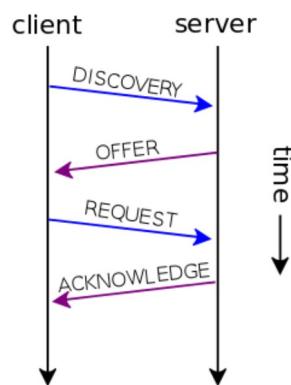
# Protocol

## 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.

The screenshot shows a Wireshark interface with a list of network frames. The frames are color-coded by protocol: blue for DNS, green for TCP, and yellow for HTTP. The table below provides a detailed view of the captured frames:

No.	Time	Source	Destination	Protocol	Info
504	152.158291	192.168.12.21	66.187.224.210	DNS	Standard query A www.redhat.com
505	152.249444	66.187.224.210	192.168.12.21	DNS	Standard query response A 209.132.177.50
506	152.250911	192.168.12.21	209.132.177.50	TCP	48890 > http [SYN] Seq=0 Len=0 MSS=1460 TSV=1535
507	152.311251	209.132.177.50	192.168.12.21	TCP	http > 48890 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len
508	152.311321	192.168.12.21	209.132.177.50	TCP	48890 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 TS
509	152.311541	192.168.12.21	209.132.177.50	HTTP	GET / HTTP/1.1
510	152.387371	209.132.177.50	192.168.12.21	TCP	http > 48890 [ACK] Seq=1 Ack=498 Win=6864 Len=0
511	152.405161	209.132.177.50	192.168.12.21	TCP	[TCP segment of a reassembled PDU]
512	152.405201	192.168.12.21	209.132.177.50	TCP	48890 > http [ACK] Seq=498 Ack=1369 Win=8576 Len
513	152.413511	209.132.177.50	192.168.12.21	TCP	[TCP segment of a reassembled PDU]
514	152.413561	192.168.12.21	209.132.177.50	TCP	48890 > http [ACK] Seq=498 Ack=2737 Win=11312 Le
515	152.450581	192.168.12.21	209.132.177.50	TCP	48891 > http [SYN] Seq=0 Len=0 MSS=1460 TSV=1535
516	152.476851	209.132.177.50	192.168.12.21	TCP	[TCP segment of a reassembled PDU]
517	152.476901	192.168.12.21	209.132.177.50	TCP	48890 > http [ACK] Seq=498 Ack=4105 Win=14048 Le

Frame details:

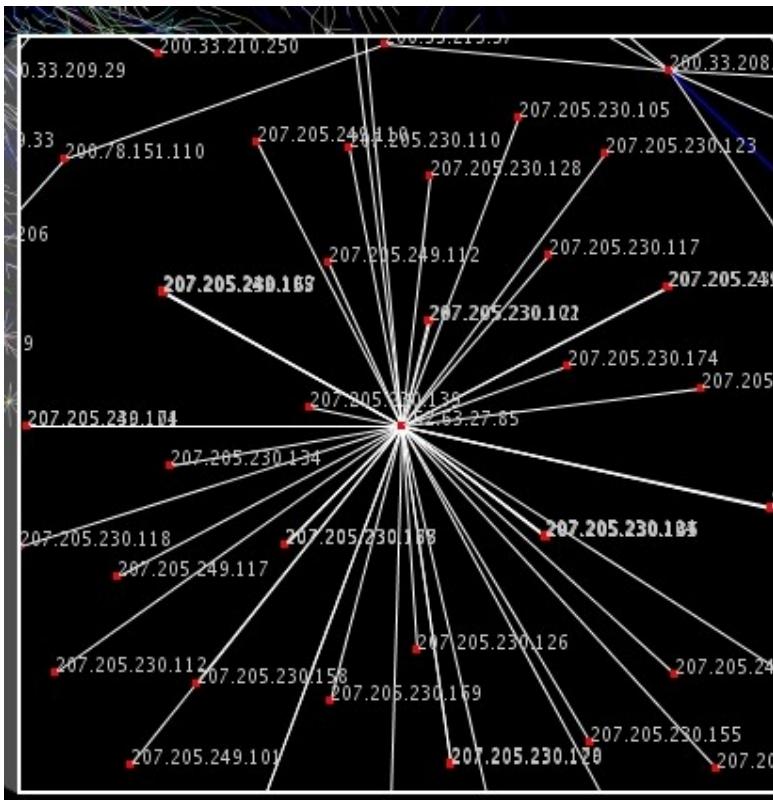
- ▶ Frame 507 (74 bytes on wire, 74 bytes captured)
- ▶ Ethernet II, Src: Amit\_04:ae:54 (00:50:18:04:ae:54), Dst: Intel\_e3:01:f5 (00:0c:f1:e3:01:f5)
- ▶ Internet Protocol, Src: 209.132.177.50 (209.132.177.50), Dst: 192.168.12.21 (192.168.12.21)
- ▼ Transmission Control Protocol, Src Port: http (80), Dst Port: 48890 (48890), Seq: 0, Ack: 1, Len: 0

# 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?

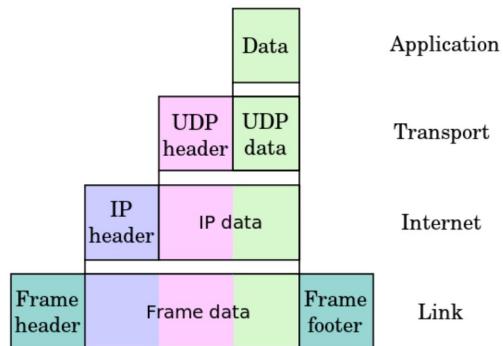
## I. IP (Internet Protocol, IP address)



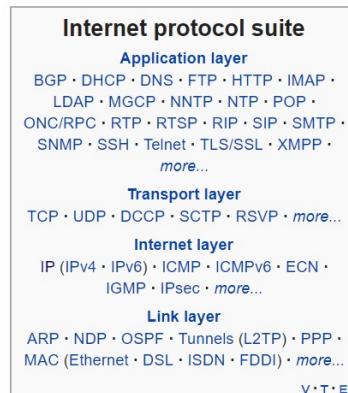
2. TCP (Transmission Control Protocol) TCP provides reliable, ordered, and error-checked delivery of a stream

# TCP/IP

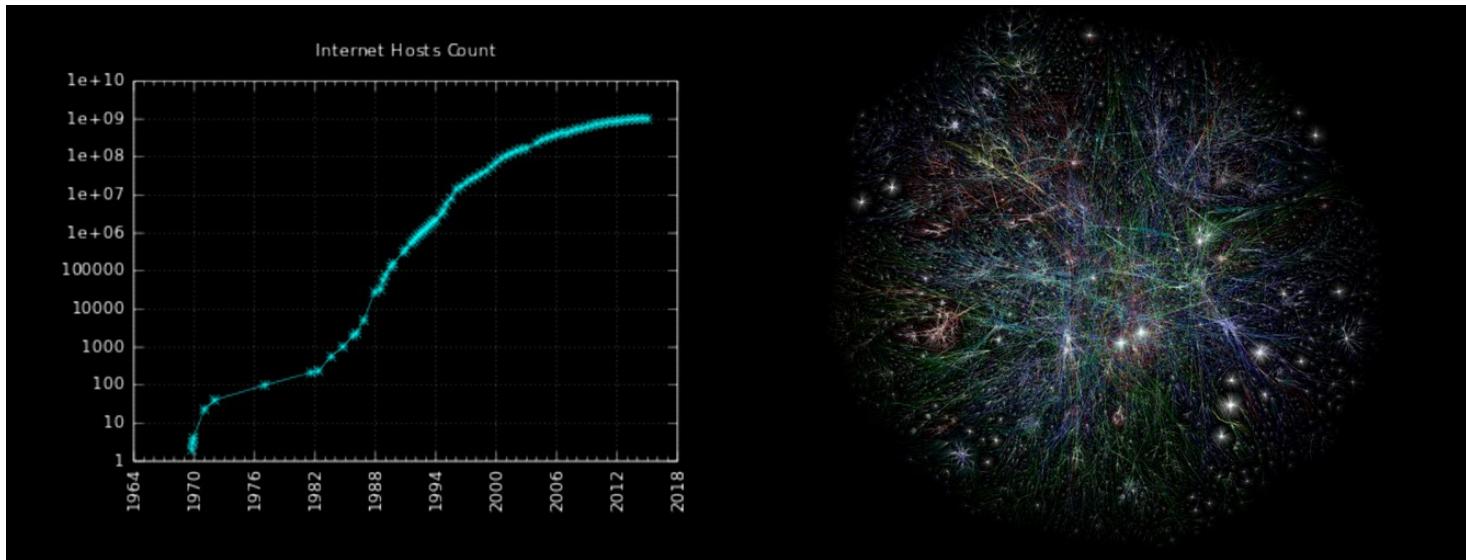
## ■ Four Layers



## ■ Application layer runs many protocols



# This is why and how Internet growed in size



# 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.

# 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.



# HTTP/HTML

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

# HTTP Request/Response

## ■ Request

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

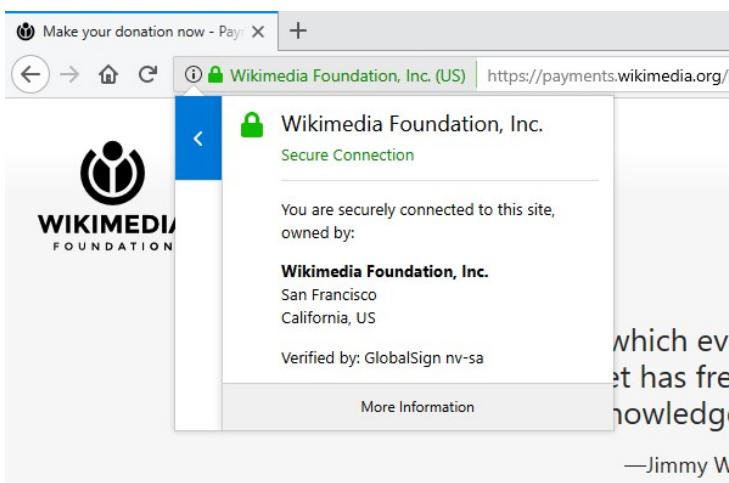
## ■ Response

```
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 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](https://en.wikipedia.org/wiki/World_Wide_Web)
- Resource: [https://en.wikipedia.org/wiki/World\\_Wide\\_Web#/media  
/File:Web\\_Index.svg](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.

# Launch in the cloud: AWS

Yang Ye  
MFE FF8828  
2017

# Content

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

# 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 institutude, 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



Apply to join AWS Educate

Step 1/3: Choose your role



Student



Educator



US Veteran



Institution



Company/Recruiter



<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\*

Educate

Category\*

Credit Inquiry

Subject\*

Applying credit



## 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\*

I am doing a course in NTU Singapore. I would like to request for education credit for my account snid

## 2. Setup AWS

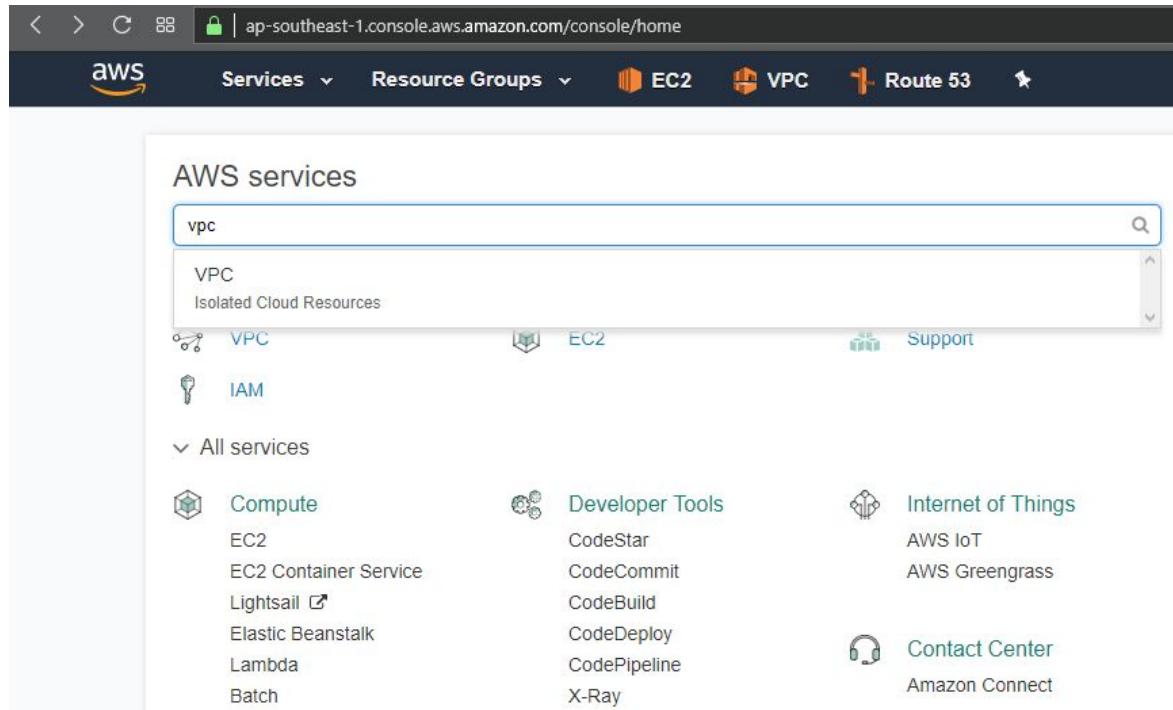
After you login to AWS

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”

The screenshot shows the AWS VPC Dashboard. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and 'EC2' icon. On the left, there's a sidebar titled 'Virtual Private Cloud' with links for 'Your VPCs', 'Subnets', 'Route Tables', and 'Internet Gateways'. The main area is titled 'Resources' with a refresh icon. It features two prominent buttons: 'Start VPC Wizard' (in blue) and 'Launch EC2 Instances' (in grey). Below these buttons, a note says 'Note: Your Instances will launch in the Asia Pacific (Singapore) region'. Underneath, it lists Amazon VPC resources in the Singapore region: 2 VPCs, 0 Egress-only Internet Gateways, 3 Route Tables, 0 Elastic IPs, 2 Internet Gateways, 3 Subnets, 2 Network ACLs, and 0 VPC Peering Co.

VPC Dashboard

Filter by VPC:

Select a VPC

Virtual Private Cloud

Your VPCs

Subnets

Route Tables

Internet Gateways

Resources

Start VPC Wizard

Launch EC2 Instances

Note: Your Instances will launch in the Asia Pacific (Singapore)

You are using the following Amazon VPC resources in (Singapore) region:

2 VPCs	2 Internet Gateways
0 Egress-only Internet Gateways	3 Subnets
3 Route Tables	2 Network ACLs
0 Elastic IPs	0 VPC Peering Co

## 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:**\*  (65531 IP addresses available)

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

**VPC name:**

**Public subnet's IPv4 CIDR:**\*  (251 IP addresses available)

**Availability Zone:**\*

**Subnet name:**

You can add more subnets after AWS creates the VPC.

---

**Service endpoints**

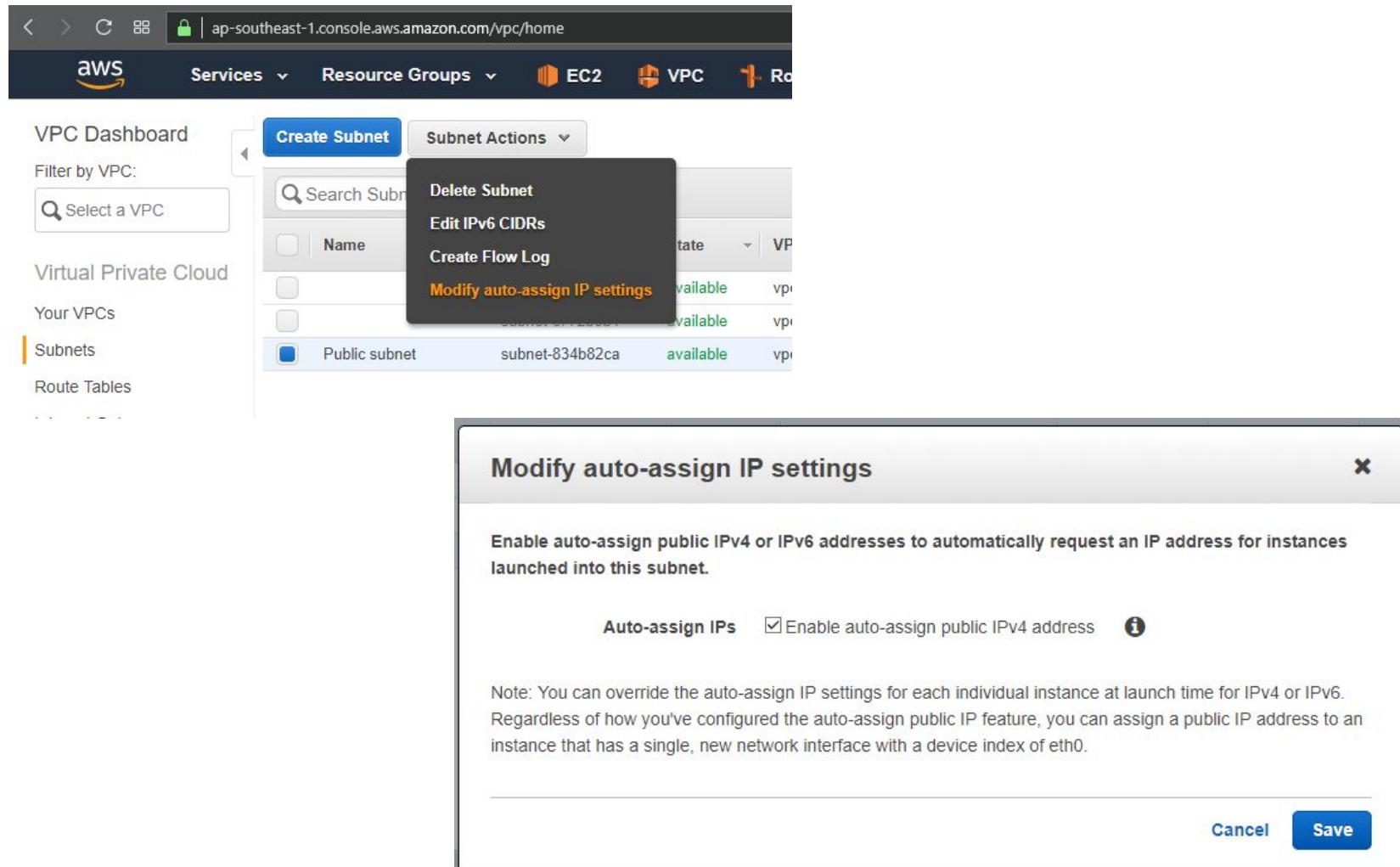
**Add Endpoint**

---

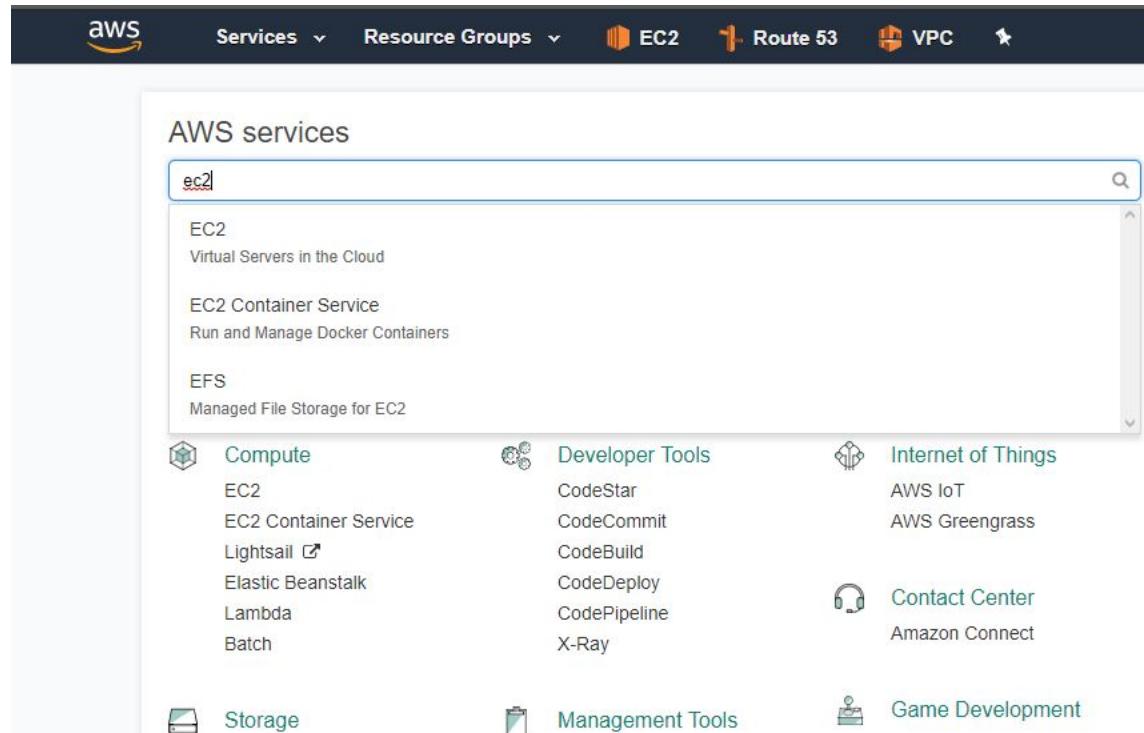
**Enable DNS hostnames:**\*  Yes  No

**Hardware tenancy:**

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

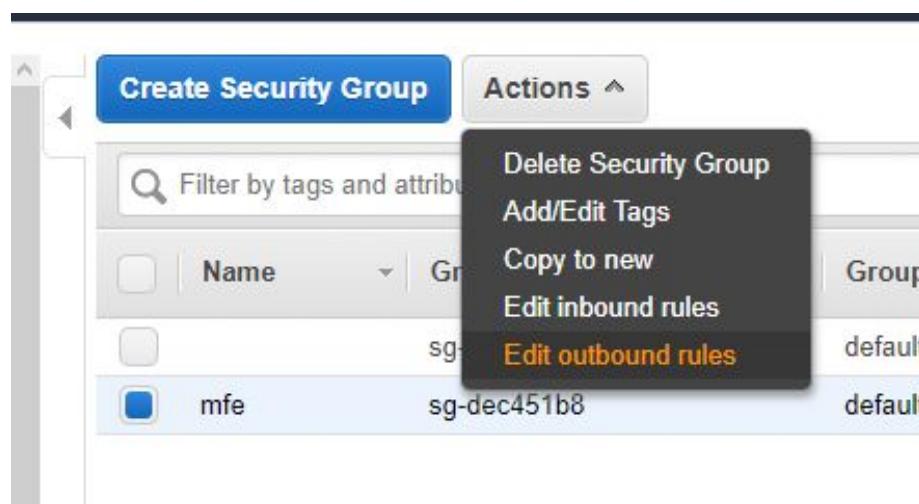


### 3. Go to EC2



3.1 In VPC wizard, it created new security group.

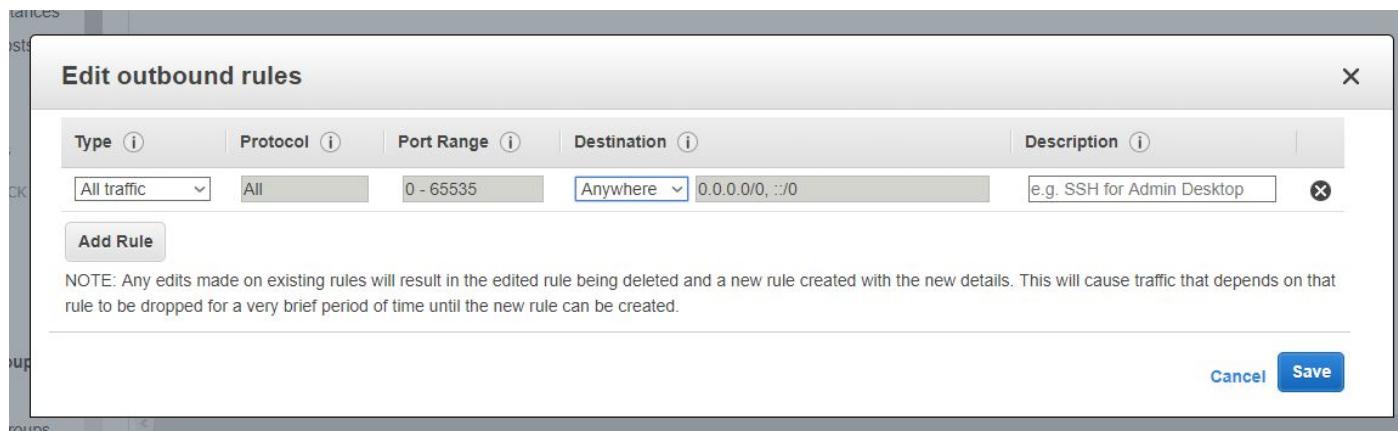
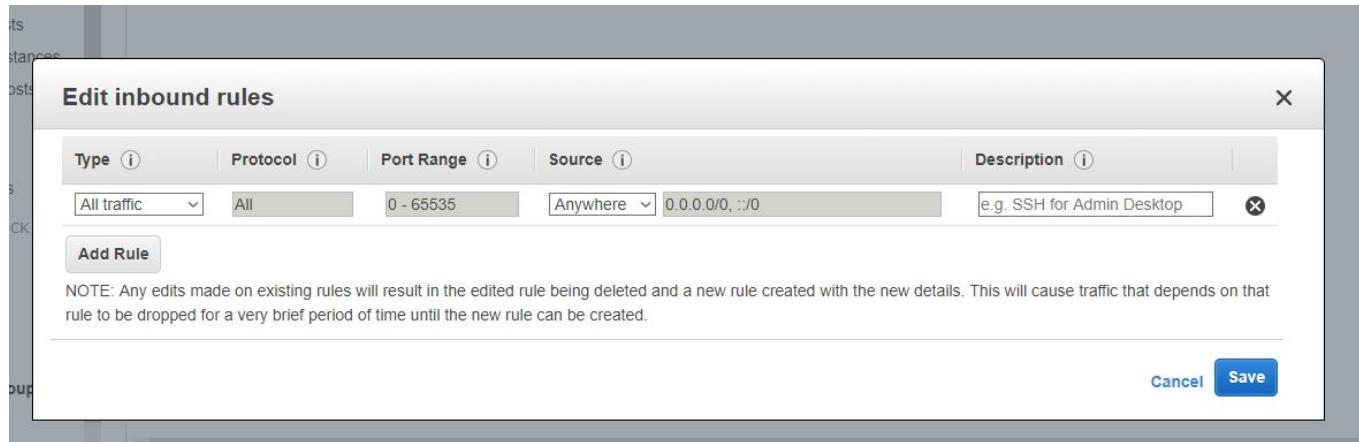
Selected the newly created security group.  
Edit Inbound rules and edit outbound rules.



## 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 Dashboard**

- Events
- Tags
- Reports
- Limits

**INSTANCES**

- Instances
- Spot Requests
- Reserved Instances

**IMAGES**

- AMIs
- Bundle Tasks

**ELASTIC BLOCK STORE**

- Volumes
- Snapshots

**NETWORK & SECURITY**

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

**LOAD BALANCING**

- Load Balancers

**AUTO SCALING**

- Launch Configurations
- Auto Scaling Groups

**Resources**

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0 Running Instances	0 Elastic IPs
0 Volumes	0 Snapshots
0 Key Pairs	0 Load Balancers
0 Placement Groups	1 Security Groups

Easily deploy and operate applications - use Chef recipes, manage SSH users, and more. [Try OpsWorks now.](#)

**Create Instance**

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

**Launch Instance**

Note: Your instances will launch in the US East (N. Virginia) region

**Service Health**

**Service Status:**

- US East (N. Virginia):  
This service is operating normally

**Availability Zone Status:**

- us-east-1a:  
Availability zone is operating normally
- us-east-1b:  
Availability zone is operating normally
- us-east-1c:  
Availability zone is operating normally
- us-east-1e:  
Availability zone is operating normally

[Service Health Dashboard](#)

**Scheduled Events**

**US East (N. Virginia):**

No events

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

ami-a13b59c2

This is created by [http://www.louisaslett.com/RStudio\\_AMI/](http://www.louisaslett.com/RStudio_AMI/)

Step 1: Choose an Amazon Machine Image (AMI) Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start      My AMIs      AWS Marketplace      Community AMIs

Search bar: rstudio

AWS Marketplace: 6 results for "rstudio" on AWS Marketplace  
Partner software pre-configured to run on AWS

AMI Name	Description	Architecture
RStudio-0.99.896_R-3.3.0_Julia-0.4.5_ubuntu-16.04-LTS-64bit - ami-3f9a4c5c	Ready to run RStudio + (experimental) Julia server for statistical computation ( <a href="http://www.louisaslett.com">www.louisaslett.com</a> ). Connect to instance public DNS in web browser (standard port 80), username rstudio and password rstudio	64-bit
OMPI_R_RStudioServer - ami-979720f4	[Copied ami-91b77587 from us-east-1] OMPI_R_RStudioServer	64-bit
<b>RStudio-1.0.153_R-3.4.1_Julia-0.6.0_ubuntu-16.04-LTS-64bit - ami-a13b59c2</b>	Ready to run RStudio + (experimental) Julia server for statistical computation ( <a href="http://www.louisaslett.com">www.louisaslett.com</a> ). Connect to instance public DNS in web browser (standard port 80), username rstudio and password rstudio	64-bit
SATT Analytics Platform - Base-a96c7627-bb8b-4cc2-89bd-a69e52b8431b-ami-65772400.2 - ami-ac091afe	An Advanced Analytics Platform, based on R Foundation. The Platform is equipped with RStudio server, R web application framework, Interactive visualization and dynamic report generation packages.	64-bit

Filter sidebar:  
Operating system: Amazon Linux, Cent OS, Debian, Fedora, Gentoo, OpenSUSE, Other Linux, Red Hat, SUSE Linux, Ubuntu, Windows  
Architecture: 32-bit, 64-bit  
Root device type: ebs

The third row, "RStudio-1.0.153\_R-3.4.1\_Julia-0.6.0\_ubuntu-16.04-LTS-64bit - ami-a13b59c2", is circled in blue.

# 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.

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

### Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types ▾    Current generation ▾    Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes

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

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

## Step 3: Configure Instance Details

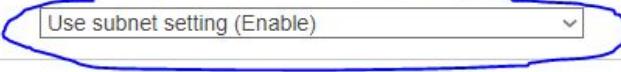
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of instance, and more.

Number of instances  Launch into Auto Scaling Group [\(i\)](#)

Purchasing option  Request Spot instances

Network [\(i\)](#)  [Create new VPC](#)

Subnet [\(i\)](#)  [Create new subnet](#)  
251 IP Addresses available

Auto-assign Public IP [\(i\)](#)  

IAM role [\(i\)](#)  [Create new IAM role](#)

Shutdown behavior [\(i\)](#)

Enable termination protection [\(i\)](#)  Protect against accidental termination

Monitoring [\(i\)](#)  Enable CloudWatch detailed monitoring  
Additional charges apply.

Tenancy [\(i\)](#)   
Additional charges will apply for dedicated tenancy.

## 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.

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    **6. Configure Security Group**    7. Review

**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	<a href="#">Copy to new</a>

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	

*haha!!!*

# EC2 Step 7: Review and Launch

## Click the blue button.

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Add Tags   6. Configure Security Group   7. Review

### 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.

**⚠ Improve your instances' security. Your security group, default, is open to the world.**

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

**AMI Details** [Edit AMI](#)

	RStudio-1.0.153_R-3.4.1_Julia-0.6.0_ubuntu-16.04-LTS-64bit - ami-a13b59c2
	Ready to run RStudio + (experimental) Julia server for statistical computation ( <a href="http://www.louisaslett.com">www.louisaslett.com</a> ). Connect to instance public DNS in web brower (standard port 80), username rstudio and password rstudio
Root Device Type:	ebs
Virtualization type:	hvm

**Instance Type** [Edit 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** [Edit security groups](#)

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

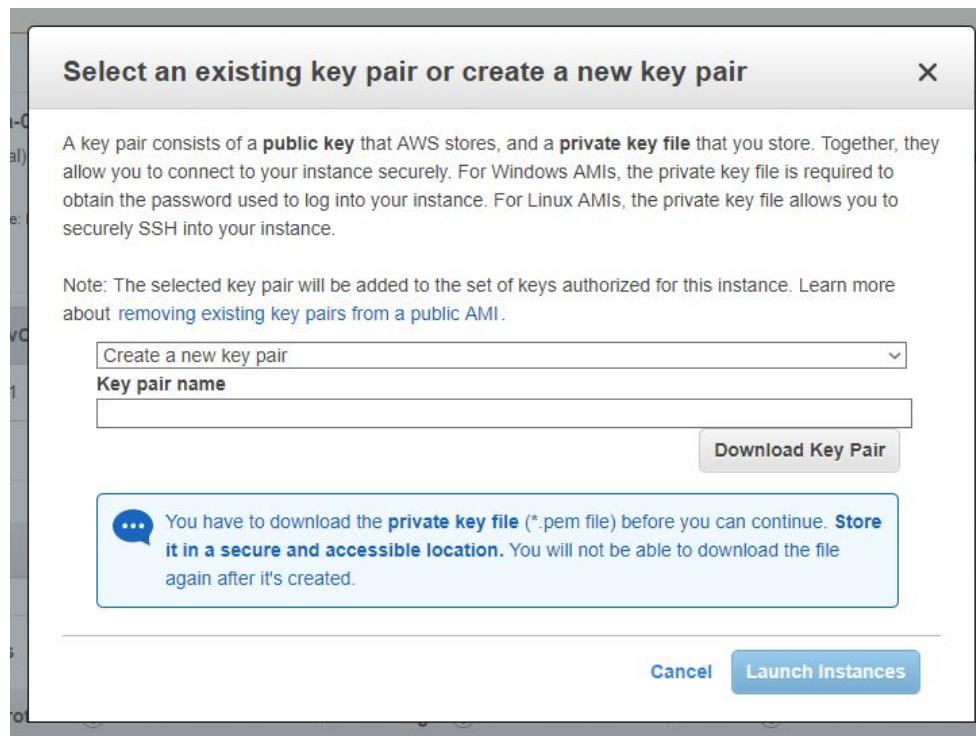
All selected security groups inbound rules

Type (i)	Protocol (i)	Port Range (i)	Source (i)	Description (i)
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	

[Cancel](#) [Previous](#) **Launch** [Go to](#)

# 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/>.

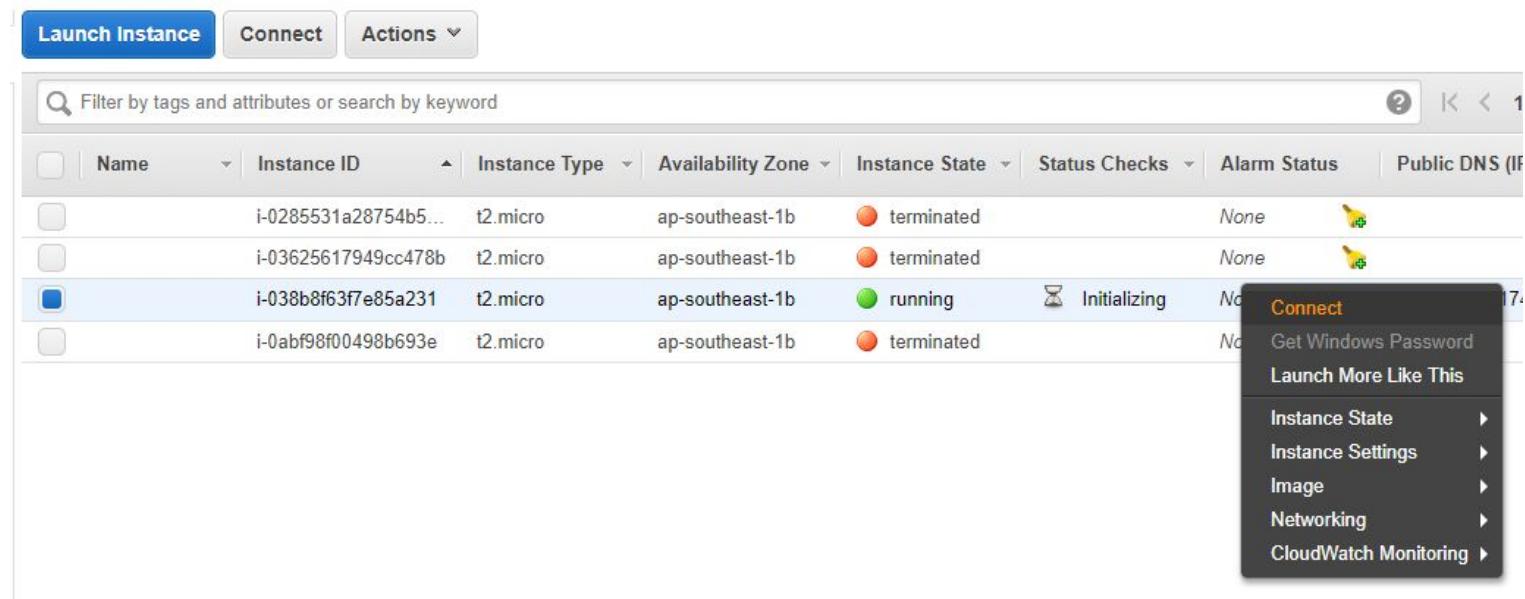
```
ssh -i 'c:\Users\yourusername\Downloads\MyKeyPair.pem' ubuntu@{IP_Address}
```

- Mac: skip download Git. Go straight

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

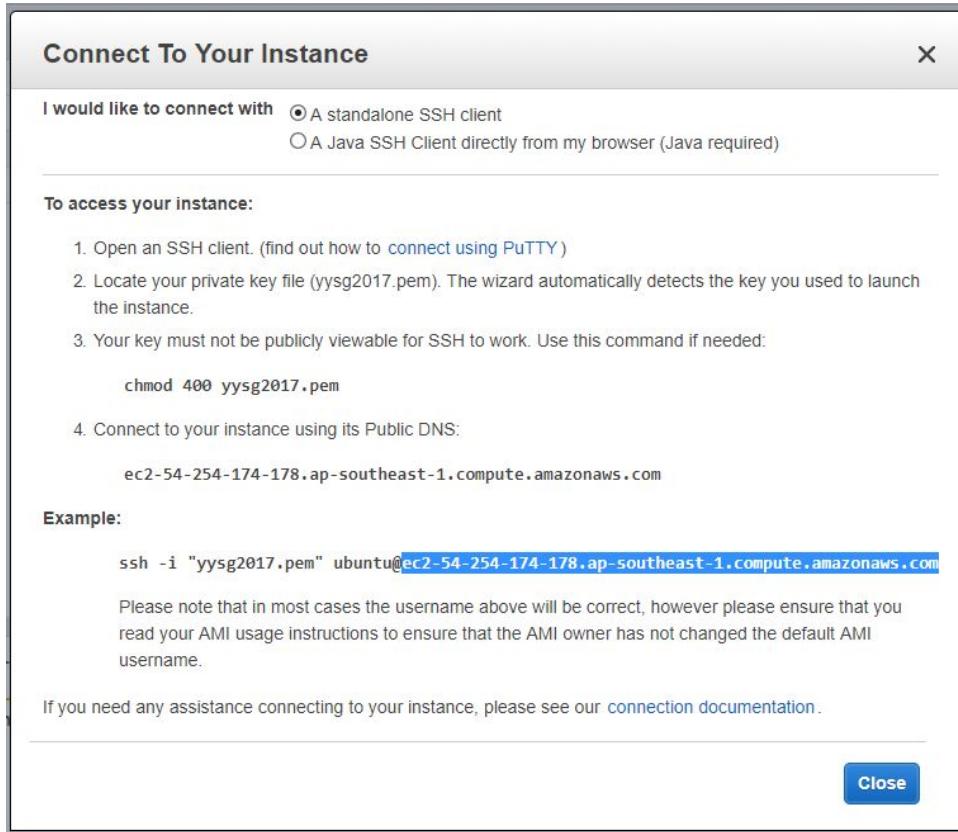
# Control the instance.

- 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/>

Initial password: rstudio/rstudio.

The screenshot shows a web browser window with the address bar containing the URL <http://ec2-54-254-174-178.ap-southeast-1.compute.amazonaws.com/auth-sign-in>. The main content is a "Sign in to RStudio" form. The form has a light gray background and a white input area. It contains fields for "Username" (containing "rstudio") and "Password" (containing a series of dots). There is also a checkbox labeled "Stay signed in" and a blue "Sign In" button at the bottom. The browser interface includes standard navigation buttons (back, forward, search) and a tab bar with the RStudio logo.

Sign in to RStudio

Username:  
rstudio

Password:  
.....

Stay signed in

Sign In

# RStudio Setup

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

The screenshot shows the RStudio interface with the following components:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Two tabs open: "app.R" and "app.R". The code in "app.R" is:

```
1 library(shiny)
2 
3 ui <- pageWithSidebar(
4   headerPanel("Interactive Histogram"),
5   sidebarPanel(
6     numericInput("n", "Generate this many points",
7                 min = 1, value = 1000),
8     selectInput("family", "From this family",
9                 choices = c("Normal",
10                  "Uniform",
11                  "Exponential"),
12                 selected = "normal"),
13     sliderInput("bins", "number of bins",
14                 min = 1, max = 100, value = 50)
15   ),
16   mainPanel(
17     plotOutput("histogram")
18   )
19 )
20 
```

- Environment:** Shows values and functions defined in the current session.
- Files:** Shows the project structure with files like .Rhistory, R, ShinyApps, and Welcome.R.
- Console:** Displays R command history and output.

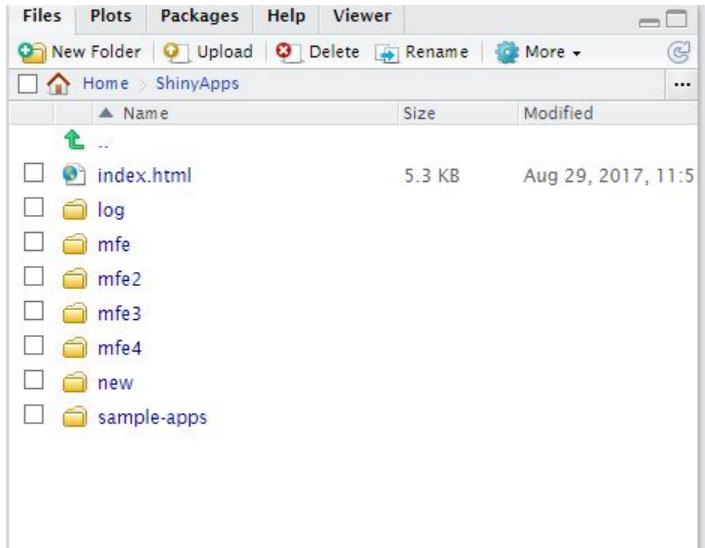
```
The downloaded source packages are in
  '/tmp/RtmpPAbzdr/downloaded_packages'
> runApp('ShinyApps/mfe2')

Listening on http://127.0.0.1:7737
> runApp('ShinyApps/mfe2')

Listening on http://127.0.0.1:7737
> git
Error: object 'git' not found
> runApp('new')

Listening on http://127.0.0.1:7737
> |
```

# Shiny Server



/home/**rstudio**/ShinyApps/**new**/

/ShinyApps/**new**/

[http://ec2-54-254-174-178.ap-southeast-1.compute.amazonaws.com/shiny/\*\*rstudio\*\*/\*\*new\*\*/](http://ec2-54-254-174-178.ap-southeast-1.compute.amazonaws.com/shiny/rstudio/new/)

# 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.

# 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.

# 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 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 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: 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.

# Markdown: Inline formatting

## Strikethrough

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

## Superscripts and subscripts

H<sub>2</sub>O is a liquid.  $2^{10}$  is 1024. H<sub>2</sub>O 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, 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.

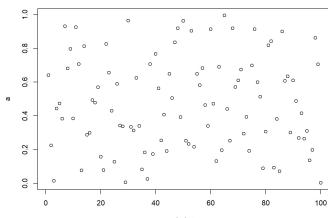
# Chunk options

- `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

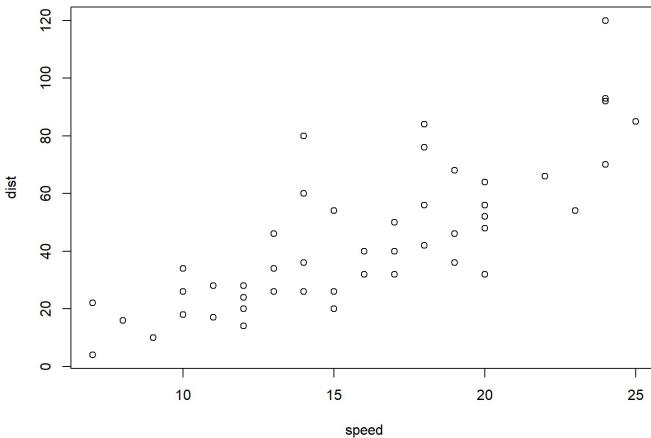
```
```{r kable}
knitr::kable(
  mtcars[1:5, ],
  caption = "A knitr kable."
)
...``
```

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(
  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

- 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))))
    )
  )
)
```

```
        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

```
---
```

```
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!*

[Runchee Technology](#)[Product](#)[About Us](#)[Contact Us ▾](#)

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