COM5940: NEW MEDIA BUSINESS MODEL & INNOVATION
DATA PREPARATION AND EXPLORATORY ANALYSIS

CUHK BUSINESS SCHOOL
Center for Entrepreneurship

Bernard Suen
Center for Entrepreneurship
Chinese University of Hong Kong

Today's Agenda

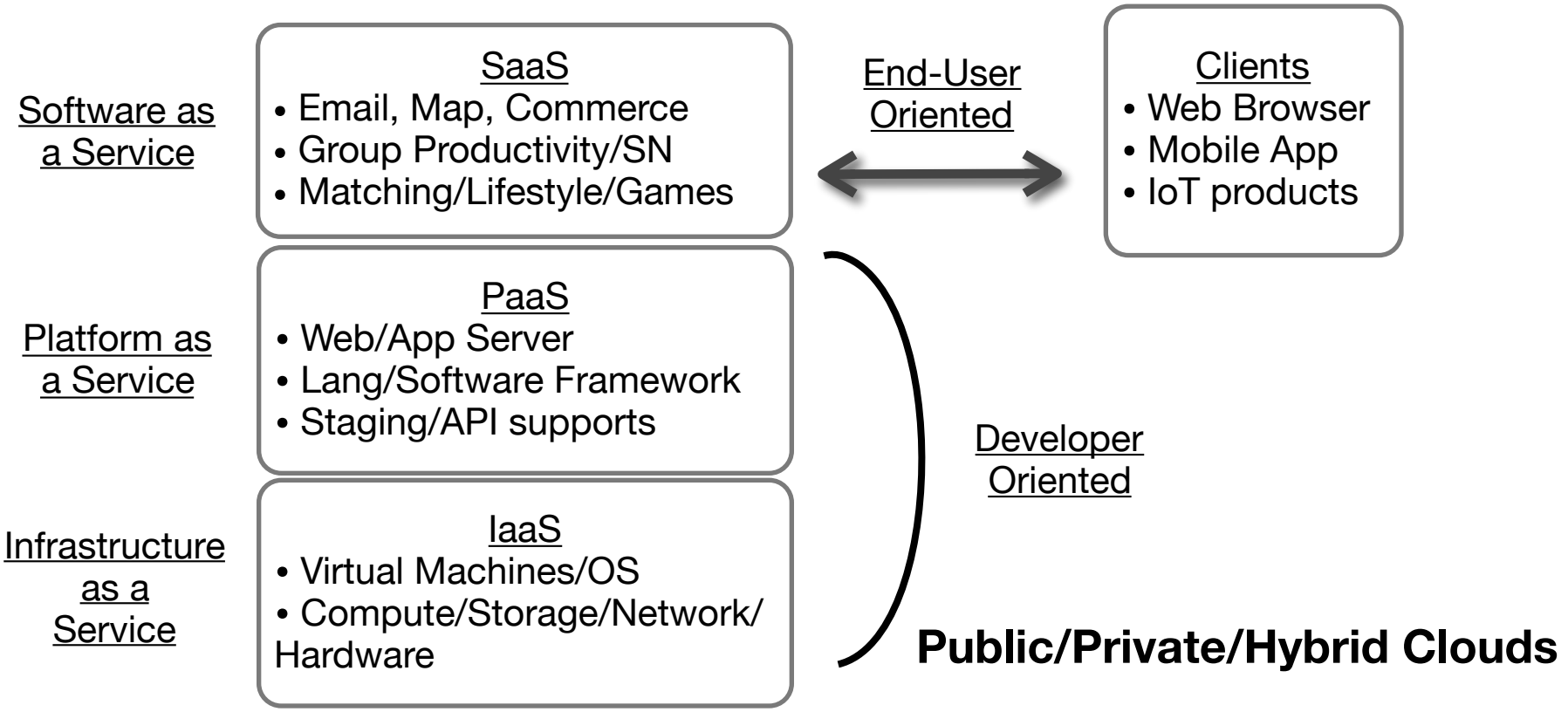
1. Latest serverless development in cloud computing.
2. Import, export, transform, and clean data in Pandas.
3. Use of Pandas for filtering data
4. Use of Descriptive Statistics in Exploratory Data Analysis.
5. Use of Matplotlib to visualize dataset.

Latest Serverless Development in Cloud Computing

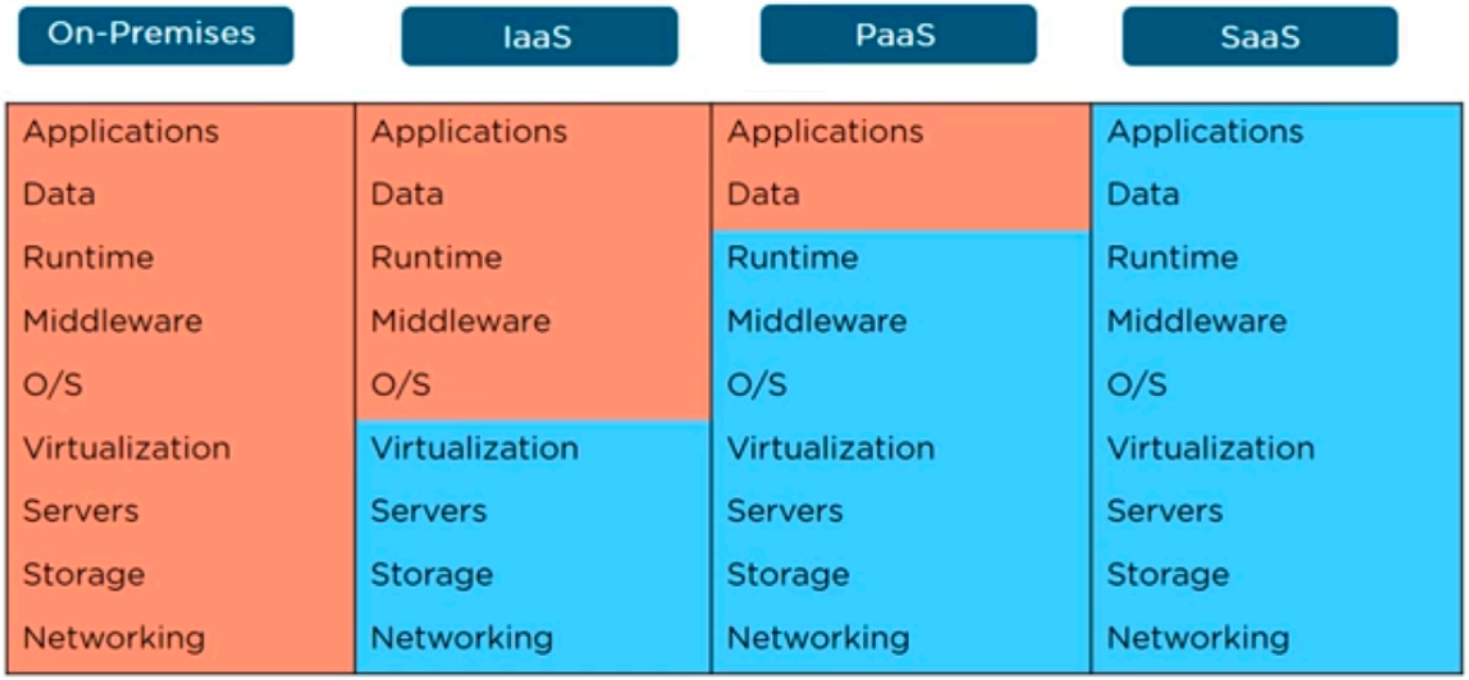
Data Wrangling and Cleaning with Pandas

Descriptive Statistics & Data Visualization

云计算的三层架构


Software as a Service
SaaS
• Email, Map, Commerce
• Group Productivity/SN
• Matching/Lifestyle/Games
End-User Oriented
Clients
• Web Browser
• Mobile App
• IoT products
Platform as a Service
PaaS
• Web/App Server
• Lang/Software Framework
• Staging/API supports
Developer Oriented
Infrastructure as a Service
IaaS
• Virtual Machines/OS
• Compute/Storage/Network/ Hardware
Public/Private/Hybrid Clouds

Difference between IaaS, PaaS and SaaS



Source: SimplyLearn

Managed by you

Managed by Vendor

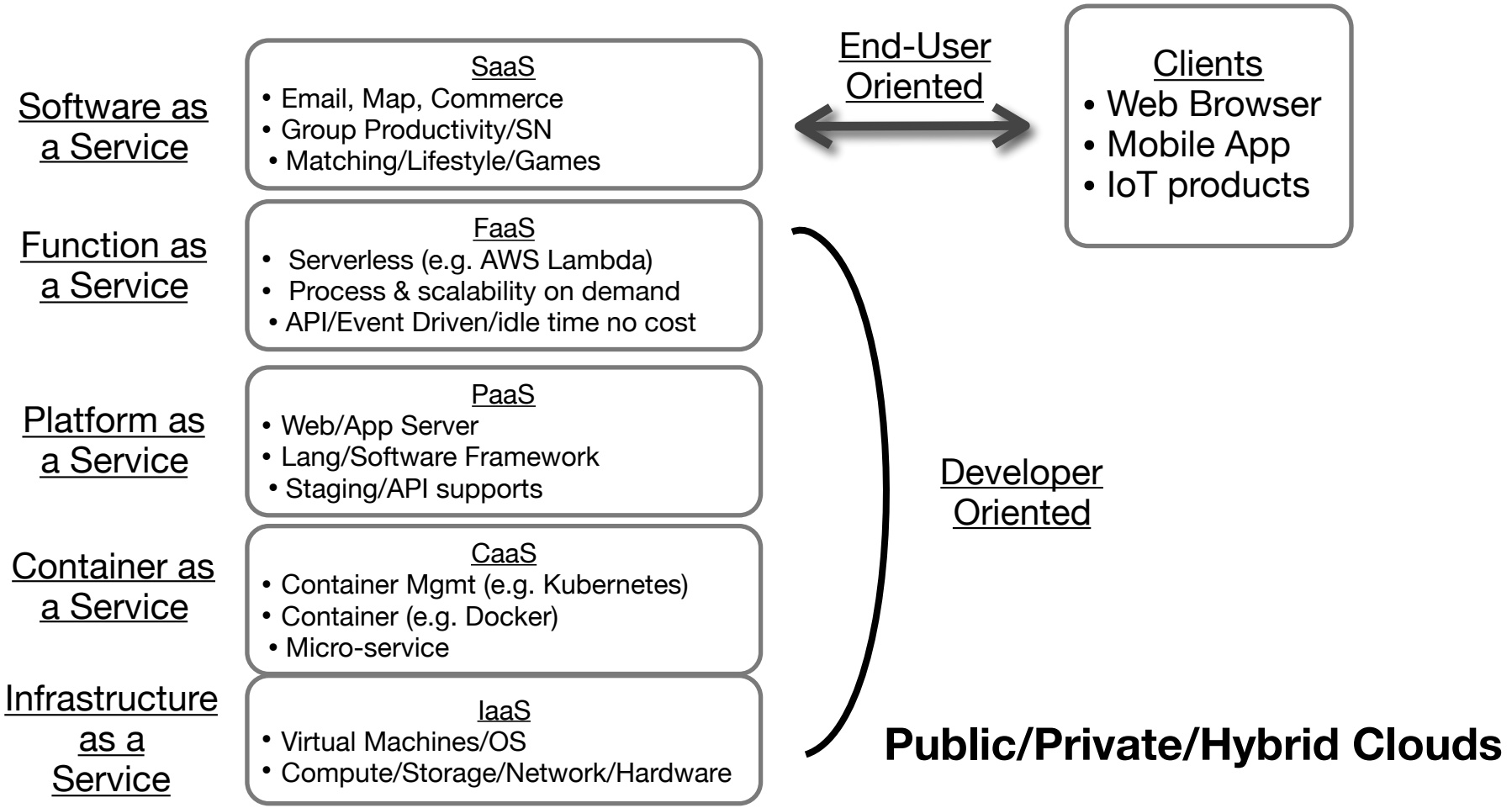
現今在云计算领域内的三大服务

VIRTUAL MACHINE

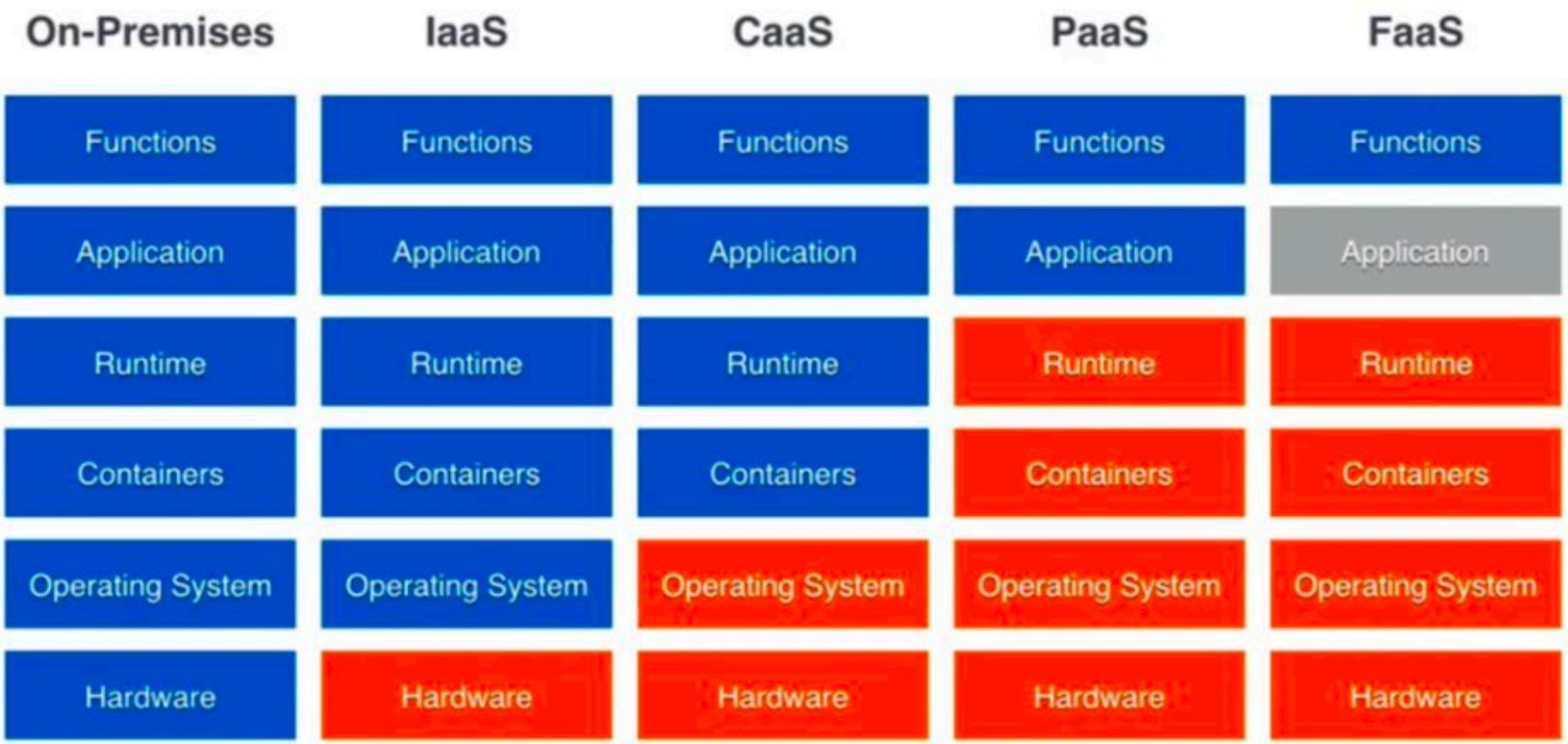
CONTAINER

SERVERLESS

云计算服务架构重组


Software as a Service
SaaS
• Email, Map, Commerce
• Group Productivity/SN
• Matching/Lifestyle/Games
End-User Oriented
Clients
• Web Browser
• Mobile App
• IoT products
Function as a Service
FaaS
• Serverless (e.g. AWS Lambda)
• Process & scalability on demand
• API/Event Driven/idle time no cost
Platform as a Service
PaaS
• Web/App Server
• Lang/Software Framework
• Staging/API supports
Developer Oriented
Container as a Service
CaaS
• Container Mgmt (e.g. Kubernetes)
• Container (e.g. Docker)
• Micro-service
Infrastructure as a Service
IaaS
• Virtual Machines/OS
• Compute/Storage/Network/Hardware
Public/Private/Hybrid Clouds

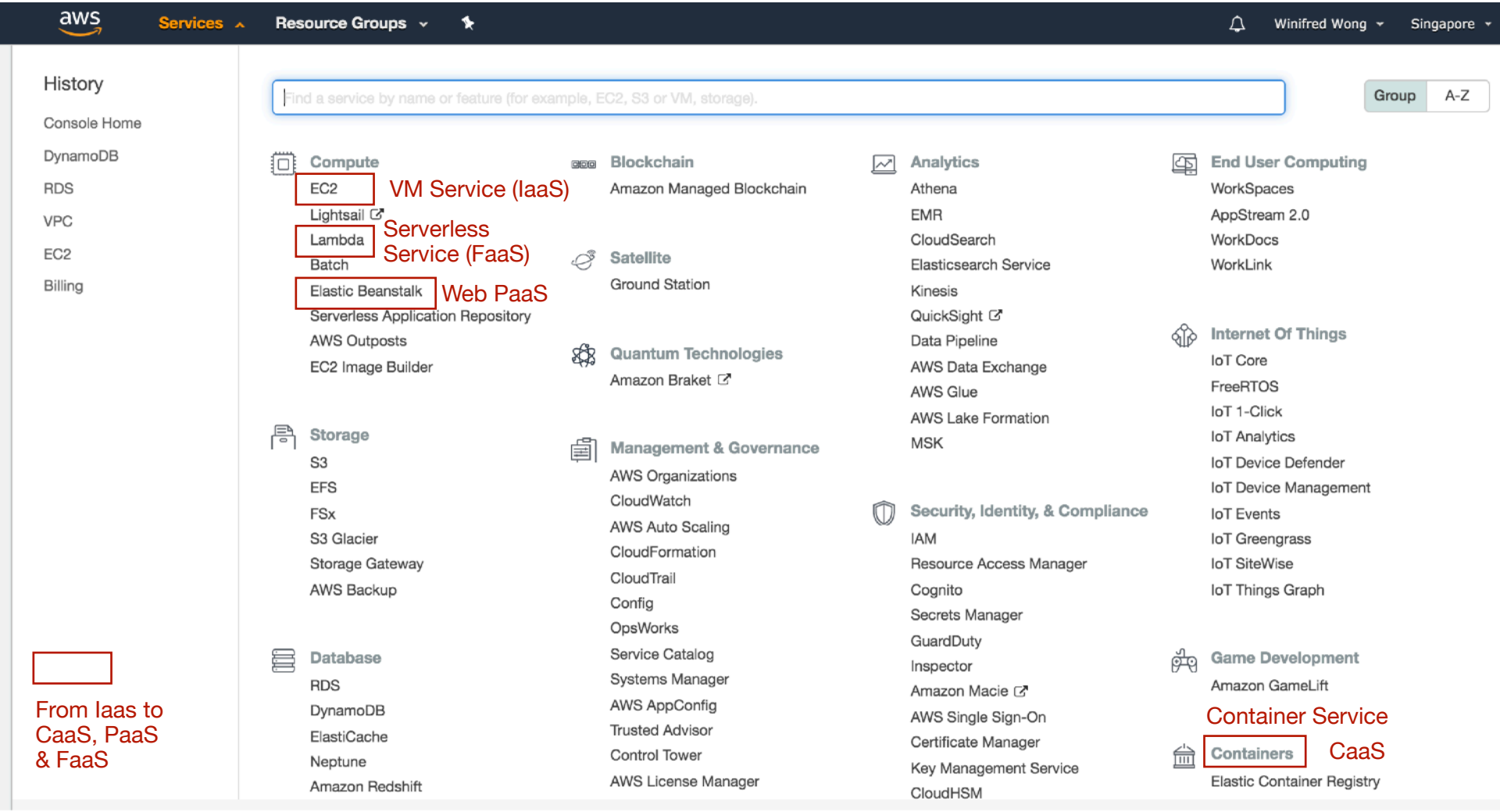
云计算的发展



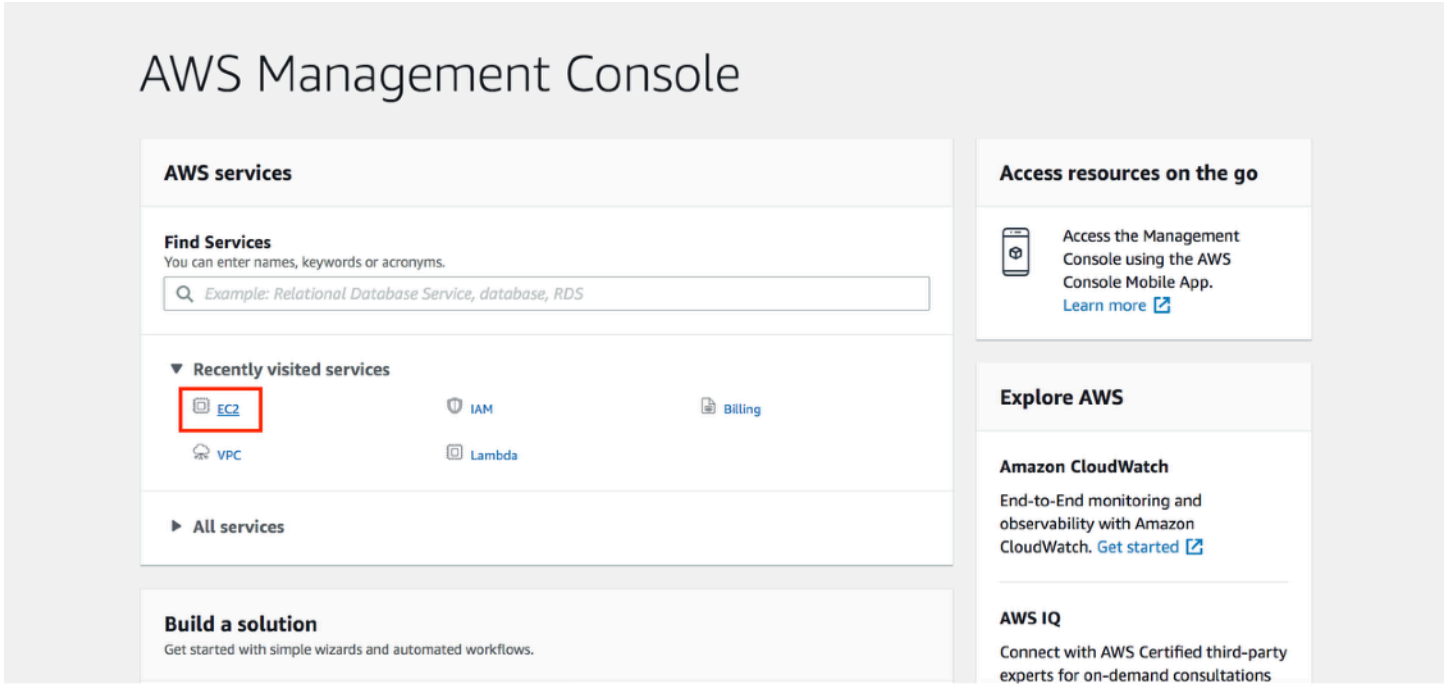
Source: Tencent Cloud

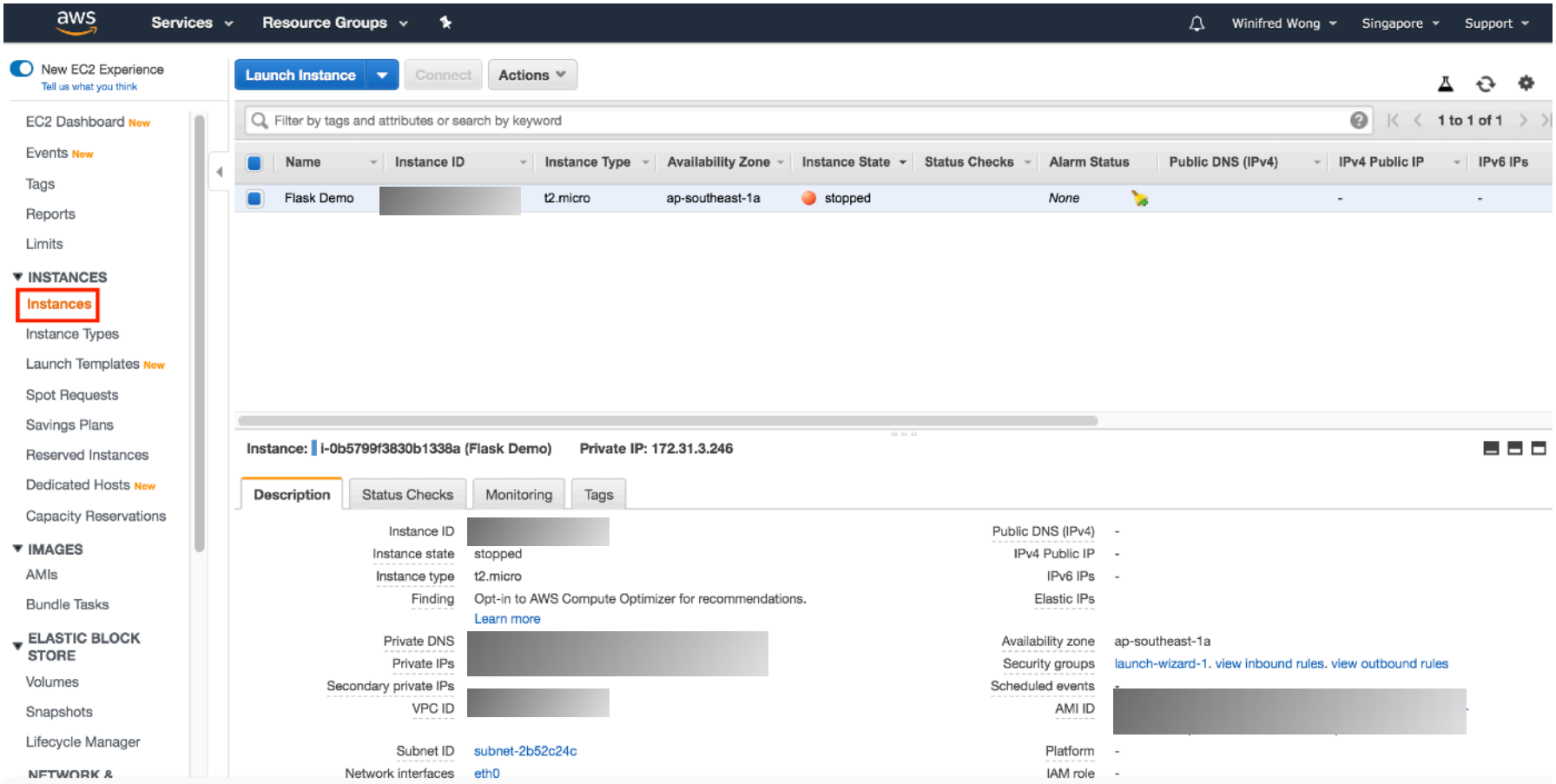
Why the new trends in containerisation and serverless (i.e. FaaS) services?

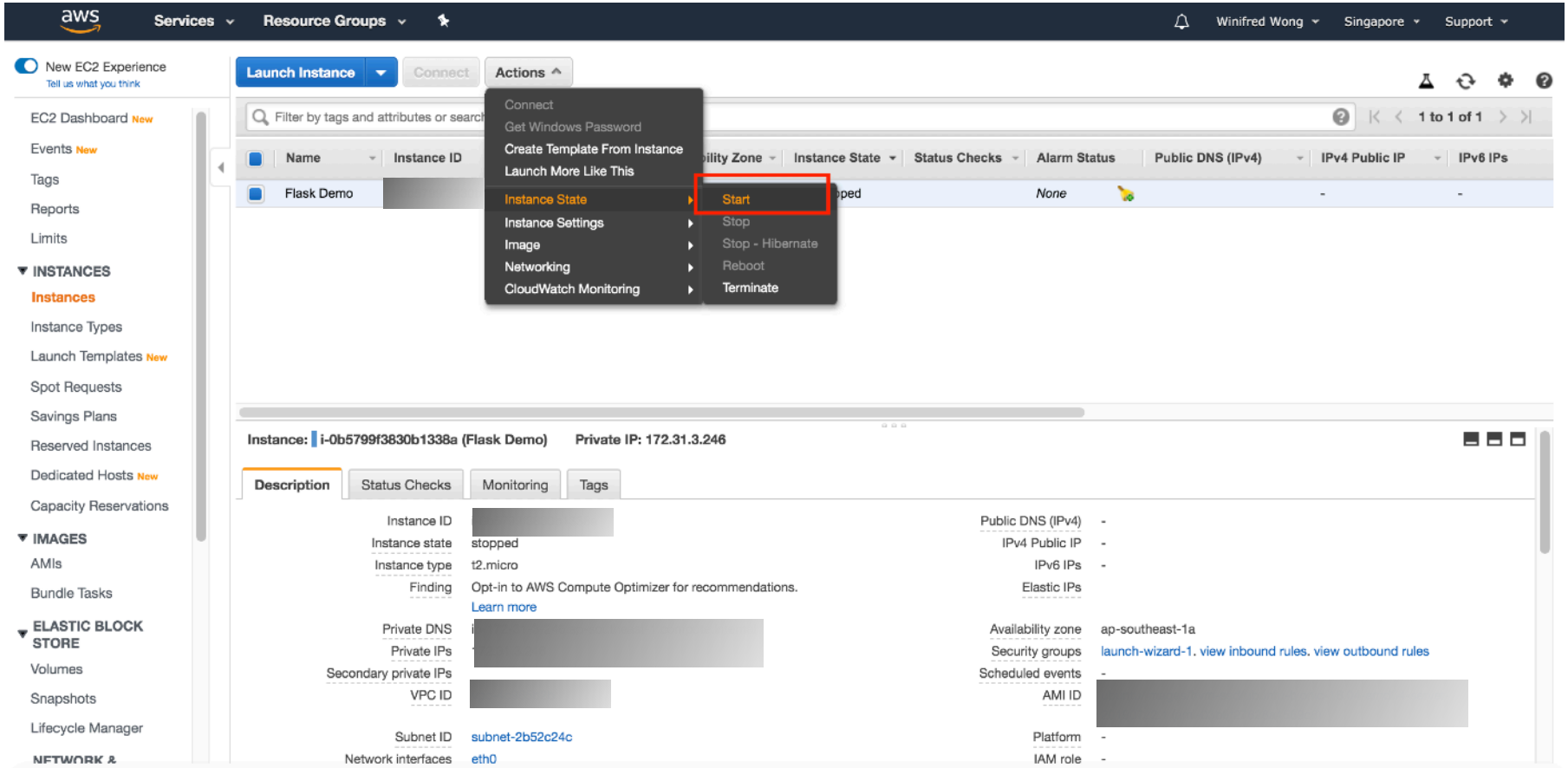
Major Services from AWS

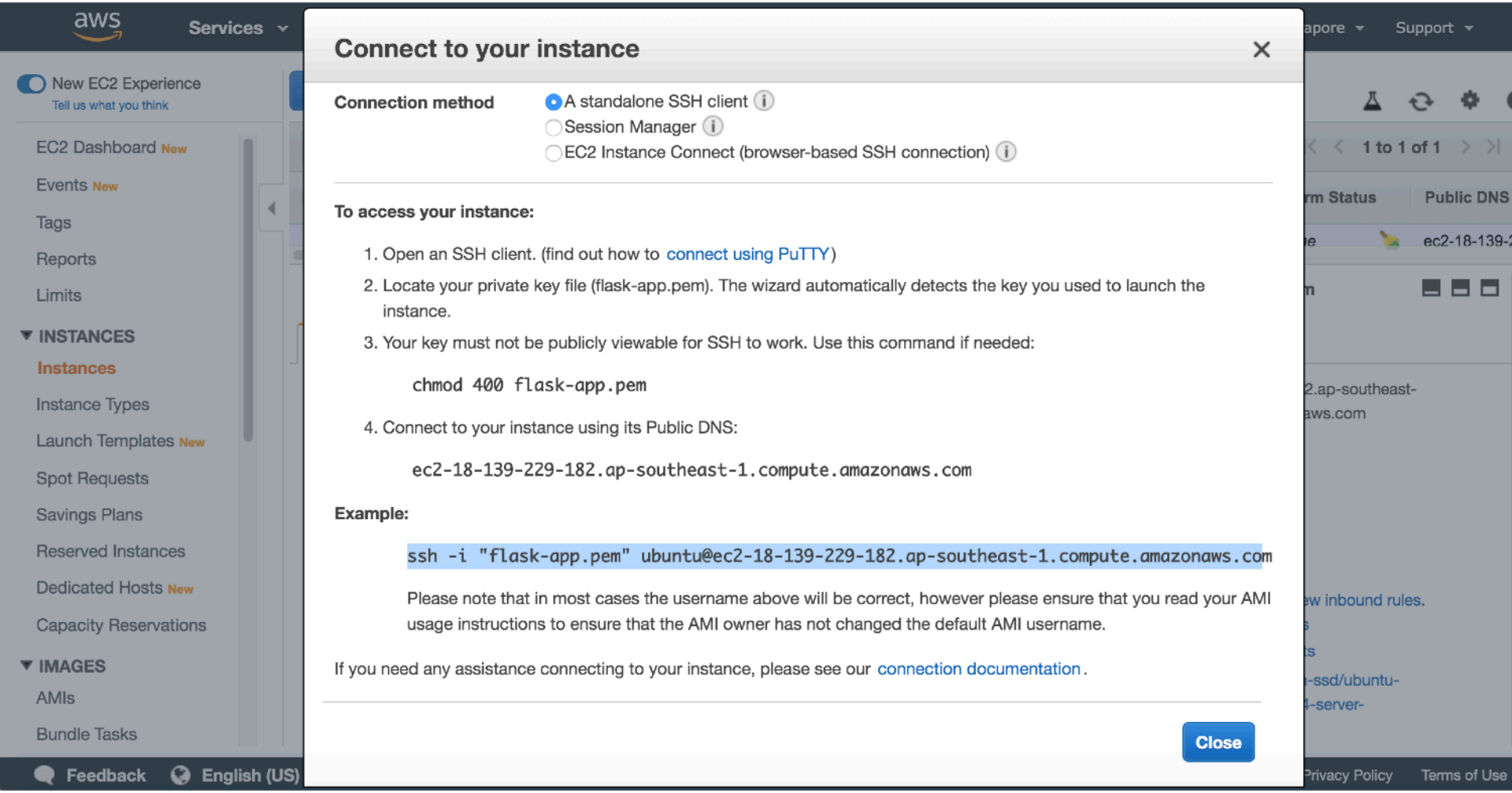

aws
Services
Resource Groups
Winifred Wong
Singapore
History
Console Home
DynamoDB
RDS
VPC
EC2
Billing
Find a service by name or feature (for example, EC2, S3 or VM, storage).
Group
A-Z
Compute
EC2
VM Service (IaaS)
Lightsail
Lambda
Serverless Service (FaaS)
Batch
Elastic Beanstalk
Web PaaS
Serverless Application Repository
AWS Outposts
EC2 Image Builder
Storage
S3
EFS
FSx
S3 Glacier
Storage Gateway
AWS Backup
Database
RDS
DynamoDB
ElastiCache
Neptune
Amazon Redshift
Blockchain
Amazon Managed Blockchain
Satellite
Ground Station
Quantum Technologies
Amazon Braket
Management & Governance
AWS Organizations
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
AWS AppConfig
Trusted Advisor
Control Tower
AWS License Manager
Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Data Exchange
AWS Glue
AWS Lake Formation
MSK
Security, Identity, & Compliance
IAM
Resource Access Manager
Cognito
Secrets Manager
GuardDuty
Inspector
Amazon Macie
AWS Single Sign-On
Certificate Manager
Key Management Service
CloudHSM
End User Computing
WorkSpaces
AppStream 2.0
WorkDocs
WorkLink
Internet Of Things
IoT Core
FreeRTOS
IoT 1-Click
IoT Analytics
IoT Device Defender
IoT Device Management
IoT Events
IoT Greengrass
IoT SiteWise
IoT Things Graph
Game Development
Amazon GameLift
Container Service
Containers
CaaS
Elastic Container Registry
From Iaas to CaaS, PaaS & FaaS

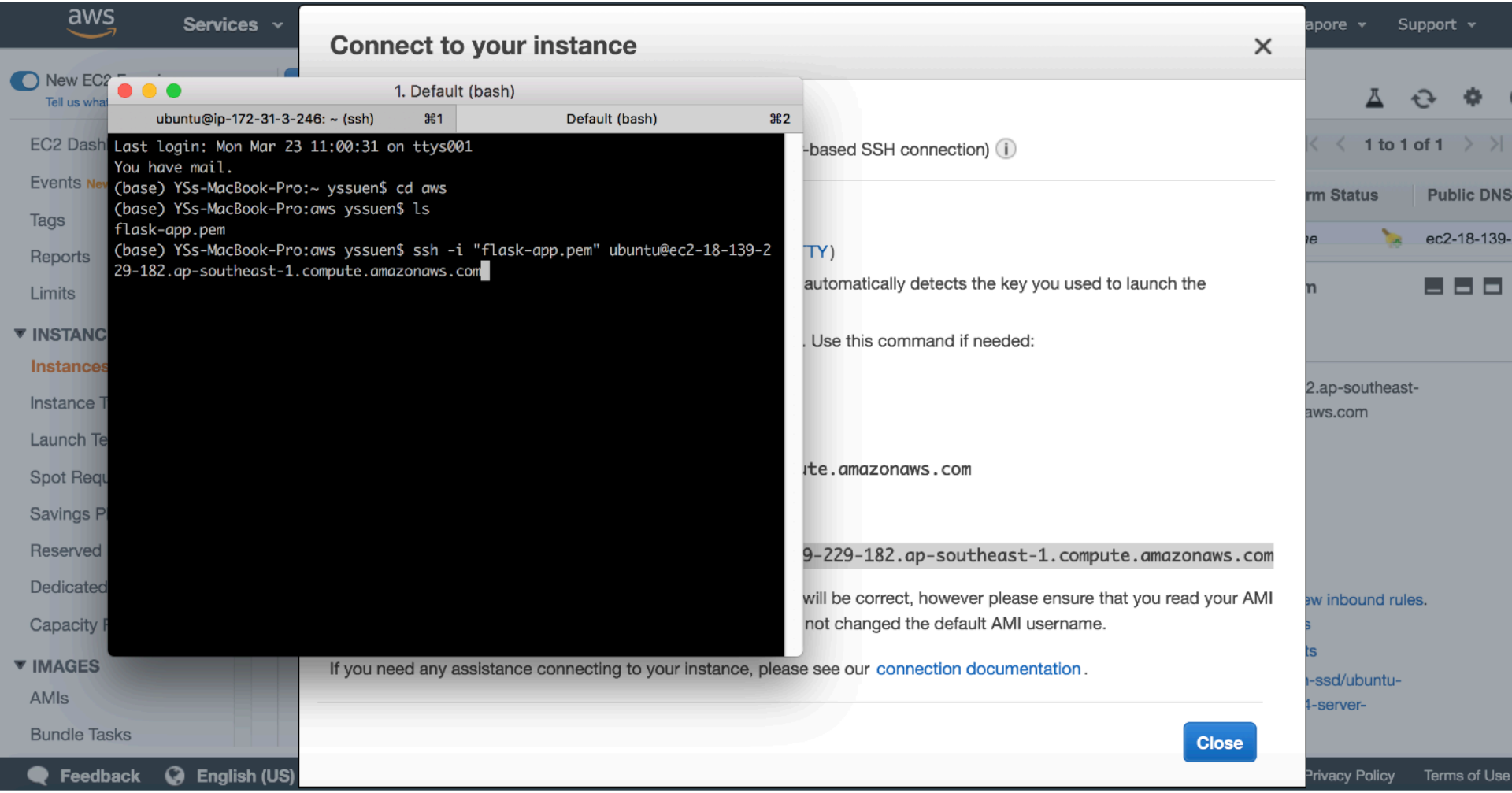
Demonstration of Virtual Machine (Local VirtualBox vs. AWS Remote EC2)


AWS Management Console
AWS services
Find Services
You can enter names, keywords or acronyms.
Example: Relational Database Service, database, RDS
Recently visited services
EC2
IAM
Billing
VPC
Lambda
All services
Build a solution
Get started with simple wizards and automated workflows.
Access resources on the go
Access the Management Console using the AWS Console Mobile App.
Learn more
Explore AWS
Amazon CloudWatch
End-to-End monitoring and observability with Amazon CloudWatch. Get started
AWS IQ
Connect with AWS Certified third-party experts for on-demand consultations


aws
Services
Resource Groups
Winifred Wong
Singapore
Support
New EC2 Experience
Tell us what you think
Launch Instance
Connect
Actions
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates New
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
Lifecycle Manager
Filter by tags and attributes or search by keyword
1 to 1 of 1
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
IPv4 Public IP
IPv6 IPs
Flask Demo
t2.micro
ap-southeast-1a
stopped
None
Instance: i-0b5799f3830b1338a (Flask Demo)
Private IP: 172.31.3.246
Description
Status Checks
Monitoring
Tags
Instance ID
Instance state
stopped
Instance type
t2.micro
Finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Private DNS
Private IPs
Secondary private IPs
VPC ID
Subnet ID
subnet-2b52c24c
Network interfaces
eth0
Public DNS (IPv4)
IPv4 Public IP
IPv6 IPs
Elastic IPs
Availability zone
ap-southeast-1a
Security groups
launch-wizard-1. view inbound rules. view outbound rules
Scheduled events
AMI ID
Platform
IAM role

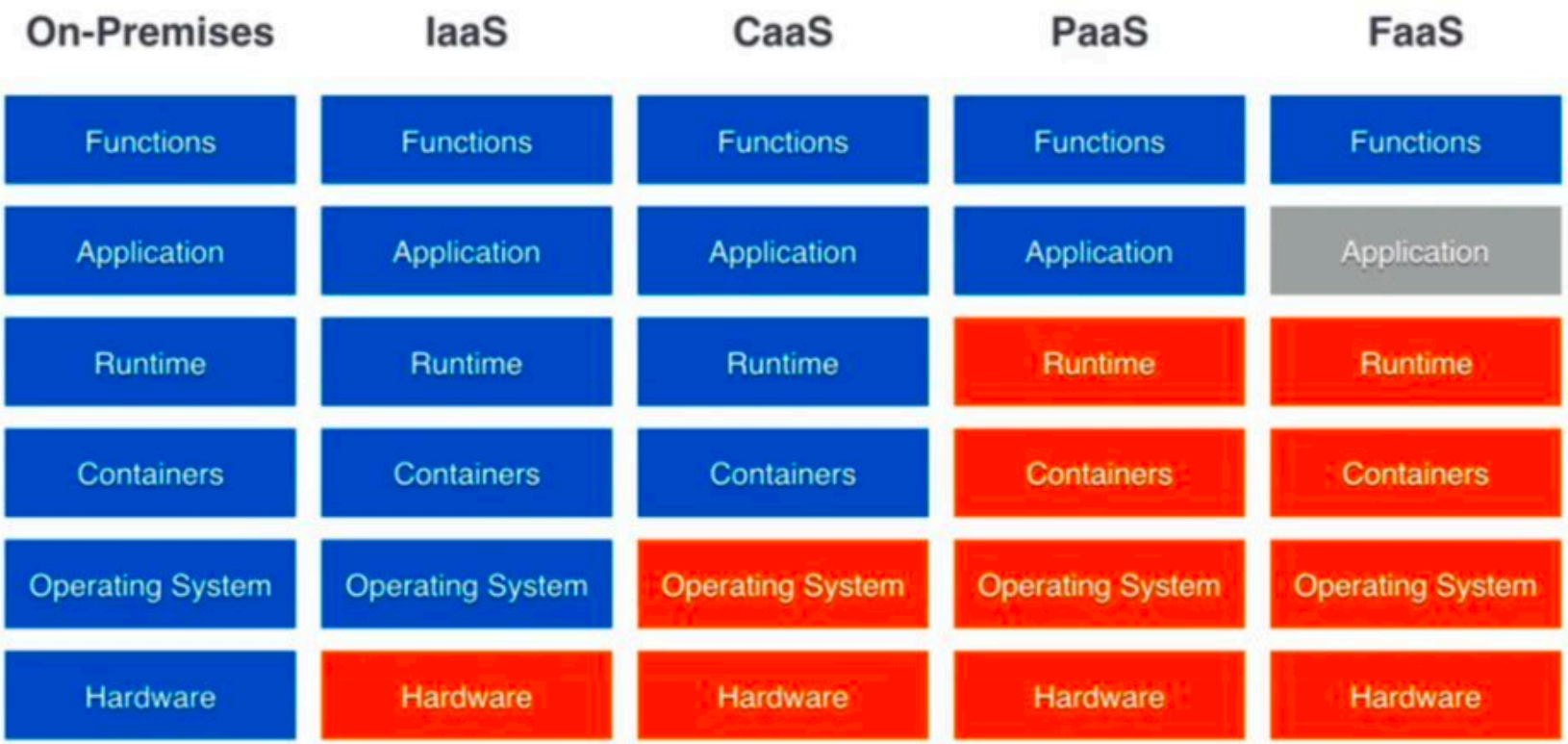

aws
Services
Resource Groups
Winifred Wong
Singapore
Support
New EC2 Experience
Tell us what you think
Launch Instance
Connect
Actions
Connect
Get Windows Password
Create Template From Instance
Launch More Like This
Instance State
Instance Settings
Image
Networking
CloudWatch Monitoring
Start
Stop
Stop - Hibernate
Reboot
Terminate
Filter by tags and attributes or search
1 to 1 of 1
Name
Instance ID
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
IPv4 Public IP
IPv6 IPs
Flask Demo
None
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates New
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
Lifecycle Manager
Instance: i-0b5799f3830b1338a (Flask Demo)
Private IP: 172.31.3.246
Description
Status Checks
Monitoring
Tags
Instance ID
Instance state
stopped
Instance type
t2.micro
Finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Private DNS
Private IPs
Secondary private IPs
VPC ID
Subnet ID
subnet-2b52c24c
Network interfaces
eth0
Public DNS (IPv4)
IPv4 Public IP
IPv6 IPs
Elastic IPs
Availability zone
ap-southeast-1a
Security groups
launch-wizard-1. view inbound rules. view outbound rules
Scheduled events
AMI ID
Platform
IAM role


aws
Services
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates New
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
Feedback
English (US)
1 to 1 of 1
Public DNS
Connect to your instance
Connection method
A standalone SSH client
Session Manager
EC2 Instance Connect (browser-based SSH connection)
To access your instance:
1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (flask-app.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:
chmod 400 flask-app.pem
4. Connect to your instance using its Public DNS:
ec2-18-139-229-182.ap-southeast-1.compute.amazonaws.com
Example:
ssh -i "flask-app.pem" ubuntu@ec2-18-139-229-182.ap-southeast-1.compute.amazonaws.com
Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.
If you need any assistance connecting to your instance, please see our connection documentation.
Close
Privacy Policy
Terms of Use


Services
Support
Connect to your instance
1. Default (bash)
ubuntu@ip-172-31-3-246: ~ (ssh) ⌘1
Default (bash) ⌘2
Last login: Mon Mar 23 11:00:31 on ttys001
You have mail.
(base) YSs-MacBook-Pro:~ yssuen$ cd aws
(base) YSs-MacBook-Pro:aws yssuen$ ls
flask-app.pem
(base) YSs-MacBook-Pro:aws yssuen$ ssh -i "flask-app.pem" ubuntu@ec2-18-139-229-182.ap-southeast-1.compute.amazonaws.com
-based SSH connection)
automatically detects the key you used to launch the
. Use this command if needed:
ute.amazonaws.com
9-229-182.ap-southeast-1.compute.amazonaws.com
will be correct, however please ensure that you read your AMI
not changed the default AMI username.
If you need any assistance connecting to your instance, please see our connection documentation.
Close
EC2 Dash
Events
Tags
Reports
Limits
INSTANCES
Instances
IMAGES
AMIs
Bundle Tasks
Feedback
English (US)
Privacy Policy
Terms of Use

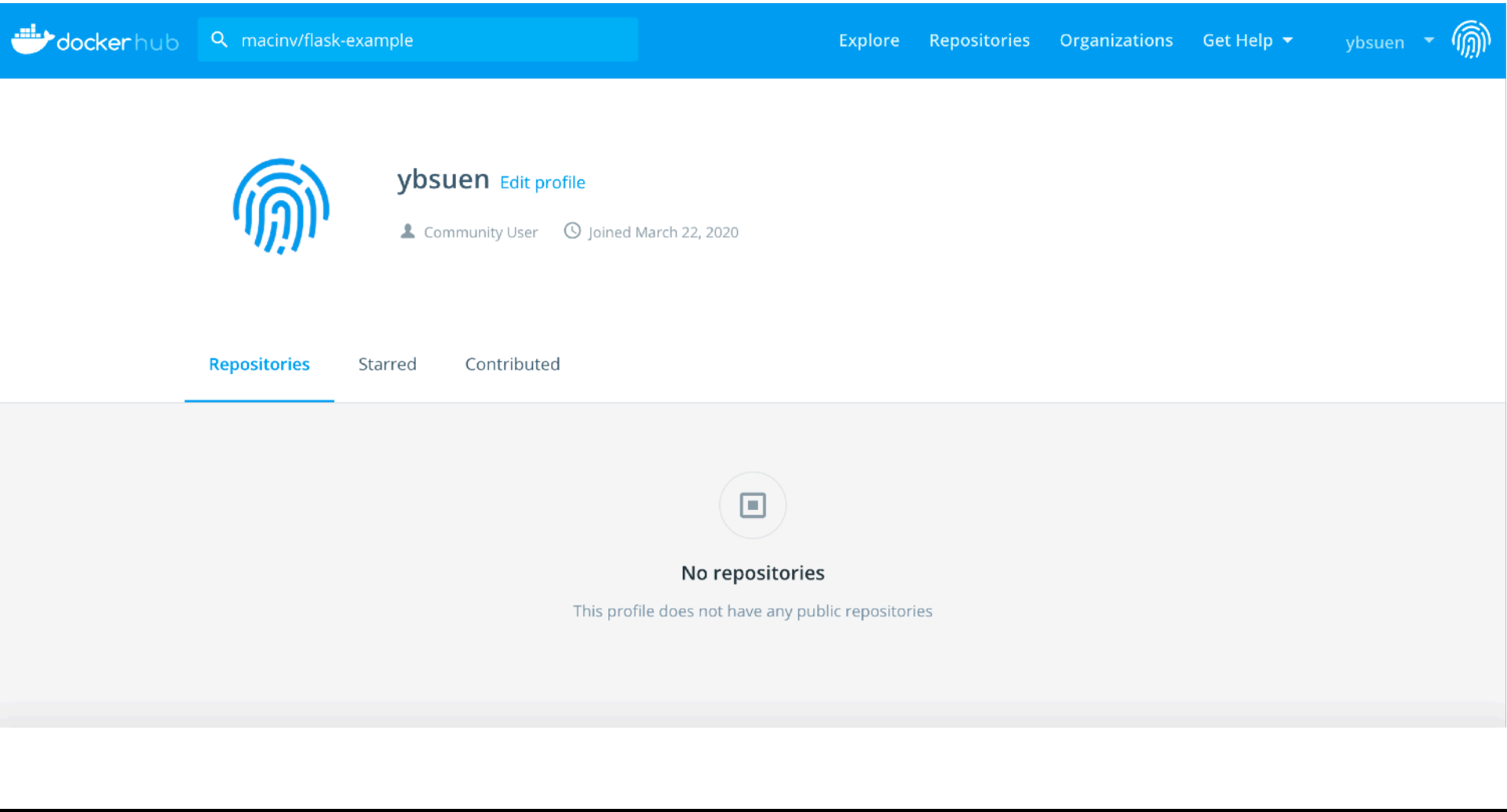
The Advantages of CaaS over IaaS

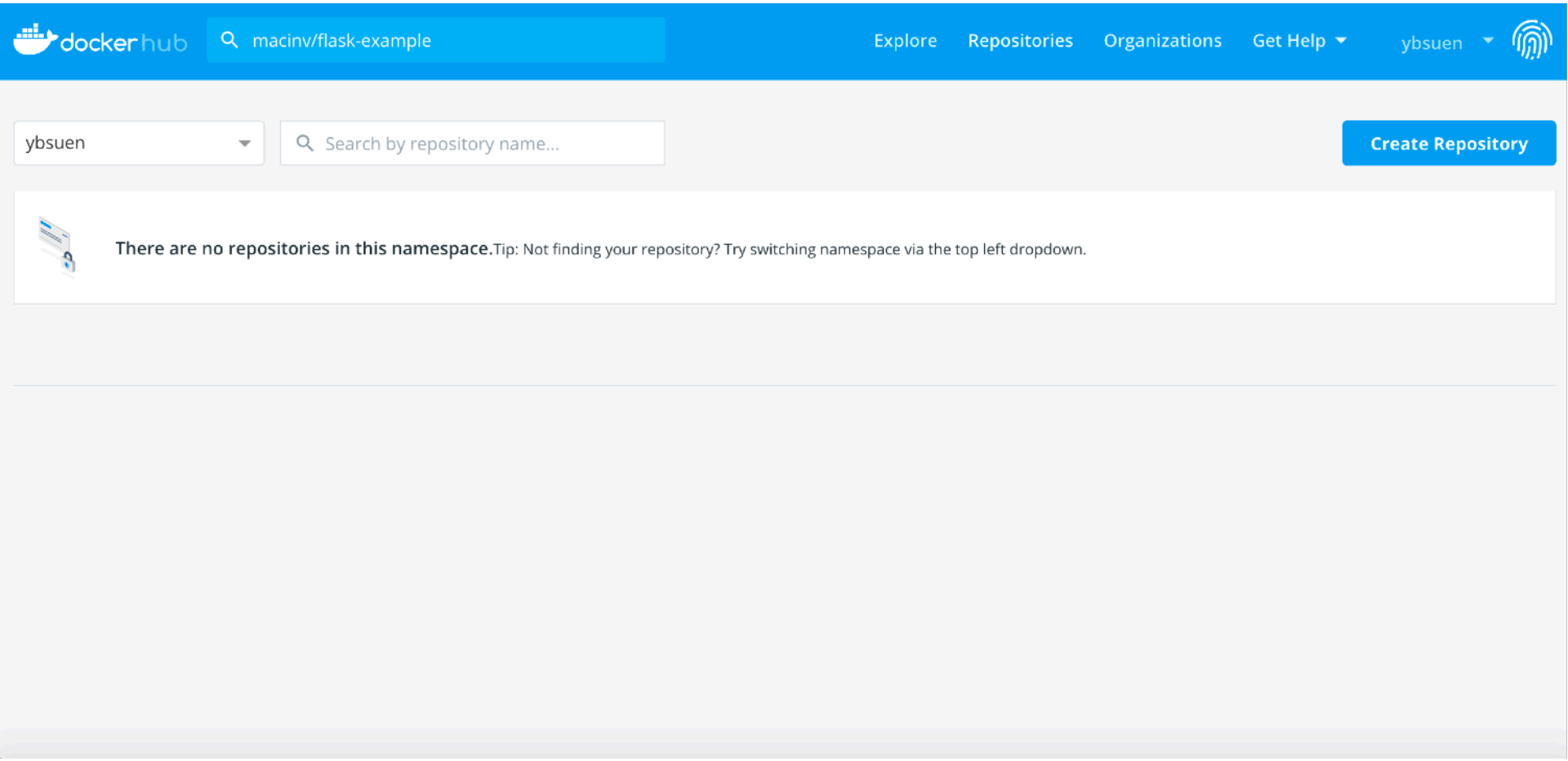
云计算的发展

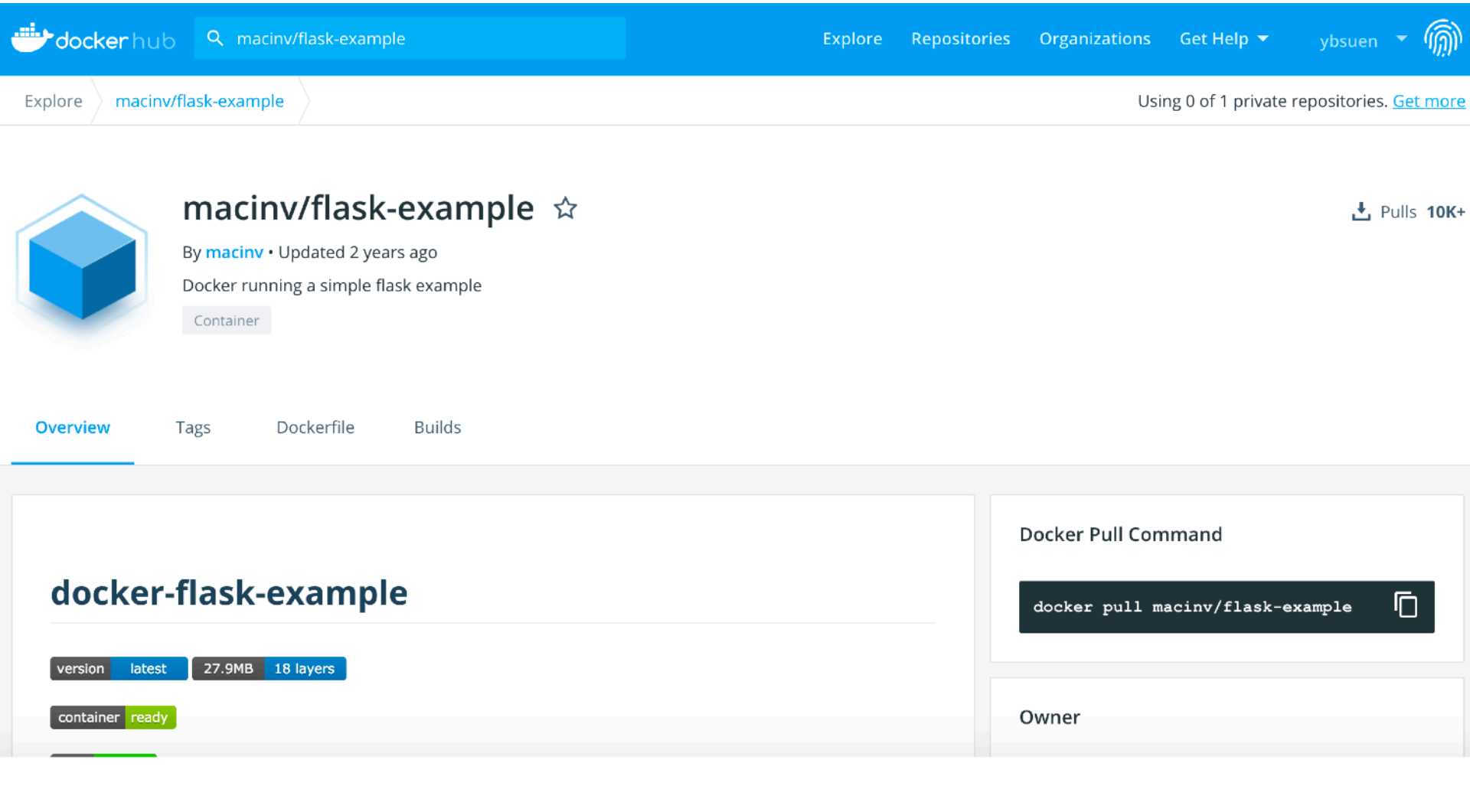


Source: Tencent Cloud

Introducing DockerHub


dockerhub
macinv/flask-example
Explore
Repositories
Organizations
Get Help
ybsuen
ybsuen
Edit profile
Community User
Joined March 22, 2020
Repositories
Starred
Contributed
No repositories
This profile does not have any public repositories


docker hub
macinv/flask-example
Explore
Repositories
Organizations
Get Help
ybsuen
ybsuen
Search by repository name...
Create Repository
There are no repositories in this namespace.Tip: Not finding your repository? Try switching namespace via the top left dropdown.


docker hub
macinv/flask-example
Explore
Repositories
Organizations
Get Help
ybsuen
Explore
macinv/flask-example
Using 0 of 1 private repositories. Get more
macinv/flask-example
By macinv • Updated 2 years ago
Docker running a simple flask example
Container
Pulls 10K+
Overview
Tags
Dockerfile
Builds
docker-flask-example
version latest
27.9MB 18 layers
container ready
Docker Pull Command
docker pull macinv/flask-example
Owner

docker-flask-example

version latest | 27.9MB 18 layers

container ready

build passing

Docker running a simple flask example

To run (at port `8080`):

```
docker run -p 8080:8080 macinv/flask-example
```

Docker Pull Command

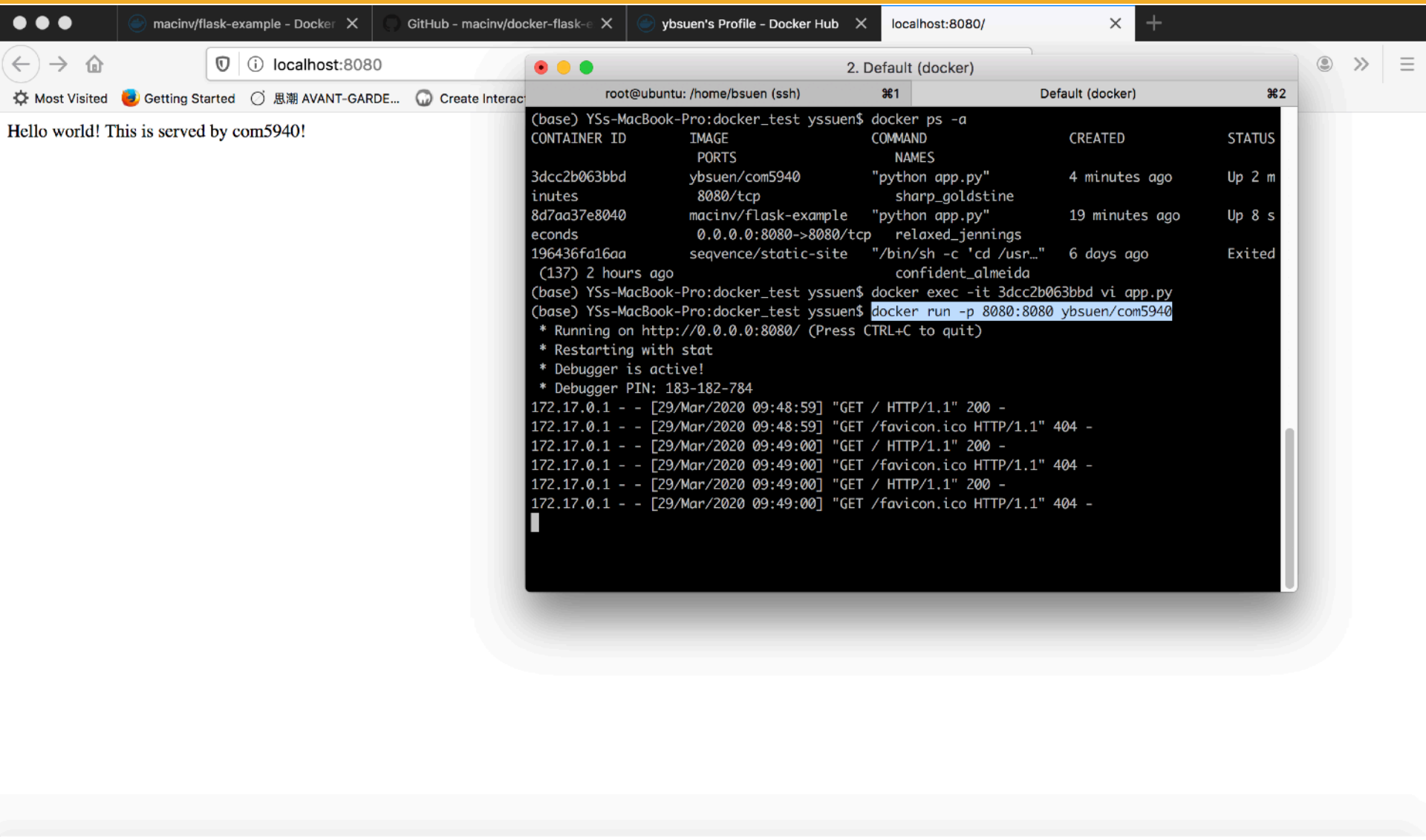
```
docker pull macinv/flask-example
```

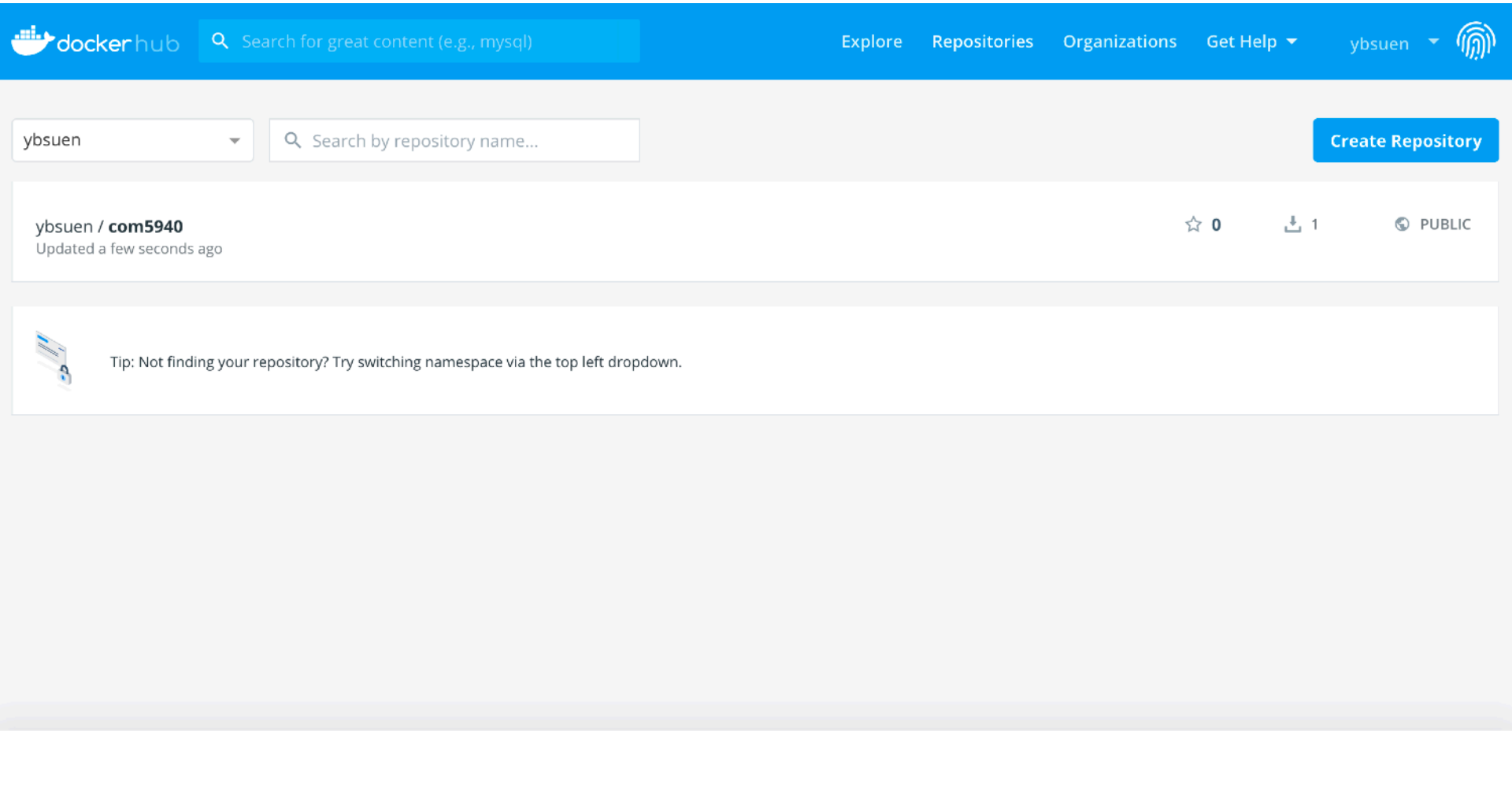
Owner

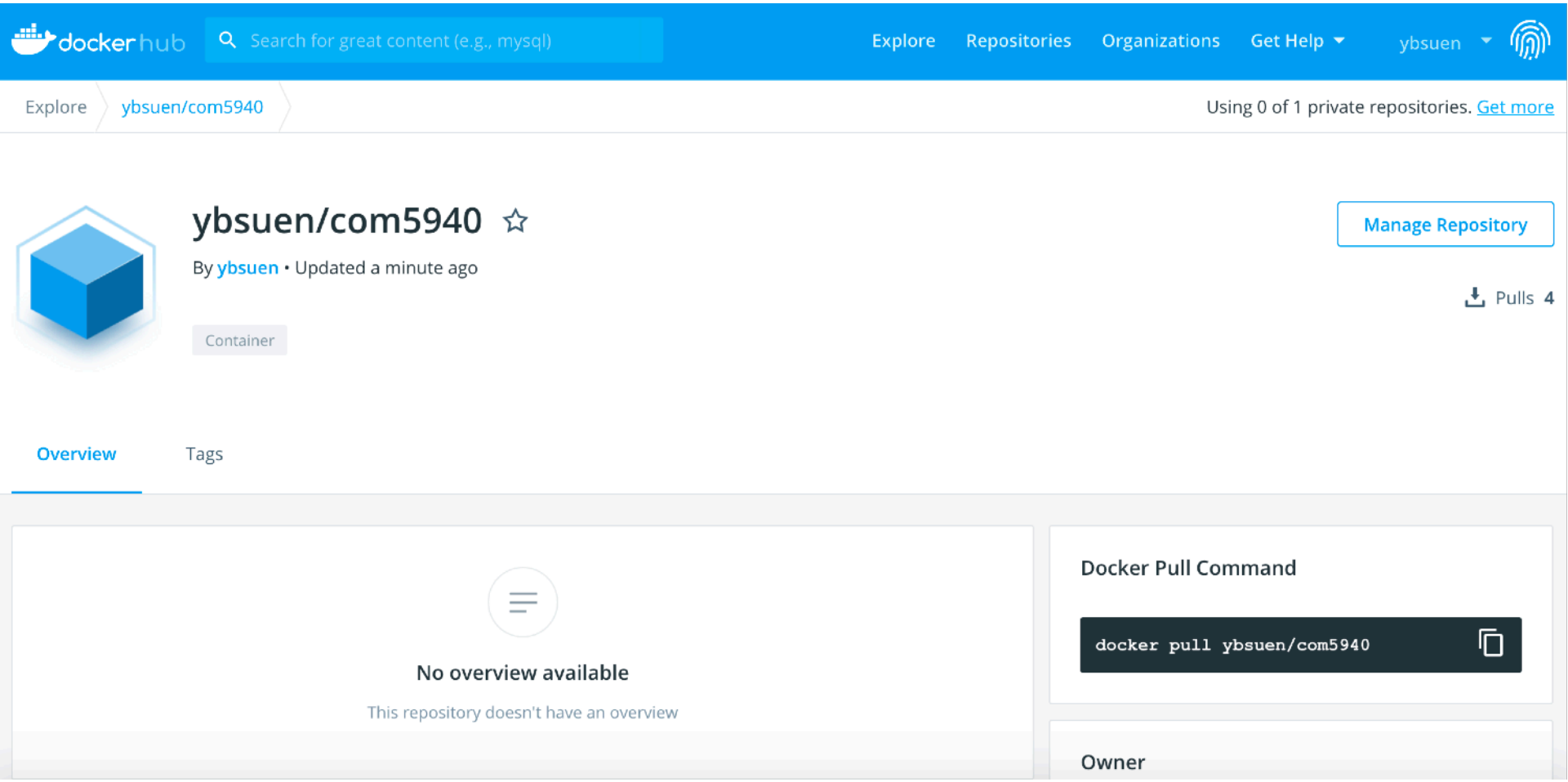
macinv

Source Repository

Github
macinv/docker-flask-example


macinv/flask-example - Docker
GitHub - macinv/docker-flask-e
ybsuen's Profile - Docker Hub
localhost:8080/
localhost:8080
Most Visited
Getting Started
Create Interac
Hello world! This is served by com5940!
2. Default (docker)
root@ubuntu: /home/bsuen (ssh) ⌘1
Default (docker) ⌘2
(base) YSs-MacBook-Pro:docker_test yssuen$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
3dcc2b063bbd ybsuen/com5940 "python app.py" 4 minutes ago Up 2 m
inutes 8080/tcp sharp_goldstine
8d7aa37e8040 macinv/flask-example "python app.py" 19 minutes ago Up 8 s
econds 0.0.0.0:8080->8080/tcp relaxed_jennings
196436fa16aa seqvence/static-site "/bin/sh -c 'cd /usr…" 6 days ago Exited
(137) 2 hours ago confident_almeida
(base) YSs-MacBook-Pro:docker_test yssuen$ docker exec -it 3dcc2b063bbd vi app.py
(base) YSs-MacBook-Pro:docker_test yssuen$ docker run -p 8080:8080 ybsuen/com5940
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 183-182-784
172.17.0.1 - - [29/Mar/2020 09:48:59] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [29/Mar/2020 09:48:59] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [29/Mar/2020 09:49:00] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [29/Mar/2020 09:49:00] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [29/Mar/2020 09:49:00] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [29/Mar/2020 09:49:00] "GET /favicon.ico HTTP/1.1" 404 -


docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
ybsuen
ybsuen
Search by repository name...
Create Repository
ybsuen / com5940
Updated a few seconds ago
0
1
PUBLIC
Tip: Not finding your repository? Try switching namespace via the top left dropdown.


docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
ybsuen
Explore
ybsuen/com5940
Using 0 of 1 private repositories. Get more
ybsuen/com5940
By ybsuen • Updated a minute ago
Container
Manage Repository
Pulls 4
Overview
Tags
No overview available
This repository doesn't have an overview
Docker Pull Command
docker pull ybsuen/com5940
Owner

Comparison Between IaaS and CaaS

	IaaS	CaaS
Ease of Setup		X
Installation & Maintenance of OS	X	
No Setup of Hardware	X	X
Ease of Deployment		X
Control Over Processing Environment	X	

From Containerization to Serverless

为什么说每个开发者都应该关注Serverless ？

快科技
发布时间：03-27 15:56 | 快科技官方百家号

如果说云计算是过去、现在和未来，那么Serverless就是云计算的未来。Gartner报告中表明，2020年全球将有20%的企业部署Serverless。Serverless以其学习成本低、运维友好、灵活性高的特点为开发、运维团队所喜爱，在国内外也开始走上了大规模考察乃至实践的应用之路。

但不可否认的是，Serverless还没有出现最佳实践，国内国外也都处在探索的过程中。3月21日，腾讯云TVP技术闭门会请来了国内外Serverless技术的佼佼者、使用者坐而论道，共话Serverless从概念到实践的发展之路。

作者最新文章

都说自己是影像旗舰 为何只有华为P40系列真正占住了？

云存储BackBlaze达成1EB硬盘：视频电话能打237823年

新光源？物理学家发现超越黑体


TVP Tencent Cloud Valuable Professional
Serverless 大咖说
“
腾讯内部支持微信小程序等核心应用的serverless技术，已对所有用户开放。提供一站式开发、部署，运维serverless解决方案。
”
Yunong Xiao
· 腾讯云中间件 总经理兼首席架构师
眺望曙光
TVP技术闭门会线上特辑

Source: 快科技


TVP Tencent Cloud Valuable Professional
Serverless 大咖说
“
Serverless 是一种未来的开发方式，属于每一个开发者。
”
王俊杰
· 腾讯云 Serverless 技术专家
眺望曙光
TVP技术闭门会线上特辑

知乎 首页 发现 等你来答 国内版 N 号房事件

前端开发 云服务 前端工程师 后端工程师 Serverless

前端为什么要关注 Serverless?

最近打算做一个serverless的项目，思考时产生一个疑问：前端为什么要接serverless这事？交给后端不行吗？不知道大家为什么在一直说什么...显示全部

关注问题 写回答 邀请回答 添加评论 分享 ...

37 个回答 默认排序

腾讯云 Serverless

Serverless爆发，AWS微软卡位，腾讯云阿里云华为云紧跟

发布时间：2020-03-17 17:47:55 来源：中国软件网 作者：刘学习

[摘要]Serverless落地应用增多，市场格局初定，云服务商纷纷试水，机会与挑战并存。

3月10日，云原生基金会CNCF发布了2019年年度调查报告。41%的受访者表示已经在使用无服务器Serverless计算，20%的受访者表示计划在未来12-18个月应用Serverless计算。

之前的3月1日，市场研究机构Forrester发布的最新报告认为，Serverless计算的兴起，让FaaS成为继IaaS、PaaS、SaaS之后一种新的云计算能力提供方式。

在Serverless计算最主要的技术方向——函数即服务平台(FaaS)方面，AWS、微软两家企业居于领导者象限，腾讯云、阿里云与谷歌云、Nimbella、IBM、Cloudflare等处于强劲表现者象限，Oracle则处于挑战者象限。

云计算企业已经习惯了“大投入有大回报”的行业规律。从2014年开始，云服务商不管是AWS、微软、谷歌云，还是国内的阿里云、腾讯云、华为云、百度智能云等，在Serverless上都有大笔投入。

2020年，Serverless服务市场将迎来大爆发的机会。而现在Serverless服务市场新座次的排

案例解读

查看更多

三一集团：不转型必死无疑，传统制造企业怎样破茧重生

临矿集团：浪潮云+助力开启数字化新征程

每日新闻

查看更多

- 3月27日：华为开发者大会鲲鹏+昇腾产业布局; 中国光通信企业遭遇美国调查
- 3月26日：中国新建4G和5G基站超6.3万个; 任

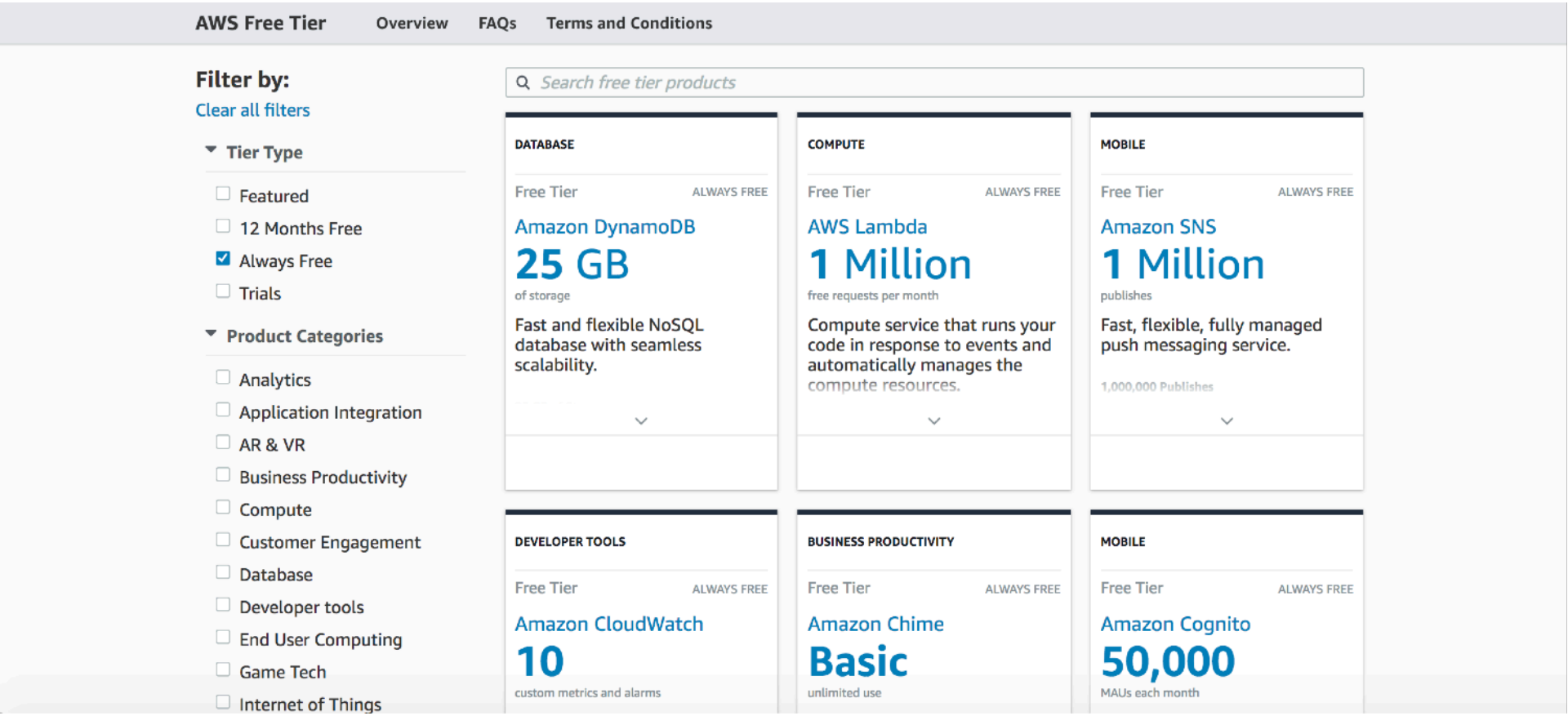
3月25日 16:00–17:00

直播 | 挑战不可能！闲鱼Serverless + Flutter研发效率提升30%

阿里技术 2020-03-25

随着无线、小程序、IoT的发展、5G的到来，移动研发越来越向多端化发展。传统的移动架构存

Why the widespread interest?


AWS Free Tier
Overview
FAQs
Terms and Conditions
Filter by:
Clear all filters
Tier Type
Featured
12 Months Free
Always Free
Trials
Product Categories
Analytics
Application Integration
AR & VR
Business Productivity
Compute
Customer Engagement
Database
Developer tools
End User Computing
Game Tech
Internet of Things
Search free tier products
DATABASE
Free Tier
ALWAYS FREE
Amazon DynamoDB
25 GB
of storage
Fast and flexible NoSQL database with seamless scalability.
COMPUTE
Free Tier
ALWAYS FREE
AWS Lambda
1 Million
free requests per month
Compute service that runs your code in response to events and automatically manages the compute resources.
MOBILE
Free Tier
ALWAYS FREE
Amazon SNS
1 Million
publishes
Fast, flexible, fully managed push messaging service.
1,000,000 Publishes
DEVELOPER TOOLS
Free Tier
ALWAYS FREE
Amazon CloudWatch
10
custom metrics and alarms
BUSINESS PRODUCTIVITY
Free Tier
ALWAYS FREE
Amazon Chime
Basic
unlimited use
MOBILE
Free Tier
ALWAYS FREE
Amazon Cognito
50,000
MAUs each month

linode

Why Linode | Products | Solutions | Pricing | Community | Sign Up

Configuration

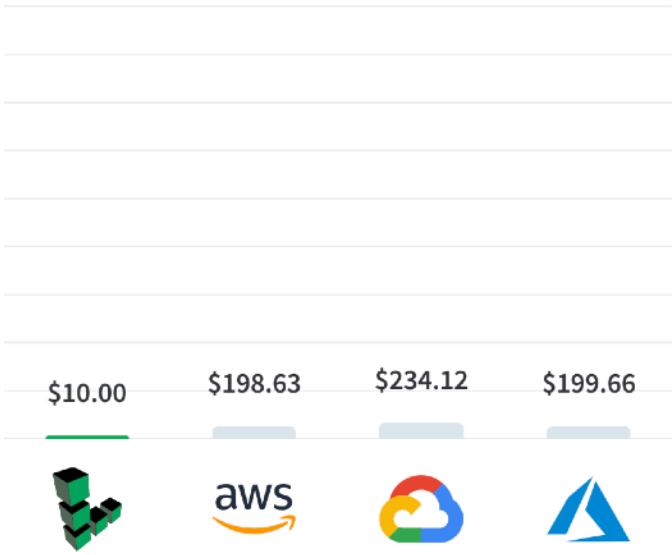
CPU	Memory	Storage	Transfer
1	2GB	50GB	2TB

Additional Storage

+0 TB

Additional Transfer

+0 TB



Let's look at compute instances:

2 vCPUs + 8 GB RAM

~ 0.0928 USD/ hour

2 vCPUs + 8 GB RAM

~ 0.096 USD/ hour

Source: SimplyLearn


INEXPENSIVE:
$0.00001667/GB/S
02:14
-08:39
PYCON
CANADA

Let's look at compute instances:

2 vCPUs + 8 GB RAM

~ 0.0928 USD/ hour

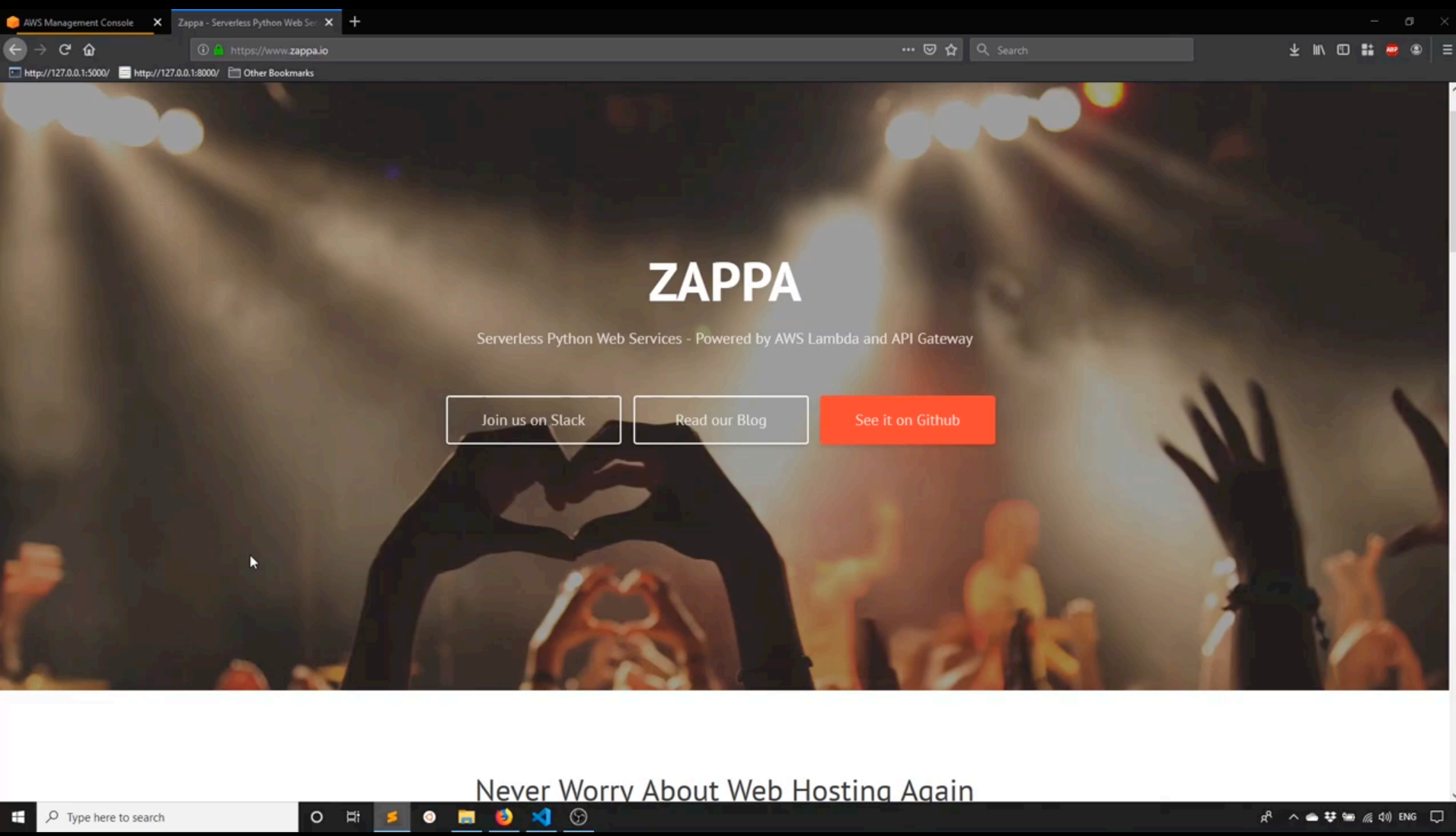
~ .060012 USD/ hour (**Lambda**)

2 vCPUs + 8 GB RAM

~ 0.096 USD/ hour

Source: SimplyLearn

Doesn’t charge for idle time (Since no VM instance is running).


AWS Management Console
Zappa - Serverless Python Web Ser
https://www.zappa.io
Search
http://127.0.0.1:5000/
http://127.0.0.1:8000/
Other Bookmarks
ZAPPA
Serverless Python Web Services - Powered by AWS Lambda and API Gateway
Join us on Slack
Read our Blog
See it on Github
Never Worry About Web Hosting Again
Type here to search
ENG

Comparison Between PaaS and FaaS

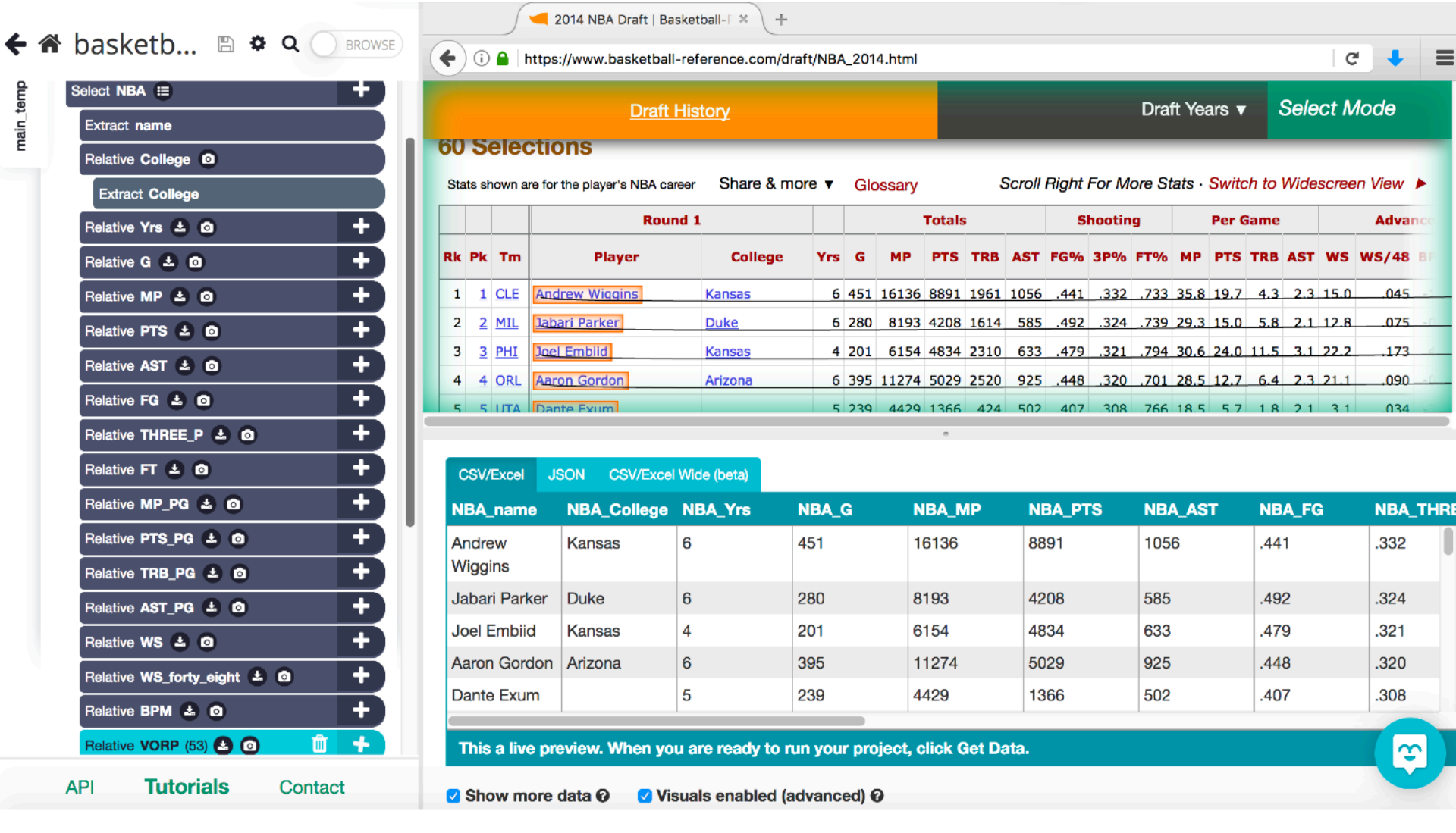
	PaaS	FaaS
Ease of Setup		X
Maintenance of Platform	X	
Setup of Servers	X	**Minimal**
No Payment for Idle Time		X
Control Over App Platform	X	

Latest Serverless Development in Cloud Computing

Data Wrangling and Cleaning with Pandas

Descriptive Statistics & Data Visualization

ParseHub vs. Scrapy/BeautifulSoup


basketb...
BROWSE
main_temp
Select NBA
Extract name
Relative College
Extract College
Relative Yrs
Relative G
Relative MP
Relative PTS
Relative AST
Relative FG
Relative THREE_P
Relative FT
Relative MP_PG
Relative PTS_PG
Relative TRB_PG
Relative AST_PG
Relative WS
Relative WS_forty_eight
Relative BPM
Relative VORP (53)
API
Tutorials
Contact
2014 NBA Draft | Basketball-
https://www.basketball-reference.com/draft/NBA_2014.html
Draft History
Draft Years
Select Mode
60 Selections
Stats shown are for the player's NBA career
Share & more
Glossary
Scroll Right For More Stats · Switch to Widescreen View
Round 1
Totals
Shooting
Per Game
Rk Pk Tm Player College Yrs G MP PTS TRB AST FG% 3P% FT% MP PTS TRB AST WS WS/48
1 1 CLE Andrew Wiggins Kansas 6 451 16136 8891 1961 1056 .441 .332 .733 35.8 19.7 4.3 2.3 15.0 .045
2 2 MIL Jabari Parker Duke 6 280 8193 4208 1614 585 .492 .324 .739 29.3 15.0 5.8 2.1 12.8 .075
3 3 PHI Joel Embiid Kansas 4 201 6154 4834 2310 633 .479 .321 .794 30.6 24.0 11.5 3.1 22.2 .173
4 4 ORL Aaron Gordon Arizona 6 395 11274 5029 2520 925 .448 .320 .701 28.5 12.7 6.4 2.3 21.1 .090
5 5 UTA Dante Exum 5 239 4429 1366 424 502 .407 .308 .766 18.5 5.7 1.8 2.1 3.1 .034
CSV/Excel
JSON
CSV/Excel Wide (beta)
NBA_name NBA_College NBA_Yrs NBA_G NBA_MP NBA_PTS NBA_AST NBA_FG NBA_THRE
Andrew Wiggins Kansas 6 451 16136 8891 1056 .441 .332
Jabari Parker Duke 6 280 8193 4208 585 .492 .324
Joel Embiid Kansas 4 201 6154 4834 633 .479 .321
Aaron Gordon Arizona 6 395 11274 5029 925 .448 .320
Dante Exum 5 239 4429 1366 502 .407 .308
This a live preview. When you are ready to run your project, click Get Data.
Show more data
Visuals enabled (advanced)

In [5]:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
import csv

url = "http://www.basketball-reference.com/draft/NBA_2014.html"
html = urlopen(url)
soup = BeautifulSoup(html,'html.parser')
column_headers = []
for th in soup.findAll('tr',limit=2)[1].findAll('th'):
    column_headers.append(th.getText())
data_rows = soup.findAll('tr')[2:]  # skip the first 2 header rows
player_data = []
for i in range(len(data_rows)):  # for each table row
    player_row = []
    # for each table data element from each table row
    for th in data_rows[i].findAll('th',limit=1):
        player_row.append(th.getText())
    for td in data_rows[i].findAll('td'):
        # get the text content and append to the player_row
        player_row.append(td.getText())
    # then append each pick/player to the player_data matrix
    player_data.append(player_row)
df = pd.DataFrame(player_data, columns=column_headers)
df.to_csv('projects/nba.csv', sep='\t', encoding='utf-8')
df.head(60)
```

Out[5]:

	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	TRB	...	3P%	FT%	MP	PTS	TRB	AST	WS	WS/48	BPM	VORP
0	1	1	CLE	Andrew Wiggins	Kansas	6	451	16136	8891	1961	...	.332	.733	35.8	19.7	4.3	2.3	15.0	.045	-1.9	0.3
1	2	2	MIL	Jabari Parker	Duke	6	280	8193	4208	1614	...	.324	.739	29.3	15.0	5.8	2.1	12.8	.075	-0.9	2.3
2	3	3	PHI	Joel Embiid	Kansas	4	201	6154	4834	2310	...	.321	.794	30.6	24.0	11.5	3.1	22.2	.173	4.4	10.0
3	4	4	ORL	Aaron Gordon	Arizona	6	396	11310	5040	2526	...	.319	.701	28.6	12.7	6.4	2.4	21.2	.090	-0.1	5.5
4	5	5	UTA	Dante Exum		5	239	4429	1366	424	...	.308	.766	18.5	5.7	1.8	2.1	3.1	.034	-3.3	-1.5

Scraping Data from Multiple Pages

Scraping Hong Kong Economic Journal (http://startupbeat.hkej.com)

```
1  import requests
2  import csv
3  import pandas as pd
4  from bs4 import BeautifulSoup
5
6  # quote_page = requests.get('http://startupbeat.hkej.com/?tag=fintech&paged=1')
7  # soup = BeautifulSoup(quote_page.text,'html.parser')
8
9  header = ['page #','title','url','details','post date']
10 data = []
11 # Display and store away 2 pages of scrapped data from startupbeat.hkej.com
12 for i in range(1,4):
13     quote_page = requests.get('http://startupbeat.hkej.com/?tag=fintech&paged='+str(i))
14     print("\n***** Page " + str(i) +" in action *****")
15     soup = BeautifulSoup(quote_page.content,'html.parser')
16
17     for article in soup.find_all("div", attrs={"class":"archive-text"}):
18     # for article in soup.find_all('div',class_='archive-text'):
19         page_no = str(i)
20         title = article.a.text.encode('utf-8').strip()
21         decoded_title = title.decode('utf-8')
22         url = article.a.get('href')
23         details = article.p.text.encode('utf-8').strip()
24         decoded_details = details.decode('utf-8')
25         post_date = article.div.ul.li.text
26         print(decoded_title)
27         print(url)
28         print(decoded_details)
```

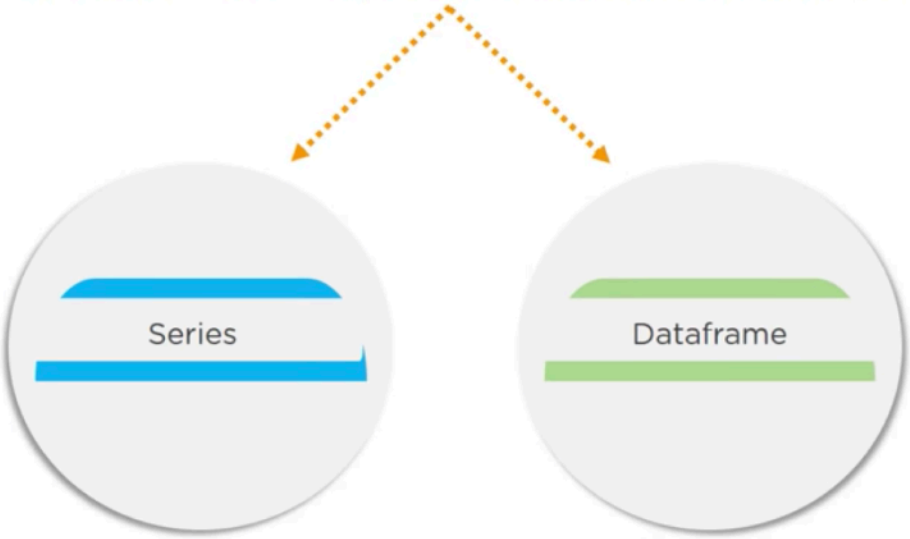
	page #	title	url	details	post date
0	1	無現金支付進一步突圍 (方保僑)	http://startupbeat.hkej.com/?p=85359	抗疫過程中，很多生活細節亦隨之改變，例如以往我經常會帶備小量現金傍身，但其實鈔票布滿細菌，所...	Posted March 23, 2020
1	1	金融科技協會憂肺疫礙融資	http://startupbeat.hkej.com/?p=85349	香港金融科技協會（FTAHK）創會兼董事會成員董欣怡受訪時指出，「我們估計（疫情）第一波影響...	Posted March 23, 2020
2	1	比特幣疫市瀉 避險價值淪喪 投資平台eToro：目前太投機 未來仍看好	http://startupbeat.hkej.com/?p=85333	比特幣（Bitcoin）一直為極具爭議的新類型「資產」隨着近期新冠肺炎疫情在全球爆發，股、債...	Posted March 23, 2020
3	1	黑客松大賽避疫Hack From Home	http://startupbeat.hkej.com/?p=85099	康宏（01019）及數碼港合力舉辦首屆「FINSPIRE FinTech Online Ha...	Posted March 16, 2020
4	1	港初創Oriente籌1.56億攻菲國FinTech	http://startupbeat.hkej.com/?p=84744	本港金融科技初創Oriente周三（4日）宣布，已從資產投資公司Silverhorn Gro...	Posted March 6, 2020
5	1	AI 平台配對融資服務 助港中小企疫市撲水	http://startupbeat.hkej.com/?p=84639	運用AI科技為企業配對融資服務的金融科技平台FinMonster，其創辦人鄭文耀及陳健明受訪...	Posted March 5, 2020
6	1	FinTech初創去年共吸2644億 逾億美元融資83宗 勁飆八成	http://startupbeat.hkej.com/?p=84225	科技媒體TechCrunch報道指出，全球金融科技的初創企業，去年共錄得1912宗融資交易，...	Posted February 25, 2020
7	1	雷蛇數碼金融瞄準年輕人 藉玩家商戶網絡 拓展東南亞	http://startupbeat.hkej.com/?p=84176	以開發電競及電子遊戲設備成名的雷蛇，近年積極拓展金融科技業務。公司早前宣布，已向新加坡申請數...	Posted February 24, 2020
8	1	OpenRice撐食肆 外賣自取免收佣	http://startupbeat.hkej.com/?p=84158	OpenRice《開飯喇》宣布，由2月21日起至4月30日將全額豁免「外賣自取」的服務佣金，...	Posted February 22, 2020
9	1	AI財策師助散戶穩健增值 度身訂造投資組合 一萬元入場	http://startupbeat.hkej.com/?p=84078	由多名前銀行家共同開發的智能投資平台Kristal.AI，利用人工智能科技分析風險承受力、目...	Posted February 20, 2020
10	2	本地加密幣交易商Amber估值7.8億	http://startupbeat.hkej.com/?p=83984	美國《福布斯》報道，本港加密貨幣交易科企Amber近日獲得A輪融資2800萬美元（約2.18...	Posted February 18, 2020
11	2	超級App正在形成 (老占)	http://startupbeat.hkej.com/?p=83640	馬雲曾經講過，如果銀行不改變，我們去改變銀行。二○一八年六月，螞蟻金服完成140億美元的C輪...	Posted February 8, 2020

Comparison Between ParseHub and BeautiSoup

	ParseHub	Scrapy/BeautifulSoup
Ease of Use	X	
Crawling	X	X
Parsing	X	X
Speed		X
Control		X

Exploratory analysis using Pandas

Let's understand the two most common terms used in Pandas:

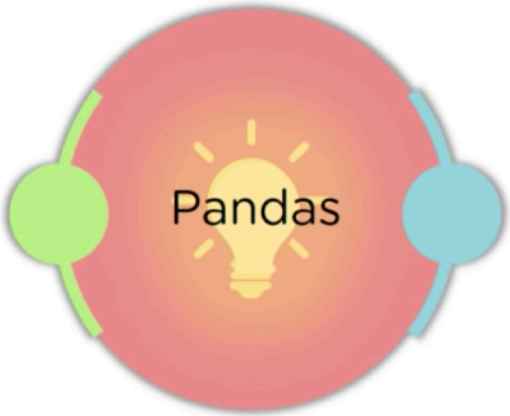


Source: SimplyLearn

Exploratory analysis using Pandas

Series

A Series is a one-dimensional object that can hold any data type such as integers, floats and strings



DataFrame

A DataFrame is a two dimensional object that can have columns with potential different data types

Source: SimplyLearn

From List & Dictionary to DataFrame

In [20]:

```
# Define a list of dictionaries
sales = [{'account': 'Jones LLC', 'Jan': 150, 'Feb': 200, 'Mar': 140},
         {'account': 'Alpha Co',  'Jan': 200, 'Feb': 210, 'Mar': 215},
         {'account': 'Blue Inc',  'Jan': 50,  'Feb': 90,  'Mar': 95 }]
df = pd.DataFrame(sales) # Assign the list of dictionaries to a dataframe
df # display dataframe
```

Out[20]:

	account	Jan	Feb	Mar
0	Jones LLC	150	200	140
1	Alpha Co	200	210	215
2	Blue Inc	50	90	95

In [21]:

```
# Define a dictionary of list
sales = {'account': ['Jones LLC', 'Alpha Co', 'Blue Inc'],
         'Jan': [150, 200, 50],
         'Feb': [200, 210, 90],
         'Mar': [140, 215, 95]}
df = pd.DataFrame.from_dict(sales) # Assign the dictionary of lists to a dataframe
df # display dataframe
```

Out[21]:

	account	Jan	Feb	Mar
0	Jones LLC	150	200	140
1	Alpha Co	200	210	215
2	Blue Inc	50	90	95

Saving Dataframe to CSV and Retrieving CSV into Dataframe

In [5]:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
import csv

url = "http://www.basketball-reference.com/draft/NBA_2014.html"
html = urlopen(url)
soup = BeautifulSoup(html,'html.parser')
column_headers = []
for th in soup.findAll('tr',limit=2)[1].findAll('th'):
    column_headers.append(th.getText())
data_rows = soup.findAll('tr')[2:]  # skip the first 2 header rows
player_data = []
for i in range(len(data_rows)):  # for each table row
    player_row = []
    # for each table data element from each table row
    for th in data_rows[i].findAll('th',limit=1):
        player_row.append(th.getText())
    for td in data_rows[i].findAll('td'):
        # get the text content and append to the player_row
        player_row.append(td.getText())
    # then append each pick/player to the player_data matrix
    player_data.append(player_row)
df = pd.DataFrame(player_data, columns=column_headers)
df.to_csv('projects/nba.csv', sep='\t', encoding='utf-8')
df.head(60)
```

Out[5]:

	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	TRB	...	3P%	FT%	MP	PTS	TRB	AST	WS	WS/48	BPM	VORP
0	1	1	CLE	Andrew Wiggins	Kansas	6	451	16136	8891	1961	...	.332	.733	35.8	19.7	4.3	2.3	15.0	.045	-1.9	0.3
1	2	2	MIL	Jabari Parker	Duke	6	280	8193	4208	1614	...	.324	.739	29.3	15.0	5.8	2.1	12.8	.075	-0.9	2.3
2	3	3	PHI	Joel Embiid	Kansas	4	201	6154	4834	2310	...	.321	.794	30.6	24.0	11.5	3.1	22.2	.173	4.4	10.0
3	4	4	ORL	Aaron Gordon	Arizona	6	396	11310	5040	2526	...	.319	.701	28.6	12.7	6.4	2.4	21.2	.090	-0.1	5.5
4	5	5	UTA	Dante Exum		5	239	4429	1366	424	...	.308	.766	18.5	5.7	1.8	2.1	3.1	.034	-3.3	-1.5

In [31]:

```
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
```

Out[31]:

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
0	0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0	454.0	16242.0	8943.0	...	0.332	0.732	35.8	19.7	4.4	2.3	15.0	0.044	-1.9	0.2
1	1	2	2.0	MIL	Jabari Parker	Duke	6.0	280.0	8193.0	4208.0	...	0.324	0.739	29.3	15.0	5.8	2.1	12.8	0.075	-0.9	2.3
2	2	3	3.0	PHI	Joel Embiid	Kansas	4.0	202.0	6181.0	4864.0	...	0.322	0.794	30.6	24.1	11.5	3.1	22.4	0.174	4.5	10.0
3	3	4	4.0	ORL	Aaron Gordon	Arizona	6.0	399.0	11414.0	5082.0	...	0.318	0.701	28.6	12.7	6.4	2.4	21.4	0.090	-0.1	5.5
4	4	5	5.0	UTA	Dante Exum	NaN	5.0	239.0	4429.0	1366.0	...	0.308	0.766	18.5	5.7	1.8	2.1	3.1	0.034	-3.3	-1.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	57	56	56.0	DEN	Devyn Marble	Iowa	2.0	44.0	457.0	97.0	...	0.222	0.375	10.4	2.2	1.6	0.7	-0.3	-0.028	-5.5	-0.4
58	58	57	57.0	IND	Louis Labeyrie	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
59	59	58	58.0	SAS	Jordan McRae	Tennessee	4.0	123.0	1696.0	846.0	...	0.355	0.772	13.8	6.9	1.8	1.4	1.7	0.048	-2.1	0.0
60	60	59	59.0	TOR	Xavier Thames	San Diego State	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
61	61	60	60.0	SAS	Cory Jefferson	Baylor	2.0	58.0	581.0	205.0	...	0.125	0.583	10.0	3.5	2.8	0.3	0.9	0.071	-3.3	-0.2

Slicing and Dicing the Data

In [31]:

```
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
```

Out[31]:

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
0	0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0	454.0	16242.0	8943.0	...	0.332	0.732	35.8	19.7	4.4	2.3	15.0	0.044	-1.9	0.2
1	1	2	2.0	MIL	Jabari Parker	Duke	6.0	280.0	8193.0	4208.0	...	0.324	0.739	29.3	15.0	5.8	2.1	12.8	0.075	-0.9	2.3
2	2	3	3.0	PHI	Joel Embiid	Kansas	4.0	202.0	6181.0	4864.0	...	0.322	0.794	30.6	24.1	11.5	3.1	22.4	0.174	4.5	10.0
3	3	4	4.0	ORL	Aaron Gordon	Arizona	6.0	399.0	11414.0	5082.0	...	0.318	0.701	28.6	12.7	6.4	2.4	21.4	0.090	-0.1	5.5
4	4	5	5.0	UTA	Dante Exum	NaN	5.0	239.0	4429.0	1366.0	...	0.308	0.766	18.5	5.7	1.8	2.1	3.1	0.034	-3.3	-1.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	57	56	56.0	DEN	Devyn Marble	Iowa	2.0	44.0	457.0	97.0	...	0.222	0.375	10.4	2.2	1.6	0.7	-0.3	-0.028	-5.5	-0.4
58	58	57	57.0	IND	Louis Labeyrie	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
59	59	58	58.0	SAS	Jordan McRae	Tennessee	4.0	123.0	1696.0	846.0	...	0.355	0.772	13.8	6.9	1.8	1.4	1.7	0.048	-2.1	0.0
60	60	59	59.0	TOR	Xavier Thames	San Diego State	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
61	61	60	60.0	SAS	Cory Jefferson	Baylor	2.0	58.0	581.0	205.0	...	0.125	0.583	10.0	3.5	2.8	0.3	0.9	0.071	-3.3	-0.2

In [32]:
```
1 df.index # show number of entries
```
showing rows

Out[32]:
```
RangeIndex(start=0, stop=62, step=1)
```

In [33]:
```
1 df.info() # show field names and types
```
showing columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 23 columns):
Unnamed: 0    62 non-null int64
Rk            61 non-null object
Pk            60 non-null float64
Tm            60 non-null object
Player        60 non-null object
College       46 non-null object
Yrs           53 non-null float64
G             53 non-null float64
MP            53 non-null float64
PTS           53 non-null float64
TRB           53 non-null float64
AST           53 non-null float64
FG%           53 non-null float64
3P%           51 non-null float64
FT%           53 non-null float64
MP.1          53 non-null float64
PTS.1         53 non-null float64
TRB.1         53 non-null float64
AST.1         53 non-null float64
WS            53 non-null float64
WS/48         53 non-null float64
BPM           53 non-null float64
VORP          53 non-null float64
dtypes: float64(18), int64(1), object(4)
```

In [31]:

```
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
```

Out[31]:

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
0	0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0	454.0	16242.0	8943.0	...	0.332	0.732	35.8	19.7	4.4	2.3	15.0	0.044	-1.9	0.2
1	1	2	2.0	MIL	Jabari Parker	Duke	6.0	280.0	8193.0	4208.0	...	0.324	0.739	29.3	15.0	5.8	2.1	12.8	0.075	-0.9	2.3
2	2	3	3.0	PHI	Joel Embiid	Kansas	4.0	202.0	6181.0	4864.0	...	0.322	0.794	30.6	24.1	11.5	3.1	22.4	0.174	4.5	10.0
3	3	4	4.0	ORL	Aaron Gordon	Arizona	6.0	399.0	11414.0	5082.0	...	0.318	0.701	28.6	12.7	6.4	2.4	21.4	0.090	-0.1	5.5
4	4	5	5.0	UTA	Dante Exum	NaN	5.0	239.0	4429.0	1366.0	...	0.308	0.766	18.5	5.7	1.8	2.1	3.1	0.034	-3.3	-1.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	57	56	56.0	DEN	Devyn Marble	Iowa	2.0	44.0	457.0	97.0	...	0.222	0.375	10.4	2.2	1.6	0.7	-0.3	-0.028	-5.5	-0.4
58	58	57	57.0	IND	Louis Labeyrie	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
59	59	58	58.0	SAS	Jordan McRae	Tennessee	4.0	123.0	1696.0	846.0	...	0.355	0.772	13.8	6.9	1.8	1.4	1.7	0.048	-2.1	0.0
60	60	59	59.0	TOR	Xavier Thames	San Diego State	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
61	61	60	60.0	SAS	Cory Jefferson	Baylor	2.0	58.0	581.0	205.0	...	0.125	0.583	10.0	3.5	2.8	0.3	0.9	0.071	-3.3	-0.2

In [37]:
```
df.loc[0:6,'Rk':'Yrs'] # Slicing dataframe by row range and column range: df.loc[row range,column range]
```

Out[37]:

	Rk	Pk	Tm	Player	College	Yrs
0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0
1	2	2.0	MIL	Jabari Parker	Duke	6.0
2	3	3.0	PHI	Joel Embiid	Kansas	4.0
3	4	4.0	ORL	Aaron Gordon	Arizona	6.0
4	5	5.0	UTA	Dante Exum	NaN	5.0
5	6	6.0	BOS	Marcus Smart	Oklahoma State	6.0
6	7	7.0	LAL	Julius Randle	Kentucky	6.0

In [16]:
```
# Notice the difference in indexing between Pandas Frame and data structures such as string and list.
x = "this is a string"
print(x[0:6])
y = [1,2,3,4,5,6,7]
print(y[0:6])
```

```
this i
[1, 2, 3, 4, 5, 6]
```

In [38]:
```
df.loc[[0,2,4],['Rk','Tm','College']] # Selectively pick row 0,2&4 of column NBA_name, NBA_Pk, & NBA_Tm
```

Out[38]:

	Rk	Tm	College
0	1	CLE	Kansas
2	3	PHI	Kansas
4	5	UTA	NaN

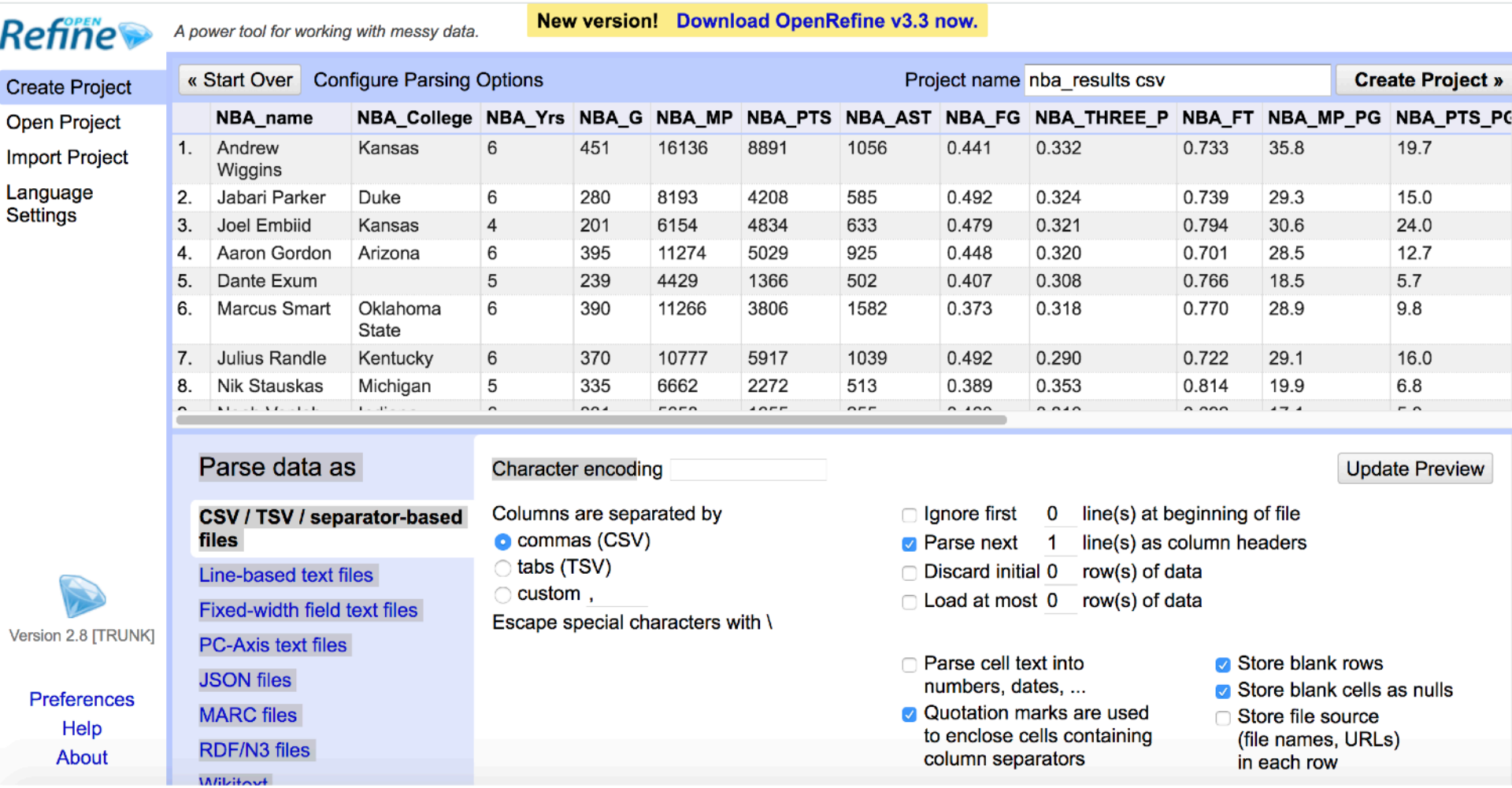
In [39]:
```
df.loc[0:4,['Player','College','Rk']] # Print the range from 0 to 4 of only column NBA_name,
                                      # NBA_Pk, and NBA_Tm.
```

Out[39]:

	Player	College	Rk
0	Andrew Wiggins	Kansas	1
1	Jabari Parker	Duke	2
2	Joel Embiid	Kansas	3
3	Aaron Gordon	Arizona	4
4	Dante Exum	NaN	5

Cleaning

Open Refine vs. Pandas


OpenRefine
A power tool for working with messy data.
New version! Download OpenRefine v3.3 now.
Create Project
Open Project
Import Project
Language Settings
« Start Over
Configure Parsing Options
Project name nba_results csv
Create Project »
NBA_name NBA_College NBA_Yrs NBA_G NBA_MP NBA_PTS NBA_AST NBA_FG NBA_THREE_P NBA_FT NBA_MP_PG NBA_PTS_PG
1. Andrew Wiggins Kansas 6 451 16136 8891 1056 0.441 0.332 0.733 35.8 19.7
2. Jabari Parker Duke 6 280 8193 4208 585 0.492 0.324 0.739 29.3 15.0
3. Joel Embiid Kansas 4 201 6154 4834 633 0.479 0.321 0.794 30.6 24.0
4. Aaron Gordon Arizona 6 395 11274 5029 925 0.448 0.320 0.701 28.5 12.7
5. Dante Exum 5 239 4429 1366 502 0.407 0.308 0.766 18.5 5.7
6. Marcus Smart Oklahoma State 6 390 11266 3806 1582 0.373 0.318 0.770 28.9 9.8
7. Julius Randle Kentucky 6 370 10777 5917 1039 0.492 0.290 0.722 29.1 16.0
8. Nik Stauskas Michigan 5 335 6662 2272 513 0.389 0.353 0.814 19.9 6.8
Parse data as
CSV / TSV / separator-based files
Line-based text files
Fixed-width field text files
PC-Axis text files
JSON files
MARC files
RDF/N3 files
Character encoding
Update Preview
Columns are separated by
commas (CSV)
tabs (TSV)
custom ,
Escape special characters with \
Ignore first 0 line(s) at beginning of file
Parse next 1 line(s) as column headers
Discard initial 0 row(s) of data
Load at most 0 row(s) of data
Parse cell text into numbers, dates, ...
Quotation marks are used to enclose cells containing column separators
Store blank rows
Store blank cells as nulls
Store file source (file names, URLs) in each row
Version 2.8 [TRUNK]
Preferences
Help
About

	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	TRB	AST	FG%	3P%	FT%	MP	PTS	TRB	AST	WS	WS/48	BPM	V
0	1	1	CLE	Andrew Wiggins	Kansas	6	451	16136	8891	1961	1056	.441	.332	.733	35.8	19.7	4.3	2.3	15.0	.045	-1.9	
1	2	2	MIL	Jabari Parker	Duke	6	280	8193	4208	1614	585	.492	.324	.739	29.3	15.0	5.8	2.1	12.8	.075	-0.9	
2	3	3	PHI	Joel Embiid	Kansas	4	201	6154	4834	2310	633	.479	.321	.794	30.6	24.0	11.5	3.1	22.2	.173	4.4	
3	4	4	ORL	Aaron Gordon	Arizona	6	396	11310	5040	2526	934	.448	.319	.701	28.6	12.7	6.4	2.4	21.2	.090	-0.1	
4	5	5	UTA	Dante Exum		5	239	4429	1366	424	502	.407	.308	.766	18.5	5.7	1.8	2.1	3.1	.034	-3.3	
5	6	6	BOS	Marcus Smart	Oklahoma State	6	391	11298	3824	1391	1584	.373	.317	.772	28.9	9.8	3.6	4.1	20.1	.085	-0.3	
6	7	7	LAL	Julius Randle	Kentucky	6	371	10809	5949	3339	1039	.493	.292	.722	29.1	16.0	9.0	2.8	20.2	.090	-0.5	
7	8	8	SAC	Nik Stauskas	Michigan	5	335	6662	2272	688	513	.389	.353	.814	19.9	6.8	2.1	1.5	3.8	.028	-3.0	
8	9	9	CHH	Noah Vonleh	Indiana	6	331	5658	1655	1739	255	.460	.310	.692	17.1	5.0	5.3	0.8	9.3	.079	-2.3	
9	10	10	PHI	Elfrid Payton	LA-Lafayette	6	383	11236	4199	1682	2536	.452	.290	.624	29.3	11.0	4.4	6.6	14.7	.063	-0.5	
10	11	11	DEN	Doug McDermott	Creighton	6	403	8040	3302	882	339	.465	.413	.825	20.0	8.2	2.2	0.8	13.6	.081	-2.1	
11	12	12	ORL	Dario Šarić		4	295	7800	3566	1811	617	.436	.352	.837	26.4	12.1	6.1	2.1	13.4	.083	-0.9	
12	13	13	MIN	Zach LaVine	UCLA	6	353	10857	6240	1279	1275	.447	.375	.819	30.8	17.7	3.6	3.6	12.0	.053	-0.6	
13	14	14	PHO	T.J. Warren	NC State	6	320	9151	4856	1307	379	.504	.351	.775	28.6	15.2	4.1	1.2	18.1	.095	-0.4	
14	15	15	ATL	Adreian Payne	Michigan State	4	107	1403	429	315	66	.406	.254	.680	13.1	4.0	2.9	0.6	-0.6	-.019	-5.9	
15	16	16	CHI	Jusuf Nurkić		5	310	7103	3608	2480	589	.491	.071	.666	22.9	11.6	8.0	1.9	16.4	.111	0.2	
16	17	17	BOS	James Young	Kentucky	4	95	812	219	96	28	.367	.277	.563	8.5	2.3	1.0	0.3	0.8	.046	-3.4	

1. Remove incomplete entries (null/ empty cells)
2. Identify inconsistency (duplicates and inconsistent names)
3. Fix incorrect data entries (e.g. wrong data types)
4. Rename, combine and split columns

Do you still remember how we did that in Open Refine?

Eliminate Null Values

In [78]:

```
# Display all the entries (i.e.rows) with the "college" column has null value
# Finding null values
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
df[df['College'].isnull()]
```

Out[78]:

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VOR
4	4	5	5.0	UTA	Dante Exum	NaN	5.0	239.0	4429.0	1366.0	...	0.308	0.766	18.5	5.7	1.8	2.1	3.1	0.034	-3.3	-1.
11	11	12	12.0	ORL	Dario Šarić	NaN	4.0	298.0	7903.0	3625.0	...	0.356	0.835	26.5	12.2	6.2	2.1	13.9	0.084	-0.8	2.
15	15	16	16.0	CHI	Jusuf Nurkić	NaN	5.0	310.0	7103.0	3608.0	...	0.071	0.666	22.9	11.6	8.0	1.9	16.4	0.111	0.2	3.
19	19	20	20.0	TOR	Bruno Caboclo	NaN	6.0	96.0	1233.0	407.0	...	0.314	0.836	12.8	4.2	2.6	0.7	1.5	0.057	-2.4	-0.
24	24	25	25.0	HOU	Clint Capela	NaN	6.0	334.0	8674.0	4075.0	...	0.000	0.526	26.0	12.2	9.7	1.0	36.4	0.201	1.9	8.
26	26	27	27.0	PHO	Bogdan Bogdanović	NaN	3.0	201.0	5632.0	2674.0	...	0.370	0.803	28.0	13.3	3.2	3.5	8.0	0.068	-0.7	1.
30	30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
31	31	Rk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
32	32	31	31.0	MIL	Damien Inglis	NaN	1.0	20.0	156.0	36.0	...	0.231	0.875	7.8	1.8	1.6	0.5	-0.1	-0.024	-5.6	-0.
42	42	41	41.0	DEN	Nikola Jokić	NaN	5.0	373.0	10819.0	6319.0	...	0.338	0.825	29.0	16.9	9.7	5.4	48.0	0.213	7.2	25.
44	44	43	43.0	ATL	Edy Tavares	NaN	2.0	13.0	101.0	33.0	...	NaN	0.273	7.8	2.5	2.5	0.3	0.2	0.118	-0.4	0.
52	52	51	51.0	NYK	Thanasis Antetokounmpo	NaN	2.0	20.0	99.0	51.0	...	0.000	0.467	5.0	2.6	1.0	0.5	0.2	0.095	-1.9	0.
53	53	52	52.0	PHI	Vasilije Micić	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
54	54	53	53.0	MIN	Alessandro Gentile	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal

In [77]:

```
# Finding null values
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
df1 = df.dropna(subset=['College'])
df1
```

Out[77]:

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
0	0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0	454.0	16242.0	8943.0	...	0.332	0.732	35.8	19.7	4.4	2.3	15.0	0.044	-1.9	0.2
1	1	2	2.0	MIL	Jabari Parker	Duke	6.0	280.0	8193.0	4208.0	...	0.324	0.739	29.3	15.0	5.8	2.1	12.8	0.075	-0.9	2.3
2	2	3	3.0	PHI	Joel Embiid	Kansas	4.0	202.0	6181.0	4864.0	...	0.322	0.794	30.6	24.1	11.5	3.1	22.4	0.174	4.5	10.0
3	3	4	4.0	ORL	Aaron Gordon	Arizona	6.0	399.0	11414.0	5082.0	...	0.318	0.701	28.6	12.7	6.4	2.4	21.4	0.090	-0.1	5.5
5	5	6	6.0	BOS	Marcus Smart	Oklahoma State	6.0	394.0	11409.0	3888.0	...	0.318	0.774	29.0	9.9	3.6	4.0	20.3	0.086	-0.3	4.9
6	6	7	7.0	LAL	Julius Randle	Kentucky	6.0	375.0	10934.0	6032.0	...	0.295	0.724	29.2	16.1	9.0	2.8	20.5	0.090	-0.5	4.1
7	7	8	8.0	SAC	Nik Stauskas	Michigan	5.0	335.0	6662.0	2272.0	...	0.353	0.814	19.9	6.8	2.1	1.5	3.8	0.028	-3.0	-1.7

In [87]:

```
# Finding null values
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
```

In [87]:

```
# Finding null values
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
```

Out[87]:

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
0	0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0	454.0	16242.0	8943.0	...	0.332	0.732	35.8	19.7	4.4	2.3	15.0	0.044	-1.9	0.2
1	1	2	2.0	MIL	Jabari Parker	Duke	6.0	280.0	8193.0	4208.0	...	0.324	0.739	29.3	15.0	5.8	2.1	12.8	0.075	-0.9	2.3
2	2	3	3.0	PHI	Joel Embiid	Kansas	4.0	202.0	6181.0	4864.0	...	0.322	0.794	30.6	24.1	11.5	3.1	22.4	0.174	4.5	10.0
3	3	4	4.0	ORL	Aaron Gordon	Arizona	6.0	399.0	11414.0	5082.0	...	0.318	0.701	28.6	12.7	6.4	2.4	21.4	0.090	-0.1	5.5
4	4	5	5.0	UTA	Dante Exum	NaN	5.0	239.0	4429.0	1366.0	...	0.308	0.766	18.5	5.7	1.8	2.1	3.1	0.034	-3.3	-1.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	57	56	56.0	DEN	Devyn Marble	Iowa	2.0	44.0	457.0	97.0	...	0.222	0.375	10.4	2.2	1.6	0.7	-0.3	-0.028	-5.5	-0.4
58	58	57	57.0	IND	Louis Labeyrie	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
59	59	58	58.0	SAS	Jordan McRae	Tennessee	4.0	123.0	1696.0	846.0	...	0.355	0.772	13.8	6.9	1.8	1.4	1.7	0.048	-2.1	0.0
60	60	59	59.0	TOR	Xavier Thames	San Diego State	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
61	61	60	60.0	SAS	Cory Jefferson	Baylor	2.0	58.0	581.0	205.0	...	0.125	0.583	10.0	3.5	2.8	0.3	0.9	0.071	-3.3	-0.2

Eliminate Duplicate Values

```python
# import pandas as pd
import numpy as np

#Create a DataFrame
d = {
    'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',
            'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],
    'Age':[26,24,23,22,23,24,26,24,22,23,24,24],

       'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}

df = pd.DataFrame(d,columns=['Name','Age','Score'])
df
```

	Name	Age	Score
0	Alisa	26	85
1	Bobby	24	63
2	jodha	23	55
3	jack	22	74
4	raghu	23	31
5	Cathrine	24	77
6	Alisa	26	85
7	Bobby	24	63
8	kumar	22	42
9	Alisa	23	62
10	Alex	24	89

```
df["is_duplicate"]= df.duplicated()
df
```

	Name	Age	Score	is_duplicate
0	Alisa	26	85	False
1	Bobby	24	63	False
2	jodha	23	55	False
3	jack	22	74	False
4	raghu	23	31	False
5	Cathrine	24	77	False
6	Alisa	26	85	True
7	Bobby	24	63	True
8	kumar	22	42	False
9	Alisa	23	62	False
10	Alex	24	89	False
11	Cathrine	24	77	True

```
df_removed=df.loc[0:11,'Name':'Score']
df_removed
```

	Name	Age	Score
0	Alisa	26	85
1	Bobby	24	63
2	jodha	23	55
3	jack	22	74
4	raghu	23	31
5	Cathrine	24	77
6	Alisa	26	85
7	Bobby	24	63
8	kumar	22	42
9	Alisa	23	62
10	Alex	24	89
11	Cathrine	24	77

```
df = df_removed.drop_duplicates()
df
```

	Name	Age	Score
0	Alisa	26	85
1	Bobby	24	63
2	jodha	23	55
3	jack	22	74
4	raghu	23	31
5	Cathrine	24	77
8	kumar	22	42
9	Alisa	23	62
10	Alex	24	89

```
# Display all the entries (i.e.rows) with the "college" column has null value
# Finding null values
import pandas as pd
df = pd.read_csv("projects/nba.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
df[df['College'].isnull()]
```

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VOR
4	4	5	5.0	UTA	Dante Exum	NaN	5.0	239.0	4429.0	1366.0	...	0.308	0.766	18.5	5.7	1.8	2.1	3.1	0.034	-3.3	-1.
11	11	12	12.0	ORL	Dario Šarić	NaN	4.0	298.0	7903.0	3625.0	...	0.356	0.835	26.5	12.2	6.2	2.1	13.9	0.084	-0.8	2.
15	15	16	16.0	CHI	Jusuf Nurkić	NaN	5.0	310.0	7103.0	3608.0	...	0.071	0.666	22.9	11.6	8.0	1.9	16.4	0.111	0.2	3.
19	19	20	20.0	TOR	Bruno Caboclo	NaN	6.0	96.0	1233.0	407.0	...	0.314	0.836	12.8	4.2	2.6	0.7	1.5	0.057	-2.4	-0.
24	24	25	25.0	HOU	Clint Capela	NaN	6.0	334.0	8674.0	4075.0	...	0.000	0.526	26.0	12.2	9.7	1.0	36.4	0.201	1.9	8.
26	26	27	27.0	PHO	Bogdan Bogdanović	NaN	3.0	201.0	5632.0	2674.0	...	0.370	0.803	28.0	13.3	3.2	3.5	8.0	0.068	-0.7	1.
30	30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
31	31	Rk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
32	32	31	31.0	MIL	Damien Inglis	NaN	1.0	20.0	156.0	36.0	...	0.231	0.875	7.8	1.8	1.6	0.5	-0.1	-0.024	-5.6	-0.
42	42	41	41.0	DEN	Nikola Jokić	NaN	5.0	373.0	10819.0	6319.0	...	0.338	0.825	29.0	16.9	9.7	5.4	48.0	0.213	7.2	25.
44	44	43	43.0	ATL	Edy Tavares	NaN	2.0	13.0	101.0	33.0	...	NaN	0.273	7.8	2.5	2.5	0.3	0.2	0.118	-0.4	0.
52	52	51	51.0	NYK	Thanasis Antetokounmpo	NaN	2.0	20.0	99.0	51.0	...	0.000	0.467	5.0	2.6	1.0	0.5	0.2	0.095	-1.9	0.
53	53	52	52.0	PHI	Vasilije Micić	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
54	54	53	53.0	MIN	Alessandro Gentile	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal

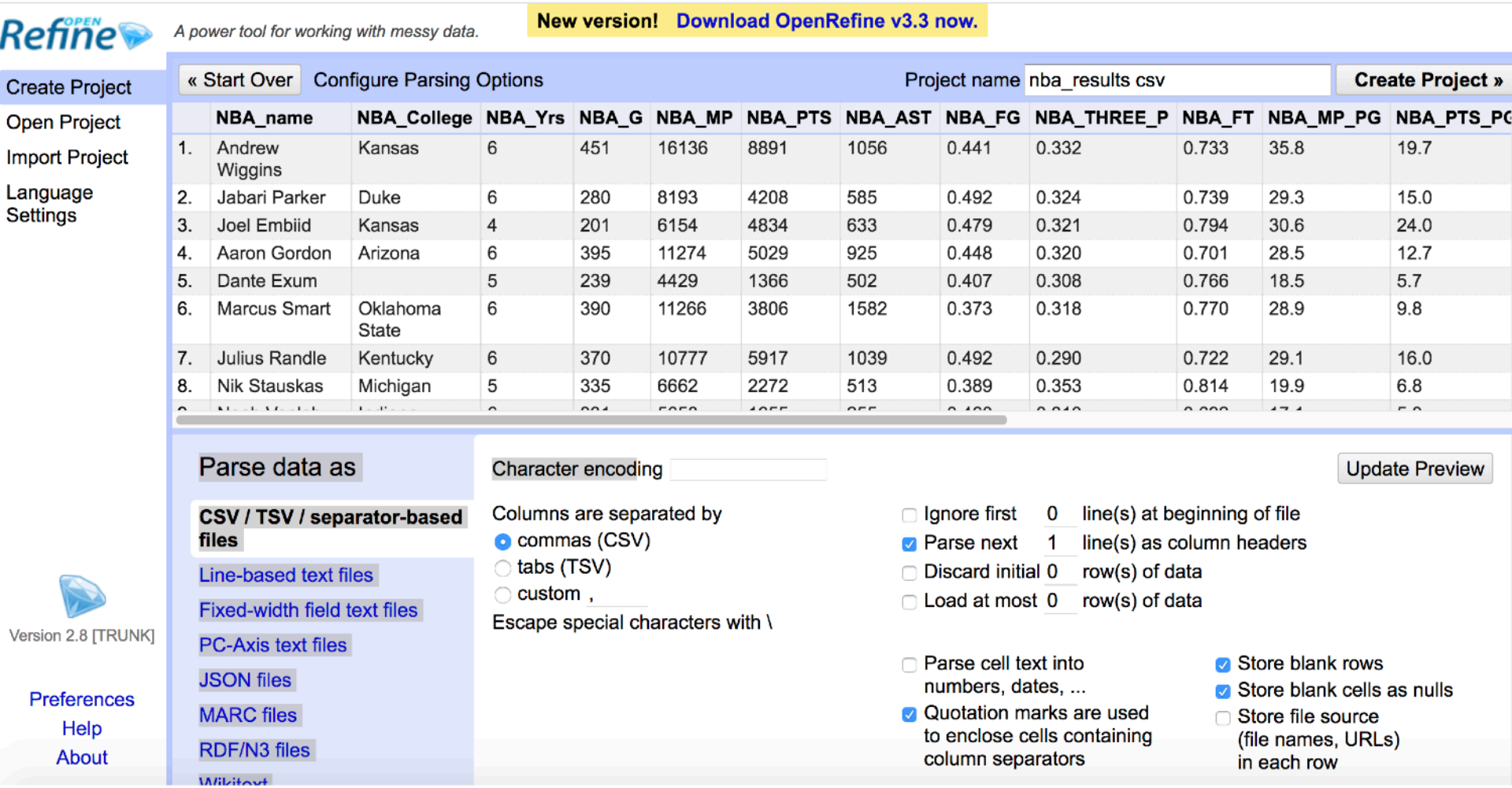
```
1 df[df['Pk'].isnull()]
```

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
30	30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	31	Rk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2 rows × 23 columns

```
1 df[df['G'] > 400]
```

	Unnamed: 0	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	...	3P%	FT%	MP.1	PTS.1	TRB.1	AST.1	WS	WS/48	BPM	VORP
0	0	1	1.0	CLE	Andrew Wiggins	Kansas	6.0	454.0	16242.0	8943.0	...	0.332	0.732	35.8	19.7	4.4	2.3	15.0	0.044	-1.9	0.2
10	10	11	11.0	DEN	Doug McDermott	Creighton	6.0	403.0	8040.0	3302.0	...	0.413	0.825	20.0	8.2	2.2	0.8	13.6	0.081	-2.1	-0.1
40	40	39	39.0	PHI	Jerami Grant	Syracuse	6.0	447.0	10909.0	4111.0	...	0.348	0.666	24.4	9.2	3.9	1.1	22.3	0.098	-0.9	3.1
47	47	46	46.0	WAS	Jordan Clarkson	Missouri	6.0	445.0	12054.0	6573.0	...	0.341	0.818	27.1	14.8	3.2	2.6	14.0	0.056	-0.7	3.9


OpenRefine
A power tool for working with messy data.
New version! Download OpenRefine v3.3 now.
Create Project
Open Project
Import Project
Language Settings
« Start Over
Configure Parsing Options
Project name nba_results csv
Create Project »
NBA_name NBA_College NBA_Yrs NBA_G NBA_MP NBA_PTS NBA_AST NBA_FG NBA_THREE_P NBA_FT NBA_MP_PG NBA_PTS_PG
1. Andrew Wiggins Kansas 6 451 16136 8891 1056 0.441 0.332 0.733 35.8 19.7
2. Jabari Parker Duke 6 280 8193 4208 585 0.492 0.324 0.739 29.3 15.0
3. Joel Embiid Kansas 4 201 6154 4834 633 0.479 0.321 0.794 30.6 24.0
4. Aaron Gordon Arizona 6 395 11274 5029 925 0.448 0.320 0.701 28.5 12.7
5. Dante Exum 5 239 4429 1366 502 0.407 0.308 0.766 18.5 5.7
6. Marcus Smart Oklahoma State 6 390 11266 3806 1582 0.373 0.318 0.770 28.9 9.8
7. Julius Randle Kentucky 6 370 10777 5917 1039 0.492 0.290 0.722 29.1 16.0
8. Nik Stauskas Michigan 5 335 6662 2272 513 0.389 0.353 0.814 19.9 6.8
Parse data as
CSV / TSV / separator-based files
Line-based text files
Fixed-width field text files
PC-Axis text files
JSON files
MARC files
RDF/N3 files
Character encoding
Update Preview
Columns are separated by
commas (CSV)
tabs (TSV)
custom ,
Escape special characters with \
Ignore first 0 line(s) at beginning of file
Parse next 1 line(s) as column headers
Discard initial 0 row(s) of data
Load at most 0 row(s) of data
Parse cell text into numbers, dates, ...
Quotation marks are used to enclose cells containing column separators
Store blank rows
Store blank cells as nulls
Store file source (file names, URLs) in each row
Version 2.8 [TRUNK]
Preferences
Help
About

	Rk	Pk	Tm	Player	College	Yrs	G	MP	PTS	TRB	AST	FG%	3P%	FT%	MP	PTS	TRB	AST	WS	WS/48	BPM	V
0	1	1	CLE	Andrew Wiggins	Kansas	6	451	16136	8891	1961	1056	.441	.332	.733	35.8	19.7	4.3	2.3	15.0	.045	-1.9	
1	2	2	MIL	Jabari Parker	Duke	6	280	8193	4208	1614	585	.492	.324	.739	29.3	15.0	5.8	2.1	12.8	.075	-0.9	
2	3	3	PHI	Joel Embiid	Kansas	4	201	6154	4834	2310	633	.479	.321	.794	30.6	24.0	11.5	3.1	22.2	.173	4.4	
3	4	4	ORL	Aaron Gordon	Arizona	6	396	11310	5040	2526	934	.448	.319	.701	28.6	12.7	6.4	2.4	21.2	.090	-0.1	
4	5	5	UTA	Dante Exum		5	239	4429	1366	424	502	.407	.308	.766	18.5	5.7	1.8	2.1	3.1	.034	-3.3	
5	6	6	BOS	Marcus Smart	Oklahoma State	6	391	11298	3824	1391	1584	.373	.317	.772	28.9	9.8	3.6	4.1	20.1	.085	-0.3	
6	7	7	LAL	Julius Randle	Kentucky	6	371	10809	5949	3339	1039	.493	.292	.722	29.1	16.0	9.0	2.8	20.2	.090	-0.5	
7	8	8	SAC	Nik Stauskas	Michigan	5	335	6662	2272	688	513	.389	.353	.814	19.9	6.8	2.1	1.5	3.8	.028	-3.0	
8	9	9	CHH	Noah Vonleh	Indiana	6	331	5658	1655	1739	255	.460	.310	.692	17.1	5.0	5.3	0.8	9.3	.079	-2.3	
9	10	10	PHI	Elfrid Payton	LA-Lafayette	6	383	11236	4199	1682	2536	.452	.290	.624	29.3	11.0	4.4	6.6	14.7	.063	-0.5	
10	11	11	DEN	Doug McDermott	Creighton	6	403	8040	3302	882	339	.465	.413	.825	20.0	8.2	2.2	0.8	13.6	.081	-2.1	
11	12	12	ORL	Dario Šarić		4	295	7800	3566	1811	617	.436	.352	.837	26.4	12.1	6.1	2.1	13.4	.083	-0.9	
12	13	13	MIN	Zach LaVine	UCLA	6	353	10857	6240	1279	1275	.447	.375	.819	30.8	17.7	3.6	3.6	12.0	.053	-0.6	
13	14	14	PHO	T.J. Warren	NC State	6	320	9151	4856	1307	379	.504	.351	.775	28.6	15.2	4.1	1.2	18.1	.095	-0.4	
14	15	15	ATL	Adreian Payne	Michigan State	4	107	1403	429	315	66	.406	.254	.680	13.1	4.0	2.9	0.6	-0.6	-.019	-5.9	
15	16	16	CHI	Jusuf Nurkić		5	310	7103	3608	2480	589	.491	.071	.666	22.9	11.6	8.0	1.9	16.4	.111	0.2	
16	17	17	BOS	James Young	Kentucky	4	95	812	219	96	28	.367	.277	.563	8.5	2.3	1.0	0.3	0.8	.046	-3.4	

1. Remove incomplete entries (null/ empty cells)
2. Identify inconsistency (i.e. facet and clustering)
3. Fix incorrect data entries (e.g. wrong data types)
4. Rename, combine and split columns

Comparison Between OpenRefine and Pandas

	Open Refine	Pandas
Ease of Use	X	
Pre-built Features	X	X
Grouping	X	X
Speed		X
Control		X

Latest Serverless Development in Cloud Computing

Data Wrangling and Cleaning with Pandas

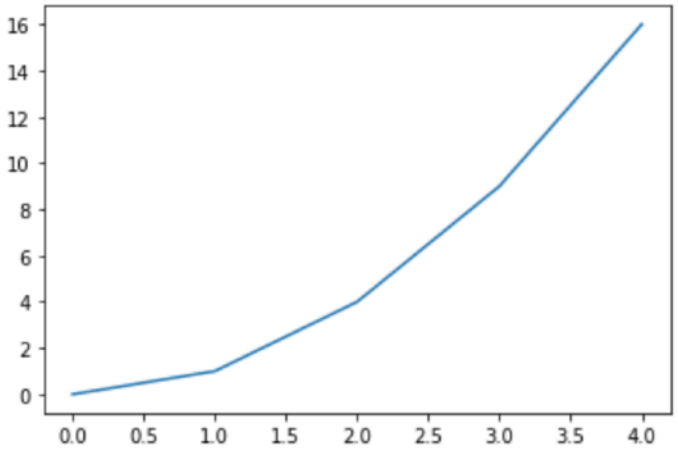
Descriptive Statistics & Data Visualization

Basic Descriptive Statistic Functions in Python

```
# Compare Numpy and Scipy statistical functions and my own functions
import numpy as np
from scipy import stats

def round(x):
    pos = str(x).find(".")+1
    val = str(x)[pos:pos+1]
    if (int(val) >= 5):
        return(str(x+1)[0:pos-1])
    else:
        return(str(x)[0:pos-1])

def average_calc(list):
    total = 0
    counter = 0
    for i in list:
        total = total + i
        # print(counter,i)
        counter = counter + 1
    # print ("total:",total)
    # print("no. of items:",counter)
    return(total/counter)

def median_calc(list):
    sorted_list = sorted(list)
    # print("sorted list:",sorted_list)
    result = 0
    position = int(round(len(sorted_list) / 2))
    if len(sorted_list) % 2 == 0:
```

Learning Matplotlib

```
:
1 from matplotlib import pyplot as plt
2 x_values = [0, 1, 2, 3, 4]
3 y_values = [0, 1, 4, 9, 16]
4 plt.plot(x_values, y_values)
5 plt.show()
```

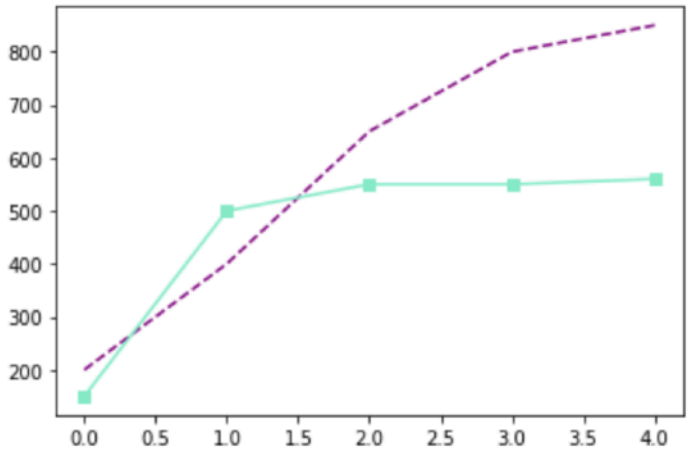



```
:
1 from matplotlib import pyplot as plt
2 days = range(7)
3 money_spent = [10,12,12,10,14,22,24]
4 plt.plot(days,money_spent)
5 plt.show()
```

Creating plot style by varying colors, linestyles, and markers.

Add clarity to the visualization by using title, label and legend.

```
from matplotlib import pyplot as plt
time = [0, 1, 2, 3, 4]
revenue = [200, 400, 650, 800, 850]
costs = [150, 500, 550, 550, 560]
plt.plot(time, revenue, color='purple',linestyle='--')
plt.plot(time, costs, color='#82edc9',marker='s')
plt.show()
```



THANK YOU FOR YOUR TIME!