



## Research & Development Team

# Node.js

[www.pnpsw.com](http://www.pnpsw.com)

[sommai.k@gmail.com](mailto:sommai.k@gmail.com)

**081-754-4663**

**Line Id : sommai.k**

Software

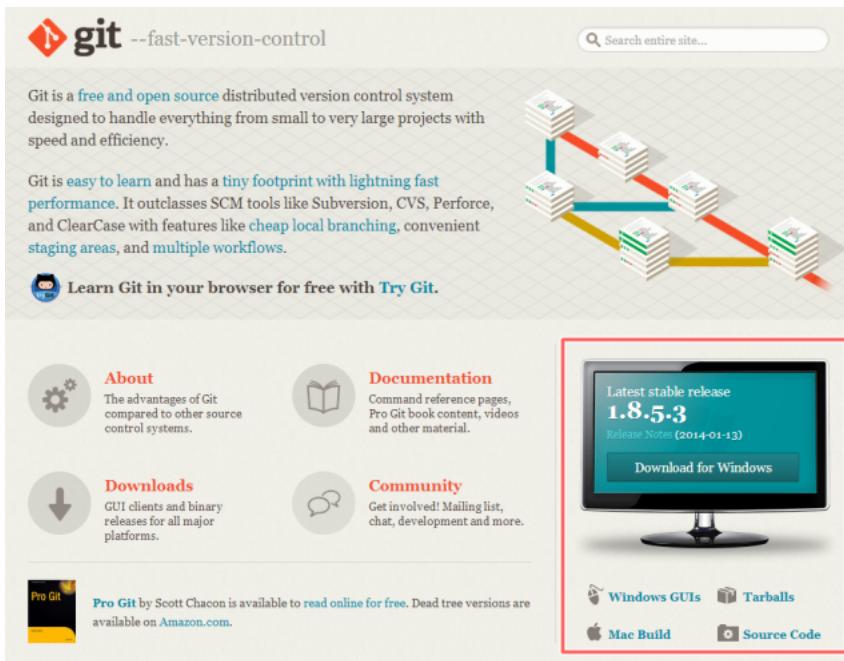
# INSTALLATION

การติดตั้ง

# **GIT FOR WINDOWS**

# การติดตั้ง GIT สำหรับ Windows

- เข้า website <https://git-scm.com/downloads>
- เลือก Download Git เพื่อติดตั้งบน Windows



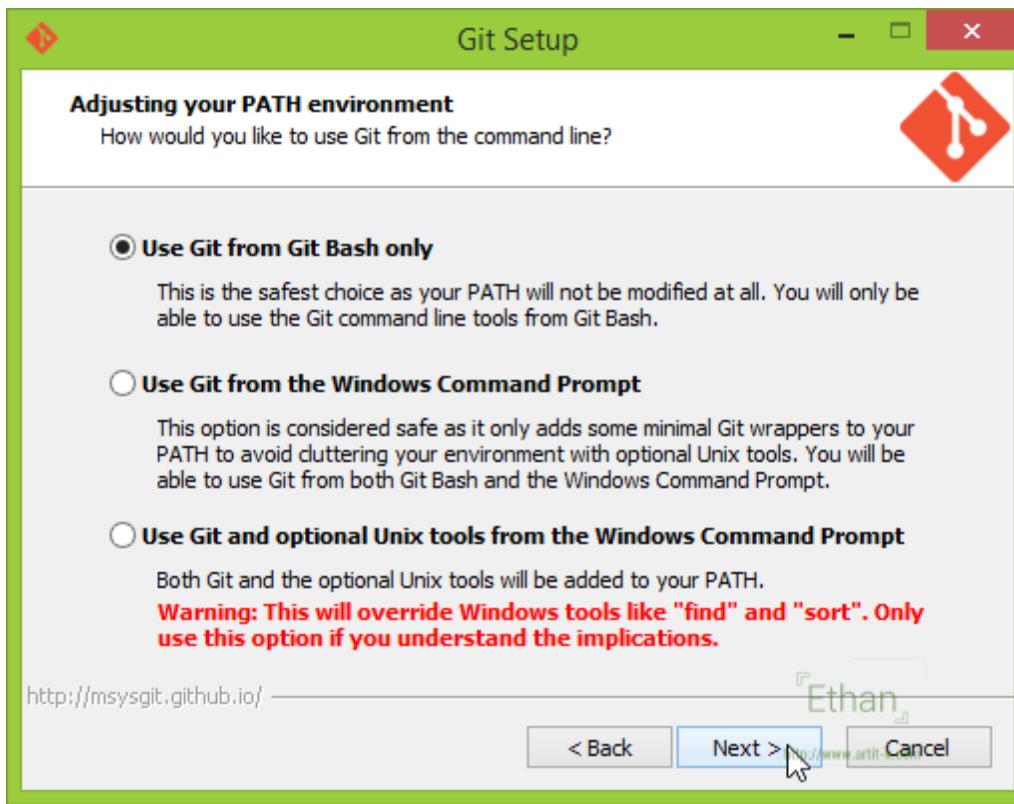
# การติดตั้ง GIT สำหรับ Windows #2

- ติดตั้งโดยใช้สิทธิ Administrator ในการติดตั้ง



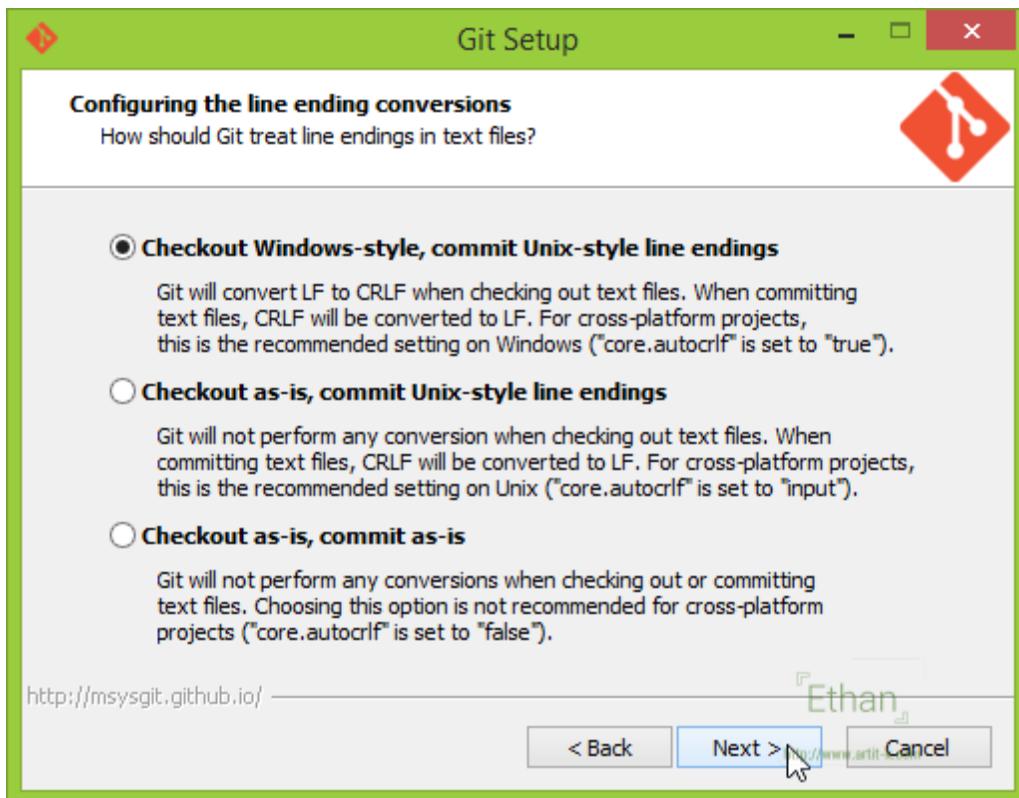
# การติดตั้ง GIT สำหรับ Windows #3

- เลือกติดตั้งแบบ **Use Git from the Windows Command Prompt**



# การติดตั้ง GIT สำหรับ Windows #4

- เลือกเป็น **Checkout Windows-style, commit Unix-style line endings**



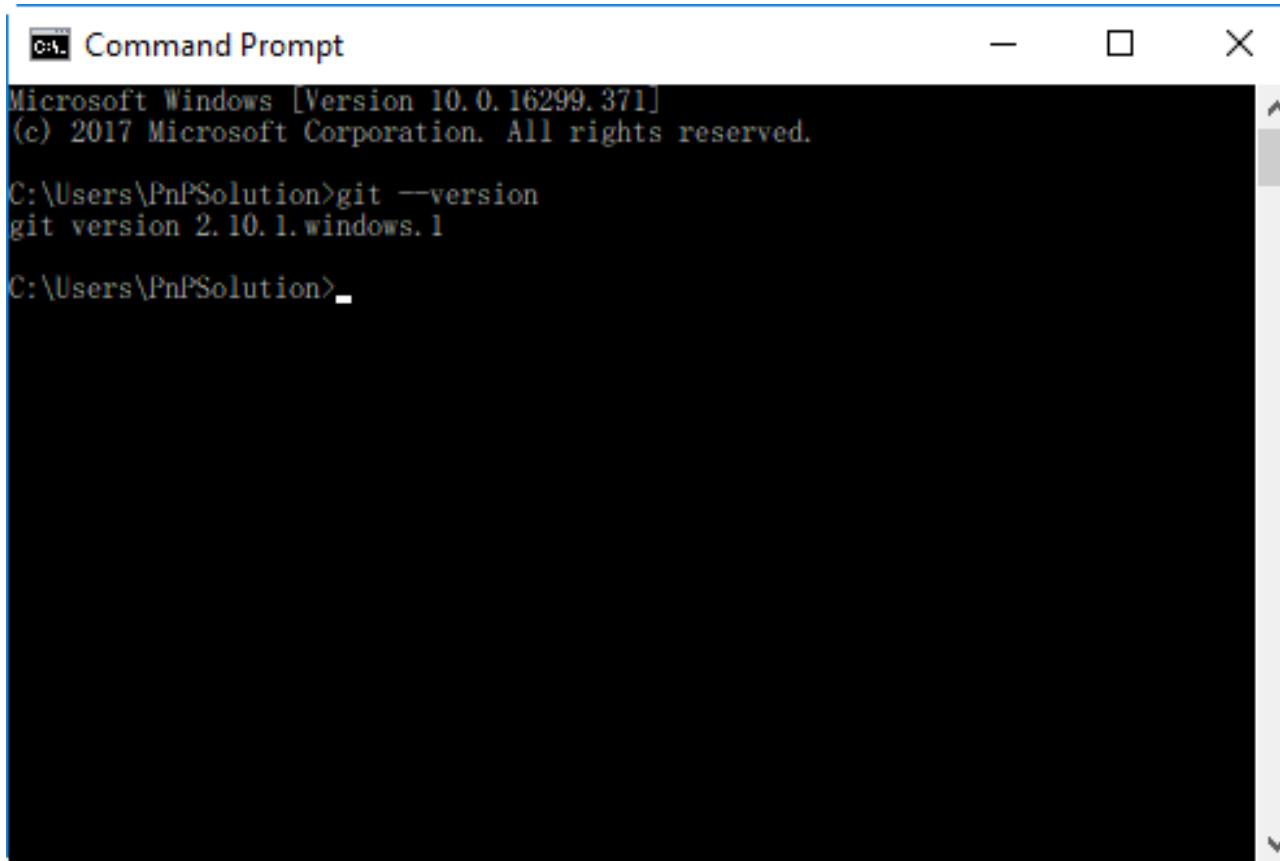
# การติดตั้ง GIT สำหรับ Windows #5

- กดปุ่ม next ไปจนถึงหน้าสุดท้าย



# ทดสอบหลังการติดตั้ง Git

- เปิดโปรแกรม cmd และพิมพ์คำสั่ง git --version



```
Command Prompt
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\PnPSSolution>git --version
git version 2.10.1.windows.1

C:\Users\PnPSSolution>
```

การติดตั้ง

# **DOCKER FOR WINDOWS**

# การติดตั้ง Docker สำหรับ Windows

- ความต้องการขั้นพื้นฐานสำหรับการติดตั้ง Docker for windows
  - Windows ต้องเป็น version 64bit เท่านั้น
  - Windows 7 or higher
  - Cpu ต้องมีความสามารถ Hyper-v สามารถเปิดได้ที่ BIOS

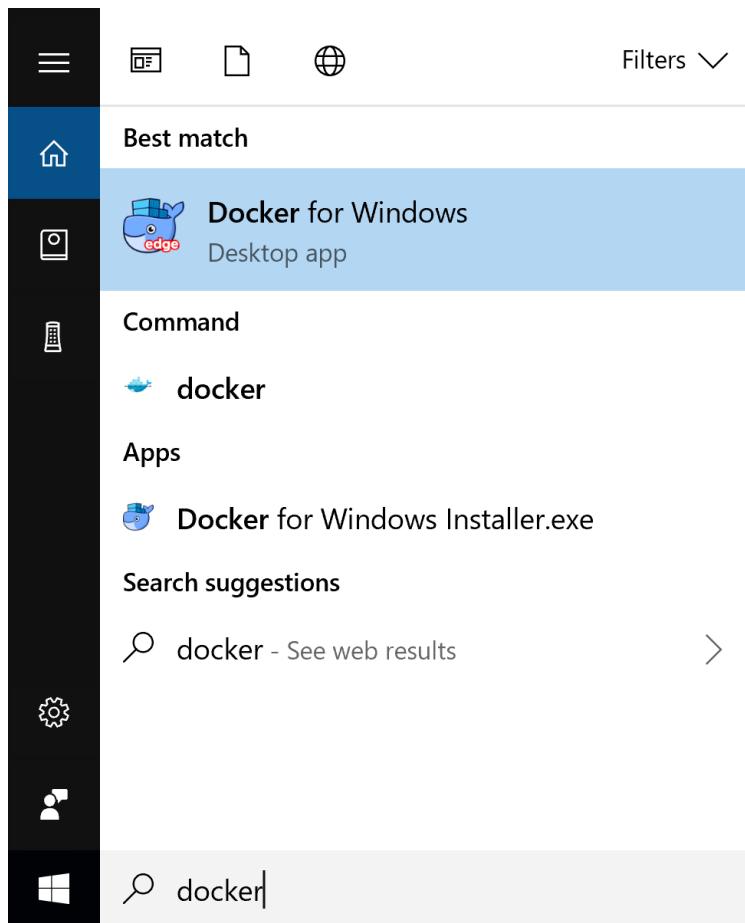


# การติดตั้ง Docker สำหรับ Windows #2

- เข้า link  
<https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe>
- กด next ไปเรื่อยๆ จนสิ้นสุดการติดตั้ง

# การติดตั้ง Docker สำหรับ Windows #3

- หลังจากติดตั้งเสร็จแล้วให้ทำการเปิด docker ดังนี้



# การติดตั้ง Docker สำหรับ Windows #4

- จะมี docker run อยู่ที่ taskbar ดังภาพ



- ทดสอบการติดตั้งโดยการเปิด cmd และพิมพ์คำสั่งดังนี้
- docker ps

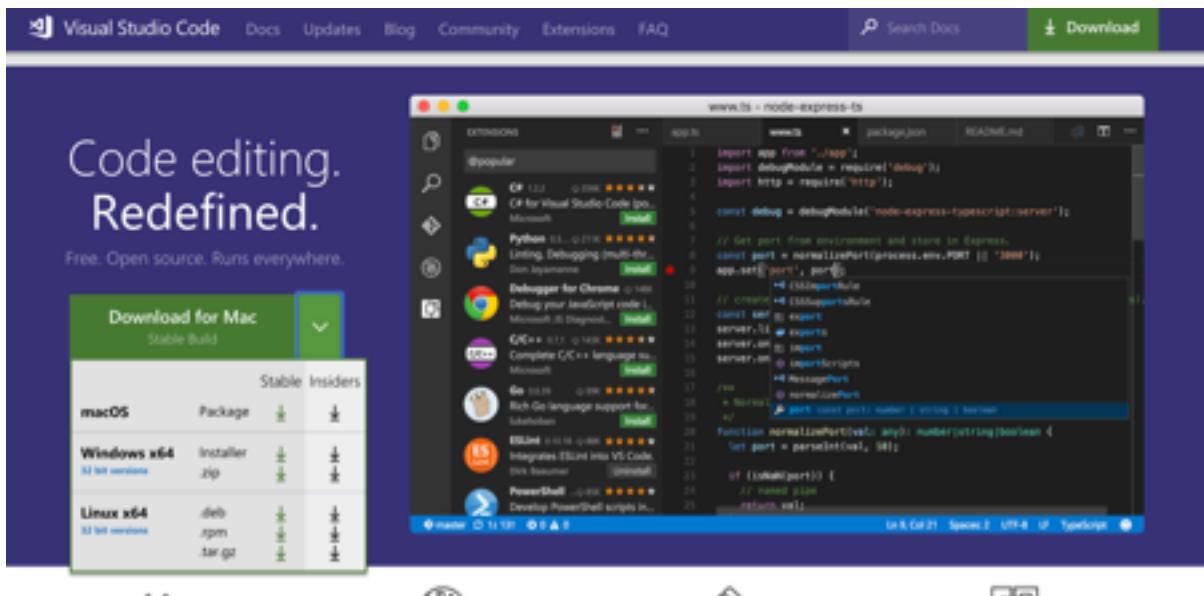
Sommais-MacBook-Pro:~ sommaik\$ docker ps	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

การติดตั้ง

# VISUAL STUDIO CODE

# การติดตั้ง VSC

- เข้าไปที่ website <https://code.visualstudio.com/>
- เลือก download สำหรับ windows (stable)



การติดตั้ง

# NODE.JS

# การติดตั้ง Node.js

- สำหรับ Windows <https://nodejs.org/en/download/current/>
- สำหรับ Linux / Mac <https://nodejs.org/en/download/package-manager/>

The screenshot shows the Node.js Downloads page. At the top, there are two tabs: "LTS Recommended For Most Users" (light green background) and "Current Latest Features" (dark green background). Below the tabs, there are three main download options:

- Windows Installer**: Represented by a Windows logo icon. Below it, there are links for "Windows Installer (.msi)" (node-v10.1.0-x64.msi) and "Windows Binary (.zip)".
- macOS Installer**: Represented by a macOS logo icon. Below it, there are links for "macOS Installer (.pkg)" (node-v10.1.0.pkg) and "macOS Binary (.tar.gz)".
- Source Code**: Represented by a cube icon. Below it, there is a link for "node-v10.1.0.tar.gz".

Below these options is a table showing available architectures for the "Current" version:

32-bit	64-bit	
32-bit	64-bit	
	64-bit	
	64-bit	
	64-bit	
ARMv6	ARMv7	ARMv8
node-v10.1.0.tar.gz		

การติดตั้ง

# TYPESCRIPT

# การติดตั้ง Typescript

เปิด terminal / cmd และพิมพ์คำสั่ง

- npm install -g typescript

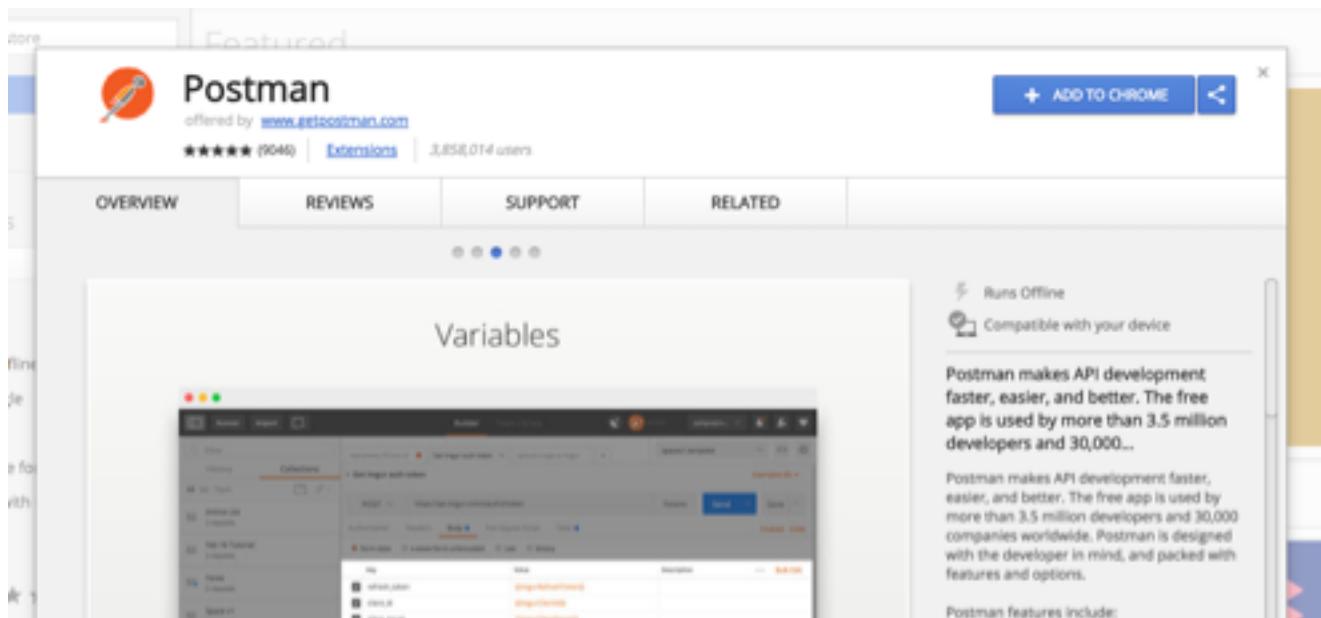
การติดตั้ง

# POSTMAN

# การติดตั้ง postman

- เข้า url

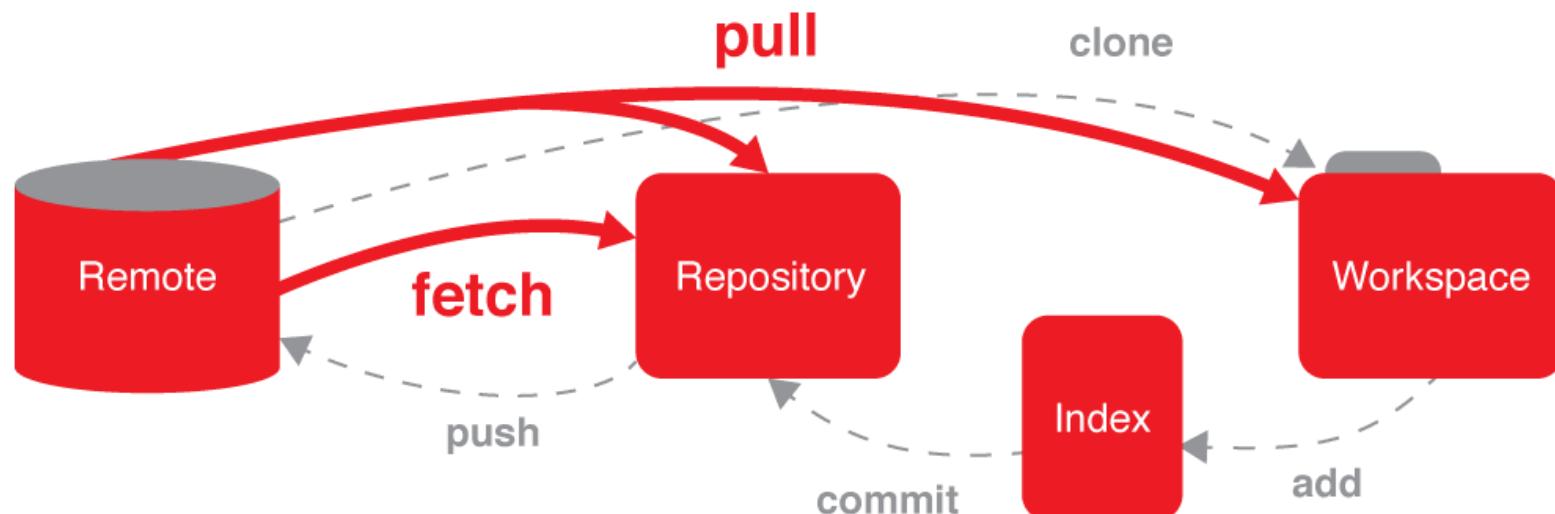
<https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbnccccdomop?hl=en>



Software version control with

**GIT**

# Overview



# Create a new Git Repository (Remote)

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner      Repository name

 Sommailk / XXX 

Great repository names are short and memorable. Need inspiration? How about [improved-adventure](#).

Description (optional)

 Public  
Anyone can see this repository. You choose who can commit.

 Private  
You choose who can see and commit to this repository.

Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None  Add a license: None 

**Create repository**

# CONFIGURE TOOLING

## Sets the name you want attached to your commit transactions

- git config --global user.name "[name]"
- git config --global user.name "XXX"

## Sets the email you want attached to your commit transactions

- git config --global user.email "[email address]"
- git config --global user.email "XXXX@hotmail.com"

## Enables helpful colorization of command line output

- git config --global color.ui auto

# CREATE REPOSITORIES

**Creates a new local repository with the specified name**

- git init [project-name]
- git init XXX

**Downloads a project and its entire version history**

- git clone [url]
- git clone <https://github.com/Sommaik/XXX.git>

# MAKE CHANGES

**Lists all new or modified files to be committed**

- git status

**Shows file differences not yet staged**

- git diff

**Snapshots the file in preparation for versioning**

- git add [file]
- git add readme.txt
- git add .

**Shows file differences between staging and the last file version**

- git diff --staged

# MAKE CHANGES

**Unstages the file, but preserve its contents**

- git reset [file]
- git reset readme.txt

**Records file snapshots permanently in version history**

- git commit -m "[descriptive message]"
- git commit -m "Initial Project"

# GROUP CHANGES

**Lists all local branches in the current repository**

- git branch

**Creates a new branch**

- git branch [branch-name]
- git branch XXX

**Switches to the specified branch and updates the working directory**

- git checkout [branch-name]
- git checkout XXX

# GROUP CHANGES

**Combines the specified branch's history into the current branch**

- git merge [branch]
- git merge XXX

**Deletes the specified branch**

- git branch -d [branch-name]
- git branch -d XXX

# REFACTOR FILENAMES

**Deletes the file from the working directory and stages the deletion**

- git rm [file]
- git rm XXX.txt

**Removes the file from version control but preserves the file locally**

- git rm --cached [file]
- git rm --cached XXX.txt

**Changes the file name and prepares it for commit**

- git mv [file-original] [file-renamed]
- git mv XXX.txt YYY.txt

# SUPPRESS TRACKING

A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

- `*.log`
- `build/`
- `temp-*`

**Lists all ignored files in this project**

- `git ls-files --other --ignored --exclude-standard`

# SAVE FRAGMENTS

**Temporarily stores all modified tracked files**

- git stash

**Restores the most recently stashed files**

- git stash pop

**Lists all stashed changesets**

- git stash list

**Discards the most recently stashed changeset**

- git stash drop

# REVIEW HISTORY

**Lists version history for the current branch**

- git log

**Lists version history for a file, including renames**

- git log --follow [file]
- git log --follow XXX.txt

**Shows content differences between two branches**

- git diff [first-branch]...[second-branch]

**Outputs metadata and content changes of the specified commit**

- git show [commit]
- git show XXX

# REDO COMMITS

**Undoes all commits after [commit], preserving changes locally**

- git reset [commit]
- git reset XXX

**Discards all history and changes back to the specified commit**

- git reset --hard [commit]
- git reset --hard XXX

# SYNCHRONIZE CHANGES

**Downloads all history from the repository bookmark**

- git fetch [bookmark]
- git fetch origin

**Combines bookmark's branch into current local branch**

- git merge [bookmark]/[branch]
- git merge origin/master2

# SYNCHRONIZE CHANGES

**Uploads all local branch commits to Git**

- git push [alias] [branch]
- git push origin master

**Downloads bookmark history and incorporates changes**

- git pull

Software development with  
**NODE.JS**

# ความรู้เบื้องต้นเกี่ยวกับ Node.js

ประวัติ ความเป็นมา สถานการณ์ที่萌芽แก่การใช้งาน

Node.js เป็นภาษาที่ทำงานอยู่ในผู้ Server ซึ่ง Syntax ที่ใช้ก็คือ JavaScript โดยจะออกแบบมาให้ทำงานแบบ Event-Driven ก็คือ จะทำงานเมื่อเกิดเหตุการณ์ตามที่กำหนดไว้ โดยสามารถกำหนดให้ทำงานแบบ Asynchronous (ทำงานในลำดับต่อไปโดยไม่ต้องรอให้งานก่อนหน้าเสร็จ) หรือ Synchronous (ทำงานต่อไป เมื่องานแรกเสร็จแล้ว) ก็ได้

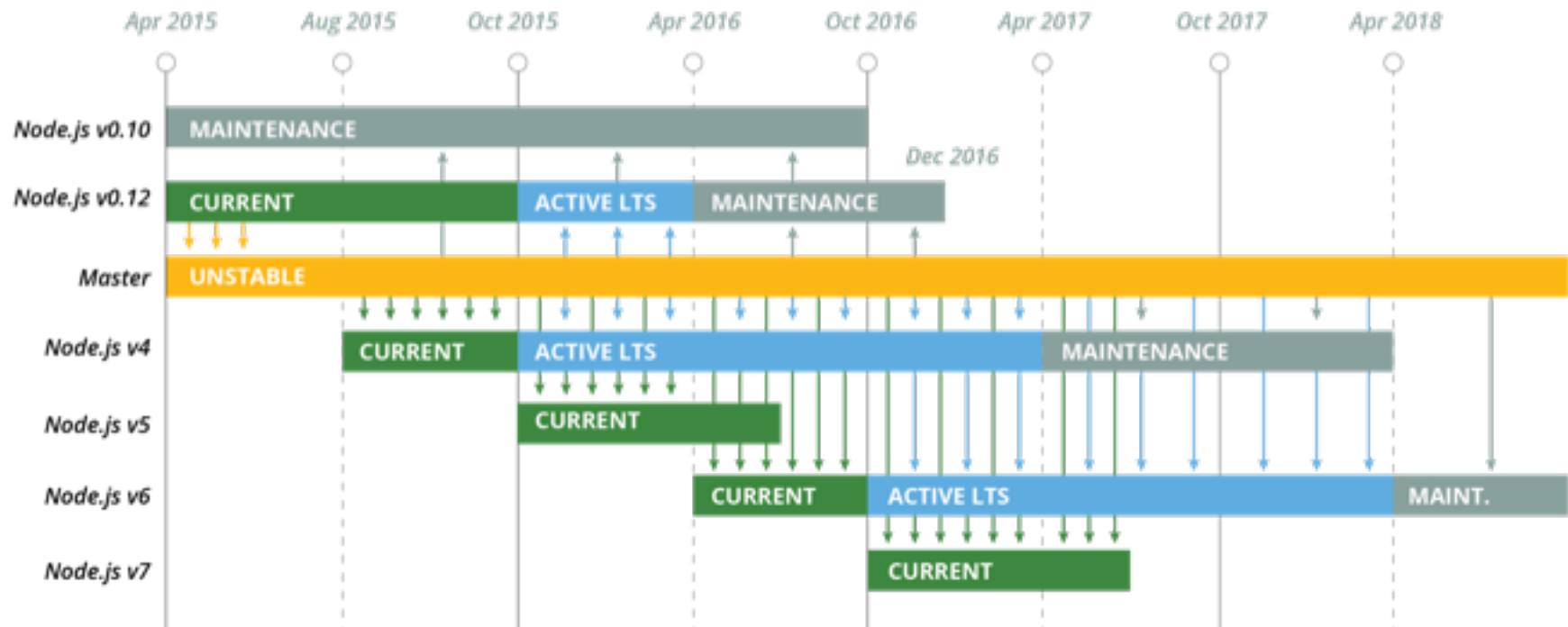
Node.js จะใช้ Compiler จาก Google JavaScript Engine V8 ซึ่งทำให้การประมวลผลรวดเร็วมาก คล้ายกับ parallel execution แต่ความจริงมันคือ single thread คือ แต่ละงานจะถูกนำมาเข้าคิวไว้ แล้วค่อยประมวลผลตามคิว

# ความรู้เบื้องต้นเกี่ยวกับ Node.js

node.js เหมาะสำหรับ งานที่ประมวลผลสั่ง server ซึ่งเป็นงานที่อาจจะต้อง interface หรือไม่ต้องก็ได้ เช่น การทำตัวเองเป็น http server ในการดึงหน้าเว็บมาแสดงผลให้กับ user หรือการเปิด socket เพื่อรับส่งข้อมูลกับระหว่าง server กับ user ที่อาจจะเอาไปทำเป็นห้อง chat , ทำเกม, ระบบที่ป้อนข้อมูลเพื่อคำนวณ เอาผลลัพธ์ เป็นต้น หรือ ตัวอย่างงานที่ไม่ต้อง interface เช่นเอาไปทำ spider crawler เว็บ หรือ การรับค่าจาก streaming เหล่านี้ไม่จำเป็นต้อง interface ต่างกันใช้ node.js ทำงานด้วยกันทั้งนั้น

# เลือก version ก่อนการติดตั้ง

## Node.js Long Term Support Release Schedule



COPYRIGHT © 2015 NODESOURCE, LICENSED UNDER CC-BY 4.0

# สร้าง node.js simple server

- สร้าง file ชื่อ app.js

```
var http = require('http');
http.createServer(function(req,res){
    res.writeHead(200,['Content-Type':'text/plain']);
    res.end('yes you can\n');
}).listen(8000);
console.log("server running port 8000");
```

- เปิด command line และพิมพ์ node app.js หรือ node app
- เปิด browser เข้า url http://localhost:8000/

# Node.js การใช้งาน Core APIs

- การเรียกใช้ module

```
const fs = require('fs');

fs.watchFile('message.txt', function(curr, prev){
    fs.readFile('message.txt', 'utf8', function(err, data){
        if (err) throw err;
        console.log(data);
    });
});
```

# การตั้งค่า npm สำหรับ project

- เปิด command ไปที่ folder ที่เก็บ project
- พิมพ์คำสั่ง npm init -y

# คำสั่งต่างๆ ของ npm

- ติดตั้ง module จาก package.json  
npm install
- ติดตั้ง module เฉพาะ module  
npm install module
- ติดตั้ง module และเก็บไว้ใช้ใน project  
npm install module --save
- ติดตั้ง module และเก็บไว้ใช้ก่อไป  
npm install module -g

Basic programming with  
**JAVASCRIPT**

# JavaScript

ภาษาจาวาสคริปต์ คือ ภาษาโปรแกรมคล้ายภาษา C ถูกใช้ร่วมกับภาษาอิเวชทีเอ็มแอลในการพัฒนาเว็บเพจ ประมวลผลในเครื่องของผู้ใช้ ช่วยให้การนำเสนอเป็นแบบโต้ตอบกับผู้ใช้ได้ในระดับหนึ่ง ภาษานี้มีชื่อเดิมว่า LiveScript ถูกพัฒนาโดย Netscape Navigator เพื่อช่วยให้เว็บเพจสามารถแสดงเนื้อหา ที่มีการเปลี่ยนแปลงได้ ตามเงื่อนไข หรือสภาพแวดล้อมที่แตกต่างกัน หรือโต้ตอบกับผู้ใช้ได้มากขึ้น ประเภทภาษา HTML ที่เป็นภาษาพื้นฐานของเว็บเพจ ทำได้เพียงแสดงข้อมูลแบบคงที่ (Static Display)

# การแทรก JavaScript ลง HTML

```
<html>
<body>

<script language="javascript">
.....
</script>

</body>
</html>
```

ÃƒŒ

```
<html>
<body>

<script type="text/javascript">
.....
</script>

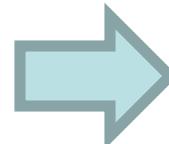
</body>
</html>
```

แสดงข้อความ **Hello World!** ที่หน้าเว็บ

```
<html>
<body>

<script type="text/javascript">
document.write("Hello World!");
</script>

</body>
</html>
```



**Hello World!**

**document.write เป็นคำสั่งที่ใช้เขียนผลลัพธ์บนหน้าเว็บ**

# การเรียกใช้ไฟล์ JavaScript ที่อยู่ภายนอก

- บันทึกไฟล์ JavaScript ให้มีนามสกุล .js
- เขียนส่วนของ tag script ให้ src ว้างอิงไปที่ไฟล์ ที่บันทึกไว้ ดังนี้

```
<html>
<head>
<script src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

# JavaScript Comments

- การใส่หมายเหตุบรรทัดเดียว ให้ใส่เครื่องหมาย // ไว้หน้าบรรทัดนั้น
- การใส่หมายเหตุหลายบรรทัด เริ่มต้นด้วย /\* และปิดท้ายด้วย \*/

```
<script type="text/javascript">
// This will write a header:
document.write("<h1>This is a header</h1>");
// This will write two paragraphs:
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

```
<script type="text/javascript">
/*
The code below will write
one header and two paragraphs
*/
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

# Variable

- ชื่อของตัวแปรใน JavaScript สามารถขึ้นต้นด้วยตัวอักษรใหญ่ (A-Z) ตัวอักษรเล็ก (a-z) และ เครื่องหมาย \_ ตามด้วย ตัวอักษร ตัวเลข หรือ เครื่องหมาย \_ ก็ได้
- ชื่อตัวแปรใน JavaScript จะเข้มงวดในการใช้ตัวอักษรใหญ่เล็ก ด้วย เช่น Sum SUM sum จะถือว่าไม่เป็นตัวแปรเดียวกัน

# Variable

ประกาศตัวแปร x และ carname

```
var x;  
var carname;
```

ประกาศตัวแปร x และ carname พร้อมกำหนดค่าเริ่มต้น

```
var x=5;  
var carname="Volvo";
```

ตัวแปรจะถูกประกาศอัตโนมัติ เมื่อมีการกำหนดค่าโดยไม่ต้องประกาศ var

```
x=5;  
carname="Volvo";
```

# Data Types

- Integer
- Number
- Boolean
- String
- Array
- Object
-

# Special Characters

Code	Outputs
\'	single quote
\\"	double quote
\&	ampersand
\\"	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

# Array

- Array ก็คือข้อมูลหลายๆตัวมาเรียงกันเป็นลำดับ เช่น

```
var employee = new Array(5)
employee[0] = "Bill"
employee[1] = "Bob"
employee[2] = "Ted"
employee[3] = "Alice"
employee[4] = "Sue"
```

ในการใช้จริงเราไม่จำเป็นต้องกำหนด length ก็ได้โดย length จะ  
ยึดหยุ่นได้ตามตัวแปรลำดับสุดท้ายT

```
var employee = new Array("Bill", "Bob", "Ted", "Alice",
"Sue");
```

# Arithmetic Operator

กำหนดให้  $y = 5$

Operator	Description	Example	Result
+	Addition	$x=y+2$	$x=7$
-	Subtraction	$x=y-2$	$x=3$
*	Multiplication	$x=y*2$	$x=10$
/	Division	$x=y/2$	$x=2.5$
%	Modulus (division remainder)	$x=y \% 2$	$x=1$
++	Increment	$x=++y$	$x=6$
--	Decrement	$x=--y$	$x=4$

# Logical Operator

กำหนดให้  $x = 6$  และ  $y=3$

Operator	Description	Example
<code>&amp;&amp;</code>	and	$(x < 10 \&\& y > 1)$ is true
<code>  </code>	or	$(x==5    y==5)$ is false
<code>!</code>	not	$!(x==y)$ is true

# Comparison Operator

กำหนดให้  $x = 5$

Operator	Description	Example
<code>==</code>	is equal to	<code>x==8</code> is false
<code>====</code>	is exactly equal to (value and type)	<code>x====5</code> is true <code>x===="5"</code> is false
<code>!=</code>	is not equal	<code>x!=8</code> is true
<code>&gt;</code>	is greater than	<code>x&gt;8</code> is false
<code>&lt;</code>	is less than	<code>x&lt;8</code> is true
<code>&gt;=</code>	is greater than or equal to	<code>x&gt;=8</code> is false
<code>&lt;=</code>	is less than or equal to	<code>x&lt;=8</code> is true

# Assignment Operators

กำหนดให้  $x = 10$  และ  $y = 5$

Operator	Example	Same As	Result
=	$x=y$		$x=5$
$+=$	$x+=y$	$x=x+y$	$x=15$
$-=$	$x-=y$	$x=x-y$	$x=5$
$*=$	$x*=y$	$x=x*y$	$x=50$
$/=$	$x/=y$	$x=x/y$	$x=2$
$%=$	$x\%=y$	$x=x \% y$	$x=0$

# if Statement

```
if (condition)
{
code to be executed if condition is true
}
```

รูปแบบ

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
{
document.write("<b>Good morning</b>");
}
</script>
```

ตัวอย่าง

# if...else if...else Statement

```
if (condition1)
{
    code to be executed if condition1 is true
}
else if (condition2)
{
    code to be executed if condition2 is true
}
else
{
    code to be executed if condition1 and
    condition2 are not true
}
```

၅၂။

# switch Statement

```
switch(n)
{
    case 1:
        execute code block 1
        break;
    case 2:
        execute code block 2
        break;
    default:
        code to be executed if n is
        different from case 1 and 2
}
```

ຈຸບແນບ

# for Loop

```
for (ประกาศตัวแปรใหม่พร้อมกำหนดค่าเริ่มต้น; เงื่อนไขการหยุด; เพิ่มค่าให้ตัวแปร) {  
    คำสั่งต่างๆ ที่จะให้ทำซ้ำ  
}
```

```
<html>  
<body>  
<script type="text/javascript">  
var i=0;  
for (i=0;i<=10;i++)  
{  
document.write("The number is " + i);  
document.write("<br />");  
}  
</script>  
</body>  
</html>
```

# while loop

```
while (condition) {  
    ชุดคำสั่ง  
}
```

```
<html>  
<body>  
<script type="text/javascript">  
var i=0;  
while (i<=10)  
{  
document.write("The number is " + i);  
document.write("<br />");  
i=i+1;  
}  
</script>  
</body>  
</html>
```

# Do..While

Do while จะเป็นการวน loop ชนิดที่กำหนดกว่า เงื่อนไขจะเป็นจริง concept จะคล้ายๆกับ while แต่อย่างสับสนนะ while จะทำงานจนเงื่อนไขเป็นเท็จ แต่ do..while จะทำงานเงื่อนไขเป็นจริง

```
do {  
    ชุดคำสั่ง  
} while (condition);
```

# Break Statement

จะทำหน้าที่หยุดการทำงานของ loop แบบทันทีทันใด ดังนั้นเมื่อ break ทำงาน loop จะหยุดการทำงานอย่างกะทันหัน

```
a = new Array(5,4,3,2,1)
sum = 0
for (i=0;i<a.length;i++)
{
    if (i==3) {break;}
    sum += a[i];
}
document.write(sum);
```

# Continue Statement

Continue ก็ทำงานคล้ายๆกับ Break คือเข้ามาขัดขวางการทำงานแต่ก็มีข้อแตกต่างตรงที่ Break ใช้หยุด loop ทั้งหมด แต่ Continue ใช้เพื่อหยุดแค่ loop ปัจจุบันเพียง loop เดียว หรือจะเรียกการทำงานของมันว่า Skip ก็ได้

```
i = 1 ;
sum = 0;
while (i<10)
{
    i*=2;
    if (i==4) {continue;}
    sum += i+1;
}
```

# Function

function คือโปรแกรมย่ออย่างที่ทำงานอย่างใดอย่างหนึ่ง ถูกสร้างขึ้นแยกออกจากโปรแกรมหลักเพื่อให้สามารถเรียกใช้ได้อย่างสะดวก

การสร้าง function ของ JavaScript มีรูปแบบดังนี้

```
function ชื่อฟังก์ชัน(พารามิเตอร์1,พารามิเตอร์2,...){
```

```
    คำสั่งต่าง ๆ
```

```
}
```

# Return Statement

เป็น Function ที่คืนค่าได้ เรามองว่ามันเป็นตัวแปรตัวหนึ่ง ที่เก็บค่าๆหนึ่งอยู่ได้เลย

```
function ชื่อฟังก์ชัน(พารามิเตอร์1,พารามิเตอร์2,...){  
    คำสั่งต่าง ๆ  
    return ค่าที่ส่งออกไป  
}
```

# Event គីឡូវ៉ាទិរ

Event ก็คือ Action ต่างๆที่เกิดขึ้นกับส่วนต่างๆในเว็บเพจ เช่น เมื่อเราเอา mouse ไปหัวตัว link ก็จะเกิด event onmouseover ที่ตัว link พ่อเรา mouse ออก ก็จะเกิด event onmouseout พ่อเรา click ก็จะเกิด event onclick เป็นต้น การทำงานของ event ก็จะมีอยู่ 2 ขั้นตอน คือ

1. ตรวจสอบการเกิด event กี่เรากำหนดไว้
  2. เมื่อเกิด event ขึ้น ก็จะไปเรียก function หรือคำสั่งต่างๆมาทำงาน

# Event

Event	ความหมาย
onAbort	เกิดเมื่อผู้ใช้ยกเลิกการ load ภาพ
onBlur	เกิดเมื่ออ้อบเจกต์นั้นถูกย้าย focus ออกไป
onChange	เกิดเมื่อผู้ใช้เปลี่ยนแปลงค่าในฟอร์มรับข้อมูล
onClick	เกิดเมื่ออ้อบเจกต์นั้นถูก click
onError	เกิดเมื่อการ load เอกสารหรือภาพเกิดข้อผิดพลาด
onFocus	เกิดเมื่ออ้อบเจกต์นั้นถูก focus
onLoad	เกิดเมื่อโหลดเอกสารเสร็จ
onMouseover	เกิดเมื่ออ้อบเจกต์นั้นถูกเลื่อน mouse pointer ไปทับ
onmouseout	เกิดเมื่ออ้อบเจกต์นั้นถูกเลื่อน mouse pointer ที่หันอยู่ ออกไป
onSelect	เกิดเมื่อผู้ใช้เลือกข้อความ(ใช้ mouse ลาก)ในช่องรับ ข้อความ
onSubmit	เกิดเมื่อผู้ใช้ submit แบบฟอร์ม
onUnload	เกิดเมื่อผู้ใช้ออกจากเว็บเพจ

# วิธีการใช้ Event

เราจะใส่ event ลงไปใน tag ของ html เลย เช่น เวลาจะกำตัว link เราใช้ tag <A> ถ้าจะทำให้มันมีข้อความ Alert ขึ้นเวลาเรา mouse ไป over เขียนโค้ดได้ดังนี้

```
<a href=""  
onmouseover="window.alert('Onmouseover ทำงาน')">  
ทดสอบ onmouseover  
</a>
```

# Try...Catch Statement

```
try
{
    //Run some code here
}
catch(err)
{
    //Handle errors here
}
```

Design application with

# **MODULE PATTERN**

# Module Pattern

- PATTERN 1: DEFINE A GLOBAL
- PATTERN 2: EXPORT AN ANONYMOUS FUNCTION
- PATTERN 3: EXPORT A NAMED FUNCTION
- PATTERN 4: EXPORT AN ANONYMOUS OBJECT
- PATTERN 5: EXPORT A NAMED OBJECT
- PATTERN 6: EXPORT AN ANONYMOUS PROTOTYPE
- PATTERN 7: EXPORT A NAMED PROTOTYPE

# PATTERN 1: DEFINE A GLOBAL

File : foo.js

```
foo = function () {  
    console.log('foo!');  
}
```

File : app.js

```
require('./foo.js');  
foo();
```

# PATTERN 2: EXPORT AN ANONYMOUS FUNCTION

File : bar.js

```
module.exports = function () {  
    console.log('bar!');  
}
```

File : app.js

```
var bar = require('./bar.js');  
bar();
```

# PATTERN 3: EXPORT A NAMED FUNCTION

File : fiz.js

```
exports.fiz = function () {  
    console.log('fiz!');  
}
```

File : app.js

```
var fiz = require('./fiz.js').fiz;  
fiz();
```

# PATTERN 4: EXPORT AN ANONYMOUS OBJECT

File : buz.js

```
var Buzz = function () {};
Buzz.prototype.log = function () { console.log('buzz!');

};

module.exports = new Buzz();
```

File : app.js

```
var buzz = require('./buz.js');
buzz.log();
```

# PATTERN 5: EXPORT A NAMED OBJECT

File : baz.js

```
var Baz = function () {};
Baz.prototype.log = function () { console.log('baz!');
};
exports.Baz = new Baz();
```

File : app.js

```
var baz = require('./baz.js').Baz;
baz.log();
```

# PATTERN 6: EXPORT AN ANONYMOUS PROTOTYPE

File : doo.js

```
var Doo = function () {};
Doo.prototype.log = function () { console.log('doo!'); }
module.exports = Doo;
```

File : app.js

```
var Doo = require('./doo.js');
var doo = new Doo();
doo.log();
```

# PATTERN 7: EXPORT A NAMED PROTOTYPE

File : qux.js

```
var Qux = function () {};
Qux.prototype.log = function () { console.log('baz!'); };
exports.Qux = Qux;
```

File : app.js

```
var Qux = require('./qux.js').Qux;
var qux = new Qux();
qux.log();
```

# การจัดการเหตุการณ์ต่างๆ (Event Handling)

```
const http = require('http');
const server = http.createServer().listen(8000);

server.on('request', function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('yes you can\n');

});

console.log("server running port 8000");
```

Node.js Programming

# CONTROL FLOW

# Synchronous Control Flow

File : sync.js

```
const fs = require('fs');
data = fs.readFileSync('abc.txt', 'utf8');
console.log(data);
console.log("something else");
```

# Asynchronous Control Flow

```
var fs = require("fs");
fs.readFile('abc.txt', 'utf8', function(err,data){
  if(!err) {
    console.log(data);
  }
});
console.log("something else");
```

Asynchronous Control Flow with  
**ASYNC MODULE**

# ติดตั้ง async module

- เปิด terminal / cmd เข้าไปที่ project ที่ต้องการติดตั้ง
- รันคำสั่ง ดังนี้
- npm install async --save

# async pattern

- Waterfall การทำงานจะเป็นลำดับขั้นทำขั้นแรกเสร็จก่อนแล้วถึงจะทำขั้นต่อไป ถ้ามี error จะหยุดทำงาน ผลลัพธ์ที่ได้จากการรันจะส่งต่อ ๆ กันไปในแต่ละ function
- Series การทำงานจะเป็นลำดับขั้นทำขั้นแรกเสร็จก่อนแล้วถึงจะทำขั้นต่อไป ถ้ามี error จะหยุดทำงาน ผลลัพธ์ที่ได้จะรวมเป็น array
- Parallel การทำงานจะเริ่มพร้อมกันทุก function ผลลัพธ์ที่ได้จะรวมเป็น array

# Waterfall

```
async.waterfall([
  function(callback) {
    callback(null, 'one', 'two');
  },
  function(arg1, arg2, callback) {
    // arg1 now equals 'one' and arg2 now equals 'two'
    callback(null, 'three');
  },
  function(arg1, callback) {
    // arg1 now equals 'three'
    callback(null, 'done');
  }
], function (err, result) {
  // result now equals 'done'
});
```

# Series

```
async.series([
  function(callback) {
    // do some stuff ...
    callback(null, 'one');
  },
  function(callback) {
    // do some more stuff ...
    callback(null, 'two');
  }
],
// optional callback
function(err, results) {
  // results is now equal to ['one', 'two']
});
```

# Parallel

```
async.parallel([
  function(callback) {
    setTimeout(function() {
      console.log('one');
      callback(null, 'one');
    }, 200);
  },
  function(callback) {
    setTimeout(function() {
      console.log('two');
      callback(null, 'two');
    }, 100);
  }
],
// optional callback
function(err, results) {
  console.dir(results);
});
```

Develop rest api with

# **EXPRESS FRAMEWORK**

# Express framework

- การติดตั้ง และการเตรียมการ
- npm install express --save
- File : app.js

```
var express = require('express');
var app = express();
var server = require('http').Server(app);
server.listen(8000);
app.use(express.static('web'));
console.log("server running port 8000");
```

# การใช้งาน URL Routing

- Route methods

```
app.get('/', function (req, res) {  
    res.render('index', {  
        title : "hello",  
        message : "my world"  
    });  
});
```

```
app.post('/', function (req, res) {  
    res.send('POST request to the homepage');  
});
```

# การใช้งาน URL Routing

- Route paths

/ab?cd = match acd and abcd.

/ab+cd = match abcd, abbcd, abbbcd, and so on.

/ab\*cd = match abcd, abxcd, abRANDOMcd, ab123cd, and so on.

/ab(cd)?e = match /abe and /abcde.

/a/ = match anything with an “a” in the route name.

./.\*fly\$/ = match butterfly and dragonfly, but not butterflyman, dragonfly man, and so on.

# การใช้งาน URL Routing

- Route parameters

Route path: /users/:userId/books/:bookId

Request URL: http://localhost:3000/users/34/books/8989

req.params: { "userId": "34", "bookId": "8989" }

# การใช้งาน URL Routing

- Route handlers

```
var cb0 = function (req, res, next) {
  console.log('CB0');
  next();
}

var cb1 = function (req, res, next) {
  console.log('CB1');
  next();
}

var cb2 = function (req, res) {
  res.send('Hello from C!');
}

app.get('/example/c', [cb0, cb1, cb2]);
```

# การใช้งาน URL Routing

```
app.route()
```

```
app.route('/book')
  .get(function(req, res) {
    res.send('Get a random book');
  })
  .post(function(req, res) {
    res.send('Add a book');
  })
  .put(function(req, res) {
    res.send('Update the book');
  });
});
```

# การใช้งาน Simple Route Middleware

```
var express = require('express');
var router = express.Router();

// middleware that is specific to this router
router.use(function timeLog(req, res, next) {
  console.log('Time: ', Date.now());
  next();
});

// define the home page route
router.get('/', function(req, res) {
  res.send('Birds home page');
});

// define the about route
router.get('/about', function(req, res) {
  res.send('About birds');
});

module.exports = router;
```

```
var birds = require('./birds');

app.use('/birds', birds);
```

# การใช้งานฐานข้อมูลแบบ MySQL

- ติดตั้ง
- <https://dev.mysql.com/downloads/installer/>
- Client GUI
- <https://software.dell.com/products/toad-for-mysql/>
- Node.js
- `npm install mysql --save`

# Connect to mysql from node.js

```
var mysql = require('mysql');
```

```
var pool = mysql.createPool({  
    host : "localhost",  
    user : "train",  
    password : "train",  
    database : "train",  
    connectionLimit : 10,  
    multipleStatements :true  
});
```

# Run sql statement

```
pool.getConnection(function(err,connection){  
    if (err) {  
  
    }else{  
        connection.query(sql, function(err,rows){  
            connection.release();  
            if(!err) {  
  
                }else{  
  
            }  
        });  
    }  
});
```

# หลักการพัฒนา Real-time Web application ด้วย web socket

- ติดตั้ง
- npm install socket.io --save
- Server

```
var io = require('socket.io')(server);
io.on('connection', function (socket) {
  socket.on('hello', function (data) {
    socket.emit('news', "");
  });
});
```

# หลักการพัฒนา Real-time Web application ด้วย web socket

- Client

```
<script src="/socket.io/socket.io.js"></script>
<script>
    var socket = io.connect('http://localhost');
    socket.on('news', function (data) {
        console.log(data);
    });
    function send(){
        socket.emit('hello', "");
    }
</script>
```

# Send Email

- install

```
npm install nodemailer --save
```

- Example

```
var mailer = require("nodemailer");
var smtpTransport = mailer.createTransport(config.smtp);
var mail = {
  to : 'to@email.com',
  subject: `hello world`,
  html: `this is a book`
}
smtpTransport.sendMail(mail, function(error, response){
  smtpTransport.close();
  if(error){
  }else{
  }
});
```

# Generate Excel

- install

```
npm install excel4node --save
```

- Example

```
var wb = new xl.Workbook();
var ws = wb.addWorksheet('Sheet 1');
ws.cell(1, 1).string("Reconcile report")
    .style({
        font: {
            bold : true
        }
    });
wb.write(excelFileName, cb);
```

# PM2 (Process Management)

- Install
- npm install pm2 -g
  
- Start
- pm2 start app.js --name my-api
  
- List Process
- pm2 list
  
- Show Process Description
- pm2 describe 0

# PM2 (Command)

- Monitor

pm2 monit

- Show Logs

pm2 logs

- Delete Log

pm2 flush

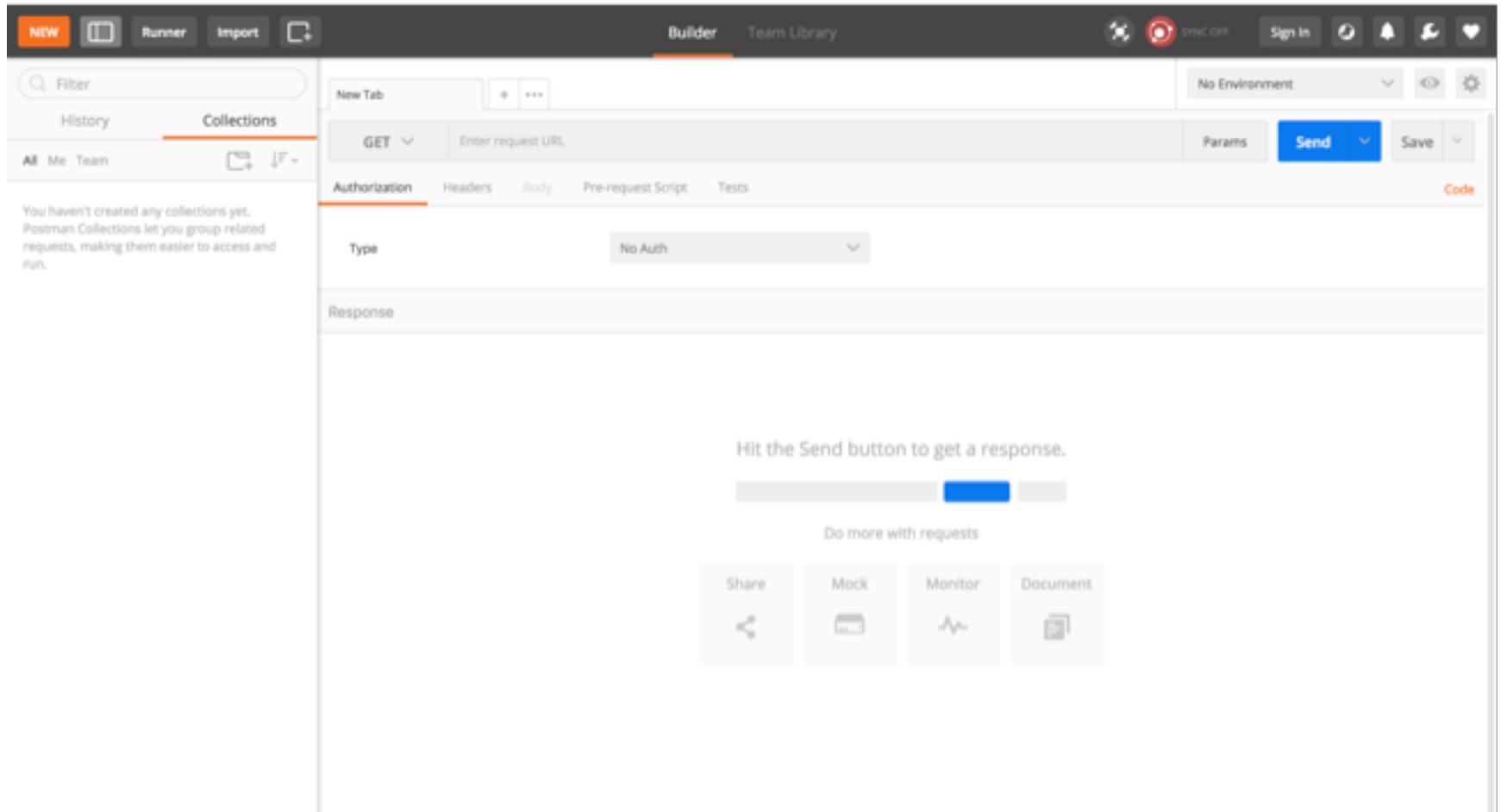
- Stop / Restart Process

pm2 stop all / process id

pm2 restart all / process id

Test rest api with  
**POSTMAN**

# POSTMAN MAIN SCREEN



The screenshot shows the Postman application interface. At the top, there is a navigation bar with buttons for NEW, Runner, Import, and a search/filter field. To the right of the search field are tabs for Builder and Team Library, along with icons for Sync (OFF), Sign In, and notifications.

The main workspace is titled "New Tab". It contains a request configuration area with a "GET" method selected, an "Enter request URL" input field, and tabs for Authorization, Headers, Body, Pre-request Script, and Tests. The Authorization tab is currently active, showing a dropdown menu set to "No Auth".

Below the request configuration is a large, empty "Response" area with the placeholder text "Hit the Send button to get a response." A blue "Send" button is located at the bottom of this area. Below the response area is a section titled "Do more with requests" containing four buttons: Share (with a share icon), Mock (with a mock icon), Monitor (with a monitor icon), and Document (with a document icon).

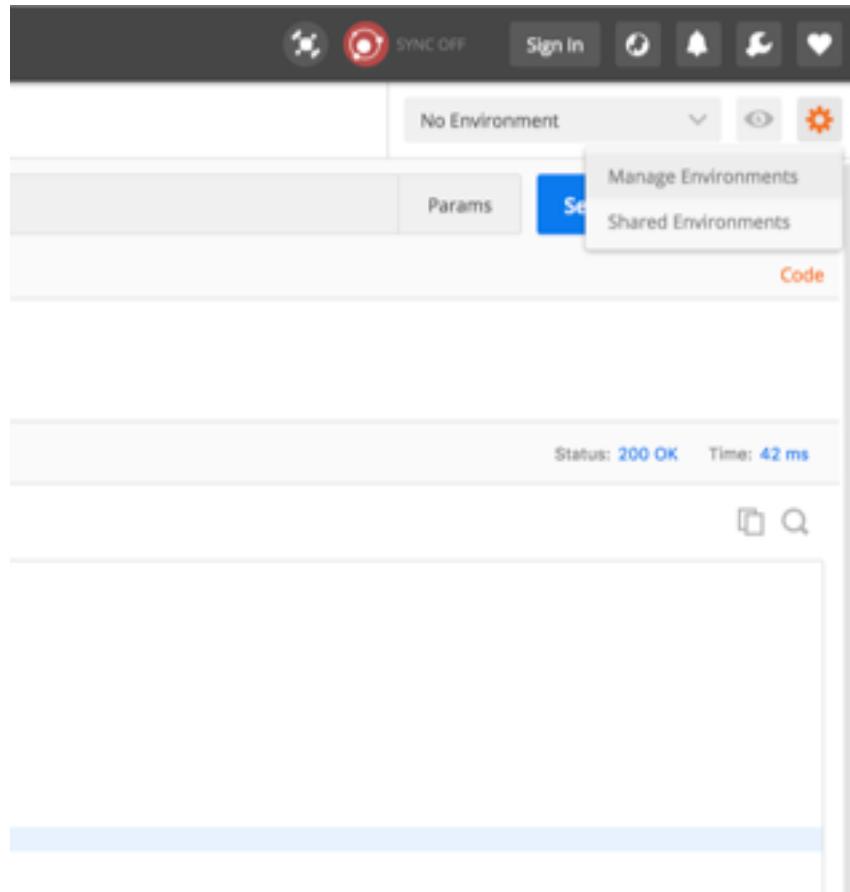
On the left side of the workspace, there is a sidebar with sections for History, Collections (which is currently selected), All, Me, and Team. A message indicates that no collections have been created yet, stating: "You haven't created any collections yet. Postman Collections let you group related requests, making them easier to access and run."

# Create new collection

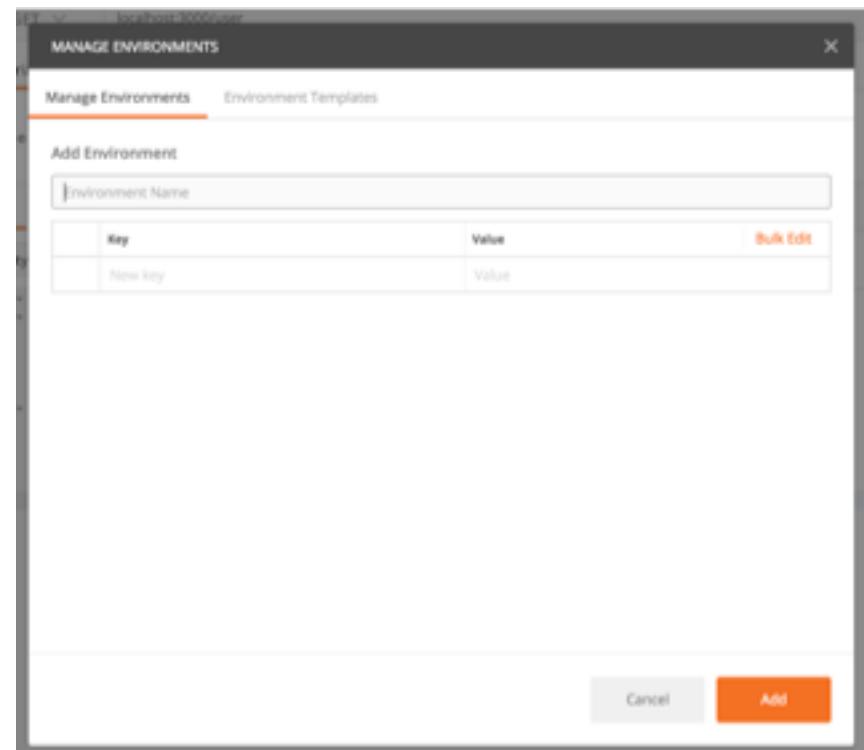
The screenshot shows the Postman application interface. At the top, there is a navigation bar with buttons for 'NEW' (orange), 'Runner' (grey), 'Import' (grey), and a 'Collection' icon (grey). Below the navigation bar is a search bar labeled 'Filter'. Underneath the search bar, there are tabs for 'History' and 'Collections'. The 'Collections' tab is currently selected, indicated by an orange underline. Below the tabs, there are buttons for 'All', 'Me', and 'Team'. A large blue arrow points from the text 'You haven't created any collections yet.' towards the 'Collection' icon in the navigation bar. To the right of the 'Collection' icon is a dropdown menu with options like 'Auth', 'T', 'Body', 'Pr', and 'P'. Below the tabs, there is a message: 'You haven't created any collections yet. Postman Collections let you group related requests, making them easier to access and run.'



# Manage Environments



The screenshot shows a web-based interface for managing environments. At the top, there's a navigation bar with icons for sync status (SYNC OFF), sign in, and other system functions. Below the bar, a dropdown menu shows "No Environment". A sidebar on the left has tabs for "Params" and "Code", with "Code" currently selected. A dropdown menu in the center says "Manage Environments" and "Shared Environments". At the bottom, it displays "Status: 200 OK" and "Time: 42 ms".

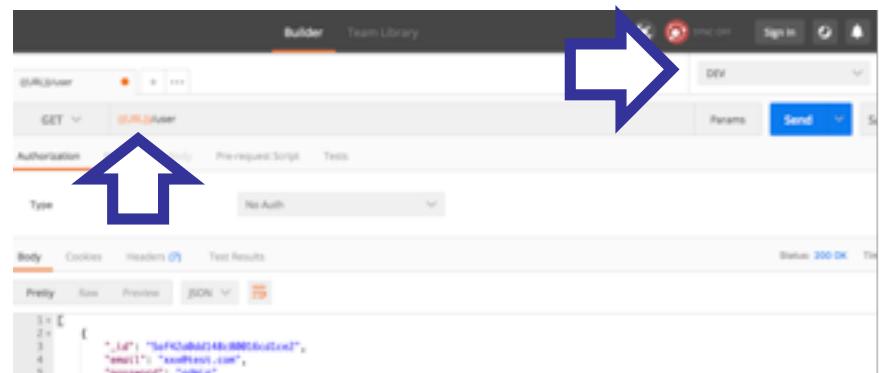
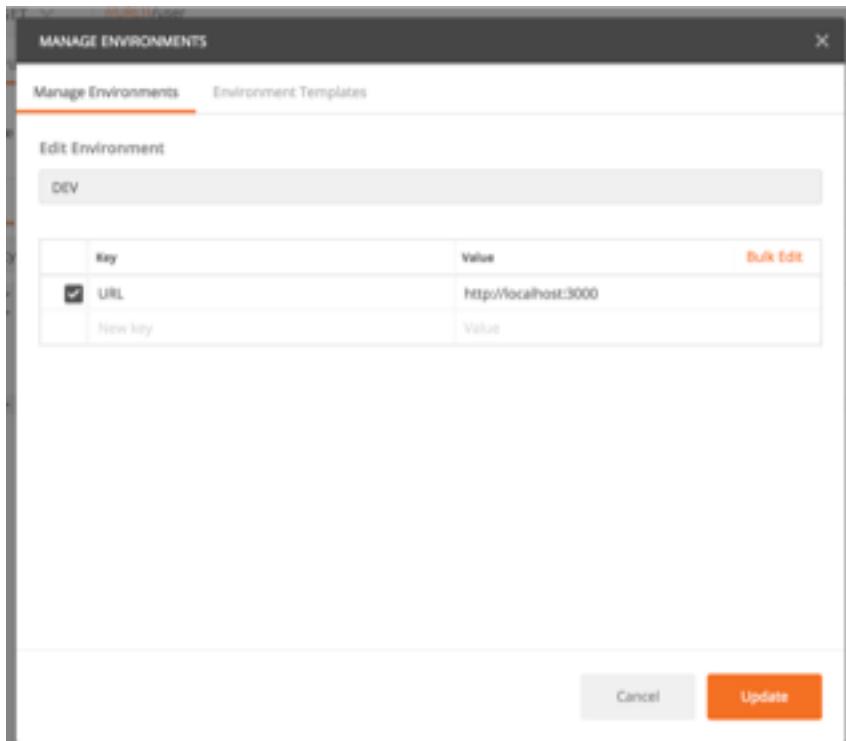


The screenshot shows a modal dialog titled "MANAGE ENVIRONMENTS". It has two tabs: "Manage Environments" (selected) and "Environment Templates". The "Add Environment" section contains a form with fields for "Environment Name" and "Key" and "Value". There are "Bulk Edit" and "Cancel" buttons at the bottom right.

Key	Value
New Key	Value

# Manage Environments

- ว่างถึงตัวแปร  
ใน environment ด้วย  
คำสั่ง {{ชื่อตัวแปร}}

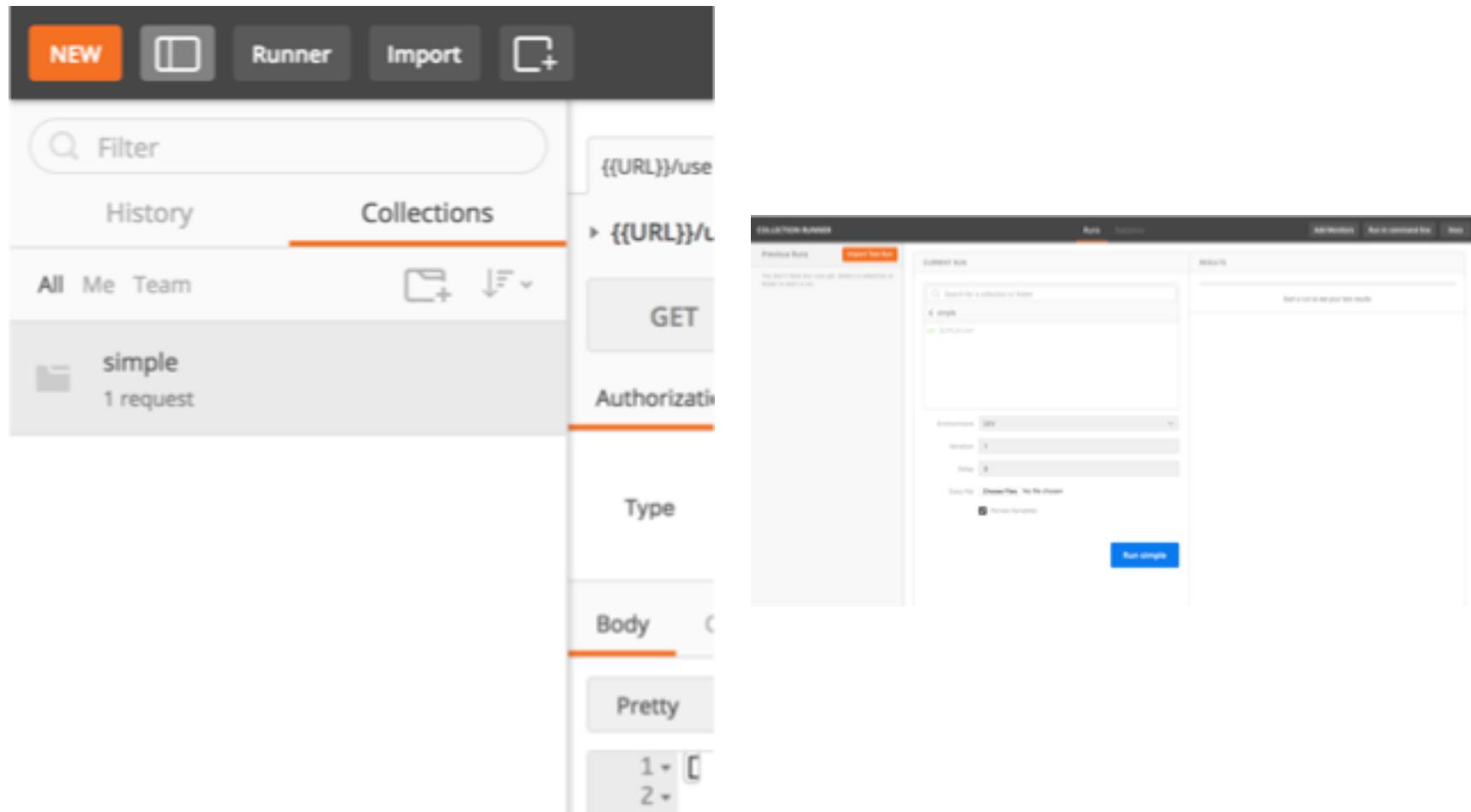


The screenshot shows a REST API request builder interface. The URL is set to 'GET /User'. The 'Authorization' dropdown is set to 'No Auth'. The 'Body' tab is selected, showing a JSON structure with a placeholder: {{id}}. The JSON content is:

```
[{"id": "5a42aabb248c088056ca5e0d", "email": "test@test.com", "password": "admin"}]
```

A blue arrow points upwards from the 'Manage Environments' dialog to the 'Body' field of the API request, indicating the connection between the two.

# POSTMAN RUNNER



The screenshot shows the Postman Runner interface. At the top, there are buttons for NEW, Runner, Import, and a plus sign icon. Below that is a search bar labeled "Filter". The main navigation bar includes "History" and "Collections", with "Collections" being the active tab. Underneath, there are links for "All", "Me", and "Team", along with a folder icon and a download icon. A sidebar on the left lists a collection named "simple" which contains "1 request". The main content area displays a collection structure with tabs for "Authorization", "Type", "Body", and "Pretty". The "Authorization" tab is currently selected, showing fields for "Type", "Name", "Value", and "Scope". The "Body" tab shows a dropdown menu with options "1" and "2". The "Pretty" tab has a "1" button. On the right side, there is a preview pane titled "COLLECTION RUNNER" showing a list of requests and a "Run" button.

พื้นฐาน

# TYPESCRIPT

# Basics of Typescript

- JavaScript that scale
- Starts and ends with JavaScript
- Strong tools for large apps
- State of the art JavaScript

# Basics of Typescript #2

## Create file app.ts

```
var message:string = "Hello World"  
console.log(message)
```

## Compile

```
tsc app.ts
```

## Run

```
node app.js
```

## Output

```
Hello World
```

# TypeScript – Keywords

break	as	any	switch
case	if	throw	else
var	number	string	get
module	type	instanceof	typeof
public	private	enum	export
finally	for	while	void
null	super	this	new
in	return	true	false
any	extends	static	let
package	implements	interface	function
new	try	yield	const
continue	do	catch	

# TypeScript and OOP

```
class Greeting {  
    greet():void {  
        console.log("Hello World!!!")  
    }  
}  
  
var obj = new Greeting();  
obj.greet();
```

# Built-in types

- number
- string
- Boolean
- void
- null
- undefined
- any

# Variable Declaration

var [identifier] : [type] = value ;

**Ex:** var name:string = 'my name is angular.io';

var [identifier] : [type];

**Ex:** var name:string;

var [identifier] = value ;

**Ex:** var name = 'my name is angular.io';

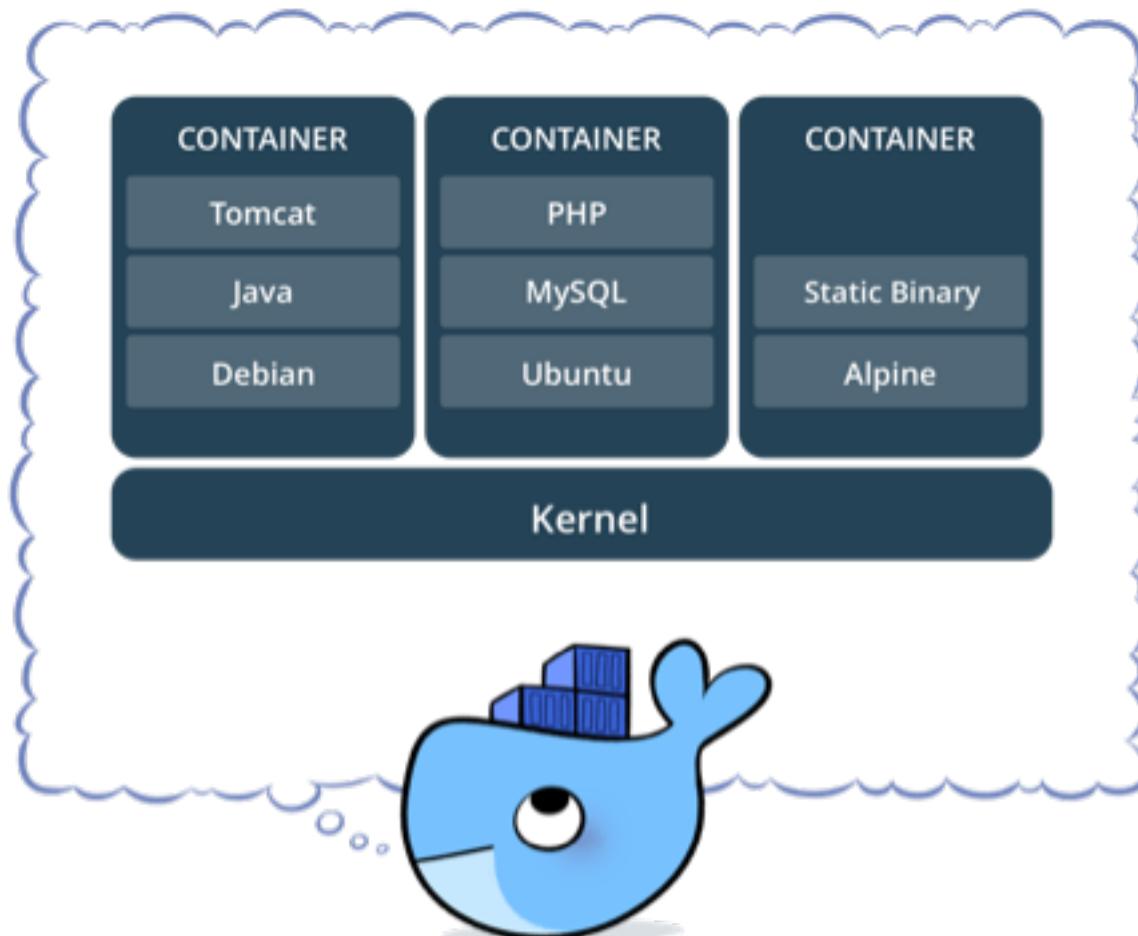
var [identifier] ;

**Ex:** var name;

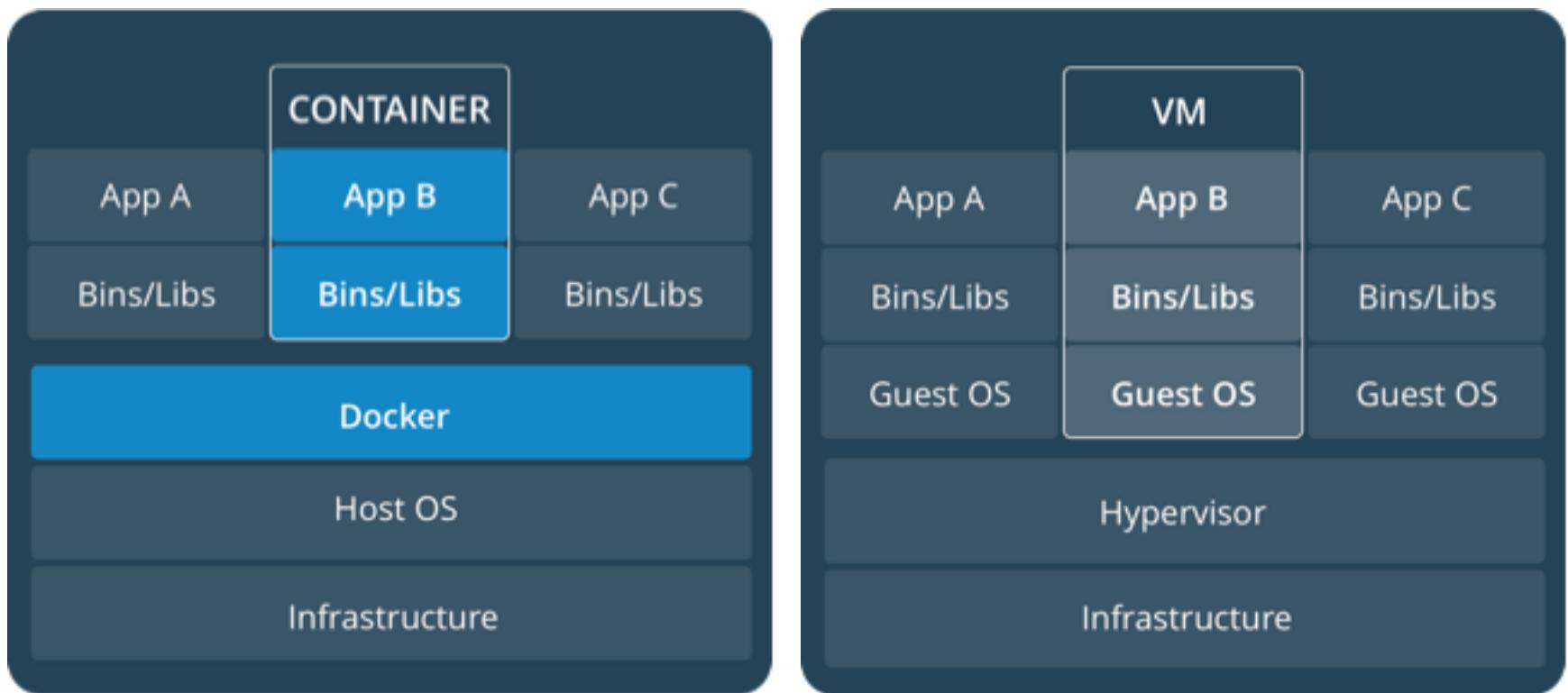
Containers virtualize with

**DOCKER**

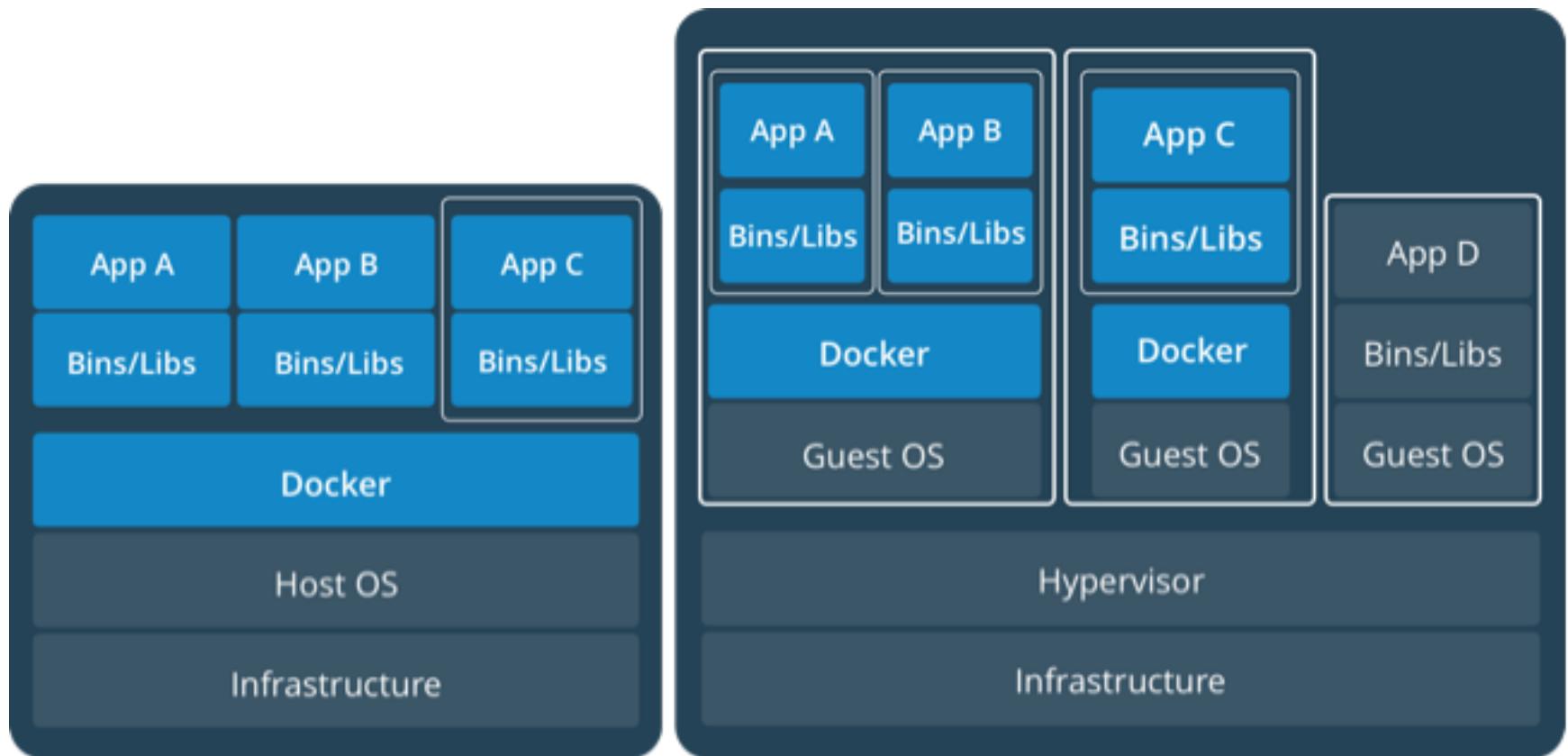
# About Container



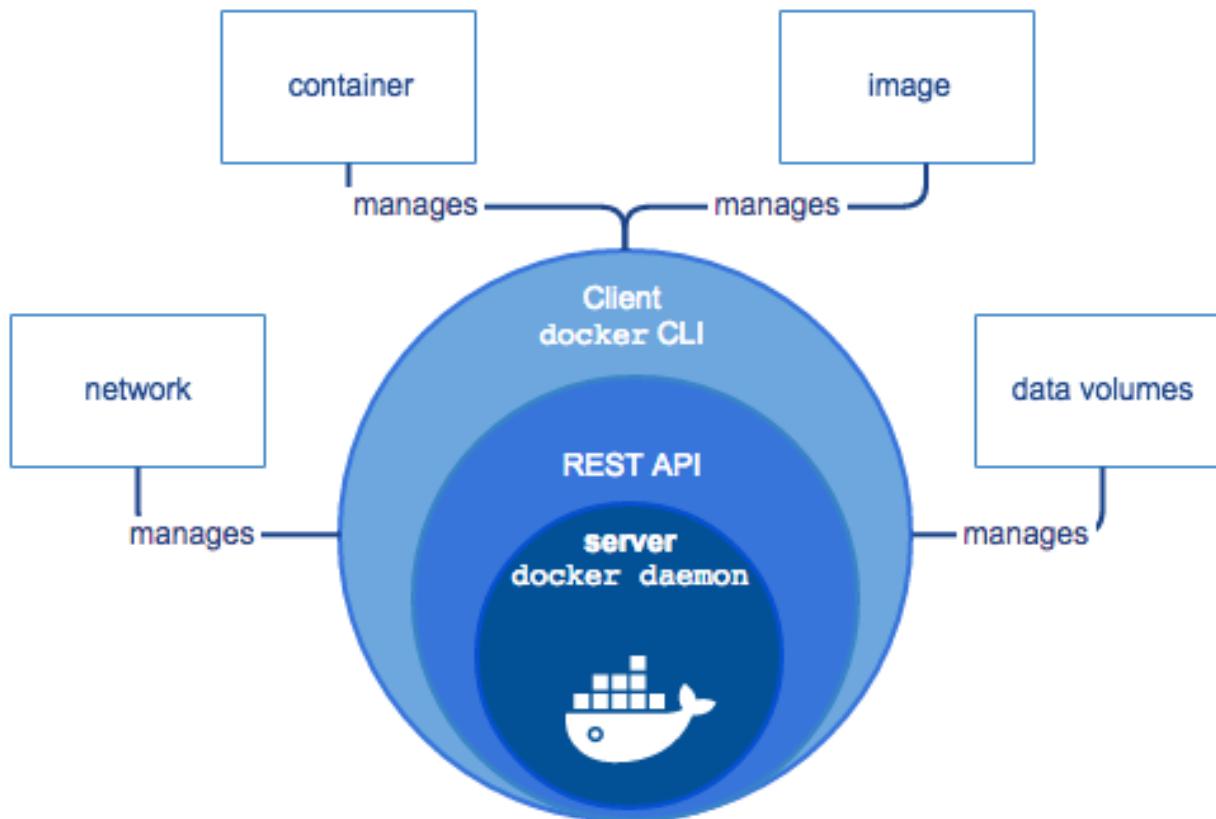
# Comparing Containers and Virtual Machines



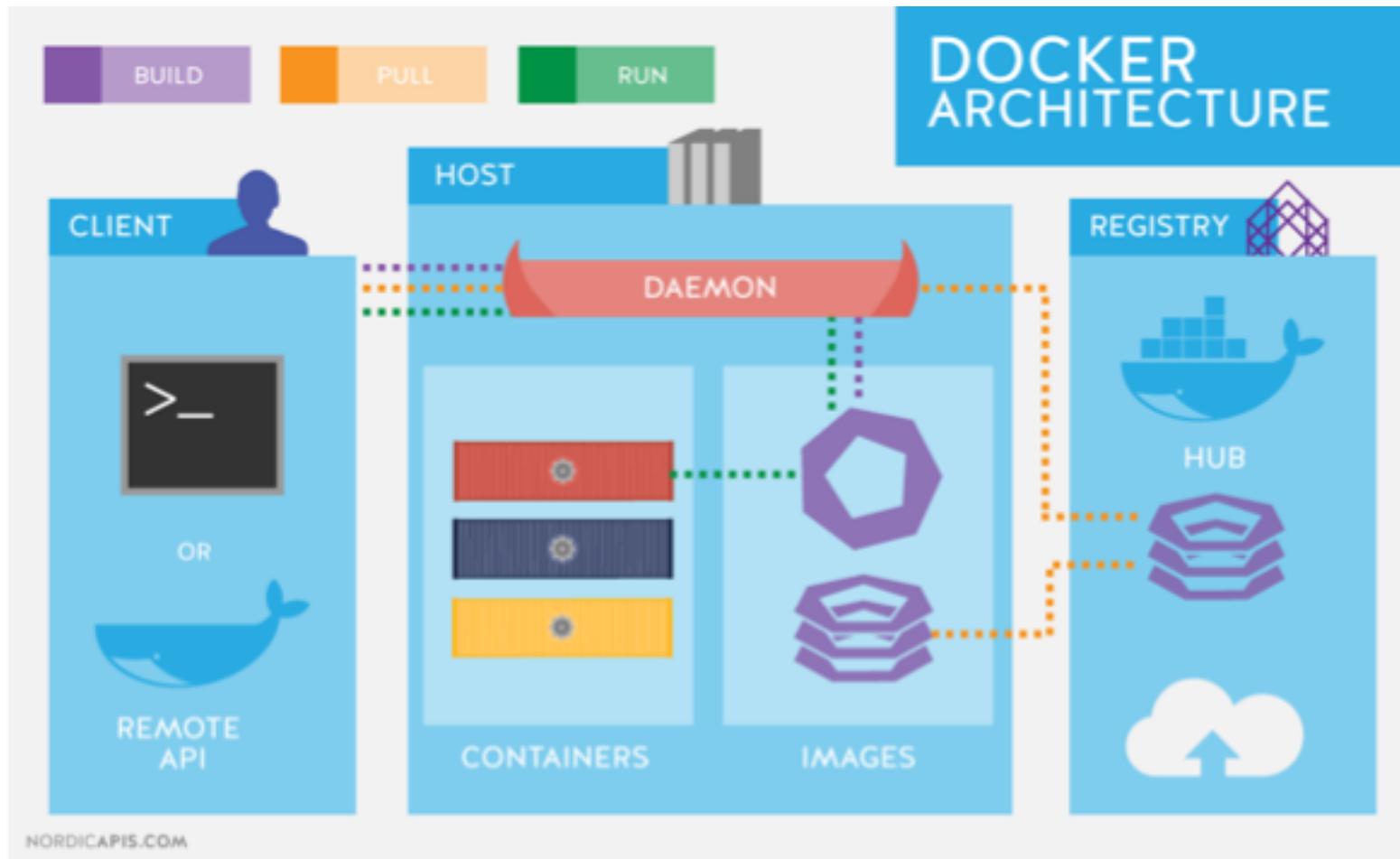
# Containers and Virtual Machines Together



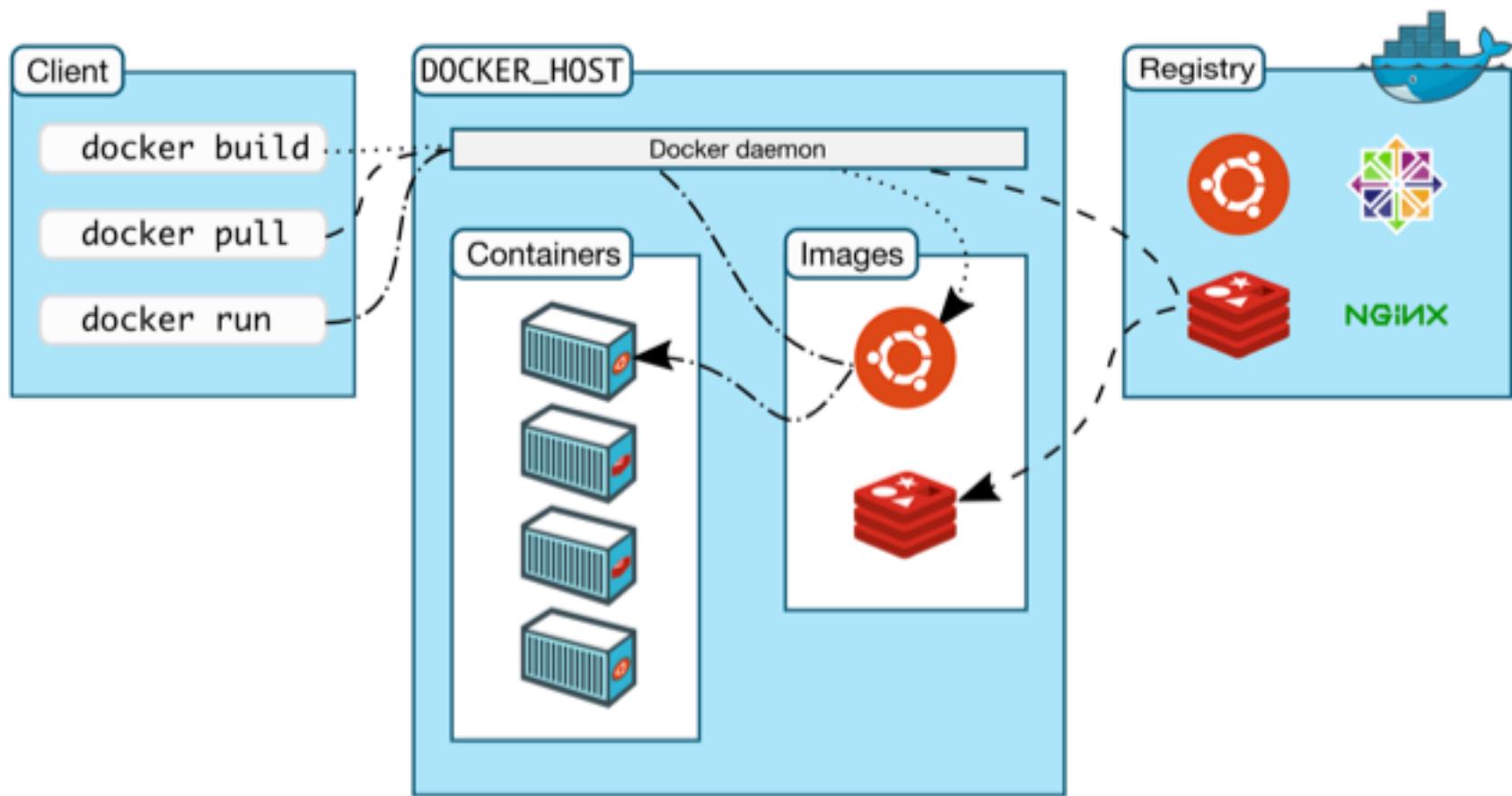
# Docker engine



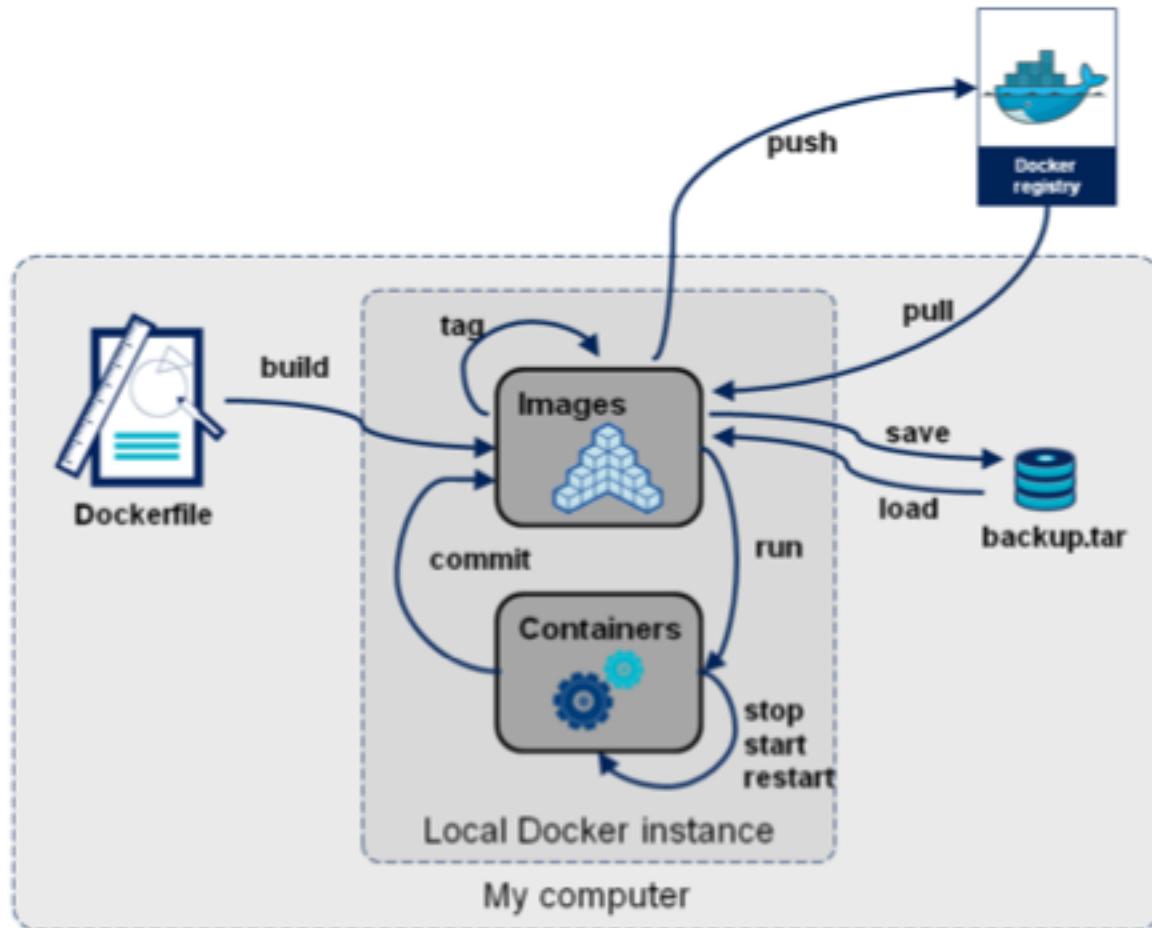
# Docker Architecture



# Docker Architecture#2



# Docker Architecture#3



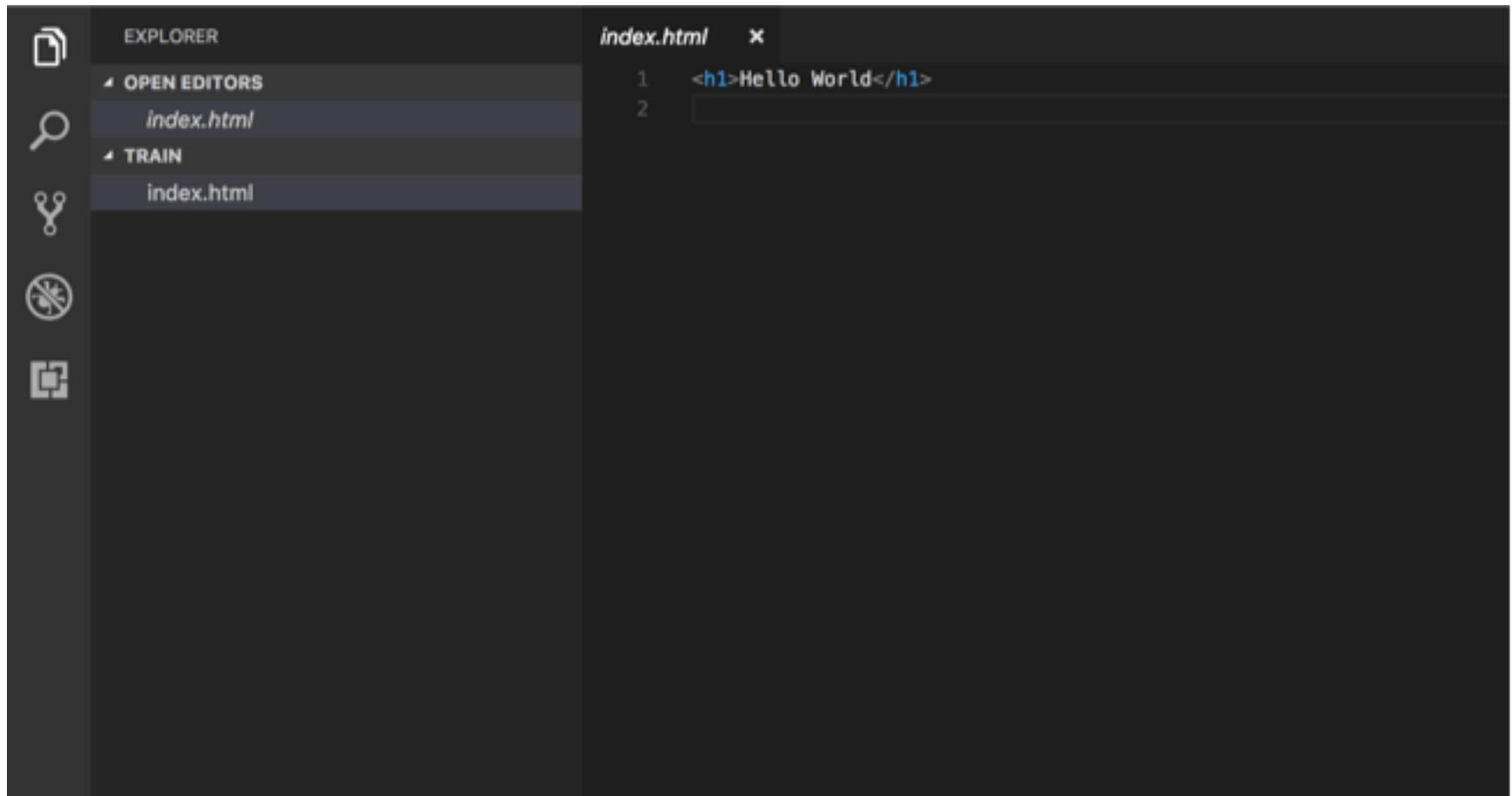
# Hello World Docker

- เปิด cmd และรันคำสั่งดังนี้

```
docker run --name some-nginx \
-v /your_path:/usr/share/nginx/html:ro \
-p 80:80 \
-d nginx
```

\*\*\* link สำหรับหา image <https://hub.docker.com>

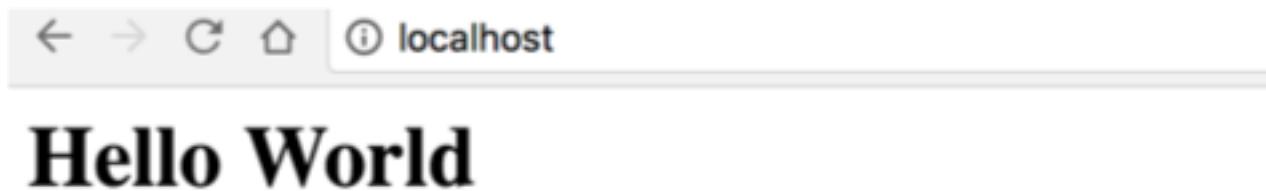
# สร้าง file index.html



The screenshot shows a dark-themed code editor interface. On the left is a vertical toolbar with icons for file operations like Open, Save, Find, and Copy/Paste. To its right is the Explorer panel, which lists 'OPEN EDITORS' containing 'index.html' and 'TRAIN' also containing 'index.html'. The main workspace is titled 'index.html' and contains the following code:

```
1 <h1>Hello World</h1>
2
```

# เข้า url `http://localhost`



# Docker Command

## Login

- docker login
- docker login -u <user\_name>
- docker login -u <user\_name> -p <password>

## Logout

- docker logout

## List all image

- docker images
- docker image ls

# Docker Command

## Search image

- docker search <image name>

## Pull image

- docker pull <image name>

## Create container from image

- docker create <options> <image name>
  - --name
  - -v
  - -p
- docker create --name ubuntu14 -v /user/sommaik:/home ubuntu:14.04

# Docker Command

## Start Container

- docker start <container\_id> or <container\_name>

## Stop Container

- docker stop <container\_id> or <container\_name>

## Stop all container

- docker stop \$(docker ps -a -q)

## List all container

- docker ps <options>

# Docker Command

## Pause Container

- docker pause <container\_id> or <container\_name>

## Unpause Container

- docker unpause <container\_id> or <container\_name>

## Exec Container

- docker exec -it <container\_id> bash

## Inspect Container

- docker inspect <container\_id>

# Docker Command

## Logs container

- docker logs

## Commit Container

- docker commit <container\_id> <new\_image\_name>
- docker commit 2x5t aloha

## Push Image

- docker push <account>/<image name>

## Tag

- docker tag ubuntu ubuntu-x

# Docker Command

## Export container

- docker export <container\_id> > <to\_path>

## Import container

- docker import - <from\_path>

## Save Image

- docker save <image name> > <to path>
- docker save <image name>:<tag> > <to path>

## Load Image

- docker load < <from path>

# Docker Command

## Remove container

- docker rm <container\_id>

## Remove all stop container

- docker rm \$(docker ps -a -q)

## Remove Image

- docker rmi <image\_id>

# Docker Network

- docker network ls
- docker network create <network\_name> default bridge
- docker network create --subnet 10.0.0.1/24 <network\_name>
- docker network inspect <network\_name> or <container\_id>
- docker network create my-net (create images networks)
- docker run --network <network\_name> <image\_name>
- docker run -it --name <container\_name> --net--alias alias2 --network <network\_name> <image\_name>

# Docker parameter

## Run in the background

- -d

## Create name to container is running

- --name

## Port mapping

- -p (local\_port:container\_port)

## Container host name

- -h

# Docker parameter

## Environment

- -e

## Map volume paths

- -v

## Keep STDIN open even if not attached

- -i

## Allocate a pseudo-TTY

- -t

# Dockerfile

## FROM

- FROM <image>[:<tag>]
- FROM ubuntu:14.04

## RUN

- RUN <command>
- RUN echo “Hello World”

## EXPOSE

- EXPOSE <port>
- EXPOSE 8080

# Dockerfile

## ENV

- ENV <key> <value>

## CMD

- CMD command param1 param2

## WORKDIR

- WORKDIR /path/to/workdir

## VOLUME

- VOLUME /path

# Dockerfile

## Build

- docker build <option> <path>
  - -t tag name

## Example

- docker build -t first .

# Compose file

## file name

- docker-compose.yaml

## Example

```
version: '3'
services:
  jenkins:
    container_name: jenkins
    image: jenkins
    volumes:
      - ./jenkins:/var/jenkins_home
    ports:
      - 8080:8080
      - 5000:5000
  ubuntu:
    container_name: ubuntu14
    image: "ubuntu:14.04"
```

# Docker Compose

- docker-compose up -d –build
- docker-compose up --force-recreate
- docker-compose ps
- docker-compose scale web=5
- docker-compose stop
- docker-compose rm

# Any questions?

