

# Kubernetes + Raspberry pi

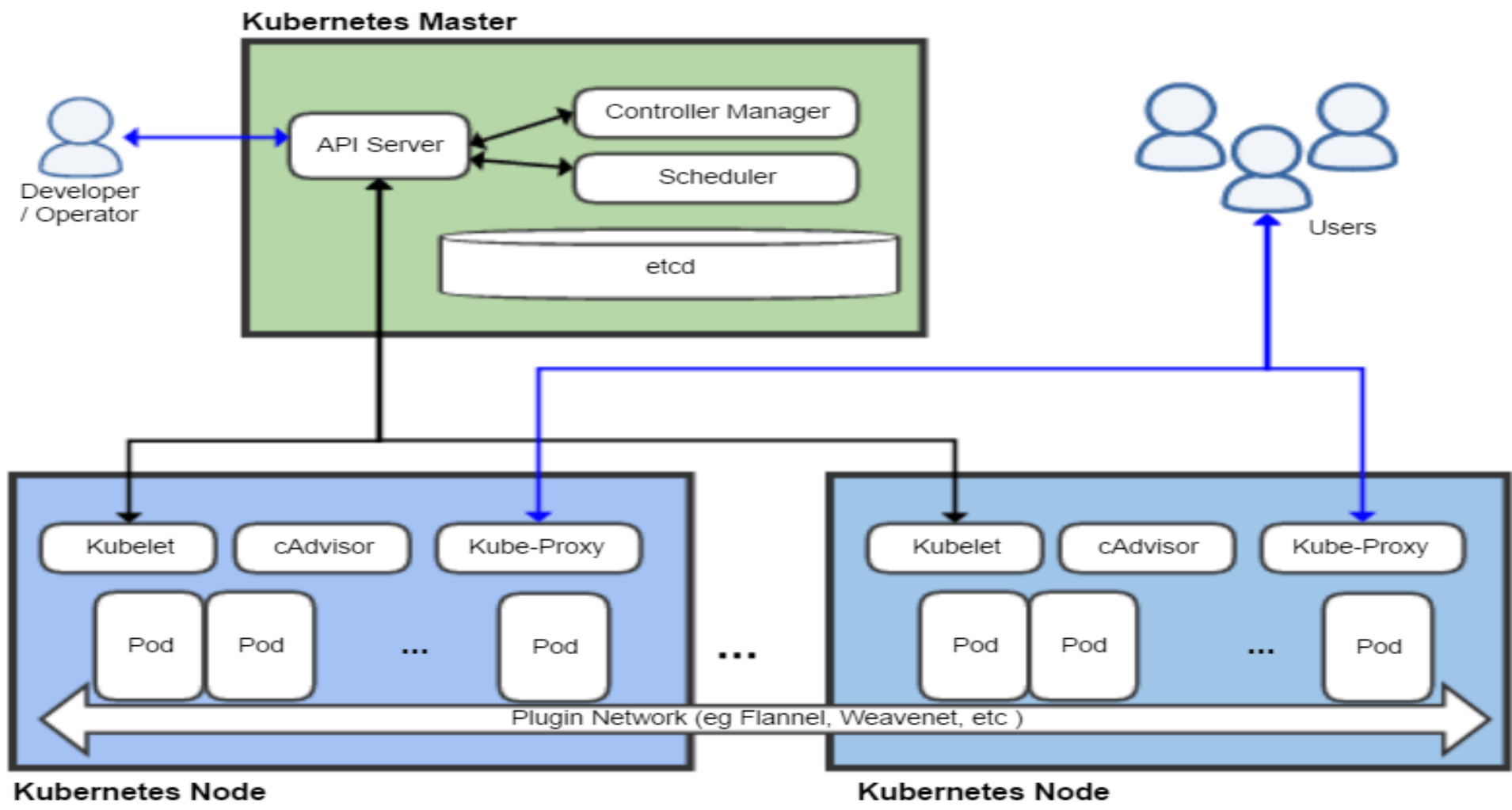
組 員 名 單

104062336 張育瑋



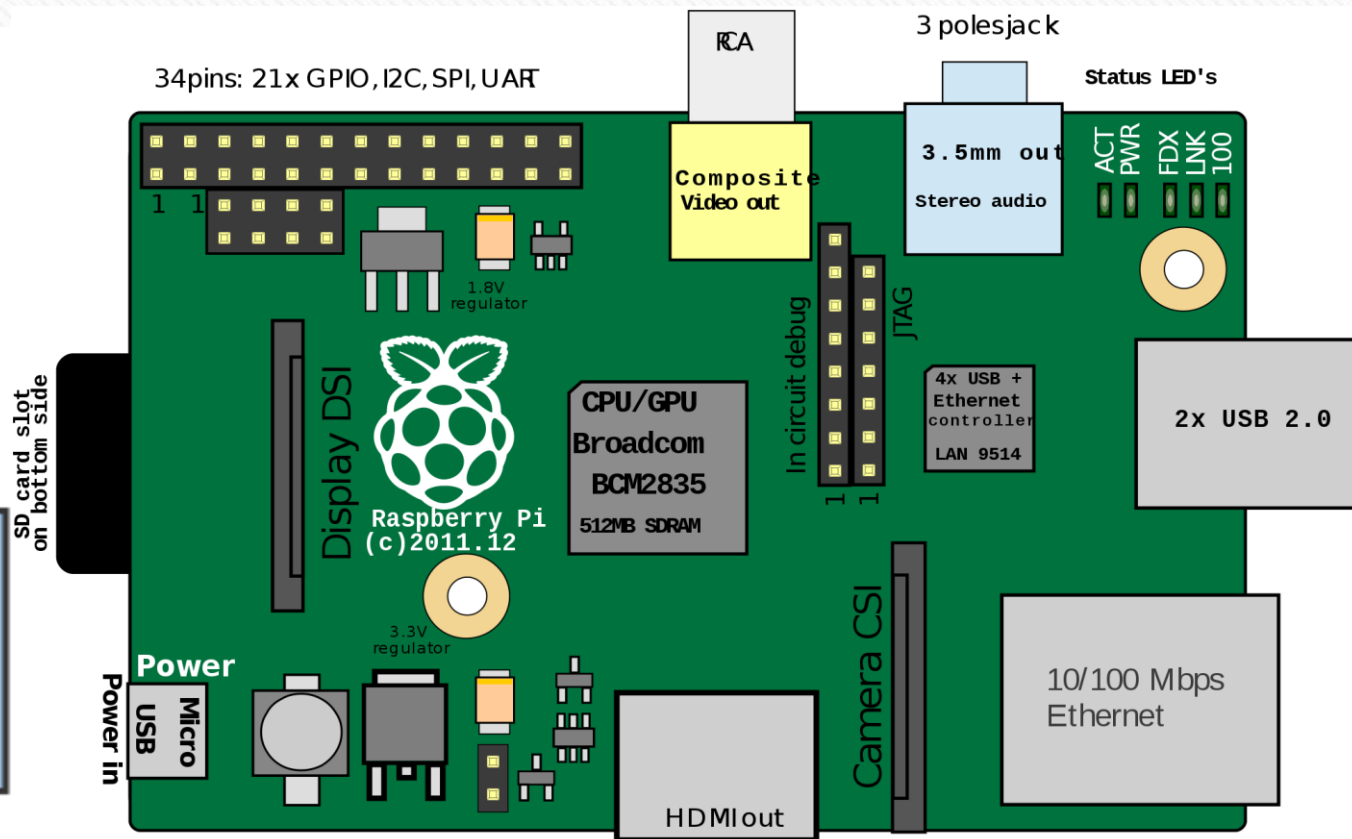
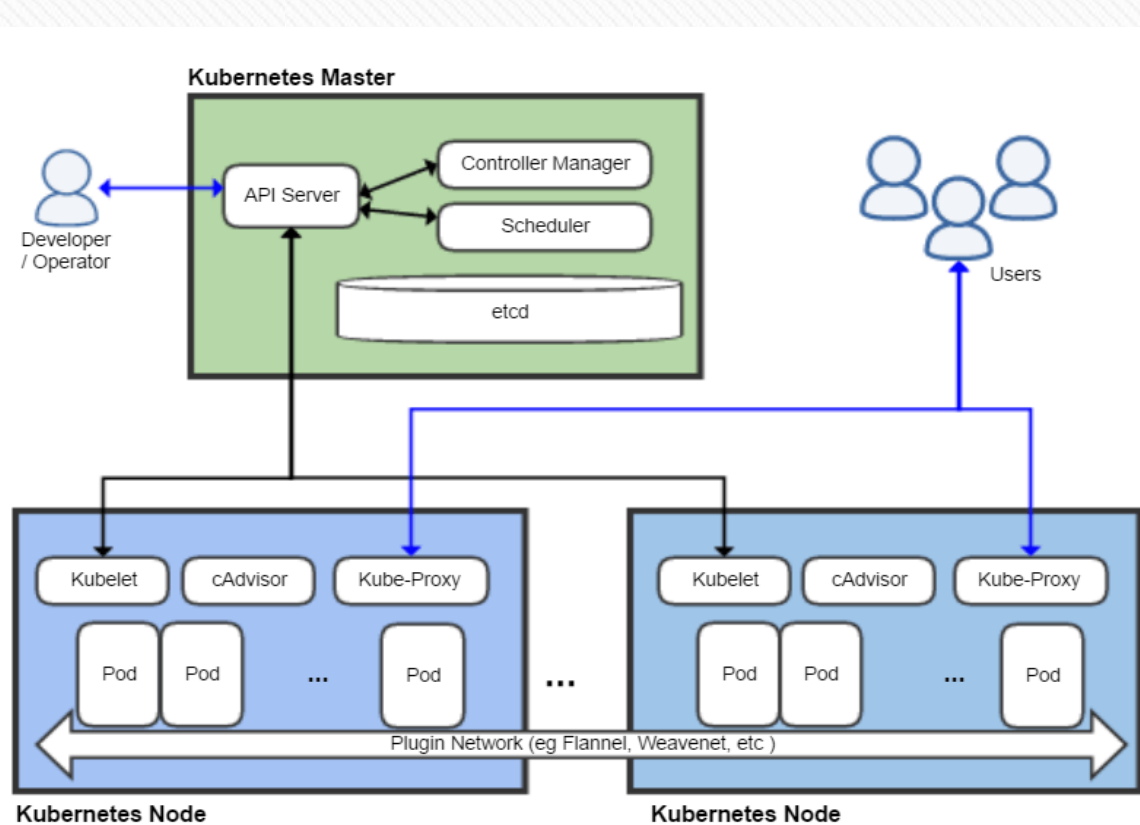
## 動機

- 近年來輕量級容器的發展已經使開發、打包和部署應用程式的方式都大受影響.而其中在輕量級容器管理的工具中,最受人矚目非Kubernetes莫屬. **Kubernetes** 是用於自動部署、擴展和管理容器化 (containerized) 應用程式的開源系統.它旨在提供「跨主機集群的自動部署、擴展以及運行應用程式容器的平台」.也因此,本專題打算使用kubernetes以管理Raspberry pi 上的輕量級容器.
- Kubernetes特別適合微服務這樣的架構。Kubernetes也提供了良好的服務發現 (Service discovery)機制, 讓每個服務彼此可以通信。最重要的是Kubernetes可以提供自動擴展服務, 甚至還可以對大規模的容器作滾動更新(Rolling update) 以及回滾機制(Rolling back/Undo)
- Kubernetes遵循master-slave .Kubernetes Master是集群的主要控制單元,用於管理其工作負載並指導整個系統的通信. Kubernetes控制平面的各種部件如下:



# 整體目標與成果

主要結構如下圖所示, Master與Node由Raspberry pi來負責



# Kubernetes的佈置與研究

## Minikube:單機版

```
/usr/bin/kubectl
--2017-11-30 16:35:29-- https://storage.googleapis.com/kubernetes-release/release/v1.0.1/bin/linux/amd64/kubectl
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.24.16, 2404:6800:4008:803::2010
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.24.16|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20341304 (19M) [application/octet-stream]
Saving to: 'kubectl.1'

kubectl.1          100%[=====] 19.40M  19.7MB/s   in 1.0s

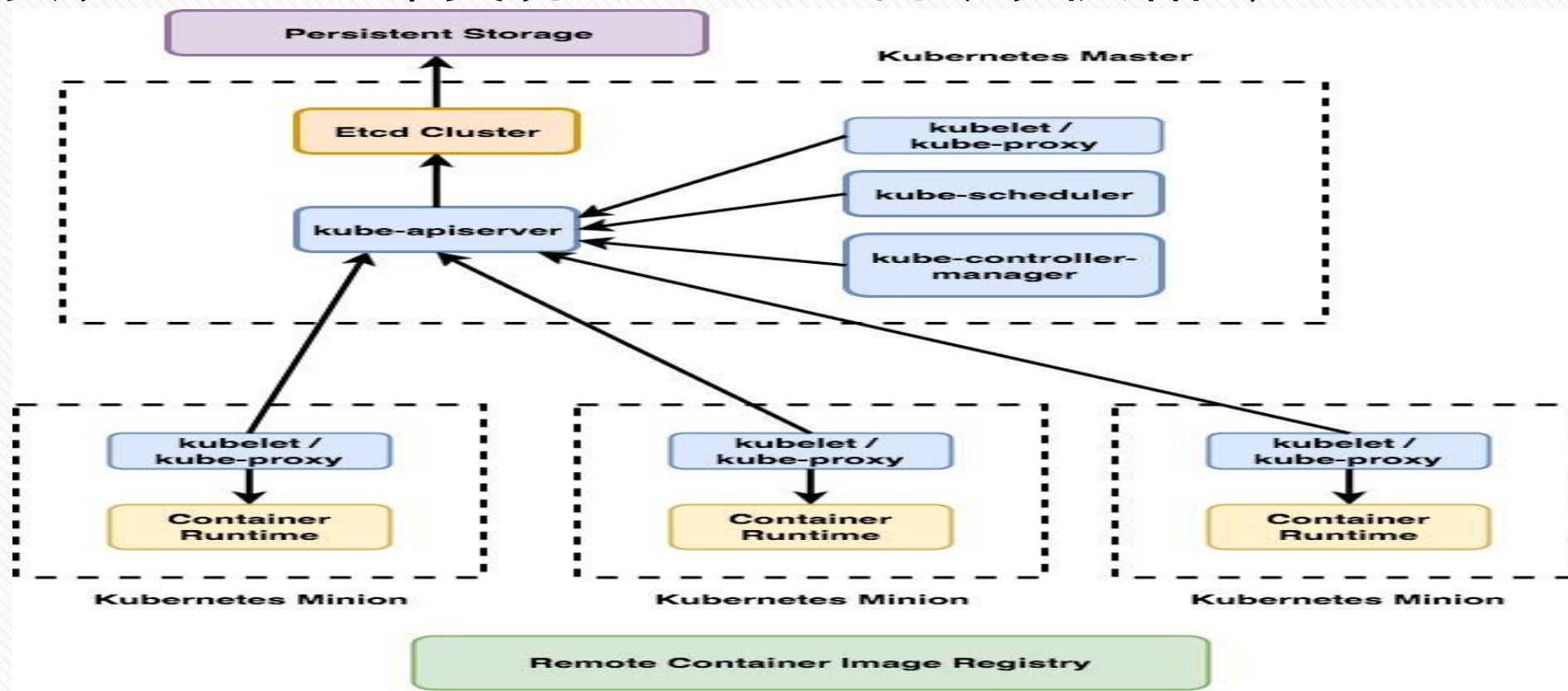
2017-11-30 16:35:30 (19.7 MB/s) - 'kubectl.1' saved [20341304/20341304]

root@ubuntu:/home/andychang# kubectl version
Client Version: version.Info{Major:"1", Minor:"8", GitVersion:"v1.8.4", GitCommit:"9befc2b8928a9426501d3bf62f72849d5cbcd5a3", GitTreeState:"clean", BuildDate:"2017-11-20T05:28:34Z", GoVersion:"go1.8.3", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server 127.0.0.1:8443 was refused - did you specify the right host or port?
root@ubuntu:/home/andychang# kubectl cluster-info
Kubernetes master is running at https://127.0.0.1:8443
```

# Kubernetes的佈置與研究

Kubernetes由許多元件組合而成,而不同的元件又有許多版本,如果欲直接安裝所有元件,版本相容性是一個課題.

藉由安裝kubeadm來實現kubernetes的環境初始化,





# Kubernetes的佈置與研究

## Kubernetes master初始化

```
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

kubeadm join --token 29191f.c885b4fdcc9e76e6 10.211.55.27:6443 --discovery-token-ca-cert-hash
sha256:4628a304648b8aae
```

當有其他node要加入此cluster時, 最下面兩行提供的資訊將派上用場.  
接下來我們為了要讓pod能夠互相溝通,我們必須設置pod network.

# Kubernetes的佈置與研究

我們可藉由官網:<https://kubernetes.io/docs/concepts/cluster-administration/addons/> 所提供的 pod network 選擇其一來設置.

設置完成後,我們可藉由**kubectl get pods --all-namespaces** 指令來得知目前的配置資訊,如下圖所示:

```
root@ubuntu:/home/robotox# kubectl get pods --all-namespaces
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE
kube-system  etcd-ubuntu                             1/1     Running   0           10m
kube-system  kube-apiserver-ubuntu                   1/1     Running   0           10m
kube-system  kube-controller-manager-ubuntu          1/1     Running   0           11m
kube-system  kube-dns-6f4fd4bdf-9gq7s                0/3     Pending   0           11m
kube-system  kube-flannel-ds-j9l2x                    1/1     Running   0           20s
kube-system  kube-proxy-vnrpw                          1/1     Running   0           11m
kube-system  kube-scheduler-ubuntu                    1/1     Running   0           10m
```

至此k8s的佈置已經告一段落.

我們可以藉由建立相關的RC定義檔 (.yaml) 嘗試建立pod以及service, 再利用以下指令:

```
Kubectl create -f nginx-deployment.yaml
```

即可生成相關服務的pod.

```
Kubectl get rc(replication controller)
```

```
Kubectl get pods
```



# Kubernetes的佈置與研究

官網提供的方法: Communicate Between Containers in the Same Pod Using a Shared Volume

<https://kubernetes.io/docs/tasks/access-application-cluster/communicate-containers-same-pod-shared-volume/>

利用.yaml檔建立pod,並在檔案中宣告Shared Volume以及欲建立之containers:

在yaml 檔內定義Pod的結構, 這裡使用官方提供的two-container做示範 在yaml裡面設定一個volume, 可視為一個所有container共通的資料夾, 接著 在各個container中設立此shared volume的位置。這個範例在debian-container裡面對 這個共用的資料夾創立了一個html檔, 並使用nginx-container去呼叫,如下圖所示。

```
apiVersion: v1
kind: Pod
metadata:
  name: two-containers
spec:
  restartPolicy: Never
  volumes:
  - name: shared-data
    emptyDir: {}
  containers:
  - name: nginx-container
    image: nginx
    volumeMounts:
    - name: shared-data
      mountPath: /usr/share/nginx/html
  - name: debian-container
    image: debian
    volumeMounts:
    - name: shared-data
      mountPath: /pod-data
    command: ["/bin/sh"]
    args: ["-c", "echo Hello from the debian container > /pod-data/index.html"]
```

可從另一個container裡面  
下指令看到另一個  
container對此shared  
volume的改動

```
root@two-containers:~# curl localhost
Hello from the debian container
root@two-containers:~#
```

# Kubernetes的佈置與研究

Dashboard:可將你kubernetes系統中的狀況顯示出來

The screenshot shows the Kubernetes Dashboard interface. At the top, there is a search bar and a '+ CREATE' button. The main navigation bar is blue and labeled 'Workloads'. On the left, a sidebar lists various Kubernetes resources: Cluster, Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Namespace (set to 'default'), Workloads (highlighted), Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, and Stateful Sets.

The main content area displays three sections:

- Deployments:** A table with columns: Name, Labels, Pods, Age, Images. One deployment is listed: 'guids' with label 'run: guides', 1 / 1 pods, and an age of 'an hour'.
- Pods:** A table with columns: Name, Status, Restarts, Age. One pod is listed: 'guids-2617315942-lz...' with status 'Running', 0 restarts, and an age of 'an hour'.
- Replica Sets:** A table with columns: Name, Labels, Pods, Age, Images. One replica set is listed: 'guids-2617...' with labels 'pod-tem...' and 'run: guides', 1 / 1 pods, and an age of 'an hour'.

# Kubernetes的佈置與研究

We can use Kubectl to create and launch Deployments, Replication Controllers and expose them via Services without writing *yaml* definitions

```
kubectl run http --image=katacoda/docker-http-server:latest --replicas=1  
kubectl get deployments  
kubectl describe deployment http
```

The description includes how many replicas are available, labels specified and the events associated with the deployment. These events will highlight any problems and errors that might have occurred

```
$ kubectl get deployments
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
http	1	1	1	0	2s

# Kubernetes的佈置與研究

```
izeInakri: ~/kubetest kubectl describe all
Name:                sweet-toucan-postgresql
Namespace:           default
CreationTimestamp:   Fri, 24 Aug 2018 19:44:07 +0200
Labels:              app=postgresql
                    chart=postgresql-0.15.0
                    heritage=Tiller
                    release=sweet-toucan
Annotations:         deployment.kubernetes.io/revision=1
Selector:            app=postgresql,release=sweet-toucan
Replicas:            1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:       Recreate
MinReadySeconds:    0
Pod Template:
  Labels:  app=postgresql
          release=sweet-toucan
Containers:
  sweet-toucan-postgresql:
    Image:  postgres:9.6.2
    Port:  5432/TCP
    Requests:
      cpu:          100m
      memory:       256Mi
    Liveness:  exec [sh -c exec pg_isready --host $POD_IP] delay=60s timeout=5s period=10s #success=1 #failure=6
    Readiness: exec [sh -c exec pg_isready --host $POD_IP] delay=5s timeout=3s period=5s #success=1 #failure=3
    Environment:
      POSTGRES_USER:      postgres
      PGUSER:             postgres
      POSTGRES_DB:
      POSTGRES_INITDB_ARGS:
```



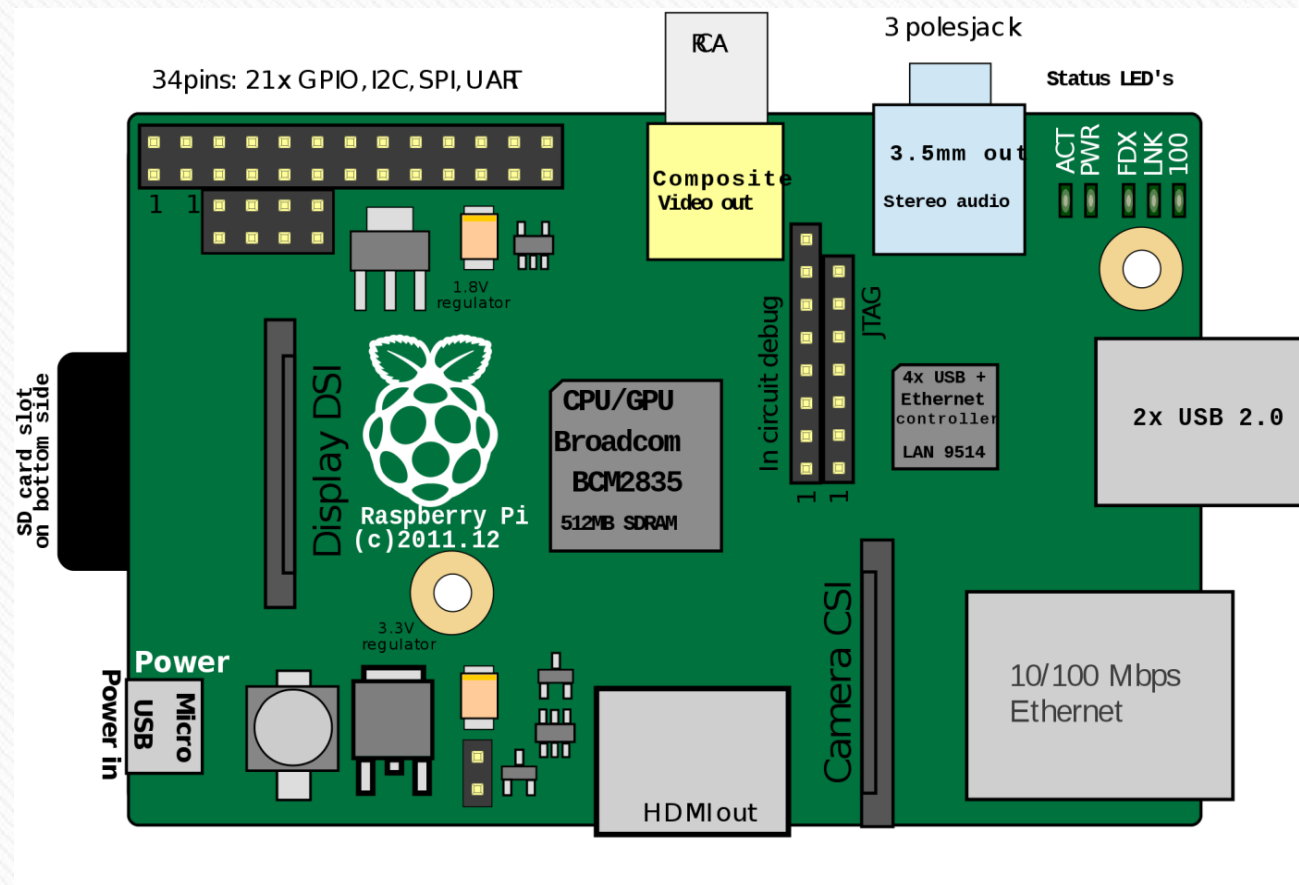
# Kubernetes的佈置與研究

```
$ kubectl get deployments
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
http          1         1         1            0          2s
$ kubectl describe deployment http
Name:          http
Namespace:    default
CreationTimestamp: Thu, 03 May 2018 08:16:06 +0000
Labels:       run=http
Annotations:  deployment.kubernetes.io/revision=1
Selector:     run=http
Replicas:     1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 1 max unavailable, 1 max surge
Pod Template:
  Labels:  run=http
  Containers:
    http:
      Image:          katacoda/docker-http-server:latest
      Port:           <none>
      Host Port:     <none>
      Environment:   <none>
      Mounts:        <none>
      Volumes:       <none>
  Conditions:
    Type           Status   Reason
    ----           -
    Available      True    MinimumReplicasAvailable
    Progressing    True    NewReplicaSetAvailable
  OldReplicaSets: <none>
```

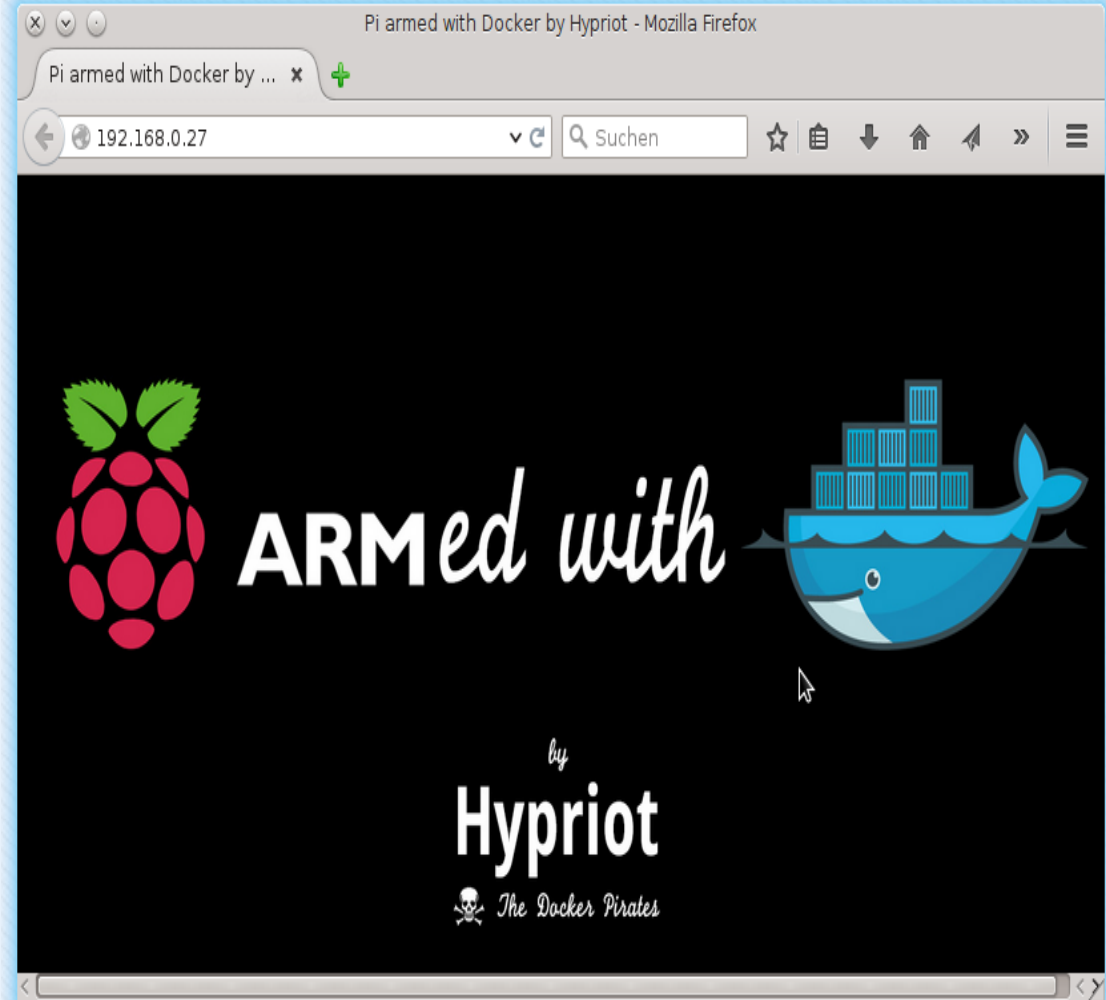


# 在RaspberryPi上安裝Raspbian以及kubernetes

燒錄SD卡以及無限網路設定以及設定node  
等等工作來完成Raspberry pi環境的佈置與設定.



# 在RaspberryPi上安裝Raspbian.Docker以及kubernetes



# 在RaspberryPi上安裝Raspbian以及kubernetes

```
$sudo nano /etc/network/interfaces
```

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
#iface wlan0 inet manual
auto wlan0
iface wlan0 inet static
Address
Netmask
Gateway
wpa-ssid
wpa-psk
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```



# 在RaspberryPi上安裝Raspbian以及kubernetes

```
Starting Update UTP about System Boot/Shutdown...
Starting Network Time Synchronization...
[ OK ] Started Update UTP about System Boot/Shutdown.
[ OK ] Started Raise network interfaces.
[ OK ] Started Network Time Synchronization.
[ OK ] Reached target System Initialization.
[ OK ] Listening on D-Bus System Message Bus Socket.
[ OK ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[ OK ] Listening on triggerhappy.socket.
[ OK ] Reached target Sockets.
[ OK ] Reached target Basic System.
Starting Disable WiFi if country not set...
Starting triggerhappy global hotkey daemon...
Starting dhcpcd on all interfaces...
Starting LSB: Switch to ondemand cpu governor (unless shift key is pressed)...
[ OK ] Started D-Bus System Message Bus.
Starting System Logging Service...
Starting Avahi mDNS/DNS-SD Stack...
Starting Configure Bluetooth Modems connected by UART...
Starting LSB: Autogenerate and use a swap file...
[ OK ] Started Regular background program processing daemon.
Starting Save/Restore Sound Card State...
[ OK ] Started Daily Cleanup of Temporary Directories.
Starting Login Service...
[ OK ] Reached target System Time Synchronized.
[ OK ] Started Daily apt download activities.
[ OK ] Started Daily apt upgrade and clean activities.
[ OK ] Reached target Timers.
[ OK ] Started triggerhappy global hotkey daemon.
[ OK ] Started Disable WiFi if country not set.
[ OK ] Started Save/Restore Sound Card State.
[ OK ] Started System Logging Service.
[ OK ] Started Avahi mDNS/DNS-SD Stack.
[ OK ] Started Login Service.
Starting Load/Save RF Kill Switch Status...
[ OK ] Started Load/Save RF Kill Switch Status.
[ OK ] Started LSB: Autogenerate and use a swap file.
[ OK ] Started LSB: Switch to ondemand cpu governor (unless shift key is pressed).
[ OK ] Started Configure Bluetooth Modems connected by UART.
Starting Bluetooth service...
[ OK ] Started Bluetooth service.
[ OK ] Reached target Bluetooth.
Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Started dhcpcd on all interfaces.
[ OK ] Reached target Network.
Starting OpenSSH Secure Shell server...
Starting Permit User Sessions...
[ OK ] Reached target Network is Online.
Starting Daily apt download activities...
Starting /etc/rc.local Compatibility...
[ OK ] Started Permit User Sessions.
[ OK ] IP address is 2a02:810c:290:4639:6cae:d5a:5f99:96a4
[ OK ] Started /etc/rc.local Compatibility.
Starting Hold until boot process finishes up...
Starting Terminate Plymouth Boot Screen...

#Raspbian GNU/Linux 9 raspberrypi tty1
#Raspberrypi login:
```

# 在RaspberryPi上安裝Raspbian以及kubernetes

```
pi@master:~ $ sudo su
root@master:/home/pi# kubectl get nodes
NAME        STATUS    ROLES    AGE     VERSION
master      Ready    master   4m30s   v1.12.1
root@master:/home/pi# kubectl get nodes
NAME        STATUS    ROLES    AGE     VERSION
master      Ready    master   4m34s   v1.12.1
root@master:/home/pi# kubectl get pods --namespace=kube-system
NAME                                READY    STATUS    RESTARTS   AGE
coredns-576cbf47c7-nh77j            1/1     Running   0           4m27s
coredns-576cbf47c7-s47n9            1/1     Running   0           4m27s
etcd-master                          1/1     Running   0           4m31s
kube-apiserver-master                1/1     Running   0           4m36s
kube-controller-manager-master       1/1     Running   0           4m31s
kube-proxy-vn2ks                     1/1     Running   0           4m27s
kube-scheduler-master                1/1     Running   0           4m36s
weave-net-mxrmv                      2/2     Running   0           3m20s
root@master:/home/pi#
```



# 在RaspberryPi上安裝Raspbian以及kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: two-containers
spec:

  restartPolicy: Never

  volumes:
  - name: shared-data
    emptyDir: {}

  containers:

  - name: nginx-container
    image: nginx
    volumeMounts:
    - name: shared-data
      mountPath: /usr/share/nginx/html

  - name: debian-container
    image: debian
    volumeMounts:
    - name: shared-data
      mountPath: /pod-data
    command: ["/bin/sh"]
    args: ["-c", "echo Hello from the debian container > /pod-data/index.html"]
```

# 在RaspberryPi上安裝Raspbian以及kubernetes

Operation	Syntax	Description
<b>annotate</b>	<code>kubectl annotate (-f FILENAME \   TYPE NAME \   TYPE/NAME) KEY_1=VAL_1 ... KEY_N=VAL_N [--overwrite] [--all] [--resource-version=version] [flags]</code>	Add or update the annotations of one or more resources.
<b>api-versions</b>	<code>kubectl api-versions [flags]</code>	List the API versions that are available.
<b>apply</b>	<code>kubectl apply -f FILENAME [flags]</code>	Apply a configuration change to a resource from a file or stdin.
<b>attach</b>	<code>kubectl attach POD -c CONTAINER [-i] [-t] [flags]</code>	Attach to a running container either to view the output stream or interact with the container (stdin).
<b>autoscale</b>	<code>kubectl autoscale (-f FILENAME \   TYPE NAME \   TYPE/NAME) [--min=MINPODS] --max=MAXPODS [--cpu-percent=CPU] [flags]</code>	Automatically scale the set of pods that are managed by a replication controller.
<b>cluster-info</b>	<code>kubectl cluster-info [flags]</code>	Display endpoint information about the master and services in the cluster.
<b>config</b>	<code>kubectl config SUBCOMMAND [flags]</code>	Modifies kubeconfig files. See the individual subcommands for details.
<b>create</b>	<code>kubectl create -f FILENAME [flags]</code>	Create one or more resources from a file or stdin.
<b>delete</b>	<code>kubectl delete (-f FILENAME \   TYPE [NAME \   /NAME \   -l label \   --all]) [flags]</code>	Delete resources either from a file, stdin, or specifying label selectors, names, resource selectors, or resources.
<b>describe</b>	<code>kubectl describe (-f FILENAME \   TYPE [NAME_PREFIX \   /NAME \   -l label]) [flags]</code>	Display the detailed state of one or more resources.
<b>edit</b>	<code>kubectl edit (-f FILENAME \   TYPE NAME \   TYPE/NAME) [flags]</code>	Edit and update the definition of one or more resources on the server by using the default editor.

# Reference

- <https://tutorials.ubuntu.com/tutorial/install-kubernetes-with-conjure-up#0>
- <https://kubernetes.io/docs/setup/independent/install-kubeadm/>
- <https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/>
- <https://kubernetes.io/docs/concepts/cluster-administration/addons/>
- <https://kubernetes.io/docs/tasks/access-application-cluster/communicate-containers-same-pod-shared-volume/>
- <https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment/>
- <https://blog.hypriot.com/post/setup-kubernetes-raspberry-pi-cluster/8>.
- <https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/>
- <https://kubernetes.io/docs/reference/kubectl/docker-cli-to-kubectl/>

# Reference

<http://docs.kubernetes.org.cn/312.html>

<https://www.ithome.com.tw/news/122908>



謝謝聆聽

THANKS FOR YOUR LISTENING