-This Doc : https://tinyurl.com/yybwptwn

- What is kubernetes ?
    - Container orchestrator
    - Workload placement
    - Infra abstraction
    - Desired state
- Benefits
    - Speed of deployment
    - Change quickly
    - Upgrade quickly
    - Recover quickly
    - Resource management abstraction
- K8s principles
    - Desired state - declarative language
    - Controllers - control loops
    - One master - API server - core of k8s
- K8s API server
    - API objects - represent state of the system
    - Enable configuration of stat
        - Declaratively - desc a deployment
        - Imperatively - running sequence of commands
    - RESTful API - HTTP over JSON
        - Sole way to interact with k8s
    - API objects
        - PODs - collection of containers deployment as a unit
        - COntrollers - keep the system in desired state
        - Services - access point to deployed applications
        - Storage - data storage , persistent storage
    - PODs
        - Represents One or more containers
        - Unit of scheduling
        - Most basic unit of work
        - Defines resources required for deployment
        - Ephemeral - no Pod is ever redeployed , no state maintained
        - Atomicity - its there or its not there
        - K8s job is keeping the pods running - keeping desired state
        - K8s keeps track of the state of a pod
            - Health check if pod is running - liveness probes
    - Controllers
        - Creates and manages pods for you
        - Define your desired state

        - Respond to pod state and heath
        - Types of controllers (API objects)
            - ReplicaSet
                - Number of replicas - number of pods
            - Deployment
                - Manages rollout of ReplicaSet
            - There are many more controllers
    - Services
        - Add persistency to ephemeral world
        - Networking abstraction for pod access
        - IP and DNS
        - Redeployed pods automatically updated with service access staying intact
        - Scaled by adding/removing pods
        - Load balancing
    - Storage in K8S

K8S architecture

- Master - control functions of the cluster, primary access point of cluster
- Node -where app pods run, implement networking, ensure reachability
- Scheduled/Add Ons - example DNS

Master Components

- API server - access point for cluster and admin operations
- Cluster store - keeping state of k8s objects
- Scheduler - scheduling
- Controller Manager - keeping things in desired state

Kubectl - primary admin command

Node Components

- Kubelet - starting pods
- kube-proxy - pod networking and implementing abstaration for service
- Container Runtime - pulling container images from repo and providing execution env for containers
- Kublete and kube-proxy talk to api server
- Kubelet, kube-proxy, Container runtime run all on node in cluster including master

Scheduled/Add-on pods

- Provides special service to the cluster - example
    - DNS
        - Pods nodes and services register their IPs
        - Used for service discovery
    - Ingress

        - Advances http load balancer / content router
    - Dashboard

K8s networking requirements

- All pods can communicate with each other on all Nodes
- All nodes can communicate with all pods
- No Network Address Translation (NAT)

Networking Fundamentals

Installation and Configuration of Kubernetes

Installation Considerations

- Where to install ?
    - Cloud
        - IaaS - virtual machines
        - PaaS - Managed Service :
    - On-Prem
        - Bare Metal
        - Virtual Machines
- Which one do you choose ? skills, strategy, options
    - https://v1-13.docs.kubernetes.io/docs/setup/pick-right-solution/
- Cluster Networking
- Scalability
- High Availability
- Disaster Recovery

Installation Methods

- Desktop - dev environment, playing around
- Kubeadm
- From Scratch - a great way to learn the hardway
    - https://kubernetes.io/docs/setup/scratch/
    - https://github.com/kelseyhightower/kubernetes-the-hard-way/
- Cloud Scenarios

System Requirements

- Linux
- 2 CPUs
- 2GB RAM
- Swap Disabled

Container Runtime

- Container Runtime Interface(CRI) compliant - Docker

Networking

- Connectivity between all nodes

Cluster Network Ports

Component	Ports(tcp)	Used By
API	6443	All
etcd	2379-2380	API/etcd
Scheduler	10251	Self
Controller Manager	10252	Self
Kubelet	10250	Control Plane
Kubelet	10250	Control Plane
NodePort	30000-32767	All

Installation Overview
1- Install K8S
2- Create the cluster
3- Configure Pod networking
4- Join Nodes to cluster

Required Packages
On all nodes and master
- kubelet
- kubeadm
- kubectl
- Container Runtime - Docker

Installing a Cluster with kubeadm

```
kubeadm init
```

Step-1 : pre-flight checks
Step-2 : creates CA (cert)
Step-3 : Generate kubeconfig files
Step-4 : Generates static pod manifests (for control plane containers)
Step-5 : Starts up the Control plane
Step-6: Taints the master
Step-7 : Generates a Bootstrap Token
Step-8: Start Add-On Pods : DNS and kube-proxy

Certificate Authority (CA)

- Self-signed CA
- Can be part of an external PKI

  Securing cluster communications

  Auth of users and kubelets

  CA files : /etc/kubernetes/pki

  Kubeconfig files
  - Used to define how to connect to your cluster
    - Cert info
    - Cluster location
  - /etc/kubernetes
    - Admin.conf : kubernetes-admin
    - Kubelet.conf
    - Controller-manager.conf
    - Scheduler.conf

  Static Pod Manifests
- Manifest describes configuration
- /etc/k8s/manifests
- Monitored by kubelet

Pod networking

Overlay networking
Flannel - layer 3 virtual network
Calico - L3 and policy based traffic management
Weave Net - multi-host Docker network

Creating a master
Adding a Node to a Cluster

Managed Cloud Deployment Scenarios

AWS -> EKS
GCP-> GKE
AZURE -> AKS

Using kubectl

- Kube-control

- primary CLI tool
- Control the k8s cluster
- All queries go thru the API server

Operations - what you want to do
Resources - what you want to do it to
Output - output format

Kubectl Operations

- apply/create - create resources
- Run - start a pod from an image
- Explain - documentation of resources
- Delete - delete resource
- Get - list resources
- Describe- detailed resource info
- Exec - execute a command on a container
- Logs - view logs on a container

Kubectl Resource

- Nodes (no)
- Podes (po)
- services(svc)
- … many more

Kubectl Output

- Wide - output additional info to stdout
- Yaml - YAML format API Object
- Json - JSON formatted API object

Kubectl cheatsheet

Application Deployment

- Imperative - manually one command at a time via CLI

```
kubectl create deployment ngnix --image=ngnix
kubectl run ngnix --image=ngnix
```

- Declarative : define desired state in code

  Manifest - YAML JSON

```
kubectl apply -f deployment.yaml
```

Basic Manifest - Pod

```
apiVersion: v1
Kind: pod # kind of object
metadata:
  name: ngnix-pod #name of object
spec:
  containers:
   -  name: nginx
   -  image: nginx
```

Deploying Resources:
Pods
Deployments
Services

```
root@aend:/home/pi# kubectl run hello-world
--image=index.docker.io/alabemhar/helloapp:v7
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be
removed in a future version. Use kubectl run --generator=run-pod/v1 or
kubectl create instead.
deployment.apps/hello-world created
root@aend:/home/pi# kubectl run hello-world-pod
--image=index.docker.io/alabemhar/helloapp:v7 --generator=run-pod/v1
pod/hello-world-pod created
root@aend:/home/pi#
```

```
root@aend:/home/pi# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
hello-world-84f64c54f9-6cq6r   1/1     Running   0          119s
hello-world-pod                1/1     Running   0          52s
root@aend:/home/pi# kubectl get deployments
```

```
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-world   1/1     1            1           2m8s
root@aend:/home/pi# kubectl get deployments -o wide
NAME          READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS    IMAGES
SELECTOR
hello-world   1/1     1            1           2m19s   hello-world
index.docker.io/alabemhar/helloapp:v7   run=hello-world
root@aend:/home/pi#
```

```
root@aend:/home/pi# kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP
NODE    NOMINATED NODE   READINESS GATES
hello-world-84f64c54f9-6cq6r   1/1     Running   0          17m
10.244.2.9    hulet   <none>           <none>
hello-world-pod                1/1     Running   0          16m
10.244.1.12   arat    <none>           <none>
root@aend:/home/pi#
```

Check what is created in “hulet”

```
root@hulet:/home/pi# docker ps -a
CONTAINER ID        IMAGE                     COMMAND
CREATED             STATUS                    PORTS               NAMES
607506675b0b        alabemhar/helloapp        "./main"                 22
minutes ago      Up 22 minutes
k8s_hello-world_hello-world-84f64c54f9-6cq6r_default_dc6b928d-9f2e-11e9-9fc
d-b827eb748129_0
50f823ff7385        k8s.gcr.io/pause:3.1      "/pause"                 22
minutes ago      Up 22 minutes
k8s_POD_hello-world-84f64c54f9-6cq6r_default_dc6b928d-9f2e-11e9-9fcd-b827eb
748129_0
89ea3c62c75f        ef3b5d63729b              "/opt/bin/flanneld -..."   2
days ago          Up 2 days
k8s_kube-flannel_kube-flannel-ds-arm-wfsj2_kube-system_78890fc3-9d45-11e9-9
ac1-b827eb748129_4
113a9b002b30        k8s.gcr.io/kube-proxy     "/usr/local/bin/kube..."   2
days ago          Up 2 days
k8s_kube-proxy_kube-proxy-cr9tm_kube-system_78869c36-9d45-11e9-9ac1-b827eb7
48129_0
a5fe9716e254        ef3b5d63729b              "/opt/bin/flanneld -..."   2
```

```
days ago          Exited (1) 2 days ago
k8s_kube-flannel_kube-flannel-ds-arm-wfsj2_kube-system_78890fc3-9d45-11e9-9
ac1-b827eb748129_3
79a409117757        quay.io/coreos/flannel   "cp -f /etc/kube-fla..."   2
days ago          Exited (0) 2 days ago
k8s_install-cni_kube-flannel-ds-arm-wfsj2_kube-system_78890fc3-9d45-11e9-9a
c1-b827eb748129_0
10a6eb286432        k8s.gcr.io/pause:3.1     "/pause"                   2
days ago          Up 2 days
k8s_POD_kube-proxy-cr9tm_kube-system_78869c36-9d45-11e9-9ac1-b827eb748129_0
d6365d39cdf8        k8s.gcr.io/pause:3.1     "/pause"                   2
days ago          Up 2 days
k8s_POD_kube-flannel-ds-arm-wfsj2_kube-system_78890fc3-9d45-11e9-9ac1-b827e
b748129_0
root@hulet:/home/pi#
```

Checking logs for a pod

```
root@aend:/home/pi# kubectl logs hello-world-pod
```

Launch a shell into a container using kubectl

```
root@aend:/home/pi# kubectl exec -it hello-world-pod -- /bin/bash
bash-5.0# hostname
hello-world-pod
bash-5.0# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1450 qdisc
noqueue state UP
    link/ether 86:36:cc:dd:c2:21 brd ff:ff:ff:ff:ff:ff
    inet 10.244.3.8/24 scope global eth0
       valid_lft forever preferred_lft forever
bash-5.0# exit
exit
root@aend:/home/pi#
```

Deployments are made up of replica sets

Replica sets are made up of pods (number of pods)

```
root@aend:/home/pi# kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-world   1/1     1            1           60m
root@aend:/home/pi# kubectl get deployment hello-world
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-world   1/1     1            1           60m
root@aend:/home/pi# kubectl get replicaset
NAME                     DESIRED   CURRENT   READY   AGE
hello-world-84f64c54f9   1         1         1       60m
root@aend:/home/pi# kubectl get po
NAME                           READY   STATUS    RESTARTS   AGE
hello-world-84f64c54f9-6cq6r   1/1     Running   0          60m
hello-world-pod                1/1     Running   0          16m
root@aend:/home/pi#
```

Closer look at deployment

```
root@aend:/home/pi# kubectl describe deployment hello-world | more
```

Closer look at pod

```
root@aend:/home/pi# kubectl describe  pod hello-world-84f64c54f9-6cq6r |
more
```

Expose deployment as as Service with persistent IP Address

```
root@aend:/home/pi# kubectl expose deployment hello-world --port=80
--target-port=8080
service/hello-world exposed
root@aend:/home/pi# kubectl get service hello-world
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
hello-world   ClusterIP   10.111.240.215   <none>        80/TCP    28s
root@aend:/home/pi#
```

Closer look at Service

```
root@aend:/home/pi# kubectl describe service hello-world
```

```
Name:              hello-world
Namespace:         default
Labels:            run=hello-world
Annotations:       <none>
Selector:          run=hello-world
Type:              ClusterIP
IP:                10.111.240.215
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.244.2.9:8080
Session Affinity:  None
Events:            <none>
root@aend:/home/pi#
```

Test Access to Service

```
root@aend:/home/pi# curl http://10.111.240.215:80
ሰላም ! እኔ  hello-world-84f64c54f9-6cq6r ነኝ
root@aend:/home/pi#
```

To access pod application directly , for troubleshooting …

```
root@aend:/home/pi# kubectl get endpoints hello-world
NAME          ENDPOINTS         AGE
hello-world   10.244.2.9:8080   23m
root@aend:/home/pi#
```

```
root@aend:/home/pi# curl http://10.244.2.9:8080
ሰላም ! እኔ  hello-world-84f64c54f9-6cq6r ነኝ
root@aend:/home/pi#
```

We can use kubectl to output YAML, which we can use for seeing other resources using declarative resource creation (manifest files)

To get the YAML that represents hello-world service

```
root@aend:/home/pi# kubectl get service hello-world -o yaml
apiVersion: v1
kind: Service
```

```
metadata:
  creationTimestamp: "2019-07-05T15:53:42Z"
  labels:
    run: hello-world
  name: hello-world
  namespace: default
  resourceVersion: "343783"
  selfLink: /api/v1/namespaces/default/services/hello-world
  uid: 0fee4a92-9f3d-11e9-9fcd-b827eb748129
spec:
  clusterIP: 10.111.240.215
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    run: hello-world
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
root@aend:/home/pi#
```

--export strip off cluster-specific information

Files representing the service and deployment

```
root@aend:/home/pi# kubectl get service hello-world -o yaml --export >
service-hello-world.yaml
root@aend:/home/pi# kubectl get deployment hello-world -o yaml --export >
deployment-hello-world.yaml
```

To get detailed description/documentation of API objects we can use the kubectl explain command

```
root@aend:/home/pi# kubectl explain service.spec | more
root@aend:/home/pi# kubectl explain service.spec.ports | more
root@aend:/home/pi# kubectl explain service.spec.ports.targetPort | more
```

To declaratively re-create all the resources … delete all

```
root@aend:/home/pi# kubectl delete service hello-world
root@aend:/home/pi# kubectl delete deployment hello-world
root@aend:/home/pi# kubectl delete pod hello-world-pod
root@aend:/home/pi# kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   2d13h
root@aend:/home/pi#
```

Now we can deploy the resource declaratively

```
root@aend:/home/pi# kubectl apply -f deployment-hello-world.yaml
deployment.extensions/hello-world created
root@aend:/home/pi# kubectl apply -f service-hello-world.yaml
service/hello-world created
root@aend:/home/pi# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/hello-world-84f64c54f9-lmqwr    1/1     Running   0          14s

NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/hello-world   ClusterIP   10.103.26.4   <none>        80/TCP    3s
service/kubernetes    ClusterIP   10.96.0.1     <none>        443/TCP
2d13h

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hello-world   1/1     1            1           14s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/hello-world-84f64c54f9   1         1         1       14s
root@aend:/home/pi#
```

Lets Scale up our deployment by increasing the number of replicas

EDIT deployment-hello-world.yaml and change replicas from 1 to 3

Apply the deployment change

```
root@aend:/home/pi# kubectl apply -f deployment-hello-world.yaml
deployment.extensions/hello-world configured
root@aend:/home/pi# kubectl get deployment hello-world
```

```
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-world   3/3     3            3           5m15s
root@aend:/home/pi#
```

Note the change in the service endpoints

```
root@aend:/home/pi# kubectl describe service hello-world
Name:              hello-world
Namespace:         default
Labels:            run=hello-world
Annotations:       kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"creationT
imestamp":null,"labels":{"run":"hello-world"},"name":"hello-wor...
Selector:          run=hello-world
Type:              ClusterIP
IP:                10.103.26.4
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.244.1.13:8080,10.244.2.10:8080,10.244.3.9:8080
Session Affinity:  None
Events:            <none>
root@aend:/home/pi#
```

```
root@aend:/home/pi# kubectl get endpoints hello-world
NAME          ENDPOINTS                                            AGE
hello-world   10.244.1.13:8080,10.244.2.10:8080,10.244.3.9:8080   12m
root@aend:/home/pi#
```

Verify the loadbalancing effect by using curl repeatedly

```
root@aend:/home/pi# kubectl get service hello-world
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
hello-world   ClusterIP   10.103.26.4   <none>        80/TCP    11m
root@aend:/home/pi# curl http://10.103.26.4:80
ሰላም ! እኔ  hello-world-84f64c54f9-shlqc ነኝ
root@aend:/home/pi# curl http://10.103.26.4:80
ሰላም ! እኔ  hello-world-84f64c54f9-28szq ነኝ
root@aend:/home/pi# curl http://10.103.26.4:80
ሰላም ! እኔ  hello-world-84f64c54f9-shlqc ነኝ
root@aend:/home/pi# curl http://10.103.26.4:80
ሰላም ! እኔ  hello-world-84f64c54f9-shlqc ነኝ
root@aend:/home/pi# curl http://10.103.26.4:80
```

```
ሰላም ! እኔ  hello-world-84f64c54f9-lmqwr ነኝ
root@aend:/home/pi#
```

To change deployment configuration on the fly: updates sent to API server immediately

```
root@aend:/home/pi# kubectl edit deployment hello-world
```

Exposing Service to External IP address

- Lets first delete the existing service
- And recreate it as follows

```
root@aend:/home/pi# kubectl delete service hello-world
root@aend:/home/pi# kubectl expose deployment hello-world
--type=LoadBalancer --port=80 --target-port=8080 --name=hello-world
root@aend:/home/pi# kubectl get service ceservice -o yaml --export
Flag --export has been deprecated, This flag is deprecated and will be
removed in future.
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    run: hello-world
  name: ceservice
  selfLink: /api/v1/namespaces/default/services/ceservice
spec:
  externalIPs:
  - 192.168.1.25
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 31672
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    run: hello-world
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer: {}
root@aend:/home/pi# kubectl get service ceservice -o yaml --export >
```

```
updated-hello-world-service.yaml
Flag --export has been deprecated, This flag is deprecated and will be
removed in future.
root@aend:/home/pi#
```

Note : the differences in the manifest file

```
root@aend:/home/pi# kubectl get service ceservice -o yaml --export >
updated-hello-world-service.yaml
Flag --export has been deprecated, This flag is deprecated and will be
removed in future.
root@aend:/home/pi# kubectl get service ceservice
NAME        TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
ceservice   LoadBalancer   10.96.79.84   192.168.1.25   80:31672/TCP   21m
root@aend:/home/pi#
```

Service can now be accessed at : http://192.168.1.25

- Demonstrate rolling upgrade : update docker image version

```
root@aend:/home/pi# kubectl edit deployment hello-world
```

RPI K8S Cluster

1 - static IP and hostname

```
root@raspberrypi:/home/pi# cat /etc/dhcpcd.conf
#static IP configuration

interface enxb827eb748129
static ip_address=192.168.1.25/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

root@raspberrypi:/home/pi#
```

```
pi@arat:~ $ sudo vi /etc/hostname
```

2 - disable bt and wifi

```
root@raspberrypi:/home/pi# echo "dtoverlay=pi3-disable-wifi" | sudo tee -a
/boot/config.txt
root@raspberrypi:/home/pi# echo "dtoverlay=pi3-disable-bt" | sudo tee -a
/boot/config.txt
root@raspberrypi:/home/pi# systemctl disable hciuart
root@raspberrypi:/home/pi# sudo reboot
```

3 - update apt

```
root@raspberrypi:/home/pi# apt-get update
```

4 - Install ansible on the first node

```
root@raspberrypi:/home/pi# apt-get install ansible
root@raspberrypi:/home/pi# ansible --version
ansible 2.2.1.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = Default w/o overrides
root@raspberrypi:/home/pi#
```

5 - SSH key exchange for password-less login

```
pi@raspberrypi:~ $ ssh-keygen
```

And create ~/.ssh/authorized_keys

6 - Update /etc/hosts

7 - Reset kubeadm when playbook fails

```
kubeadm reset
```

8 - proxy for dashboard access

```
root@aend:/home/pi# kubectl proxy --address <MASTER_IP> --port=8001
```

```
--accept-hosts='^*$'
```

9 - ssh tunnel to access dashboard

```
ssh -L 8001:localhost:8001 pi@192.168.1.25
```

We can not login with kubeconfig since certs are not supposed to leave the master
Use -> token

10 - Generate admin user account

Create the following file : dashboard-adminuser.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system
```

Create admin user :

```
kubectl apply -f dashboard-adminuser.yaml
```

Create the following file : dashboard-adminuser.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kube-system
```

Create the clusterRole binding:

```
kubectl apply -f dashboard-adminuser.yaml
```

11- Get the dashboard access token

```
root@aend:/home/pi# kubectl -n kube-system describe secret $(kubectl -n
kube-system get secret | grep admin-user | awk '{print $1}')
Name:         admin-user-token-2xqsc
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: admin-user
              kubernetes.io/service-account.uid:
d5b5cb6c-9e52-11e9-9fcd-b827eb748129

Type:  kubernetes.io/service-account-token

Data
====
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW
50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3Rlb
SIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2Vy
LXRva2VuLTJ4cXNjIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY29
1bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3Nlcn
ZpY2UtYWNjb3VudC51aWQiOiJkNWI1Y2I2Yy05ZTUyLTExZTktOWZjZC1iODI3ZWI3NDgxMjkiL
CJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.m7d
Cdi3w8REHpEq2jvaPIOeSbKizKbcV95Ol6xszh1H4lSLOtE64PBoBafJCx2eDjs1rsUdM9qpyYm
wAsVlZFtT5fm6QCo1EpKYJ0APd-ejqn6fyYOpx2ffvnlUEWtw0ITM6BtC8FayXXNijhuW57anRH
_jddd_fk_CiAR9nhmOSfF1plc79K-JO1_lYveBVMpMuy-U5VfV0uZENEnfRu65c_svW-peVZsob
Fec8GC_AYZ3Ya4zVrWv91FWaAlC9xa-1PDYpNnvGzJ2oWGEMhmeZcblhlS2uV13OAN_cH-gEU5T
zu6T0bkpI-t1wV6v2v79qdLsoPwhRYGeNv2B0fw
ca.crt:     1025 bytes
namespace:  11 bytes
root@aend:/home/pi#
```

Access Dashboard :
http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/overview?namespace=default

References :

https://kubernetes.io/docs/tutorials/kubernetes-basics/
https://github.com/rak8s/rak8s
https://evalle.xyz/posts/setting-up-a-kubernetes-1-14-raspberry-pi-cluster-using-kubeadm/
https://github.com/kubernetes/dashboard/wiki/Creating-sample-user
https://kubernetes.io/docs/reference/kubectl/cheatsheet/
https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/
https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/#create-a-secret-in-the-cluster-that-holds-your-authorization-token