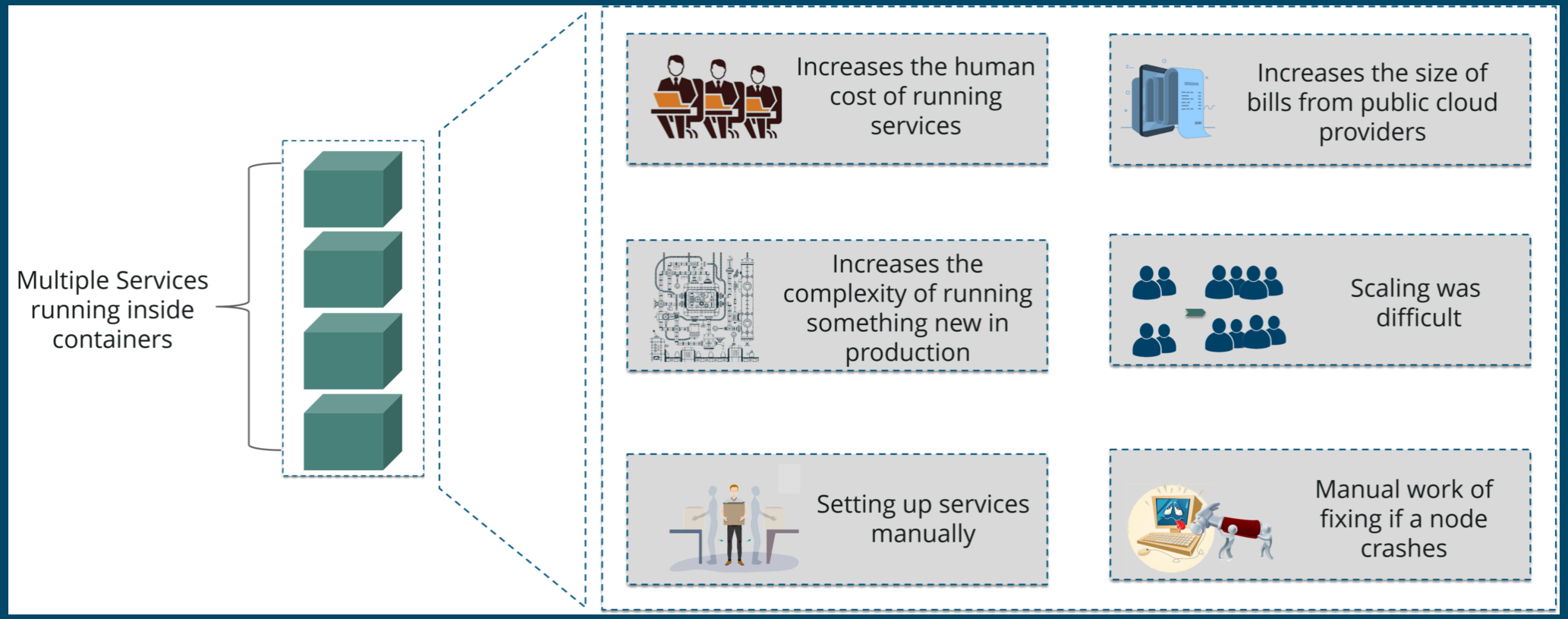
Kubernetes

Fundamentals for big data analysis

Before Kubernetes

- Manual management of cloud applications is a problem



Kubernetes Features

01

Automated Scheduling

Kubernetes provides advanced scheduler to launch container on cluster nodes

02

Self Healing Capabilities

Rescheduling, replacing and restarting the containers which are died.

03

Automated rollouts and rollback

Kubernetes supports rollouts and rollbacks for the desired state of the containerized application

04

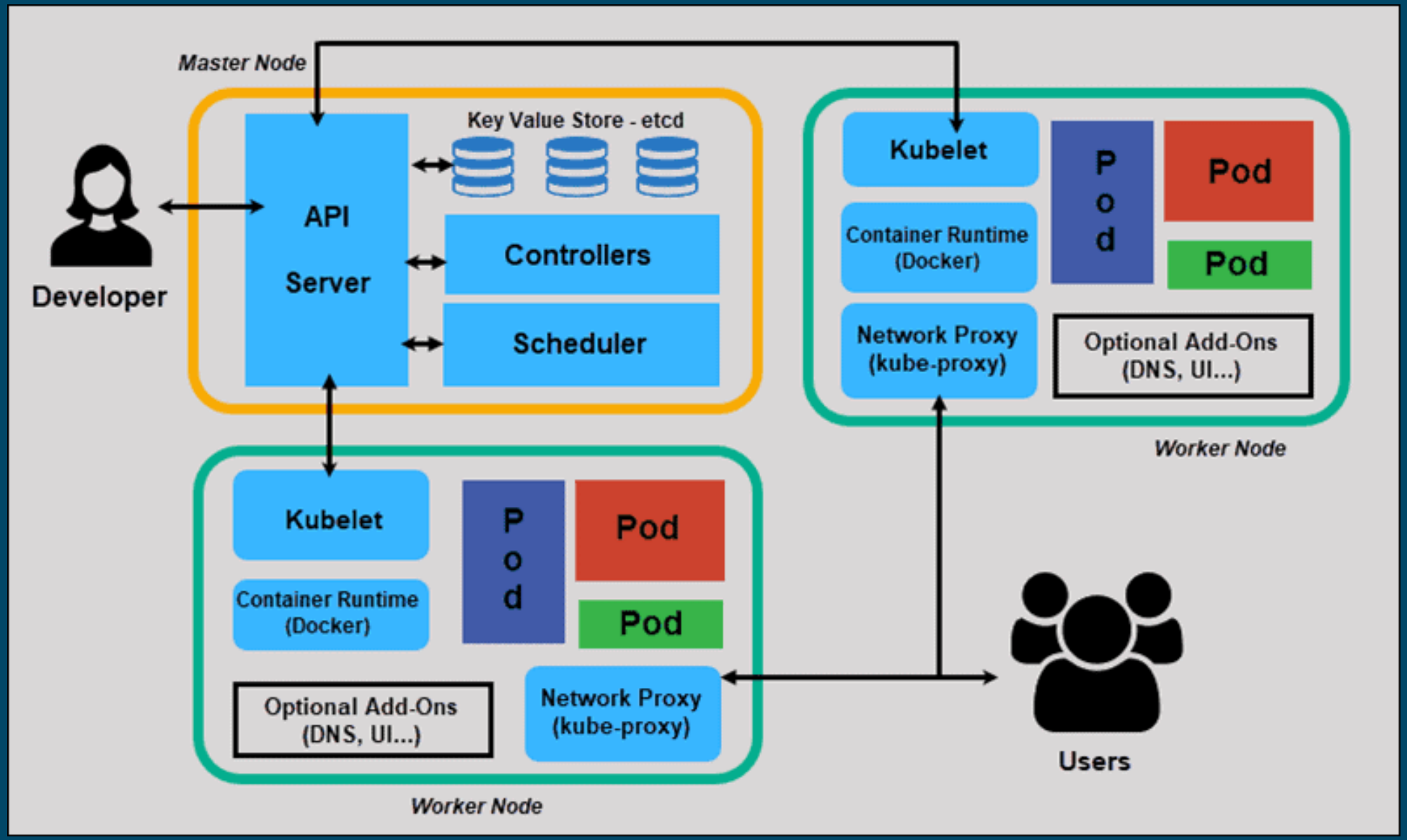
Horizontal Scaling and Load Balancing

Kubernetes can scale up and scale down the application as per the requirements

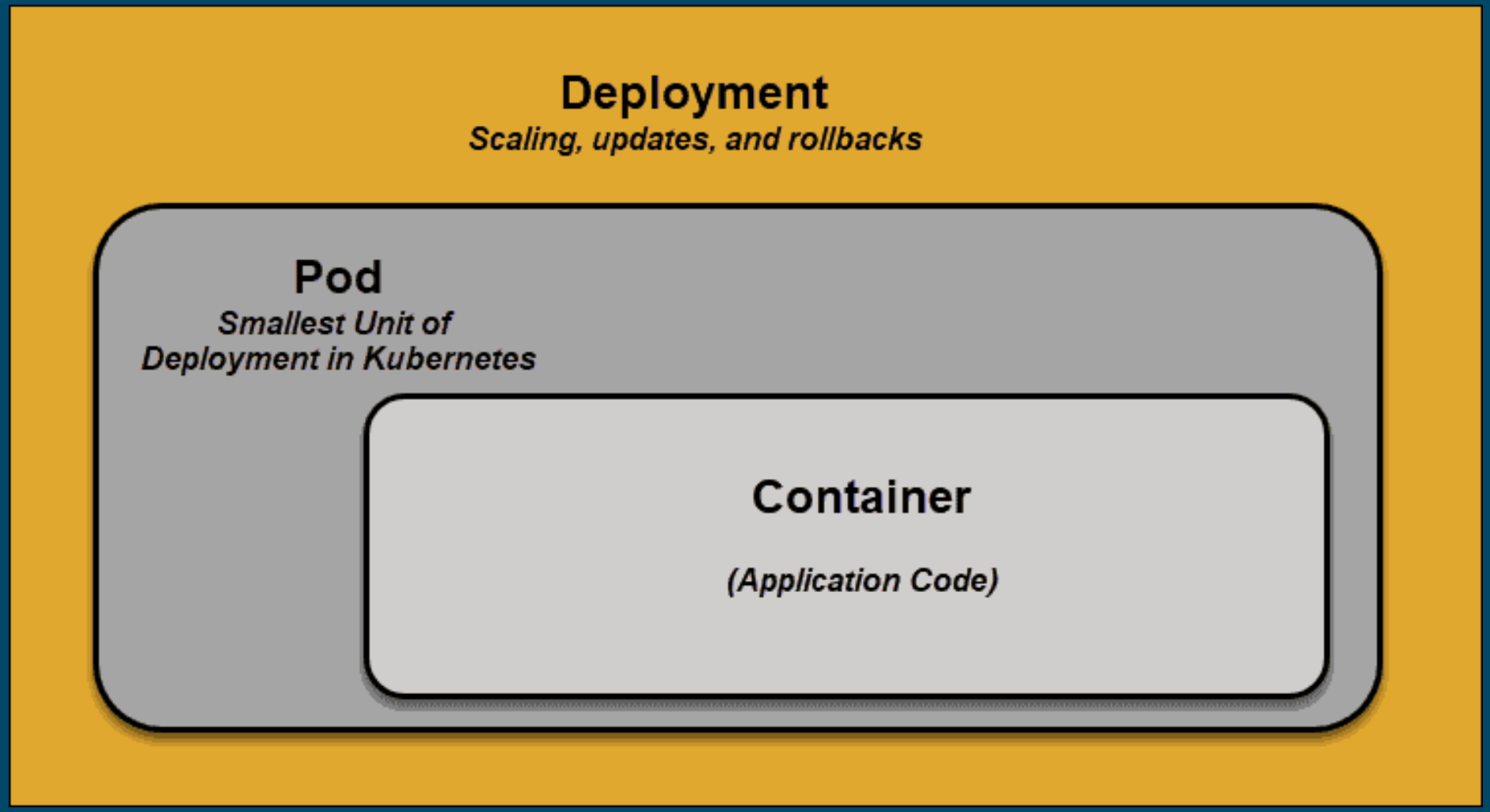
What is Kubernetes?

- Kubernetes, or k8s for short, is a system for automating application deployment.
- Modern applications are dispersed across clouds, virtual machines, and servers.
- Administering apps manually is no longer a viable option.
- K8s transforms virtual and physical machines into a unified API surface.
- A developer can then use the Kubernetes API to deploy, scale, and manage containerized applications.

Kubernetes Architecture



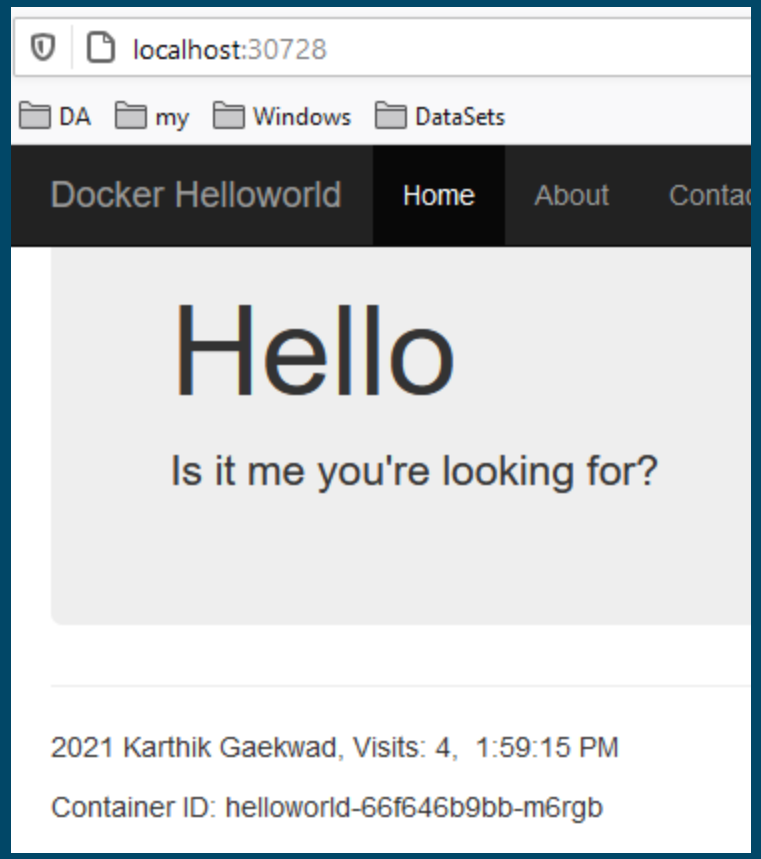
Deployment


Deployment
Scaling, updates, and rollbacks
Pod
Smallest Unit of
Deployment in Kubernetes
Container
(Application Code)

Deploy an image to cluster using YAML

Hello World web app

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld
spec:
  selector:
    matchLabels:
      app: helloworld
  replicas: 1 # tells deployment to run 1 pods matching the template
  template: # create pods using pod definition in this template
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
      - name: helloworld
        image: karthequian/helloworld:latest
        ports:
        - containerPort: 80
```



Labs from: https://www.linkedin.com/learning/learning-kubernetes ; https://github.com/karthequian/Kubernetes

Hello World – lab steps
Deploy a web app from an image to cluster

- Directory 03_04

```
kubectl get all
kubectl create -f helloworld.yaml
kubectl expose deployment helloworld --type=NodePort
```

- Wait…
  - kubectl describe services helloworld
  - Open browser to NodePort
- Review the files
  - helloworld.yaml
  - https://github.com/karthequian/docker-helloworld

```
Name:                     helloworld
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=helloworld
Type:                     NodePort
IP:                       10.108.143.41
LoadBalancer Ingress:     localhost
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30728/TCP
Endpoints:                10.1.0.113:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

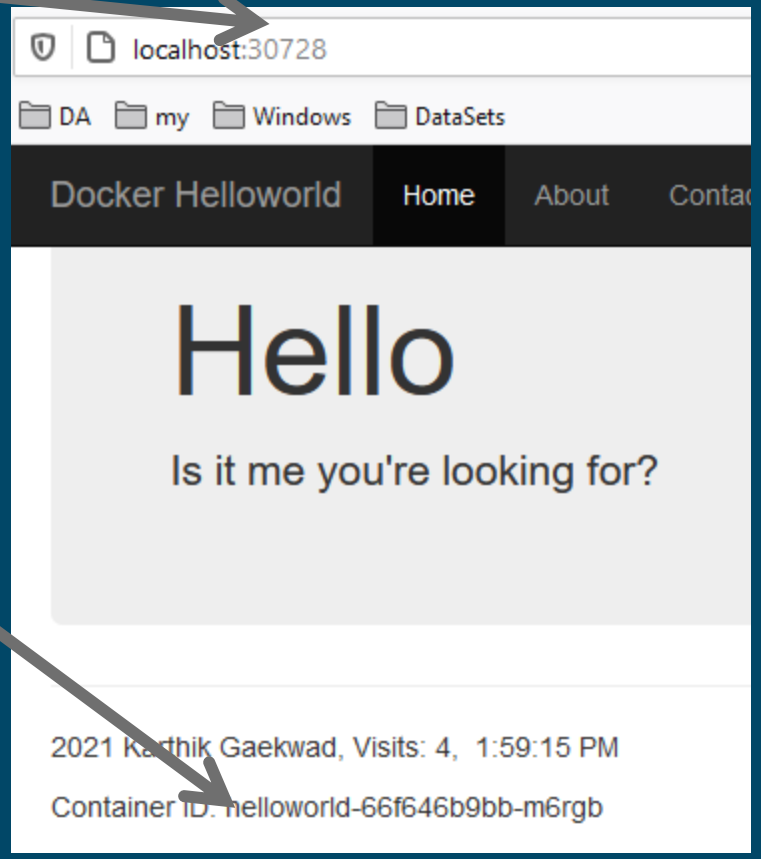
Hello Word
Pod is a web app

```
NAME                                READY   STATUS    RESTARTS   AGE
pod/helloworld-66f646b9bb-m6rgb     1/1     Running   0          170m

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/helloworld    NodePort    10.108.143.41   <none>        80:30728/TCP   150m
service/kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP        6d

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/helloworld    1/1     1            1           170m

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/helloworld-66f646b9bb    1         1         1       170m
```



Hello World – Deleting deployment

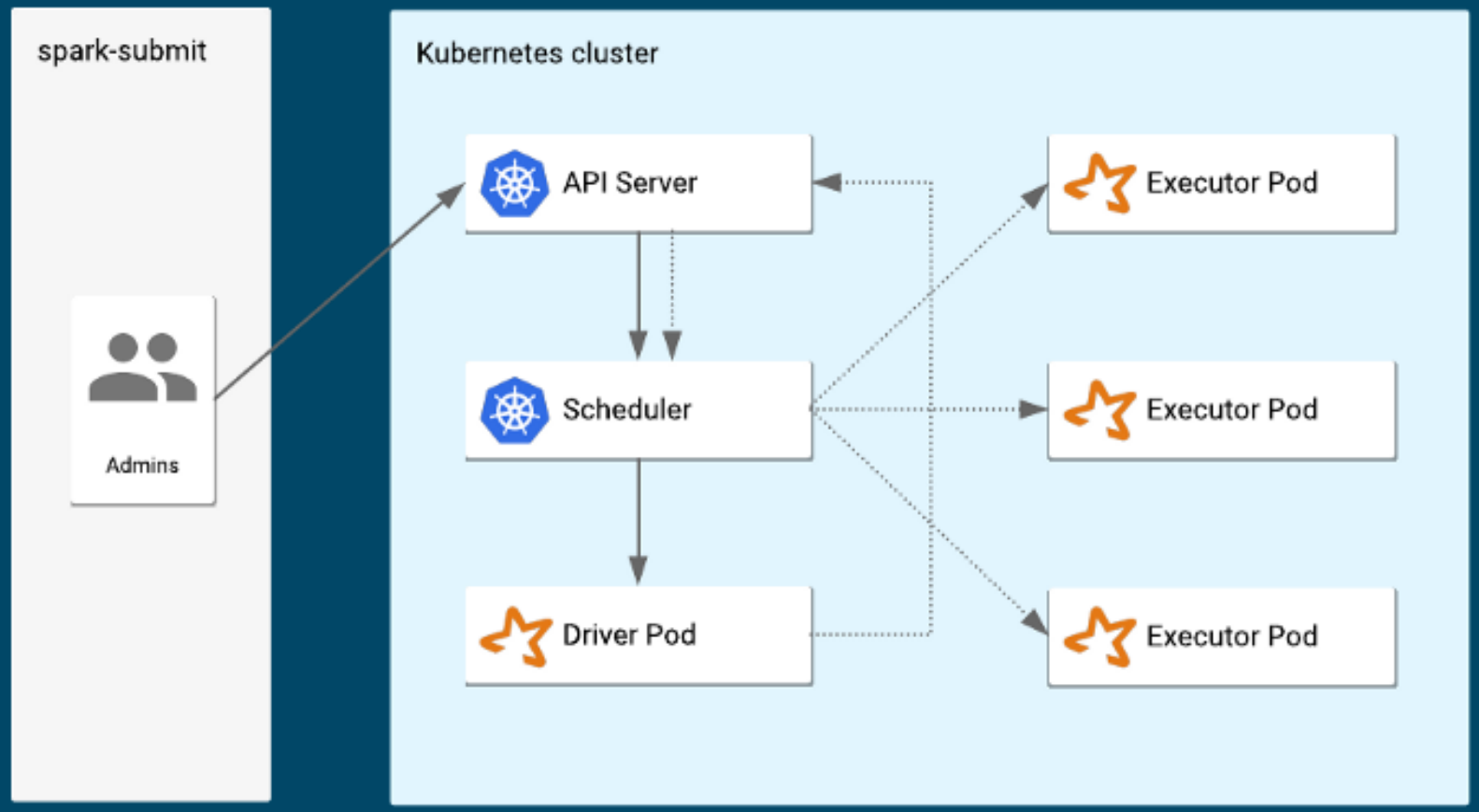
- kubectl delete -f helloworld.yaml

Deploy an image to cluster using Spark submit

spark_submit

- Spark contains a cluster manager than runs spark jobs over Kubernetes clusters
- spark_submit provides the means to run a spark program from a Docker image over a Kubernetes cluster

spark_submit



https://medium.com/@toddrosner/spark-on-kubernetes-b18f16ac95ad

```bash
#!/bin/bash

# Run on Docker for Windows with data and source in Docker image

# Spark on Kubernetes doesn't support submitting locally stored files with spark-submit
# This means many submit parameters will not work unless the path is http, including py-files, files, archives
# * https://stackoverflow.com/questions/61711673/how-to-submit-pyspark-job-on-kubernetes-minikube-using-spark-submit
# Rely on run_env.py to load and run modules, which must be built into the Docker image (or mounted)

# Local spark to run spark-submit
# TODO: Check that this is the path to your spark.
# TODO: If running in WSL2 Linux, then ensure that this is the path to spark install in WSL2 Linux
SPARK_HOME=/opt/spark

# URL of Kubernetes master on Docker desktop
MASTER=k8s://https://localhost:6443

# Module here is source zip (not egg file). Must create a zip file from the source
PYSPARK_APP_MODULE=airbnb

# Image execution, values may be overridden with Kubernetes: spark.kubernetes.driver.limit.cores, spark.kubernetes.dri
EXECUTORS=1
EXECUTOR_MEMORY=2g
DRIVER_MEMORY=1g

# Image which includes run_env.py in working directory
# TODO: Ensure that this is the name of the image you built
IMAGE=wrobinson/airbnb:1.0

# working directory on image (default set by spark image creation, docker-image-tool.sh)
# https://levelup.gitconnected.com/spark-on-kubernetes-3d822969f85b
WORKING_DIR=/opt/spark/work-dir

SCRIPT=local://${WORKING_DIR}/run_env.py  # Run your module
#SCRIPT=local://${WORKING_DIR}/sleep.py   # For debugging (e.g., checking the mount)
# Shell into container; kubectl exec <CONTAINER ID> -ti /bin/bash

SPARK_CMD="$SPARK_HOME/bin/spark-submit \
  --master ${MASTER} \
  --deploy-mode cluster \
  --driver-memory ${DRIVER_MEMORY}
  --executor-memory ${EXECUTOR_MEMORY} \
  --conf spark.executor.instances=${EXECUTORS} \
  --conf spark.dynamicAllocation.enabled=false \
  --conf spark.kubernetes.container.image=${IMAGE} \
  --name ${PYSPARK_APP_MODULE} \
  --conf spark.kubernetes.executor.label.app=${PYSPARK_APP_MODULE} \
  --conf spark.kubernetes.driver.label.app=${PYSPARK_APP_MODULE} \
  --conf spark.kubernetes.driverEnv.PYSPARK_MAJOR_PYTHON_VERSION=3 \
  --conf spark.kubernetes.driverEnv.PYSPARK_APP_MODULE=${PYSPARK_APP_MODULE} \
  ${SCRIPT} \
  --master=${MASTER} --py_files=${WORKING_DIR}/${PYSPARK_APP_MODULE}.zip"

echo ${SPARK_CMD}
echo
eval ${SPARK_CMD}
```

```bash
#!/bin/bash

# Run on Docker for Windows with data and source in Docker image

# Spark on Kubernetes doesn't support submitting locally stored files with spark-s
# This means many submit parameters will not work unless the path is http, includi
# * https://stackoverflow.com/questions/61711673/how-to-submit-pyspark-job-on-kubernetes-minikube-using-spark-submit
# Rely on run_env.py to load and run modules, which must be built into the Docker image (or mounted)

# Local spark to run spark-submit
# TODO: Check that this is the path to your spark.
# TODO: If running in WSL2 Linux, then ensure that this is the path to spark install in WSL2 Linux
SPARK_HOME=/opt/spark

# URL of Kubernetes master on Docker desktop
MASTER=k8s://https://localhost:6443

# Module here is source zip (not egg file). Must create a zip file from the source
PYSPARK_APP_MODULE=airbnb
```

Path to local install of Spark

Path to Kubernetes server

Name of our Python module
Place it in environment variable (of the image)
run_env.py Script will read this name to start the module

```
21  # Image execution, values may be overridden with Kubernetes: spark.kubernetes.driver.limit.cores, spark.kubernetes.dr
22  EXECUTORS=1
23  EXECUTOR_MEMORY=2g
24  DRIVER_MEMORY=1g
25
26  # Image which includes run_env.py in working directory
27  # TODO: Ensure that this is the name of the image you built
28  IMAGE=wrobinson/airbnb:1.0
29
30  # working directory on image (default set by spark image creation, docker-image-too
31  # https://levelup.gitconnected.com/spark-on-kubernetes-3d822969f85b
32  WORKING_DIR=/opt/spark/work-dir
33
34  SCRIPT=local://${WORKING_DIR}/run_env.py  # Run your module
35  #SCRIPT=local://${WORKING_DIR}/sleep.py   # For debugging (e.g., checking the mount)
36  # Shell into container; kubectl exec <CONTAINER ID> -ti /bin/bash
37
```

Parameters for a Kubernetes cluster

Image to run
(same on all nodes)

Spark working directory (on image)

Script to run to start our program

Spark Submit
spark knows how to start image in Kubernetes

```
SPARK_CMD="$SPARK_HOME/bin/spark-submit \
  --master ${MASTER} \
  --deploy-mode cluster \
  --driver-memory ${DRIVER_MEMORY}
  --executor-memory ${EXECUTOR_MEMORY} \
  --conf spark.executor.instances=${EXECUTORS} \
  --conf spark.dynamicAllocation.enabled=false \
  --conf spark.kubernetes.container.image=${IMAGE} \
  --name ${PYSPARK_APP_MODULE} \
  --conf spark.kubernetes.executor.label.app=${PYSPARK_APP_MODULE} \
  --conf spark.kubernetes.driver.label.app=${PYSPARK_APP_MODULE} \
  --conf spark.kubernetes.driverEnv.PYSPARK_MAJOR_PYTHON_VERSION=3 \
  --conf spark.kubernetes.driverEnv.PYSPARK_APP_MODULE=${PYSPARK_APP_MODULE} \
  ${SCRIPT} \
  --master=${MASTER} --py_files=${WORKING_DIR}/${PYSPARK_APP_MODULE}.zip"
```

Parameters defined above

```
SCRIPT=local://${WORKING_DIR}/run_env.py
```

Notice we'll start the program with the ${SCRIPT}

py_files points to our program, which is local to the image

run_env.py
run a module specified in environment var

```
25  def main():
26      logger = logging.getLogger("run_env")
27      print("run_env executing in {}".format(os.getcwd()))
28      print(" the python sys path is: {}".format(sys.path))
29      for f in listdir("."):
30          print("file/folder in this directory: {}".format(f))
31      env_key = "PYSPARK_APP_MODULE"
32      print("Loading module specified in environment variable {}.".format(env_key))
33      module_name = None
34      try:
35          module_name = os.environ.get(env_key)
36      except KeyError:
37          logger.error("No environment variable {} specified. Cannot load module.".format(env_key))
38      if module_name:
39          module = importlib.import_module(module_name)
40          print("Imported module {}. Calling main".format(module_name))
41          module.main()
42          print("Done in main. Exiting.".format(module_name))
```

Get module name from ENV

Run the module main()

Common Kubernetes commands

- kubectl
  - get all
  - get pods --all-namespaces
  - describe pod POD_NAME
  - logs
  - logs –l app=airbnb
  - delete pod/POD_NAME
  - config get-contexts
  - config use-context docker-desktop

Kubernetes cheat sheet

COMMANDS	FUNCTION
Kubectl get pods	Lists all current pods
Kubectl describe pod<name>	Describes the pod names
Kubectl get rc	List all replication controllers
Kubectl get rc -- namespace="namespace"	Lists replication controllers in namespace
Kubectl describe rc <name>	Shows the replication controller name
Kubectl get cvc	Lists the services
Kubectl describe svc<name>	Shows the service name
Kubectl delete pod<name>	Deletes the pod
Kubectl get nodes -w	Watch nodes continuously

FUNCTION	COMMAND
Execute command on service by selecting container.	Kubectl exec<service><commands>[-c< $container>]
Get logs from service for a container	Kubectl logs -f<name>>[-c< $container>]
Watch the kubelet logs	Watch -n 2 cat/var/log/kublet.log
Show metrics for node	Kubectl top node
Show metrics for pods	Kubectl top pod

Launch a pod with a name an image : Kubectl run<name> --image=<image-name>
Create a service in <manifest.yaml> : Kubectl create -f <manifest.yaml>
Scale replication counter to count the number of instances : Kubectl scale --replicas=<count>

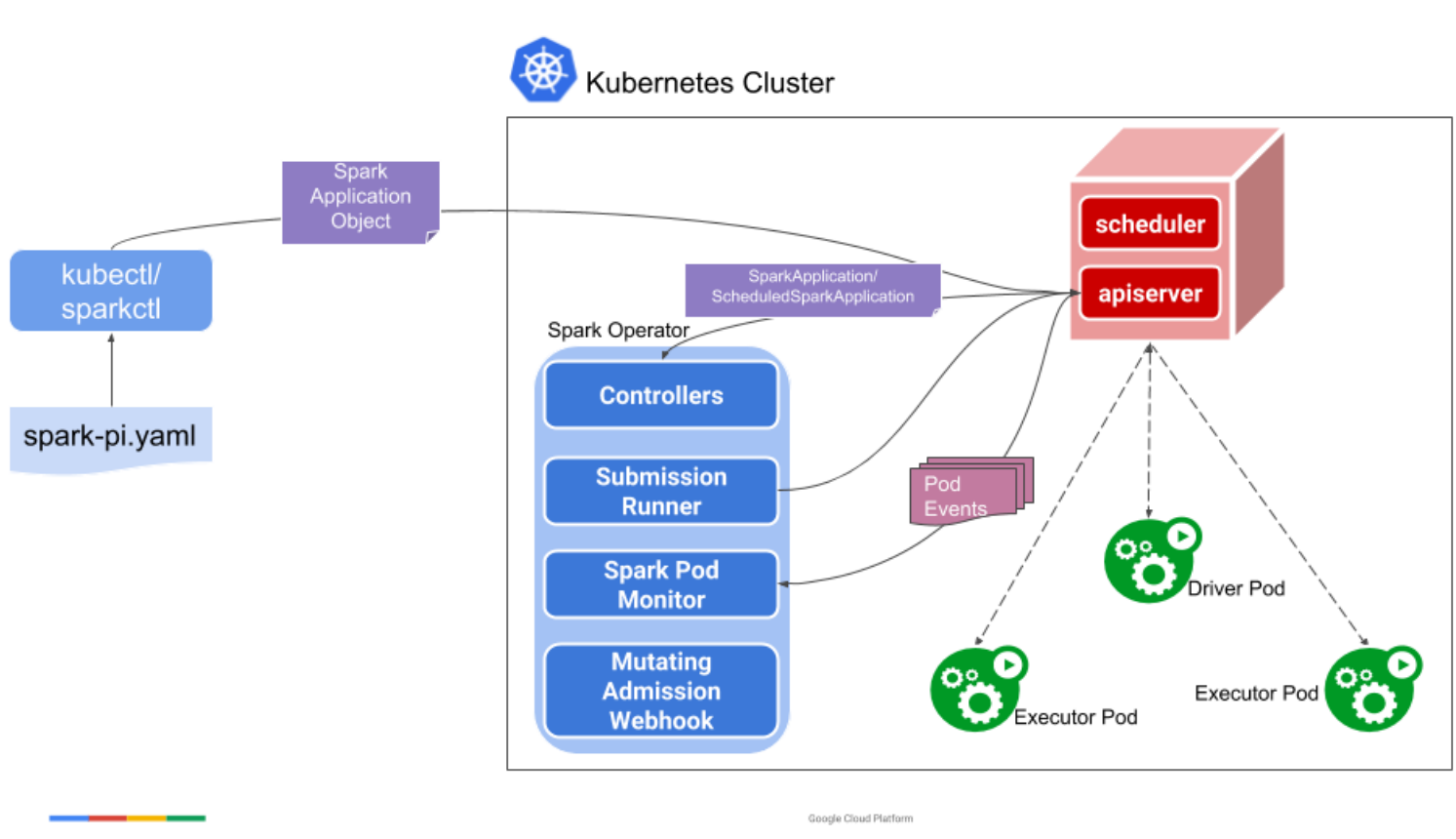
Map external port to internal replication port : Expose rc<name> --port=<external>--target-port=<internal>
To stop all pod in <n> : Kubectl drain<n>-- delete-local-data--force--ignore-daemonset
Allow master nodes to run pods : Kubectltaintnodes --all-node-role.kuernetes.io/master-

https://intellipaat.com/mediaFiles/2019/03/Kubernetes-Cheat-Sheet.pdf

Deploy an image to cluster using Spark operator

Spark operator

- Improvement over spark_submit
- Uses standard Kubernetes deployment (YAML)



Important to remember

- Kubernetes is a system for automating application deployment
  - A developer can then use the Kubernetes API to deploy, scale, and manage containerized applications
- A Docker contains runs in a pod defined by a deployment
- Spark contains a cluster manager than runs spark jobs over Kubernetes clusters
  - spark_submit provides the means to run a spark program from a Docker image over a Kubernetes cluster