



{K}ODE{K}LOUD

Rolling Updates & Rollbacks



Rollout and Versioning



Revision 1



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.0

Revision 2



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1

Rollout Command

```
> kubectl rollout status deployment/myapp-deployment
```

```
Waiting for rollout to finish: 0 of 10 updated replicas are available...  
Waiting for rollout to finish: 1 of 10 updated replicas are available...  
Waiting for rollout to finish: 2 of 10 updated replicas are available...  
Waiting for rollout to finish: 3 of 10 updated replicas are available...  
Waiting for rollout to finish: 4 of 10 updated replicas are available...  
Waiting for rollout to finish: 5 of 10 updated replicas are available...  
Waiting for rollout to finish: 6 of 10 updated replicas are available...  
Waiting for rollout to finish: 7 of 10 updated replicas are available...  
Waiting for rollout to finish: 8 of 10 updated replicas are available...  
Waiting for rollout to finish: 9 of 10 updated replicas are available...  
deployment "myapp-deployment" successfully rolled out
```

```
> kubectl rollout history deployment/myapp-deployment
```

```
deployments "myapp-deployment"
```

```
REVISION  CHANGE-CAUSE
```

```
1          <none>
```

```
2          kubectl apply --filename=deployment-definition.yml --record=true
```

Deployment Strategy

Recreate



nginx:1.7.0



nginx:1.7.0



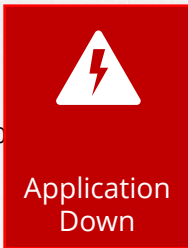
nginx:1.7.0



nginx:1.7.0



nginx:1.7.0



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



nginx:1.7.1



Rolling Update



nginx:1.7.0



nginx:1.7.1



nginx:1.7.0



nginx:1.7.1



nginx:1.7.0



nginx:1.7.1



nginx:1.7.0



nginx:1.7.1



nginx:1.7.0



nginx:1.7.1



Kubectl apply

```
> kubectl apply -f deployment-definition.yml  
deployment "myapp-deployment" configured
```

```
> kubectl set image deployment/myapp-deployment \  
    nginx=nginx:1.9.1  
deployment "myapp-deployment" image is updated
```

deployment-definition.yml

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: myapp-deployment  
  labels:  
    app: myapp  
    type: front-end  
spec:  
  template:  
    metadata:  
      name: myapp-pod  
      labels:  
        app: myapp  
        type: front-end  
    spec:  
      containers:  
      - name: nginx-container  
        image: nginx:1.7.1  
  replicas: 3  
  selector:  
    matchLabels:  
      type: front-end
```

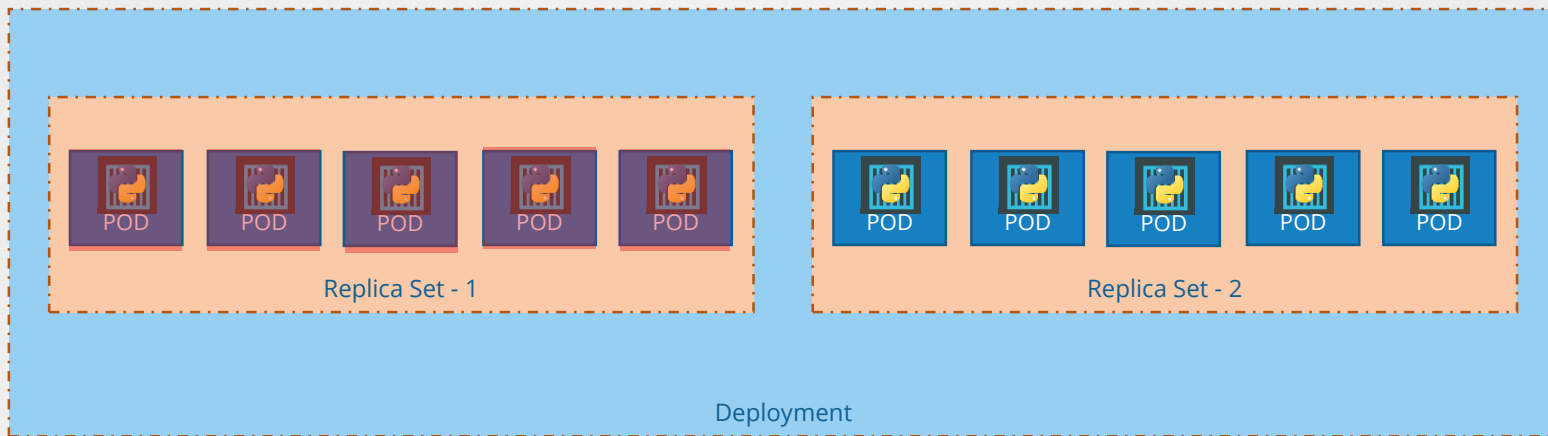
```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:          myapp-deployment
Namespace:     default
CreationTimestamp: Sat, 03 Mar 2018 17:01:55 +0800
Labels:        app=myapp
               type=front-end
Annotations:    deployment.kubernetes.io/revision=2
               kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","me
s\\Google...      kubernetes.io/change-cause=kubectl apply --filename=d:\\Mumshad Files\\Google Drive\\Udemy\\Kubernet
Selector:       type=front-end
Replicas:       5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:   Recreate
MinReadySeconds: 0
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
    nginx-container:
      Image:        nginx:1.7.1
      Port:         <none>
      Environment:  <none>
      Mounts:       <none>
      Volumes:      <none>
  Conditions:
    Type           Status    Reason
    ----           -
    Available      True     MinimumReplicasAvailable
    Progressing    True     NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-deployment-54c7d6ccc (5/5 replicas created)
Events:
  Type    Reason              Age    From          Message
  ----    -
  Normal  ScalingReplicaSet   11m    deployment-controller  Scaled up replica set myapp-deployment-6795844b58 to 5
  Normal  ScalingReplicaSet   1m     deployment-controller  Scaled down replica set myapp-deployment-6795844b58 to 0
  Normal  ScalingReplicaSet   56s    deployment-controller  Scaled up replica set myapp-deployment-54c7d6ccc to 5
```

Recreate

```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:          myapp-deployment
Namespace:     default
CreationTimestamp: Sat, 03 Mar 2018 17:16:53 +0800
Labels:        app=myapp
               type=front-end
Annotations:    deployment.kubernetes.io/revision=2
               kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metadat
Files\\Google...      kubernetes.io/change-cause=kubectl apply --filename=d:\\Mumshad Files\\Google Drive\\Udemy\\Kubernets\\De
Selector:       type=front-end
Replicas:       5 desired | 5 updated | 6 total | 4 available | 2 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
    nginx-container:
      Image:        nginx
      Port:         <none>
      Environment:  <none>
      Mounts:       <none>
      Volumes:      <none>
  Conditions:
    Type           Status    Reason
    ----           -
    Available      True     MinimumReplicasAvailable
    Progressing    True     ReplicaSetUpdated
OldReplicaSets:   myapp-deployment-67c749c58c (1/1 replicas created)
NewReplicaSet:    myapp-deployment-7d57dbdb8d (5/5 replicas created)
Events:
  Type    Reason              Age    From          Message
  ----    -
  Normal  ScalingReplicaSet   1m     deployment-controller  Scaled up replica set myapp-deployment-67c749c58c to 5
  Normal  ScalingReplicaSet   1s     deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 2
  Normal  ScalingReplicaSet   1s     deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 4
  Normal  ScalingReplicaSet   1s     deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 3
  Normal  ScalingReplicaSet   0s     deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 3
  Normal  ScalingReplicaSet   0s     deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 4
  Normal  ScalingReplicaSet   0s     deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 2
  Normal  ScalingReplicaSet   0s     deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 5
  Normal  ScalingReplicaSet   0s     deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 1
```

RollingUpdate

Upgrades



```
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
myapp-deployment-67c749c58c	0	0	0	22m
myapp-deployment-7d57dbdb8d	5	5	5	20m

Rollback

```
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
myapp-deployment-67c749c58c	0	0	0	22m
myapp-deployment-7d57dbdb8d	5	5	5	20m

```
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
myapp-deployment-67c749c58c	5	5	5	22m
myapp-deployment-7d57dbdb8d	0	0	0	20m



POD



POD



POD



POD



POD

Replica Set - 1



POD



POD



POD



POD



POD

Replica Set - 2

Deployment

```
> kubectl rollout undo deployment/myapp-deployment
```

```
deployment "myapp-deployment" rolled back
```

kubectl run

```
> kubectl run nginx --image=nginx
```

```
deployment "nginx" created
```

Summarize Commands

Create

```
> kubectl create -f deployment-definition.yml
```

Get

```
> kubectl get deployments
```

Update

```
> kubectl apply -f deployment-definition.yml
```

```
> kubectl set image deployment/myapp-deployment nginx=nginx:1.9.1
```

Status

```
> kubectl rollout status deployment/myapp-deployment
```

```
> kubectl rollout history deployment/myapp-deployment
```

Rollback

```
> kubectl rollout undo deployment/myapp-deployment
```

COMMANDS & ARGUMENTS

```
▶ docker run ubuntu
```

```
▶ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
--------------	-------	---------	---------	--------	-------

```
▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
45aacca36850	ubuntu	"/bin/bash"	43 seconds ago	Exited (0) 41 seconds ago	



```
# Install Nginx.
```

```
RUN \
  add-apt-repository -y ppa:nginx/stable && \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx
```

```
# Define mountable directories.
```

```
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d"]
```

```
# Define working directory.
```

```
WORKDIR /etc/nginx
```

```
# Define default command.
```

```
CMD ["nginx"]
```

```
ARG MYSQL_SERVER_PACKAGE_URL=https://repo.mysql.com/yum/mysql-8.0-community/docker/x86_64
```

```
ARG MYSQL_SHELL_PACKAGE_URL=https://repo.mysql.com/yum/mysql-tools-community/el/7/x86_64
```

```
# Install server
```

```
RUN rpmkeys --import https://repo.mysql.com/RPM-GPG-KEY-mysql \
  && yum install -y $MYSQL_SERVER_PACKAGE_URL $MYSQL_SHELL_PACKAGE_URL libpwquality \
  && yum clean all \
  && mkdir /docker-entrypoint-initdb.d
```

```
VOLUME /var/lib/mysql
```

```
COPY docker-entrypoint.sh /entrypoint.sh
```

```
COPY healthcheck.sh /healthcheck.sh
```

```
ENTRYPOINT ["/entrypoint.sh"]
```

```
HEALTHCHECK CMD /healthcheck.sh
```

```
EXPOSE 3306 33060
```

```
CMD ["mysqld"]
```



```
# Pull base image.
FROM ubuntu:14.04

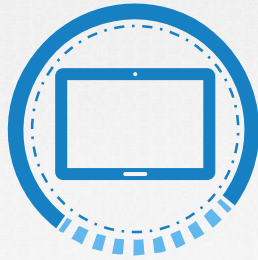
# Install.
RUN \
    sed -i 's/# \(\.*multiverse$\)\1/g' /etc/apt/sources.list && \
    apt-get update && \
    apt-get -y upgrade && \
    apt-get install -y build-essential && \
    apt-get install -y software-properties-common && \
    apt-get install -y byobu curl git htop man unzip vim wget && \
    rm -rf /var/lib/apt/lists/*

# Add files.
ADD root/.bashrc /root/.bashrc
ADD root/.gitconfig /root/.gitconfig
ADD root/.scripts /root/.scripts

# Set environment variables.
ENV HOME /root

# Define working directory.
WORKDIR /root

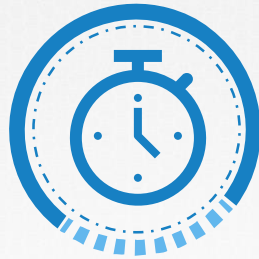
# Define default command.
CMD ["bash"]
```

??

```
▶ docker run ubuntu [COMMAND]
```

```
▶ docker run ubuntu sleep 5
```



```
FROM Ubuntu
```

```
CMD sleep 5
```

CMD command param1

CMD ["command", "param1"]

CMD sleep 5

CMD ["sleep", "5"]

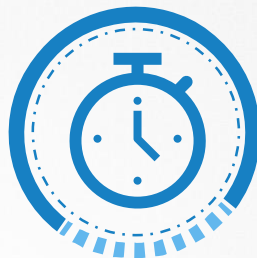


CMD ["sleep 5"]



```
▶ docker build -t ubuntu-sleeper .
```

```
▶ docker run ubuntu-sleeper
```



FROM Ubuntu

CMD sleep 5

```
▶ docker run ubuntu-sleeper sleep 10
```

Command at Startup: sleep 10

FROM Ubuntu

ENTRYPOINT ["sleep"]

```
▶ docker run ubuntu-sleeper 10
```

Command at Startup:

```
▶ docker run ubuntu-sleeper
sleep: missing operand
Try 'sleep --help' for more information.
```

Command at Startup:

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

```
▶ docker run ubuntu-sleeper
```

```
sleep: missing operand  
Try 'sleep --help' for more information.
```

Command at Startup:

```
▶ docker run ubuntu-sleeper 10
```

Command at Startup:

```
▶ docker run --entrypoint sleep 2.0 ubuntu-sleeper 10
```

Command at Startup:

```
▶ docker run --name ubuntu-sleeper ubuntu-sleeper
```

```
▶ docker run --name ubuntu-sleeper ubuntu-sleeper ["10"]
```

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-pod
spec:
  containers:
    - name:
      image:
      args:
```

```
▶ kubectl create -f pod-definition.yml
```


FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

```
▶ docker run --name ubuntu-sleeper \  
  --entrypoint["sleep 2.0"] \  
  ubuntu-sleeper 10
```

pod-definition.yml

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: ubuntu-sleeper-pod  
spec:  
  containers:  
    - name: ubuntu-sleeper  
      image: ubuntu-sleeper  
      command: ["10"]
```

```
▶ kubectl create -f pod-definition.yml
```

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]



pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-pod
spec:
  containers:
    - name: ubuntu-sleeper
      image: ubuntu-sleeper
      command: ["sleep2.0"]
      args: ["10"]
```


ENVIRONMENT VARIABLES



Environment Variables

app.py

```
import os
from flask import Flask

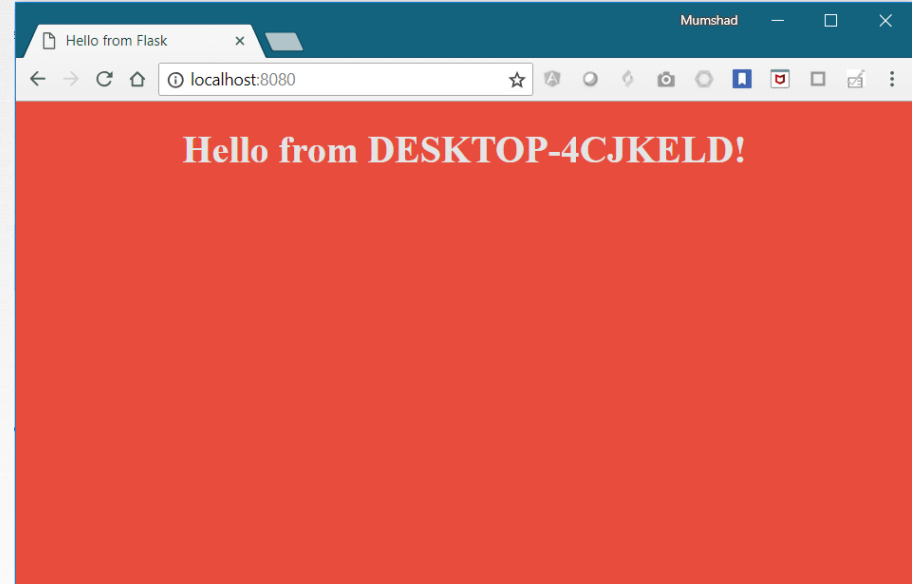
app = Flask(__name__)

...

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```



▶ python app.py

Environment Variables

app.py

```
import os
from flask import Flask

app = Flask(__name__)

...

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

Environment Variables

app.py

```
import os
from flask import Flask

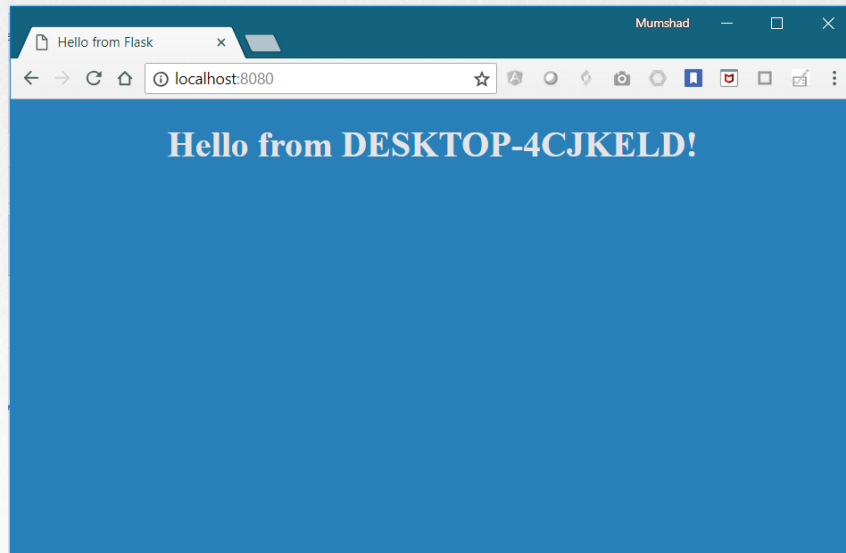
app = Flask(__name__)

...

color = os.environ.get('APP_COLOR')

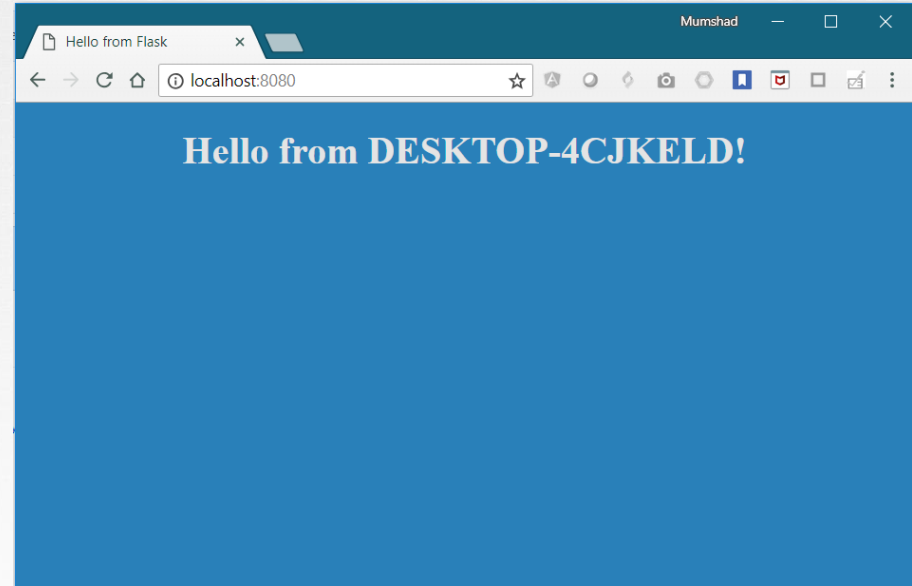
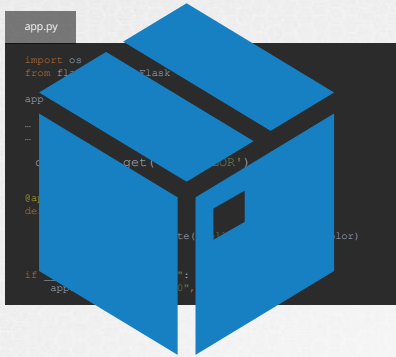
@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```



```
export APP_COLOR=blue; python app.py
```

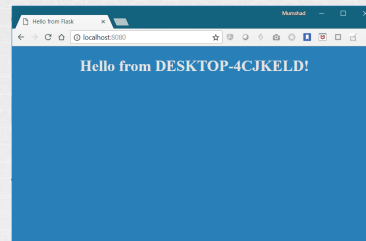
ENV Variables in Docker



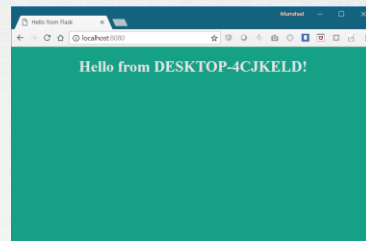
```
docker run simple-webapp:latest
```

ENV Variables in Docker

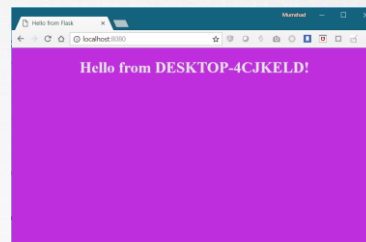
```
▶ docker run -e APP_COLOR=blue simple-webapp-color
```



```
▶ docker run -e APP_COLOR=green simple-webapp-color
```



```
▶ docker run -e APP_COLOR=pink simple-webapp-color
```

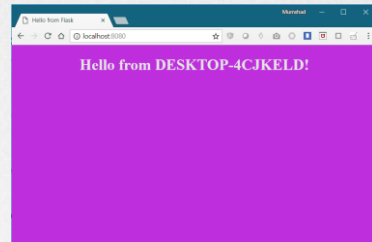


ENV Variables in Kubernetes

```
docker run -e APP_COLOR=pink simple-webapp-color
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    env:
      - name: APP_COLOR
        value: pink
```



ENV Value Types

env:

- **name:** APP_COLOR
value: pink



env:

- **name:** APP_COLOR
valueFrom:
configMapKeyRef:



env:

- **name:** APP_COLOR
valueFrom:
secretKeyRef:



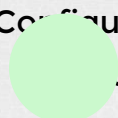
Course Objectives



Core Concepts



Configuration



ConfigMaps



SecurityContexts



Resource Requirements



Secrets



ServiceAccounts



Multi-Container Pods



Observability



Pod Design



Services & Networking



State Persistence

ConfigMaps

ConfigMap

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
spec:
  containers:
    - name: simple-webapp-color
      image: simple-webapp-color
      ports:
        - containerPort: 8080

      env:
        - name: APP_COLOR
          value: blue
        - name: APP_MODE
          value: prod
```

ConfigMaps

ConfigMap

```
APP_COLOR: blue  
APP_MODE: prod
```



Create ConfigMap



Inject into Pod

pod-definition.yaml

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: simple-webapp-color  
spec:  
  containers:  
    - name: simple-webapp-color  
      image: simple-webapp-color  
      ports:  
        - containerPort: 8080  
  
      envFrom:  
        - configMapRef:  
            name: app-config
```

Create ConfigMaps

ConfigMap

```
APP_COLOR: blue  
APP_MODE: prod
```

Imperative

▶ `kubectl create configmap`

Declarative

▶ `kubectl create -f`



Create ConfigMap

Create ConfigMaps

ConfigMap

```
APP_COLOR: blue
APP_MODE: prod
```

Imperative

```
▶ kubectl create configmap  
   <config-name> --from-literal=<key>=<value>
```

```
▶ kubectl create configmap \  
   app-config --from-literal=APP_COLOR=blue \  
   --from-literal=APP_MODE=prod
```

```
▶ kubectl create configmap  
   <config-name> --from-file=<path-to-file>
```

```
▶ kubectl create configmap \  
   app-config --from-file=app_config.properties
```



Create ConfigMap

Create ConfigMaps

ConfigMap

```
APP_COLOR: blue  
APP_MODE: prod
```

Declarative

▶ `kubectl create -f`

config-map.yaml

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: app-config  
data:  
  APP_COLOR: blue  
  APP_MODE: prod
```

▶ `kubectl create -f config-map.yaml`



Create ConfigMap

Create ConfigMaps

app-config

```
APP_COLOR: blue  
APP_MODE: prod
```

mysql-config

```
port: 3306  
max_allowed_packet: 128M
```

redis-config

```
port: 6379  
rdb-compression: yes
```



Create ConfigMap

View ConfigMaps

```
▶ kubectl get configmaps
```

NAME	DATA	AGE
app-config	2	3s

```
▶ kubectl describe configmaps
```

```
Name:          app-config
Namespace:     default
Labels:        <none>
Annotations:   <none>
```

```
Data
```

```
====
```

```
APP_COLOR:
```

```
----
```

```
blue
```

```
APP_MODE:
```

```
----
```

```
prod
```

```
Events:  <none>
```

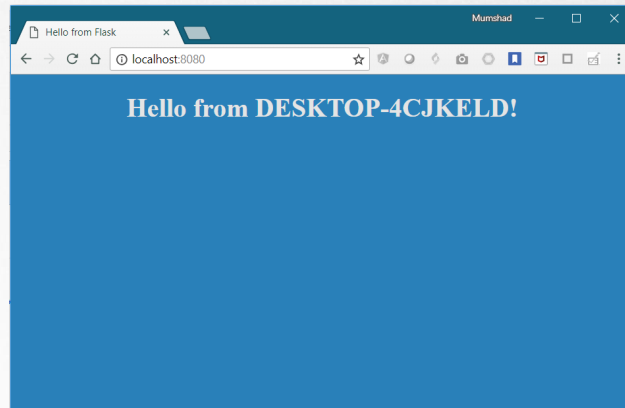

ConfigMap in Pods


pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    envFrom:
      - configMapRef:
          name: app-config
```

config-map.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_COLOR: blue
  APP_MODE: prod
```



 `kubectl create -f pod-definition.yaml`

ConfigMap in Pods

```
envFrom:  
  - configMapRef:  
      name: app-config
```

ENV

SINGLE ENV

```
env:  
  - name: APP_COLOR  
    valueFrom:  
      configMapKeyRef:  
        name: app-config  
        key: APP_COLOR
```

```
volumes:  
  - name: app-config-volume  
    configMap:  
      name: app-config
```

VOLUME

Kubernetes Secrets

Web-MySQL Application

app.py

```
import os
from flask import Flask

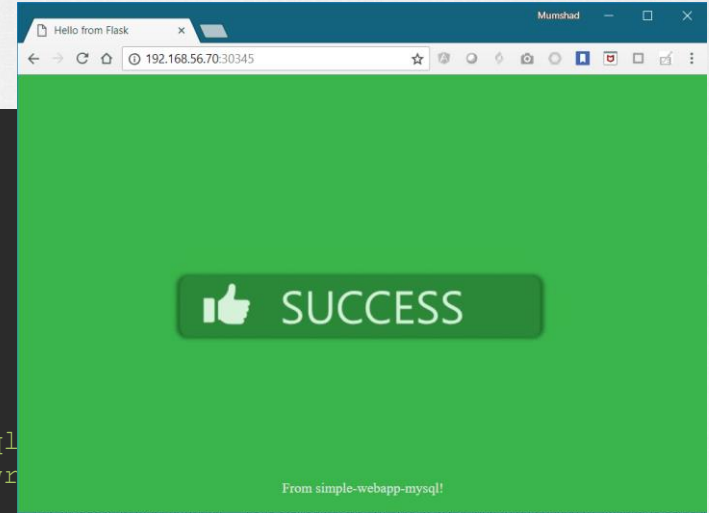
app = Flask(__name__)

@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                           user='root', password='password')

    return render_template('hello.html', color=fetchcolor())

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```



Web-MySQL Application

app.py

```
import os
from flask import Flask

app = Flask(__name__)

@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                           user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

Web-MySQL Application

app.py

```
import os
from flask import Flask

app = Flask(__name__)

@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                           user='root', password='paswr')

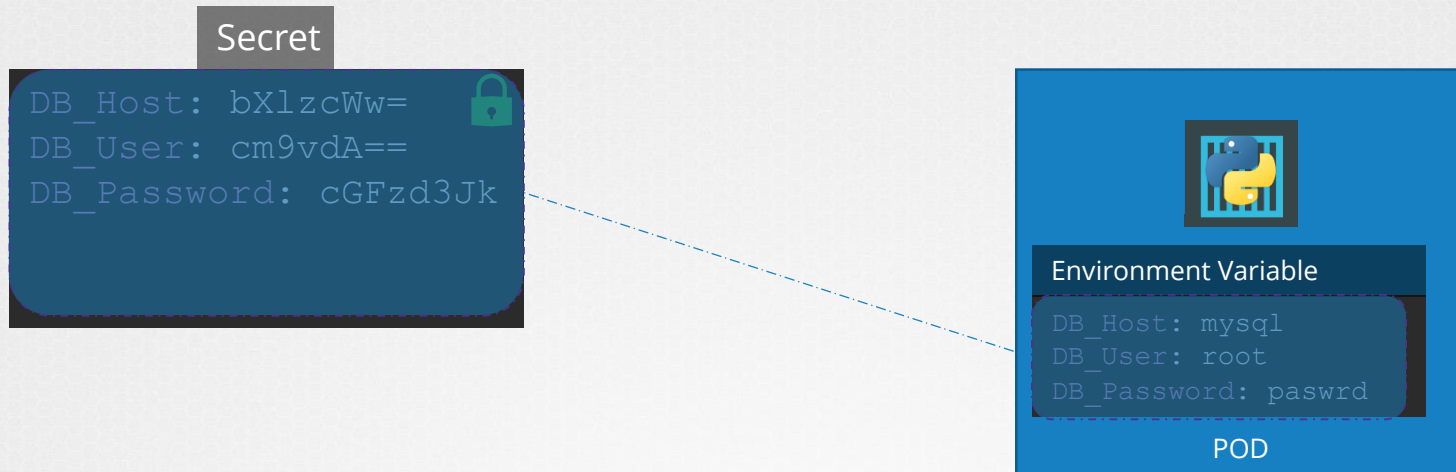
    return render_template('hello.html', color=fetchcolor())

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

config-map.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DB_Host: mysql
  DB_User: root
  DB_Password: paswr
```

Secret



Create Secret



Inject into Pod

Create Secrets

Secret

```
DB_Host: mysql  
DB_User: root  
DB_Password: paswrđ
```

Imperative

▶ `kubectl create secret generic`

Declarative

▶ `kubectl create -f`



Create Secret

Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswr
```

Imperative

```
▶ kubectl create secret generic
   <secret-name> --from-literal=<key>=<value>
```

```
▶ kubectl create secret generic \
  app-secret --from-literal=DB_Host=mysql \
             --from-literal=DB_User=root
             --from-literal=DB_Password=paswr
```

```
▶ kubectl create secret generic
   <secret-name> --from-file=<path-to-file>
```

```
▶ kubectl create secret generic \
  app-secret --from-file=app_secret.properties
```



Create Secret

Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswr
```

Declarative

```
▶ kubectl create -f
```

secret-data.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzZd3Jk
```

```
▶ kubectl create -f secret-data.yaml
```



Create Secret

| Encode Secrets

Secret

Declarative

▶ kubectl create -f

```
DB_Host: mysql
DB_User: root
DB_Password: paswrđ
```



```
DB_Host: bX1zcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
▶ echo -n 'mysql' | base64
```

```
bX1zcWw=
```

```
▶ echo -n 'root' | base64
```

```
cm9vdA==
```

```
▶ echo -n 'paswrđ' | base64
```

```
cGFzd3Jk
```

```
secret-c
  apiVersion: v1
  kind: Secret
  metadata:
    name: app-secret
```

```
data:
  mysql: bX1zcWw=
  root: cm9vdA==
  paswrđ: cGFzd3Jk
```

▶ kubectl create -f secret-data.yaml

View Secrets

```
▶ kubectl get secrets
```

NAME	TYPE	DATA	AGE
app-secret	Opaque	3	10m
default-token-mvtkv	kubernetes.io/service-account-token	3	2h

```
▶ kubectl describe secrets
```

Name: app-secret
Namespace: default
Labels: <none>
Annotations: <none>

Type: Opaque

Data

====

DB_Host: 10 bytes
DB_Password: 6 bytes
DB_User: 4 bytes

```
▶ kubectl get secret app-secret -o yaml
```

```
apiVersion: v1
data:
  DB_Host: bXlzcWw=
  DB_Password: cGFzd3Jk
  DB_User: cm9vdA==
kind: Secret
metadata:
  creationTimestamp: 2018-10-18T10:01:12Z
  labels:
    name: app-secret
    name: app-secret
    namespace: default
  uid: be96e989-d2bc-11e8-a545-080027931072
type: Opaque
```

| Decode Secrets

Secret

Declarative

kubectl create -f

```
DB_Host: mysql
DB_User: root
DB_Password: paswr
```



```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'bXlzcWw=' | base64 --decode
```

```
mysql
```

```
echo -n 'cm9vdA==' | base64 --decode
```

```
root
```

```
echo -n 'cGFzd3Jk' | base64 --decode
```

```
paswr
```

```
secret:
  type: Opaque
  kind: Secret
  metadata:
    name: app-secret
```

```
mysql:
  secretKeyRef:
    name: app-secret
    key: password
```

create -f secret-data.yaml

Secrets in Pods

pod-definition.yaml


```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
    - name: simple-webapp-color
      image: simple-webapp-color
      ports:
        - containerPort: 8080
  envFrom:
    - secretRef:
        name: app-secret
```

secret-data.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```



Inject into Pod

 `kubectl create -f pod-definition.yaml`

Secrets in Pods

```
envFrom:  
  - secretRef:  
      name: app-config
```

ENV

SINGLE ENV

```
env:  
  - name: DB_Password  
    valueFrom:  
      secretKeyRef:  
        name: app-secret  
        key: DB_Password
```

```
volumes:  
  - name: app-secret-volume  
    secret:  
      secretName: app-secret
```

VOLUME

Secrets in Pods as Volumes

```
volumes:  
- name: app-secret-volume  
  secret:  
    secretName: app-secret
```

VOLUME

```
ls /opt/app-secret-volumes
```

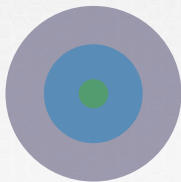
```
DB_Host      DB_Password  DB_User
```

```
cat /opt/app-secret-volumes/DB_Password
```

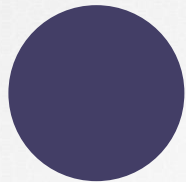
```
paswrđ
```

Inside the Container

Kubernetes Multi-Container PODs

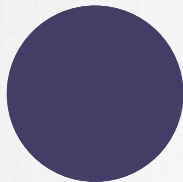


MONOLITH

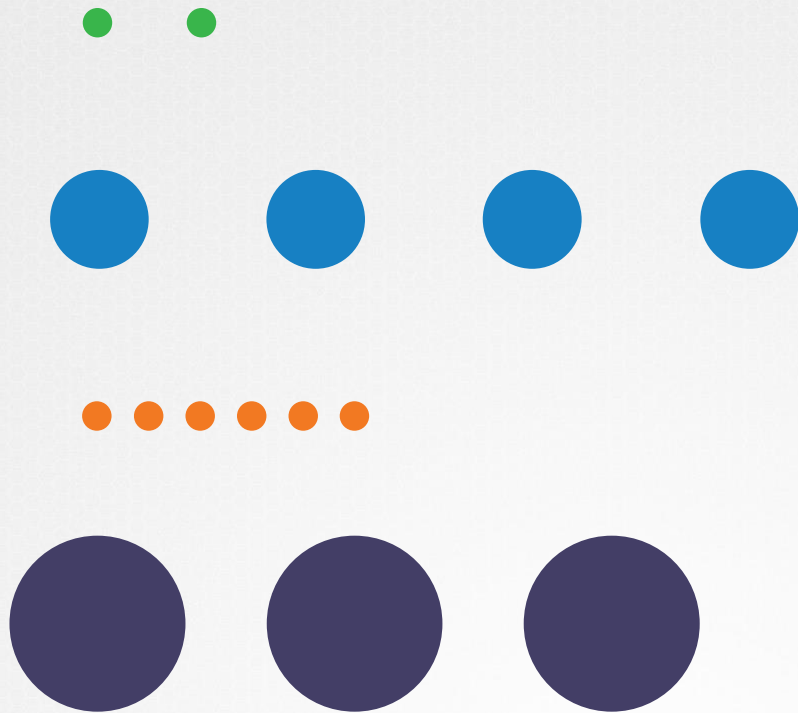


MICROSERVICES

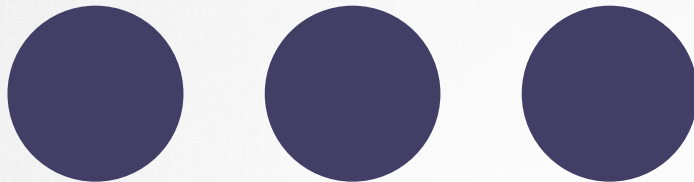




MICROSERVICES



MICROSERVICES

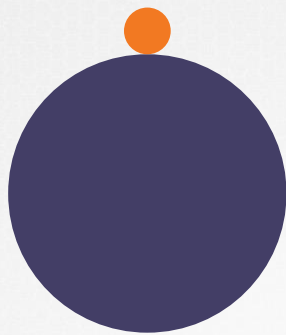


MICROSERVICES



LOG Agent

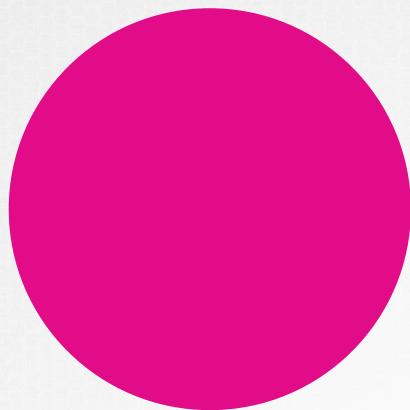
WEB Server





LOG Agent

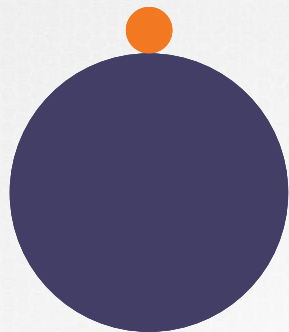
WEB Server

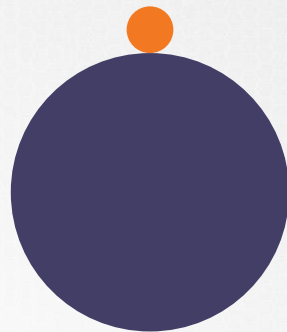
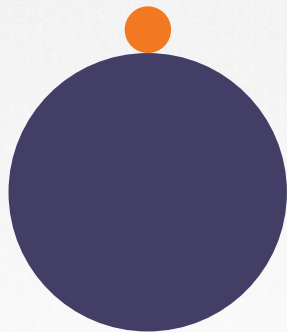
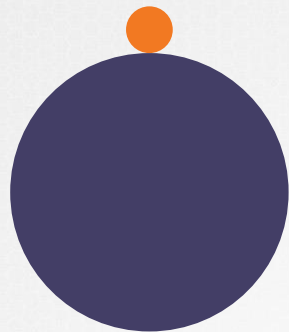




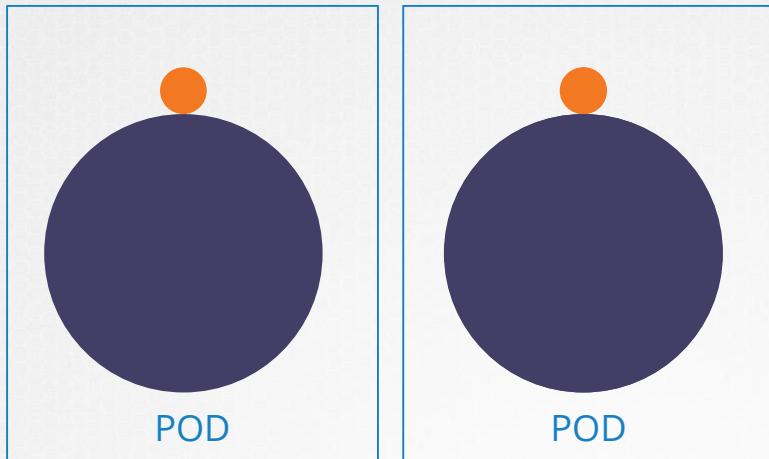
LOG Agent

WEB Server

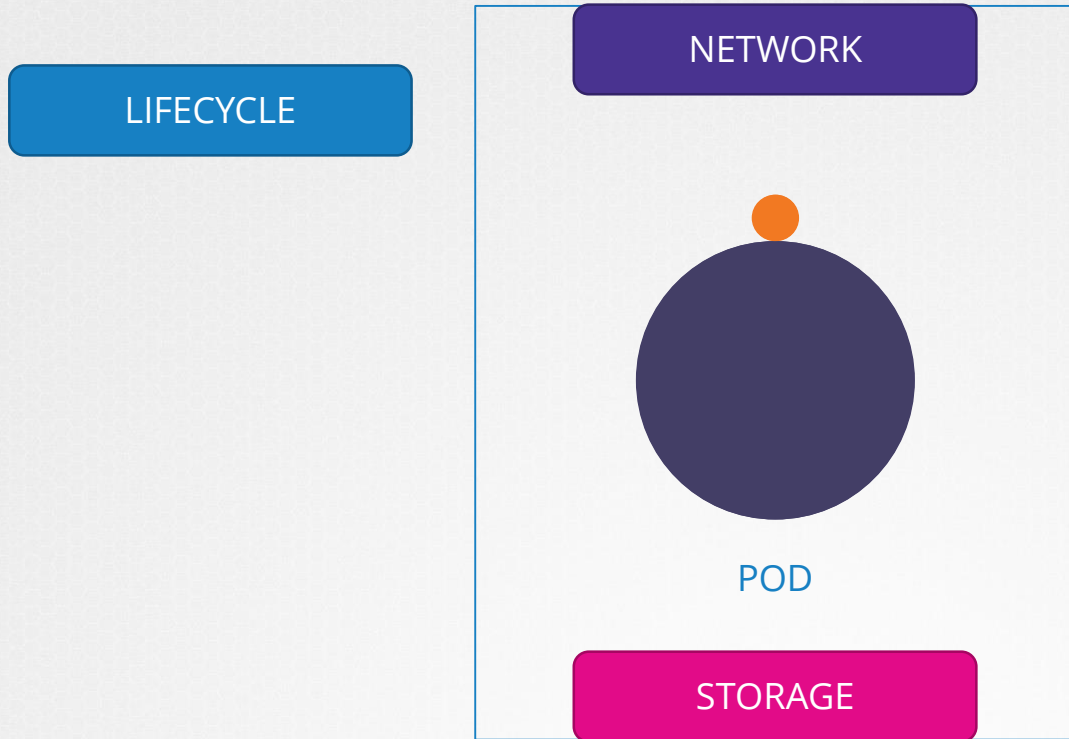




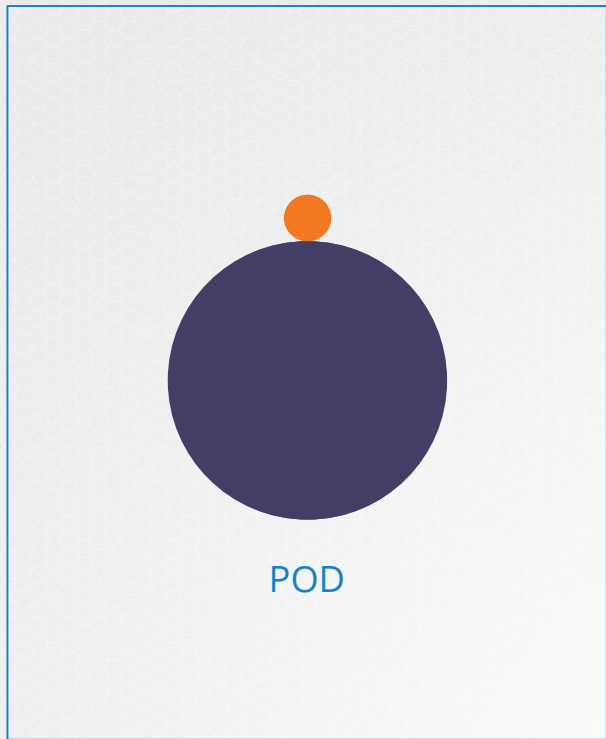
Multi-Container PODs



| Multi-Container PODs



Create



pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp
  labels:
    name: simple-webapp
spec:
  containers:
    - name: simple-webapp
      image: simple-webapp
      ports:
        - containerPort: 8080
    - name: log-agent
      image: log-agent
```