Operating Kubernetes on IBM Cloud
Part 1 - Essentials

Aco Vidovič
Hybrid Cloud Build Team Leader
IBM Central & Eastern Europe

Agenda

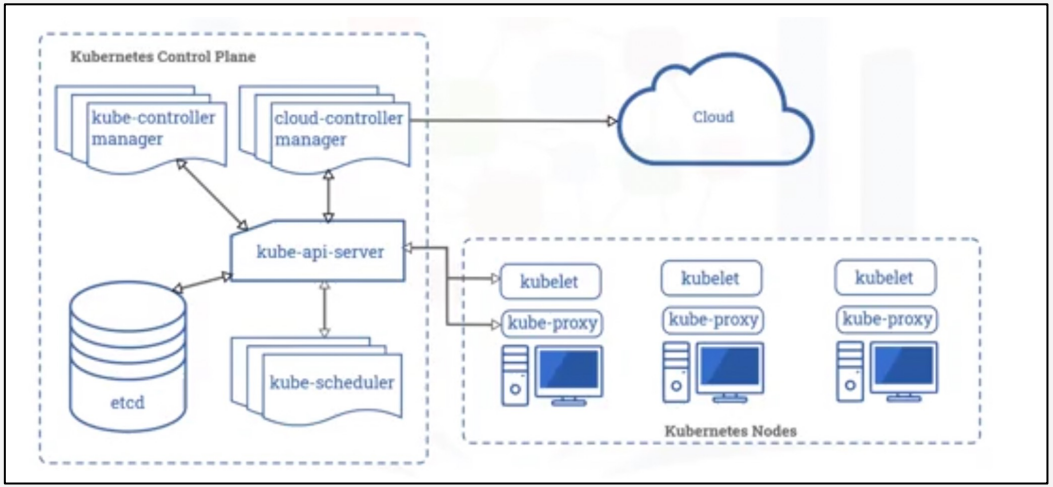
- Part 1 - Lecture:
  - Kubernetes objects
  - Working with Kubernetes Command Line Interface (CLI)
  - Application autoscaling and rolling updates
  - Deploying Kubernetes to IBM Cloud

Break

- Part 2 – Demo and hands-on labs:
  - Create your free instance of IKS
  - Deploy container image to Kubernetes
  - Updating apps with rolling updates and rollbacks
  - Using YAML configuration files
  - Scaling and auto-scaling
  - Connecting to a back-end service

Access materials:
https://ibm.biz/operating-iks-workshop

Managed Kubernetes cluster



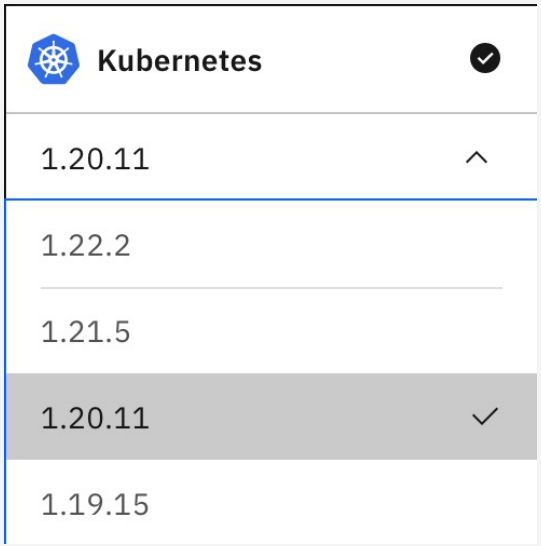
- Kubernetes cluster is made of:
  - One or more **Nodes** (also known as Worker nodes). This is where apps run.
  - **Control Plane** (also known as the Master node) which manages nodes
- Cloud Controller Manager in the Control Plane talks to the underlying cloud infrastructure
- Nodes are created by the cloud provider, not by Kubernetes

More info...

IBM Kubernetes Service

- A **certified, managed Kubernetes service** providing an intuitive user experience with on-going cluster management.
- Built-in **security and isolation** to enable rapid delivery of apps.
- Available in six IBM regions WW, including **20+ datacenters**.
- Fully dedicated, **single tenant Kubernetes clusters** deployed within customer account and network
- Seamless **integrated with IBM Cloud services**
- **Portability** with native Kubernetes experience and full **API support**

IBM **Cloud**
Kubernetes Service



More info on IKS...

Our hands-on exercises today

There are 2 ways to run hands-on lab today, choose the one you prefer:

- Use a demo environment at IBM Developer web site. No setup needed.

- Run the lab on your own Kubernetes cluster:
  - You create a cluster run by the IBM Kubernetes Service (IKS)
  - Download lab instructions files to your computer
  - Use CLI on your computer to interact with your IKS cluster

Preparing your own lab environment (optional)

If you choose to run the lab in your own environment:

1. Start a creation of a free Kubernetes cluster
   - In your web browser, go to https://ibm.biz/operating-iks-workshop
   - Download the instructions in file *Lab-1-create-iks-cluster.pdf*
   - Read and follow the instructions in the PDF file

2. Allow the cluster creation process enough time to finish. In the meantime, we will continue with lecture.

3. After 15-20 min, come back to the lab to complete it by setting up all necessary CLI tools.

Demo of this lab

Kubernetes objects

Kubernetes objects

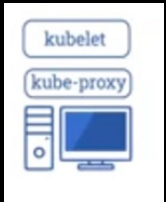
- Kubernetes objects are persistent entities. They represent the state of your cluster and define the desired state for your workload.
- Work with objects using Kubernetes APIs through:
  - kubectl CLI or
  - your programs using one of the Client Libraries

- Kubernetes has many **types** of objects: Nodes, Pods, Deployments, Services, Volumes….
  - Containers are not Kubernetes objects, they live in Pods

- Each object has configuration that includes (among others) two parts: **spec** and **status**
  - *Spec* depicts the desired state of object – as you want it to be
    - **A human being (you)** usually defines spec
  - *Status* of the object is what currently is
    - Kubernetes makes sure the status matches your spec
    - Kubernetes keeps track of the status. You can see it if you want to.

Node

- A type of Kubernetes object
- The worker machine in Kubernetes
  - Can be virtual or physical machine
- Managed by the Kubernetes control plane



Container runtime in Node

- Downloads images from registries and runs containers
- Kubernetes implements an interface so that this component is pluggable
- Available runtimes:
  - Docker – the most well known
  - CRI-O
  - containerd
  - and any other implementation of the Kubernetes Container Runtime Interface (CRI)

More info...

Labels and selectors

Labels 🏷️

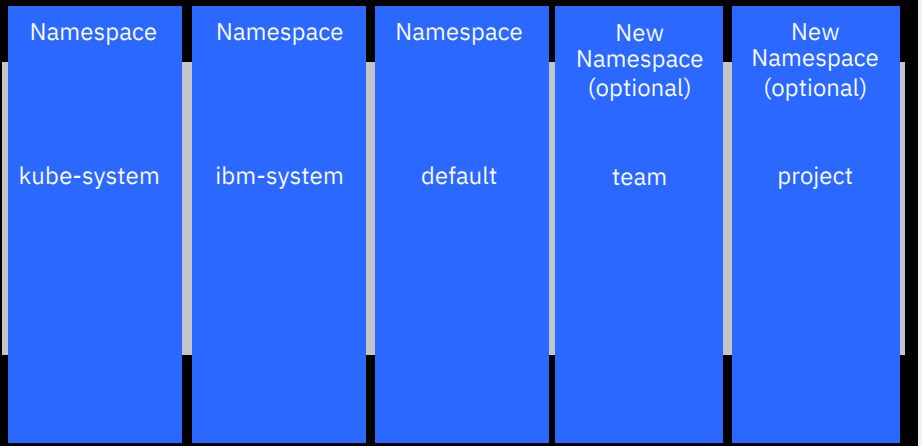
- Kubernetes objects have them. Purpose:
  - Identification of objects
  - Organization and grouping of objects
- Key/value pairs attached to objects
- Labels are not unique
  - Same label can be assigned to many objects.
    - Example: give the same label to all objects belonging to the same app
  - Each object can have zero, one or more labels
- Examples of labels:
  - **“app” : “ticketing”** – marks an object belonging to an *app* named “*ticketing*”
  - **“stage” : “prod”** – marks an object belonging to the production stage

Selectors

- Kubernetes’ means to select objects using their labels

Namespace

- Namespace is a type of object
- It helps with virtualisation of a physical cluster
- Segregate cluster by some meaningful criteria such as team, project etc.
- Necessary with larger number of users



Object names

- Each object has a name
- Names are **unique** for a type of resource **within a namespace**. You can give the same name to:
  - Two objects of *different types* in the same namespace:
    - A pod and a service named *guestbook* in namespace *default*
  - Two objects of the same type in *different namespaces*:
    - Pod named *guestbook* in namespace *default and a pod guestbook* in namespace *team*

Pod

- Pod is a smallest unit of work in Kubernetes
- It represents processes running in your cluster
- Container is in a Pod
  - Rarely more than one container in a pod
  - Replicating a pod scales an application horizontally
- Like other Kubernetes objects, Pod has a configuration described in its configuration file:
  - Either in YAML format
  - Or JSON

YAML configuration file describing a Pod

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
    labels:
      app: guestbook
4 spec:
    containers:
    - image: ibmcom/guestbook:v1
      name: guestbook
      ports:
      - containerPort: 3000
```

These four fields are present in every Kubernetes configuration file

Deployment

- An object that provides **updates** to Pod
- Contains a Pod configuration
  - In the template field
- Can run multiple **replicas** of a pod
  - Replica is identical copy of a pod
  - For scalability and redundancy
- Suitable for stateless applications
- A Deployment update triggers a Pod rollout
- Deployment controls:
  - Rollout of a pod to new version
  - Roll back to previous version

YAML file describing a Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-v1
  labels:
    app: guestbook
    version: "1.0"
spec:
  replicas: 3
  selector:
    matchLabels:
      app: guestbook
  template:
    metadata:
      labels:
        app: guestbook
        version: "1.0"
    spec:
      containers:
      - name: guestbook
        image: ibmcom/guestbook:v1
        ports:
        - name: http-server
          containerPort: 3000
```

Pod definition

Why do we need Deployments

A real-life microservices example: Book a Flight online

ReplicaSet

- ReplicaSet is a type of object
- Like Deployment, it maintains a set of identical pods
- Configuration looks similar to Deployment. It has:
  - Number of replicas
  - Pod template
  - Selector to identify which Pods it can acquire
- When you create a Deployment, ReplicaSet is automatically created
- It is possible but not recommended to explicitly create a ReplicaSet. Better create a Deployment, because Deployment:
  - Does everything that ReplicaSet does
  - Plus it can do application rollout and roll back

YAML file describing a ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  labels:
    app: guestbook
spec:
  replicas: 3
  selector:
    matchLabels:
      app: guestbook
  template:
    metadata:
      labels:
        app: guestbook
    spec:
      containers:
      - image: ibmcom/guestbook:v1
        name: guestbook
        ports:
        - containerPort: 3000
```

Service

- Service is a type of Kubernetes object
- It makes Deployments and Pods available to the outside world
- Your users or other apps can not directly access Deployments or Pods
  - Deployment/Pod **must be exposed through a Service**, only then your users and apps can access them
  - Pods are not-permanent, they are created and destroyed, while **Services are permanent**

- Service types:
  - ClusterIP – only available from within a cluster
  - NodePort - available via *IPaddress:Port*
  - LoadBalancer – available via a Cloud provider's load balancer
  - ExternalName – available via a CNAME record

YAML file describing a Service

```
apiVersion: v1
kind: Service
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
  selector:
    app: guestbook
  type: LoadBalancer
```

Working with IKS via Command Line Interface (CLI)

IBM Cloud CLI
- The CLI to work with IBM Cloud and all its resources
- The command is: `ibmcloud`
- It has several plugins, including:
  - IBM Kubernetes service (IKS). Command: `ibmcloud ks`
  - Container registry. Command: `ibmcloud cr`

Kubernetes CLI - `kubectl`
- Key tool for working with Kubernetes – platform independent
- Provides functionality for working with Kubernetes clusters and their workloads
- Two command types:
  - Imperative commands:
    - quickly create, update, and delete Kubernetes objects
    - easiest to learn
  - Declarative commands:
    - configuration files define one or more objects
    - no operation is specified

Working with IKS from CLI – sequence of use

1. Establish connection to your IBM cloud account.
In terminal window on your computer, run command:

```
ibmcloud login -a cloud.ibm.com -r <region> -g <resource-group>
```

2. Establish connection (set the Kubernetes context) to your IKS cluster.
On your computer run:

```
ibmcloud ks cluster config --cluster <your-cluster-name>
```

3. Verify that you can connect to your IKS cluster.
On your computer run:

```
kubectl config current-context
```

Kubernetes imperative commands examples

```
kubectl run nginx –image nginx
```

Pros of imperative commands:

- Quickly create, update, and delete Kubernetes objects
- Easiest to learn

Cons

- Don’t provide any audit trail
- Not very flexible

```
kubectl create –f nginx.yaml
```

- Use configuration template
- Specify an operation such as create, replace, or delete

Declarative commands

```
kubectl apply -f nginx.yaml
```

- Configuration files define one or more objects
- No operation is specified
  - Relevant operations are inferred by kubectl
- Work on files and directories
- Configuration files define desired state, and Kubernetes actualizes that state
- Preferred method for production systems

Viewing Kubernetes resources with *kubectl* command

Two popular commands to view your Kubernetes resources:

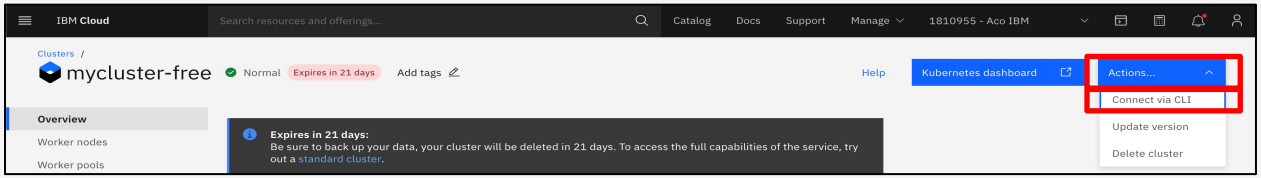
- `kubectl get`: to list resources and get basic data about them
- `kubectl describe`: to get details for a specific resource

```
kubectl get deployments -namespace kube-system
```

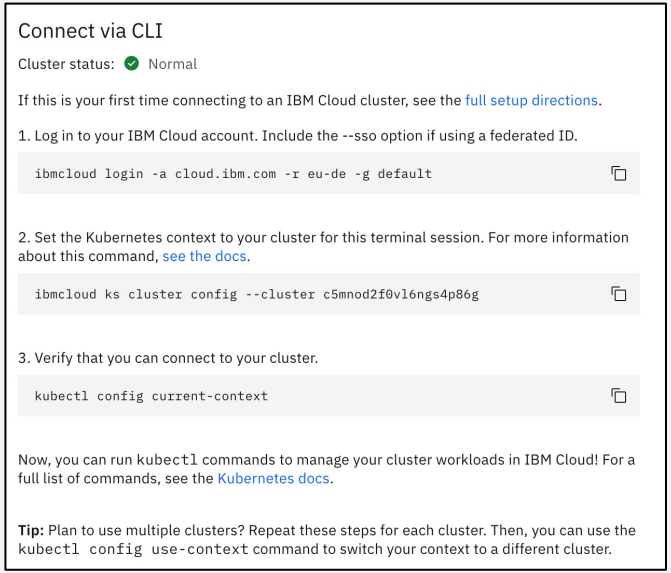
```
kubectl describe deployment kube-dns-amd64 -namespace kube-system
```

Finish the set up of your Kubernetes cluster (optional)

1. In your IBM Cloud console, go to your cluster, click **Actions**, then **Connect via CLI**



2. Follow the instructions in pop-up window



Often used commands: listing resources with *kubectl get*

```
😀 => kubectl get deployments
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
guestbook-v1    3/3     3            3           26h
redis-master    1/1     1            1           26h
redis-slave     2/2     2            2           26h
```

```
😀 => kubectl get namespaces
NAME              STATUS   AGE
default           Active   28h
ibm-cert-store    Active   28h
ibm-operators     Active   28h
ibm-system        Active   28h
kube-node-lease   Active   28h
kube-public       Active   28h
kube-system       Active   28h
```

```
😀 => kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
guestbook-v1-7676687667-klw2q    1/1     Running   0          26h
guestbook-v1-7676687667-s2gwl    1/1     Running   0          26h
guestbook-v1-7676687667-wvf4m    1/1     Running   0          26h
redis-master-7bf5f6d487-t9t89    1/1     Running   0          26h
redis-slave-58db457857-5pngx     1/1     Running   0          26h
redis-slave-58db457857-shgh9     1/1     Running   0          26h
```

```
😀 => kubectl get services
NAME           TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
guestbook      LoadBalancer   172.21.80.230    <pending>     3000:30614/TCP   27h
kubernetes     ClusterIP      172.21.0.1       <none>        443/TCP          28h
redis-master   ClusterIP      172.21.183.178   <none>        6379/TCP         26h
redis-slave    ClusterIP      172.21.254.149   <none>        6379/TCP         26h
```

```
😀 => kubectl get services -o wide
NAME           TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE   SELECTOR
guestbook      LoadBalancer   172.21.70.225    <pending>     3000:32338/TCP   17m   app=guestbook
kubernetes     ClusterIP      172.21.0.1       <none>        443/TCP          29h   <none>
redis-master   ClusterIP      172.21.183.178   <none>        6379/TCP         27h   app=redis,role=master
redis-slave    ClusterIP      172.21.254.149   <none>        6379/TCP         27h   app=redis,role=slave
```

There is a long / short command version :

- *kubectl get* ***pods*** *OR kubectl get* ***po***
- ***deployments*** *OR* ***deploy***
- ***services*** *OR* ***svc***
- ***namespaces*** *OR* ***ns***
- etc…

Help: *kubectl get –h*

For more output:
`-o wide`

Obtain details about any object with *kubectl describe*

```
😀 => kubectl describe deployment guestbook-v1
Name:                   guestbook-v1
Namespace:              default
CreationTimestamp:      Sun, 28 Feb 2021 19:07:40 +0100
Labels:                 app=guestbook
                        version=1.0
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=guestbook
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=guestbook
           version=1.0
  Containers:
   guestbook:
    Image:        ibmcom/guestbook:v1
    Port:         3000/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   guestbook-v1-7676687667 (3/3 replicas created)
Events:          <none>
```

kubectl describe ***deployment***

kubectl describe ***svc***

kubectl describe ***pod***

Etc...

Creating and deleting resources with *kubectl*

Create object imperatively:

- *kubectl create <resource-type> <resource-name> …*
- *kubectl create -f <file-name>*

Declaratively: *kubectl apply -f <file-name>*

Delete: *kubectl delete <resource-type> <resource-name>*

```
😀 => kubectl apply -f guestbook-service.yaml
service/guestbook created
```

```
😀 => kubectl get service guestbook
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
guestbook   LoadBalancer   172.21.70.225   <pending>     3000:32338/TCP   49s
```

```
😀 => kubectl delete service guestbook
service "guestbook" deleted
```

Manage Kubernetes applications

Some common operations. Let Kubernetes do:

- Auto-scaling – automatically scale your app up or down
- Rolling updates – update applications to another version while they are still running

Auto-scaling – how to do it?

You can auto-scale a Deployment or a ReplicaSet using command `kubectl autoscale`

Example: auto-scale between 2 and 4 replicas while keeping the CPU utilization around 10%:

```
😀 => kubectl autoscale deployment guestbook-v1 --min=2 --max=4 --cpu-percent=10
horizontalpodautoscaler.autoscaling/guestbook-v1 autoscaled
```

```
😀 => kubectl get pods
NAME                              READY   STATUS        RESTARTS   AGE
guestbook-v1-7676687667-c2dkf     0/1     Terminating   0          11m
guestbook-v1-7676687667-c66mp     1/1     Running       0          28m
guestbook-v1-7676687667-m2ghg     1/1     Running       0          16m
guestbook-v1-7676687667-s2jxd     0/1     Terminating   0          16m
guestbook-v1-7676687667-vtxwv     1/1     Running       0          28m
guestbook-v1-7676687667-zqzkq     1/1     Running       0          16m
```

And after a while…

```
😀 => kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
guestbook-v1-7676687667-c66mp     1/1     Running   0          30m
guestbook-v1-7676687667-m2ghg     1/1     Running   0          18m
guestbook-v1-7676687667-vtxwv     1/1     Running   0          30m
guestbook-v1-7676687667-zqzkq     1/1     Running   0          18m
redis-master-7bf5f6d487-t9t89     1/1     Running   0          29h
```

Auto-scaling – behind the scenes

A Horizontal Pod Autoscaler (HPA) was created behind the scenes

```
😀 => kubectl get hpa
NAME           REFERENCE                   TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
guestbook-v1   Deployment/guestbook-v1   <unknown>/10%   2         4         4          9m19s
```

You can create a standalone HPA using YAML, but it is recommended autoscale with *kubectl*

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: guestbook-v1
  namespace: default
spec:
  maxReplicas: 4
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: guestbook-v1
  targetCPUUtilizationPercentage: 10
```

Manage Kubernetes applications

Some common operations.

- Auto-scaling
- Rolling updates – update applications to another version while they are still running

Application update strategies

- **Recreate**: *terminate the old* app version, then release new one
- **Rolling**: at the same time terminate old pods and create new ones, so some pods are always running. *There is no app downtime*
- **Blue-green:** release a new version alongside the old version then *switch traffic*
- **Canary:** release a *new version to a subset of users*, then proceed to a full rollout

Rolling updates

- Replicas and autoscaling are important to *minimize downtime* and service interruptions
- Rolling updates are a way to roll out application changes in an automated and controlled fashion throughout your pods
- Work with Pod templates defined in Deployments
- Allow for rollback if something went wrong with new version

Enable your Deployment configuration for rolling updates

1. Add *liveness* and *readiness probes* to your configuration file. This ensures deployments are marked ready appropriately.

Add these two sections into the Deployment's config file:

```
livenessProbe:
  httpGet:
    path: /
    port: 9080
  initialDelaySeconds: 300
  periodSeconds: 15
readinessProbe:
  httpGet:
    path: /
    port: 9080
    initialDelaySeconds: 45
    periodSeconds: 5
```

2. Add rolling update strategy to your YAML file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-test
spec:
  replicas: 10
  selector:
    matchLabels:
      service: http-server
  minReadySeconds: 5
  progressDeadlineSeconds: 600
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 50%
      maxSurge: 2
```

Rolling updates: rollout the new version - V2

`kubectl set image…`

```
😀 => kubectl set image deployment/guestbook-v1 guestbook=ibmcom/guestbook:v2
deployment.apps/guestbook-v1 image updated
```

To check what happened: `kubectl rollout status…`

```
😀 => kubectl rollout status deployments/guestbook-v1
deployment "guestbook-v1" successfully rolled out
```

```
😀 => kubectl describe po guestbook-v1-8489ddbb68-2bpfn
Name:         guestbook-v1-8489ddbb68-2bpfn
Namespace:    default
Priority:     0
Node:         10.144.51.71/10.144.51.71
Start Time:   Tue, 02 Mar 2021 01:06:25 +0100
Labels:       app=guestbook
              pod-template-hash=8489ddbb68
              version=1.0
Annotations:  cni.projectcalico.org/podIP: 172.30.97.22/32
              cni.projectcalico.org/podIPs: 172.30.97.22/32
              kubernetes.io/psp: ibm-privileged-psp
Status:       Running
IP:           172.30.97.22
IPs:
  IP:           172.30.97.22
Controlled By:  ReplicaSet/guestbook-v1-8489ddbb68
Containers:
  guestbook:
    Container ID:   containerd://b5943c6de620b9371bc53c1c2e263021ee845d85c7be518bf76c1259791326f0
    Image:          ibmcom/guestbook:v2
```

Rolling updates: rollback to previous version

If the update doesn't work well: `kubectl rollout undo`…

```
😀 => kubectl rollout undo deployment/guestbook-v1
deployment.apps/guestbook-v1 rolled back
```

```
😀 => kubectl get pods
NAME                           READY   STATUS              RESTARTS   AGE
guestbook-v1-7676687667-6wqqq  0/1     ContainerCreating   0          1s
guestbook-v1-7676687667-gzwmg  1/1     Running             0          4s
guestbook-v1-7676687667-mx768  0/1     ContainerCreating   0          0s
guestbook-v1-7676687667-thtwj  1/1     Running             0          4s
guestbook-v1-8489ddbb68-2bpfn  1/1     Running             0          7m25s
guestbook-v1-8489ddbb68-frzgz  1/1     Terminating         0          7m22s
guestbook-v1-8489ddbb68-qntt2  1/1     Terminating         0          7m25s
guestbook-v1-8489ddbb68-vv8px  0/1     Terminating         0          7m22s
```

Rolling updates: check the rollback status

```
😀 => kubectl rollout status deployments/guestbook-v1
deployment "guestbook-v1" successfully rolled out
```

```
😀 => kubectl describe po guestbook-v1-7676687667-6wqqq
Name:         guestbook-v1-7676687667-6wqqq
Namespace:    default
Priority:     0
Node:         10.144.51.71/10.144.51.71
Start Time:   Tue, 02 Mar 2021 01:13:49 +0100
Labels:       app=guestbook
              pod-template-hash=7676687667
              version=1.0
Annotations:  cni.projectcalico.org/podIP: 172.30.97.18/32
              cni.projectcalico.org/podIPs: 172.30.97.18/32
              kubernetes.io/psp: ibm-privileged-psp
Status:       Running
IP:           172.30.97.18
IPs:
  IP:           172.30.97.18
Controlled By:  ReplicaSet/guestbook-v1-7676687667
Containers:
  guestbook:
    Container ID:   containerd://f3dd26af8204b2f5f485e4c3b182b40cb2c31298dcf904b16bab132d1b379234
    Image:          ibmcom/guestbook:v1
```

Hands-on lab

In this lab, you will work with Deployments, Services, scaling and updates on IKS

- Deploy container image with the Guestbook app to your cluster
- Experiment with replicas, application update and rollback
- Use YAML configuration files
- Persist data by connecting the app to a back-end service (database)

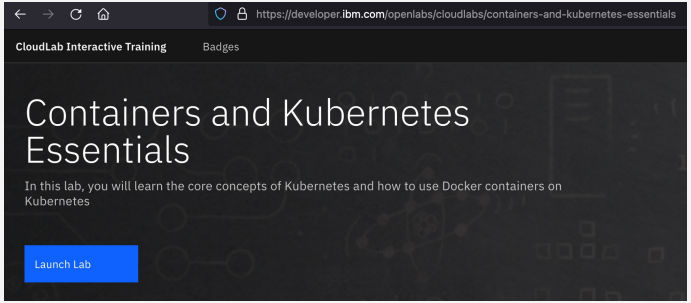
How to do the hands-on lab

Option 1: Do the lab using your own Kubernetes cluster on IBM Cloud

1. In your web browser, go to https://ibm.biz/operating-iks-workshop
2. Download the file *Lab-2-iks-essentials.pdf*
3. Read and follow the instructions in the PDF file

Option 2: Do the lab in a pre-configured demo environment (faster)

- In a demo environment at IBM Developer web site, click on **Launch Lab** and follow instructions.



Resources

Part 2 of this workshop: Operating Kubernetes on IBM Cloud, Part 2 - Best Practices

- November 3 at 16:00 – 18:00 CET
- Same URL as today

IBM Kubernetes Service docs

Videos: Introduction to Container Orchestration with IBM Cloud, Part 1 and Part 2

Credly (Acclaim) badge - Operating Kubernetes on IBM Cloud

Thank you!