Rules!

1. be fast, avoid creating yaml manually from scratch
2. use only [kubernetes.io/docs](https://kubernetes.io/docs/home/) for help.

## Notices

* This challenge was tested on k8s 1.18. Please let us know should you encounter any issues in the comments

## Scenario Setup

You will start a two node cluster on your machine, one master and one worker. For this you need to install VirtualBox and vagrant, then:

git clone [git@github.com](mailto:git@github.com):wuestkamp/cka-example-environments.git  
cd cka-example-environments/cluster1  
./up.sh  
  
vagrant ssh cluster1-master1  
**vagrant@cluster1-master1**:**~**$ sudo -i  
**root@cluster1-master1**:~# kubectl get node

You should be connected as root@cluster1-master1 . You can connect to other worker nodes using root, like ssh root@cluster1-worker1

If you want to destroy the environment again run ./down.sh . **You should destroy the environment after usage so no more resources are used!**

# **Investigate Multi container Pod issue**

1. Get the amount of nodes plus their status and all available kubectl contexts.
2. In namespace management there is a pod named web-server, check its status.
3. Find the reason / error in the pod logs.
4. Directly gather the logs of the docker containers and check for issues.
5. Fix the pod and ensure its running.

# **Scheduler Playground**

1. How many pods and services are in namespace management? Create a command that uses jsonpath to output the number.
2. How many schedulers are running? Find the parameters each is started with.
3. There is an existing deployment named what-a-deployment in namespace development . With which scheduler are the pods of that deployment scheduled?
4. Create a pod of image nginx:1.16.1-alpine in namespace development using the scheduler kube-scheduler-amazing and check its running + which node it got assigned.
5. Create another pod of image nginx:1.16.1-alpine in namespace development using the scheduler kube-scheduler-none-existing. What status does the pod have and what do the logs show?

# **Advanced Scheduling**

# How many pods of deployment coredns are running on which nodes.

# Why are coredns pods running the nodes they are?

# Show the config of coredns . The actual Corefile not the k8s yaml.

# Create a deployment of image httpd:2.4.41-alpine with 3 replicas which can run on all nodes. How many are running on each node?

# Now change the pods to only run on master nodes

# **Node Management**

# Drain the node cluster1-worker1. Save single pods manually to restore afterwards.

# Remove the node cluster1-worker1 temporarily from the cluster.

# Add the worker node back to the cluster and allow scheduling again. Restore the single pods.

# Create a pod of image httpd:2.4.41-alpine and confirm its scheduled on cluster1-worker1.

# Expose the pod via a NodePort service and connect to it using curl. You should be able to connect to the internal IP of cluster1-worker1 from your local machine.

# Suspend the master node (vagrant suspend cluster1-master1), then check: is the pod still reachable via the NodePort service?

# Resume the master node (vagrant resume cluster1-master1) and confirm node status normal.