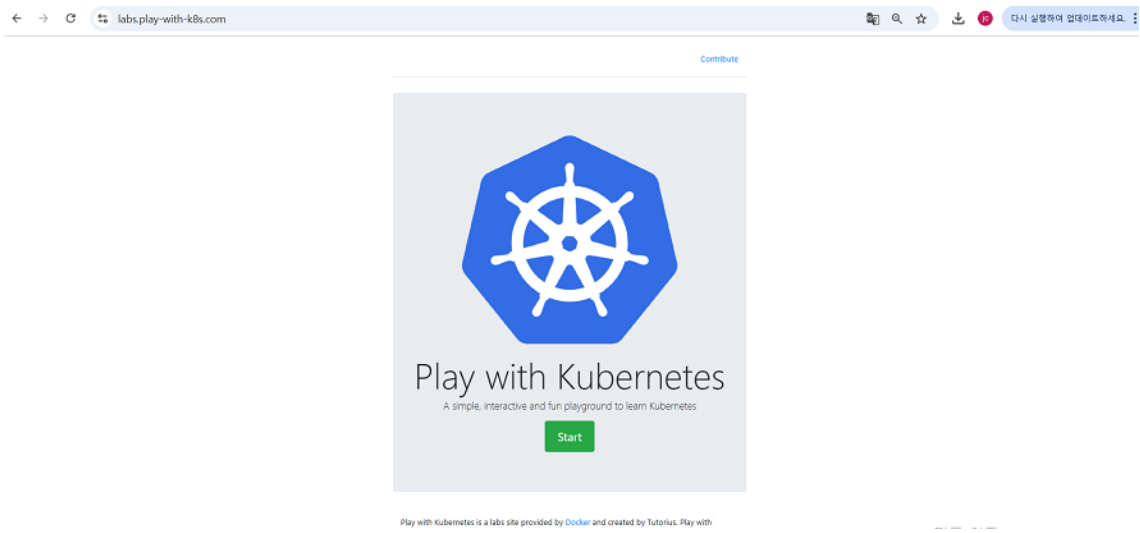
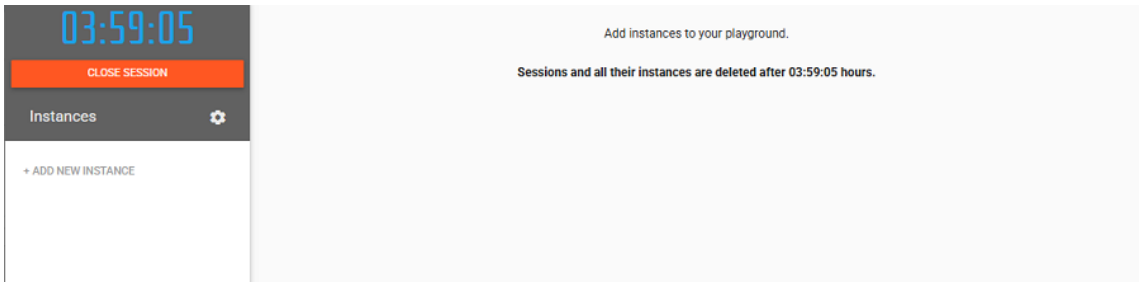
Play with kubernetes

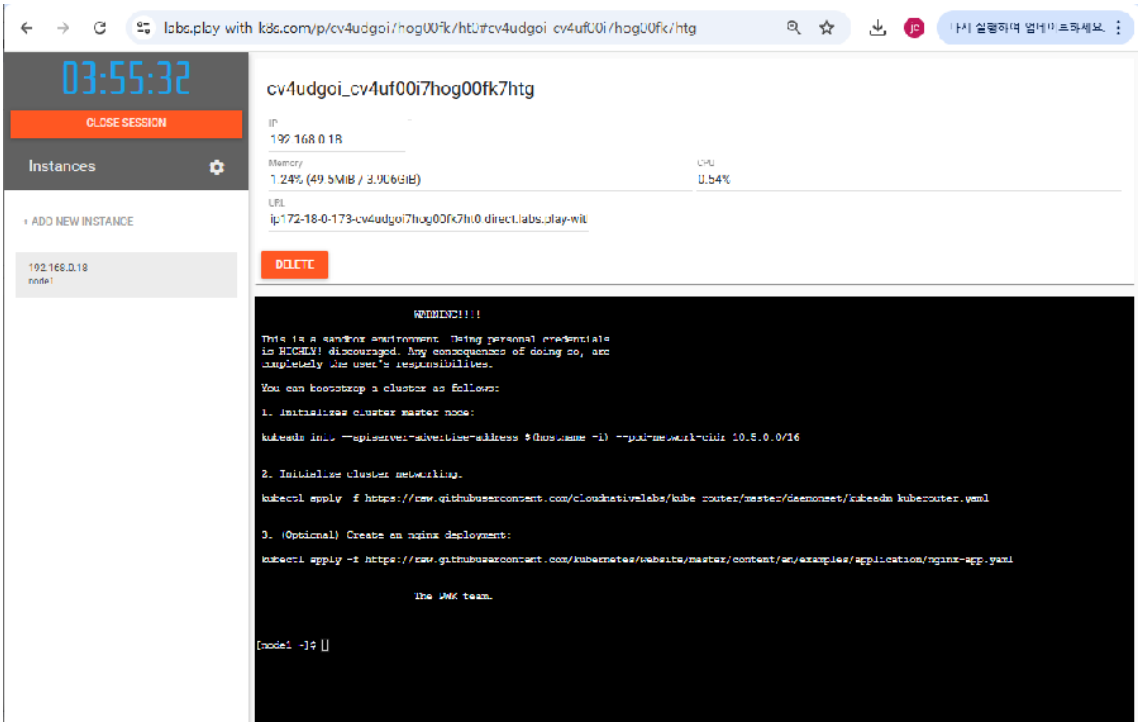
- https://labs.play-with-k8s.com
- Docker 에서 제공하고 Tutorius 에서 만든 사이트.



사이트 초기 페이지. start 버튼으로 시작.



한번 접속하면 4시간까지만 세션을 유지할 수 있다. 세션이 끊어지면 했던 작업들이 전부 날라감.



kubernets

왼쪽 메뉴에서 Add New Instance 선택 -> 화면에서 처럼 클러스터를 구성하기 위한 명령어가 출력됨.

->

```
[node1 ~]$ kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16
Initializing machine ID from random generator.
...
...
        [WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit status 1
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

[WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit status 1

- 이 경고는 커널 빌드시 설정확인을 제공하는 모듈이므로 **무시해도 된다**. 커널 빌드를 할 것이 아니기 때문에.

[WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist

- 이 경고는 kubernetes 가 브리지 네트워크 필터링 설정을 확인할때 해당 설정 파일이 존재하지 않아서 발생한다. bridge-nf-call-iptables 는 리눅스 커널이 브리지 네트워크 패킷을 iptables 규칙에 따라 필터링 할수 있도록 하는 기능. 여러가지 자료를 찾아본 결과 play with kubernetes 환경에서는 **무시해도 된다.**(설정을 하려고 해도 이유는 알수 없지만 설정이 되지 않는다. sysctl net.bridge.bridge-nf-call-iptables=1 ; 여기서 에러가 발생)

2. Initialize cluster networking:

$ kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

<= calico, flannel, weave-net 등의 CNI(container network interface) 가 있는데 여기서는 kube-router 를 사용

```
[node1 ~]$ kubectl get nodes
NAME     STATUS    ROLES           AGE      VERSION
node1    Ready     control-plane   4m57s    v1.27.2
[node1 ~]$ kubectl get pods -o wide
No resources found in default namespace.
[node1 ~]$ kubectl get pods -n kube-system
NAME                              READY    STATUS     RESTARTS    AGE
coredns-5d78c9869d-4q8m5          1/1      Running    0           5m3s
coredns-5d78c9869d-6qn9k          1/1      Running    0           5m3s
etcd-node1                        1/1      Running    0           5m13s
kube-apiserver-node1              1/1      Running    0           5m19s
kube-controller-manager-node1     1/1      Running    0           5m15s
kube-proxy-s5d79                  1/1      Running    0           5m3s
kube-router-d5nsb                 1/1      Running    0           36s
kube-scheduler-node1              1/1      Running    0           5m15s
[node1 ~]$
```

3. 클러스터를 테스트할 pod 를 실행

kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml

<= 이 파일을 실행하는 대신에 간단하게 아래처럼 명령어로 대신할 수 도 있다

```
[node1 ~]$ kubectl run pod apache --image httpd
pod/pod created
[node1 ~]$ kubectl get pods
NAME    READY   STATUS      RESTARTS   AGE
pod     0/1     Pending     0          6s
[node1 ~]$
```

*. 기본적으로 control-plane node 에는 Noschedule taint 가 설정되어 있어서 pod 가 스케줄링 되지 않는다. 아 래 명령어로 확인.

```
[node1 ~]$ kubectl describe node node1 | grep -i taint
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
```

control plane 에 pod 를 스케줄링 하고자 한다면 taint 를 제거하거나 적절한 toleration 을 설정해야 한다. 아래 명령어는 NoSchedule taint 를 무시하는 명령어.

```
[node1 ~]$ kubectl patch pod apache -p '{"spec":{"tolerations":[{"key":"node-role.kubernetes.io/control-plane","effect":"NoSchedule"}]}}'

pod/apache patched

[node1 ~]$ kubectl get pods
NAME     READY   STATUS              RESTARTS   AGE
apache   0/1     ContainerCreating   0          48s
[node1 ~]$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
apache   1/1     Running   0          52s
[node1 ~]$
```

node1 만으로도 이렇게 pod 를 실행해서 test 를 할수는 있지만 다양한 테스트와 안정적인 실행환경을 위해서는 한대만으로는 부족하다. 최소한 3대 이상으로 테스트 하는것을 권장한다. 나머지는 worker node 로 구성하면 된다.

.

왼쪽 메뉴에서 Add New Instance 를 선택한다음 master 노드(현재 node1) 에서 클러스터 조인을 위한 토큰 생성후 worker node 에서 실행하면 클러스트에 조인이 된다.

[**node1** ~]$ kubeadm token create --print-join-command

[**node2** ~]$ vi token

[node2 ~]$ cat token

kubeadm join 192.168.0.13:6443 --token y0o915.6iyidzhv1elnkc6d
--discovery-token-ca-cert-hash
sha256:0fde0dfcd8779620a3e9447585ece62c0b4ce5df1f5a4e0159a7c8f52c65520e

[node2 ~]$ chmod u+x token

[node2 ~]$ ./token

Initializing machine ID from random generator.

W0307 15:00:09.380388 758 initconfiguration.go:120] Usage of CRI endpoints without URL scheme is deprecated and can cause kubelet errors in the future. Automatically prepending scheme "unix" to the "criSocket" with value "/run/docker/containerd/containerd.sock". Please update your configuration!

[preflight] Running pre-flight checks

[preflight] The system verification failed. Printing the output from the verification:

KERNEL_VERSION: 4.4.0-210-generic

OS: Linux

CGROUPS_CPU: enabled

CGROUPS_CPUACCT: enabled

CGROUPS_CPUSET: enabled

CGROUPS_DEVICES: enabled

CGROUPS_FREEZER: enabled

CGROUPS_MEMORY: enabled

CGROUPS_PIDS: enabled

CGROUPS_HUGETLB: enabled

CGROUPS_BLKIO: enabled

[WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit status 1

[WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist

[preflight] Reading configuration from the cluster...

[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'

[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"

[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"

[kubelet-start] Starting the kubelet

[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[**node1** ~]$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
node1 Ready control-plane 39m v1.27.2
node2 Ready <none> 4m9s v1.27.2
[node1 ~]$

node2 가 쿠버네티스 클러스터에 추가되었고 상태가 정상인것을 확인할 수 있다.

똑 같은 방법으로 한대 더 추가해서 cluster 에 참여시킨다.
아래처럼 token 파일에 저장하지 않고 복사 붙여넣기로 바로 실행을 해도 된다.
[**node3** ~]$ kubeadm join 192.168.0.13:6443 --token y0o915.6iyidzhv1elnkc6d
--discovery-token-ca-cert-hash
sha256:0fde0dfcd8779620a3e9447585ece62c0b4ce5df1f5a4e0159a7c8f52c65520e

[**node1** ~]$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
node1 Ready control-plane 43m v1.27.2
node2 Ready <none> 8m42s v1.27.2
node3 Ready <none> 20s v1.27.2
[node1 ~]$

master node(control-plane) 1대, worker node 2대로 kubernetes cluster 완료.
* 주의 세션이 끊어지면 모든 node 가 사라짐.(세션은 최대 4시간 동안만 지속됨)

테스트
[node1 ~]$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/appl
ication/nginx-app.yaml
service/my-nginx-svc created
deployment.apps/my-nginx created
[node1 ~]$ kubectl get pods
NAME READY STATUS RESTARTS AGE
apache 1/1 Running 0 30m
my-nginx-cbdccf466-dxt7g 0/1 ContainerCreating 0 6s
my-nginx-cbdccf466-r4fdb 0/1 ContainerCreating 0 6s

```
my-nginx-cbdccf466-wn7zw   0/1     ContainerCreating   0          7s
[node1 ~]$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
apache                      1/1     Running   0          31m
my-nginx-cbdccf466-dxt7g    1/1     Running   0          16s
my-nginx-cbdccf466-r4fdb    1/1     Running   0          16s
my-nginx-cbdccf466-wn7zw    1/1     Running   0          17s
[node1 ~]$ kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP
NODE    NOMINATED NODE   READINESS GATES
apache                      1/1     Running   0          31m   10.5.0.4   node1
<none>           <none>
my-nginx-cbdccf466-dxt7g    1/1     Running   0          23s   10.5.1.3   node2
<none>           <none>
my-nginx-cbdccf466-r4fdb    1/1     Running   0          23s   10.5.1.2   node2
<none>           <none>
my-nginx-cbdccf466-wn7zw    1/1     Running   0          24s   10.5.2.2   node3
<none>           <none>
[node1 ~]$

[node1 ~]$ curl 10.5.1.2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
```

nginx.com.

Thank you for using nginx.
</body>
</html>

쿠버네티스 클러스터가 정상적으로 설치된것을 확인함.