# Problem Statement-2

Deploy a local k8s cluster (using minikube, k3s, or anything else) and deploy the DVWA application. Showcase/demo 3 attack surfaces as mentioned in its documentation.

**Solution:**

**This guide outlines the steps needed to set up a Kubernetes cluster using kubeadm.**

## Pre-requisites

Ubuntu OS (Xenial or later)

sudo privileges

Internet access

t2.medium instance type or higher

AWS Setup

Make sure your all instance are in same Security group.

Expose port 6443 in the Security group, so that worker nodes can join the cluster.

## Execute on Both "Master" & "Worker Node"

## Run the following commands on both the master and worker nodes to prepare them for kubeadm.

```
# disable swap

sudo swapoff -a

# Create the .conf file to load the modules at bootup

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br_netfilter

EOF

sudo modprobe overlay

sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
```

```
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF
# Apply sysctl params without reboot
sudo sysctl --system
## Install CRIO Runtime
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg
sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o
sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"
# Add Kubernetes APT repository and install required packages
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
sudo apt-get update -y
sudo apt-get install -y jq
sudo systemctl enable --now kubelet
sudo systemctl start kubelet
```

Execute ONLY on "Master Node"

```
sudo kubeadm config images pull

sudo kubeadm init

mkdir -p "$HOME"/.kube

sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config

sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config

# Network Plugin = calico

kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command
```

You will get kubeadm token, Copy it.

## Execute on ALL of your Worker Node's

Perform pre-flight checks

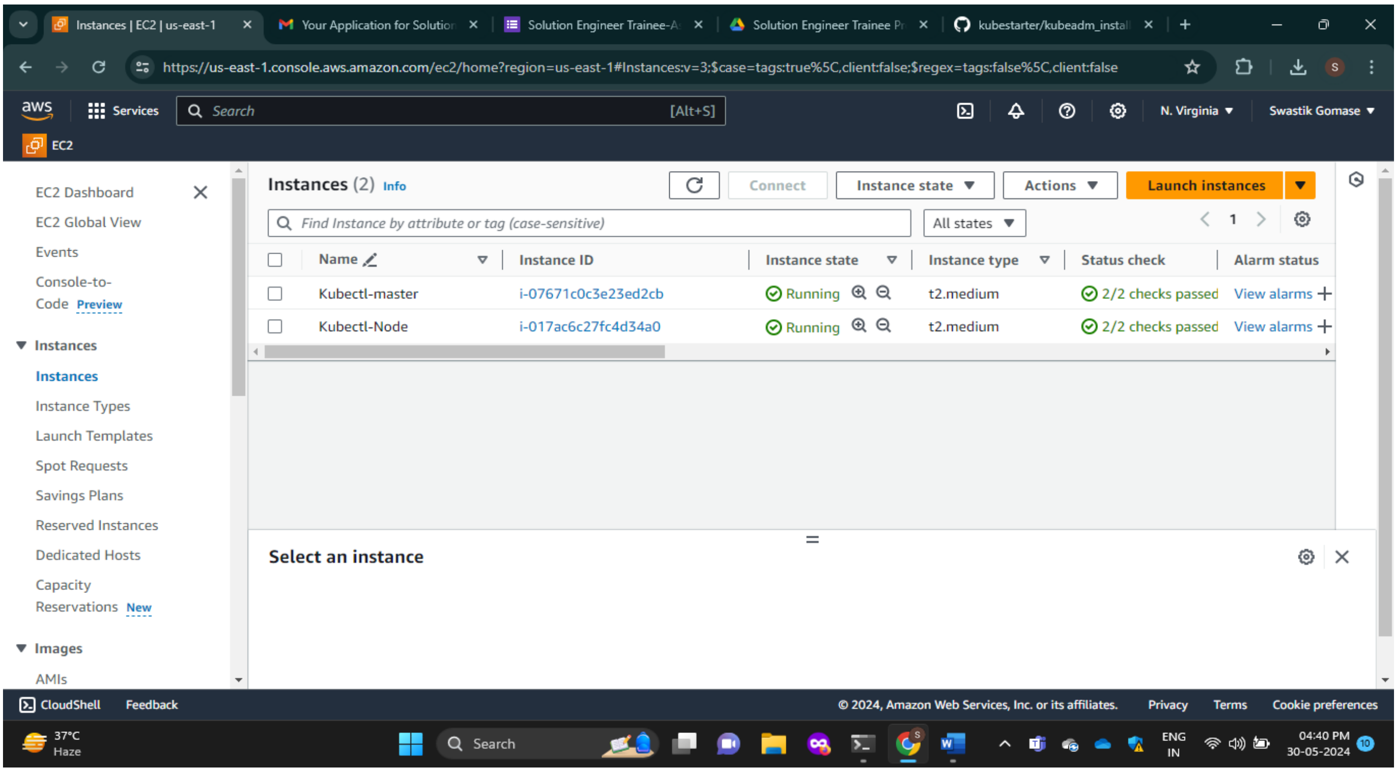
```
sudo kubeadm reset pre-flight checks
```

Paste the join command you got from the master node and append --v=5 at the end.

```
sudo your-token --v=5
```

Use sudo before the token

## Below attached ScreenShot of this Deployment. (EC2 MACHINE)



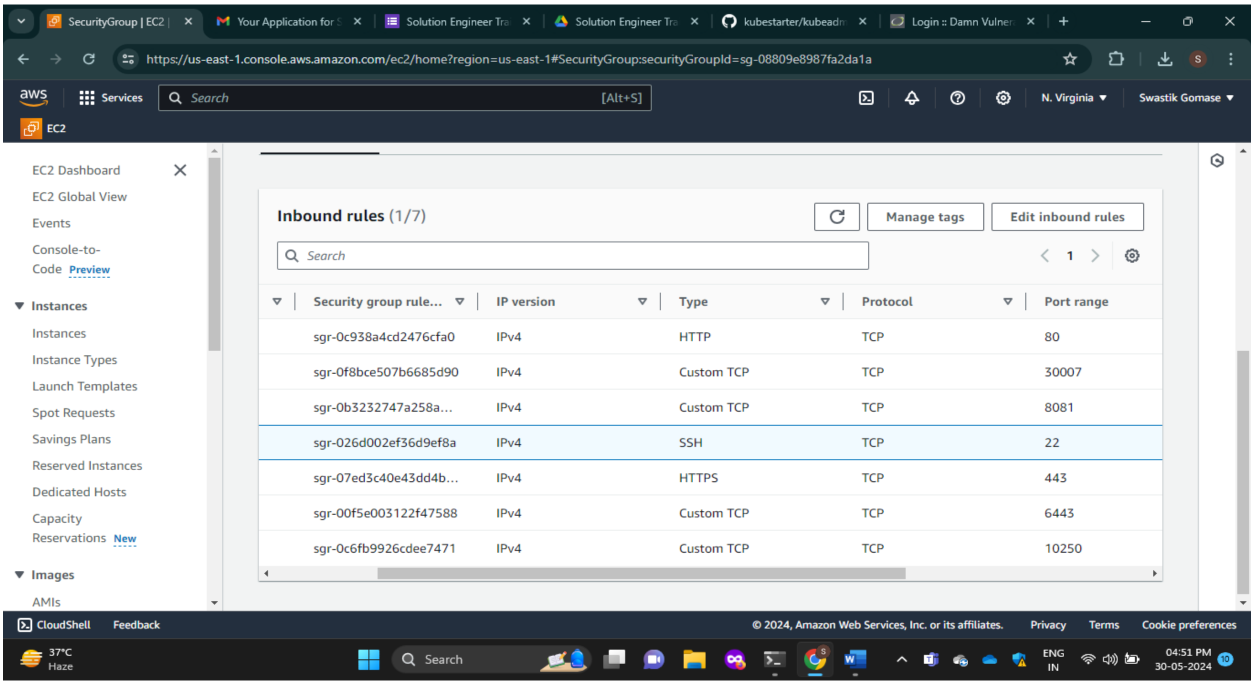
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false

Instances (2)

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status |
|---|---|---|---|---|---|
| Kubectl-master | i-07671c0c3e23ed2cb | Running | t2.medium | 2/2 checks passed | View alarms |
| Kubectl-Node | i-017ac6c27fc4d34a0 | Running | t2.medium | 2/2 checks passed | View alarms |

## Security Groups On Worker Node.

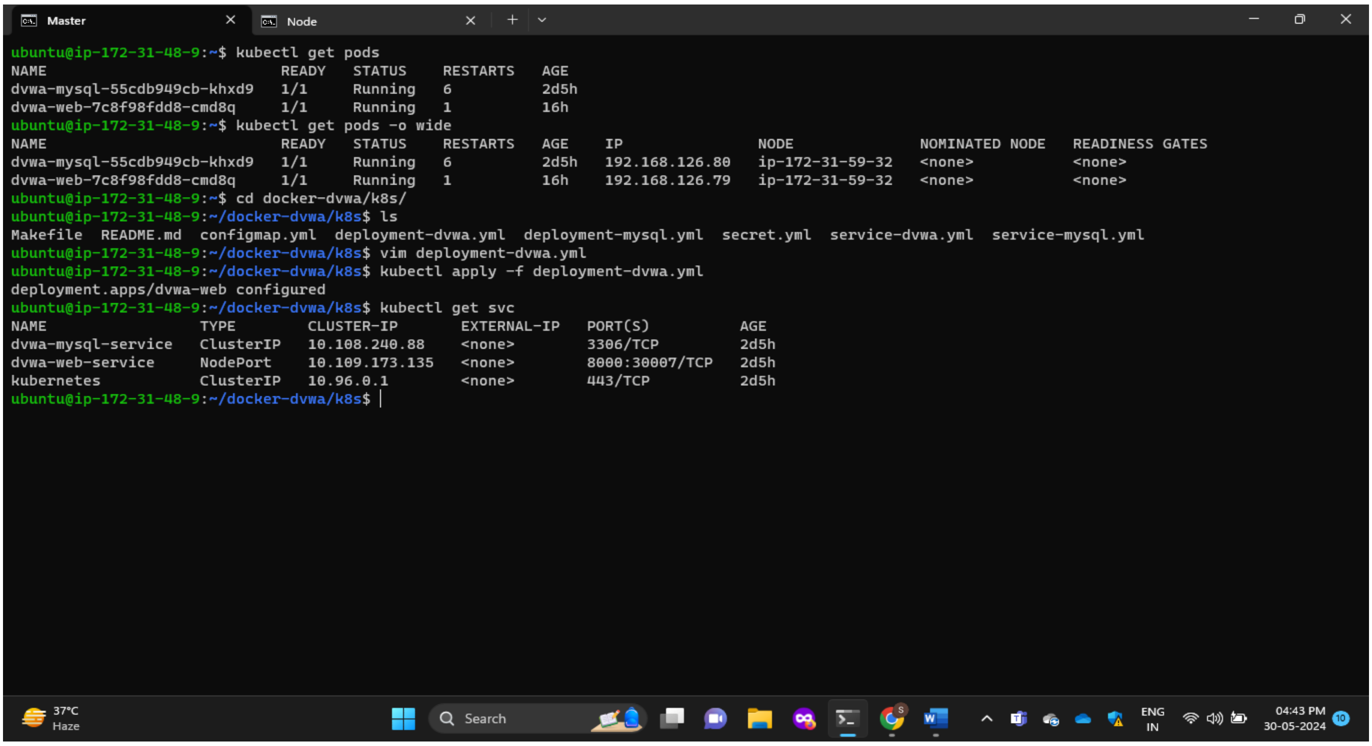
## Port 30007 is open because I access this application through NodePort service,so 30007 to container port no i.e 8000 or 8081



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:securityGroupId=sg-08809e8987fa2da1a

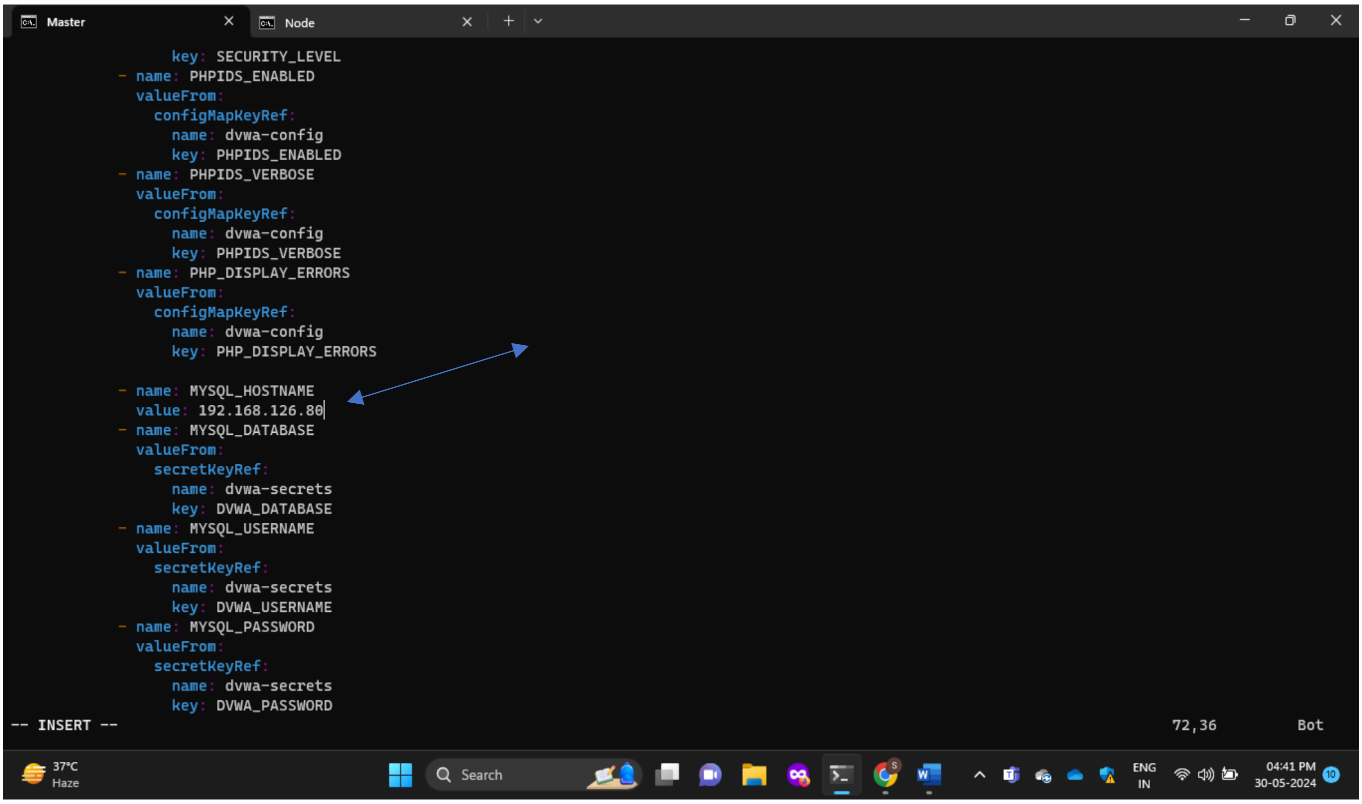
Inbound rules (1/7)

| Security group rule... | IP version | Type | Protocol | Port range |
|---|---|---|---|---|
| sgr-0c938a4cd2476cfa0 | IPv4 | HTTP | TCP | 80 |
| sgr-0f8bce507b6685d90 | IPv4 | Custom TCP | TCP | 30007 |
| sgr-0b3232747a258a... | IPv4 | Custom TCP | TCP | 8081 |
| sgr-026d002ef36d9ef8a | IPv4 | SSH | TCP | 22 |
| sgr-07ed3c40e43dd4b... | IPv4 | HTTPS | TCP | 443 |
| sgr-00f5e003122f47588 | IPv4 | Custom TCP | TCP | 6443 |
| sgr-0c6fb9926cdee7471 | IPv4 | Custom TCP | TCP | 10250 |



```
ubuntu@ip-172-31-48-9:~$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
dvwa-mysql-55cdb949cb-khxd9       1/1     Running   6          2d5h
dvwa-web-7c8f98fdd8-cmd8q         1/1     Running   1          16h
ubuntu@ip-172-31-48-9:~$ kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE    IP                NODE              NOMINATED NODE   READINESS GATES
dvwa-mysql-55cdb949cb-khxd9       1/1     Running   6          2d5h   192.168.126.80    ip-172-31-59-32   <none>           <none>
dvwa-web-7c8f98fdd8-cmd8q         1/1     Running   1          16h    192.168.126.79    ip-172-31-59-32   <none>           <none>
ubuntu@ip-172-31-48-9:~$ cd docker-dvwa/k8s/
ubuntu@ip-172-31-48-9:~/docker-dvwa/k8s$ ls
Makefile  README.md  configmap.yml  deployment-dvwa.yml  deployment-mysql.yml  secret.yml  service-dvwa.yml  service-mysql.yml
ubuntu@ip-172-31-48-9:~/docker-dvwa/k8s$ vim deployment-dvwa.yml
ubuntu@ip-172-31-48-9:~/docker-dvwa/k8s$ kubectl apply -f deployment-dvwa.yml
deployment.apps/dvwa-web configured
ubuntu@ip-172-31-48-9:~/docker-dvwa/k8s$ kubectl get svc
NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
dvwa-mysql-service   ClusterIP   10.108.240.88    <none>        3306/TCP         2d5h
dvwa-web-service     NodePort    10.109.173.135   <none>        8000:30007/TCP   2d5h
kubernetes           ClusterIP   10.96.0.1        <none>        443/TCP          2d5h
ubuntu@ip-172-31-48-9:~/docker-dvwa/k8s$ 
```

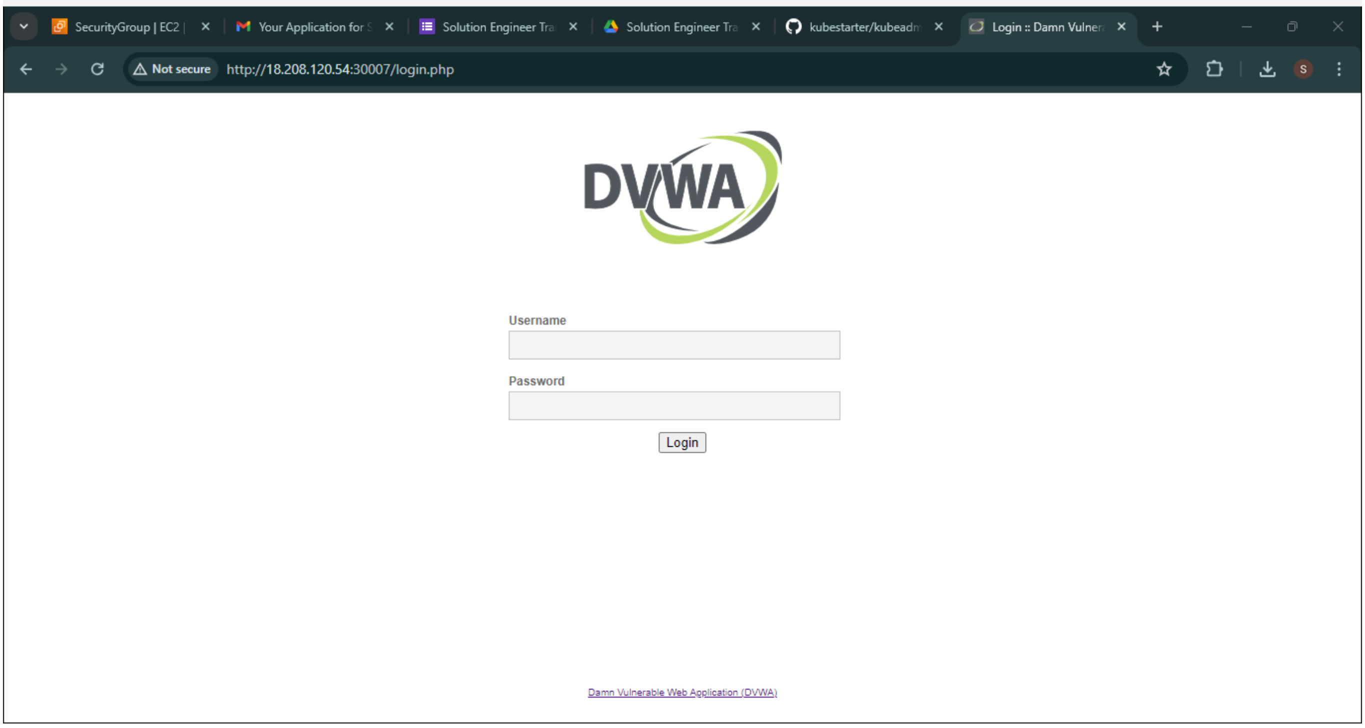
## ADDED IP ADDRESS OF MYSQL POD IN DEPLOYMENT.YAML



```
                  key: SECURITY_LEVEL
            - name: PHPIDS_ENABLED
              valueFrom:
                configMapKeyRef:
                  name: dvwa-config
                  key: PHPIDS_ENABLED
            - name: PHPIDS_VERBOSE
              valueFrom:
                configMapKeyRef:
                  name: dvwa-config
                  key: PHPIDS_VERBOSE
            - name: PHP_DISPLAY_ERRORS
              valueFrom:
                configMapKeyRef:
                  name: dvwa-config
                  key: PHP_DISPLAY_ERRORS

            - name: MYSQL_HOSTNAME
              value: 192.168.126.80
            - name: MYSQL_DATABASE
              valueFrom:
                secretKeyRef:
                  name: dvwa-secrets
                  key: DVWA_DATABASE
            - name: MYSQL_USERNAME
              valueFrom:
                secretKeyRef:
                  name: dvwa-secrets
                  key: DVWA_USERNAME
            - name: MYSQL_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: dvwa-secrets
                  key: DVWA_PASSWORD
-- INSERT --                                                                                                                        72,36         Bot
```

THIS IS FINAL DEPLOYMENT OF APPLICATION HOSTED ON WORKER NODE BY USING KUBEADM.


SecurityGroup | EC2 |
Your Application for S
Solution Engineer Tra
Solution Engineer Tra
kubestarter/kubeadm
Login :: Damn Vulnera
Not secure
http://18.208.120.54:30007/login.php
DVWA
Username
Password
Login
Damn Vulnerable Web Application (DVWA)