## Attack-Simulator Deployment Guide v1.27

This document is divided into two parts:

* **Docker Local Deployment**: In this mode, the container runs as a standalone web service but **cannot** interact with any Kubernetes cluster, thus it cannot execute kubectl commands.
* **Kubernetes Cluster Deployment**: In this mode, we will create a Service Account with the highest cluster permissions for the container, enabling it to execute kubectl commands from within the Pod to operate on the entire cluster.

### ⚠️ Important Security Warning

In the Docker local deployment section, we will grant the user permission to execute any command inside the container. This means the container could be used for any dangerous operations. In the Kubernetes deployment section, we will grant the container **cluster-admin (Cluster Administrator) permissions**. This is the highest level of permission in Kubernetes, meaning the application inside the container can perform **unrestricted operations on the cluster**, including but not limited to:

* Viewing all Secrets (keys) in the cluster.
* Deleting any namespaces, nodes, or workloads.
* Modifying cluster configurations, which could lead to a cluster crash.
* Deploying cryptocurrency miners or other malware.

Please be sure to use this deployment scenario **only in completely isolated, non-production test clusters**. Do not use this configuration in any production or critical environment!

### Part 1: Docker Local Deployment

In this mode, the container runs only as a standard web application, listening on port 8080. It does not have the credentials or network environment to access the Kubernetes API.

#### Deployment Steps

1. Pull the Docker Image  
   Open your terminal and execute the following command to pull the specified image:  
   Bash  
   docker pull paliguoqing/attack-simulator:v1-1750402414
2. Run the Container  
   Use the following command to run the container and map the container's port 8080 to your host's port 8080:  
   Bash  
   docker run -d --name attack-simulator-local -p 8080:8080 paliguoqing/attack-simulator:v1-1750402414  
   * -d: Run the container in the background (detached mode).
   * --name attack-simulator-local: Assign an easy-to-identify name to the container.
   * -p 8080:8080: Map port 8080 of the host to port 8080 of the container.

#### Usage and Verification

1. Check Container Status  
   Confirm the container is running:  
   Bash  
   docker ps  
     
   You should see a container named attack-simulator-local running.
2. Access the Web Service  
   Use curl or a browser to access the local port 8080:  
   Bash  
   curl http://localhost:8080  
     
   If the service is running correctly, you should receive a response from the application. Double-click the text at the bottom of the login page, and the system will automatically fill in the username and password.

### Part 2: Kubernetes Cluster Deployment

In this mode, we will create a Deployment and assign it a ServiceAccount that has cluster administrator permissions.

#### Deployment Steps

To achieve the simplest deployment, we will merge all Kubernetes resources (ServiceAccount, ClusterRoleBinding, Deployment, Service) into a single YAML file.

1. Create the Deployment File  
   Create a file named attack-simulator-k8s.yaml and copy the following content into it.  
   YAML  
   # attack-simulator-k8s.yaml  
     
   # 1. Define a ServiceAccount for our Pod to use  
   apiVersion: v1  
   kind: ServiceAccount  
   metadata:  
    name: attack-sa  
    namespace: default  
   ---  
   # 2. Create a ClusterRoleBinding to bind the built-in 'cluster-admin' role   
   # (the highest permission role) to the 'attack-sa' ServiceAccount we just created.   
   # This is the key step to grant maximum permissions.  
   apiVersion: rbac.authorization.k8s.io/v1  
   kind: ClusterRoleBinding  
   metadata:  
    name: attack-sa-cluster-admin-binding  
   subjects:  
   - kind: ServiceAccount  
    name: attack-sa # Name of the ServiceAccount  
    namespace: default # The namespace where the ServiceAccount resides  
   roleRef:  
    kind: ClusterRole  
    name: cluster-admin # Reference the built-in cluster-admin role in Kubernetes  
    apiGroup: rbac.authorization.k8s.io  
   ---  
   # 3. Create a Deployment to run the attack-simulator container  
   apiVersion: apps/v1  
   kind: Deployment  
   metadata:  
    name: attack-simulator-deployment  
    labels:  
    app: attack-simulator  
   spec:  
    replicas: 1  
    selector:  
    matchLabels:  
    app: attack-simulator  
    template:  
    metadata:  
    labels:  
    app: attack-simulator  
    spec:  
    # Specify that the Pod should use the ServiceAccount we created and authorized  
    serviceAccountName: attack-sa  
    containers:  
    - name: attack-simulator  
    image: paliguoqing/attack-simulator:v1-1750402414  
    ports:  
    - containerPort: 8080  
   ---  
   # 4. Create a Service to expose the Deployment's port 8080.  
   # Using NodePort type for easy access from outside the cluster.  
   apiVersion: v1  
   kind: Service  
   metadata:  
    name: attack-simulator-service  
   spec:  
    type: NodePort # Use NodePort type; you can also use LoadBalancer if on a cloud provider  
    selector:  
    app: attack-simulator  
    ports:  
    - protocol: TCP  
    port: 8080 # The port the Service listens on internally  
    targetPort: 8080 # The port exposed by the container  
    # nodePort: 30080 # You can leave this unspecified, and K8s will automatically assign a port
2. Apply the Deployment File  
   In your terminal, use kubectl to apply this YAML file to create all the resources:  
   Bash  
   kubectl apply -f attack-simulator-k8s.yaml  
     
   After the command is executed, Kubernetes will create the ServiceAccount, ClusterRoleBinding, Deployment, and Service in sequence.

#### Usage and Verification

1. Check Pod and Service Status  
   Wait a few seconds for the Pod to start, then check the status:  
   Bash  
   # Check if the Pod is running (STATUS should be Running)  
   kubectl get pods -l app=attack-simulator  
     
   # Check the Service and note the NodePort  
   kubectl get service attack-simulator-service  
     
   The output of get service will look something like this:  
   NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
   attack-simulator-service NodePort 10.101.88.123 <none> 8080:3xxxx/TCP 1m  
     
   Note the 3xxxx port number in the PORT(S) column; this is the NodePort.
2. Access the Web Service  
   Get the IP address of any of your cluster nodes, and then access the service via NodeIP:NodePort.  
   Bash  
   # Get node IP (choose one)  
   kubectl get nodes -o wide  
     
   # Assuming the node IP is 192.168.1.100 and the NodePort is 31234  
   curl http://192.168.1.100:31234  
     
   Double-click the text at the bottom of the login page, and the system will automatically fill in the username and password. Congratulations, you have successfully deployed an attack-simulator Pod with cluster administrator privileges. Please remember to clean up these resources after you finish testing.

### Cleanup Deployment

After testing is complete, use the YAML file you created to delete all related resources. Please do not expose and run the service on the public internet for extended periods to ensure the security of the test environment:

Bash

kubectl delete -f attack-simulator-k8s.yaml