



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE

## INF6103 : Cybersécurité des Infrastructures Critiques

---

TP 1 : Cybersécurité des infrastructures critiques infonuagiques

---

## Introduction

De plus en plus d'entreprises remplacent des serveurs locaux pour confier la gestion de leurs données à l'infonuagique. Cette tendance s'étend aujourd'hui à de nombreuses infrastructures critiques. A titre d'exemple, on peut citer le secteur maritime qui recommande l'utilisation du PPU (Portable Pilot Unit). Un PPU permet à un pilote ayant en charge un navire, d'accéder dans le nuage, à l'ensemble des données de navigation relatives à sa mission. Le PPU représente ainsi un excellent système d'aide au pilotage, notamment dans des conditions de visibilité limitée. Plus d'information ici.

Dans le secteur médical, l'utilisation de l'infonuagique se développe également. Dans les systèmes d'imagerie médicale basés sur l'infonuagique, les fonctions de stockage des dossiers médicaux et de traitement des images sont fournies par des serveurs externalisés dans le nuage. L'infonuagique apporte de nouvelles solutions aux besoins de stockage croissants de données des centres de radiologie et remplace les serveurs PACS (Picture Archiving and Communication System) locaux et jusque-là confinés dans les établissements de radiologie. Plus d'information ici.

Cependant, l'utilisation de l'infonuagique dans des secteurs critiques n'est pas sans risque. Le manque ou l'absence de sécurité peut créer des brèches dont l'exploitation peut avoir des conséquences catastrophiques. A titre d'exemple, l'incident survenu le 5 février 2021 dans la centrale de traitement de l'eau de la ville de Oldsmar en Floride. Un individu non identifié a tenté de changer le dosage d'hydroxyde de sodium, ce qui aurait eu pour conséquence d'empoisonner l'eau si la malveillance n'avait pas été détectée par les personnels de la centrale. L'attaquant a profité d'une mauvaise gestion des mots de passe, de l'absence de pare-feu et d'un système d'exploitation dépassé reposant sur Windows 7. Il a ainsi pu accéder au système SCADA de la centrale en passant par le logiciel de télémaintenance TeamViewer. Plus d'information ici.

Dans ce TP, vous incarnez tantôt le rôle d'un administrateur d'infrastructure infonuagique, tantôt celui d'un pentester. Hazardous Chemicals Corporation (HCC), une entreprise de fabrication de produits chimique dangereux, a entrepris la migration de son infrastructure vers le nuage informatique. Cette infrastructure est principalement composée de systèmes de contrôle industriel qui font fonctionner les cuves de produits chimiques. Afin d'optimiser les coûts et les performances, HHC opte pour une solution reposant sur des conteneurs et hébergée chez le fournisseur de service Cloud Solutions Provider (CSP). Vous êtes sollicités pour mener à bien le déploiement de l'infrastructure de HHC sur le nuage informatique de CSP.

Le présent TP se concentre sur les aspects de sécurité liés aux applications déployées dans l'infonuagique. Plus spécifiquement, on considère des applications réalisées à l'aide de la technologie de conteneurs. Plus d'informations sur les conteneurs et leur orchestration sont disponibles en-ligne sur :

- <https://www.docker.com/101-tutorial>
- <https://kubernetes.io/docs/concepts/>

Le TP se compose de 3 parties :

1. Prise en main de l'environnement conteneurisé : Docker et Minikube.
2. Sécurité de l'API de l'orchestrateur et contrôle d'accès.
3. Test de pénétration sur l'infrastructure infonuagique.

## Étape Préliminaire

Cette étape doit être effectuée avant la première séance du TP. L'environnement du TP consiste en une machine virtuelle avec Minikube fonctionnant avec le driver Docker :

1. Préparation d'une machine virtuelle :

- 2 vCPUs
- 2 Go de mémoire vive
- 20 Go de disque dur
- 2 interfaces réseaux (NAT et privé hôte)
- Système d'exploitation : Ubuntu Server 22.04.3 LTS
- Installation minimale avec OpenSSH

2. Installation de l'environnement Docker : <https://docs.docker.com/engine/install/ubuntu/> ensuite <https://docs.docker.com/engine/install/linux-postinstall/>

3. Installation de Minikube : <https://minikube.sigs.k8s.io/docs/start/>

4. Installation de Kubectl : <https://kubernetes.io/fr/docs/tasks/tools/install-kubectl/>

5. Test de l'environnement avec `minikube start --driver=docker`.

6. Installation de Skopeo : `sudo apt install skopeo`

NOTE : Pour vous faciliter la tâche, connectez-vous à la machine virtuelle en ssh après l'avoir installé.

## 1 Première partie [13/33]

Cette partie du TP aborde :

- 1.1 La manipulation des conteneurs Docker.
- 1.2 La validation de l'intégrité et authenticité des images Docker.
- 1.3 La manipulation basique d'objets Kubernetes.

### 1.1 Opérations sur des images Docker

La première étape sera de vous assurer que la plateforme de conteneurisation est opérationnelle. Pour cela, utilisez la commande `docker info`.

- 1.1.1 Avant de commencer le déploiement, vous exécutez pour la première fois la commande `docker run alpine`. Que pouvez-vous déduire du résultat de la commande ? [0.5 point]

Votre tâche est maintenant de déployer le serveur de surveillance vidéo des cuves de produits chimiques. Vous devez l'installer à partir du Dockerfile présent dans le dépôt Github suivant : <https://github.com/thibault-leblanc/inf6103>.

- 1.1.2 Expliquez le Dockerfile utilisé pour l'installation. [1 point]
- 1.1.3 Construisez une nouvelle image nommée `hcc` depuis le Dockerfile. [1 point]
- 1.1.4 Déployez le container en ajoutant la capacité `NET_ADMIN` et en transférant le port 80 vers l'hôte. [1 point]
- 1.1.5 Confirmez le fonctionnement des caméras de surveillance depuis votre navigateur. [0.5 point]

### 1.2 Validation de l'intégrité et authenticité des images Docker

Une application conteneurisée présentée sous forme d'image Docker dans un entrepôt doit être protégée contre des tentatives de modification de la part des utilisateurs/développeurs malveillants. Il est crucial d'avoir des outils qui nous permettent de valider l'intégrité et l'authenticité de l'image avant de la lancer. Pour cela, vous allez utiliser l'outil Skopeo qui supporte le format des images Docker et OCI en général.

Après avoir généré l'image du serveur de vidéo surveillance, vous devez maintenant générer sa signature. Cette signature vous permettra de vous assurer que l'image n'a pas été altérée entre le moment de sa génération et son déploiement. Pour cela, vous utiliserez l'outil Skopeo.

- 1.2.1 De quoi avez-vous besoin pour signer les images Docker ? [1 point]
- 1.2.2 Générez une signature associée à l'image du serveur de surveillance vidéo ? [1 point]

HCC a été victime d'une cyberattaque interne, un employé mécontent a eu accès aux images Docker. Les investigateurs n'excluent pas la possibilité qu'il les ai modifiées pour insérer une porte dérobée.

1.2.3 Créez une nouvelle image à partir de l'image du serveur de vidéo surveillance en y installant `netcat` [1 point]

1.2.4 Comment Skopeo peut détecter la modification de l'image du serveur de vidéo surveillance ? [1 point]

### 1.3 Opérations dans le cluster Minikube

Avec l'augmentation du nombre de clients de CSP, la gestion des conteneurs avec Docker devient difficile. L'entreprise décide donc d'utiliser Kubernetes pour orchestrer le déploiement des conteneurs.

Dans cette partie, vous utiliserez `kubectl` pour interagir avec le plan de contrôle de Kubernetes. Vous allez commencer par déployer un objet de type Deployment avec l'image d'un serveur web Nginx et l'exposer. Éliminez le conteneur déployé avec Docker dans la partie précédente avant d'exécuter ces trois instructions suivantes :

- `kubectl create deployment serveur-web --image=nginx`
- `kubectl expose deployment serveur-web --type=NodePort --port=80`
- `minikube service serveur-web`

1.3.1 De la même manière, déployez un objet Deployment `hcc-video-surveillance` avec l'image que vous avez créée du serveur de vidéo surveillance. Que remarquez-vous ? [1 point]

1.3.2 Identifiez le problème en consultant le fichier log de minikube. [1 point]

1.3.3 Déployez l'objet en évitant l'erreur et assurez-vous qu'il est accessible. [1 point]

La granularité des objets Kubernetes demandant des ressources commence avec le Pod, l'unité la plus basique implémentée par Kubernetes. Un Pod contient un ou plusieurs conteneurs. Lorsqu'il s'agit de multiplier les instances des conteneurs, Kubernetes multiplie les Pods. Pour assurer un certain niveau de disponibilité, une pratique est de déployer des objets de type Replicaset. Par exemple, un Replicaset qui déclare 2 instances d'un même Pod, mène Kubernetes à toujours assurer ce nombre de Pods dans le cluster.

Pour s'assurer de la haute disponibilité du serveur de vidéo surveillance, HCC vous demande d'en créer une copie.

1.3.2 Déployez un Replicaset avec deux répliques du service `hcc-video-surveillance`. [1 point]

1.3.3 Testez la haute disponibilité du service dans le cas de l'arrêt de l'une des deux répliques. [1 point]

## 2 Deuxième Partie [13/33]

### 2.1 TLS en Kubernetes

2.1.1 Comment pouvons-nous vérifier (avec une commande `kubectl` ou `minikube`) que l'API du serveur écoute en TLS (HTTPS) par défaut ? [1 point]

Le client typique pour invoquer l'API de Kubernetes est `kubectl`. Nous allons interagir avec le cluster en utilisant un autre client, tel que `curl`. Sachant que le l'API serveur écoute sur HTTPS, il faut que le client ait le bon certificat pour établir la session TLS. Il s'agit d'obtenir un certificat signé par la CA (Autorité de Certification) propre au cluster.

NOTE : par défaut, Kubernetes utilise une CA interne au cluster pour la gestion des certificats des composants internes de Kubernetes. D'autres options sont disponibles également.

2.1.2 Quel est le certificat de la CA interne du cluster, le certificat du client `minikube` et sa clé étant donnée la commande `kubectl config view` ? [1.5 point]

Inspectez les données antérieures avec `openssl` :

- `openssl x509 -text -noout -in <certificat_client_minikube>`
- `openssl x509 -text -noout -in <certificat_ca_cluster>`
- `openssl rsa -in <clé_du_client> -check`

Nous allons réutiliser les certificats et clé antérieurs avec `curl` mais une conversion au format .pem est peut-être nécessaire. Copiez ces données dans le dossier local et passez à la question suivante.

2.1.3 Convertissez les certificats et clé antérieurs au format .pem en utilisant `openssl` [1 point]

2.1.4 Comment pouvez-vous maintenant interagir avec le cluster en utilisant `curl` et les paramètres "credentials" antérieurs ? Donnez au minimum 1 exemple. [1 point]

---

Deuxième séance de TP

---

### 2.2 RBAC

Pour cette partie, vous êtes amené à configurer un système de contrôle d'accès sur le cluster Kubernetes. HCC fait appel à vous et vous informe des détails de votre mission :

Le système de vidéo surveillance des cuves doit être mis à jour. Maxime, un membre de l'équipe de développement, est en charge de la mise à jour de la partie applicative. Maxime est appelé à intervenir directement sur le cluster Kubernetes pour mener sa mission. Vous devez vous assurer que le développeur exécute ses tâches de manière isolée, sans interférer avec les ressources du cluster auxquelles il ne doit pas avoir accès.

Pour répondre aux besoins de HCC, vous allez implémenter une politique de contrôle d'accès RBAC (Role Based Access Control) pour des utilisateurs devant avoir les mêmes permissions dans

le cluster actuel. On considère un seul utilisateur (maxime) qui se verra attribuer un namespace et travaillera sur son projet dans le cluster. La politique de contrôle d'accès est décrite plus bas.

Vous optez pour une authentification des utilisateurs à l'aide d'un certificat signé par la CA du cluster. Les étapes typiques sont :

1. création d'un utilisateur.
2. obtention des bons credentials (certificats) qui permettent à cet utilisateur d'être authentifié à chaque accès au cluster.
3. création des objets Kubernetes nécessaires pour ce nouvel utilisateur : namespace, Role, RoleBinding, etc., en fonction de la politique visée.

Commencez par créer un namespace "development" pour le nouvel utilisateur :

```
kubectl create ns development
```

2.2.1 Quelle est votre démarche pour :

- a) créer un nouvel utilisateur appelé maxime sur Linux. [0.5 point]
- b) obtenir un certificat signé par l'autorité de certification propre au cluster avec `openssl` ? [1.5 point]

Nous allons configurer les credentials du nouvel utilisateur dans le cluster avec la commande :

```
kubectl config set-credentials maxime --client-certificate=<chemin_vers_maxime.crt>  
--client-key=<chemin_vers_maxime.key>
```

Le résultat doit être : `User "maxime" set`.

Ensuite, nous allons configurer le contexte pour l'utilisateur maxime avec la commande :

```
kubectl config set-context maxime-context --cluster=minikube  
--namespace=development --user=maxime
```

Résultat: `context "maxime-context" created`

2.2.2 Quelle est votre interprétation du résultat des commandes suivantes : [1 points]

- `kubectl --context=maxime-context get pods`
- `kubectl --context=maxime-context get -n kube-system pods`
- `kubectl -n development --context=maxime-context get pod`

La politique de contrôle d'accès RBAC que nous ciblons permettra à l'utilisateur "maxime" de créer, lire, écrire les objets Kubernetes deployment, services, replicaset et pods dans son namespace development. Les permissions en langage Kubernetes sont des "verbs" et nous utiliserons les verbes suivants : ["list", "get", "watch", "create", "update", "patch", "delete"].

La politique implique l'attribution d'un rôle auquel des permissions sont affectées. Un utilisateur est affecté à un rôle et obtient ainsi les permissions de ce rôle. Concrètement, il s'agit de créer un objet Role et un objet RoleBinding. Nous allons décrire ces objets via des fichiers .yaml.

Dans un fichier `role-dev.yaml`, ajoutez :

```
1 kind: Role
2 apiVersion: rbac.authorization.k8s.io/v1
3 metadata:
4   namespace: development
5   name: developer
6 rules:
7 - apiGroups: ["", "extensions", "apps"]
8   resources: ["deployments", "replicasets", "pods"]
9   verbs: ["list", "get", "watch", "create", "update", "patch", "delete"]
10 # You can use ["*"] for all verbs
```

Dans un fichier `rolebind.yaml`, ajoutez :

```
1 kind: RoleBinding
2 apiVersion: rbac.authorization.k8s.io/v1
3 metadata:
4   name: developer-role-binding
5   namespace: development
6 subjects:
7 - kind: User
8   name: maxime
9   apiGroup: ""
10 roleRef:
11   kind: Role
12   name: developer
13   apiGroup: ""
```

Créez par la suite les objets Kubernetes afférents :

- `kubectl create -f role-dev.yaml`
- `kubectl create -f rolebind.yaml`

Testez que le résultat obtenu est conforme à la politique RBAC. Par exemple :

`kubectl -n development --context=maxime-context get pod` (ceci ne devrait plus indiquer une Erreur).

[2.2.3] Continuez en créant un Deployment avec l'image du serveur de vidéo surveillance en tant qu'utilisateur "maxime" dans l'espace "development", et en exposant le port 80. [1 point]

Le service lié au déploiement du serveur de surveillance vidéo doit être exposé. Référez-vous aux opérations sur minikube, Section 1.3. Tentez donc la commande suivante :

```
kubectl --context=maxime-context expose deployment hcc-video-dev --type=NodePort --port=80
```

2.2.4 Comment interprétez-vous le résultat de l'opération antérieure ? [1.5 point]

2.2.5 Comment pouvez-vous résoudre cette situation ? [1.5 point]

2.2.5 Est-ce que le service est devenu accessible ? Si oui, listez les commandes et une capture d'écran avec votre fureteur accédant au service nginx ou avec curl. [1.5 point]

### 3 Troisième Partie [7/33]

Pour renforcer la sécurité de sa nouvelle infrastructure, HCC fait appel à vos services pour effectuer un test de pénétration. Vous convenez du scénario suivant : vous êtes un concurrent de HCC et également client de CSP. Pour vous permettre d'effectuer vos tâches d'administration, CSP vous crée un conteneur administrateur.

- 3.1 Créez un conteneur `ubuntu` administrateur avec l'argument `--privileged` [0.5 point]
- 3.2 Accédez au système de fichiers de la machine hôte depuis le conteneur administrateur. [1.5 points]
- 3.3 Assurez-vous d'avoir un accès permanent et furtif à l'hôte. [1 points]
- 3.4 Modifiez l'image du serveur de vidéo surveillance de HCC sur Kubernetes en y ajoutant un programme malveillant (exemple: `nmmap`). [1 points]
- 3.5 Quelles sont les vulnérabilités que vous avez identifiées ? [1 point]
- 3.6 Quelles contre-mesures recommandez-vous pour empêcher cette attaque ? [1 point]
- 3.7 Implémentez une de ces contre-mesures. [1 point]