

Azure – Plateforme de calcul (virtualisation) - Partie 2

lundi 13 mars 2023 09:13

Formateur : Ihab ABADI

Déploiement de plusieurs conteneurs avec **Docker-Compose** sur Azure :

Manière de déploiement Kubernetes de façon sécurisée : <https://docs.rke2.io/>

Création Registre :

Créer un Registre de conteneurs

Fonctions de base Réseau Chiffrement Étiquettes Vérifier + créer

Azure Container Registry vous permet de générer, stocker et gérer les artefacts et images conteneur dans un registre privé pour tous les types de déploiement de conteneurs. Utilisez les registres de conteneurs Azure avec vos pipelines de développement et de déploiement de conteneurs existants. Utilisez Azure Container Registry Tasks pour générer des images conteneur dans Azure à la demande, ou pour automatiser les builds déclenchées par les mises à jour du code source, les mises à jour de l'image de base d'un conteneur ou les minuteurs. [En savoir plus](#)

Détails du projet

Abonnement * Abonnement Azure 1

Groupe de ressources * m2i-formation

Détails de l'instance

Nom du Registre * m2iformationihab

Emplacement * East US

Zones de disponibilité ☒ Activé

SKU * Premium

Se connecter à Azure depuis VSCode :

```
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker login azure
login succeeded
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker>
```

Il est nécessaire de créer un contexte pour agir sur Azure (choix abonnement/groupe de ressource)

```
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker context create aci aci-ihab-conte
xt
Using only available subscription : Abonnement Azure 1 (c49e632f-5dd4-4c37-b1a6-578c7faa36fd)
? Select a resource group [Use arrows to move, type to filter]
> create a new resource group
m2i-formation (eastus)
```

Lister les contexte :

```
• Successfully created aci context "aci-ihab-context"
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker context ls
NAME                TYPE      DESCRIPTION          DOCKER ENDPOINT
aci-ihab-context    aci       m2i-formation@eastus
default *           moby      Current DOCKER_HOST based configuration   npipe:///./pipe/docker_engine
ws.com (default)    swarm    https://E187D13635E677240D9BD67D661920FC.gr7.us-east-2.eks.amazonaws.com
desktop-linux       moby      npipe:///./pipe/dockerDesktopLinuxEngine
```

Changer de contexte : `docker contexte use <nom_contexte>`

```
• PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker context use aci-ihab-context
aci-ihab-context
```

Tag et build des images :

```
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker tag vote-image m2informationihab.azurecr.io/vote-image
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker>
m2informationihab.azurecr.io/vote-image
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker push m2informationihab.azurecr.io/vote-image
Using default tag: latest
```

Nepas oublier l'activation de l'utilisateur administrateur sur le registry avant le `docker push` :



The screenshot shows a configuration window for a Docker Registry. It includes fields for 'Nom du Registre' (m2informationihab), 'Serveur de connexion' (m2informationihab.azurecr.io), 'Utilisateur administrateur' (a toggle switch set to 'Activé'), 'Nom d'utilisateur' (m2informationihab), and 'Mot de passe' (with a 'Régénérer' button).

Dans le docker compose : il faut changer le **chemin** des images pour que le code fasse appel aux images stocké sur **Azure**

```
interval: "5s"
vote:
  image: m2informationihab.azurecr.io/vote-image
  ports:
    - "4000:80"
  depends_on:
    - redis-network
result:
  image: m2informationihab.azurecr.io/result-image
  ports:
```

Une fois modifié, on exécute la commande `docker-compose up` pour exécuter le déploiement de notre infrastructure

```
accessible. Please check the image and Registry credential.
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker-compose up
[+] Running 0/1
 - Group tpdocker Creating
```

Il faut aussi faire attention à la région où l'on cherche à déployer, on peut rencontrer des problèmes de ressource.

Pour mitiger cet effet, on peut affecter des limitations à nos **Pods**

```
services:
  redis:
    image: redis:5.0-alpine3.10
    networks:
      - redis-network
    # volumes:
    #   - "./redis-check:/redis-check"
    healthcheck:
      test: /redis-check/script.sh
      interval: "5s"
  Deploy:
```

```
resources:
  limits:
    cpus: '0.5'
    memory: 200M
```

Nouvelle tentative de up après les changements

```
'5.1' GB memory 'Linux' OS"
PS C:\Users\Administrateur\Desktop\Azure\aci\TP Docker> docker-compose up
• [+] Running 6/6
  - Group tpdocker Created 3.6s
  - redis Created 93.2s
  - db Created 93.2s
  - vote Created 93.2s
  - result Created 93.2s
  - worker Created 93.2s
```

Historique des commandes utilisé pour ce déploiement :

```
Id CommandLine
--
1 try { . "c:\Users\Administrateur\AppData\Local\Programs\Micros
2 cd '.\aci\TP Docker\'
3 ls
4 docker-compose up
5 docker-compose up --build
6 docker login azure
7 docker context create aci aci-ihab-context
8 docker context ls
9 docker context use aci-ihab-context
10 docker images
11 docker context use default
12 docker images
13 docker build -it vote-image ./vote/.
14 docker build -t vote-image ./vote/.
15 docker tag vote-image m2informationihab.azurecr.io/vote-image
16 docker push m2informationihab.azurecr.io/vote-image
17 docker push m2informationihab.azurecr.io/vote-image
18 docker push m2informationihab.azurecr.io/vote-image
19 docker push m2informationihab.azurecr.io/vote-image
20 docker login m2informationihab.azurecr.io
21 docker login m2informationihab.azurecr.io
22 docker push m2informationihab.azurecr.io/vote-image
23 docker push m2informationihab.azurecr.io/result-image
24 docker push m2informationihab.azurecr.io/worker-image
25 docker context use aci-ihab-context
26 docker-compose up
27 docker-compose up
28 docker-compose up
29 docker-compose up
30 docker-compose up
```

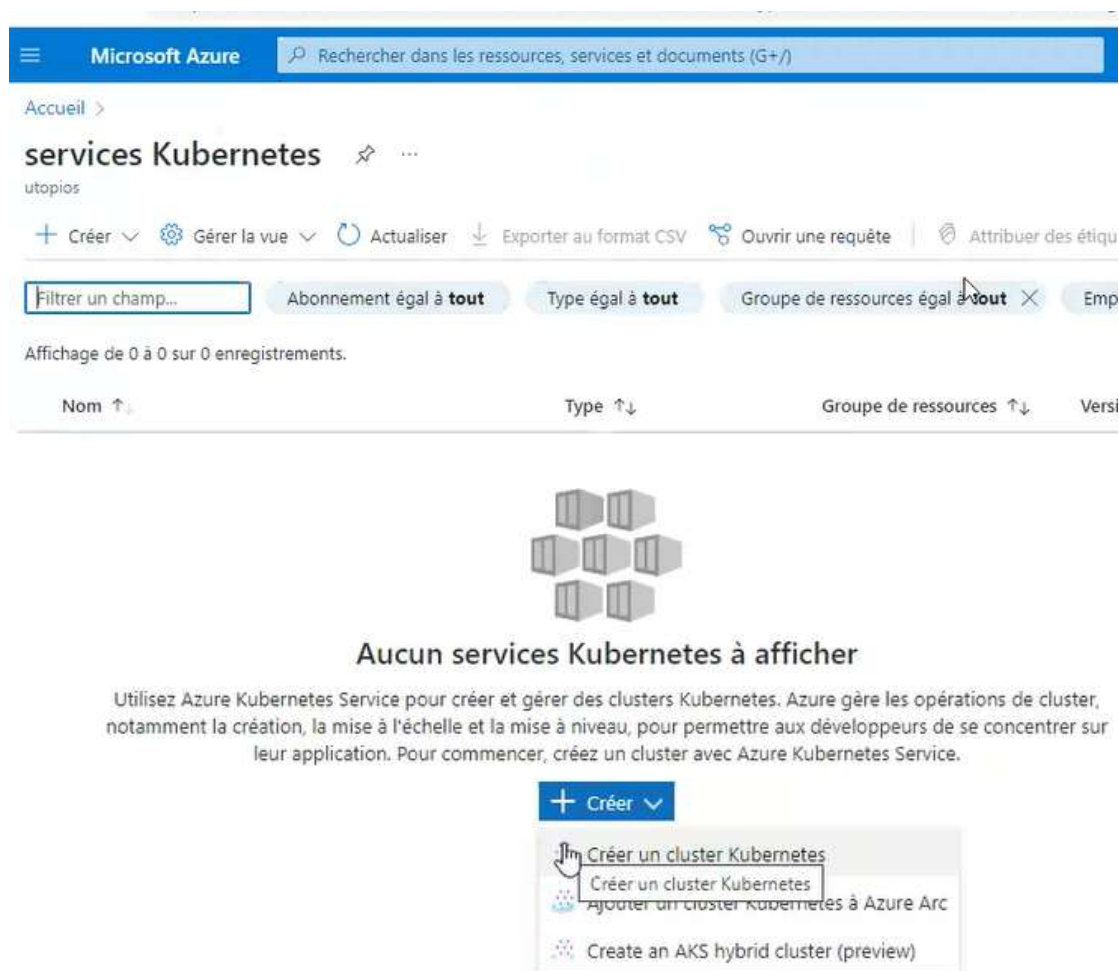
Autre Plateforme de Calcul : AKS

<https://azure.microsoft.com/fr-fr/products/kubernetes-service/>

<https://learn.microsoft.com/fr-fr/azure/aks/>

Azure Kubernetes Service (AKS) offre le moyen le plus rapide de commencer à développer et à déployer des applications natives cloud dans Azure, dans les centres de données ou à la périphérie, avec des pipelines et des protection code-à-cloud intégrés. Bénéficiez d'une gestion et d'une gouvernance unifiées pour les clusters Kubernetes locaux, en périphérie et multiclouds. Interagissez avec les services Azure de sécurité, d'identité, de gestion des coûts et de migration.

Pour créer notre cluster : menu services Kubernetes



Microsoft Azure

Rechercher dans les ressources, services et documents (G+)

Accueil >

services Kubernetes

utopios

+ Créer ▼ ⚙️ Gérer la vue ▼ ↻ Actualiser ⬇ Exporter au format CSV 🔗 Ouvrir une requête 🏷 Attribuer des étiquettes

Filtrer un champ... Abonnement égal à tout Type égal à tout Groupe de ressources égal à tout Emp

Affichage de 0 à 0 sur 0 enregistrements.

Nom ↑	Type ↑↓	Groupe de ressources ↑↓	Versi
-------	---------	-------------------------	-------

Aucun services Kubernetes à afficher

Utilisez Azure Kubernetes Service pour créer et gérer des clusters Kubernetes. Azure gère les opérations de cluster, notamment la création, la mise à l'échelle et la mise à niveau, pour permettre aux développeurs de se concentrer sur leur application. Pour commencer, créez un cluster avec Azure Kubernetes Service.

+ Créer ▼

- Créer un cluster Kubernetes
- Créer un cluster Kubernetes
- Ajouter un cluster Kubernetes à Azure Arc
- Create an AKS hybrid cluster (preview)

Création du projet avec les différents menus de configuration :

Détails du projet

Sélectionnez un abonnement pour gérer les coûts et les ressources déployées. Utilisez les groupes de ressources comme des dossiers pour organiser et gérer toutes vos ressources.

Abonnement * ⓘ

Abonnement Azure 1 ▼

Groupe de ressources * ⓘ

m2i-formation ▼

[Créer nouveau](#)

Détails du cluster

Configuration prédéfinie du cluster

Nom du cluster Kubernetes * ⓘ

Région * ⓘ

Zones de disponibilité ⓘ

AKS pricing tier ⓘ

Version de Kubernetes * ⓘ

Automatic upgrade ⓘ

Free

The cluster management is free, but you'll be charged for VM, storage, and networking usage. Best for experimenting, learning, simple testing, or workloads with fewer than 10 nodes.

Standard

Recommended for mission-critical and production workloads. Includes Kubernetes control plane autoscaling, workload-intensive testing, and up to 5,000 nodes per cluster. Uptime SLA is 99.95% for clusters using Availability Zones and 99.9% for clusters not using Availability Zones.

Free ▼

1.24.9 (par défaut) ▼

Enabled with patch (recommended) ▼

Pool de nœuds principal

Nombre et taille des nœuds dans le pool de nœuds principal de votre cluster. Pour les charges de travail de production, il est recommandé d'avoir au moins 3 nœuds à des fins de résilience. Pour les charges de travail de développement ou de test, un seul nœud est nécessaire. Si vous souhaitez ajouter des pools de nœuds supplémentaires ou afficher d'autres options de configuration pour ce pool de nœuds, accédez à l'onglet « Pools de nœuds » ci-dessus. Vous pourrez ajouter des pools de nœuds une fois votre cluster créé. [En savoir plus sur les pools de nœuds dans Azure Kubernetes Service](#)

Taille de nœud * ⓘ

Standard D2s v3

2 processeurs virtuels, 8 Gio de mémoire

▲ B4ms Standard est recommandée pour la configuration dev/test.

[Changer la taille](#)

Méthode de mise à l'échelle * ⓘ

☐ Manuel

☒ Mise à l'échelle automatique

▲ La mise à l'échelle automatique est recommandée pour la configuration de développement/test.

Plage du nombre de nœuds * ⓘ

1



5

De base **Pools de nœuds** Accès Mise en réseau Intégrations Avancé Étiquettes Vérifier + créer

Pools de nœuds

En plus du pool de nœuds principal nécessaire configuré sous l'onglet Informations de base, vous pouvez également ajouter des pools de nœuds facultatifs pour gérer diverses charges de travail [En savoir plus sur les pools de nœuds](#)

+ Ajouter un pool de nœuds Supprimer

Nom	Mode	Type de système d'e...	Nombre de nœuds	Taille de nœud
<input type="checkbox"/> agentpool	Système	Linux	1-5	Standard_D2s_v3

Activer les nœuds virtuels

Les nœuds virtuels permettent une mise à l'échelle expansible reposant sur des instances de conteneur Azure serverless.

[En savoir plus sur les nœuds virtuels](#)

Activer les nœuds virtuels ⓘ

☐

Chiffrement de disque de système d'exploitation du pool de nœuds

Par défaut, tous les disques sont chiffrés au repos dans AKS avec des clés gérées par Microsoft. Pour mieux contrôler le chiffrement, vous pouvez fournir vos propres clés à l'aide d'un jeu de chiffrement de disque issu d'Azure Key Vault. Le jeu de chiffrement de disque est utilisé pour chiffrer les disques de système d'exploitation de tous les pools de nœuds du cluster.

[En savoir plus](#)

Type de chiffrement

(Par défaut) Chiffrement au repos avec une clé gérée par la plateforme



Vous pouvez changer les paramètres réseau de votre cluster, notamment en activant le routage des applications HTTP et en configurant votre réseau à l'aide des options « Kubenet » ou « Azure CNI » :

- Le plug-in de réseau « **kubenet** » crée un réseau virtuel pour votre cluster à l'aide des valeurs par défaut.
- Le plug-in de réseau « **Azure CNI** » permet aux clusters d'utiliser un réseau virtuel nouveau ou existant avec des adresses personnalisables. Les pods d'application sont connectés directement au réseau virtuel, ce qui permet une intégration native aux fonctionnalités de réseau virtuel.

[En savoir plus sur les réseaux dans Azure Kubernetes Service](#)

Configuration réseau ⓘ

☒ Kubenet

☐ Azure CNI

Préfixe du nom DNS * ⓘ

benoit-test-dns



Routage du trafic

Équilibreur de charge ⓘ

Standard

Activer le routage d'application HTTP ⓘ



Sécurité

Activer le cluster privé ⓘ

☐

Définir des plages d'adresses IP autorisées ⓘ ☐

Stratégie réseau ⓘ

☒ Aucun

☐ Calico

☐ Azure

i La stratégie réseau Azure n'est pas compatible avec le réseau kubernetes.

On connecte notre Kubernetes à notre registry :

De base Pools de nœuds Accès Mise en réseau **Intégrations** Avancé Étiquettes Vérifier + créer

Connectez votre cluster AKS avec des services supplémentaires.

Microsoft Defender for Cloud
Microsoft Defender for Cloud provides unified security management and advanced threat protection across hybrid cloud workloads. [Learn more](#) ⓘ

✔ Your subscription is protected by Microsoft Defender for Cloud basic plan.

Azure Container Registry
Connectez votre cluster à un registre de conteneurs Azure pour activer des déploiements sans interruption à partir d'un registre d'images privé. Vous pouvez créer un registre ou en choisir un que vous avez déjà. [En savoir plus sur Azure Container Registry](#) ⓘ

Registre de conteneurs ▼

[Créer](#)

Confirmation de notre configuration avant déploiement :

✔ Validation réussie

De base Pools de nœuds Accès Mise en réseau Intégrations



De base

Abonnement	Abonnement Azure 1
Groupe de ressources	m2i-formation
Région	Japan East
Nom du cluster Kubernetes	benoit-test-kubernetes
Version de Kubernetes	1.24.9
Automatic upgrade	Correctif

Pools de nœuds

Pools de nœuds	1
Activer les nœuds virtuels	Désactivé

✔ **Votre déploiement a été effectué**

	Nom du déploiement : microsoft.aks-20230313102241	Heure de début : 13/03/2023 10:33:11
	Abonnement : Abonnement Azure 1	ID de corrélation : 7b5fd4c8-5267-4bbe-8de6-2a6d60835ac4 
	Groupe de ressources : m2i-formation	

▼ **Détails du déploiement**

Étapes suivantes

- Créer une application de démarrage rapide Recommandé
- Créer un déploiement Kubernetes Recommandé
- Intégrer des déploiements automatiques dans votre cluster Recommandé
- Se connecter au cluster Recommandé

[Accéder à la ressource](#)
[Se connecter au cluster](#)

Envoyer des commentaires

[Partagez votre expérience avec le déploiement](#)

Se connecter au cluster :

Se connecter à benoit-test-kubernetes

Connectez-vous à votre cluster à l'aide de l'outil en ligne de commande

- Ouvrir Cloud Shell ou Azure CLI
- Exécuter les commandes suivantes

Vérifier les nodes :

```
PS C:\Atelier\Azure> kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
aks-agentpool-11418744-vmss000000 Ready    agent    25m   v1.24.9
PS C:\Atelier\Azure>
```

Création pods :

```
PS C:\Users\Administrateur\Desktop\Azure\aks> kubectl create -f .\pod.yml
pod/debug created
PS C:\Users\Administrateur\Desktop\Azure\aks> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
debug     1/1     Running   0           10s
PS C:\Users\Administrateur\Desktop\Azure\aks>
```

Créer Supprimer Actualiser Afficher les étiquettes Envoyer des commentaires			
Déploiements Pods Jeux de réplicas Objets StatefulSet Jeux de démons Travaux Tâches cron			
Filtrer par nom de déploiement <input type="text" value="Entrer le nom de déploiement complet"/>		Filtrer par espace de noms <input type="text" value="Tous les espaces de noms"/>	
		<input type="button" value="Add label filter"/>	
<input type="checkbox"/>	Nom	Espace de noms	Prêt
<input type="checkbox"/>	coredns	kube-system	✓ 2/2
<input type="checkbox"/>	coredns-autoscaler	kube-system	✓ 1/1
<input type="checkbox"/>	konnectivity-agent	kube-system	✓ 2/2
<input type="checkbox"/>	metrics-server	kube-system	✓ 2/2
<input type="checkbox"/>	azure-policy	kube-system	✓ 1/1
<input type="checkbox"/>	azure-policy-webhook	kube-system	✓ 1/1
<input type="checkbox"/>	gatekeeper-audit	gatekeeper-system	✓ 1/1
<input type="checkbox"/>	gatekeeper-controller	gatekeeper-system	✓ 2/2

Création du fichier `deployment.yml` pour générer nos `nodes` sur Azure

```
aks > api-python > k8s-resources > ! deployment.yml > ...
```

```

1  apiVersion: apps/v1
2  kind: deployment
3  meta-data:
4    name: api-python-deployment
5    labels:
6      name: api-python-deployment

```

```

spec:
  selector:
    matchLabels:
      app: api-python-deployment
  template:
    meta-data:
      labels:
        app: api-python-deployment
    spec:
      containers:
        - name: api-python
          image: m2information.azurecr.io/api-python
          resources:
            limits:
              memory: "128Mi"
              cpu: "200m"

```

Push de l'image :

```

on: Resolving host m2information.azurecr.io: lookup m2information.azurecr.io: no such host
PS C:\Users\Administrateur\Desktop\Azure> docker tag api-python m2informationhab.azurecr.io/api-python
PS C:\Users\Administrateur\Desktop\Azure> docker push m2informationhab.azurecr.io/api-python
Using default tag: latest
The push refers to repository [m2informationhab.azurecr.io/api-python]

```

Création des pods avec le fichier `deployment.yml`

```

PS C:\Users\Administrateur\Desktop\Azure\aks\api-python> kubectl create -f .\k8s-resources\deployment.yml
deployment.apps/api-python-deployment created
PS C:\Users\Administrateur\Desktop\Azure\aks\api-python> kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
api-python-deployment-6c9c549874-kh8sm 0/1     ContainerCreating   0           13s
debug                                  1/1     Running             0           51m
PS C:\Users\Administrateur\Desktop\Azure\aks\api-python> kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
api-python-deployment-6c9c549874-kh8sm 0/1     ContainerCreating   0           17s
debug                                  1/1     Running             0           52m
PS C:\Users\Administrateur\Desktop\Azure\aks\api-python>

```

Création du fichier `service.yml` :

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: service-api-python
5  spec:
6    type: NodePort
7    selector:
8      app: api-python-deployment
9    ports:
10     - port: 80
11     targetPort: 8080

```



```
PS C:\Users\Administrateur\Desktop\Azure\aks\api-python> kubectl create -f .\k8s-resources\service.yml
service/service-api-python created
```

(pour avoir l'adresse IP externe : il a fallu passer le type en [LoadBalancer](#))

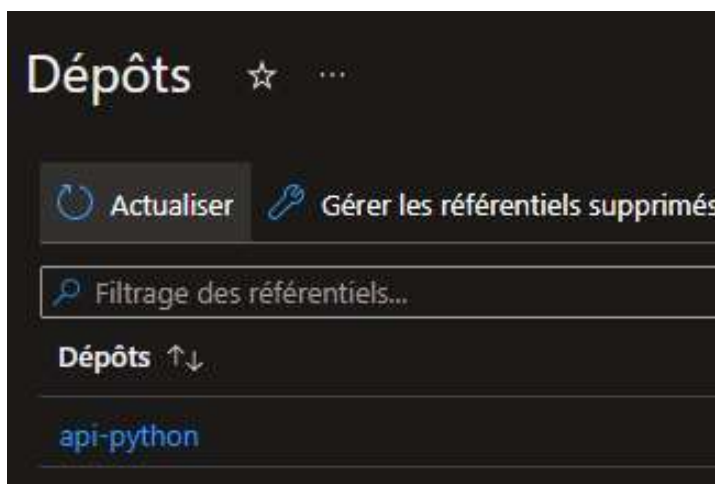


Exercice : Déploiement de l'application sur nos propres cluster AKS :

Connexion à mon repository sur Azure et push de l'image récupérée sur le dockerFiles existant dans l'application :

<input type="checkbox"/>	Name ↑	Tag	Status	Created
<input type="checkbox"/>	api-python 6eb2e772d6eb	latest	Unused	less than a minute a

```
PS C:\Atelier\Azure\AKS - Kubernetes\api-python> docker login benoitm2iregistry.azurecr.io
Username: benoitm2iregistry
Password:
Login Succeeded
PS C:\Atelier\Azure\AKS - Kubernetes\api-python> docker push benoitm2iregistry.azurecr.io/api-python
Using default tag: latest
The push refers to repository [benoitm2iregistry.azurecr.io/api-python]
bc4ee8855244: Pushing [=====>] 12.41MB
d8eb378e2176: Pushed
e029d074b8ca: Pushed
28fa86444e63: Pushing [=====>] 10.77MB
85158601c190: Pushed
0cdca02a73c8: Pushing [=====>] 27.79MB/46.45MB
0baa05faf31a: Pushing [=====>] 19.05MB
ac504d030134: Pushing [=====>] 130.5MB/528.8MB
52ebb9a789db: Waiting
86d774dafc20: Waiting
da68f13652fc: Waiting
cf2e8433dbf2: Waiting
```



Création Pods :

Déploiement.yml

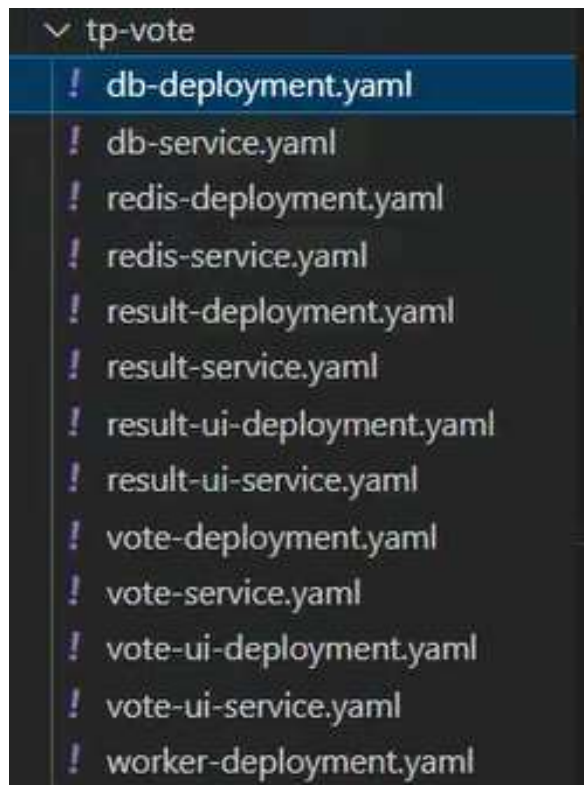
```
PS C:\Atelier\Azure\AKS - Kubernetes\api-python> kubectl create -f .\k8s-resources\deployment.yml
deployment.apps/api-python-creation created
```

Service.yml

```
PS C:\Atelier\Azure\AKS - Kubernetes\api-python> kubectl create -f .\k8s-resources\service.yml
```

```
PS C:\Atelier\Azure\AKS - Kubernetes\api-python> kubectl create -f .\aks-resources\service.yaml
service/service-api-python created
```

Correction déploiement application vote :



Exemple de configuration déploiement :

```
name: result
spec:
  replicas: 1
  selector:
    matchLabels:
      app: result
  template:
    metadata:
      labels:
        app: result
    spec:
      containers:
        - image: m2informationhab.azurecr.io/result-image
          name: result
          imagePullPolicy: Always
          ports:
            - containerPort: 80
```

Exemple de configuration service :

```
aks > tp-vote > ! vote-service.yaml > {} spec
1  apiVersion: v1
2  kind: Service
3  metadata:
4    labels:
5      app: vote
6    name: vote
7  spec:
8    type: LoadBalancer
```

```

8   type: LoadBalancer
9   ports:
10    - name: "vote-service"
11      port: 80
12      targetPort: 4000
13   selector:
14     app: vote
15

```

```

PS C:\Users\Administrateur\Desktop\Azure\aks> kubectl create -f .\tp-vote\
deployment.apps/db created
service/db created
deployment.apps/redis created
service/redis created
deployment.apps/result created
service/result created
deployment.apps/vote created
service/vote created
deployment.apps/worker created

```

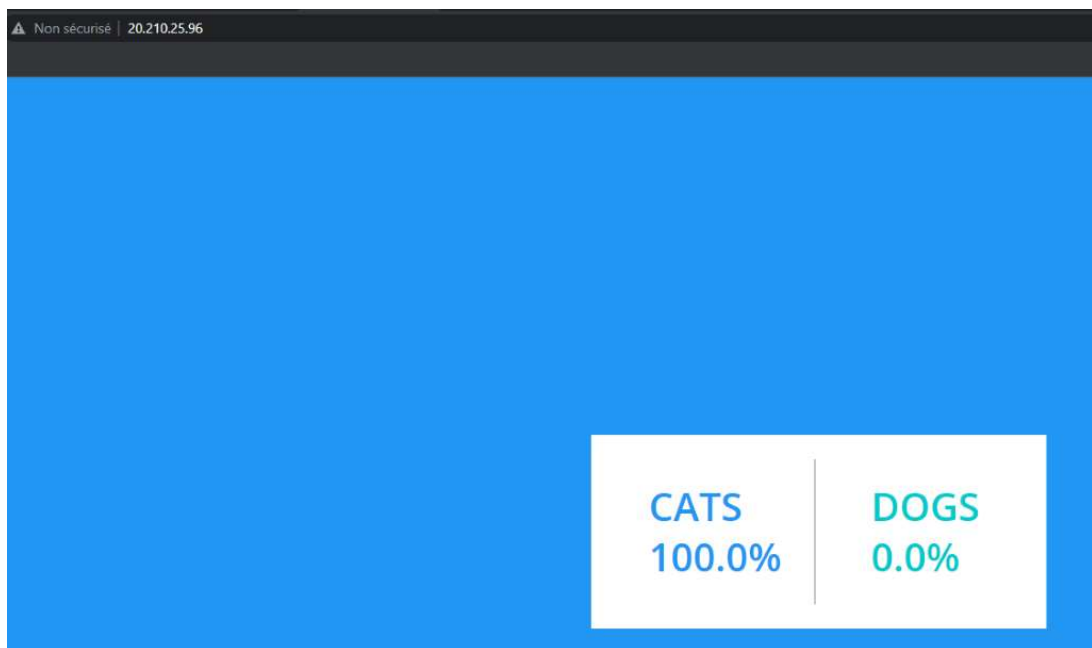
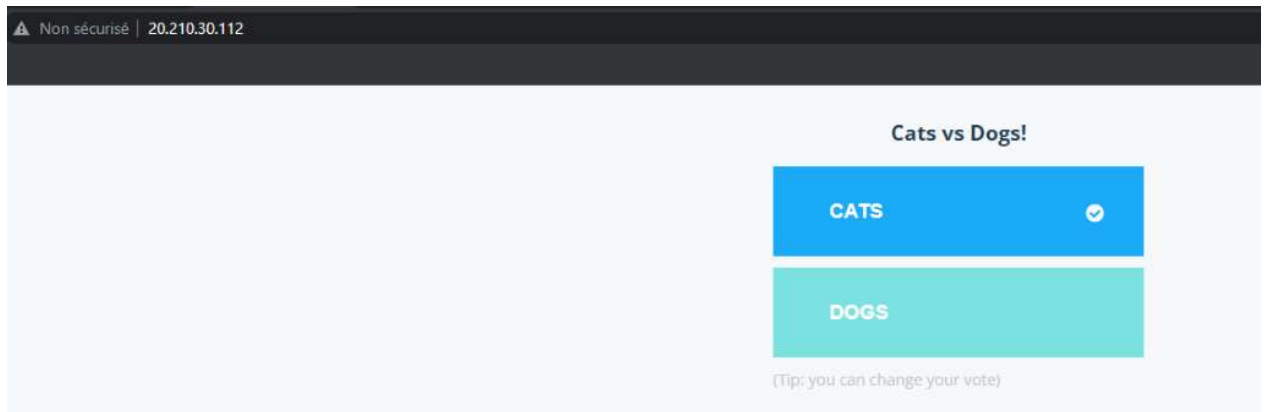
Déploiements				
Pods				
Jeux de réplicas				
Objets StatefulSet				
Jeux de démons				
Travaux				
Filtrer par nom de déploiement				
Filtrer par espace de noms				
Entrer le nom de déploiement complet				
Tous les espaces de noms				
A				
<input type="checkbox"/>	Nom	Espace de noms	Prêt	Mis à jour
<input type="checkbox"/>	coredns	kube-system	✓ 2/2	2
<input type="checkbox"/>	coredns-autoscaler	kube-system	✓ 1/1	1
<input type="checkbox"/>	konnectivity-agent	kube-system	✓ 2/2	2
<input type="checkbox"/>	metrics-server	kube-system	✓ 2/2	2
<input type="checkbox"/>	azure-policy	kube-system	✓ 1/1	1
<input type="checkbox"/>	azure-policy-webhook	kube-system	✓ 1/1	1
<input type="checkbox"/>	gatekeeper-audit	gatekeeper-system	✓ 1/1	1
<input type="checkbox"/>	gatekeeper-controller	gatekeeper-system	✓ 2/2	2
<input type="checkbox"/>	db	default	✓ 1/1	1
<input type="checkbox"/>	redis	default	✓ 1/1	1
<input type="checkbox"/>	appli-result	default	✓ 1/1	1
<input type="checkbox"/>	appli-vote	default	✓ 1/1	1
<input type="checkbox"/>	appli-worker	default	✓ 1/1	1

Ici tout est en loadbalancer, mais il serait plus intéressant de configurer les services qui n'ont pas besoin d'être exposé dans un autre type pour qu'il ne soit pas accessible à l'extérieur.

services							
Entrées							
Filtrer par nom service							
Filtrer par espace de noms							
Entrer le nom de service complet							
Tous les espaces de noms							
<input type="checkbox"/>	Nom	Espace de noms	État	Type	Adresse IP du ...	Adresse IP exter...	Ports
<input type="checkbox"/>	kubernetes	default	✓ Ok	ClusterIP	10.0.0.1		443/TCP
<input type="checkbox"/>	kube-dns	kube-system	✓ Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP
<input type="checkbox"/>	metrics-server	kube-system	✓ Ok	ClusterIP	10.0.39.16		443/TCP

<input type="checkbox"/>	azure-policy-webhook-service	kube-system	✔ Ok	ClusterIP	10.0.208.88	443/TCP
<input type="checkbox"/>	gatekeeper-webhook-service	gatekeeper-system	✔ Ok	ClusterIP	10.0.210.188	443/TCP
<input type="checkbox"/>	db	default	✔ Ok	LoadBalancer	10.0.65.133	20.210.28.40 5432:30073/TCP
<input type="checkbox"/>	appli-result	default	✔ Ok	LoadBalancer	10.0.21.12	20.210.25.96 80:30940/TCP
<input type="checkbox"/>	appli-vote	default	✔ Ok	LoadBalancer	10.0.70.39	20.210.30.112 80:30827/TCP
<input type="checkbox"/>	redis	default	✔ Ok	LoadBalancer	10.0.113.63	20.210.28.223 6379:30668/TCP

Résultat :



Configure Liveness, Readiness and Startup Probes

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>

<https://kubernetes.io/fr/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#:~:text=Les%20readiness%20et%20liveness%20probes,red%C3%A9marr%C3%A9s%20en%20cas%20de%C3%A9faillance.>

Le Kubelet utilise les liveness probes pour détecter quand redémarrer un conteneur. Par exemple, les Liveness probes pourraient attraper un deadlock dans le cas où une application est en cours d'exécution, mais qui est incapable de traiter les requêtes. Le redémarrage d'un conteneur dans un tel état rend l'application plus disponible malgré les bugs.

Le Kubelet utilise readiness probes pour savoir quand un conteneur est prêt à accepter le trafic. Un Pod est considéré comme prêt lorsque tous ses conteneurs sont prêts. Ce signal sert notamment à contrôler les pods qui sont utilisés comme backends pour les Services. Lorsqu'un Pod n'est pas prêt, il est retiré des équilibres de charge des Services.

Le Kubelet utilise startup probes pour savoir quand une application d'un conteneur a démarré. Si une telle probe est configurée, elle désactive les contrôles de liveness et readiness jusqu'à cela réussit, en s'assurant que ces probes n'interfèrent pas avec le démarrage de l'application. Cela peut être utilisé dans le cas des liveness checks sur les conteneurs à démarrage lent, en les évitant de se faire tuer par le Kubelet avant qu'ils ne soient opérationnels.

À ajouter dans le bloc container :

```
spec:
  containers:
  - name: api-python
    image: m2informationihab.azurecr.io/api-python
    resources:
      limits:
        memory: "128Mi"
        cpu: "200m"
    ports:
      - containerPort: 8080
    livenessProbe:
      httpGet:
        path: /check
        ports: 8080
```

```
- containerPort: 8080
livenessProbe:
  httpGet:
    path: /check
    ports: 8080
    initialDelaySeconds: 5
    periodSeconds: 10
```

Le path correspond à un chemin dans l'application : ici le chemin `/check` retournera `Ok`

```
aks > api-python > app.py
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route("/")
5 def firstAction():
6     return "Hello from our app"
7
8 @app.route("/check")
9 def check():
10     return "Ok"
11
12 if __name__ == "__main__":
13     app.run(host="0.0.0.0", port=8080)
```

On peut aussi effectuer une commande `exec` pour que le `livelessProbe` retourne quelque chose :

```
livenessProbe:
  ## Une requête HTTP
  # httpGet:
  #   path: /check2
```

```
# port: 8080
# initialDelaySeconds: 2
# periodSeconds: 3
## Une commande shell ou bash
exec:
  command:
    - "cat app.py"
```

Autre exemple de commande pour que le `livelessProbe` vérifie si un port est bien ouvert :

```
## Vérification d'ouverture d'un TCP Socket
tcpSocket:
  port: 5000
```

```
ports:
  - containerPort: 8080
#livenessProbe:
#readinessProbe:
startupProbe:
```

Network-Policy, communication avec les schedulers

https://fr.wikipedia.org/wiki/Ordonnancement_dans_les_syst%C3%A8mes_d'exploitation

```
app.py  ! deployment.yml  ! network-policy.yml X Dockerfile
aks > api-python > k8s-resources > ! network-policy.yml > {} spec > [ ] ingress > {} 0 >
4   name: first-network-policy
5   spec:
6     podSelector:
7       matchLabels:
8         app: api-python-deployment
9     policyTypes:
10      - Ingress
11      - Egress
12    ingress:
13      - from:
14        - podSelector:
15          matchLabels:
16            app: client-api-python
17        port:
18          - protocol: TCP
19            port: 8080
```

```
egress:
  - to:
    - podSelector:
      matchLabels:
        app: target-api-python
    port:
      - protocol: TCP
```



```
port: 8080
```

Possible aussi de faire un fichier daemon.yml :

[https://fr.wikipedia.org/wiki/Daemon_\(informatique\)#~:text=Un%20daemon%20\(prononc%C3%A9%20%2F%CB%88di%CB%90%2Ccontr%C3%B4le%20direct%20d'un%20utilisateur](https://fr.wikipedia.org/wiki/Daemon_(informatique)#~:text=Un%20daemon%20(prononc%C3%A9%20%2F%CB%88di%CB%90%2Ccontr%C3%B4le%20direct%20d'un%20utilisateur)

Daemon (informatique)

36 langues

Article Discussion

Lire Modifier Modifier le code Voir l'historique

Pour les articles homonymes, voir Démon.

Un **daemon** (prononcé /di.man/ ou /de.man/, du grec δαίμων - divinité), mot anglais qui signifie « *daimôn* », souvent traduit erronément par **démon**^[1], est un type de *programme informatique*, un *processus* ou un ensemble de processus qui s'exécute en arrière-plan plutôt que sous le contrôle direct d'un utilisateur.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: api-python-deployment
  labels:
    name: api-python-deployment
spec:
  selector:
    matchLabels:
      app: api-python-deployment
  template:
    metadata:
      labels:
        app: api-python-deployment
    spec:
      containers:
        - name: api-python
```

```
spec:
  affinity:
    nodeAffinity:
      #requiredDuringSchedulingIgnoredDuringExecution:
      preferredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          matchLabels:
            type: test
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      #preferredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
            - key: type
              operator: In
              values: [test]
```

```
8014548-vmss000001
kubernetes.azure.com/pole-agent
```

```
kubernetes.azure.com/role=agent
failure-domain.beta.kubernetes.io/zone=0
beta.kubernetes.io/instance-type=Standard_D2s
kubernetes.azure.com/agentpool=agentpool
type=test
storagetier=Premium_LRS
kubernetes.io/os=linux
kubernetes.azure.com/mode=system
kubernetes.azure.com/storageprofile=managed
topology.disk.csi.azure.com/zone=
```

Fichier Daemons.yml

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: api-python-deployment
  labels:
    name: api-python-deployment
spec:
  selector:
    matchLabels:
      app: api-python-deployment
  template:
    metadata:
      labels:
        app: api-python-deployment
    spec:
      containers:
        - name: api-python
          image: m2informationihab.azurecr.io/api-python
          resources:
            limits:
              memory: "128Mi"
              cpu: "200m"
          ports:
            - containerPort: 8080
          #livenessProbe:
          #readinessProbe:
          startupProbe:
            ## Une requête HTTP
            # httpGet:
            #   path: /check2
            #   port: 8080
            # initialDelaySeconds: 2
            # periodSeconds: 3
            ## Une commande shell ou bash
            # exec:
            #   command:
            #     - "cat app.py"

            ## Vérification d'ouverture d'un TCP Socket
            tcpSocket:
              port: 8080
```

Fichier Deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-python-deployment
  labels:
    name: api-python-deployment
spec:
  selector:
    matchLabels:
      app: api-python-deployment
  template:
    metadata:
      labels:
        app: api-python-deployment
    spec:
      affinity:
        nodeAffinity:
```

```

    requiredDuringSchedulingIgnoredDuringExecution:
    #preferredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
          - key: type
            operator: In
            values: [test]
containers:
- name: api-python
  image: m2iinformationihab.azurecr.io/api-python
  resources:
    limits:
      memory: "128Mi"
      cpu: "200m"
  ports:
    - containerPort: 8080
  #livenessProbe:
  #readinessProbe:
  startupProbe:
    ## Une requête HTTP
    # httpGet:
    #   path: /check2
    #   port: 8080
    # initialDelaySeconds: 2
    # periodSeconds: 3
    ## Une commande shell ou bash
    # exec:
    #   command:
    #     - "cat app.py"

    ## Vérification d'ouverture d'un TCP Socket
  tcpSocket:
    port: 8080

```

Fichier `network_policy.yml`

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: first-network-policy
spec:
  podSelector:
    matchLabels:
      app: api-python-deployment
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: client-api-python
      port:
        - protocol: TCP
          port: 8080
  egress:
    - to:
        - podSelector:
            matchLabels:
              app: target-api-python
      port:
        - protocol: TCP
          port: 8080

```

Exercice : sur l'application vote :

- Mettre en place un probe sur vote et result (/check)
- 1 worker par node (daemonSet ?)
- Ajouter les networkPolicy nécessaire

Correction :

Question 1 :

Changer le type de worker par DaemonSet

```
worker-deployment.yaml X app.py ...\vote JS server.js
aks > tp-vote > ! worker-deployment.yaml > kind
1  apiVersion: apps/v1
2  # kind: Deployment
3  kind: DaemonSet
4  metadata:
5    labels:
6      app: worker
7      name: worker
8  spec:
9    replicas: 1
10   selector:
11     matchLabels:
12       app: worker
13   template:
14     metadata:
15       labels:
16         app: worker
```

Question 2 :

```
C: > Users > Administrateur > Documents > Documentation_F
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: redis-network-policy
5  spec:
6    podSelector:
7      matchLabels:
8        app: db
9    policyTypes:
10     - Ingress
11    ingress:
12     - from:
13       - podSelector:
14         matchLabels:
15           customer-db: db
16     ports:
17       - protocol: TCP
18         port: 5432
19
```

```
! network-policy-db.yml ! network-policy-redis.yml X
C: > Users > Administrateur > Documents > Documentation_F
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: redis-network-policy
```

```

5 spec:
6   podSelector:
7     matchLabels:
8       app: redis
9   policyTypes:
10    - Ingress
11   ingress:
12     - from:
13       - podSelector:
14         matchLabels:
15           customer: redis
16     ports:
17       - protocol: TCP
18         port: 6379
19

```

```

1 db-deployment.yaml
2 db-service.yaml
3 network-policy-db.yaml
4 network-policy-redis.yaml
5 redis-deployment.yaml
6 redis-service.yaml
7 result-deployment.yaml
8 result-service.yaml
9 vote-deployment.yaml
10 vote-service.yaml
11 worker-deployment.yaml

```

Azure AppService :

Créer une application web ...

De base Déploiement Réseau Surveillance Balises Vérifier + créer

App Service Web Apps vous permet de créer, de déployer et de mettre à l'échelle des applications d'API, web et mobiles de classe Entreprise exécutées sur n'importe quelle plateforme rapidement. Respectez les exigences strictes en termes de performances, de scalabilité, de sécurité et de conformité lors de l'utilisation d'une plateforme complètement managée pour effectuer la maintenance de l'infrastructure. [En savoir plus](#)

Nom * .azurewebsites.net

Publier * ☒ Code ☐ Conteneur Docker ☐ Application web statique

Pile d'exécution *

Système d'exploitation * ☒ Linux ☐ Windows

Région * Vous ne trouvez pas votre plan App Service ? Essayez une autre région ou sélectionnez votre environnement App Service Environment.

Plans de tarification

Le niveau tarifaire du plan App Service détermine l'emplacement, les fonctionnalités, le coût et les ressources de calcul associés à votre application. [En savoir plus](#)

Plan Linux (East US) * Créer

Plan de tarification Découvrir les plans de tarification

Redondance de zone

Vous pouvez déployer un plan App Service en tant que service redondant interzone dans des régions le prenant en charge. Vous devez décider de cela au moment du déploiement car vous ne pouvez plus rendre une zone plan App Service redondante après son déploiement. [En savoir plus](#)

Redondance de zone

- ☐ **Activé** : Votre plan App Service et les applications qu'il contient seront redondants interzone. Il y aura au minimum trois instances de plan App Service.
- ☒ **Désactivé** : Votre plan App Service et les applications qu'il contient ne seront pas redondants interzone. Il y aura au minimum une instance de plan App Service.

Après le déploiement de l'application :

Application web

Nom	demoihab
Modèle de publication	Code
Pile d'exécution	Python - 3.9

Envoie de l'application sous format zip :

```
PS C:\Users\Administrateur\Desktop\Azure\aks> az webapp --name demoihab
```

La commande `SCM_DO_BUILD_DURING_DEPLOYMENT` permet de build au moment du déploiement

Ajouter/modifier le paramètre d'application

Nom	<input type="text" value="SCM_DO_BUILD_DURING_DEPLOYMENT"/>
Valeur	<input type="text" value="true"/>
<input checked="" type="checkbox"/> Paramètre de l'emplacement de déploiement	

Paramètres de l'application

Les paramètres d'application sont chiffrés au repos et transmis sur un canal chiffré. Vous pouvez choisir de les utiliser les contrôles ci-dessous. Les paramètres d'application sont exposés sous forme de variables d'environnement au moment de l'exécution. [En savoir plus](#)

[+ Nouveau paramètre d'application](#) [👁 Afficher les valeurs](#) [✎ Modification avancée](#)

Nom	Valeur	Source
SCM_DO_BUILD_DURING_DEPLOYMENT	👁 Valeur masquée. Cliquez pour l'afficher	App Service

```
PS C:\Users\Administrateur\Desktop\Azure\aks> az webapp deploy --name demoihab --resource-group m2i
-formation --src-path api-python.zip
This command is in preview and under development. Reference and support levels: https://aka.ms/CLI_refstatus
Deployment type: zip. To override deployment type, please specify the --type parameter. Possible values: war, jar, ear, zip, startup, script, static
```


Notifications



Plus d'événements dans le journal d'activité →

Tout ignorer ▾

*** Mise à jour des informations d'identification de l'utilisateur de publication En cours d'exécution ✕

Mise à jour des informations d'identification de l'utilisateur de publication

il y a quelques secondes

✓ Configuration du déploiement ✕

Déploiement configuré

il y a 3 minutes

✓ Déploiement réussi ✕

Le déploiement « Microsoft.Web-WebApp-Portal-508d8483-b18d » sur le groupe de ressources « m2i-formation » a réussi.

[Accéder à la ressource](#)

[Épingler au tableau de bord](#)

il y a 6 minutes


✓ Opération de suppression ✕

« demoihab » correctement supprimé


il y a 7 minutes


```
201
202 cd aks/api-python
203 git remote set-url azure https://api-python-ihab.scm.azurewebsites.net:443/api-python-ihab
git
204 git push azure master
205 git push azure master
206 git push azure master
207 history
```


Création webapp :


**Microsoft.Web-WebApp-Portal-24d21e7c-93cc** | Vue d'ensemble ✕ ...
Déploiement

« [Supprimer](#) [Annuler](#) [Redéployer](#) [Télécharger](#) [Actualiser](#)


 Vue d'ensemble

 Entrées

 Sorties

 Modèle

*** Le déploiement est en cours



Nom du déploiement : Microsoft.Web-WebApp-Portal-24d21e7c-93cc

Abonnement : Abonnement Azure 1

Groupe de ressources : m2i-formation

Heure de début : 15/03/2023 10:59:07

ID de corrélation : 3f2e55bd-62ac-4ea7-887e-da12

^ Détails du déploiement

Paramètres Journaux Informations d'identification FT

Déployez et générez du code à partir de votre fournisseur de code

Source

Git local

[Se déconnecter](#)

Git local

URI Git Clone

https://react-ihab.scm.azurewebsites.net:443/react-ihab






Build

Fournisseur de build	Service de build App Service
Pile d'exécution	Node
Version	Node 16 LTS

 Supprimer  Annuler  Redéployer  Télécharger  Actualiser

✓ Votre déploiement a été effectué

 Nom du déploiement : Microsoft.Web-WebApp-Port... Heure de début : 15/03/2023 11:03:46

 Enregistrer  Abandonner  Parcourir  Gérer le profil de publication  Synchroniser

[Paramètres](#) [Journaux](#) [Informations d'identification FTPS/Git local](#)

Déployez et générez du code à partir de votre fournisseur de build et source par défaut. [En savoir plus](#)

Source	Git local Se déconnecter
Git local	
URI Git Clone	https://reactihab.scm.azurewebsites.net:443/reactihab.git
Build	
Fournisseur de build	Service de build App Service
Pile d'exécution	Node
Version	Node 16 LTS

```
Administrateur@Salle_3_Z MINGW64 ~/Desktop/Azure/webapp/app-react-todo (master)
$ git remote add azure https://reactihab.scm.azurewebsites.net:443/reactihab.git

Administrateur@Salle_3_Z MINGW64 ~/Desktop/Azure/webapp/app-react-todo (master)
$ git push azure master
```

```
git remote add azure https://demobenoit.scm.azurewebsites.net:443/demoBenoit.git
git push azure master
```

