

Faire une démonstration technique de l'usage du service de conteneurs managés dans Azure

INTRODUCTION	1
CONFIGURATION	2
DEFINIR LES VARIABLES D'ENVIRONNEMENT	3
CREATION DU GROUPE DE RESSOURCE	4
PREPARATION DU REFERENTIEL GITHUB.....	4
CREATION ACR	5
GENERER LE CONTENEUR AVEC ACR	5
CREATION ENVIRONNEMENT CONTAINER APPS.....	6
DEPLOYER NOTRE IMAGE	6
CREATION APPLICATION WEB	7
DEPLOIEMENT CONTINU	9

INTRODUCTION

Il s'agit de faire une démonstration simple du déploiement d'une application web Node-Azure- Docker via le service de conteneurs managés dans Azure.

Ce guide a pour but de servir de démonstration pour un collègue et indiquera la démarche à suivre pour déployer l'application Web.

Les prérequis nécessaires pour la mise en place du déploiement de l'application web sont les suivants :

- Disposer d'un abonnement Azure
- Installer Azure CLI
- Installer l'extension "Azure Container Apps" pour l'interface CLI

CONFIGURATION

Depuis le terminal azure, dans Bash, se connecter à son compte via la commande "az login".

Vous devez vous connecter sur le lien http Microsoft avec le mot de passe donner pour être sur Azure CLI.

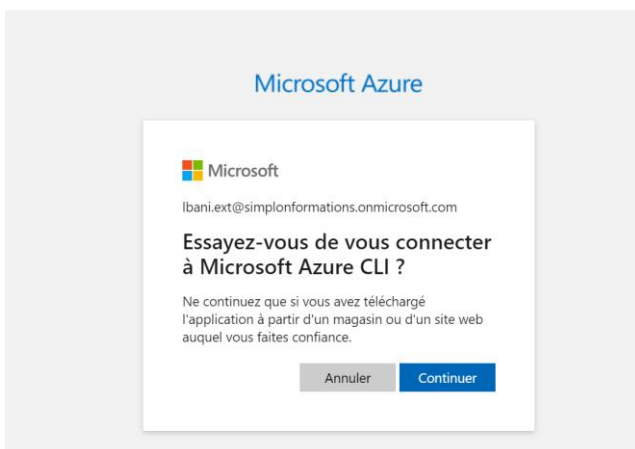
```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

lylla [ ~ ]$ az login
Cloud Shell is automatically authenticated under the initial account signed-in with. Run 'az login' only if you need to use a different account
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code C8EUQVH2K to authenticate.
```

Suivez les instructions indiquées comme ci-dessous :



Une fois l'authentification validé



Microsoft Azure Cross-platform Command Line Interface

Vous vous êtes connecté à l'application Microsoft Azure Cross-platform Command Line Interface sur votre appareil. Vous pouvez maintenant fermer cette fenêtre.

Bienvenue sur azure CLI. Voici ce que vous aurez sur votre écran côté terminal

```
[
{
  "cloudName": "AzureCloud",
  "homeTenantId": "7349d3b2-951f-41be-877e-d8ccd9f3e73c",
  "id": "393e3de3-0900-4b72-8f1b-fb3b1d6b97f1",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Simplon ARA Lyon Admin Cloud P22",
  "state": "Enabled",
  "tenantId": "7349d3b2-951f-41be-877e-d8ccd9f3e73c",
  "user": {
    "name": "lbani.ext@simplonformations.onmicrosoft.com",
    "type": "user"
  }
}
]
lylla [ ~ ]$
```

Installez l'extension de "Azure Container app" pour CLI avec la commande suivante:

Az extension -name container app -upgrade

1. Puis, saisissez les 2 commandes ci-dessous "az provider register" afin de déterminer les 2 fournisseurs de votre espace de travail.

```
}
}
]
lylla [ ~ ]$ az provider register --namespace Microsoft.App
lylla [ ~ ]$ az provider register --namespace Microsoft.OperationalInsights
lylla [ ~ ]$
```

DEFINIR LES VARIABLES D'ENVIRONNEMENT

Comme dans l'exemple ci-dessous :

RESOURCE_GROUP="lyllabbrief10"

LOCATION="francecentral"

ENVIRONMENT="lyllaenvironment"

API_NAME="node-azure-docker"

FRONTEND_NAME="node-azure-dockerui"

GITHUB_USERNAME="lylla834"

```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

lylla [ ~ ]$ RESOURCE_GROUP="lyllab10"
LOCATION="francecentral"
ENVIRONMENT="lyllaenvironment"
API_NAME="lyllapi"
FRONTEND_NAME="frontlylla"
GITHUB_USERNAME="lylla834"
lylla [ ~ ]$ ACR_NAME="registrylylla"$GITHUB_USERNAME
lylla [ ~ ]$ az group create \
```

Définir un nom de registre de conteneur avec la variable suivante "ACR_NAME" séparément des variables d'environnement.

CREATION DU GROUPE DE RESSOURCE

Grâce aux variables créer plus tôt, vous allez pouvoir créer votre groupe de ressource via la commande "az group create" pour organiser vos services conteneurs.

```
lylla [ ~ ]$ az group create \
--name $RESOURCE_GROUP \
--location "$LOCATION"

{"id": "/subscriptions/393e3de3-0900-4b72-8f1b-fb3b1d6b97f1/resourceGroups/lyllab10",
"location": "francecentral",
"managedBy": null,
"name": "lyllab10",
"properties": {
  "provisioningState": "Succeeded"
},
"tags": null,
"type": "Microsoft.Resources/resourceGroups"
}
```

PREPARATION DU REFERENTIEL GITHUB

Depuis votre compte GitHub, utiliser la fonction "Fork" afin de dupliquer l'application choisie. Cliquez sur "create fork" et donnez-lui un nom personnalisé dans votre repository dans votre compte GitHub. Ensuite, cloner l'application dans votre terminal avec la commande "git clone" suivit du lien https de l'emplacement de l'application sur GitHub.

Exemple de lien github : `git clone https://github.com/lylla834/node-azure-docker.git`

```
GITHUB_USERNAME="lylla834"
lylla [ ~ ]$ ACR_NAME="container_registerlylla"lylla834
lylla [ ~ ]$ git clone https://github.com/lylla834/containerapps-albumui.git codetocloud-ui
Cloning into 'codetocloud-ui'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 64 (delta 8), reused 7 (delta 7), pack-reused 51
Receiving objects: 100% (64/64), 35.13 KiB | 2.51 MiB/s, done.
Resolving deltas: 100% (18/18), done.
lylla [ ~ ]$
```

CREATION ACR

Vous devez créer un dépôt “Azure Container Registry” avec la commande “az acr create” pour pouvoir héberger l’image conteneur de l’application. La variable “--admin-enabled true” permettra d’activer Docker sur Azure Container Registry. Voir exemple ci –dessous :

```
az acr create \
  --resource-group $RESOURCE_GROUP \
  --name $ACR_NAME \
  --sku Basic \
  --admin-enabled true
```

```
lylla [ ~ ]$ RESOURCE_GROUP="lyllab10"
LOCATION="francecentral"
ENVIRONMENT="lyllaenvironment"
lylla [ ~ ]$ az acr create \
  --resource-group $RESOURCE_GROUP \
  --name $ACR_NAME \
  --sku Basic \
  --admin-enabled true
{
  "adminUserEnabled": true,
  "anonymousPullEnabled": false,
  "creationDate": "2023-04-26T20:07:25.494684+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
```

GENERER LE CONTENEUR AVEC ACR

Dans votre terminal azure, en utilisant les commandes “ls” pour voir l’ensemble des dossiers. Ici, on peut voir “clouddrive” et “node-azure-docke”r.

Puis pour aller dans dossier “cd node-azure-docke”, enfin “ls”. Allez au niveau du dossier “ Dockerfile”, puis construire son image avec la commande “az acr build”.

```

lylla [ ~ ]$ ls
clouddrive node-azure-docker
lylla [ ~ ]$ cd node-azure-docker/ls
bash: cd: node-azure-docker/ls: No such file or directory
lylla [ ~ ]$ cd node-azure-docker/
lylla [ ~/node-azure-docker ]$ ls
app.js Dockerfile node_modules package.json package-lock.json README.md routes.js server.js __tests__ t
lylla [ ~/node-azure-docker ]$

```

```

Queue a remote GitHub context as a Windows build, tag it, and push it to the registry.

https://docs.microsoft.com/en-US/cli/azure/acr#az_acr_build
Read more about the command in reference docs
lylla [ ~/node-azure-docker ]$ RESOURCE_GROUP="lyllabbrier10"
LOCATION="francecentral"
ENVIRONMENT="lyllaenvironment"
API_NAME="node-azure-docker"
FRONTEND_NAME="node-azure-dockerui"
GITHUB_USERNAME="lylla834"
lylla [ ~/node-azure-docker ]$ ACR_NAME="registry"$GITHUB_USERNAME
lylla [ ~/node-azure-docker ]$ az acr build --registry $ACR_NAME --image $API_NAME .
Packing source code into tar to upload...
Excluding '.gitignore' based on default ignore rules
Excluding '.git' based on default ignore rules
Uploading archived source code from '/tmp/build_archive_ea60f46927ab4d39a9e6555ea4af4dff.tar.gz'...
Sending context (582.385 KiB) to registry: registrylylla834...
Queued a build with ID: ddi
Waiting for an agent...
2023/04/26 21:02:26 Downloading source code...
2023/04/26 21:02:28 Finished downloading source code

```

CREATION ENVIRONNEMENT CONTAINER APPS

Créer un environnement dans “Azure container apps”. Ce dernier viendra définir une limite sécurisée autour des groupes d’applications de conteneur.

```

lylla [ ~/node-azure-docker ]$ az containerapp env create \
--name $ENVIRONMENT \
--resource-group $RESOURCE_GROUP \
--location $LOCATION"
No Log Analytics workspace provided.
Generating a Log Analytics workspace with name "workspace-lyllab10XFlu"
Container Apps environment created. To deploy a container app, use: az containerapp create --help

{
  "id": "/subscriptions/393e3de3-0900-4b72-8f1b-fb3bd6b97f1/resourceGroups/lyllab10/providers/Microsoft.App/managedEnvironments/lyllaenviron",
  "location": "francecentral",
  "name": "lyllaenvironment",
  "properties": {
    "appLogsConfiguration": {
      "destination": "log-analytics",
      "logAnalyticsConfiguration": {
        "customerId": "5e392a77-2c10-496f-95b7-dba170a50d9a",
        "sharedKey": null
      }
    }
  }
},

```

DEPLOYER NOTRE IMAGE

Vous allez créer l’application conteneur via la commande “azcontainerapp create” dans l’environnement définir plus tôt.

```

Run ID: ddi was successful after 35s
lylla [ ~/node-azure-docker ]$ az containerapp create \
--name $API_NAME \
--resource-group $RESOURCE_GROUP \
--environment $ENVIRONMENT \
--image $ACR_NAME.azurecr.io/$API_NAME \
--target-port 3500 \
--ingress 'external' \
--registry-server $ACR_NAME.azurecr.io \
--query properties.configuration.ingress.fqdn
No credential was provided to access Azure Container Registry. Trying to look up credentials...
Adding registry password as a secret with name "registrylylla834azurecrio-registrylylla834"

Container app created. Access your app at https://node-azure-docker.greenwave-83cf3b61.francecentral.azurecontainerapps.io/

"node-azure-docker.greenwave-83cf3b61.francecentral.azurecontainerapps.io"
lylla [ ~/node-azure-docker ]$

```

Faites attention à la ligne “target-port” afin de renseigner le bon numéro de port. On peut trouver le port exposé dans le Dockerfile via la commande “cat Dockerfile” ou modifier via le portail Azure.

Exécutez les commandes suivantes :

```

lylla [ ~ ]$ npm install
npm ERR! code ENOENT
npm ERR! syscall open
npm ERR! path /home/lylla/package.json
npm ERR! errno -2
npm ERR! enoent ENOENT: no such file or directory, open '/home/lylla/package.json'
npm ERR! enoent This is related to npm not being able to find a file.
npm ERR! enoent

npm ERR! A complete log of this run can be found in:
npm ERR! /home/lylla/.npm/_logs/2023-04-26T21_11_32_735Z-debug-0.log
lylla [ ~ ]$

```

```

npm audit fix

Run `npm audit` for details.
lylla [ ~/node-azure-docker ]$ npm start

> azure-node-api@1.0.0 start
> node server.js

Server running on localhost:5000

```

Une fois le conteneur créé, on peut vérifier le déploiement de notre application sur un navigateur web.

node-azure-docker.greenwave-83cf3b61.francecentral.azurecontainerapps.io/todos

[{"id":1,"task":"Fix Sink"}, {"id":2,"task":"Buy Groceries"}, {"id":3,"task":"Wash the dishes"}, {"id":4,"task":"Make Dinner"}]

CREATION APPLICATION WEB

Depuis le portail Azure, dans la barre de recherche, chercher “Web App”, puis cliquez sur “Créer”. Remplissez les champs obligatoires dans les onglets “basics” et “Docker” puis cliquer sur “Vérifier + créer”.

Accueil >

Créer une ressource ...

Démarrer

Rechercher dans les services et la place de marché

Création récente

Services Azure populaires

Afficher plus dans tous les services

Catégories

IA + Machine Learning

Analyse

Blockchain

Compute

Conteneurs

Bases de données

Outils de développement

DevOps

Identité

Machine virtuelle

Créer | Documents | MS Learn

Web App

Créer | Documents | MS Learn

SQL Database

Créer | Documents | MS Learn

Application de fonction

Créer | Documents

Key Vault

Créer | Documents | MS Learn

Accueil > Créer une ressource >

Créer une application web ...

Plans de tarification

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

Plan Linux (France Central) * ⓘ

(Nouveau) planappserviceylla

Créer

Plan de tarification

Gratuit F1 (Infrastructure partagée)

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.

Vérifier + créer

< Précédent

Suivant : Docker >

Microsoft Azure

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

Accueil > Créer une ressource >

Créer une application web ...

Extrayez des images conteneurs du registre de conteneurs Azure, de Docker Hub ou d'un dépôt Docker privé. App Service déploie en production l'application conteneurisée avec vos dépendances préférées en quelques secondes.

Options

Conteneur unique

Source d'image

Registre de conteneurs Azure

Options de registre de conteneurs Azure

Registre *

yllacontainerhocine699

Image *

node-azure-docker

Balise *

latest

Commande de démarrage ⓘ

Vérifier + créer

< Précédent

Suivant : Réseau >

Accueil > Créer une ressource >

Créer une application web ...

Plan App Service (nouveau)

Nom

planappserviceylla

Système d'exploitation

Linux

Région

France Central

Référence

Gratuit

ACU

Infrastructure partagée

Mémoire

1 Go de mémoire

Surveillance

Application Insights

Non activé

Déploiement

Déploiement continu

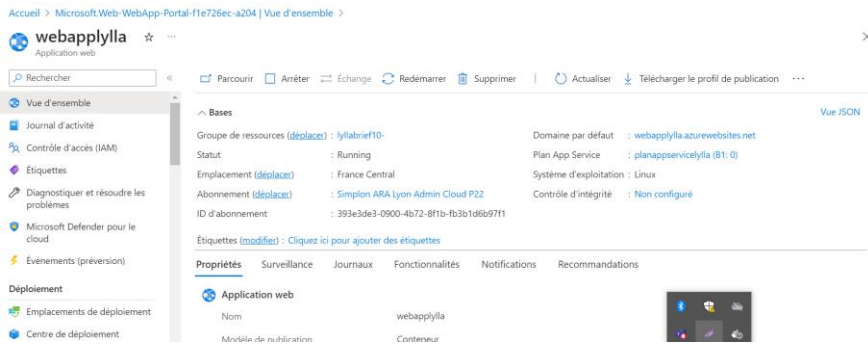
Non activé/configuré après la création de l'application

Créer

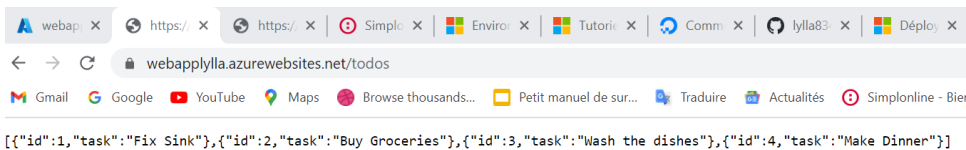
< Précédent

Suivant >

Télécharger un modèle pour automation

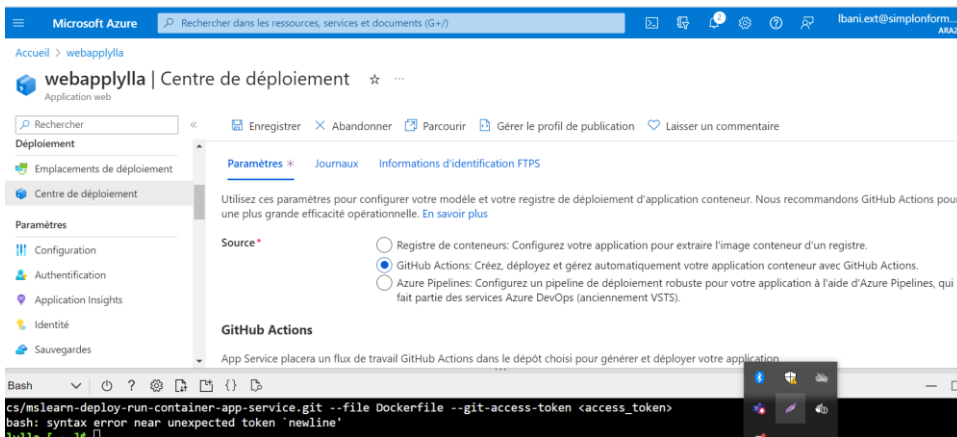


Si votre déploiement a bien fonctionné, vous aurez votre image sur le navigateur web.

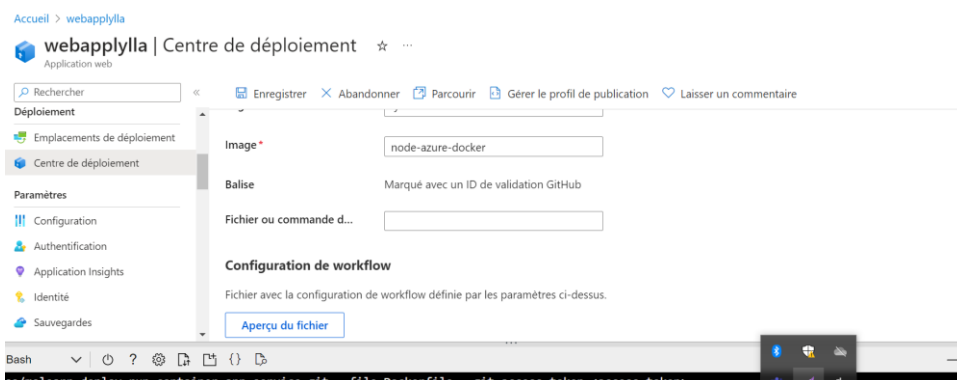


DEPLOIEMENT CONTINU

Depuis le portail azure, en passant par “centre de déploiement” sur votre gauche, sélectionnez “GitHub action” à droite, remplissez.



Enregistrez.



Vérifiez votre image sur le navigateur web, pour vérifier le déploiement continu.

