#10/02/2025

Objectif : Connecter une application Flask au cloud Azure et utiliser les fonctionnalité de celui-ci pour pouvoir faire des choses intéressantes avec notre application.

En de simples termes, tu crées ton flask app, tu la compacte avec toutes les librairies et environnement dans un fichier docker (assurer son intégrité) puis tu la deploies dans l’espace Azure pourqu’il puisse être accédé à distance (en gros que ca devienne un site web quoi).

With Docker, your **CI/CD pipeline can build a container image, push it to Azure Container Registry (ACR), and deploy it to AKS**.

1. Configuration du github :

• Comment structurer votre référentiel Git pour assurer un workflow clair et maintenable ?

• Quelle stratégie de branchement (Git Flow, Trunk-based development) soutiendra le mieux votre pipeline CI/CD ?

>> Pour le développement de notre app, nous avons opté pour un Git Flow branchement, dans lequel les nouvelles fonctionnalités seront créées dans la branche feature, ensuite elle seront stabilisées dans la branche deployment pour être enfin être intégrer à la branche Main avec que les modifications puissent être ajoutées sur la version deployée.

• Quelles règles .gitignore devez-vous appliquer pour éviter de commettre des fichiers sensibles ?

Dans le fichier .gitignore, nous avons choisi d’ajouter le dossier contenant l’environnement virtuel, l’image docker, et le fichier des configuration parce que nous voulons maintenir l’accès à l’environnement de travail ou aux configurations de l’app privé.

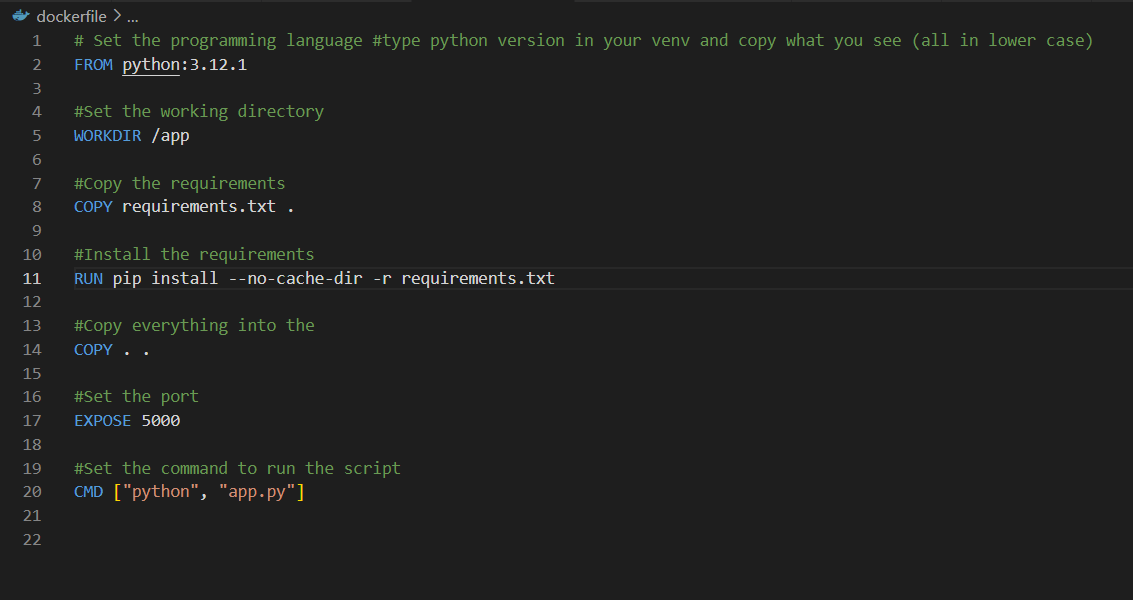
How to get API key from data gov

1. Sign up for an API key and you’ll receive it by mail : **lAV9Z7M3vk0moTQzfYpyqIPl6orVWXYVaQvgz7cD**

What do we

#23/02/2025

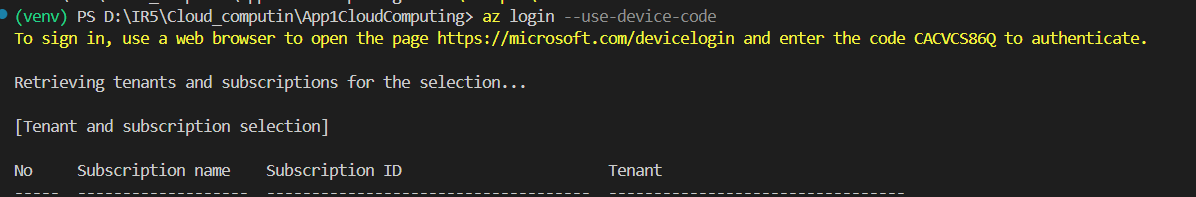
# 0 Create a Docker file



This file allows us to configure specify which app and dependencies do we want to pack in a docker file so it can be run on other device later on.

#1 Connect to the account

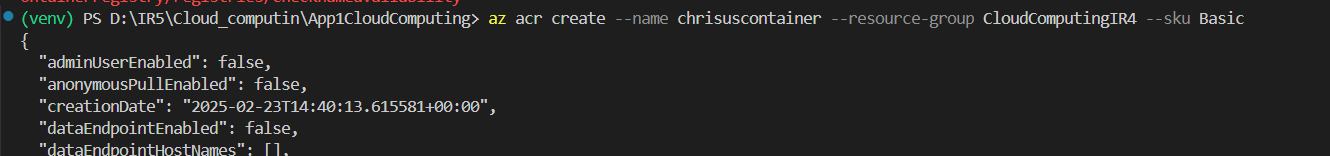
az login --use-device-code



#2 create an Azure container registry

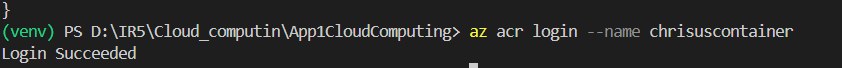
It’s a service issued by azure that stores and manages docker files

az acr create --name chrisuscontainer --resource-group CloudComputingIR4 --sku Basic



#3 Login into the ACR

az acr login --name chrisuscontainer



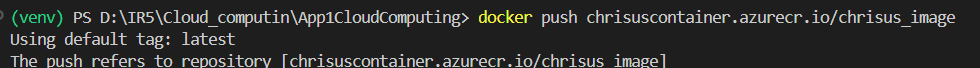
#4 Tag my docker image

docker tag chrisus\_image chrisuscontainer.azurecr.io/chrisus\_image



#5 Push the image to the ACR

docker push chrisuscontainer.azurecr.io/chrisus\_image



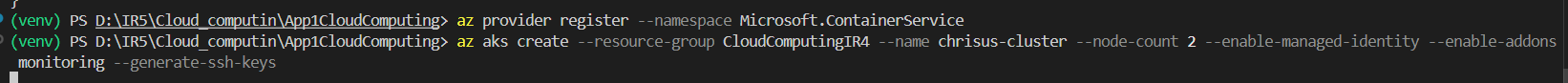
#Step 2

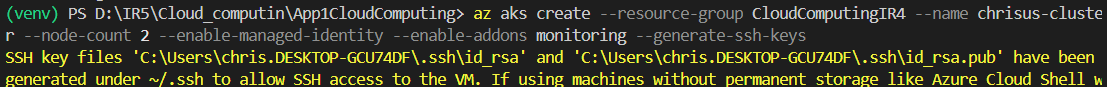
#Create a Azure Kubernetes service

We first create the container service then, we create the **cluter** in AKS.

az provider register --namespace Microsoft.ContainerService

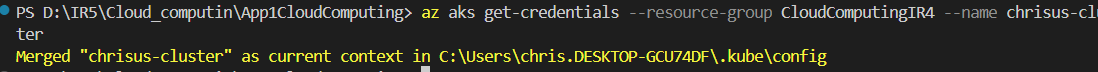
az aks create --resource-group CloudComputingIR4 --name chrisus-cluster --node-count 2 --enable-managed-identity --enable-addons monitoring --generate-ssh-keys





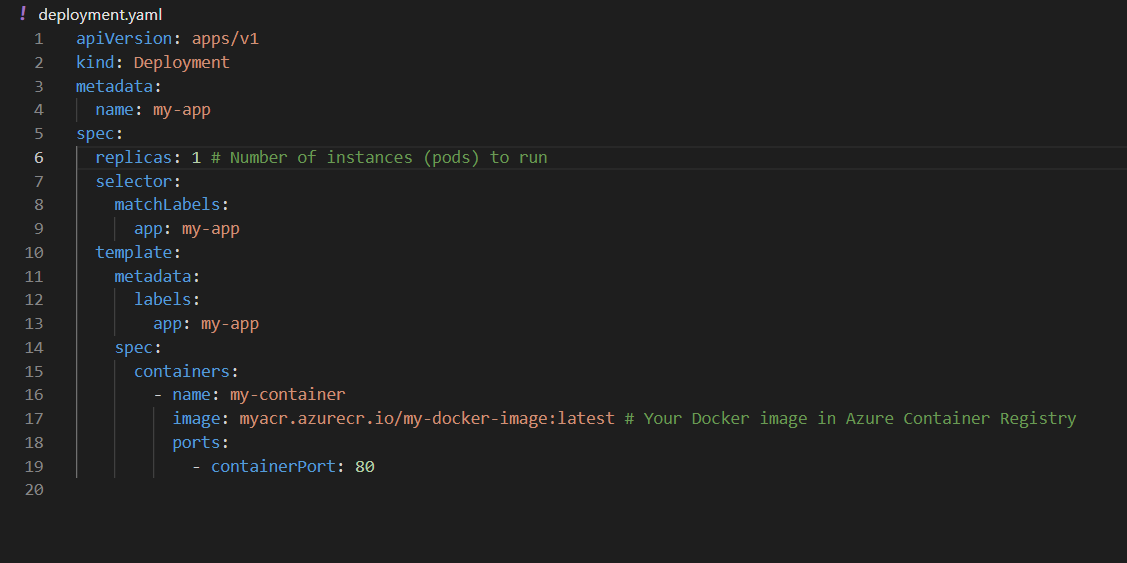
#2.1 Get Kubernetes Credentials

az aks get-credentials --resource-group <your-resource-group> --name <your-cluster-name>



Allow to retrieve the necessary authentication in order to run the “kubectl” command to work with the cluster from the command line.

#2.2 Write a deployment file



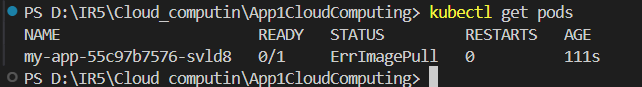
This’s the configuration that allows us to specify the AKS how to deploy with our file.

#2.4 Apply the deployment file to AKS

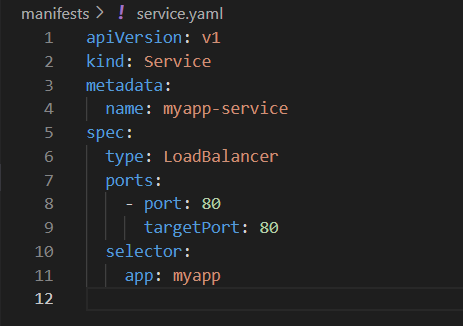
kubectl apply -f manifests/deployment.yaml



#Check instance of our app



2.5 Create a service file to handle the instance of our app

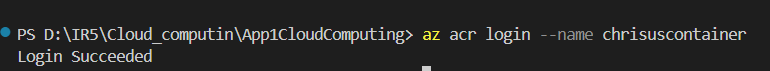


#Apply the service file

kubectl apply -f service.yaml



#



#the env file is very important it contains the key and shouldn’t be include anywhere

#to handle the environmental variables in docker, you need to specify it while running it and in the cloud you need to add it in the deployment file. Don’t build you image with your key inside, it must always remain outside.

To run a sh script you use

“sh your\_script.sh”

Connect to acr> get credential> show the credential >

