

Formation Azure

Ihab ABADI / UTOPIOS

SOMMAIRE – Partie 1

1. Approvisionnement de machines virtuelles dans Azure
2. Azure Resource Manager
3. Azure Container Registry
4. Azure Container Instances

Approvisionnement de machines virtuelles dans Azure

- Exploration des machines virtuelles Azure
- La disponibilité des machines virtuelles
- Scalabilité des vms
- Exercice

Création d'une machine virtuelle

- La création d'une machine virtuelle nécessite l'existence d'un groupe de ressources.
- La création d'une machine virtuelle peut se faire à l'aide de azure cli
- Azure nous permet de créer des machine virtuelle à partir d'une multitude d'image.
- Chaque machine virtuelle peut être défini avec une taille en fonction de la région.
- Azure nous offre la possibilité de faire un redimensionnement des machines virtuelles
- Par défaut, le seul port ouvert est le port ssh.
- Nous avons la possibilité de modifier la configuration réseau de la machine.

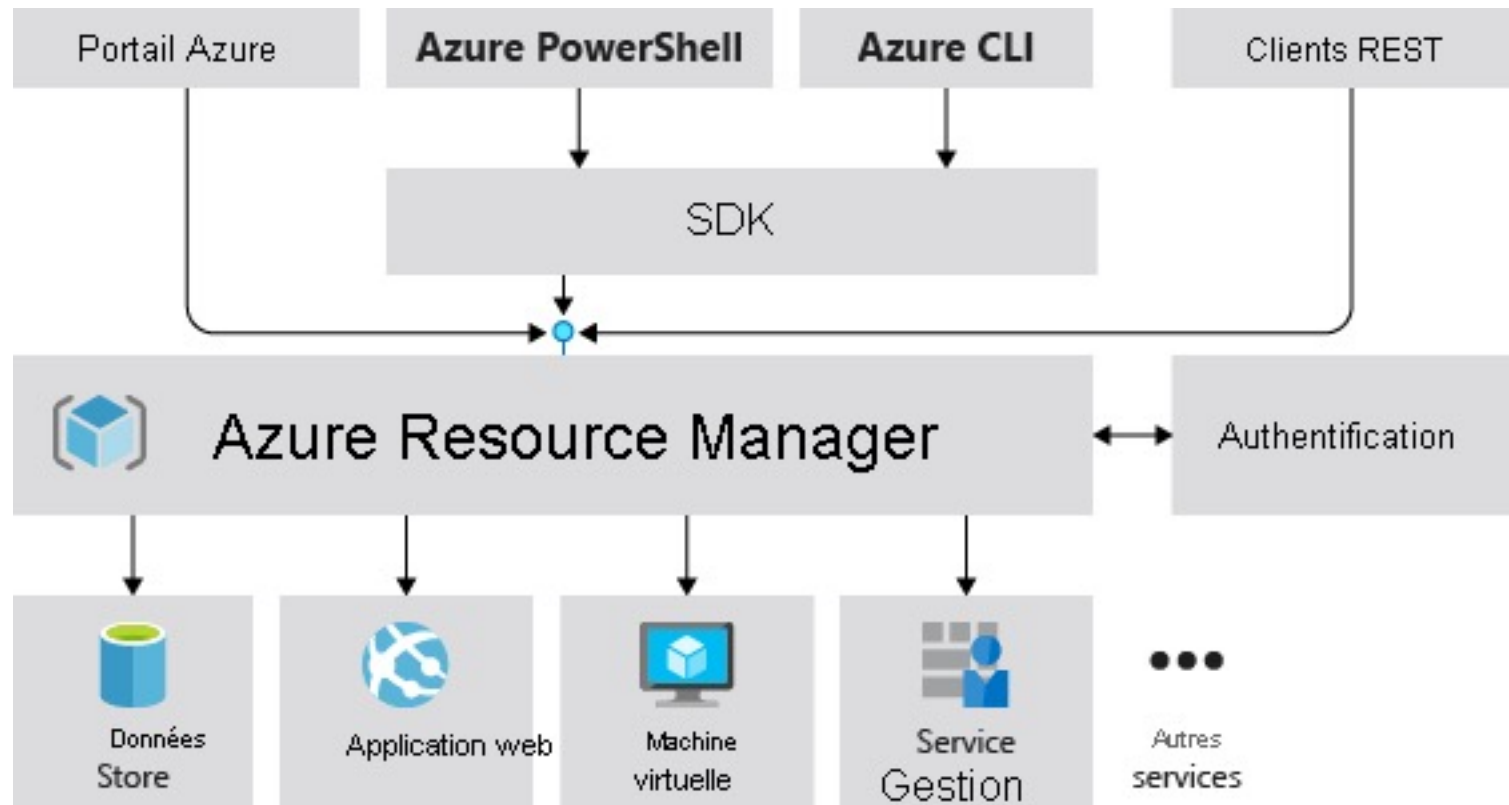
Exercice machine virtuelle

- A l'aide du cli azure,
- Créer une machine virtuelle à partir d'une image ubuntu.
- Se connecter à la machine et installer un service nginx.
- Ouvrir le port par défaut nginx sur la machine.
- Tester la connexion du serveur web nginx.

Azure Resource Manager

- Introduction
- Utilisation d'Azure Resource manager
- Déploiement de solutions avec Azure resource manager
- TP

Introduction ARM



Introduction ARM

- ARM offre la possibilité de créer des ressources azure tel que les Vms, les infrastructure réseau, système de stockage ou n'importe quel type de service d'une façon déclarative.
- ARM permet de déployer notre infrastructure d'une façon reproductibles tout au long du cycle de vie de nos applications.
- ARM offre une abstraction des commandes et des opérations à effectuer pour le déploiement.

Utilisation des ARM

- Un modèle Azure Resource Manager peut être en format json.
- Un modèle Azure Resource Manager est composé de :
 - Une partie paramètres
 - Une partie Variables
 - Une partie Ressources
 - Une partie Sorties.

Un modèle ARM peut également comporter des fonctions personnalisées

Utilisation des ARMs

- Démo

Mode de déploiement des ARMS

- Mode incrémentiel (Mode par défaut)
 - Ce mode permet ne pas supprimer les ressources dans le cas d'une mise à jour des fichiers de déploiements
- Mode complet
 - Ce mode va avoir comme conséquence la création de la totalité des ressources et la suppression des anciennes.
- Démo

Exercice ARM

- Dans un déploiement ARM Créer les ressources suivantes:
- Une ressource de type storage
- Une ressource de type networkSecurityGroups
- Une ressource de type PublicIPAddresses
- Une ressource de type virtualMachine
- Créer un fichier de configuration séparé.
- Déployer votre ARM

Azure Container Registry

- Acr est un service de registre d'image de conteneur privé.
- Acr propose trois niveaux de service base, Standard et premium.
- Acr fournit un mécanisme d'authentification basé sur Azure active Directory.
- Acr fournit un service gestion automatisé des images pour faciliter le build des images d'une pipeline d'integration continue.
- Acr prend en charge des images linux et windows
- Démo

Azure container service - Exercice

- A partir d'une application web (api rest ou autre).
- Créer une image pour votre application.
- Déployer votre image sur notre acr.

Azure Container Service

- Acs est un service qui permet de démarrer et déployer des conteneurs en tant que service à partir de nos images.
- Acs est un service pour des conteneurs isolés.
- Acs permet d'exposer les applications à l'aide d'une adresse ip public.
- Acs permet le montage de volume à partir de azure file par exemple.
- Acs permet de gérer les ressources physique.
- Acs permet le déploiement de groupes de conteneurs dans un réseaux virtuelle.
- Démo.

Azure Container Service - Exercice

- Créer un conteneur à partir de l'image créée dans l'exercice crs à l'aide du cli.
- Créer une ressource ARM pour créer un service acs, toujours à partir de notre image.
- Créer une application 2 qui communique avec notre application de l'image 1.
- Créer une image 2.
- Déployer un groupe de conteneur :
 - Conteneur 1 à partir de l'image 1
 - Conteneur 2 à partir de l'image 2

Azure App Service



Tight integration w/
Docker Hub, Azure Container Registry



Built-in CI/CD w/
Deployment Slots



Intelligent diagnostics &
troubleshooting, remote debugging

Fully managed platform



Automatic scaling
and load balancing



High availability
w/ auto-patching



Backup & recovery

Flexibility & choices



From CLI, portal, or
ARM template



Single Docker image,
multi container w/ Docker compose,



IntelliJ, Jenkin, Maven Visual Studio family


Basics

Docker

Monitoring

Tags

Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#) 


Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * 

Contoso Hotels



Resource Group * 

[Create new](#)

Instance Details

Name *

Web App name.

.azurewebsites.net

Publish *



Code

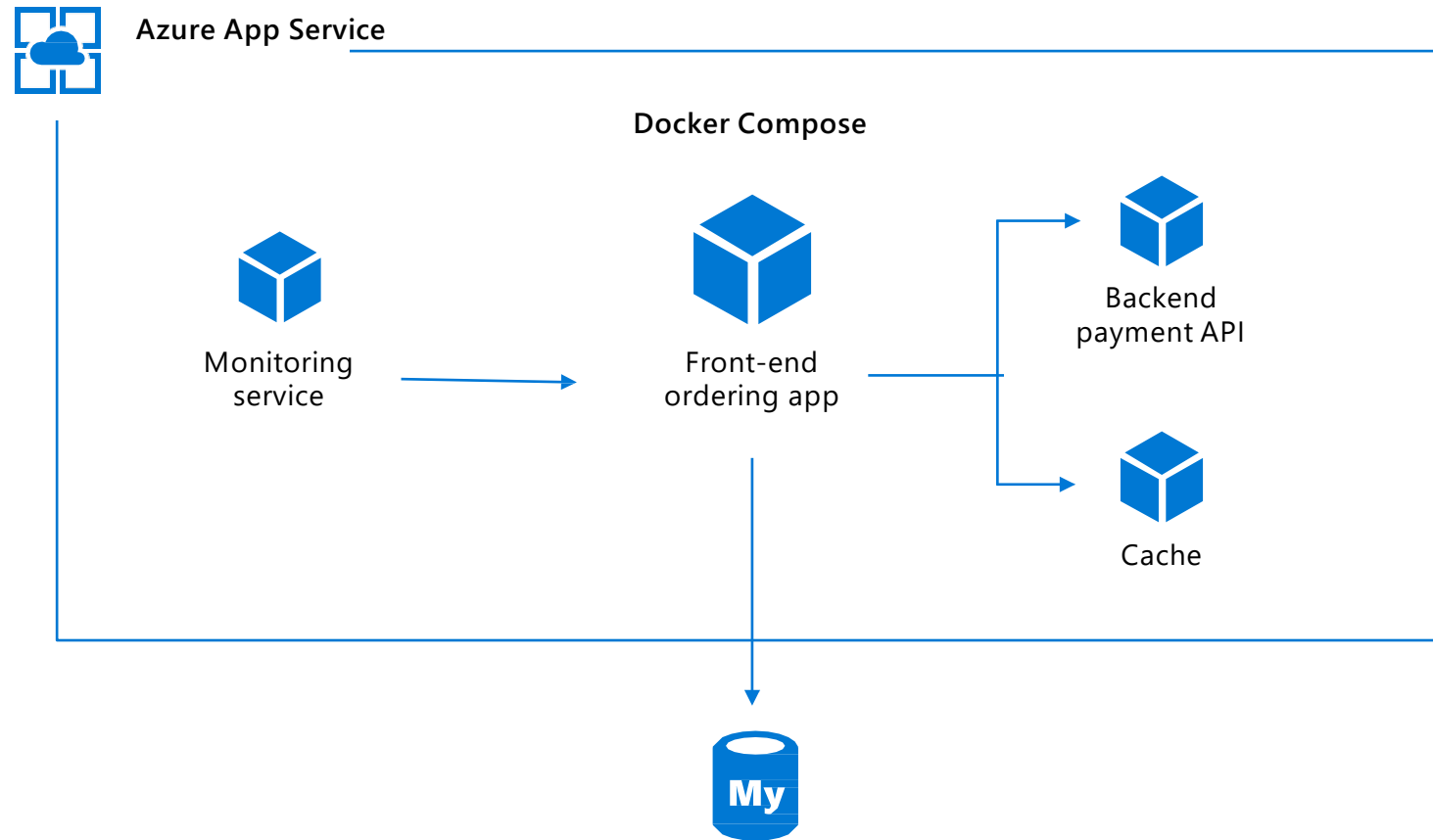


Docker Container

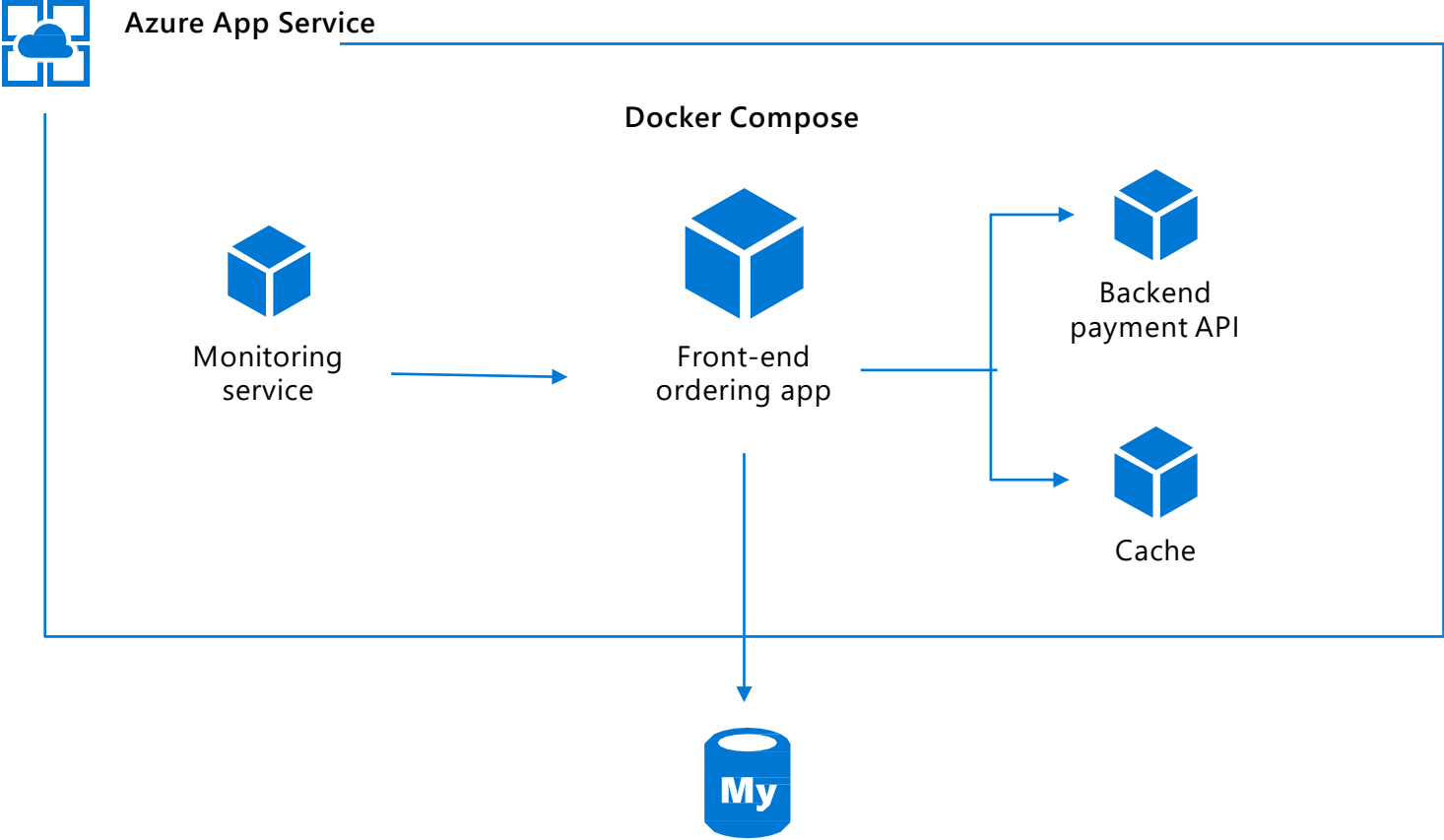
Azure App Service – Azure CLI

- `$ az webapp create --name $appName --plan AppServiceLinuxDockerPlan --deployment-container-image-name $dockerHubContainerPath -resource-group $myResourceGroup`

Azure App Service – Multi-conteneur



Multi-container sample architecture



Azure App Service – Azure CLI

- `$ az webapp create --resource-group $myResourceGroup --plan AppServiceLinuxDockerPlan --name $appName --multicontainer-config-type compose --multicontainer-config-file docker-compose-wordpress.yml`

AppServiceLinuxDocker17303 | Container settings (Classic)

App Service | Directory: Microsoft

Search (Ctrl+/)

Deployment Center (Classic)

Settings

Configuration

Container settings (Classic)

Authentication / Authorization

Authentication (preview)

Application Insights

Identity

Backups

Custom domains

TLS/SSL settings

Networking

Scale up (App Service plan)

Scale out (App Service plan)

WebJobs

Push

MySQL In App

Properties

Locks

Service plan

Public Private

Configuration File

Choose File No file chosen

Configuration

```
wordpress:
  depends_on:
    - db
  image: wordpress:latest
  ports:
    - "8000:80"
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: password
```

Continuous Deployment

On Off

Webhook URL [show url](#)

Copy

Logs

```
2021-03-07T11:22:11.071Z INFO - Pull image successful, time taken: 2 minutes and 27 seconds
2021-03-07T11:22:11.094Z INFO - Starting container for site
2021-03-07T11:22:11.094Z INFO - docker run -d -p 1371:80 --name appservicelinuxdocker17303_wordpress_0_e737865d -e WEBSITE_SITE_NAME=AppServiceLinuxDocker17303 -e WEBSITE_AUTH_ENABLED=False -e WEBSITE_ROLE_INSTANCE_ID=0 -e WEBSITE_HOSTNAME=appservicelinuxdocker17303.azurewebsites.net -e WEBSITE_INSTANCE_ID=abf0220645ea22573568610313bec0b920fd62831eeab85513f40eb93050918a wordpress:latest
2021-03-07T11:22:11.094Z INFO - Logging is not enabled for this container.
Please use https://aka.ms/linux-diagnostics to enable logging to see container logs here.
2021-03-07T11:23:40.106Z INFO - Started multi-container app
2021-03-07T11:23:40.126Z INFO - Initiating warmup request to container appservicelinuxdocker17303_wordpress_0_e737865d for site appservicelinuxdocker17303
2021-03-07T11:23:40.144Z INFO - Container appservicelinuxdocker17303_wordpress_0_e737865d for site appservicelinuxdocker17303 initialized successfully and is ready to serve requests.
```

Download

Refresh