Create a Kubernetes Cluster with AKS and Terraform

Overview

1. Get infra-as-code solution from git
2. Adjust terraform variables in a git branch
3. Setup Azure service principle and login
4. Set Azure blob storage to store terraform state for the environment
5. Create a resource group with k8s cluster, Cosmos DB account

Create blob storage to store terraform state files

1. Open Azure portal at https://portal.azure.com
2. Create "**afsdemo**" resource group
3. Create "**afsdemo$name**" storage account, replace $name with your name
4. Create "**dev**" container in the storage account
5. Locate and copy **storage account key** from the azure portal

> Note! "dev" stands for environment name. Each environment gets a separate cluster instance

Create service principal in Azure Portal

1. See https://docs.microsoft.com/en-gb/azure/azure-resource-manager/resource-group-create-service-principal-portal for details
2. Create new App registration in Azure Active Directory with following params:

   Name: **afsdemo**

   Application type: Web app / API

   Sign-on URL: https://afsdemo
3. Copy application ID (this will be the ARM_CLIENT_ID value in later example)
4. Generate new key named "**ci**" for the app registration and copy its value (this will be the ARM_CLIENT_SECRET variable)

Set azure login credentials

Set environment variables for your subscription:

```
// Locate your subscription details from Azure CLI
az account list

// Set following environment variables, replacing values with
actual result from previous command's output
set ARM_SUBSCRIPTION_ID=xxxxxxxx
set ARM_TENANT_ID=xxxxxxxx

// Set service principal environment variables from step one
set ARM_CLIENT_ID=xxxxxxx
set ARM_CLIENT_SECRET=xxxxxxx
```

> Note! these settings exist only for current terminal session and will be lost, if you close the terminal window.This is a desired behavior for CI/CD pipeline configuration!

Set environment variables for your subscription:

```
// Assing contributor permissions to the created service
principal
az role assignment create --assignee %ARM_CLIENT_ID% --role
Reader
az role assignment create --assignee %ARM_CLIENT_ID% --role
Contributor

// login using this service principal
az login --service-principal --username %ARM_CLIENT_ID%
--password %ARM_CLIENT_SECRET% --tenant %ARM_TENANT_ID%
```

Initialize and execute terraform plan

1. Clone afs-demo-infra.git repository from https://github.com/alekseigurba/afs-demo-infra.git
2. Navigate to cloned afs-demo-infra directory
3. Create a branch for your personal changes using "git checkout -b $lastname" command
4. Navigate to **afs-demo-infra/dev/** directory
5. Execute following command to create a terraform plan (replace the account name and key):

```
terraform init \
-backend-config="storage_account_name=afsdemo$name" \
-backend-config="container_name=dev" \
-backend-config="access_key=<storage account key>" \
-backend-config="key=dev.tfstate"

// Note! be sure to remove line breaks, if you run it
with CMD or powershell terminal
```

Sample output:

```
c:\dev\cloudws\afs-demo-infra\dev>terraform init
-backend-config="storage_account_name=afsdemostate"
-backend-config="container_name=dev"
-backend-config="access_key=HIDDEN"
-backend-config="key=dev.tfstate"
Initializing the backend...
Successfully configured the backend "azurerm"! Terraform
will automatically
use this backend unless the backend configuration
changes.
Initializing provider plugins...
- Checking for available provider plugins on
https://releases.hashicorp.com...
- Downloading plugin for provider "azurerm" (1.5.0)...
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running
"terraform plan" to see
any changes that are required for your infrastructure.
All Terraform commands
should now work.
If you ever set or change modules or backend
configuration for Terraform,
rerun this command to reinitialize your working
directory. If you forget, other
commands will detect it and remind you to do so if
necessary.
```

6. Create terraform plan

```
terraform plan -out out.plan -var
"client_id=%ARM_CLIENT_ID%" -var
"client_secret=%ARM_CLIENT_SECRET%"
```

It will match current state file from the state blob storage, if it exists, to the current terraform definition in our YML files.

7. Apply terraform plan

```
terraform apply out.plan
```

It will apply planned changes to the infrastrcuture

8. Browse to Azure portal to see the created infrastructure
    a. Open Kubernetes dashboard
    b. Execute commands listed on the welcome screen

```
az aks get-credentials --resource-group
afsdemo-$user-dev --name afsdemo-$user-dev
az aks browse --resource-group afsdemo-$user-dev
--name afsdemo-$user-dev
```

    c. Explore the dashboard

See following URLs for details

- https://docs.microsoft.com/en-us/azure/terraform/terraform-create-k8s-cluster-with-tf-and-aks
- https://www.hashicorp.com/blog/kubernetes-cluster-with-aks-and-terraform