ARM Workshop

0 – Setup maszyny

0.0

Upewnij się, że Twoja maszyna jest przygotowana do pracy.
Minimalny zestaw oprogramowania, potrzebny by z sukcesem zakończyć cały warsztat to:

1) Azure CLI 2.0
2) Visual Studio Code
3) Dodatki do Visual Studio Code takie jak:
    a. Azure Account
    b. Azure ARM Template Helper
    c. Azure CLI Tools
    ~~d. Azure Resource Manager Tools~~
    ~~e. Azure Resource Manager Snippets~~

Żeby sprawdzić czy wszystkie dodatki zostały zainstalowane prawidłowo stwórz:

1) Plik azscript.azcli w którym umieścisz polecenie tworzące grupę zasobów w Twojej subskrypcji
2) Plik azuredeploy.json, który będzie twoim pierwszym pustym templatem ARM (https://github.com/mifurm/ARMWorkshop)

Proponuję każde zdanie trzymać w oddzielnym folderze – będzie łatwiej nam zarządzać templateami.

*Jeśli masz ochotę, możesz stworzyć konto na GitHub lub Azure DevOps, gdzie będziesz trzymał wszystkie swoje template'y.

Zadanie 1 – Wykonanie pierwszego template'u (20 min)

W tym zadaniu utworzymy pierwszą grupę zasobów i wykonamy pierwszy template w ramach niej.

W tym celu:

- Zaloguj się za pomocą Azure CLI do Azure poprzez procedurę *az login*
- Stwórz grupę zasobów w Azure wykonując polecenie *az group create*
- Utwórz najprostszy template, który tworzy konto składowania danych w ramach grupy zasobów.
- Wykorzystaj template: https://github.com/mifurm/ARMWorkshop/blob/master/1.BasicTemplate/basictemp1.json
- Zwróć uwagę na to, by template miał możliwość wykonania się zawsze. **Pytanie: Co zrobić by tak było? Jaki element template'u trzeba zmienić?**

Zadanie 2 – Twoja pierwsza maszyna (30 min)

W tym zadaniu poznamy działanie funkcji dependsOn i utworzymy naszą pierwszą maszynę wirtualną.

W tym celu:

- Wykorzystaj template: https://github.com/mifurm/ARMWorkshop/tree/master/2.SampleVM
- Dostosuj template tak, aby był zgodny z przyjętą konwencją nazewniczą przez Ciebie. Jeśli jej nie masz, wystarczy, że wszystkie zasoby będą zawierały Twój login lub inny string w nazwie (oczywiście tam, gdzie to możliwe)
- Następnie dodaj do template'u jeszcze 2 dyski jako dyski DATA
- Jeśli udało się dodać dyski, zmień template tak, by utworzyć dodatkową kartę sieciową i dołączyć ją do maszyny.
  Druga karta sieciowa nie musi mieć publicznego adresu IP i wystarczy by była w tym samym subnecie.
- ***Jeśli Ci się udało, zmień template tak, by dodać jeszcze jeden subnet i by każda karta sieciowa była w innym subnecie**.

Zadanie 3 – Funkcja Copy

W tym zadaniu zobaczysz, jak działa funkcja copy, która ułatwi nam tworzenie środowisk z powtarzalnymi obiektami jak dyski, sieci, subnet'y.

W tym celu:

- Zapoznaj się z przykładem poniżej https://github.com/mifurm/ARMWorkshop/blob/master/3.FunctionCopy/copy.simple.json
- Uruchom go i sprawdź jakie zasoby powstały
- Teraz, wróć do swojego przykłady z Zadania 2 i zastąp to, co możliwe, funkcją copy
- Im więcej zasobów uda Ci się zastąpić tym lepiej.
- *Sprawdź z której modelu funkcji copy lepiej się korzysta i dla których zadań.

Zadanie 4 – Zabawy z maszyną i Load Balancerem (25 min)

W tym zadaniu zobaczysz, jak połączyć naszą wiedzę z poprzednich zadań i dodać do naszego wdrożenia Load Balancer.

W tym celu:

- Zapoznaj się z przykładem poniżej https://github.com/mifurm/ARMWorkshop/blob/master/4.SampleVMwithLB/azuredeploy.json
- Uruchom go i sprawdź jakie zasoby powstały
- Zobacz, czy gdzieś nie możesz znowu zastosować funkcji by Twój template był bardziej przejrzysty.
- Dodaj do template'u reguły NSG, które pozwolą na komunikację po porcie 22 i 80. Najlepiej byś nadał je tak, by były dołączone do subnetu.
- Pomyśl, jak wykorzystać tutaj ponownie funkcje.

Zadanie 5 – VM with Extension

W tym zadaniu zobaczysz, jak możesz konfigurować maszynę wirtualną po wykonaniu jej deploymentu w Azure.

W tym celu:

- Zapoznaj się z przykładem poniżej https://github.com/mifurm/ARMWorkshop/blob/master/7.%20Extensions/azuredeploy.json
- Uruchom go i sprawdź jakie zasoby powstały
- Zobacz, jak wygląda uruchomienie domyślnego skryptu.
- A teraz zmień swój poprzedni przykład tak, by na obu maszynach został zainstalowany serwer Apache lub NGINX (co wolisz).
- Jeśli Ci się uda, to na każdy serwer załaduj również domyślną stronę HTML, która się pojawi w odpowiedniej ścieżce
- *Czy model uruchomienia skryptu przez VM Extension to jedyna możliwość?

Zadanie 6 – VM with KeyVault

KeyVault to w skrócie „specjalny" pojemnik na sekrety (hasła, certyfikaty czy konfiguracje)

Zobacz, jak wygląda skrypt:
https://github.com/mifurm/ARMWorkshop/blob/master/6.%20KeyVault/sqlserver.json

Następnie, utwórz własnego KeyVaulta i zmień skrypt tak, by pobierać parametry ustawień maszyny właśnie z KeyVault.

Zadanie 7 – Nested Templates czyli jak podzielić duże zadanie na kawałki

W tym zadaniu przyda Ci się konto składowania danych lub github (wszystko, co pozwoli Ci wykonać referencję do pliku dostępnego publicznie)

Rzuć okiem jak zbudowane są tzw. Nested Templates:
https://github.com/mifurm/ARMWorkshop/tree/master/8.NestedTemplate/NestedTemplate

Żeby sprawdzić, że umiesz się nimi posługiwać zrób następujące zadanie:

- Popraw przykładowy template tak by udało Ci się go uruchomić – ten który jest, ma pewien błąd
- Następnie zobacz, jak wygląda linked template i nested template. To ważne by zrozumieć, jaka jest między nimi zależność.
- A następnie… wróć do swoich maszyn wirtualnych z zadania 4 i postaraj się podzielić budowanie całego środowiska na oddzielne, zależne template. Idealny scenariusz to taki, gdzie jak najwięcej elementów np. sieć, NSG, adresy IP i inne elementy są w oddzielnych templatech – robimy to dla „sportu", by sprawdzić, jak to działa.
- Zacznij od mały kroków, skończysz na całym wdrożeniu.
  Przykład możesz zobaczyć tutaj.

Zadanie 8 – Custom Function

Przykład własnej funkcji oraz wykorzystanie jej możesz zobaczyć tutaj:
https://github.com/mifurm/ARMWorkshop/blob/master/9.%20CustomFunction/fun.json

Zadanie 9 – Custom Role

Prosty przykład własnej roli, która jest ważna w ramach jednej w subskrypcji:
https://github.com/mifurm/ARMWorkshop/tree/master/11.%20CustomRole

Zadanie 10 – Custom Policy

Prosty przykład polityki, którą możemy zbudować na własne subskrypcji:
https://github.com/mifurm/ARMWorkshop/tree/master/10.%20Policy

Zadanie 11 – Azure Building Blocks

<TBD>

Zadanie 12 – Deployment do wielu grup zasobów

<TBD>

Zadanie 13 – Deployment do wielu subskrypcji

<TBD>