Opis sytuacji:

Pamiętasz swoje zadanie z poprzedniego tygodnia? Mam nadzieję, że tak... 🙂 jeśli nie, wróć proszę do poprzedniego tygodnia. Jeśli tego zadania jeszcze nie skoczyłeś to proszę zrób to. To mega ważne bo tutaj budujemy wiedzę 🙂

Do rzeczy... W poprzednim zadaniu zaplanowałeś migrację do Azure wykorzystującą koncepcję Lift & Shift.

Głównie planowałeś migrację z wykorzystaniem maszyn wirtualnych. Tym razem zastanów się, które z tych systemów i pod kątem jakich kryteriów z migrowałbyś do takich usług jak AKS, Service Fabric czy App Service Environment.

Zadania:

* #TYDZIEN5.1 Migracja do usług PaaS

Na bazie doświadczeń i zadania poprzedniego opisz min. 3 różne systemy, które byś zmigrował do usług, które poznałeś. Tym razem, ponieważ są tylko 3 systemy, to poproszę o min. 3 kryteria per system i uzasadnienie, kiedy to ma sens, a kiedy nie. Możesz wybrać dowolny system ☺ nie narzucam. Chcę zobaczyć, jak myślisz, planujesz i wybierasz usługi pod konkretne rozwiązania.

* #TYDZIEN5.2 Czego nie migrować do PaaS?

Jeśli zadanie pierwsze przyszło nie łatwo, to może wybierz 3 systemy, których nigdy byś nie migrował do poznanych usług? Podaj 2 przykłady i min. 2 powody.

* #TYDZIEN5.3 Service Fabric – dlaczego jest tak dobry?

Service Fabric to genialne rozwiązanie. Jeśli dwa zadania wyżej nie są dla Ciebie, to musisz poznać Service Fabric. Żeby docenić jego zalety, polecam prosty lab: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-quickstart-dotnet>

Zrób go porządnie, poczytaj też o Architekturze SF <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-architecture>, o modelu programowania: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-choose-framework> i procesie aktualizacji klastra: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-application-upgrade>

Zalety Service Fabric:

Świetnie pasuje do budowy rozproszonego systemu z wieloma niezależnymi komponentami (mikroserwisy)

W działaniu podobna do np. Kubernetesa, przewyższa go w zarządzaniu życiem wdrożonych komponentów: może zarządzać aktualizacją oprogramowania i przywracać poprzednią wersję w przypadku, gdy nowa wersja nie działa

Dostępna on-premise i w chmurze – bez względu na dostawcę

Dostępna dla Windows i Linuxa

Zapewnia świetną skalowalność i dostępność aplikacji

Wspiera zarówno bezstanowe i stanowe aplikacje

SDK pozwala uruchamiać i testować aplikację w środowisko niemal identycznym z produkcyjnym

Dostępny framework Reliable Service ułatwia tworzenie aplikacji (oferuje zgłaszanie stanu usługi, wykrywanie i komunikację z innymi usługami, otrzymywanie powiadomień o zmianach konfiguracji)

Dostępny framework Reliable Actor umożliwia implementowanie systemów wymagających np. wydajnej logiki obliczeniowej (data analytics, przetwarzanie grafów) w ramach jednowątkowego przetwarzania