Azure CLI를 사용하여 Linux 가상 머신 만들기

Windows 에서 DotNet Core 설치 후 환경변수 설정
$ setx ASPNETCORE_ENVIRONMENT "Development"

리소스 그룹 생성
$ az group create -n TestRG01 -l eastus

가상 네트워크 및 서브넷 만들기
$ az network vnet create -g testrg01 -n vnet01 --address-prefix 10.0.0.0/16 --subnet-name subnet01 --subnet-prefix 10.0.0.0/24

Public IP 생성 : 기본 주소가 동적이므로 --domain-name-label 매개 변수를 사용하여 명명된 DNS 항목을 만든다.
// dns name 은 소문자로만.
$ az network public-ip create -g testrg01 -n testip --dns-name mingyudns

2 개 vm 생성시에…
$ az network public-ip create -g testrg01 -n testip02 --dns-name mingyudns02

// 출력
"fqdn": "mingyudns.eastus.cloudapp.azure.com",
"resourceGuid": "d87dc181-0fa1-4191-93f5-f5b915894811",

Network Security Group 생성
$ az network nsg create -g testrg01 -n testsg

// 출력
"resourceGuid": "407c7fe6-4bcf-451a-8b1d-7fda8ae469cc"

Network Security Group 에 SSH Allow 룰 생성/추가
$ az network nsg rule create -g testrg01 --nsg-name testsg -n ruleSSH --protocol tcp --priority 1000 --destination-port-range 22 --access allow

Network Security Group 에 WEB Allow 룰 생성/추가
$ az network nsg rule create -g testrg01 --nsg-name testsg -n ruleWEB --protocol tcp --priority 1001 --destination-port-range 80 --access allow

추가한 룰 확인
$ az network nsg rule list -g testrg01 --nsg-name testsg

 # NIC 생성 : VM 크기에 따라 복수 가능.
 # VM 생성시 NIC 사전 생성 없이 만들어둔 AVSet 에 속하게하고, Vnet 과 Subnet 만 설정하면 자동으로 NIC 가 만들어지게 할 수도 있다. 아래 VM 생성에서 보기.

$ az network nic create -g testrg01 -n nic01 --vnet-name vnet01 --subnet subnet01 --public-ip-address testip --network-security-group testsg

$ az network nic create -g testrg01 -n nic02 --vnet-name vnet01 --subnet subnet01 --public-ip-address testip02 --network-security-group testsg

// 출력
"resourceGuid": "89db5185-3fc0-4e56-a0ca-9766b18903f5"

 # 가용성 집합 만들기 : 장애 도메인은 공통의 전원 및 네트워크 스위치를 공유하는 가상 머신 그룹을 정의합니다. 기본적으로 가용성 집합 안에 구성된 가상 머신은 최대 3개의 장애 도메인에 분산되어 있습니다. 업데이트 도메인은 동시에 다시 부팅할 수 있는 가상 머신 그룹과 기본 물리적 하드웨어를 나타냅니다. 계획된 유지 보수 중에 업데이트 도메인의 재부팅 순서는 순차적으로 진행되지 않을 수 있으며, 한 번에 하나의 업데이트 도메인만 재부팅. Azure는 가용성 집합에 VM을 배치할 때 VM을 전체 장애 및 업데이트 도메인에 자동으로 분산.

$ az vm availability-set create -g testrg01 --name testavset

추가 옵션
--platform-fault-domain-count 2
--platform-update-domain-count 5

// 기본값으로 장애 도메인 2 과 업데이트 도메인 5
  "platformFaultDomainCount": 2,
  "platformUpdateDomainCount": 5,

 # VM 생성 시에 붙여줄 SSH Key 생성 : VM 생성시 자동으로 키 생성하려면 --generate-ssh-keys 옵션
$ ssh-keygen -t rsa -b 2048

Generating public/private rsa key pair.
Enter file in which to save the key (/Users/codingstar/.ssh/id_rsa): **/Users/codingstar/.ssh/setvm_rsa**

// 확인
$ cat ~/.ssh/setvm_rsa
$ cat ~/.ssh/setvm_rsa.pub

VM 생성 : 퍼블릭 키 대입 - 생성시 기다리지 않으려면 : --no-wait 옵션.
// 아래와 같이 생성하면 기본 VM Size 는 : Standard DS1 v2 (1 vcpus, 3.5 GiB memory)

$ az vm create -g testrg01 -n vm01 -l eastus --availability-set testavset --nics. nic01 --image ubuntults --admin-user mingyu --ssh-key-value /users/codingstar/.ssh/test0205_rsa.pub

// NIC 생성 않고, Vnet 과 Subnet 만 지정하여 AVSet 에 속하게 해서 VM 생성하기.
$ az vm create -g testrg01 -n vm03 -l eastus --availability-set testavset --vnet-name vnet01 --subnet subnet01 --size Standard_DS1_V2 --image ubuntults --admin-user mingyu --ssh-key-value /users/codingstar/.ssh/test0205_rsa.pub

Asset 의 하드웨어 클러스터에서 가용성 집합에 대한 사용 가능한 모든 크기를 나열
$ az vm availability-set list-sizes -g testRG01 -n testavset --output table

# VM 에 ssh private key 로 접속
$ ssh -i /Users/codingstar/.ssh/setvm_rsa mingyu@40.117.138.103

# 업데이트 후 nginx 설치
$ sudo apt-get -y update
$ sudo apt-get -y upgrade
$ sudo apt-get install -y nginx

index.html 수정을 위해서, Html 폴더 내의 아래 파일 수정.

$ cd /var/www/html
index.nginx-debian.html

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

 # 로드밸런스 만들기

// LB 에 연결한 Public IP 주소 생성.
$ az network public-ip create -g testrg01 -n lbIP

// LB 생성.
$ az network lb create -g testrg01 -n testlb --frontend-ip-name frontEndIP --backend-pool-name backEndPool --public-ip-address lbIP

// 상태 프로브 만들기
$ az network lb probe create -g testrg01 --lb-name testlb -n testprobe --protocol tcp --port 80

// 부하 분산 장치 규칙 만들기 : 프런트 엔드 IP 구성 및 트래픽을 수신할 백 엔드 IP 풀과 필요한 원본 및 대상 포트를 함께 정의합니다. 정상 VM만 트래픽을 수신하도록 하려면 사용할 상태 프로브도 정의
$ az network lb rule create -g testrg01 --lb-name testlb -n testlbrule --protocol tcp --frontend-port 80 --backend-port 80 --frontend-ip-name frontEndIP --backend-pool-name backEndPool --probe-name testprobe

// NIC 업데이트 : **ipConfigurations 는 NIC 의 Settings 의 IP configurations 에 나옴.**
// 아래 실행 후, Settings 의 Properties 에 가서 확인하면 LB 가 셋팅되어 있음.

$ az network nic update -g testrg01 -n nic01 --add ipConfigurations.
[name=ipconfig1].loadBalancerBackendAddressPools id=/subscriptions/28697aaf-daed-4d77-b02a-c6dc954a9585/resourceGroups/testrg01/providers/Microsoft.Network/loadBalancers/testlb/backendAddressPools/backEndPool

// NIC 를 새로 생성하여 대체 하는 방법.
$ az network nic create -g testrg01 -n nic00 --vnet-name vnet01 --subnet subnet01 --network-security-group testsg --lb-name testlb --lb-address-pools backEndPool

$ az vm nic set --nics nic00 -g testrg01 --vm-name vm03

// 부하 분산장치의 IP 가져오기.
$ az network public-ip show -g testrg01 -n lbIP --query [ipAddress] --output tsv

// 기존 NIC 삭제
$ az network nic delete -n vm03VMNic -g testrg01

// 기존 2개의 NIC 에 연결된 Public IP 분리 후 삭제
$ az network nic ip-config update -n ipconfig1 -g testrg01 --nic-name nic02 --remove PublicIpAddress

$ az network public-ip delete -n testip -g testrg01

———————————————

Azure WebApp

// 1. Git을 사용하여 앱에 .NET Core 코드를 배포하기 위해 로컬 컴퓨터에서 닷넷코어로 웹 프로젝트 만들어서 사전 테스트. 하지만 닷넷코어가 깔린 VM 이라야 설치 가능.

Windows 에서 DotNet Core 설치 후 환경변수 설정
$ setx ASPNETCORE_ENVIRONMENT "Development"

 # Linux 가상 머신의 가용성 관리
https://docs.microsoft.com/ko-kr/azure/virtual-machines/linux/manage-availability

 # Azure Resource Manager 템플릿의 구조 및 구문 이해
https://docs.microsoft.com/ko-kr/azure/azure-resource-manager/resource-group-authoring-templates?toc=%2Fazure%2Fvirtual-machines%2Flinux%2Ftoc.json

$ mkdir hellodotnetcore
$ cd hellodotnetcore/
$ dotnet new web
$ dotnet run
Now listening on: https://localhost:5001
Now listening on: http://localhost:5000

$ az webapp list-runtimes -> .Net Core 는 현재 지원은 되지만 표시가 안되고 있음?

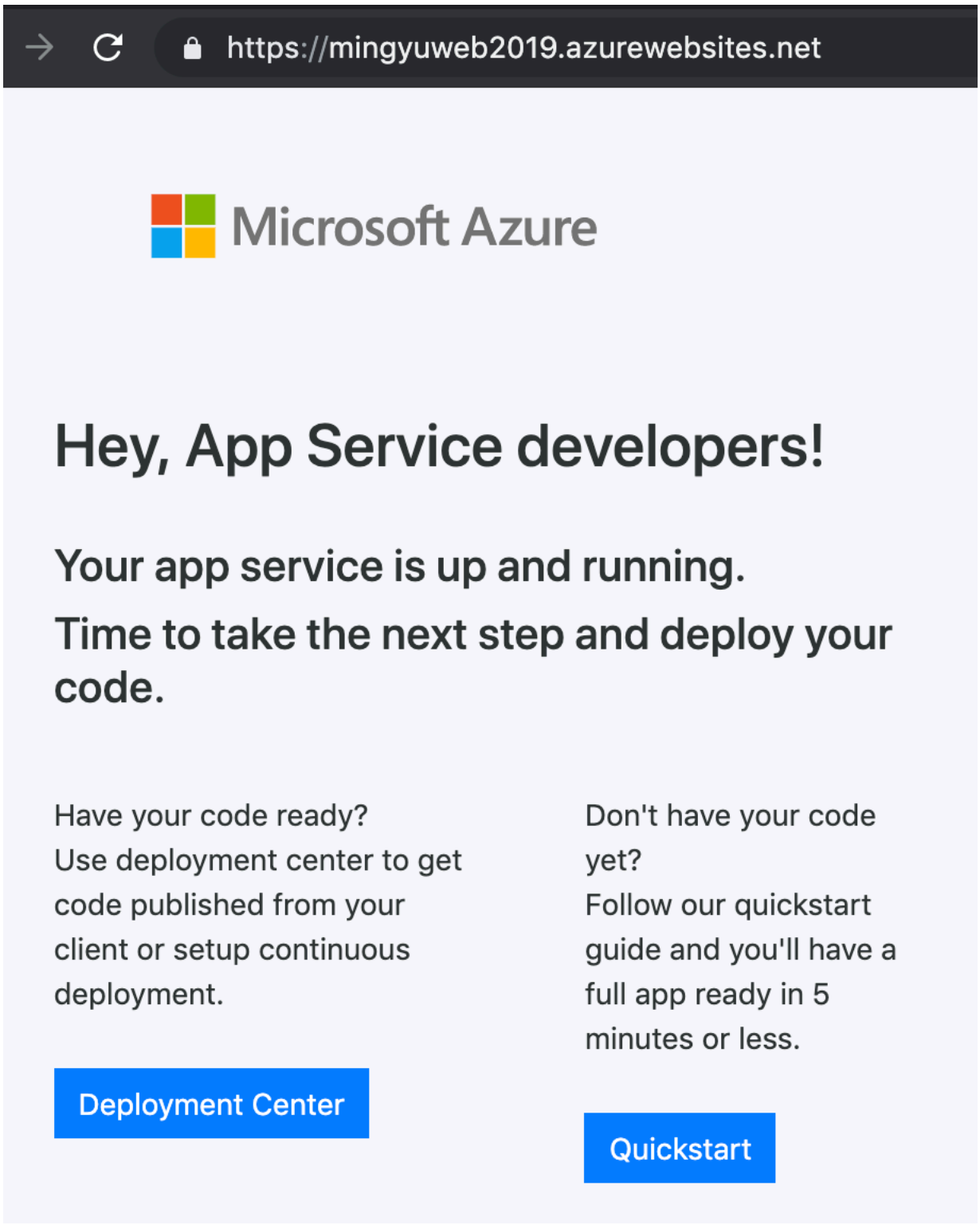
 # App Service Plan 만들기.
$ az appservice plan create --name mingyuWSPlan -g WebAppRG0609 --sku B1 --is-linux

 # Appservice plan list 보기
$ az appservice plan list -g WebAppRG0609

 # App Service Plan 정보 보기
$ az appservice plan show --name ASP-WebAppRG0609-a7c5 -g WebAppRG0609

 # WebApp 생성 : local git 에서 Deploy
$ az webapp create -g WebAppRG0609 --plan mingyuWSPlan --name mingyuWeb --runtime "DOTNETCORE|2.2" --deployment-local-git

// 출력 : deploymentLocalGitUrl
Local git is configured with url of 'https://mingyu2019@mingyuweb.scm.azurewebsites.net/mingyuWeb.git'



 # WebApp list 보기
$ az webapp list -g WebAppRG0609

 # Web App 정보 보기
$ az webapp show --name **mingyuWeb2019** -g WebAppRG0609

———————————————— Git

 # 생성한 dot net core 프로젝트 폴더 에서 아래와 같이 초기화 한다.

$ git init
Initialized empty Git repository in /Users/codingstar/Documents/
workspaces_AzureLecture/hellodotnetcore/.git/

$ git add .

$ git commit -m “first commit"

 # 배포 자격 증명 구성 : 한 번만 구성하면 모든 Azure 배포에 사용 가능.
$ az webapp deployment user set --user-name mingyu2019 --password abcde12345!

```
{
  "id": null,
  "kind": null,
  "name": "web",
  "publishingPassword": null,
  "publishingPasswordHash": null,
  "publishingPasswordHashSalt": null,
  "publishingUserName": "mingyu2019",
  "scmUri": null,
  "type": "Microsoft.Web/publishingUsers/web"
}
```


 # 로컬 Git 리포지토리에 Azure 원격을 추가
$ git remote add azure https://mingyu2019@mingyuweb.scm.azurewebsites.net:443/mingyuWeb.git

 # Azure 원격에 푸시하여 앱을 배포 : 배포 자격 증명 사용.
$ git push azure master

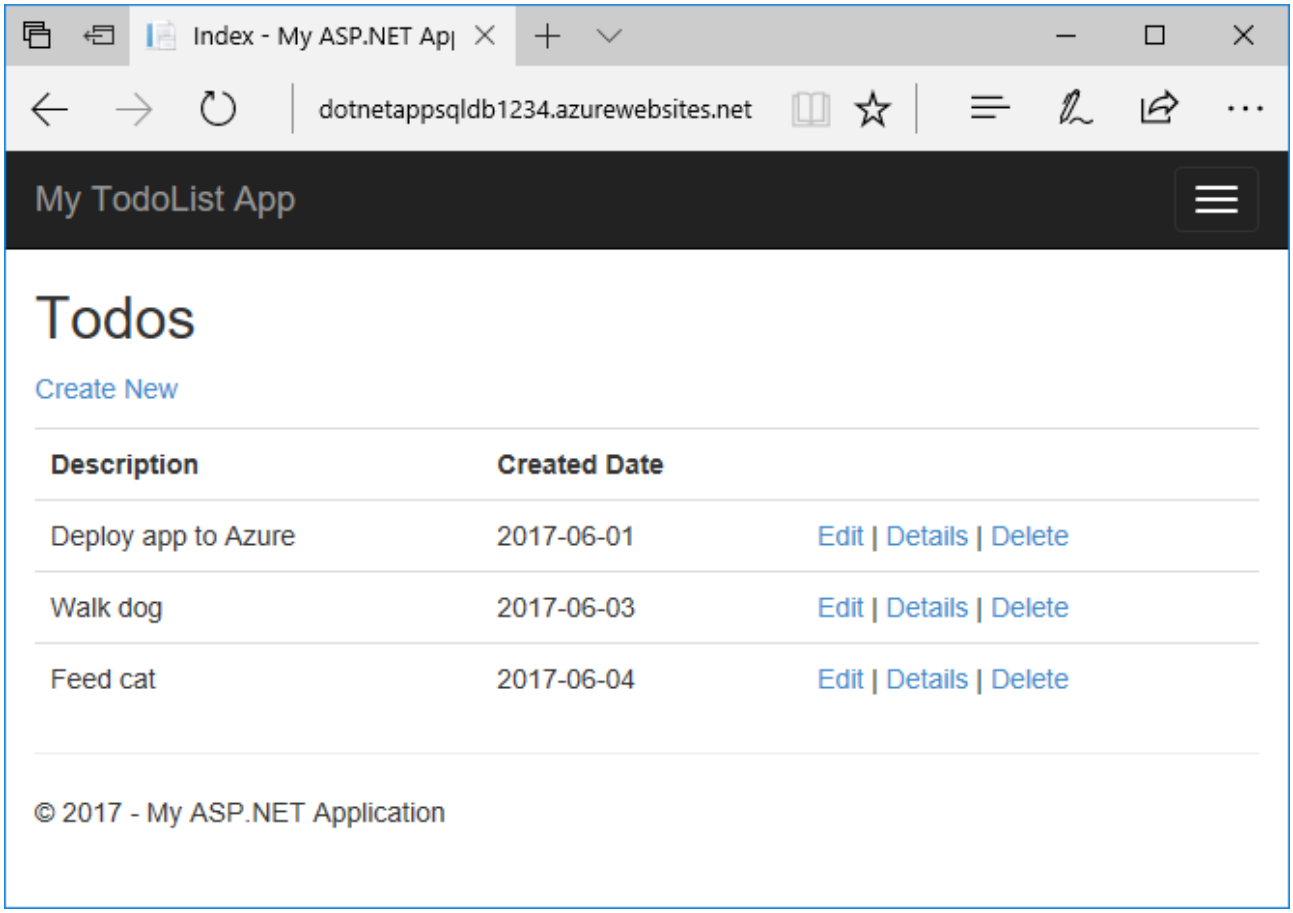
 # 수정 후 두번째 배포.
$ git commit -am “second commit"
$ git push azure master

// WebApp CLI Command
https://docs.microsoft.com/en-us/cli/azure/webapp?view=azure-cli-latest#az-webapp-list

// Azure에서 정적 HTML 웹앱 만들기
https://docs.microsoft.com/ko-kr/azure/app-service/app-service-web-get-started-html

// Linux의 App Service에서 ASP.NET Core 앱 만들기
https://docs.microsoft.com/ko-kr/azure/app-service/containers/quickstart-dotnetcore

SQL Server with WebApp on Linux



Azure에서 SQL Database 만들기
SQL Database에 .NET Core 앱 연결
Azure에 앱 배포
데이터 모델 업데이트 및 앱 다시 배포
Azure에서 진단 로그 스트림
Azure Portal에서 앱 관리

$ git clone https://github.com/azure-samples/dotnetcore-sqldb-tutorial
$ cd dotnetcore-sqldb-tutorial

$ dotnet restore
$ dotnet ef database update
$ dotnet run

 # SQL Server 생성
$ az sql server create -n minsqlsv -g testrg --admin-user mingyu --admin-password abcde12345!

서버 방화벽 규칙 구성 : 시작 IP 및 끝 IP가 0.0.0.0으로 설정되면, Azure 리소스들에 대해서만 열립니다.

$ az sql server firewall-rule create -g testrg --server minsqlsv -n AllowYourIP --start-ip-address 0.0.0.0 --end-ip-address 0.0.0.0

DB 생성
$ az sql db create -g testrg --server minsqlsv -n minDB --service-objective S0

Connection String

Server=tcp:<server-name>.database.windows.net,1433;Database=<DB Name>;User ID=<db-username>;Password=<db-password>;Encrypt=true;Connection Timeout=30;

// Client 종류 : ado.net / jdbc / odbc / php / php_pdo / sqlcmd

$ az sql db show-connection-string --server minsqlsv -n minDB -c ado.net

Linux App Service 배포 자격 증명 구성
$ az webapp deployment user set --user-name mingyu2019 --password abcde12345!

 # WebApp Plan 구성.
$ az appservice plan create -n testplan -g testrg --sku B1 --is-linux

// S3 플랜으로 리눅스 서버가 아닌 것으로 생성할 때.
$ az appservice plan create --name min2plan -g minrg --sku S3

WebApp 생성.
$ az webapp create -g testrg --plan testplan -n minWA0206 --runtime "DOTNETCORE|2.2" --deployment-local-git

// 출력되는 내용 중 deploymentLocalGitUrl 는 보관해둔다.
https://mingyu2019@az203mingyuwebapp.scm.azurewebsites.net/az203mingyuWebApp.git

https://jumingyu@minweb.scm.azurewebsites.net/minweb.git
https://jumingyu@minweb.scm.azurewebsites.net:443/minweb.git

 # WebApp 에 DB Connection 을 위한 환경 변수 구성 : SQL DB 의 Connection String 사용.

형식 : az webapp config connection-string set --resource-group myResourceGroup --name <app name> --settings MyDbConnection='<connection-string>' --connection-string-type SQLServer

// MyDbConnection 에 커넥션 스트링 저장.
$ az webapp config connection-string set -g testrg -n minWA0206 --settings MyDbConnection='Server=tcp:minsqlsv.database.windows.net,1433;Database=min

DB;User ID=mingyu;Password=abcde12345!;Encrypt=true;Connection Timeout=30;' --connection-string-type SQLServer

 # ASPNETCORE_ENVIRONMENT 설정
$ az webapp config appsettings set -n minWA0206 -g testrg --settings ASPNETCORE_ENVIRONMENT="Production"

// 코드가 프로덕션(즉, Azure 환경)에서 실행되고 있다고 감지되면, 연결 문자열을 사용하여 SQL Database에 연결.

# 프로덕션에서 SQL Database에 연결
$ git init
$ git add .
$ git commit -m "first commit - SQLDB Connect"

$ git remote add azsql https://jumingyu@minwa0206.scm.azurewebsites.net:443/minWA0206.git

$ git push azsql master

Azure CLI 로 Function 만들기

 # 로컬에서_런타임_정하고_Core_Tools 명령어로_Function_만들기
$ func init FirstFunctionProj

```
MinGyuMac:FunctionApp codingstar$ func init FirstFunctionProj
Select a worker runtime:
1. dotnet
2. node
3. python (preview)
4. powershell (preview)
Choose option: 1
dotnet

Writing /Users/codingstar/Documents/workspaces_AzureLecture/FunctionApp/FirstFunc
tionProj/.vscode/extensions.json
```

host.json 파일 편집하여 아래 그림의 내용 넣기

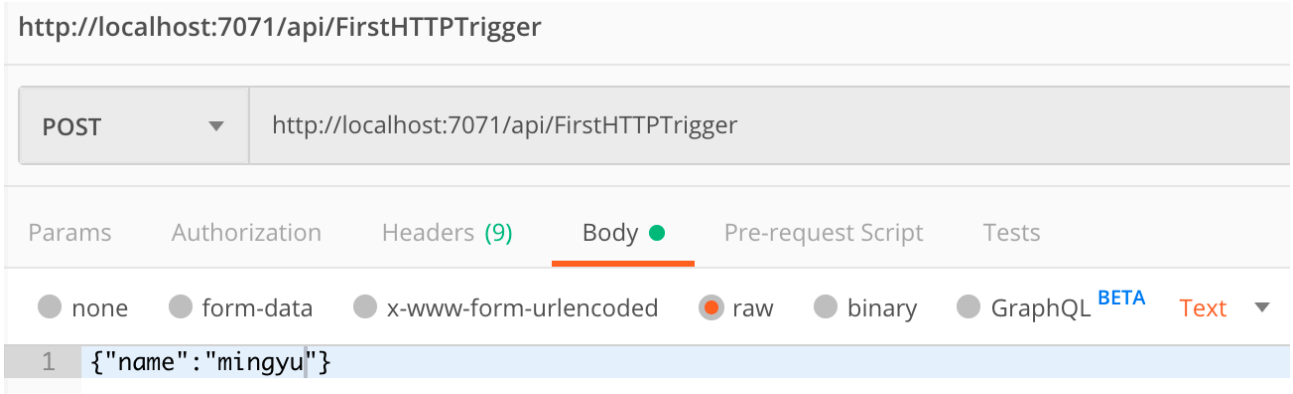
```
  GNU nano 2.0.6                File: host.json

{
    "version": "2.0",
    "extensionBundle": {
        "id": "Microsoft.Azure.Functions.ExtensionBundle",
        "version": "[1.*, 2.0.0)"
    }
}
```

# HTTPTrigger 템플릿 선택하여 Function 만들기
$ func new --name FirstHTTPTrigger --template "HttpTrigger"

# 빌드하기
$ func host start --build

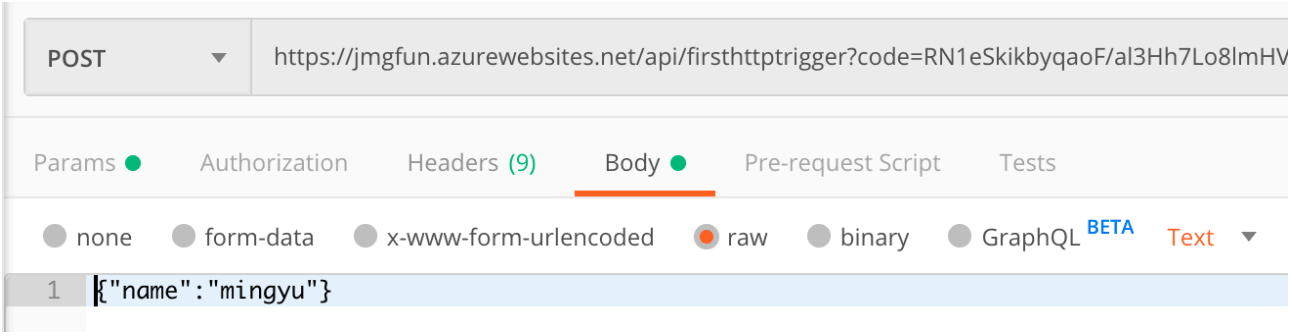
빌드_후에_로컬_브라우저에서_확인하기
[GET,POST] http://localhost:7071/api/FirstHTTPTrigger



# Azure에_FunctionApp_consumption_플랜으로_생성
$ az functionapp create -g testrg --consumption-plan-location eastus -n jmgfun --storage-account jmgfunstorage --runtime dotnet

# Publish_하기
$ func azure functionapp publish jmgfun

Invoke_해보기
Invoke url: https://jmgfun.azurewebsites.net/api/firsthttptrigger?code=RN1eSkikbyqaoF/al3Hh7Lo8lmHVwCoACYnafeCtan6bM3iPabDajw==



Azure Storage Blob Storage

 # Storage Account 생성
$ az storage account create --name az203mingyust -g az203mingyuRG --location eastus --sku Standard_LRS

 # Account Key 확인
$ az storage account keys list --account-name az203mingyust -g az203mingyuRG --output table

 # Connection String 확인
$ az storage account show-connection-string --name az203mingyust -g az203mingyuRG

 # Blob Storage 의 Container 만들기 : 리소스 그룹명 필요 없음.
$ az storage container create --name az203con --account-name az203mingyust

 # Blob 업로드
$ az storage blob upload --file t1_nayeon.jpg --container-name az203con --name t1_nayeon.jpg --account-name az203mingyust

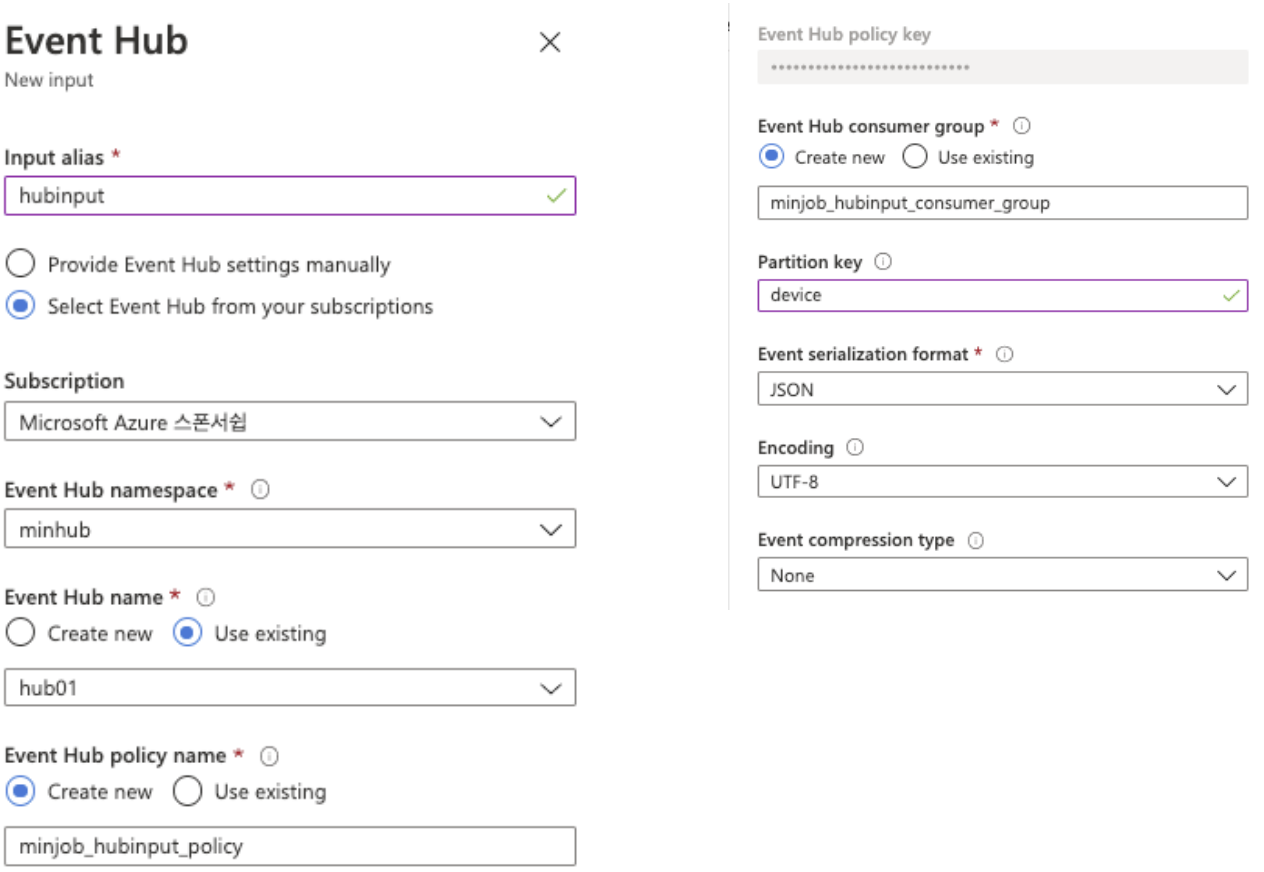
 # 폴더 생성하며 Blob 업로드
$ az storage blob upload --file twice.xml --container-name az203con --name xml/twice.xml --account-name az203mingyust

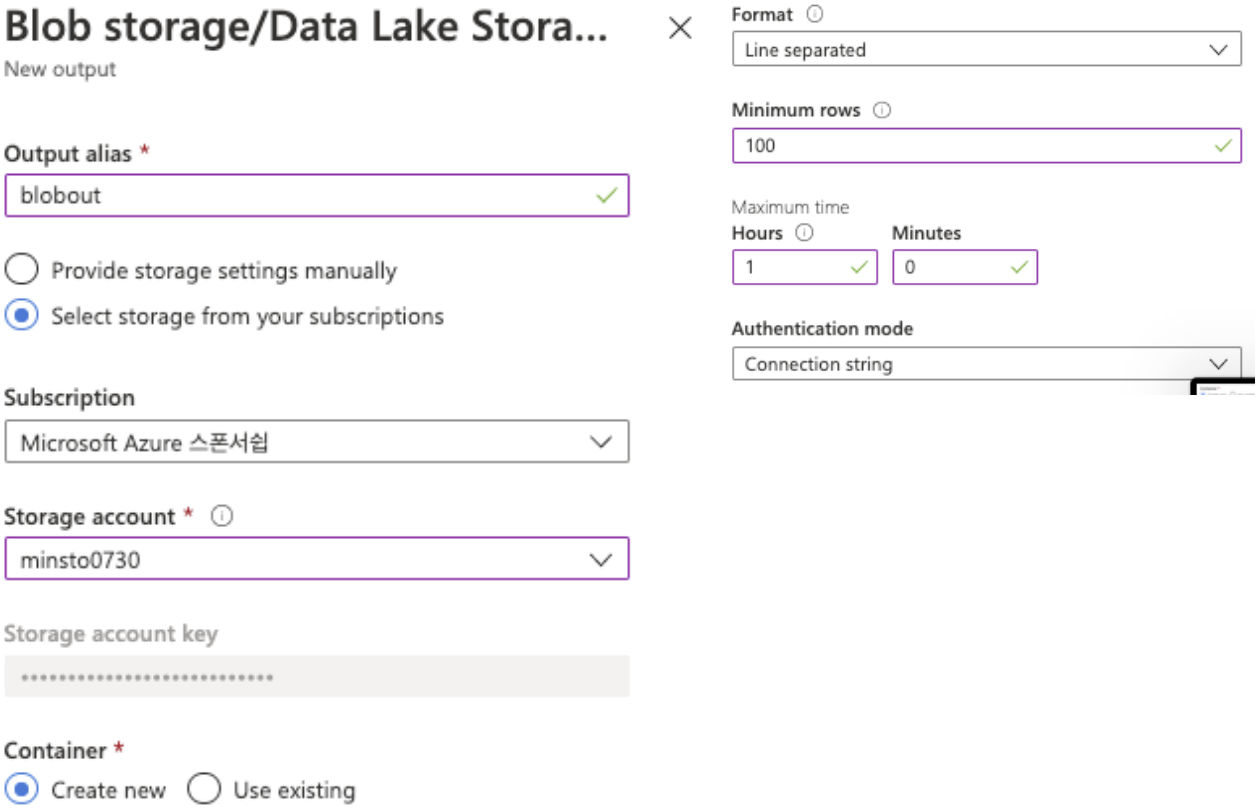
 # Blob 다운로드
$ az storage blob download --container-name az203con --name t1_nayeon.jpg --file ./temp.jpg --account-name az203mingyust

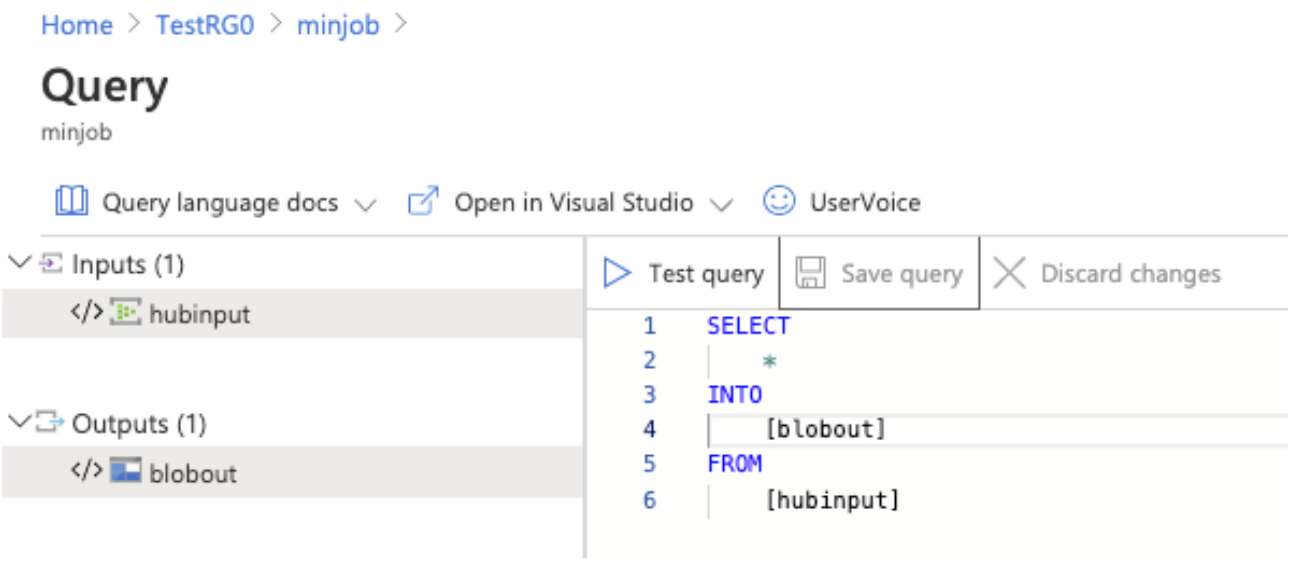
 # Blob List 보기
$ az storage blob list --container-name az203con --output table --account-name az203mingyust

 # Blob Delete
$ az storage blob delete --container-name az203con --name abc.txt --account-name az203mingyust

EventHubs 설정






Home > TestRG0 > minjob >
Query
minjob
Query language docs
Open in Visual Studio
UserVoice
Inputs (1)
hubinput
Outputs (1)
blobout
Test query
Save query
Discard changes
1 SELECT
2 *
3 INTO
4 [blobout]
5 FROM
6 [hubinput]