UNIONE EUROPEA

Ministero dello Sviluppo Economico

Regione Puglia
Dipartimento Sviluppo Economico, Innovazione, Istruzione, Formazione e Lavoro

PUGLIA FESR·FSE 2014/2020
Il futuro alla portata di tutti

BANDO INNOLABS

"Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale"

Y95B457 Progetto

serviziLocali S.p.A.

SBSoft

UNIVERSITÀ DEL SALENTO

easyPAL

Deliverable D.C1.1 - Progetto della Piattaforma EasyPAL

Data 30/05/2019

Versione V1.0

NOME DEL DOCUMENTO

Deliverable D.C1.1 - Progetto della Piattaforma EasyPAL

INFORMAZIONI GENERALI

Nome progetto	EasyPAL “Ecosistemi e Servizi Digitali in Cloud per i Cittadini e la PA Locale”
Ambito	BANDO INNOLABS “Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale”
Riservatezza	Riservatezza ai sensi dell'art. 14 dell’atto di costituzione dell’ATS denominata “Easy PAL” registrata a Lecce in data 01/06/2018 al n. 5694/1T da Notaio Pellegrino.

RESPONSABILITA'

Funzione	Nome	Data
Redatto da	SbSoft	30/05/2019
Contributi di	Unisalento, Servizi Locali, SbSoft	
Controllato e approvato da	Servizi Locali	

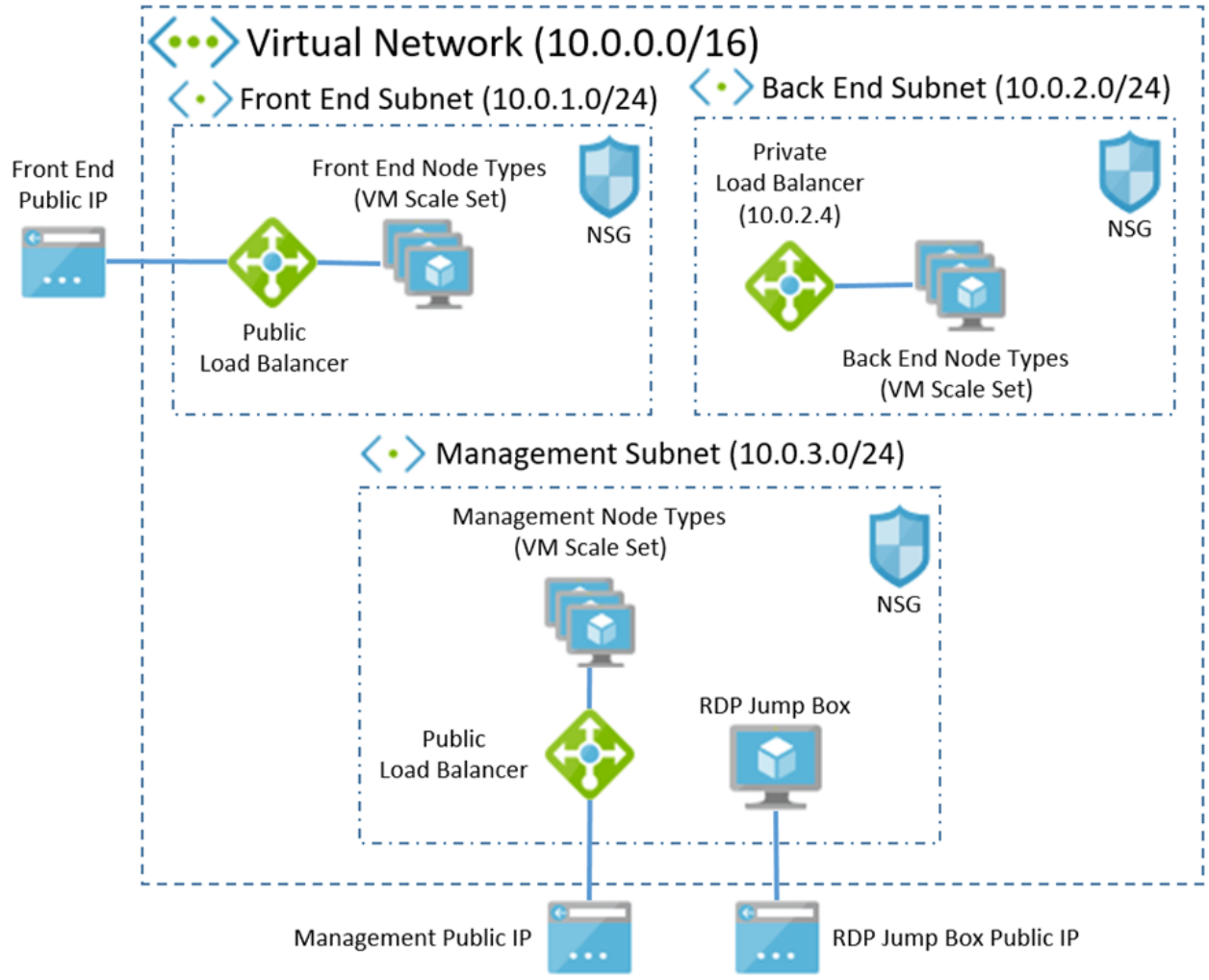
Sommario

Panoramica Generale

Il presente documento contiene informazioni utili alla progettazione dei servizi, il relativo stack di comunicazione, e altre info utili al rilascio di un cluster di Service Fabric su Azure.

Per poter adattare al meglio questa tecnologia alle esigenze del progetto EasyPal sono state adottate delle strategie di isolamento e sicurezza aggiuntive riguardo l'infrastruttura che dovrà ospitare le relative applicazioni.

Di seguito un infografica che illustra l'architettura di rete e relative misure di sicurezza progettate per ospitare gli applicativi adatta ad una fase di pre-produzione.



Come si può vedere dall'infografica sono stati isolati i domini di Back-End, Front-End, e Management, ognuno dei quali ha una propria sottorete associata ad un gruppo di sicurezza, la comunicazione dall'esterno all'applicazione è stata concessa esclusivamente al dominio di Front-End attraverso l'uso di un IP pubblico associato ad un Load Balancer esterno.

Altro punto di accesso dall'esterno è sul nodo di Management, dal quale si potrà gestire e rilasciare i micro-servizi.

È stata inoltre implementata una Jump-Box per poter accedere da remoto direttamente alle macchine virtuali del cluster in caso di Failover del cluster di Service Fabric.

Infrastruttura Azure

Dipendenze

L'infrastruttura sulla quale ospitare i micro-servizi comprende oltre a Service Fabric e al Networking illustrato in precedenza un insieme di risorse correlate.

Alcune di esse possono essere definite come dipendenze vere e proprie. Saranno le prime ad essere create e verranno inserite in gruppi di risorse separati in quanto indipendenti dalla tipologia del cluster utilizzato.

PowerShell Az Module

Per poter procedere al deploy dell'infrastruttura abbiamo bisogno del modulo PowerShell Az aggiornato.

Aprire le console PS e digitare:

```
$PSVersionTable.PSVersion
```

Assicurarsi di avere almeno la versione 5.1, altrimenti seguire la procedura:

https://docs.microsoft.com/it-it/powershell/scripting/install/installing-powershell?view=powershell-6

Se si utilizza PowerShell 5.1, è necessario installare .Net Framework 4.7.2.

Eseguire l'Upgrade del modulo Az:

https://docs.microsoft.com/en-us/powershell/azure/new-azureps-module-az?view=azps-1.5.0

Key Vault

Questa risorsa di Azure ospiterà le credenziali di accesso al client di Service Fabric tramite certificati X509. Di seguito la procedura di creazione:

- Nella cartella EasyPal / Architettura / Dipendenze editare il file KeyVault.ps1

    - Azure_Subscription : "Sottoscrizione Azure ID (Licenza)"
    - Resource_Location : "Geolocalizzazione delle risorse"
    - Cluster_Name : "Nome del cluster"
    - KV_Resource_Group : "Gruppo di risorse per le credenziali di origine"
    - KV_Name : "Nome per il registro delle credenziali su Azure"
    - KV_AdminName : " Nome amministratore key Vault"
    - IPDNSName : " Nome DNS usato per IP pubblico di front-end, nome del cluster"
    - CERT_Path : "Percorso locale per l'importazione del certificato"
    - CERT_Password : "Password del certificato"

- Eseguire lo script KeyVault.ps1 con PowerShell, verranno mostrati avvisi di conferma durante il processo.

Oltre alla risorsa lo script creerà un certificato auto-firmato, lo importerà nel registro locale e nella sezione Certificati di 'Key Vault' su Azure. Al termine del processo verrà mostrato in verde l'id della risorsa:

es: "/subscriptions/5ac0bbed-00f0-44a0-a4d7-0000fddfdg/resourceGroups/nome-GruppoDiRisorse/providers/Microsoft.KeyVault/vaults/nome-keyvault"

Prenderne nota per lo step successivo.

Application Insight

Questa risorsa permetterà il Monitoring delle applicazioni, dei contenitori, e del cluster, può essere usata per creare test di disponibilità e test di carico per valutare le performance del sistema.

Sarà possibile consultare i dati di telemetria e i log applicativi nonché le eccezioni gestite e non. Di seguito la procedura di creazione:

- Nella cartella EasyPal / Architettura / Dipendenze editare il file AppInsight.ps1
  - Azure_Subscription : "Sottoscrizione Azure ID (Licenza)"
  - AppInsightsName : "Nome Risorsa Application Insight"
  - ApplicationTagName : "Tag/Nome dell'Applicazione"
  - ResourceGroupName : "Gruppo di risorse per app insight"
  - Location : "Geolocalizzazione delle risorse"
  - OwnerApp : "Owner Account es: account.email@sbsoft.com"

- Eseguire lo script AppInsight.ps1 con PowerShell, verranno mostrati avvisi di conferma durante il processo. Al termine del processo verrà mostrato in verde la Key. es: "IKey= 3ac27b98-5818-42cf-88b8-70cf2b8ddb56"

Prenderne nota per lo step successivo.

Docker

Service Fabric, oltre ad orchestrare servizi .Net Core nativi, offre la possibilità di ospitare micro-servizi in contenitori Docker. I contenitori hanno al loro interno oltre all'eseguibile dell'applicazione un corredo di librerie del sistema operativo per cui è progettato, fornendo un ambiente isolato e indipendente dal sistema operativo dei nodi di service Fabric. Questo permette di comporre l'applicazione con servizi che sono stati scritti in linguaggi diversi. Permette inoltre di gestire la quantità di potenza di calcolo e memoria allocata al servizio indipendentemente dalle dimensioni della VM (Nodo) di service Fabric.

Per poter includere un servizio in un contenitore è necessario avere Docker sulla macchina locale di sviluppo:

https://hub.docker.com/editions/community/docker-ce-desktop-windows

Sarà necessario attivare la funzionalità Hyper-V sulla macchina locale.

Cluster Locale

Per eseguire un cluster di debug Service Fabric nel computer di sviluppo locale, installare il runtime di Service Fabric con l'SDK per Visual Studio 2017.

https://www.microsoft.com/web/handlers/webpi.ashx?command=getinstallerredirect&appid=MicrosoftAzure-ServiceFabric-CoreSDK

È necessario abilitare l'esecuzione di script inclusi nell'SDK con il comando PowerShell:

```
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser
```

Azure Service Fabric

Qui verrà descritta la procedura per la creazione di un cluster di service fabric nel cloud Azure.

Nella cartella *"EasyPal / Architettura / ServiceFabric / TestTemplate",* potete trovare un ARM template adatto ad un infrastruttura di test, per una infrastruttura di pre-produzione fare riferimento alla cartella *"EasyPal / Architettura / ServiceFabric / ProductionTemplate"*.

Il file parameters.json contiene i parametri del cluster. Editare il file, di seguito una descrizione dei campi:

- clusterLocation : Geolocalizzazione delle risorse
- clusterName : Nome del Cluster
- adminUserName : Nome Utente d'amministratore
- adminPassword : Password d'amministratore
- vmSku : Dimensioni della macchina virtuale del nodo
- certificateThumbprint : Identificativo SHA-1 Certificato (visibile in KeyVault > Certificates)
- sourceVaultValue : Id risorsa Key Vault (visibile in Proprietà di KeyVault)
- certificateUrlValue : Url del segreto (visibile in KeyVault > Certificates)
- nt0InstanceCount : Numero di nodi (VM) nel cluster
- applicationInsightsKey : Key app insight ottenuta nei passaggi precedenti
- sfReverseProxyPort : Porta del proxy inverso per la comunicazione tra i nodi del cluster

Dove non specificato lasciare valori di default.

Editare il file deploy.ps1:

- subscriptionId : Sottoscrizione Azure ID (Licenza)

- resourceGroupName : Gruppo di risorse per il cluster
- resourceGroupLocation : Geolocalizzazione delle risorse

Per procedere alla creazione lanciare lo script “deploy.ps1”, verrà chiesto un nome per la distribuzione.

Potete verificare lo stato di avanzamento tramite il portale Azure, da Gruppi di Risorse selezionate il gruppo usato per il cluster, nel menu a sinistra selezionate Distribuzioni, sarà possibile seguire in maniera dettagliata il deploy in corso o eventuali errori di distribuzione.

Verrà creato un Cluster a singolo NodeType a cui è associato un Load Balancer esterno e un gruppo di sicurezza di rete NSG.

Le app una volta distribuite nel cluster saranno raggiungibili tramite DNS associato all’IP pubblico, costruito secondo il formato:

http://clusterName.clusterLocation.cloudapp.azure.com o

https://clusterName.clusterLocation.cloudapp.azure.com

La console di gestione del cluster sarà disponibile sulla porta 19080, url:

http://clusterName.clusterLocation.cloudapp.azure.com:19080/Explorer

Nel bilanciamento di carico e nel gruppo di sicurezza le porte 80 (HTTP), 443 (HTTPS), 19080 e la 1900 sono le uniche aperte, *per testare i singoli servizi su una particolare porta assicurarsi di creare una nuova sonda e una regola nel bilanciamento di carico e nel gruppo di sicurezza, attribuendo una priorità superiore a eventuali regole di restrizione*. *E’ possibile effettuare tale operazione dal portale di Azure accedendo alla rispettiva risorsa.*

Guida ai Micro-Servizi

Creazione Servizi

Applicazioni ASPNET

Di seguito alcune linee guida sulla creazione di applicazioni web.

Un progetto d'esempio con Visual Studio è disponibile nella cartella: "EASYPAL\VisualStudioExample\EsempioMicroservizi".

Framework di destinazione

Una delle scelte iniziali particolarmente sensibile è il target di riferimento per .Net Framework, usare la versione 4.7.1 per una corretta compatibilità con ConfigurationBuilders, utile ad eseguire l'override dell'AppSetting usando le variabili d'ambiente che verranno passate nel manifesto del servizio in fase di deploy.

Alcuni link utili:

https://fluentbytes.com/category/docker/

https://docs.microsoft.com/it-it/azure/service-fabric/service-fabric-how-to-specify-environment-variables

Comunicazione tra i servizi

La comunicazione da nodo a nodo avviene internamente in una sottorete dedicata, non sarà perciò necessario configurare le diverse porte di ascolto sul gruppo di sicurezza di rete o sul Load Balancer esterno a meno che non si voglia appositamente testare la chiamata ad un singolo servizio in modo isolato.

Data la natura distribuita dei servizi all'interno del cluster non sarà possibile chiamare un endpoint statico per un determinato servizio in quanto l'istanza attiva potrebbe trovarsi su uno qualsiasi dei nodi del cluster.

Verrà usato un URL dinamico ottenuto dal servizio di Naming Discovery interno a Service Fabric presente in ogni nodo del cluster e che utilizza un proxy inverso per instradare la chiamata al nodo appropriato.

A tale scopo al momento della creazione dell'istanza del contenitore, all'interno di un nodo, Service Fabric associa una variabile d'ambiente di default: Fabric_NodeIPOrFQDN, a questa verrà associato l'IP privato interno del nodo.

Riassumendo, i passaggi per ottenere un URL da chiamare per un tipico client:

1. Ottenere l'IP privato del nodo sul quale è attiva l'istanza del client usando la variabile d'ambiente Fabric_NodeIPOrFQDN:

```
es: fqdn = Environment.GetEnvironmentVariable("Fabric_NodeIPOrFQDN");
result es: fqdn = 10.0.0.4
```

2. Usare la porta del proxy inverso (Dichiarata nell'ARM template dell'infrastruttura di Service Fabric) es: 19081
3. Usare nome applicazione e nome del servizio per creare l'endpoint base da chiamare, ad esempio un servizio client dell'applicazione chiamata Easypal che vuole effettuare una chiamata ad un servizio chiamato ApiService:

es: http://{fqdn}:{ProxyPort}/NomeApplicazione/NomeServizio

corrisponderà a: http://10.0.0.4:19081 /Easypal/ApiService

Aggiungere il supporto ad application insight

Aggiungere il supporto per Application Insight precedentemente creato:

Project -> Add -> ApplicationInsight (usare il nome di Application Insight creato in precedenza)

Seguire i suggerimenti di Visual studio e Abilitare Trace Listener.

Verrà aggiunto al progetto un file "ApplicationInsight.config", all'interno se tutto è stato eseguito correttamente troverete il tag con il Guid della Key:

```
<InstrumentationKey>000000-1b0c-4830-becb-000000000000</InstrumentationKey>
```

Aggiungere i seguenti pacchetti NuGet al progetto:

- Microsoft.ApplicationInsights.Web
- Microsoft.ApplicationInsights.ServiceFabric

In ApplicationInsight.config aggiungere il supporto a IIS nel tag:

```
<Add
Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.PerformanceCollect
orModule, Microsoft.AI.PerfCounterCollector">
    <EnableIISExpressPerformanceCounters>true</EnableIISExpressPerformanceCounters>
</Add>
```

Aggiungere il supporto per le eccezioni del browser, per applicazioni ASPNET Web MVC 4 - 5 aggiungere nella classe App_Start/FilterConfig:

```
filters.Add(new ErrorHandler.AiHandleErrorAttribute());
```

Per applicazioni ASPNET Web API 2.0 creare una nuova classe di middleware per il log delle eccezioni in App_Start/AiExceptionLogger.cs e copiare quanto segue:

```
public class AiExceptionLogger : ExceptionLogger
  {
    private TelemetryClient ai = new TelemetryClient();
    public override void Log(ExceptionLoggerContext context)
    {
      if (context != null && context.Exception != null)
      {
        ai.TrackException(context.Exception);
      }
      base.Log(context);
    }
  }
```

In App_Start/WebApiConfig.cs aggiungere nel metodo Register:

```
config.Services.Add(typeof(IExceptionLogger), new AiExceptionLogger());
```

Per ulteriori dettagli e una guida più approfondita fare riferimento alla documentazione ufficiale: https://docs.microsoft.com/it-it/azure/azure-monitor/app/asp-net-exceptions

Creazione log applicativi personalizzati

Per scrivere i log è possibile usare la classe dedicata di default ApplicationInsights.TelemetryClient:

```
using Microsoft.ApplicationInsights.TelemetryClient

var LOG = new TelemetryClient()

LOG.TrackEvent("My custom event: Questo è un evento personalizzato")
LOG.TrackTrace(”My custom trace: Questo è un log personalizzato”)
LOG.TrackException(Exception)
```

Ulteriori info su come usare il provider di log di default (ILogger) e sull'esportazione dei log in locale verranno fornite nella prossima revisione di questa documentazione.

Pubblicazione

I servizi verranno inclusi in contenitori windows con supporto pieno a IIS e con tutte le dipendenze ASP NET necessarie, non sono perciò previste operazioni particolari in fase di pubblicazione, tuttavia è consigliabile usare l'opzione di "precompilazione durante la pubblicazione" in Visual Studio 2017 in modo da evitare compilazioni a runtime di ASP NET che potrebbero influire sulle prestazioni dell'istanza del contenitore.

Contenitori Windows Docker

Creazione Immagine

Una volta pubblicati i servizi spostarsi nella relativa cartella e creare un nuovo file senza estensione chiamato Dockerfile. Si procederà ad inserire in tale file una serie di istruzioni PowerShell utili alla creazione di un Immagine Docker.

```
(Ottiene dal repository l'immagine di windows server con supporto ad AspNet e IIS)
FROM microsoft/aspnet:4.7.1-windowsservercore-ltsc2016
(Usa Powershell per customizzare immagine)
SHELL ["powershell.exe"]
(Rimuove WebSite di default in IIS)
RUN Remove-Item -Recurse C:\inetpub\wwwroot\*
RUN Remove-Website -Name 'Default Web Site'
```

```
(Crea un nuovo website usando App Pool di default e la porta del servizio)
RUN New-Website -Name 'Default Web Site' -Port 80 -PhysicalPath 'C:\inetpub\wwwroot' -ApplicationPool 'DefaultAppPool'
(Copia gli eseguibili nella cartella di IIS all'interno del contenitore)
ARG source
WORKDIR /inetpub/wwwroot
COPY . .
(Espone pubblicamente la porta usata dall'app nella rete virtuale del contenitore)
EXPOSE 80
```

Per procedere con la creazione dell'immagine in locale usare il prompt dei comandi o la console di PowerShell, spostarsi nella cartella di pubblicazione in cui è presente il dockerfile e digitare:

```
docker build -t Nome_Immagine .
```

è possibile verificarne il successo usando il comando

```
docker images
```

Per poter testare in locale il contenitore creato con tutte le limitazioni del caso, avviare un'istanza del contenitore con il comando

```
docker run –name Nome_Istanza Nome_Immagine
```

per verificarne il corretto avvio, lo stato e l'endpoint al quale raggiungere l'istanza

```
docker inspect Nome_Istanza
```

il risultato in formato json espone il parametro Networks/nat/IPAddress con il quale è possibile raggiungere l'istanza del contenitore per verificarne l'effettiva disponibilità.

Esportazione nel registro contenitori di Azure

Per poter pubblicare su Service Fabric il contenitore come servizio sarà prima necessario depositarlo nel registro contenitori di Azure o nel DockerHub. Il repository può essere creato come qualsiasi altra risorsa dal portale di Azure: portal.azure.com, una volta creato prendere nota del server di accesso. Questo indirizzo avente struttura nome_registro.azurecr.io è il tag da associare all'immagine per eseguirne l'upload. Sempre da console lanciare il comando:

```
docker tag nome_immagine nome_registro.azurecr.io/nome_immagine
```

Pubblicazione su Service Fabric

La pubblicazione dei servizi nel cluster di Service Fabric può essere svolta in diversi modi tuttavia è preferibile usare l'ambiente integrato di Visual Studio per tale operazione. A tal fine creare un'applicazione di Service Fabric in Visual Studio 2017, il nome scelto per il progetto sarà il nome dell'applicazione su Service Fabric. (non del servizio!) Nella guida alla scelta del template selezionare la voce Container, inserire il nome del servizio, il nome dell'immagine compreso di tag, username del registro contenitori, la porta dell'host di Service Fabric e quella del contenitore. (per entrambe usare la stessa usata su IIS in fase di creazione dell'immagine)

In ApplicationPackageRoot editare ApplicationManifest.xml inserendo le credenziali del registro contenitore (consultabili dal portale)

```
<RepositoryCredentials AccountName="Username" Password="Password"
PasswordEncrypted="false" />
```

Nella cartella ApplicationPackageRoot/NomeServizioPkg aprire il file ServiceManifest.xml e modificare l'endpoint aggiungendo il Protocollo, lo schema uri e la porta del listener.

```
<Endpoint Name="Nome_ServizioTypeEndpoint" UriScheme="http" Port="80" Protocol="http" />
```

Nella cartella ApplicationParameters è possibile editare i parametri dell'applicazione come il numero di istanze per ogni servizio. Il valore di default -1 indica a Service Fabric di usare tutti i nodi disponibili, il numero di istanze sarà quindi pari al numero dei nodi disponibili nel cluster.

Prima di procedere alla pubblicazione editare il PublishProfile\Cloud.xml nella sezione ClusterConnectionParameters con l'endpoint del cluster:

```
clusterName.clusterLocation.cloudapp.azure.com:19000
```

e l'identificativo SHA-1 del certificato depositato in Key Vault.

Esempio file Cloud.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<PublishProfile xmlns="http://schemas.microsoft.com/2015/05/fabrictools">
  <ClusterConnectionParameters
    ConnectionEndpoint="test1303.westeurope.cloudapp.azure.com:19000"
    X509Credential="true"
    ServerCertThumbprint="1DABE5ADEAE04555497FA5403E00324976848301"
    FindType="FindByThumbprint"
    FindValue="1DABE5ADEAE04555497FA5403E00324976848301"
    StoreLocation="CurrentUser"
```

```
    StoreName="My" />
  <ApplicationParameterFile Path="..\ApplicationParameters\Cloud.xml" />
  <CopyPackageParameters CompressPackage="true" />
  <UpgradeDeployment Mode="Monitored" Enabled="false">
    <Parameters FailureAction="Rollback" Force="True" />
  </UpgradeDeployment>
</PublishProfile>
```

Cliccando sul progetto di Service Fabric scegliere la voce Publish, nel primo menu a tendina “Target Profile” selezionate cloud, in account selezionate add Account se non presente e loggarsi con le credenziali di Azure.

Se Il file PublishProfile\Cloud.xml è stato editato correttamente la voce Connection Endpoint sarà popolata di default con l’endpoint del cluster.

In ApplicationParametersFile selezionare Cloud.xml, quindi procedere alla pubblicazione.

Se tutti i passaggi sono stati effettuati correttamente nella console di output di VS2017 verrà visualizzato un avviso di pubblicazione avvenuta con successo entro pochi secondi, tuttavia l’operazione di deploy vera e propria nel cluster verrà gestita in background da Azure e comporterà tempi più lunghi.

Si potrà comunque valutare lo stato del deploy dal portale di management del cluster http://clusterName.clusterLocation.cloudapp.azure.com:19080/Explorer

Per monitorare il cluster, i contenitori, e l’applicazione in esecuzione, o eventuali errori ed eccezioni, entrare in Azure Application Insight e tramite la funzione “cerca“ sarà possibile accedere a tutti i log e i dati di telemetria del sistema.