

Cloudy Day Lab

## Erste Schritte mit Azure Service Fabric

# 1 Einrichtung der Entwicklungsumgebung

Voraussetzungen:

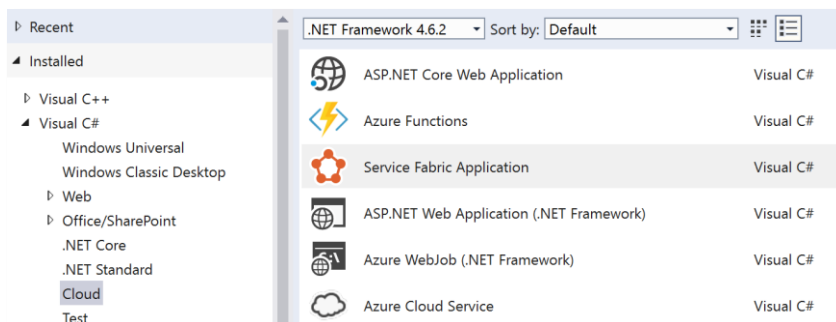
- Visual Studio 2017
- .NET Core 2.0 SDK

Weiter wie in <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-get-started> beschrieben. Die Einrichtung des lokalen Clusters (als 1-Node-Cluster) ist optional, sie erleichtert aber Entwicklung und Debugging.

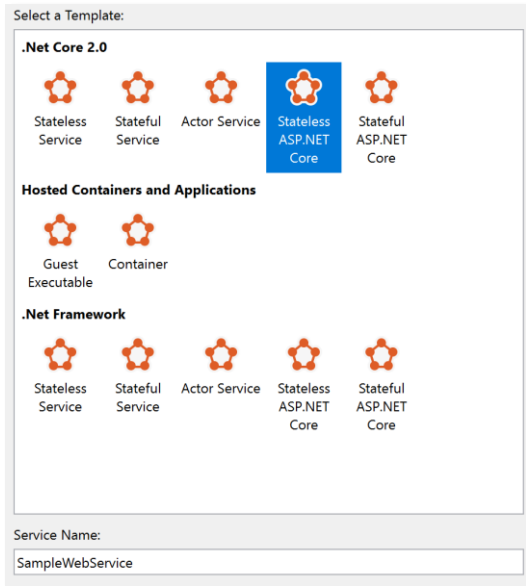
Das verwendete Beispielprojekt kann unter <https://github.com/mc5eamus/SampleServiceFabricApp> abgerufen werden.

# 2 Erstellung des Projekts

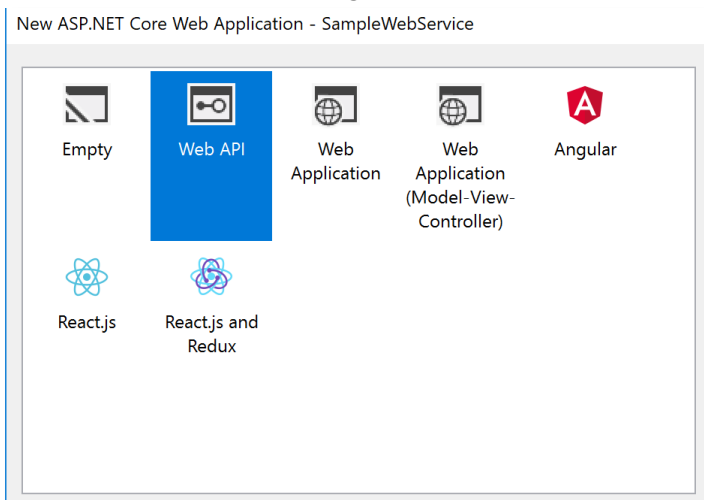
Neues Projekt vom Typ Service Fabric Application mit der Bezeichnung *SampleServiceFabricApp* erstellen.



Als Vorlage für den ersten Service «Stateless ASP.NET Core» auswählen. *SampleWebService* als Namen festlegen.



Die Art des ASP.NET Konfiguration ist Web API.



In der Solution sind nun 2 Projekte vorhanden.

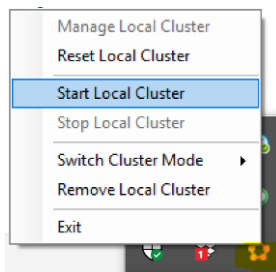
Das Projekt *SampleServiceFabricApp* beinhaltet die Anwendungsconfiguration sowie Deployment-Profil und Tools. Sofern von der Publikation der Anwendung gesprochen wird, bezieht sich die Aktion auf dieses Projekt.

Das Projekt *SampleWebService* stellt den ersten Service der Anwendung dar. Der Service ist stateless und verfügt über einen externen Endpunkt (definiert in `PackageRoot\ServiceManifest.xml`). Der angegebene Port soll hier auf `8080` (sofern nicht bereits

belegt) geändert werden. Im Übrigen ist der Service als eine ASP.NET Anwendung über Kestrel bereitgestellt und kann über http angesprochen werden. Der Beispielcontroller (Controllers\ValuesController.cs) bietet eine für die ersten Tests ausreichende Basisfunktionalität.

Das Projekt kann in der Form bereits auf einem Service Fabric Cluster publiziert und ausgeführt werden.

### 3 Publikation des Projekts auf dem lokalen Cluster (optional)



Über Service Fabric Local Cluster Manager sicherstellen, dass der lokale Cluster ausgeführt wird.

Als Zielprofil für die Publikation des App-Projekts Local.1Node.xml festlegen. Nach der erfolgreichen Publikation über *Manage Local Cluster* die Web-Oberfläche des Managers aufrufen und sicherstellen, dass die Anwendung korrekt publiziert und ausgeführt wird ist.

Der Aufruf des Value-Controllers unter <http://localhost:8080/api/values> soll den Wert ["value1", "value2"] liefern.

Die lokal publizierte Service Fabric Anwendung erlaubt das Debugging.

### 4 Publikation des Projekts auf dem Party Cluster

Microsoft bietet die Möglichkeit, unter <http://try.servicefabric.azure.com> einen 3-Node-Cluster zu erstellen und für eine Stunde zu nutzen. Die Anmeldung erfolgt mit einem Github oder einem Facebook-Account.

Nach der Anmeldung werden folgende Informationen zu dem Cluster angezeigt:

Your secure Service Fabric cluster is ready!

**ReadMe:** How to connect to a secure Party cluster?

**Certificate required to connect:**

PFX

**Service Fabric Explorer**

<https://win2433lmpj9kb.westus.cloudapp.azure.com:19080/Explorer/index.html>

**Connection endpoint**

win2433lmpj9kb.westus.cloudapp.azure.com:19000

**Expires on:**

April 10 at 23:15:38 UTC

**Time remaining**

0 hours, 59 minutes, and 59 seconds

**Available ports**


80, 8081, 8080, 20000, 20001, 20002, 20003, 20004, 20005

Die Authentifizierung für den Zugriff erfolgt über ein Client-Zertifikat, welches direkt heruntergeladen und in den persönlichen Store importiert werden soll. Das dazugehörige Passwort kann aus dem unter Readme verfügbaren Dokument bezogen werden.

Der Aufruf des Cluster Managers im Browser erfolgt mit Angabe des Client-Zertifikats. Achtung: Da der Manager nicht über ein gültiges HTTPS-Serverzertifikat verfügt, soll, je nach Browser, die Verbindung trotz Warnung, explizit zugelassen werden.

Für die Publikation der Anwendung wird das Profil Cloud.xml verwendet.

Target profile:  
PublishProfiles\Cloud.xml

 ISOLUTIONS AG  
maxim.gross@isolutions.ch

Connection Endpoint:  
win243d8flismk.westus.cloudapp.azure.com:19000

Advanced Connection Parameters

Name	Value
X509Credential	true
FindType	FindByThumbprint
FindValue	03C2E087BCDEF43BB81DAA265DF70594ED364A74
ServerCertThumbprint	03C2E087BCDEF43BB81DAA265DF70594ED364A74
StoreLocation	CurrentUser
StoreName	My
Add a parameter	

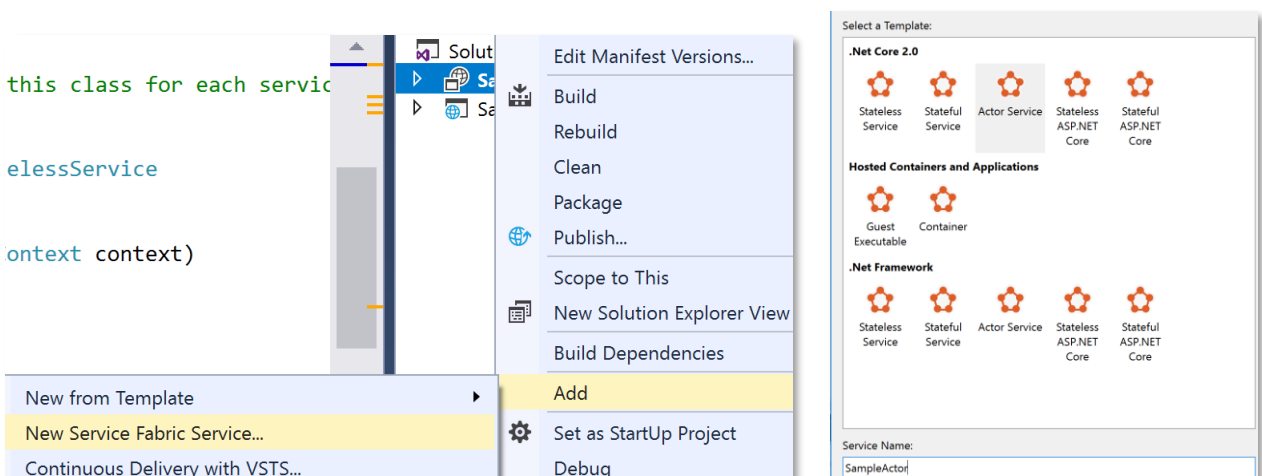
Als Endpoint soll die in der Cluster-Information angezeigte Adresse angegeben werden. Unter Advanced Connection Parameters wird in der Regel das entsprechende Zertifikat als ausgewählt angezeigt.

Nach der Publikation soll der Aufruf von `http://[cluster-address]:8080/api/values` zum gleichen Ergebnis wie bei der lokalen Publikation führen.

## 5 Erweiterung des Projekts um einen Actor Service

Die einfachste Form, in Service Fabric die Ausführung einer Aufgabe an einen Service zu delegieren, ist die Verwendung von Actor Services.

Wir fügen unserem Projekt nun einen Actor hinzu. Die Option *New Service Fabric Service* ist unter dem Add-Menü des Application-Projekts vorhanden.



Der neue Actor Service soll den Namen *SampleActor* erhalten.

Mit dem Hinzufügen vom Actor Service wurden in der Solution 2 neue Projekte erstellt:

- *SampleActor*, welches die eigentliche Implementierung des Dienstes enthält
- *SampleActor.Interfaces* mit der Schnittstellendefinition für den Dienst, welche als Referenz in dem *SampleWebService* verwendet werden soll.

## 6 Implementierung und Nutzung vom Actor Service

Die generelle Richtlinie für die Implementierung gliedert sich in folgende Schritte:

### 6.1 Design des Interfaces

*SampleActor.Interfaces\ISampleActor.cs* wird an die Aufgaben des Dienstes angepasst. In dem Beispielprojekt soll eine Umkehrung der Textzeile berechnet werden.

### 6.2 Implementierung des Interfaces

In *SampleActor\SampleActor.cs* soll die Schnittstelle implementiert werden. Das Beispielprojekt bietet einen Ausgangspunkt für die Umsetzung, u.a. wird die Nutzung des Actor Service States mit Hilfe eines Counters demonstriert.

### 6.3 Aufruf des Actors

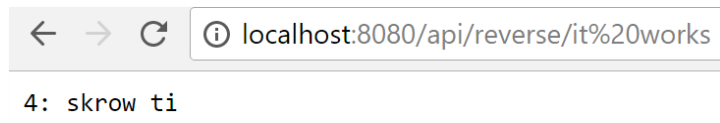
Für die Nutzung des Actors wurde im Beispielprojekt ein neuer Controller implementiert (*SampleWebService\Controllers\ReverseController.cs*).

Der Aufruf des Actors erfolgt über einen dynamischen ActorProxy, welcher u.a. über den Service-Namen *fabric:/SampleServiceFabricApp/SampleActorService* aufgelöst werden kann.

```
var actorId = new ActorId("...");
var actor = ActorProxy.Create<ISampleActor>(
    actorId,
    new Uri("fabric:/SampleServiceFabricApp/SampleActorService"));
var cancellationToken = new System.Threading.CancellationToken();
var result = await actor.Reverse(val, cancellationToken);
```

### 6.4 Publikation der Lösung und Aufruf des Dienstes

Der im Beispielprojekt realisierte Dienst lässt sich unter der Url `http://[cluster-address]:8080/api/reverse/[text]` aufrufen.



## 7 Nächste Schritte

Das erstellte Projekt bietet eine Grundlage für weitere Experimente. So kann man z.B. eine feste ID des Actors durch eine zufällige ID ersetzen (s. Beispielprojekt) und beim Aufruf auf das Verhalten des Zustandswerts achten.