

# Sponsors and Partners

## Strategic Sponsors

The logo for adform, featuring the word "adform" in a lowercase, sans-serif font. The letter "o" is stylized with a blue and green circular graphic element.

## Gold Sponsors



## Silver Sponsors





(VERY FAST INTRO – WHAT IS  
SERVICE FABRIC)

# Service Fabric advertisement ☺

## Scale

- Scale from small departmental apps to internet scale
- Utilize hardware efficiently at scale

## SINGLE CODE BASE!

## High Availability/Reliability

- Quickly react to failures of application code, machines, and networks
- Do it without complexity/cost (Don't buy a SAN)
- Do it in the cloud (without controlling the hardware)

## Performance and Latency

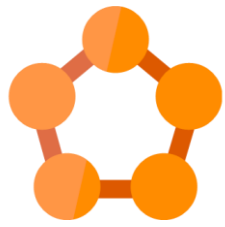
- **Millisecond response times for reads**
- **High throughput for writes**

## Management

- Perform online upgrades and patches
- Monitor and maintain health information of your distributed application
- Get real-time information about topology etc.

# Scalable AND Fast

(not OR – as usual)



# Azure Service Fabric



Lifecycle  
Mgmt

Independent  
Scaling

Independent  
Updates

Always On  
Availability

Resource  
Efficient

Stateless/  
Stateful



Public Cloud  
(Azure)



On Premises  
(Own machines)



Other Clouds  
(Amazon AWS, etc.)



# Service Fabric 101

**Cluster:** OS Instances (PHY/VIRT/DOCKER). Pool of Resources. Nodes. Environment Independent Abstraction Layer

Fault Domain & Update Domain

Active-Passive ("Primary" and "Secondary") with fast failover.

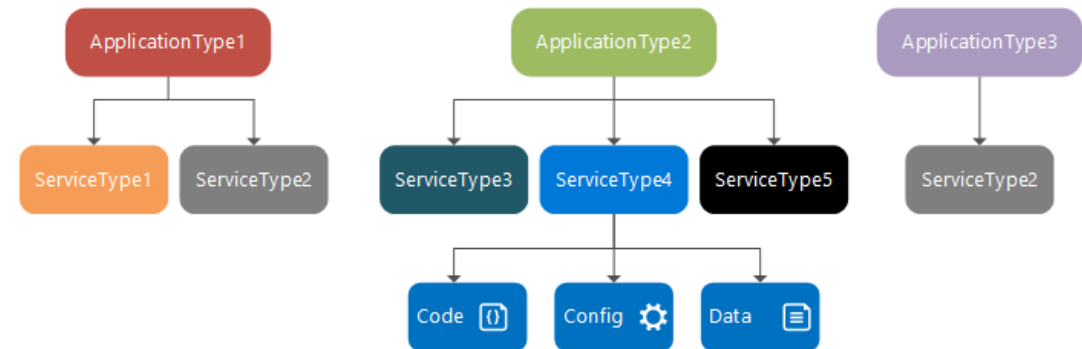
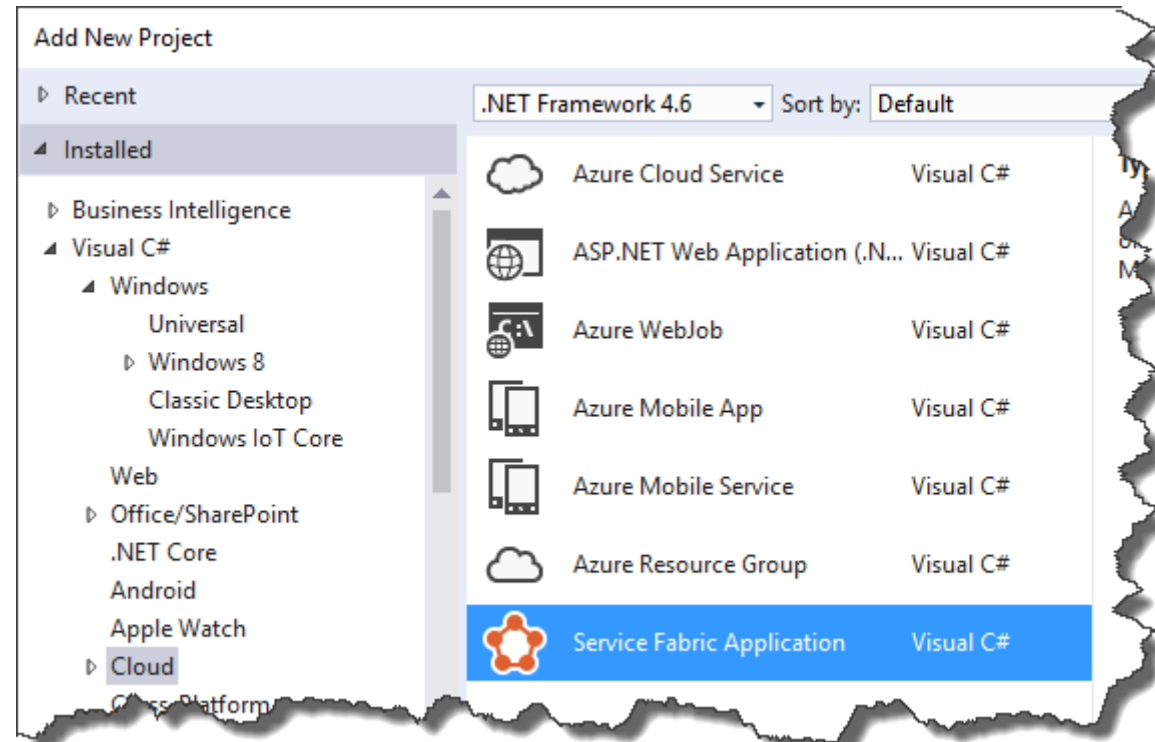
**Service:** code and state that Fabric manages.

Service **Instance** can communicate with another using any protocol

Service can have **partitions** (scale out unit)

**Replicas** in statefull services (usually one primary and two secondary, quorum)

Application is a collection of services.  
Upgradeable / downgradeable



# Service Fabric API

## Guest Executables

- Bring any exe
- Any language
- Any programming model
- Packaged as Application
- Gets versioning, upgrade, monitoring, health, etc.
- Historical: similar to the old „Cloud Services“
- Containers (as a way of deploying „guests“)

## Stateless Services

- No state or it can be retrieved from an external store
- There can be N instances, e.g. web frontends, protocol gateways, utility services
- Technically, can be partitioned – but there is no reason to do that 😊

## Reliable Services

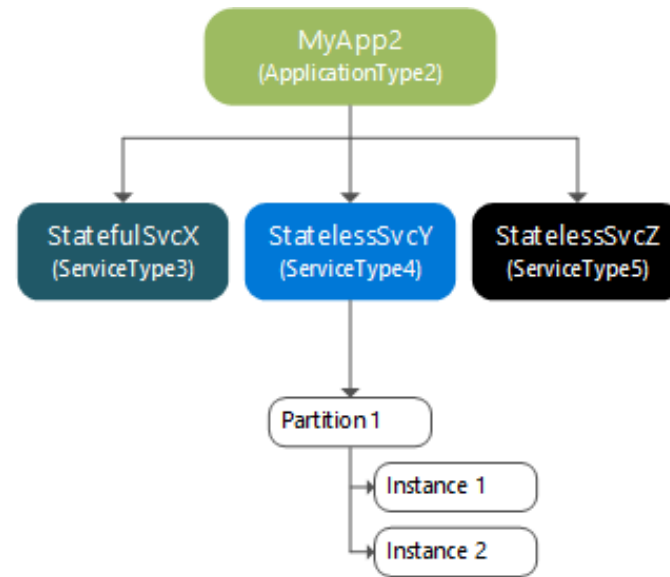
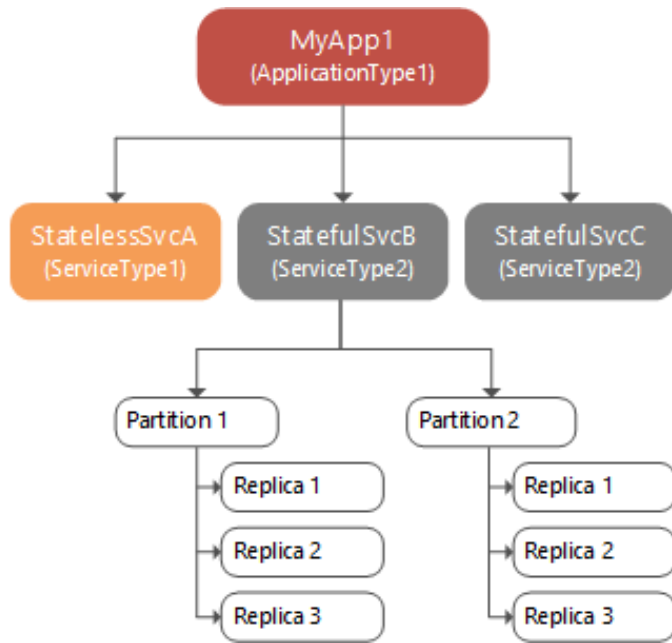
- Statefull services
- Concurrent, granular state changes
- Use of the Reliable Collections
- Transactions across collections
- Full platform integration
- N consistent copies (QUORUM) achieved through replication and **local persistence**
- Great for manual control!

## Reliable Actors

- Stateless & statefull actor objects
- Simplified programming model
- Single Threaded model
- Great for scaled out compute and state
- Many ready to use automations!
- Do not fight with “defaults”
  - If necessary – use reliable services!



# Partitions, replicas etc – logical view



Application Name	fabric/TK_2016MainSFFunctions
<b>Partition Description</b>	
Partition Scheme	Int64Range
Count	3
Low Key	1
High Key	3
Target Replica Set Size	3
Min Replica Set Size	3
Replica Restart Wait Duration Seconds	1800
Quorum Loss Wait Duration Seconds	3214202341
Stand By Replica Keep Duration Seconds	604800
Move Cost	Zero

# Physical: Fault Domain and Update Domain

## Statefull A

- Primary: 0, Secondary: 1,2

## Statefull B

- Partition 0
  - Primary: 1, Secondary: 2,3
- Partition 1
  - Primary: 2, Secondary: 3,4

## Statefull C

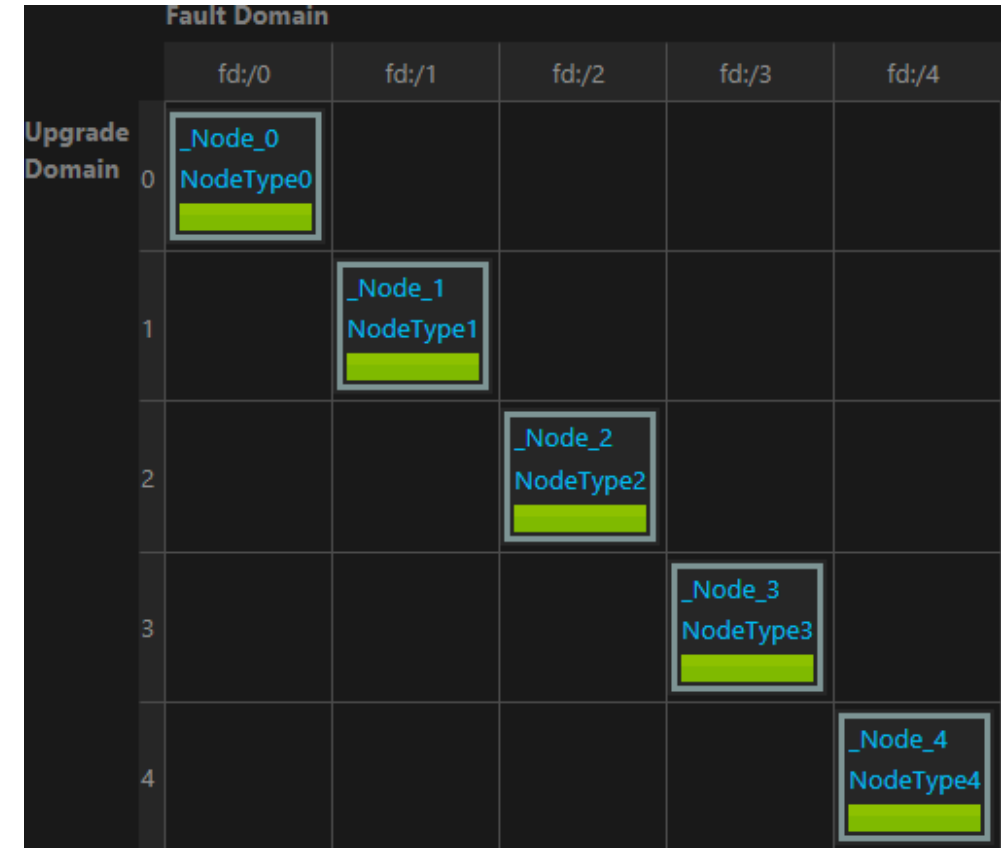
- Primary 0 (for some reasons!), Secondary 4,1

## Stateless D

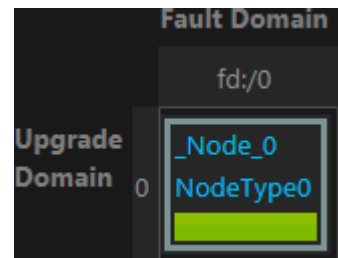
- Node 0

## Web E

- Load Balancer, Node 0,1,2,3,4



We can also have a  
"single node"  
Service Fabric!



# Demo

First, slightly more complicated than Hello World  
Service Fabric App

# Fast Demo

Guest services

# Great, but - how to design Application

## Architecture

- Any helpful patterns?
- How to design microservices architecture?



*Other talks:*

[Hands-on Experience: What it Means to Design a Domain Model](#)

[CQRS for... everyone!](#)

[DDD: Where's the Value and What's in It for Me?](#)

*(And materials at the end of this PPTX 😊)*

## Technology

- **When to use Reliable Services and when – Actors**
- **How to combine Service Fabric and App Services**
- **How to use reliable collections**
- **Technical: garbage collector, scale, upgrade, ...**

# Patterns and Practices

# Cloud Design Patterns

### Command and Query Responsibility Segregation (CQRS)

Segregate operations that read data from operations that update data by using separate interfaces. This pattern can maximize performance, scalability, and security, support evolution of the system over time through feature flexibility, and prevent update commands from causing merge conflicts at the database level.

For more info, see [Command and Query Responsibility Segregation \(CQRS\) Pattern](#)

### Compute Resource Consolidation

Consolidate multiple tasks or operations into a single computational unit. This pattern can increase compute resource utilization, and reduce the costs and management overhead associated with performing compute processing at cloud-hosted applications.

For more info, see [Compute Resource Consolidation Pattern](#)

### Competing Consumers

Enable multiple consumers to process messages received on the same messaging channel. This pattern enables a system to process multiple business concerns to optimize throughput, to improve scalability and availability, and to balance the workload.

For more info, see [Competing Consumers Pattern](#)

### Compensating Transaction

Undo the work performed by a series of steps, which together define an eventually consistent operation, if one or more of the operations fail. Operations that follow the eventual consistency model are commonly found in cloud-hosted applications that implement complex business processes and workflows.

For more info, see [Compensating Transaction Pattern](#)

### Circuit Breaker

Handle faults that may take a variable amount of time to rectify when connecting to a remote service or resource. This pattern can improve the stability and resiliency of an application.

For more info, see [Circuit Breaker Pattern](#)

### Cache-aside

Load data from a cache from a data store. This pattern can improve performance and also helps to maintain consistency between data fast in the cache and the data in the underlying data store.

For more info, see [Cache-aside Pattern](#)

### Index Table

Create indexes over the fields in data stores that are frequently referenced by query criteria. This pattern can improve query performance by allowing applications to more quickly locate the data to retrieve from a data store.

For more info, see [Index Table Pattern](#)

### Health Endpoint Monitoring

Implement functional checks within an application that external tools can access through exposed endpoints at regular intervals. This pattern can help to verify that applications and services are performing correctly.

For more info, see [Health Endpoint Monitoring Pattern](#)

### Gatekeeper

Protect applications and services by using a dedicated host instance that acts as a buffer between clients and the application or service, validates and formats requests, and passes requests and data between them. This pattern can provide an additional layer of security, and lock the data surface of the system.

For more info, see [Gatekeeper Pattern](#)

### Federated Identity

Delegate authentication to an external identity provider. This pattern can simplify development, increase the requirement for user administration, and improve the user experience of the application.

For more info, see [Federated Identity Pattern](#)

### External Configuration Store

Move configuration information out of the application deployment package to a centralized location. This pattern can provide opportunities for easier management and control of configuration data, and for sharing configuration data across applications and application instances.

For more info, see [External Configuration Store Pattern](#)

### Event Sourcing

Use an append-only store to record address ledger data, rather than the current state and use the store to materialize the domain object. Its creator domains this can avoid sync-ing the data model and the business domain, improve performance, scalability, and improve events provide consistency, and provide audit history to enable compensating actions.

For more info, see [Event Sourcing Pattern](#)

### Retry

Enable an application to handle anticipated, temporary failures when it attempts to connect to a service or resource, measured by temporarily retrying an operation that has previously failed in the application that the cause of the failure is transient. This pattern can improve the visibility of the application.

For more info, see [Retry Pattern](#)

### Queue-Based Load Leveling

Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads that may otherwise cause the service to fail or the task to timeout. This pattern can help to increase the output of tasks in demand on availability and responsiveness for both the task and the service.

For more info, see [Queue-Based Load Leveling Pattern](#)

### Priority Queue

Prioritize requests sent to services so that requests with a higher priority are received and processed more rapidly than those of a lower priority. This pattern is common in applications that offer different service-level guarantees to individual clients.

For more info, see [Priority Queue Pattern](#)

### Pipes and Filters

Decompose a task that performs complex processing into a series of discrete elements that can be reused. This pattern can improve performance, scalability, and resiliency by allowing task elements that perform the processing to be deployed and scaled independently.

For more info, see [Pipes and Filters Pattern](#)

### Materialized View

Generate pre-populated views over the data in one or more data stores when the data is formatted in a way that does not favor the required query operations. This pattern can help to request efficient querying and data retrieval, and improve performance, but conflict with each other, cause contention for shared resources, or inadvertently interfere with the work that other task instances are performing.

For more info, see [Materialized View Pattern](#)

### Leader Election

Coordinate the actions performed by a collection of collaborating task instances in a distributed application by selecting one instance as the leader that assumes responsibility for managing the other instances. This pattern can help to ensure that task instances do not conflict with each other, cause contention for shared resources, or inadvertently interfere with the work that other task instances are performing.

For more info, see [Leader Election Pattern](#)

### Valet Key

Use a token or key that provides clients with restricted direct access to a specific resource or service in order to offload data transfer operations from the application code. This pattern is particularly useful in representations that use cloud-hosted storage systems or queues, and can increase cost and increase in availability and performance.

For more info, see [Valet Key Pattern](#)

### Throttling

Control the consumption of resources used by an instance of an application, an individual request, or an entire service. This pattern can allow the system to continue to function and meet service-level agreements, even when an increase in demand places an extreme load on resources.

For more info, see [Throttling Pattern](#)

### Static Content Hosting

Deploy static content to a cloud-based storage service that can deliver these directly to the client. This pattern can reduce the requirement for potentially expensive compute instances.

For more info, see [Static Content Hosting Pattern](#)

### Sharding

Divide a data store into a set of horizontal partitions or shards. This pattern can improve scalability when storing and accessing large volumes of data.

For more info, see [Sharding Pattern](#)

### Scheduler Agent Supervisor

Coordinate a set of actions across a distributed set of services and/or other remote resources, attempt to transparently handle faults if any of these actions fail, or undo the effects of the work performed if the system cannot proceed from a fault. This pattern can add resiliency to a distributed system by enabling it to recover, and also ensure that failures to transmit exceptions, long-lasting faults, and process failures.

For more info, see [Scheduler Agent Supervisor Pattern](#)

### Runtime Reconfiguration

Design an application so that it can be reconfigured without stopping redeployment or restarting the application. This helps to maintain availability and minimize downtime.

For more info, see [Runtime Reconfiguration Pattern](#)

Going back to technology

# Few minutes on Reliable Collections

Remember:

Immutable object



# Demo

Reliable collection

# „Shopping cart“

Shopping cart, long invoice, ERP form, State per customer, Web session

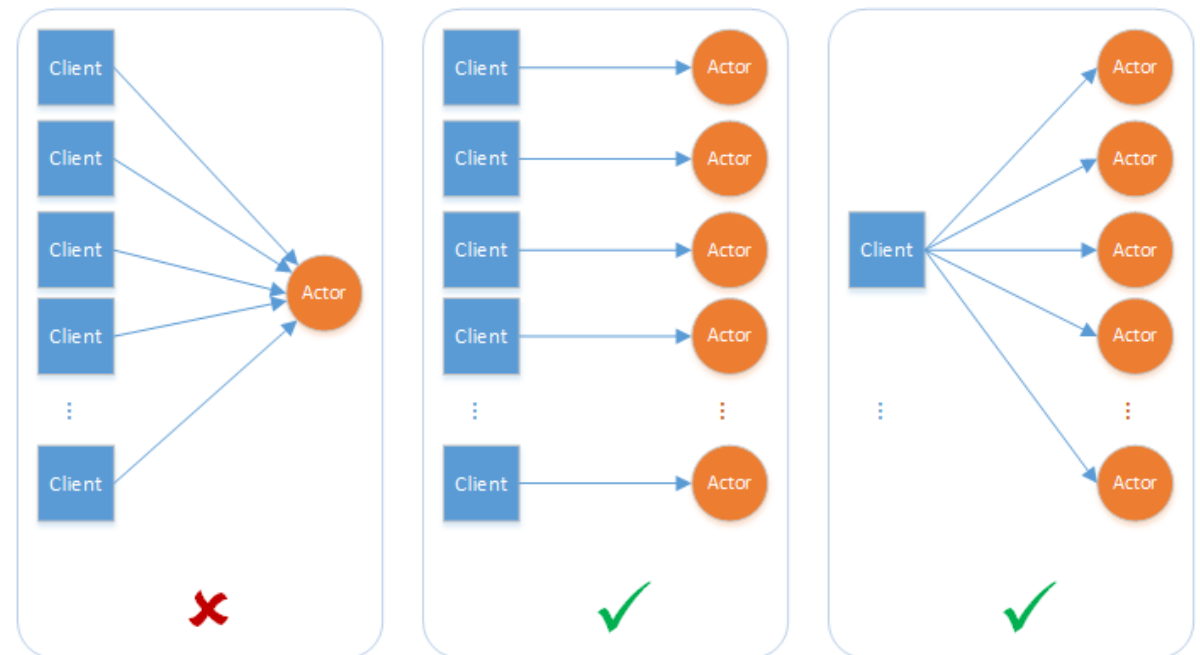
Processing jobs, Gaming session, User accounts

2 separate processes (Actors) – for “managing” cart and (optionally) for storing it in external storage

Communication between actors: direct (bigger “set of” baskets)

Actor – Stateless service (single): queue based

Customer is responsible  
for storing ID



# Demo

## Shopping Cart

Actors: Concurrency, timers,  
reminders and garbage  
collector

# Concurrency, timers and reminders

## Turn based concurrency

Internal: counter 😊

No more than one thread can be active inside the actor code at any time.

ActorServiceSettings

ActorConcurrencySettings

A -> B -> C, then C -> A reentrant, logical context

ActorGarbageCollectionSettings

(Many performance counters!)

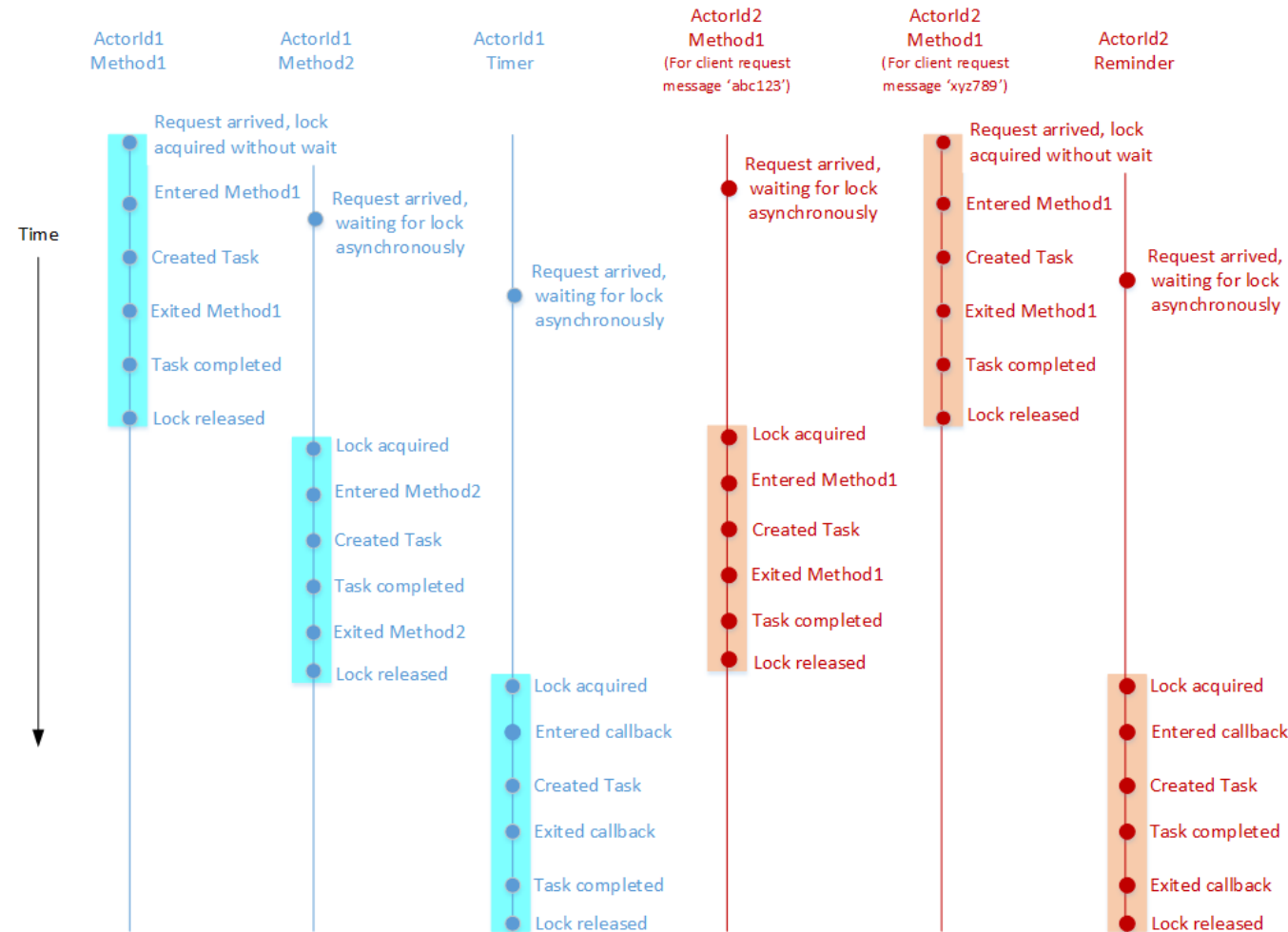
## Schedule:

Timer

Callback when Actor is alive

Reminder

Callback and can wake up (restore state) for actor



# Garbage Collector

Actor is alive when:

- Receiving a call

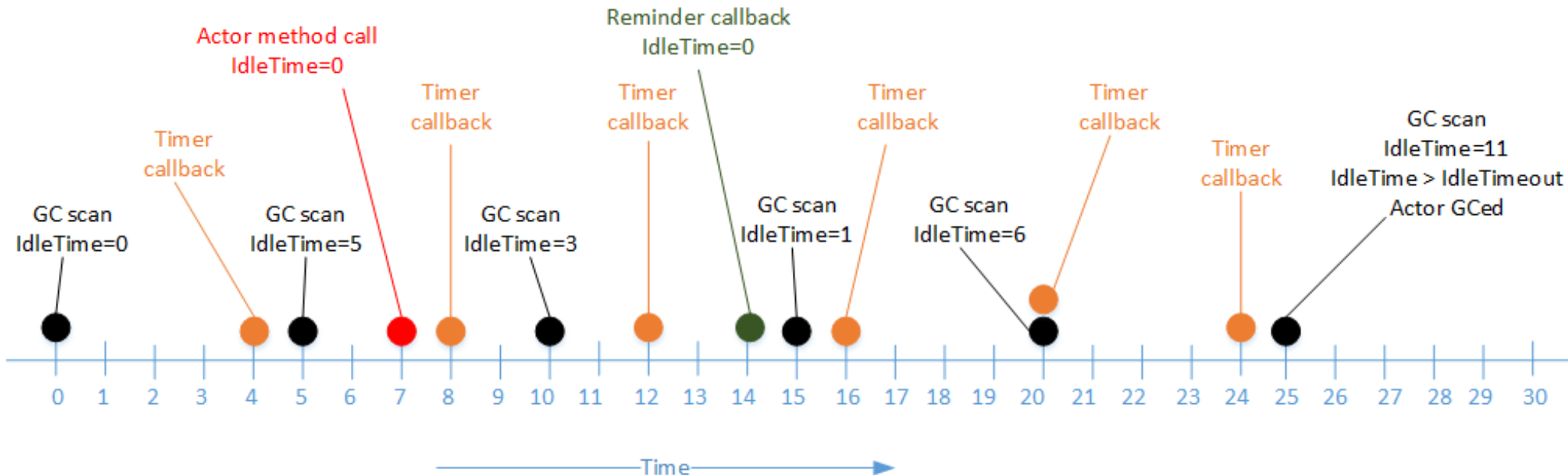
- IRemindable.ReceiveReminderAsync

ActorGarbageCollectionSetting

- IdleTimeout, ScanInterval

ScanInterval = 5, IdleTimeout = 10

```
ActorRuntime.RegisterActorAsync<Reminder1>(
    (context, actorType) => new ActorService(context, actorType,
        null, null, null,
        new ActorServiceSettings()
    {
        ActorConcurrencySettings = new ActorConcurrencySettings(
            { ReentrancyMode = ActorReentrancyMode.Disallowed },
            ActorGarbageCollectionSettings =
                //new ActorGarbageCollectionSettings(60 * 60, 60) //Default
                new ActorGarbageCollectionSettings(5 * 60, 60) // 5 min
        )
    }
)
```



# Demo (ok, code browse)

Timers and Reminders

# Timers

## 10/8/2016 10:56:05 AM - CUSTOM EVENT

OnDeactivateAsync - P1, Timers Stop  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:56:01 AM - CUSTOM EVENT

OnDeactivateAsync - P2, Timers Stop  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:55:57 AM - CUSTOM EVENT

OnDeactivateAsync - P3, Timers Stop  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:55:06 AM - CUSTOM EVENT

doWorkInTimerAsync P2, 60  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:54:06 AM - CUSTOM EVENT

doWorkInTimerAsync P2, 60  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:53:06 AM - CUSTOM EVENT

doWorkInTimerAsync P2, 60  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:53:04 AM - CUSTOM EVENT

doWorkInTimerAsync P3, 180  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:52:05 AM - CUSTOM EVENT

doWorkInTimerAsync P2, 60  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:51:04 AM - CUSTOM EVENT

doWorkInTimerAsync P2, 60  
Device type: PC Application version: 4.0

## 10/8/2016 10:50:04 AM - CUSTOM EVENT

RegisterTimerAsync - P3, 180  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:04 AM - CUSTOM EVENT

OnActivateAsync B - P3  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:04 AM - CUSTOM EVENT

Timer1 - P3  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:03 AM - CUSTOM EVENT

RegisterTimerAsync - P2, 60  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:03 AM - CUSTOM EVENT

OnActivateAsync B - P2  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:03 AM - CUSTOM EVENT

Timer1 - P2  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:02 AM - CUSTOM EVENT

RegisterTimerAsync - P1, 4800  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:02 AM - CUSTOM EVENT

OnActivateAsync B - P1  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:02 AM - CUSTOM EVENT

Timer1 - P1  
Device type: PC Application version: 4.0

---

## 10/8/2016 10:50:00 AM - CUSTOM EVENT

RunAsync  
Device type: PC Application version: 4.0



# Reminders

Dealocate (GC)

Reminder

Activation

Constructor

10/8/2016 2:51:00 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 164  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:50:42 PM - CUSTOM EVENT  
OnDeactivateAsync - PR1  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:50:17 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR3, MyReminder, 54  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:49:59 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 163  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:48:59 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 162  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:47:59 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 161  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:47:17 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR3, MyReminder, 53  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:46:58 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 160  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:45:58 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 159  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:44:58 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 158  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:44:48 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR1, MyReminder, 1  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:44:48 PM - CUSTOM EVENT  
OnActivateAsync - PR1  
Device type: PC Application version: Reminder 4.0

10/8/2016 2:44:47 PM - CUSTOM EVENT  
Reminder1 - PR1  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:08:49 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 3  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:07:49 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR3, MyReminder, 0  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:07:48 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 2  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:06:48 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 1  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:05:47 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 0  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:05:02 PM - CUSTOM EVENT  
OnDeactivateAsync - PR2  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:04:48 PM - CUSTOM EVENT  
TKRegisterReminderAsync - PR3, 180  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:04:48 PM - CUSTOM EVENT  
OnActivateAsync - PR3  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:04:47 PM - CUSTOM EVENT  
Reminder1 - PR3  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:04:47 PM - CUSTOM EVENT  
TKRegisterReminderAsync - PR2, 60  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:11:50 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 6  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:10:50 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR3, MyReminder, 1  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:10:49 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 5  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:10:42 PM - CUSTOM EVENT  
OnDeactivateAsync - PR1  
Device type: PC Application version: Reminder 4.0

10/8/2016 12:09:49 PM - CUSTOM EVENT  
ReceiveReminderAsync - PR2, MyReminder, 4  
Device type: PC Application version: Reminder 4.0

# Events: actor to client communication

Not a tool to communicate BETWEEN actors. **Best effort only – no guaranteed delivery!**

Can be used between services (statefull / stateless) and actors. Example: Actor send event to Web

Actor:

```
public interface ITKProgressEvents : IActorEvents {  
    2 references  
    void ProgressUpdated(string message);  
}  
3 references  
public interface ITKActorEventSource : IActor, IActorEventPublisher<ITKProgressEvents> {  
    2 references  
    Task<int> StartLongCalculationAsync(int param);  
}
```

```
var ev = GetEvent<ITKProgressEvents>();  
ev.ProgressUpdated($"START");  
for (int i = 0; i < param; i++) {
```

Client (WebApi in this case):

```
ITKActorEventSource actor = ActorProxy.Create<ITKActorEventSource>  
    (new ActorId(id), "fabric:/TK_EventsAndService");  
await actor.SubscribeAsync(new TKProgressEvents());  
await Task + ();
```

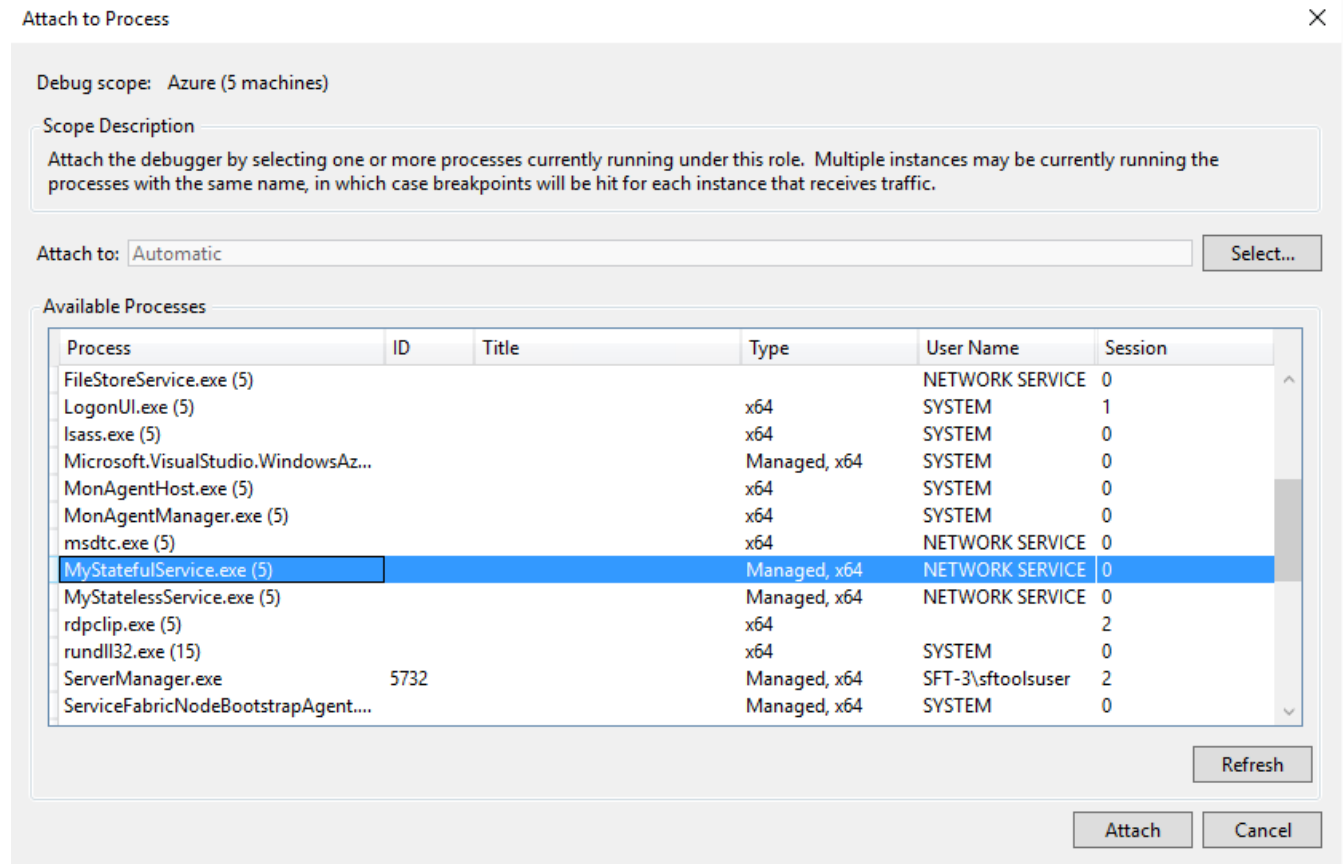
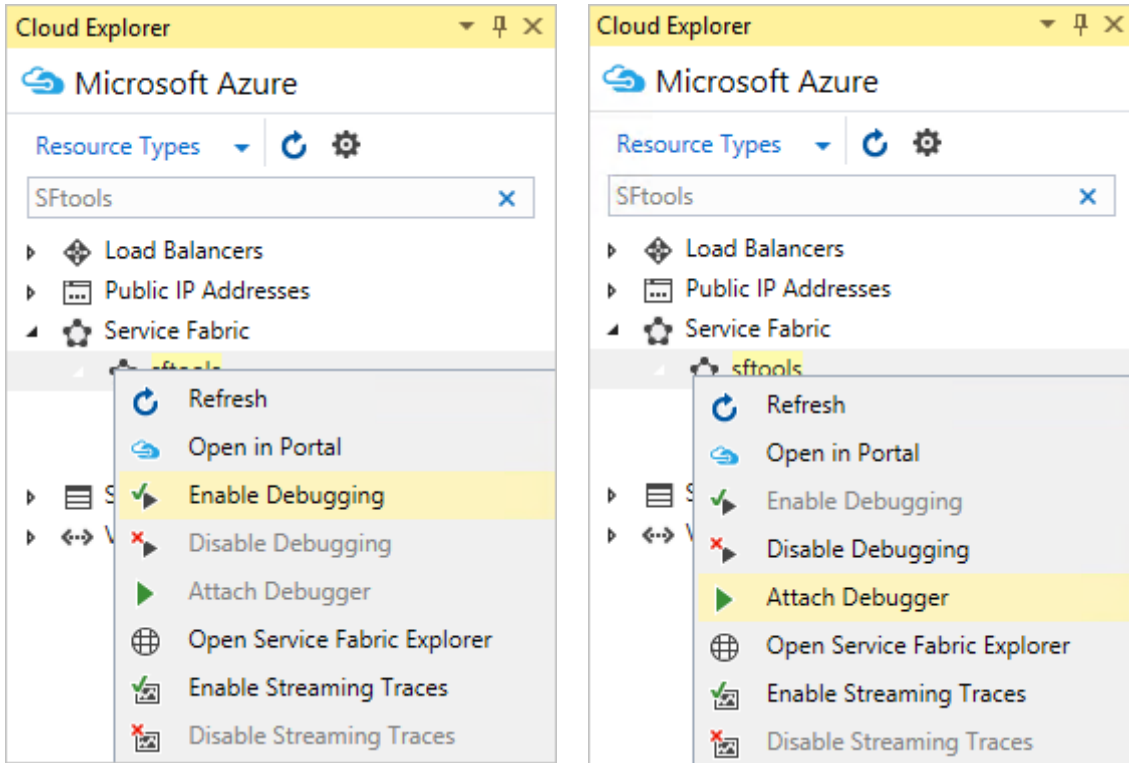
```
internal class TKProgressEvents : ITKProgressEvents {  
    3 references  
    public void ProgressUpdated(string message) {  
        ServiceEventSource.Current.Message($"ProgressUpd");  
    }  
}
```

# Demo

„Actor to client“

Debugging + Monitoring +  
Diagnostics + Health ...

# Debugging – also remotely



Better use bigger machines! (D2, D3 etc)

# Diagnostics and Troubleshooting

## Detailed System Optics

- Repair suggestions. Examples: Slow RunAsync cancellations, RunAsync failures
- All important events logged. Examples: App creation, deploy and upgrade records. All Actor method calls.

## Custom Application Tracing

- ETW == Fast Industry Standard Logging Technology
- Works across environments. Same tracing code runs on devbox and also on production clusters on Azure.
- Easy to add and system appends all the needed metadata such as node, app, service, and partition.
- Application Insight – why not
- Custom service health

## Choice of Tools

- Visual Studio Diagnostics Events Viewer
- Windows Event Viewer
- Windows Azure Diagnostics + Operational Insights
- Easy to plug in your preferred tools: Kibana, Elasticsearch and more

# Custom Service Health

System.Fabric.Health

```
HealthInformation healthInformation =  
    new HealthInformation("ServiceCode", "StateDictionary", HealthState.Error);
```

*Statefull:*

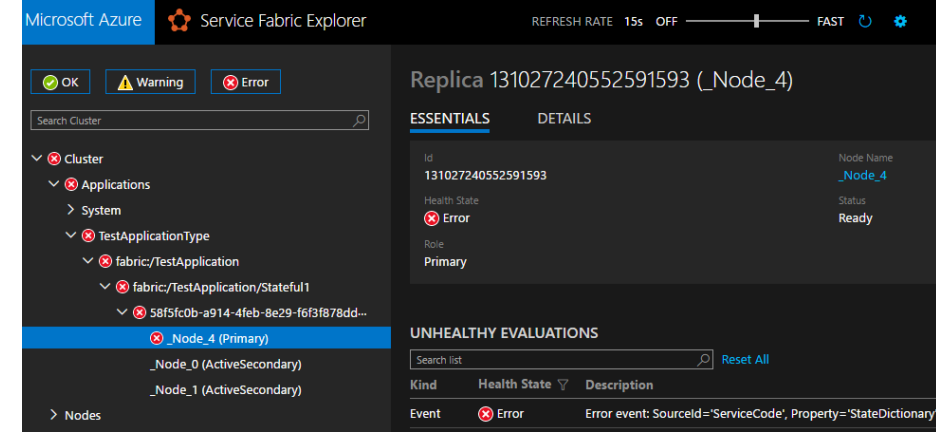
```
this.Partition.ReportReplicaHealth(healthInformation);
```

*Stateless:*

```
this.Partition.ReportInstanceHealth(healthInformation);
```

*Application, DeployedApplication, DeployedServicePackage:*

```
var activationContext = FabricRuntime.GetActivationContext();  
activationContext.ReportApplicationHealth(healthInformation);
```



# Service Placements & balancing & Metrics

```
<StatefulServiceType ServiceTypeName="MiscTelemetryType" HasPersistedState>
  <LoadMetrics>
    <!--Default: PrimaryCount, ReplicaCount,Count -->
    <!--Custom-->
    <LoadMetric Name="LARGEMEM" Weight="High" DefaultLoad="100"/>
    <!--Dynamic-->
    <LoadMetric Name="BUSINESSOPCOUNT" Weight="Low" DefaultLoad="1" Prima
```

```
<Section Name="MetricBalancingThresholds">
  <Parameter Name="MetricName1" Value="2"/>
  <Parameter Name="MetricName2" Value="3.5"/>
</Section>
```

```
<Section Name="PlacementAndLoadBalancing">
  <Parameter Name="PLBRefreshGap" Value="0.1" />
  <Parameter Name="MinPlacementInterval" Value="1.0" />
  <Parameter Name="MinConstraintCheckInterval" Value="1.0" />
  <Parameter Name="MinLoadBalancingInterval" Value="5.0" />
</Section>
```

```
<StatelessServiceType ServiceTypeName="PublicGatewayType">
  <PlacementConstraints>NodeType == NodeType1 || NodeType == NodeType2</Pl
</StatelessServiceType>
```

```
<Section Name="MetricActivityThresholds">
  <Parameter Name="Memory" Value="1536"/>
</Section>
```

See:

[Here](#), [here](#), [here](#), [here](#) and [here](#)

(Obligatory lecture!)



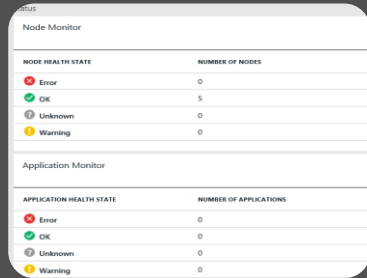


# Demo

ETW, Monitoring, custom metrics, health etc.

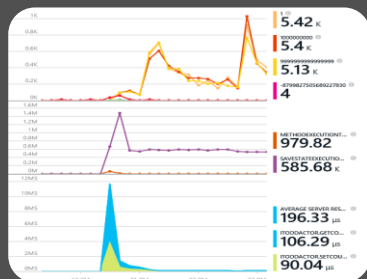
# Monitoring your Services

Visibility into how your services are doing when running in production



## Health status monitoring

- Built-in health status for cluster and services
- Flexible and extensible health store for custom app health reporting
- Allows continuous monitoring for real-time alerting on problems in production



## Performance and stress response

- Rich built-in metrics for Actors and Services programming models
- Easy to add custom application performance metrics

[Show dashboard in OMS](#)

# Demo

Cluster info

Application updates,  
Configuration, Scale, Backup

# App Lifecycle

```
Connect-ServiceFabricCluster localhost:19000
Copy-ServiceFabricApplicationPackage <folder> `
  -ImageStoreConnectionString
file:C:\SfDevCluster\Data\ImageStoreShare -
ApplicationPackagePathInImageStore TK_ETWAppInsightMetrics
```

```
Register-ServiceFabricApplicationType TK_ETWAppInsightMetrics
```

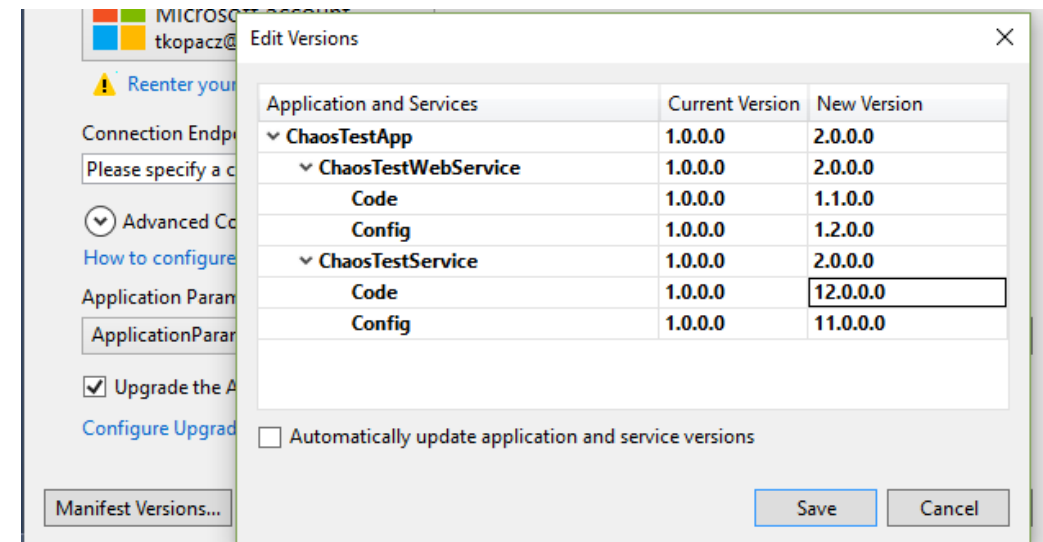
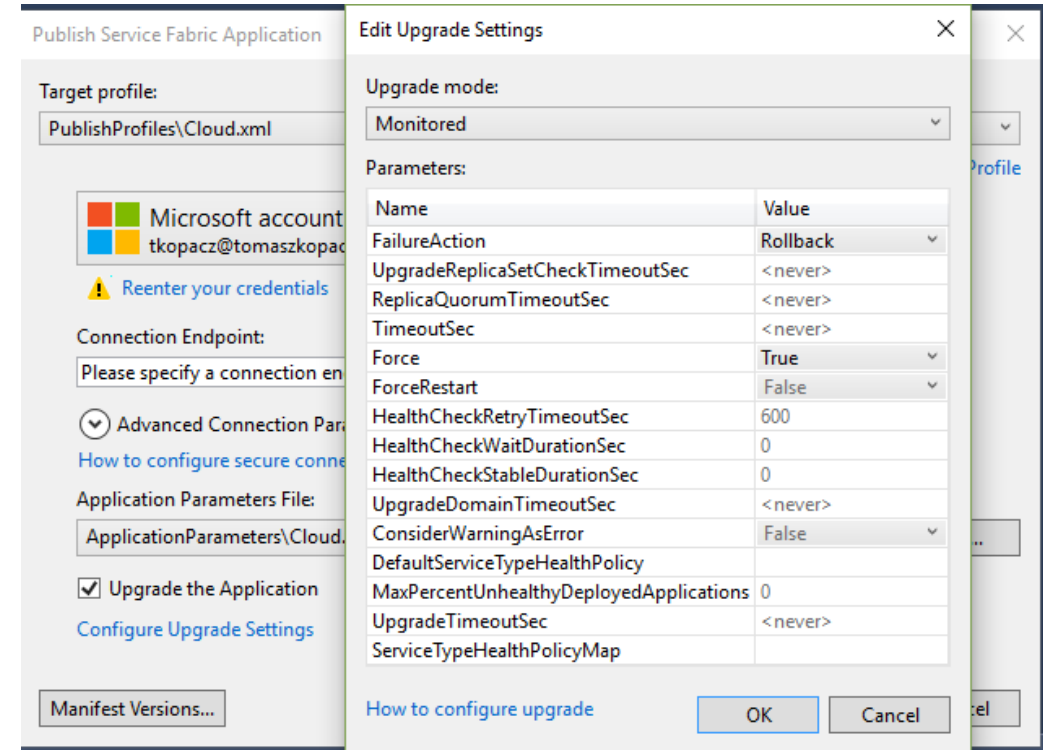
```
New-ServiceFabricApplication fabric:/TK_ETWAppInsightMetricsType
TK_ETWAppInsightMetricsType 1.0.0
```

<deploy another version>

```
Start-ServiceFabricApplicationUpgrade -ApplicationName
fabric:/TK_ETWAppInsightMetricsType -ApplicationTypeVersion 2.0.0
-HealthCheckStableDurationSec 60 -UpgradeDomainTimeoutSec 1200 -
UpgradeTimeout 3000 -FailureAction Rollback -Monitored
```

```
Remove-ServiceFabricApplication
fabric:/TK_ETWAppInsightMetricsType -Force
```

```
Unregister-ServiceFabricApplicationType
TK_ETWAppInsightMetricsType -ApplicationTypeVersion 1.0.0 -Force
```



# Configuration

This is also independent  
“upgrade” element!

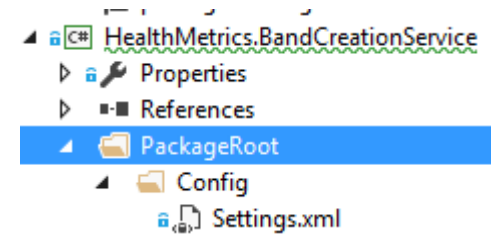
Code:

```
ConfigurationSettings configSettings =  
FabricRuntime.GetActivationContext().  
GetConfigurationPackageObject("Config").Settings;
```

```
KeyedCollection<string, ConfigurationProperty>  
serviceParameters = configSettings.Sections["HealthMetrics.BandCreationService.Settings"].Parameters
```

```
this.NumberOfCreationThreads = int.Parse(serviceParameters["NumberOfCreationThreads"].Value);
```

Can be also JSON etc – XML is build-in



```
<?xml version="1.0" encoding="utf-8" ?>  
<Settings xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/  
<Section Name="HealthMetrics.BandCreationService.Settings">  
  <Parameter Name="BandActorServiceName" Value="BandActorService" />  
  <Parameter Name="DoctorActorServiceName" Value="DoctorActorService" />  
  <Parameter Name="NumberOfCreationThreads" Value="" MustOverride="true"/>  
  <Parameter Name="MaxBandsToCreatePerServiceInstance" Value="" MustOverride="true" />  
  <Parameter Name="CountyFileName" Value="HealthMetrics.Common.csv" />  
  <Parameter Name="PeopleFileName" Value="Names.csv" />
```

# Scale (Azure)

## To remove

[Disable-ServiceFabricNode](#) & [Get-ServiceFabricNode](#) – to remove all reliable services / actors from VM  
Reduce number of VM in VMSS (usually using ARM template). Important – no less than FD / UD (5)  
Run [Remove-ServiceFabricNodeState](#) – unregister in Service Fabric Explorer (FM Service to be precise)

## To add

Increase number of VM in VMSS

Duration – for example 2 hours 49 minutes 1 second, depends on many factors!

Autoscale – the same mechanism as in other VMSS

**Remember! Reliability: one machine at the time**  
(bronze, 3 replicas)

Or more in Silver (5 replicas), Gold (7 replicas), Platinum (9 replicas); keep in mind – QUORUM!

Btw: durability: Bronze – no extra privileges, Gold – Azure infrastructure jobs can be paused for max 2h

# Backup API (for statefull services)

## To do a backup:

Invoke backup process with callback and type of backup (Full / Incremental):

```
var backupDescription = new BackupDescription(BackupOption.Full, this.BackupCallbackAsync);  
await this.BackupAsync(backupDescription, TimeSpan.FromHours(1), cancellationToken);
```

BackupCallbackAsync will receive BackupInfo with folder containing ready to save data from selected node / partition (C:\SfDevCluster\Data\\_App\\_Node\_3\TK\_2016MainSFFunctionsType\_App8\work\157ba701-f94e-4a68-85d0-e6c50e172c03\131212795996136678\trbackup)

## To restore

Override OnDataLossAsync(RestoreContext restoreCtx...)

Trigger restore

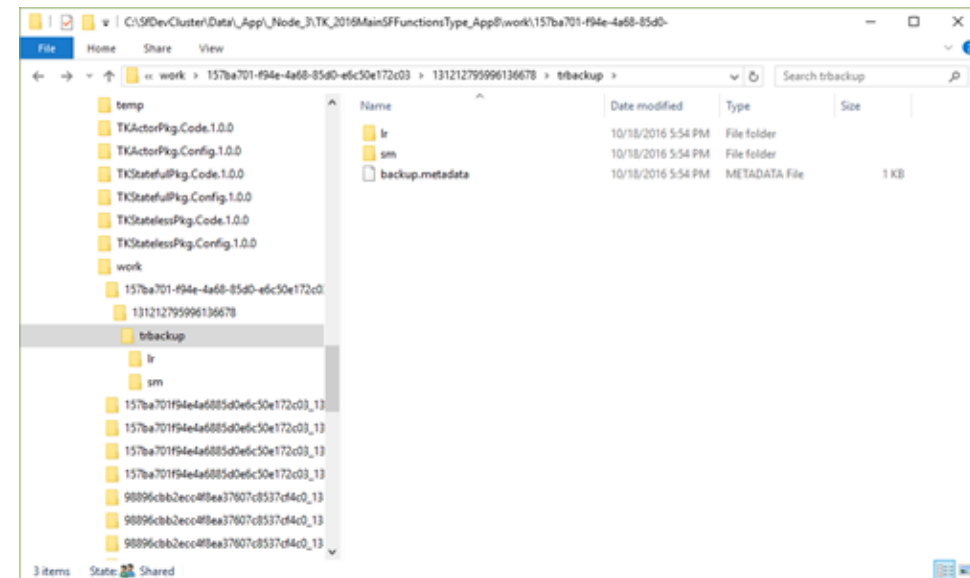
Partition data loss – automatically

Deleted or lost/corrupted service: restore service, then for each partition:

```
FabricClient.TestManagementClient.StartPartitionDataLossAsync
```

Test: Invoke-ServiceFabricPartitionDataLoss

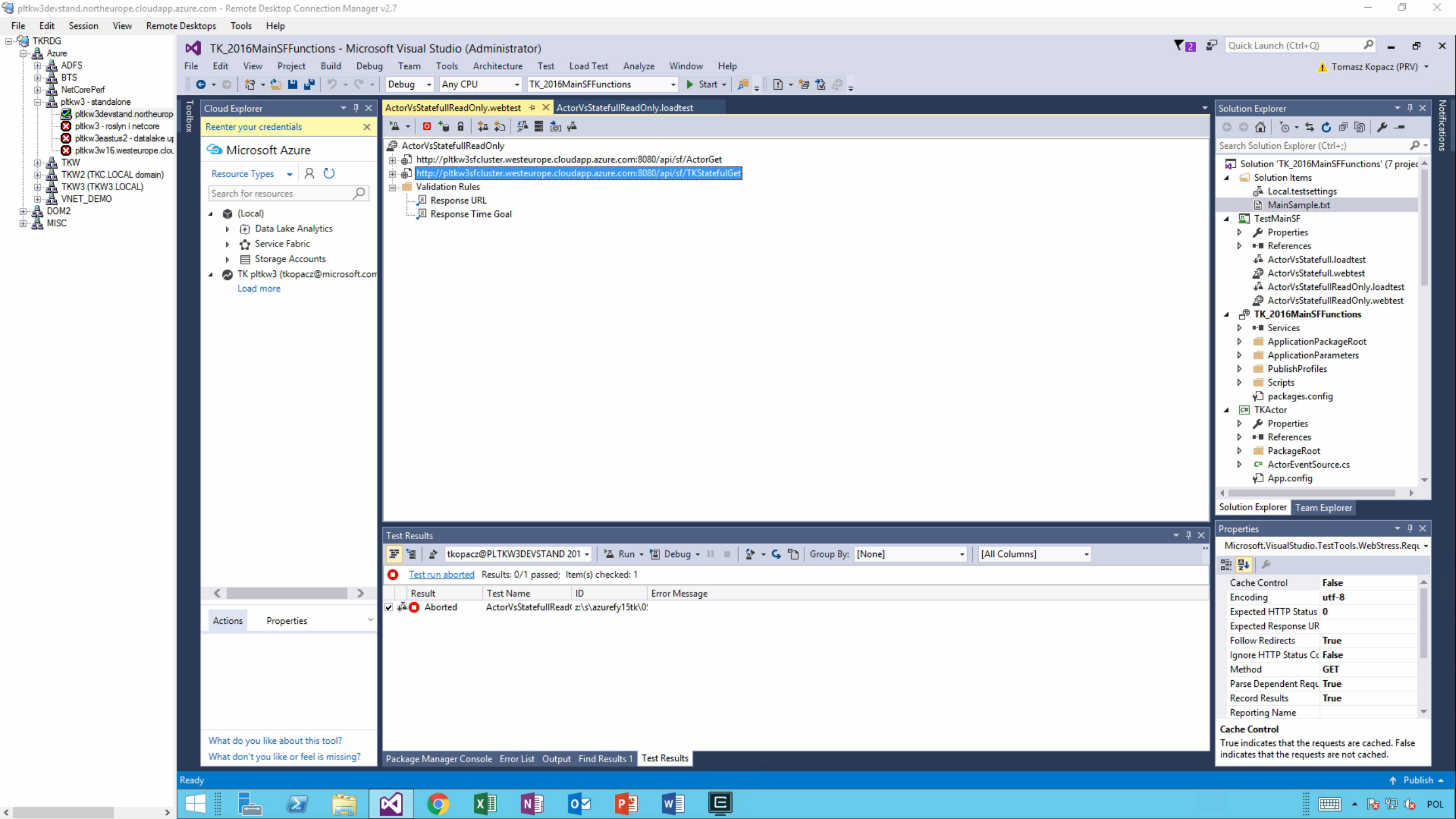
Actors: see IActorStateProvider





# Demo

Backup. Look at the code!  
Scale – video, sorry (too long!)



# Demo

Chaos testing service

Cloud Explorer

Microsoft Azure

Resource Types

Search for resources

- (Local)
- HACKATON01 (tkopacz@microsoft.com)
- TK - HDInsight PW (tkopacz@microsoft.com)
- TK pltkw3 (tkopacz@microsoft.com)

```
10 <StatelessServiceType ServiceTypeName="ChaosTestWebServiceType" />
11 </ServiceTypes>
12
13 <!-- Code package is your service executable. -->
14 <CodePackage Name="Code" Version="1.0.0.0">
15   <EntryPoint>
16     <ExeHost>
17       <Program>ChaosTest.WebService.exe</Program>
18     </ExeHost>
19   </EntryPoint>
20 </CodePackage>
21
22 <!-- Config package is your service configuration that contains an
23      independent configuration package for your service. -->
24 <ConfigPackage Name="Code" Version="1.0.0.0">
25
26   <Resources>
27     <Endpoints>
28       <!-- This endpoint is the endpoint on which to
29            listen for requests. The endpoint is shared with
30            replicated instances of the service. -->
31       <Endpoint Name="Code" Address="http://*:*:8081" />
32     </Endpoints>
33   </Resources>
34 </ConfigPackage>
```

Publish Service Fabric Application

Target profile: PublishProfiles\Cloud.xml

Microsoft tkopacz@microsoft.com

Connection Endpoint: pltkw3sfcluster.westeurope.cloudapp.azure.com:19000

Advanced Connection Parameters

Application Parameters File: ApplicationParameters\Cloud.xml

Upgrade the Application

Manifest Versions...

Publish Cancel

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'ChaosTestApplication' (4 projects)
  - Solution Items
  - ChaosTest.ChaosService
  - ChaosTest.Common
  - ChaosTest.WebService
    - Properties
    - References
    - App\_Start
    - Controllers
      - DefaultController.cs
    - Extensions
    - PackageRoot
      - Config
        - ServiceManifest.xml
    - wwwroot
      - App.config
      - ChaosTestWebServiceConfigSettings.cs
      - packages.config
      - Program.cs
      - ServiceEventSource.cs
      - Startup.cs
      - WebService.cs

Properties

ChaosTestApplication Project Properties

Application Debug Mode	Remove Application
Application URL	
Project File	ChaosTestApplication.sfproj
Project Folder	C:\AzureFY15TK\05PaaS_Service\

Output

Show output from: General

What do you like about this tool?  
What don't you like or feel is missing?

Task Runner Explorer Web Publish Activity Error List Output Find Results 1 Immediate Window Bookmarks Test Results

Application Debug Mode

Indicates behavior of application deployment as part of debugging. Remove Application - The application will be re...

# Similar action - PowerShell

```
$timeToRun = 60  
$maxStabilizationTimeSecs = 180  
$concurrentFaults = 3  
$waitTimeBetweenIterationsSec = 60
```

```
Invoke-ServiceFabricChaosTestScenario -TimeToRunMinute $timeToRun -  
MaxClusterStabilizationTimeoutSec $maxStabilizationTimeSecs -MaxConcurrentFaults  
$concurrentFaults -EnableMoveReplicaFaults -WaitTimeBetweenIterationsSec  
$waitTimeBetweenIterationsSec
```

```
$serviceName = "fabric:/WordCount/WordCountService"
```

```
Invoke-ServiceFabricFailoverTestScenario -TimeToRunMinute  
$timeToRun -MaxServiceStabilizationTimeoutSec $maxStabilizationTimeSecs -  
WaitTimeBetweenFaultsSec $waitTimeBetweenFaultsSec -ServiceName $serviceName -  
PartitionKindUniformInt64 -PartitionKey 1
```

Short: Service Fabric on  
premise, for free

# Options

## Same code can be run on:

- Service Fabric as a part of Azure Stack
- Service Fabric on single VM on anything (Hyper-V, physical, XEN)
- 3 VMs, 5 VMs, etc...

## [Materials here](#)

## Steps:

1. Setup ClusterConfig.json (nodeTypes, nodes, faultDomain, upgradeDomain, reliabilityLevel)
2. Allow communication between nodes (file sharing – simplest)
3. `.\CreateServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json -AcceptEULA`
4. **THOSE ARE REALLY ALL STEPS!**

I am excited to announce today that Azure Service Fabric for Windows Server will be generally available for download at no cost. With today's announcement, customers can now provision Service Fabric clusters in their own data centers or other cloud providers and run production workloads with the option to purchase support for ultimate confidence. One such customer is [Owners.com](#), an online platform that gives consumers a convenient

```
PS C:\Microsoft.Azure.ServiceFabric.WindowsServer.5.3.204.9494> .\CreateServiceFabricCluster.ps1 -ClusterConfigFilePath
.\ClusterConfig.json -AcceptEULA
Creating Service Fabric Cluster...
If it's taking too long, please check in Task Manager details and see if Fabric.exe for each node is running. If not, p
lease look at: 1. traces in DeploymentTraces directory and 2. traces in FabricLogRoot configured in ClusterConfig.json.
Trace folder already exists. Traces will be written to existing trace folder: C:\Microsoft.Azure.ServiceFabric.WindowsS
rver.5.3.204.9494\DeploymentTraces
Running Best Practices Analyzer...
Best Practices Analyzer completed successfully.
Creating Service Fabric Cluster...
Processing and validating cluster config.
Adding Section=Setup with 2 parameters.
Adding Section=Security with 6 parameters.
Adding Section=FileStoreService with 5 parameters.
Adding Section=FailoverManager with 4 parameters.
Adding Section=Diagnostics with 5 parameters.
Adding Section=WinFabEtWFile with 3 parameters.
Adding Section=WinFabCrashDump with 4 parameters.
Adding Section=WinFabPerfCtrFolder with 4 parameters.
Adding Section=FileShareWinFabEtW with 5 parameters.
Adding Section=FileShareWinFabCrashDump with 5 parameters.
Adding Section=FileShareWinFabPerfCtr with 5 parameters.
Adding Section=UpgradeOrchestrationService with 5 parameters.
Adding Section=ClusterManager with 3 parameters.
Adding Section=NamingService with 3 parameters.
Adding Section=ImageStoreService with 3 parameters.
Adding Section=FaultAnalysisService with 3 parameters.
Adding Section=Management with 1 parameters.
Adding Section=Federation with 1 parameters.
Adding Section=Hosting with 3 parameters.
Adding Section=HttpGateway with 1 parameters.
Adding Section=Trace/Etw with 1 parameters.
Adding Section=ReconfigurationAgent with 1 parameters.
Baseline upgrade started.
Configuring nodes.
Default installation directory chosen based on system drive of machine p1tkw3srvfab0.TKW3.LOCAL
Copying installer to all machines.
Configuring machine p1tkw3srvfab0.TKW3.LOCAL
```



Recycle Bin

```

Administrator: Windows PowerShell
PS C:\Microsoft.Azure.ServiceFabric.WindowsServer.5.3.204.9494> .\CreateServiceFabricCluster.ps1 -ClusterConfigFilePath
.\ClusterConfig.json -AcceptEULA

```

ClusterConfig.json - Notepad

```

File Edit Format View Help
{
  "name": "Sa
  "clusterCor
  "apiVersion
  "nodes": [
    {
      "nodeNa
      "iPAddr
      "nodeTy
      "faultD
      "upgrad
    },
    {
      "nodeNa
      "iPAddr
      "nodeTy
      "faultD
      "upgrad
    },
    {
      "nodeNa
      "iPAddr
      "nodeTy
      "faultD
      "upgrad
    },
    {
      "nodeNa
      "iPAddr
      "nodeTy
      "faultD
      "upgrad
    }
  ]
}

```

Local Disk (C:)

File Home Share View Extract

This PC Local Disk (C:)

Name	Date modified	Type	Size
Microsoft.Azure.ServiceFabric.WindowsServer.5.3.204.9494	10/11/2016 11:33 ...	File folder	
PerfLogs	8/22/2013 5:52 PM	File folder	
Program Files	10/11/2016 11:33 ...	File folder	
Program Files (x86)	8/22/2013 5:39 PM	File folder	
Users	10/11/2016 10:00 ...	File folder	
Windows	10/11/2016 1:02 PM	File folder	
Microsoft.Azure.ServiceFabric.WindowsServer.5.3.204.9494	10/7/2016 10:38 PM	Compressed (zipp...	56,475 KB

7 items 1 item selected 55.1 MB

Activate Windows  
Go to System in Control Panel to activate Windows.

Datcenter  
Build 9600



Security

# Options

Two aspects:

Cluster Authentication (node to node)

Server Authentication (management endpoint)

First: Internal Gateway in VNET (SF not visible)

Host: Azure

Key vault for storing certificates

Azure Active Directory for client Authentication

Service Fabric Management URL need to be in Replay URL – as in normal Web App registered in AAD

[Doc and procedures here](#)

Host: On Premise

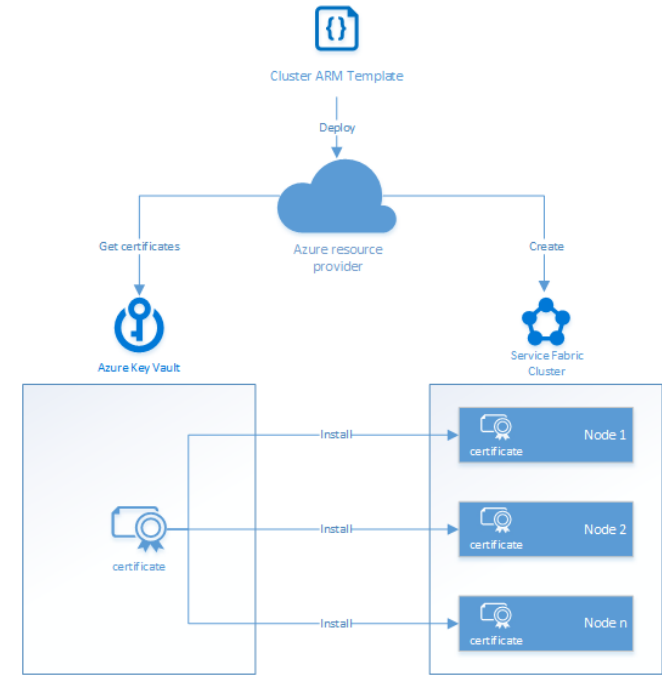
(Unsecure)

Windows Authentication (domain required!)

X509 certificate (I think that this is best options – similar to solution used in Azure)

Remember there is also app in SF (and app need to be secured as well!)

Web Apps is much more convenient for UI. Service Fabric can host backend (.NET Core, API, ...)



# Summary

# „Summary“

Service Fabric is a great platform for building applications

And for hosting microservices as well 😊

Not limited to REST-based communication model

Many internal capabilities

(and – please do not „fight“ with them!)

But – DDD, CQRS, correct architecture – as usual!

WebApi as a frontend

Materials:

<https://github.com/tkopacz/2016DeveloperDays>

# Additional links – mainly architecture

## „Required“ Reading

[Domain-Driven Design, Tackling Complexity in the Heart of Software](#)  
[Implementing Domain-Driven Design](#)

## Online Resources

[Design Patterns](#), [CQRS](#)

<http://martinfowler.com/> , [dddcommunity.org](#)

[Eric Evans on DDD: Strategic Design](#), [Domain Driven Design Quickly](#), [DDD: putting the model to work](#)  
[Patterns and Practices: CQRS](#), [CQRS Journey](#), [Introduction to CQRS](#), [Azure guidance \(general\)](#)

[Greg Young on CQRS \(YouTube\)](#), [7 Hours of Greg Young \(YouTube\)](#)

### Courses

[DDD fundamentals](#), [DDD in practice](#), [Modern architecture: CQRS + ES](#)

## Samples (bigger one!)

[IoT samples](#), [Java sample](#), [Web Reference App](#)

<https://github.com/CESARDELATORRE/MyWorld> (work in progres!)



# Sponsors and Partners

## Strategic Sponsors

The logo for adform, featuring the word "adform" in a lowercase, sans-serif font. The letter "o" is stylized with a blue and green circular graphic element.

## Gold Sponsors



## Silver Sponsors

