Microsoft

# Service Fabric – Health, Monitoring and Operational Telemetry

Microsoft Services

## Agenda

- Overview of Monitoring Services
- Health Reporting
- Event Trace Data

Microsoft

# Service Fabric – Health, Monitoring and Operational Telemetry

*Overview - Monitoring Services*

Microsoft Services

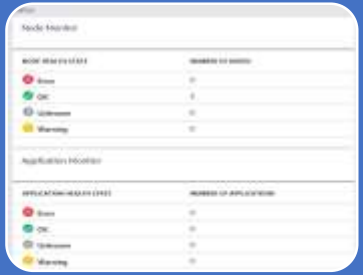
# Monitoring scenarios for your services

Visibility into how your services are doing when running in production

5.42
5.4
5.13
4
979.82
585.68
196.33
106.29
90.04

## Performance and stress response

- Built-in metrics for Actors and Services programming models
- Ability to add custom application performance metrics



Node Monitor

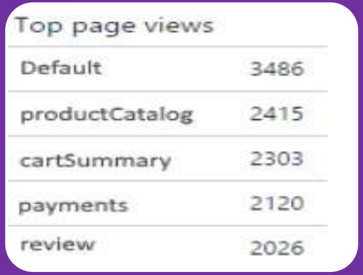
Error
OK
Unknown
Warning

Application Monitor

Error
OK
Unknown
Warning

## Health status monitoring

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



| Top page views | |
|---|---|
| Default | 3486 |
| productCatalog | 2415 |
| cartSummary | 2303 |
| payments | 2120 |
| review | 2026 |

## Business telemetry

- Collect, analyze and drive insights from your customers' interaction with your application
- Allows data-driven decision making

# 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 code runs on OneBox and also on production clusters on Azure.
- Easy to add and system appends all the needed metadata such as node, app, service, and partition.

**Choice of Tools**

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

Microsoft

## Service Fabric – Health, Monitoring and Operational Telemetry

*Health Reporting*

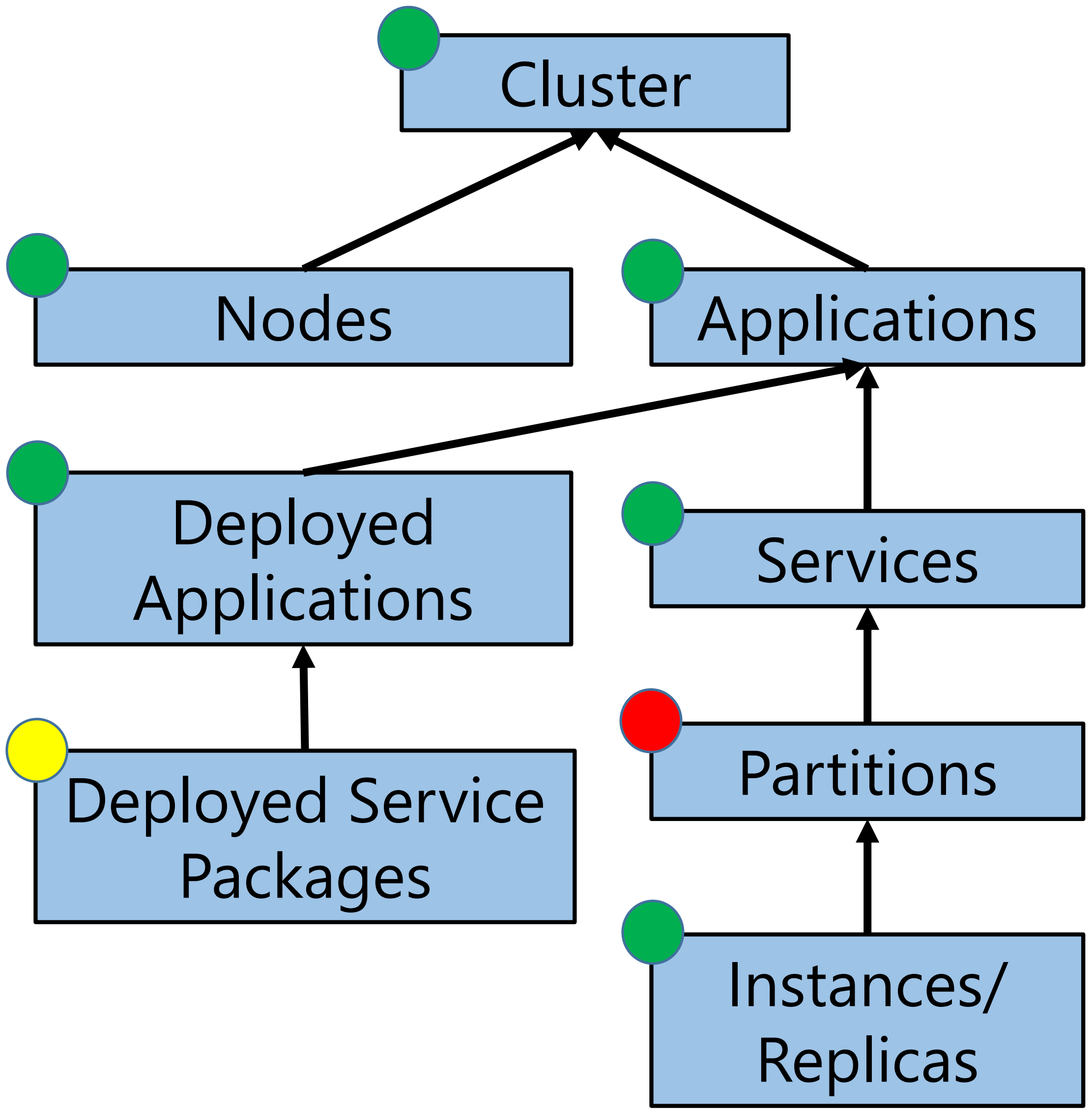
Microsoft Services

## Health Architecture – Health Store

- The health store keeps health-related information about entities in the cluster for easy retrieval and evaluation
- It is implemented as a Service Fabric persisted stateful service to ensure high availability and scalability
- The health store is part of the **fabric:/System** application, and it is available as soon as the cluster is up and running

# Health Entities, Events, and States

- Each <u>entity</u> has a set of health <u>events</u>
- Each <u>event</u> has a health <u>state</u>:
  - 🟢 OK: No issues
  - 🟡 Warning: An issue that may fix itself (ex: unexpected delay)
  - 🔴 Error: Issue requiring action
  - Unknown: Entity not in health store
- When evaluating an entity
  - Service Fabric aggregates entities and descendants' events against policy



# Health Policies

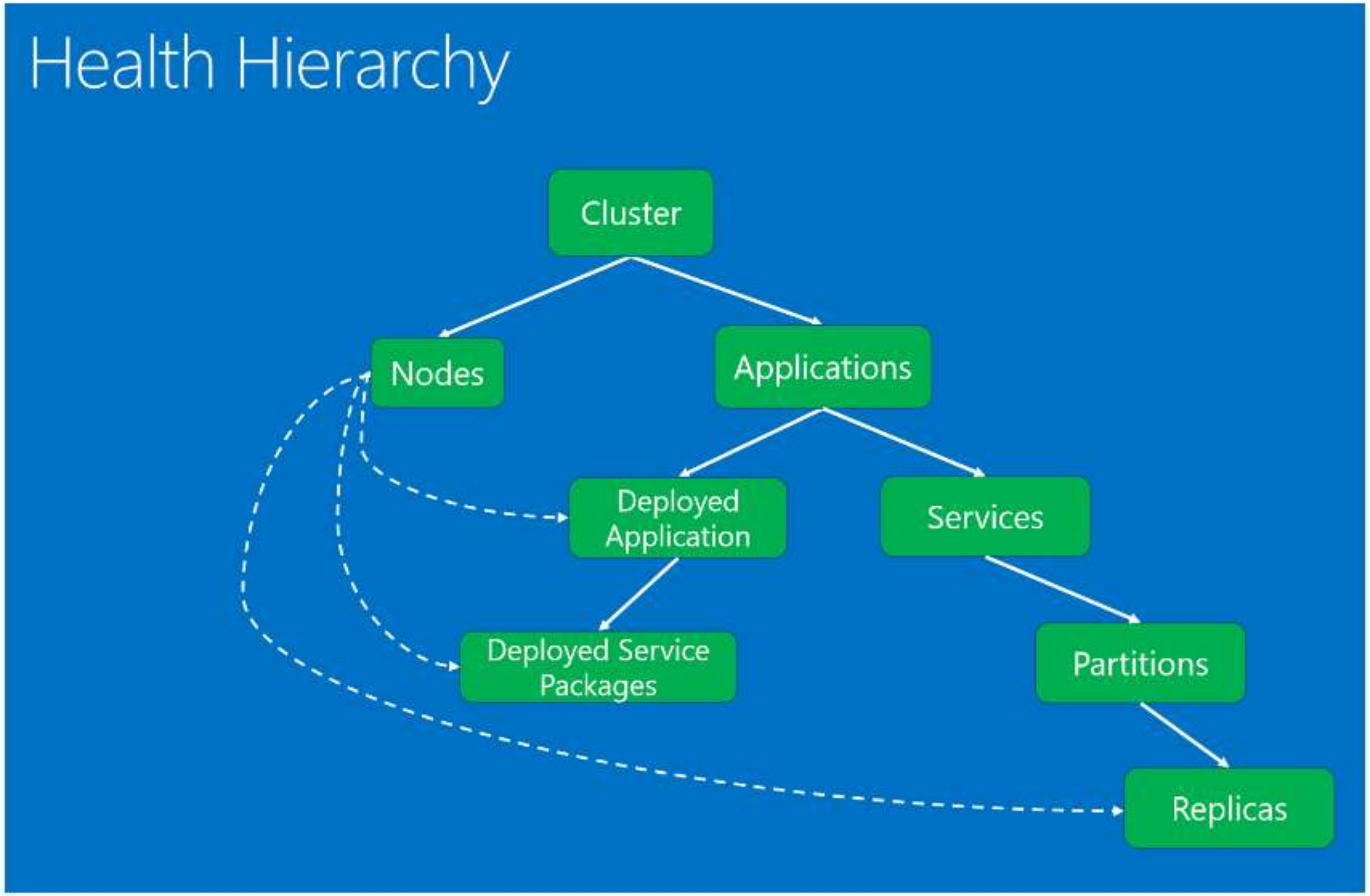
- Default: entity is healthy if it and children are healthy
  - In a world with regular failures, 20% errors might be considered warning – not to Service Fabric

- Health policies define what healthy means
  - Cluster policy can be in the cluster manifest
  - App policy can be in the application manifest
  - You can pass a custom policy when querying health

```
<FabricSettings>
  <Section Name="HealthManager/ClusterHealthPolicy">
    <Parameter Name="MaxPercentUnhealthyApplications" Value="0"/>
    <Parameter Name="MaxPercentUnhealthyNodes" Value="20"/>
  </Section>
</FabricSettings>
```

```
<Policies>
  <HealthPolicy MaxPercentUnhealthyDeployedApplications="20">
    <DefaultServiceTypeHealthPolicy
      MaxPercentUnhealthyServices="0"
      MaxPercentUnhealthyPartitionsPerService="10"
      MaxPercentUnhealthyReplicasPerPartition="0"/>
    <ServiceTypeHealthPolicy ServiceTypeName="FrontEndSvcType"
      MaxPercentUnhealthyServices="0"
      MaxPercentUnhealthyPartitionsPerService="20"
      MaxPercentUnhealthyReplicasPerPartition="0"/>
  </HealthPolicy>
</Policies>
```

## Health Architecture



## Demonstration

Health Reporting

## Health Failure Examples

- Cluster: Nodes not responding to periodic heartbeat
  - Applications: Partition could not be placed
    - Service: Failed to place replica(s)
      - Partition: Below target instance count
        - Replica: Replica taking too long to open/close
  - Node: Node down, certificate expiration, load capacity violation
  - Deployed Applications: Failed to download code package
    - Deployed service packages: Service package activation, code package activation, service type registration, download, upgrade validation

## Submitting Health Reports

- Have “watchdog” periodically check service instance
    - Watchdog code/process can be in or out of the cluster
    - Keep watchdog simple and “bug-free”
- Submit health reports via Azure PowerShell, REST, .NET API
  - .NET API batches reports and sends ~30 seconds (default)
- Submit helpful health reports that…
  - Prevent downtime, reduce issue investigation time, improve customer satisfaction
    - Ex: Diminishing disk space, bad performance, big queue size
    - Agents can poll health and take action (Ex: delete old files, send e-mails)
- Note: Reports are deleted when entity is deleted
  - To outlive entity, submit a report on the parent entity

## Demonstration

Submitting a Health Report

# What’s in a Health *Report*

- For each entity, Service Fabric stores 1 health report per SourceId/Property

| Mandatory Data | Description |
|---|---|
| Entity | Cluster, Node, App, Service, Partition, Replica, Deployed App, Deployed Service Package |
| SourceId | String uniquely identifies reporter |
| Property | Category (ex: “Storage” or “Connectivity”) |
| HealthState | Ok, Warning, Error |

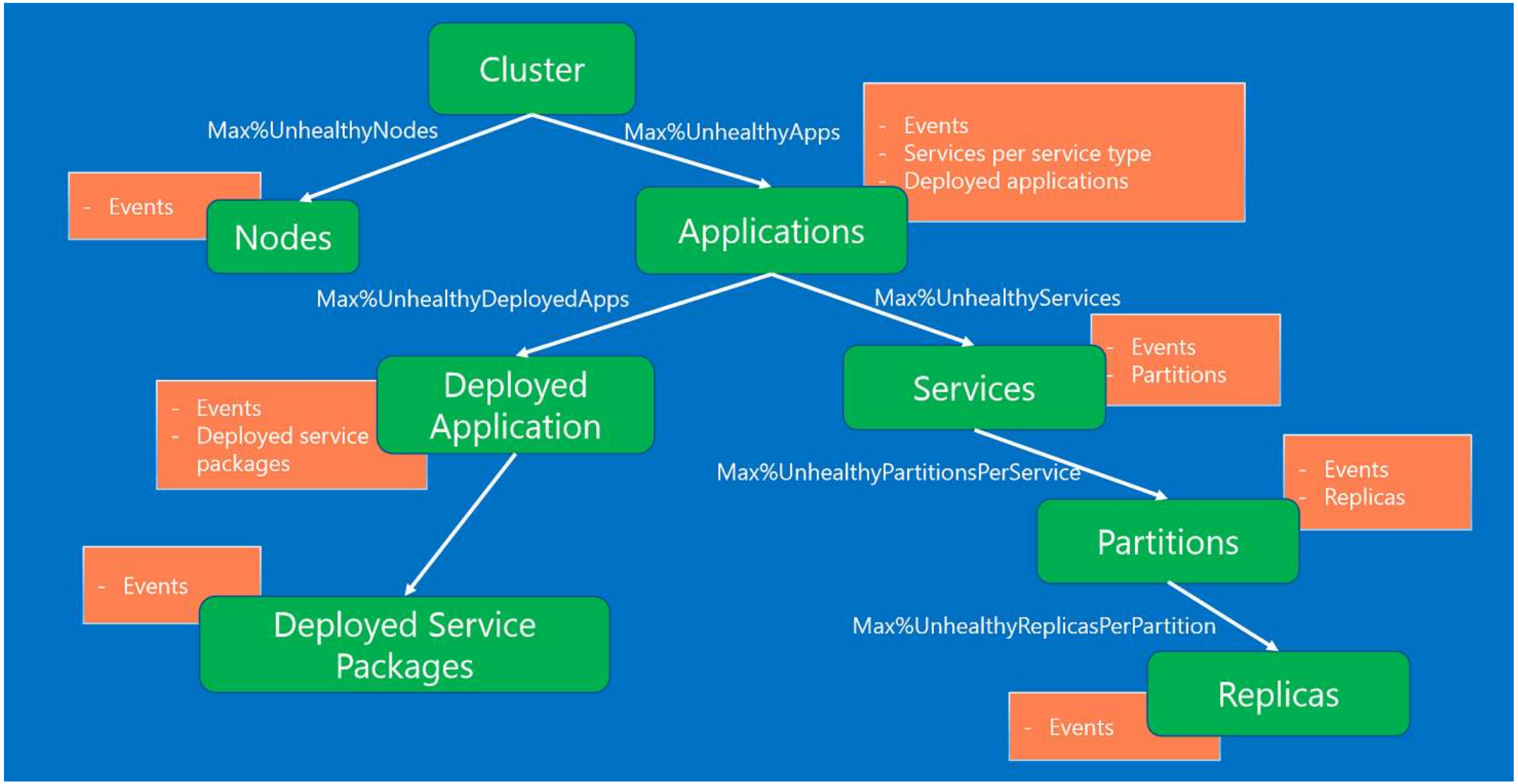
| Optional Data | Default | Description |
|---|---|---|
| Description | “” | Human readable info |
| TimeToLive | Infinite | # seconds before report is expired |
| RemoveWhenExpired | False | Useful if TTL != Infinite. If false, report’s entity is in Error; else report removed after expiration. |
| SequenceNumber | Auto-generated | Increasing integer. Use to replace old reports when reporting state transitions. |

## What's in a Health *Event*

- Service Fabric wraps a health event around a health report

| Property | Description |
|---|---|
| HealthInformation | The original health report |
| SourceUtcTimetamp | The time the health report was originally submitted |
| LastModifiedUtcTimestamp | The last time the report was modified |
| IsExpired | True if TTL expired and RemoveWhenExpired=false |
| LastOkTransitionAt<br>LastWarningTransitionAt<br>LastErrorTransitionAt | These give a history of the event's health states.<br>Ex: Alert if !Ok > 5 minutes |

# Health report aggregation



## Health Report Submission Guidance

- Never submit a report not related to health
  - Health is not a generic reporting mechanism
- Avoid reporting on state transitions because you'll have to synchronize state across failures
- Always clean up reports when no longer valid
  - Ex: Errors affect upgrades
  - So, have watchdog report periodically with TTL and RemoveWhenExpired=false
  - If the watchdog fails, set the Event's IsExpired=true and Entity's health to Error
  - To have the report self-expire, send the report with TTL and RemoveWhenExpired=true

Microsoft

## Service Fabric – Health, Monitoring and Operational Telemetry

*Event Tracing Data*

Microsoft Services

## Event Tracing for Windows

Recommended technology for tracing messages in Service Fabric

- **ETW is fast -** It was built as a tracing technology that has minimal impact on code execution times
- **ETW tracing works across local development environments and also real-world cluster setups -** This means you don't have to rewrite your tracing code when you are ready to deploy your code to a real cluster
- **Service Fabric system code also uses ETW for internal tracing -** This allows you to view your application traces interleaved with Service Fabric system traces. It also helps you to more easily understand the sequences and inter-relationships between your application code and events in the underlying system
- There is built-in support in Service Fabric Visual Studio tools to view ETW events

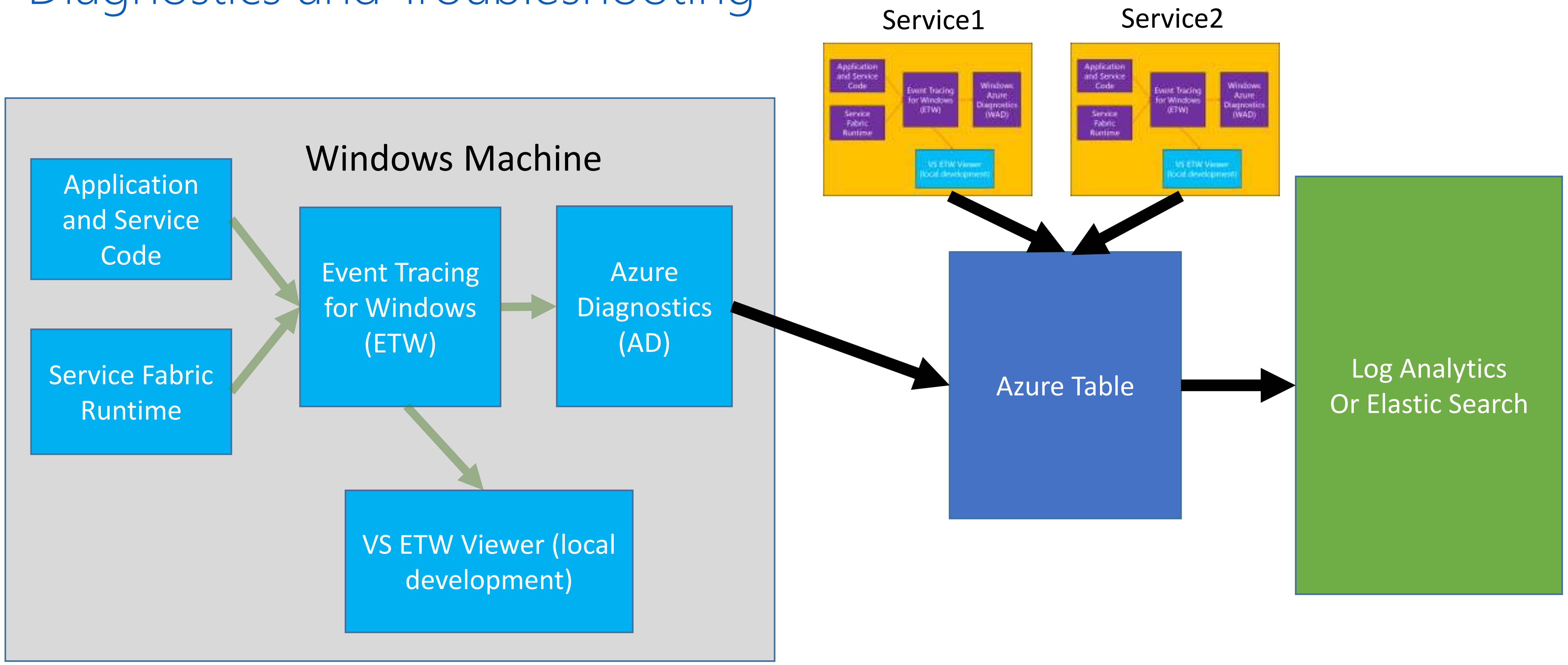
# ETW Events in Service Fabric

- Built in ETW events
  - Reliable Actors diagnostics -https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-actors-diagnostics/
  - Reliable Services diagnostics - https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-services-diagnostics/

- Custom ETW events
  - The RunAsync methods in the Visual Studio templates have examples of custom events built in
  - The EventSource classes contain events that are built into the templates, including an implementation designed for high frequency events.

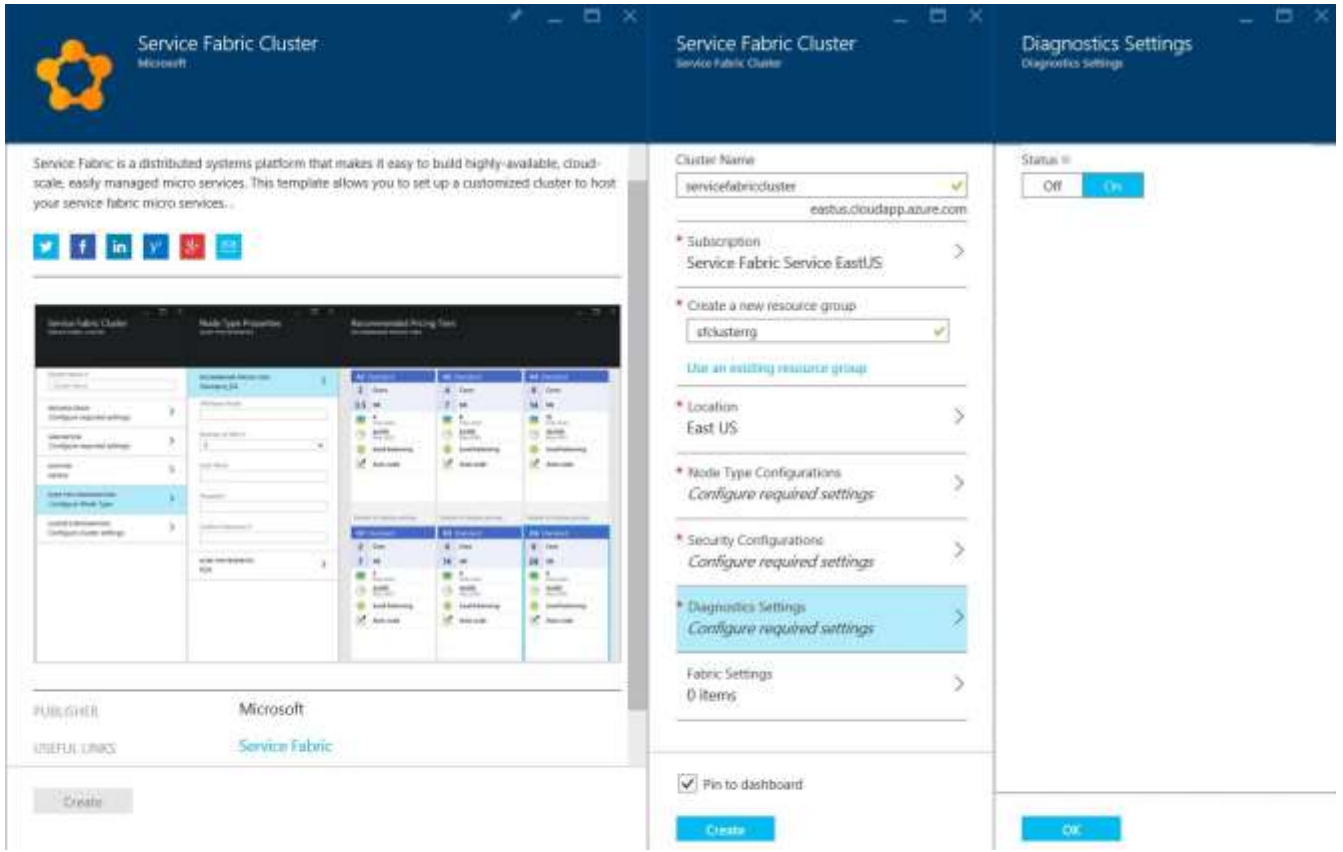
```
// Stateless...
ServiceEventSource.Current.ServiceMessage(this, "Working-{0}", ++iterations);

// Stateful...
ServiceEventSource.Current.ServiceMessage(this, "Current Counter Value: {0}",
result.HasValue ? result.Value.ToString() : "Value does not exist.");
```

# Diagnostics and Troubleshooting


Windows Machine
Application and Service Code
Service Fabric Runtime
Event Tracing for Windows (ETW)
Azure Diagnostics (AD)
VS ETW Viewer (local development)
Service1
Service2
Application and Service Code
Service Fabric Runtime
Event Tracing for Windows (ETW)
Windows Azure Diagnostics (WAD)
VS ETW Viewer (local development)
Azure Table
Log Analytics Or Elastic Search

# Diagnostics Extension

- Deployed to each VM in the cluster
- Collects logs and uploads them to a storage account
- Can configure the extension through the portal or ARM, at create time or on existing cluster



Service Fabric Cluster
Microsoft

Service Fabric is a distributed systems platform that makes it easy to build highly-available, cloud-scale, easily managed micro services. This template allows you to set up a customized cluster to host your service fabric micro services...

PUBLISHER Microsoft
USEFUL LINKS Service Fabric

Create

Service Fabric Cluster
Service Fabric Cluster

Cluster Name
servicefabriccluster
eastus.cloudapp.azure.com

* Subscription
Service Fabric Service EastUS

* Create a new resource group
sfclusterrg
Use an existing resource group

* Location
East US

* Node Type Configurations
*Configure required settings*

* Security Configurations
*Configure required settings*

* Diagnostics Settings
*Configure required settings*

Fabric Settings
0 items

[✓] Pin to dashboard

Create

Diagnostics Settings
Diagnostics Settings

Status
Off [✓ On]

OK

ARM Sample - [https://github.com/Azure/azure-quickstart-templates/tree/master/service-fabric-cluster-5-node-1-nodetype-wad](https://github.com/Azure/azure-quickstart-templates/tree/master/service-fabric-cluster-5-node-1-nodetype-wad)
Azure Diagnostics - [https://azure.microsoft.com/en-us/documentation/articles/service-fabric-diagnostics-how-to-setup-wad-operational-insights/](https://azure.microsoft.com/en-us/documentation/articles/service-fabric-diagnostics-how-to-setup-wad-operational-insights/)

# Demonstration

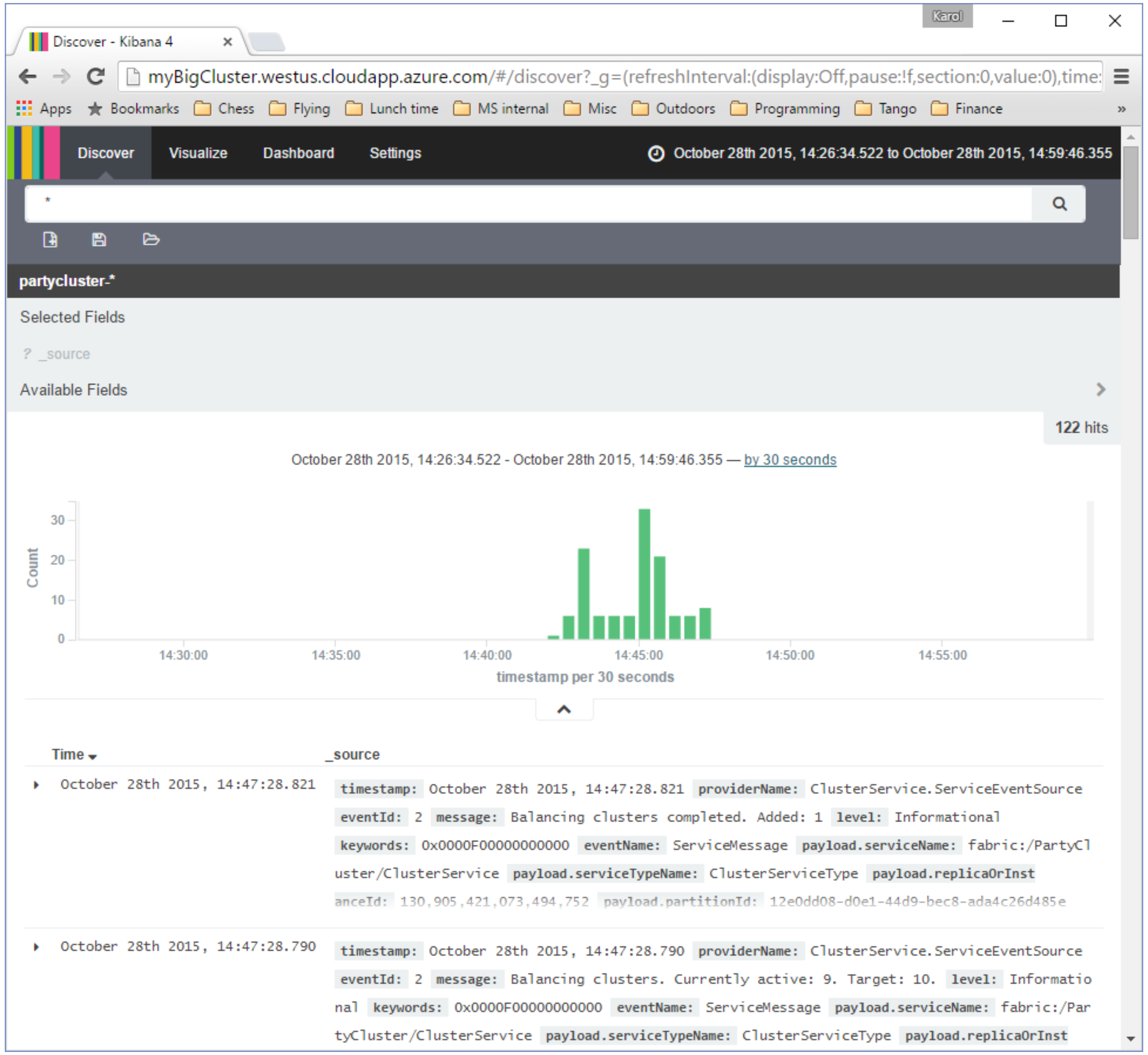
## Viewing ETW Events in Visual Studio

## Decide what to collect

- **Service Fabric logs:** Emitted by the platform to standard ETW and EventSource channels. Logs can be one of several types:
    - Operational events
    - Actor Programming Model events
    - Reliable Services Programming Model events
- **Application events:** Events emitted from your services code and written out by using the EventSource helper class provided in the Visual Studio templates

# SQL Database Elastic Search

- You can also use Elastic Search to query and make sense of the logging
- Complete step-by-step for setting this up is available here:
- [https://azure.microsoft.com/en-us/documentation/articles/service-fabric-diagnostic-how-to-use-elasticsearch/](https://azure.microsoft.com/en-us/documentation/articles/service-fabric-diagnostic-how-to-use-elasticsearch/)
- Can configure apps to send directly, or use the diagnostics extension or other aggregator



```
October 28th 2015, 14:47:28.821   timestamp: October 28th 2015, 14:47:28.821 providerName: ClusterService.ServiceEventSource
                                  eventId: 2 message: Balancing clusters completed. Added: 1 level: Informational
                                  keywords: 0x0000F00000000000 eventName: ServiceMessage payload.serviceName: fabric:/PartyCl
                                  uster/ClusterService payload.serviceTypeName: ClusterServiceType payload.replicaOrInst
                                  anceId: 130,905,421,073,494,752 payload.partitionId: 12e0dd08-d0e1-44d9-bec8-ada4c26d485e

October 28th 2015, 14:47:28.790   timestamp: October 28th 2015, 14:47:28.790 providerName: ClusterService.ServiceEventSource
                                  eventId: 2 message: Balancing clusters. Currently active: 9. Target: 10. level: Informatio
                                  nal keywords: 0x0000F00000000000 eventName: ServiceMessage payload.serviceName: fabric:/Par
                                  tyCluster/ClusterService payload.serviceTypeName: ClusterServiceType payload.replicaOrInst
```

## Recap

- Diagnostics
  - Service Fabric uses ETW + Azure Diagnostics
  - Services get EventSource class for ETW tracing
  - View VS.NET's Diagnostics Events Window or Use *Logman* to copy and view logs after the fact
- Health Reporting
  - Default health reports are provided by Service Fabric
  - Health Reports can be volatile with TTL
  - Apps', Services', Partitions', Replicas' health state are controlled by health policies
  - Report custom health events to Service Fabric via API or PowerShell

Microsoft