Starting with NServiceBus

Roland Guijt

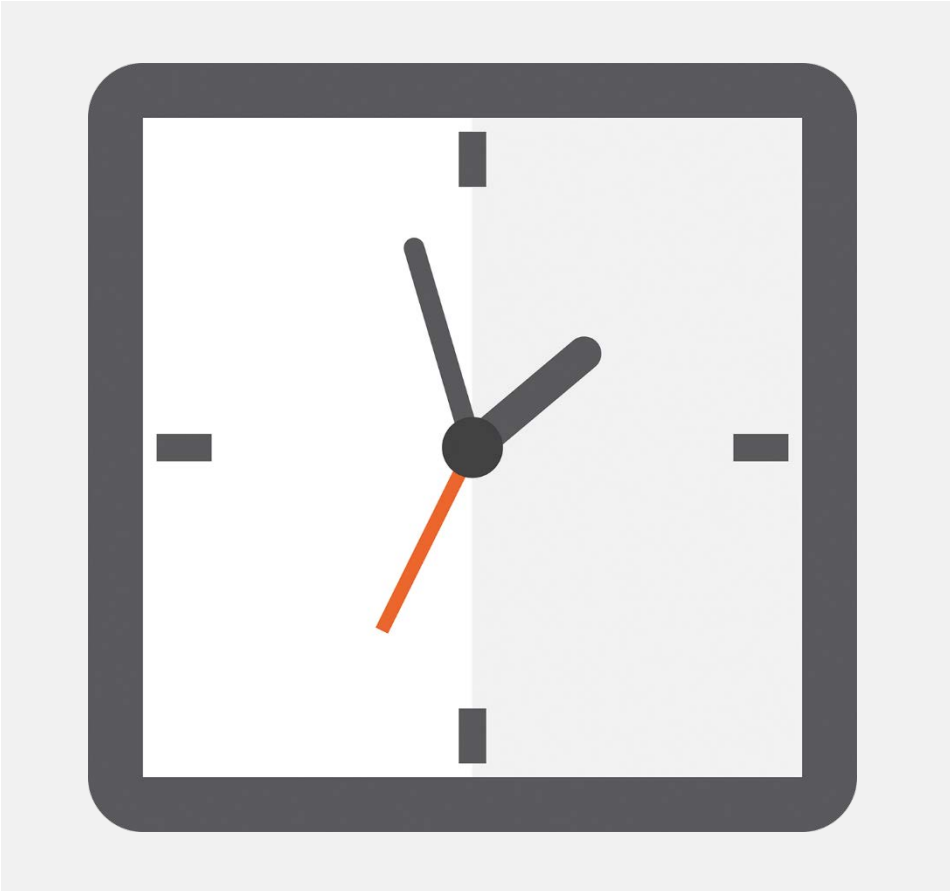
@rolandguijt | www.rmgsolutions.nl

Overview

- What is NServiceBus?
- How to prepare
- Messages
- Routing
- Configuration
- Fault Tolerance

History

- Lack of MSMQ support in .NET
- Open source since 2007
- Udi Dahan - creator/main contributor
- Requires license fee since 2010

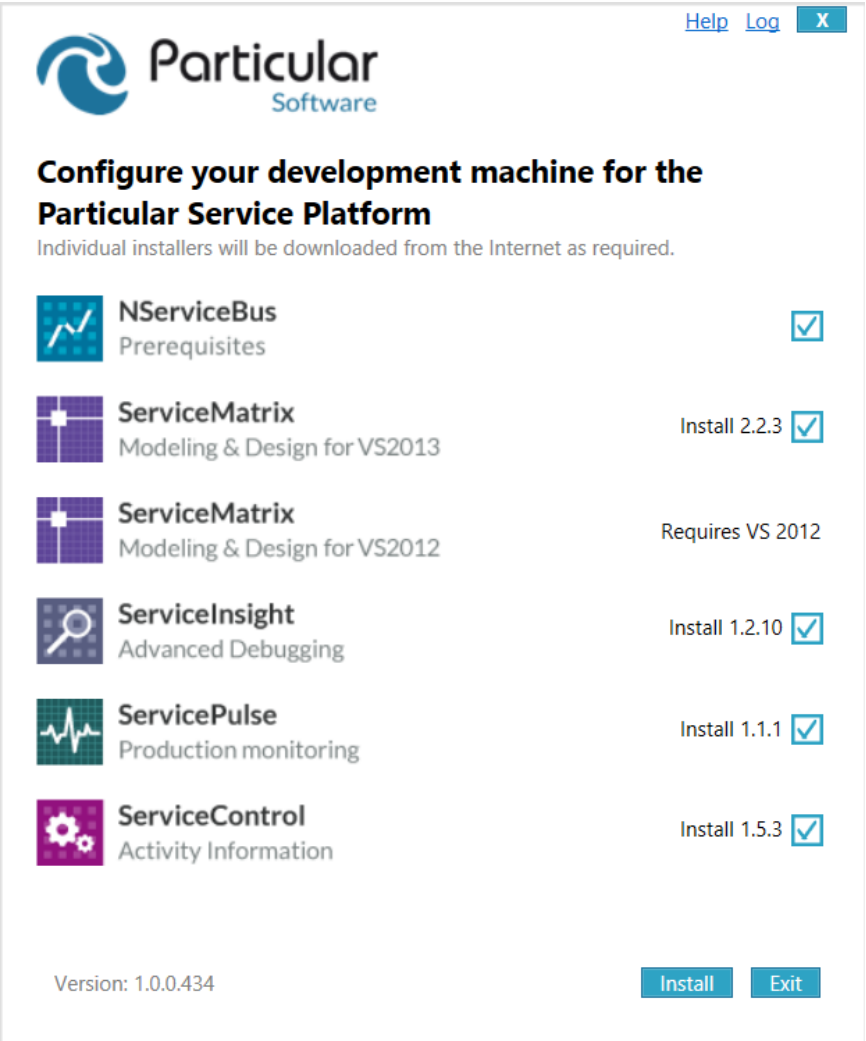


What Is NServiceBus?

- Framework enabling communication between applications using messaging
- Part of Particular Service Platform
- Lies on top of messaging back-ends called transports
- Transports are configurable
- Open source, but not free

Preparation

http://particular.net/downloads



NuGet Packages

NServiceBus

NServiceBus.Host

NServiceBus.Testing

Demo: Fire on Wheels Goes NServiceBus

FIRE ON WHEELS

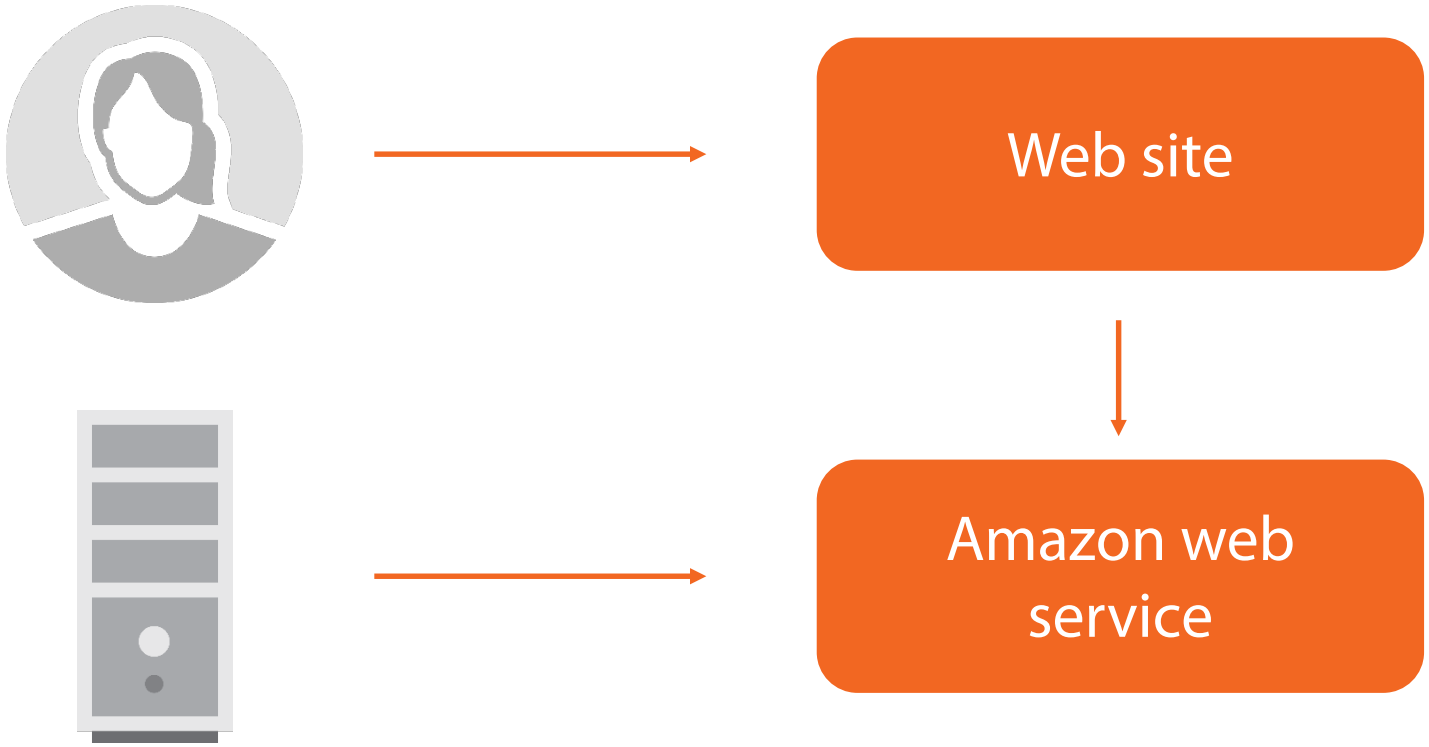
Business is even better

Orders are getting lost!

Too much load on the service

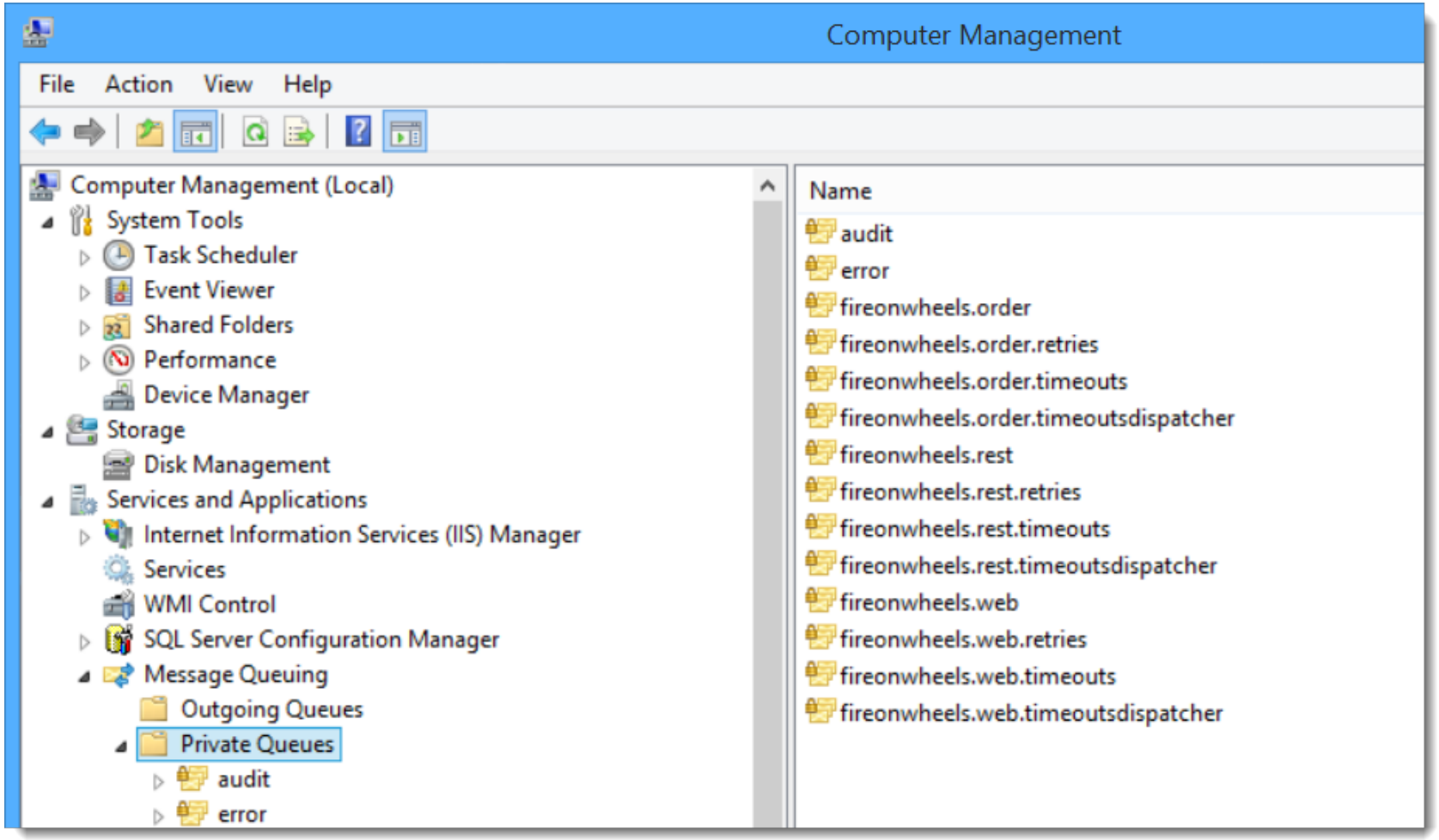
Process orders one by one

The Old Architecture



The New Architecture

MSMQ Queues



Computer Management -> Service and Applications -> Message Queueing -> Private Queues

Demo: Starting with NServiceBus

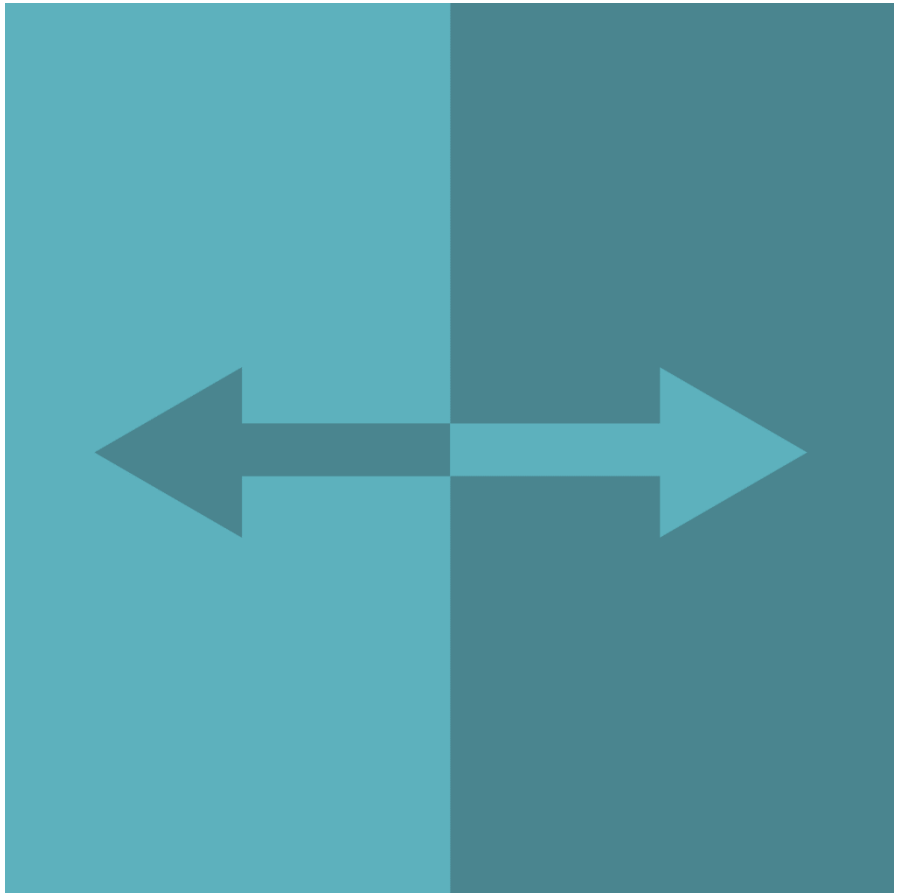
Business is even better

Orders are getting lost!

Too much load on the service

Process orders one by one

Commands



Messages

One or more senders

One receiver

Bus.Send

Marked with ICommand

Imperative: ProcessOrderCommand

Dependency Injection

NServiceBus relies on it

Built in the core

Works with NServiceBus managed types

Or use your own DI container

Assembly Scanning

IHandleMessages

Scans all assemblies

Can be limited

Events

Messages

One sender

One or more receivers

Bus.Publish

Publish/Subscribe pattern

Marked with IEvent

Past tense: OrderProcessedEvent

Routing

Can be done in config file

No need to redeploy when endpoint names change, etc.

Recommended

Commands: In sender, routed to endpoints that receive the command

Events: In receiver, routed to endpoints that receive the subscription request

```
<UnicastBusConfig>
    <MessageEndpointMappings>
    </MessageEndpointMappings>
</UnicastBusConfig>
```

Routing: The app.config Section

Routing rules or mappings go here

Every mapping in an <add> node

```
<add Messages="assembly" Endpoint="destination" />
<add Assembly="assembly" Type="namespace.type"
Endpoint="destination" />
<add Assembly="assembly" Namespace="MyMessages.Other"
Endpoint="destination" />
```

Routing: Mappings

Map entire assembly to an endpoint

Or one specific type in an assembly

Or all the types in one namespace within an assembly

Demo: Events

FIRE ON WHEELS

Publish event when order is processed

Subscribe to the event in the web application

Configuration

```
public class EndpointConfig : IConfigureThisEndpoint
{
    0 references
    public void Customize(BusConfiguration configuration)
    {
        configuration.UsePersistence<InMemoryPersistence>();
    }
}
```

```
var config = new BusConfiguration();
config.UsePersistence<InMemoryPersistence>();
config.UseTransport<MsmqTransport>();
config.EndpointName(endpointName);
config.PurgeOnStartup(true);
config.EnableInstallers();
Bus = NServiceBus.Bus.Create(config).Start();
```

Relies on defaults

Combination code/config file

IConfigureThisEndpoint

INeedInitialization

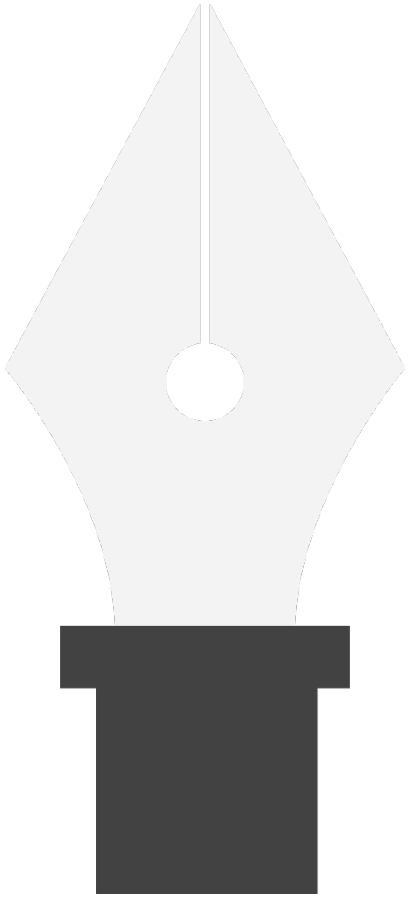
IWantToRunWhenBusStartsStops

```
NServiceBus.Bus.CreateSendOnly(config);
```

Send Only Endpoint

No overhead for receiving messages

Serialization



Default: XML, soon JSON

```
config.UseSerialization<JsonSerializer>();
```

BSON, Binary or write your own

Logging

Use built-in or your favorite logging framework

Written to console, trace and rolling file

Set threshold in config file

Debug

Info

Warn

Error

Fatal

Persistence

Used internally by NServiceBus

No default

InMemoryPersistence: Built into core, for testing

NHibernatePersistence: SQL server and Oracle

RavenDbPersistence

AzurePersistence

```
config.UsePersistence<InMemoryPersistence>()
.For(Storage.Subscriptions);
config.UsePersistence<NHibernatePersistence>()
.For(Storage.Timeouts, Storage.Sagas);
config.UsePersistence<RavenDbPersistence>()
.For(Storage.Outbox);
```

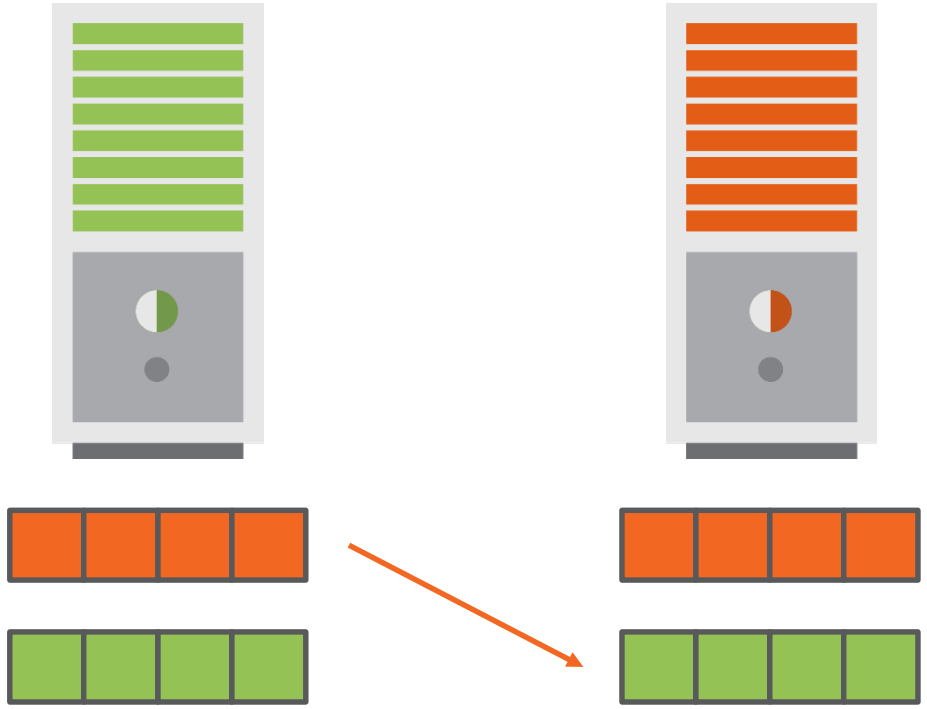
Using Multiple Persistence Types

MSMQ

Native to Windows

Decentralized: each machine has its own queues

Store and forward

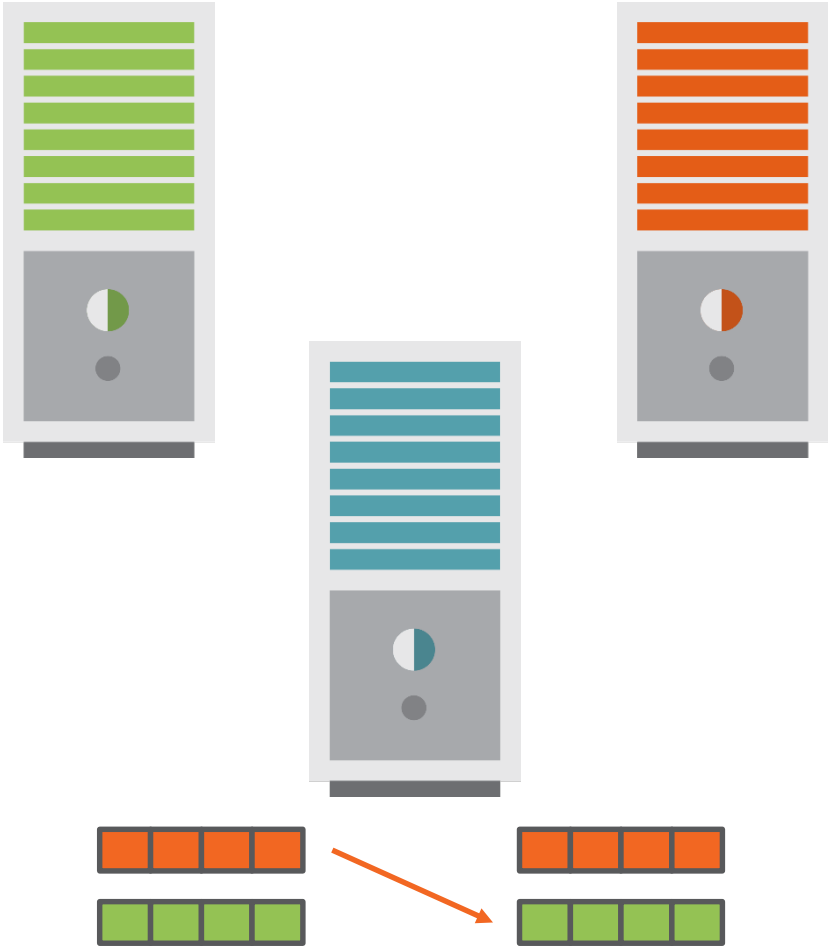


RabbitMQ

Broker style

Multiple platforms

Supports AMQP



SQL Server

Use table as queue

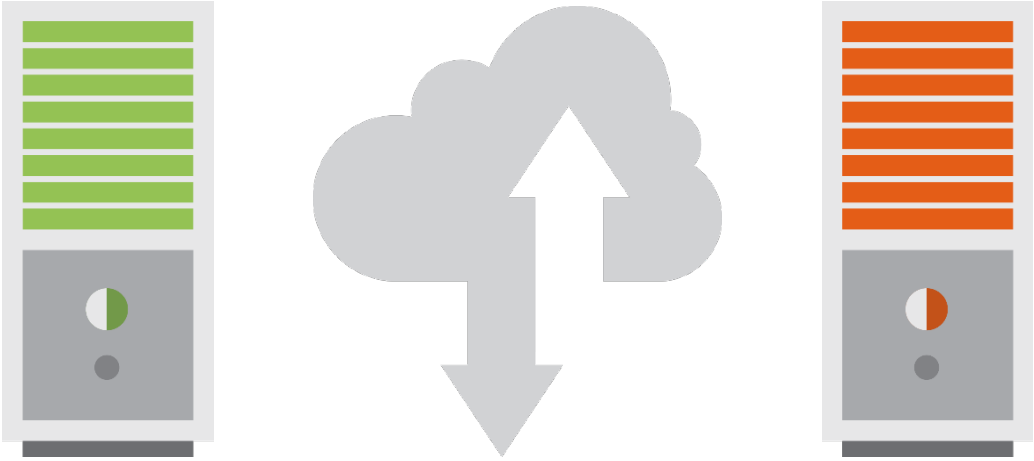
Polling

For small projects and conversions

Windows Azure

Queues

Service bus



Remember

The transport is an abstraction

Easy to switch

Configuration detail

Hosting and Deployment

NServiceBus hosted

- NServiceBus.Host NuGet package
- Configures using assembly scanning
- Runs as console app when debugging
- Can be easily installed as a Windows service
- Supports profiles

Self hosted

- No extra NuGet package required
- Create config object yourself
- Does not support profiles

```
public interface INeedToInstallSomething<T> :
INeedToInstallSomething where T : IEnvironment
{
    void Install(string identity);
}
```

Installers

Depending on configuration create e.g. queues and schemas

INeedToInstallSomething interface

Different behaviors for debugging, self-hosting, and NServiceBus hosted

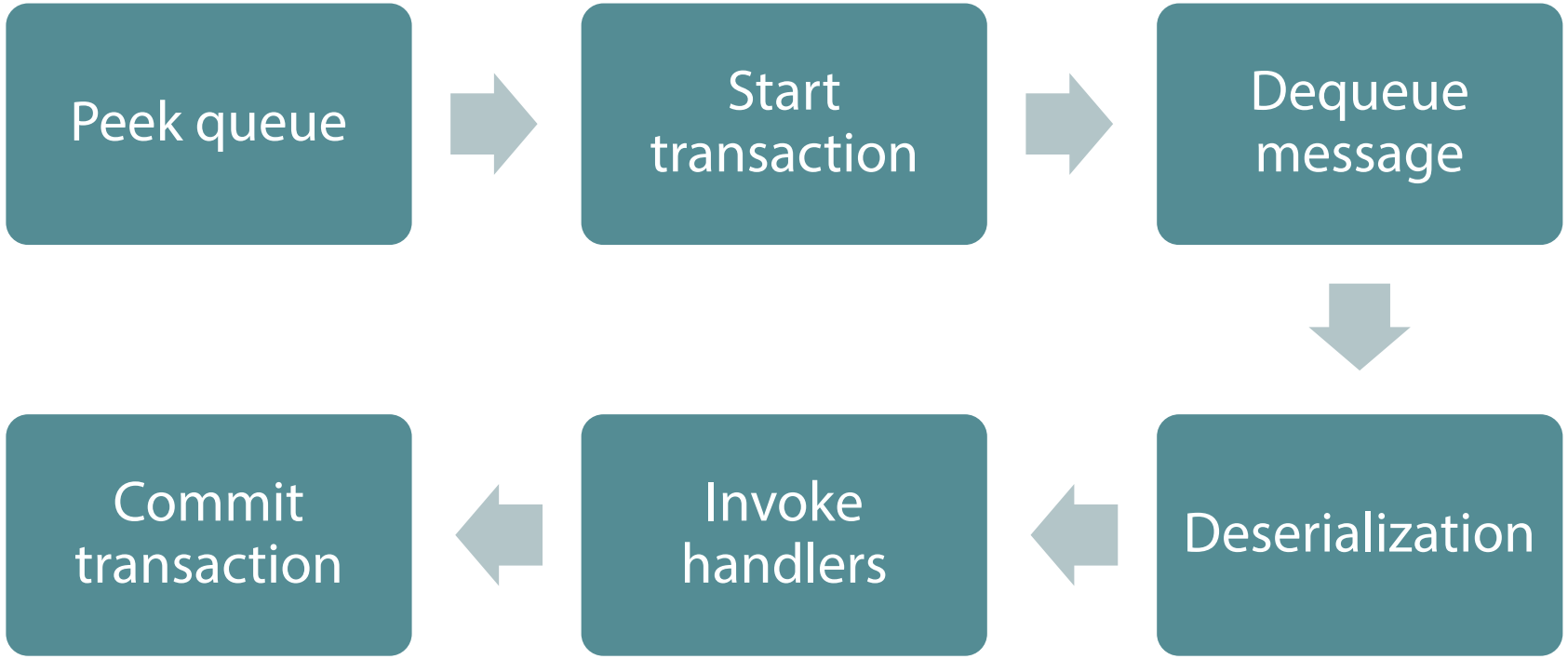
Profiles

- Can only be used in NServiceBus hosted mode
- Define a certain set of defaults
- Lite, Integration, Production built-in
- Create your own using IProfile interface
- Extend behavior using IHandleProfile interface
- Run using command line parameter or during Windows service installation

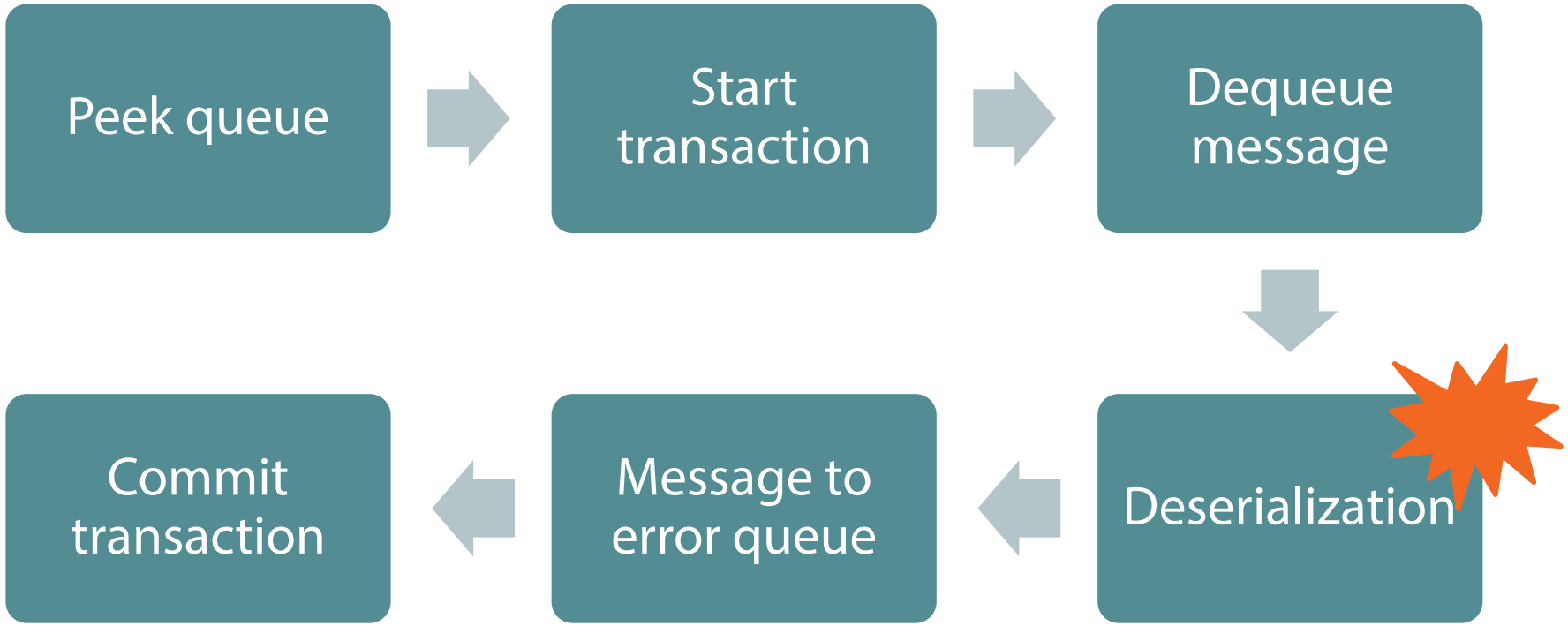
Fault Tolerance

- You know:
  Software will fail
  Infrastructure will fail
- You want:
  No loss of data
  Resiliency

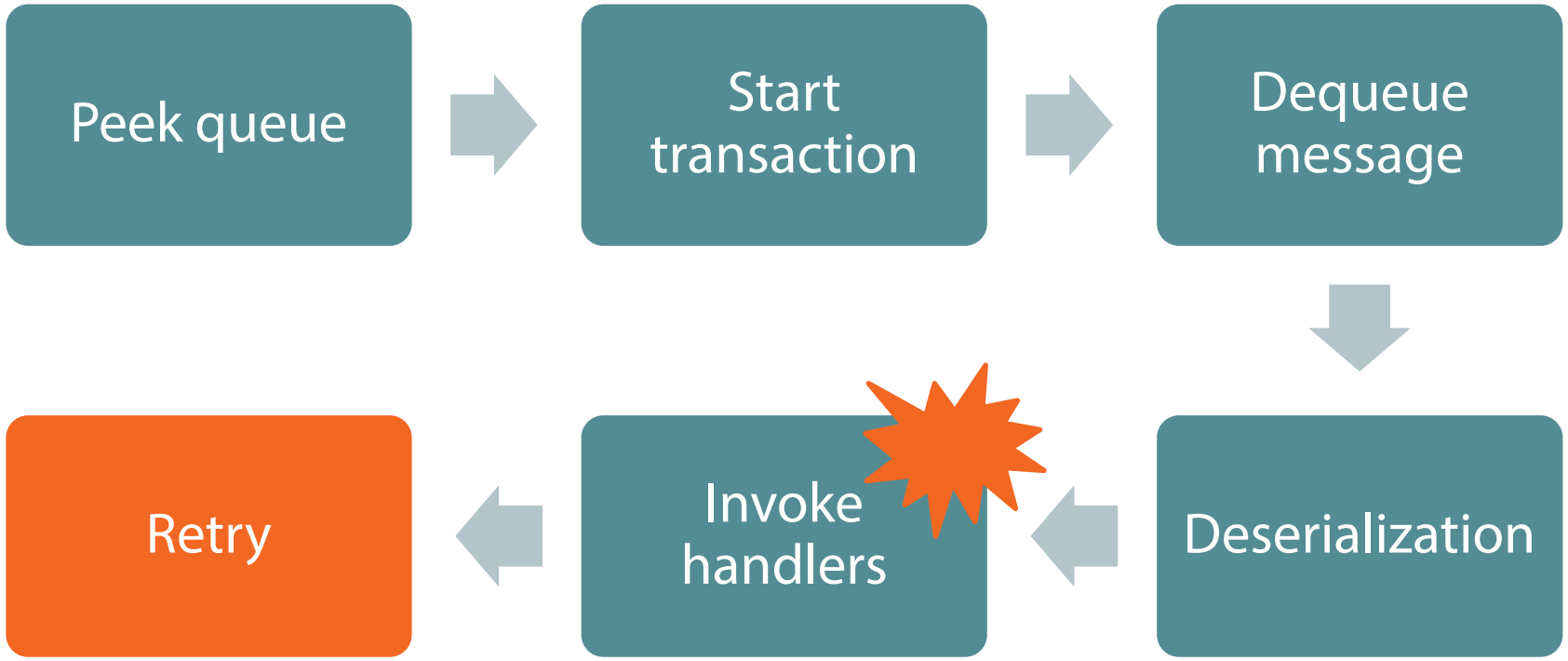
Happy Path



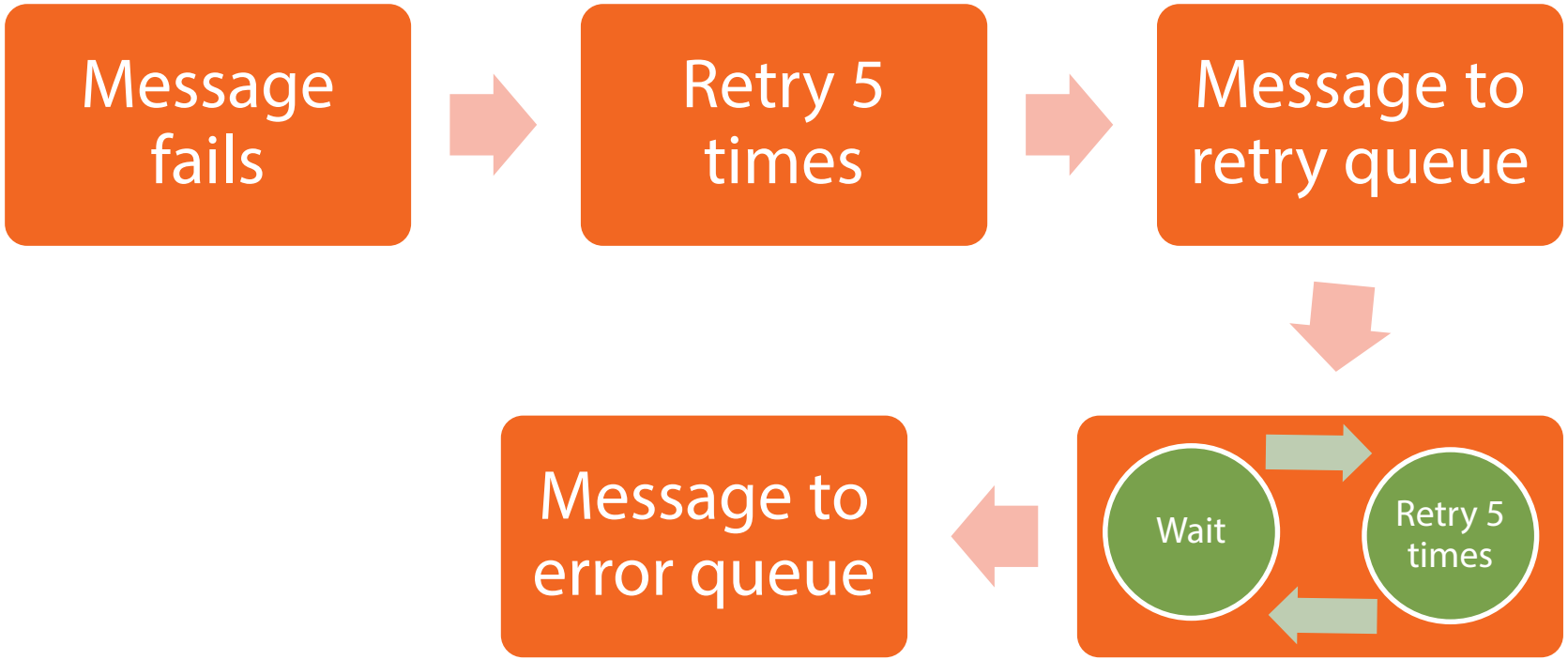
Failure During Deserialization



Failure of Handler(s)



Retries

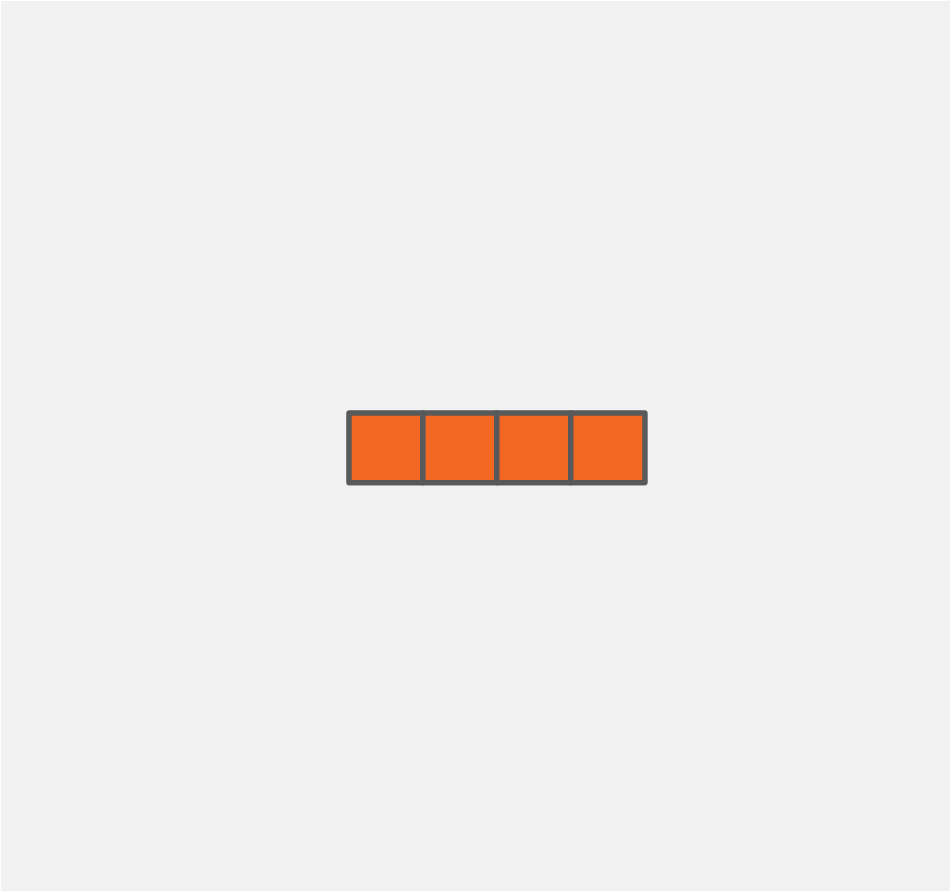


Second Level Retries (SLR)

- Configurable: time increase and number of retries
- Heads-up
  Transient errors won't show up in error queue
  Will take some time before it's in the error queue

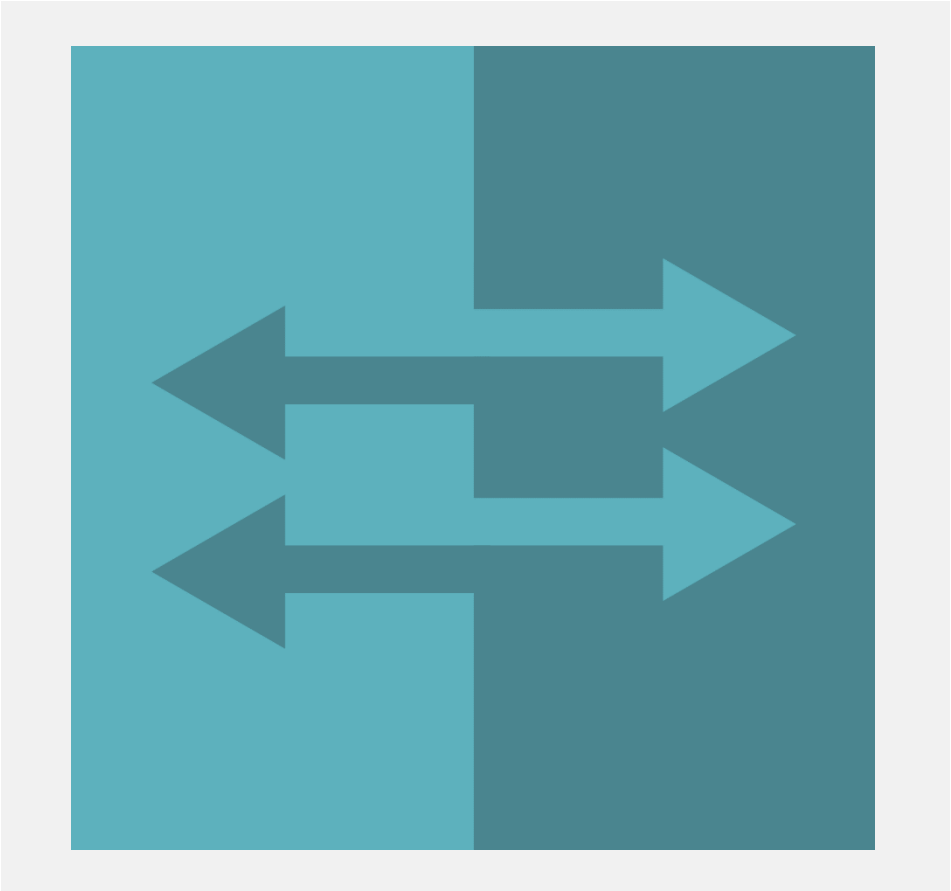
Error Queue

- Holds messages that can't be processed
- Keeps these message out of the way
- Optionally fix
- Optionally replay
- ServiceInsight or ServicePulse



Request/Response

- Send message and wait for response using queues
- Temporal coupling
- Look at alternatives



Demo: Retries and Request/Response

FIRE ON WHEELS

Configure fault tolerance

Get pricing info on the fly

Summary

NServiceBus enables messaging between applications and is highly configurable and extensible

Types of messages: commands, events, request/response

Fault tolerant