Introducing Cloud Message Queues

Elton Stoneman
geekswithblogs.net/eltonstoneman
@EltonStoneman

pluralsight
hardcore dev and IT training

Introducing Cloud Message Queues

Messaging over cloud services like **Azure** and **AWS**

High availability queues, easily accessible over **HTTPS**

Unlimited scale and **platform integration**

Goals

Cloud message queue overview

How it works, features AWS and Azure provide

Web portal administration

Creating, managing and securing queues

.NET client libraries

Features and usage

Pattern support

Fire-and-forget, request-response, publish-subscribe

What is Cloud Messaging?

y

3

Remote queues with **public endpoints**

Redundant storage and **high availability**

Connected queue technology

Cloud Message Queue Features

Security

- Public Internet endpoint
- Permissions-based security
- Integrates with provider security

Reliability

- Messages replicated
- Queues highly available
- Multiple instances and DCs

Cloud Message Queue Features

Cost

- Pay-per-use messaging
- Free for X messages per month
- Then $Y per next X messages

Limitations

- Queue size – X Gb
- Message size – Y Kb
- Queue removal – unused period

Demo 1: Administering Cloud Queues

Feature

Create and administer queues using Web portals

Task

Create and secure Azure Service Bus queue

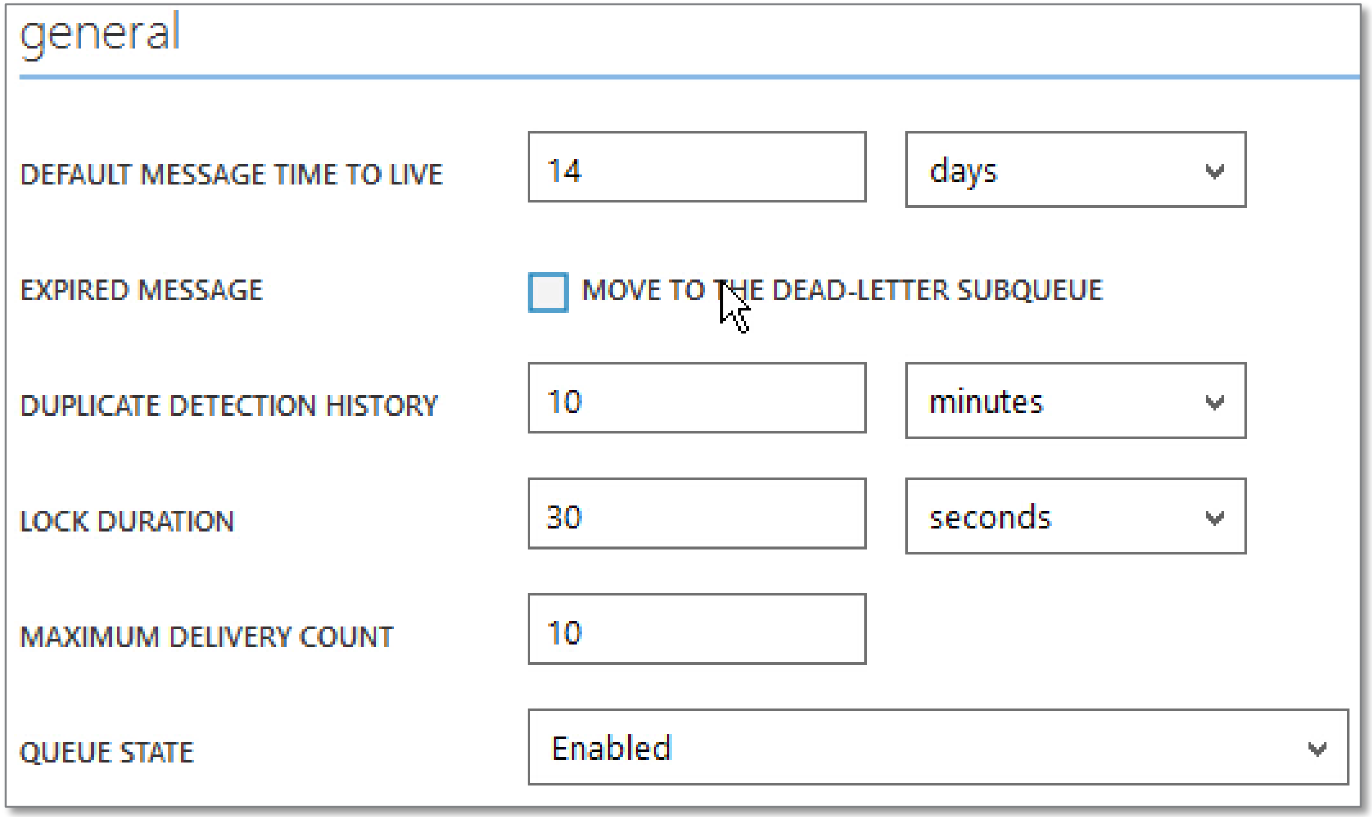
Task

Create Amazon Simple Queue Service queue

Demo 1: Administering Cloud Queues

Demo 1: Administering Cloud Queues

- **Azure Service Bus – queue settings**

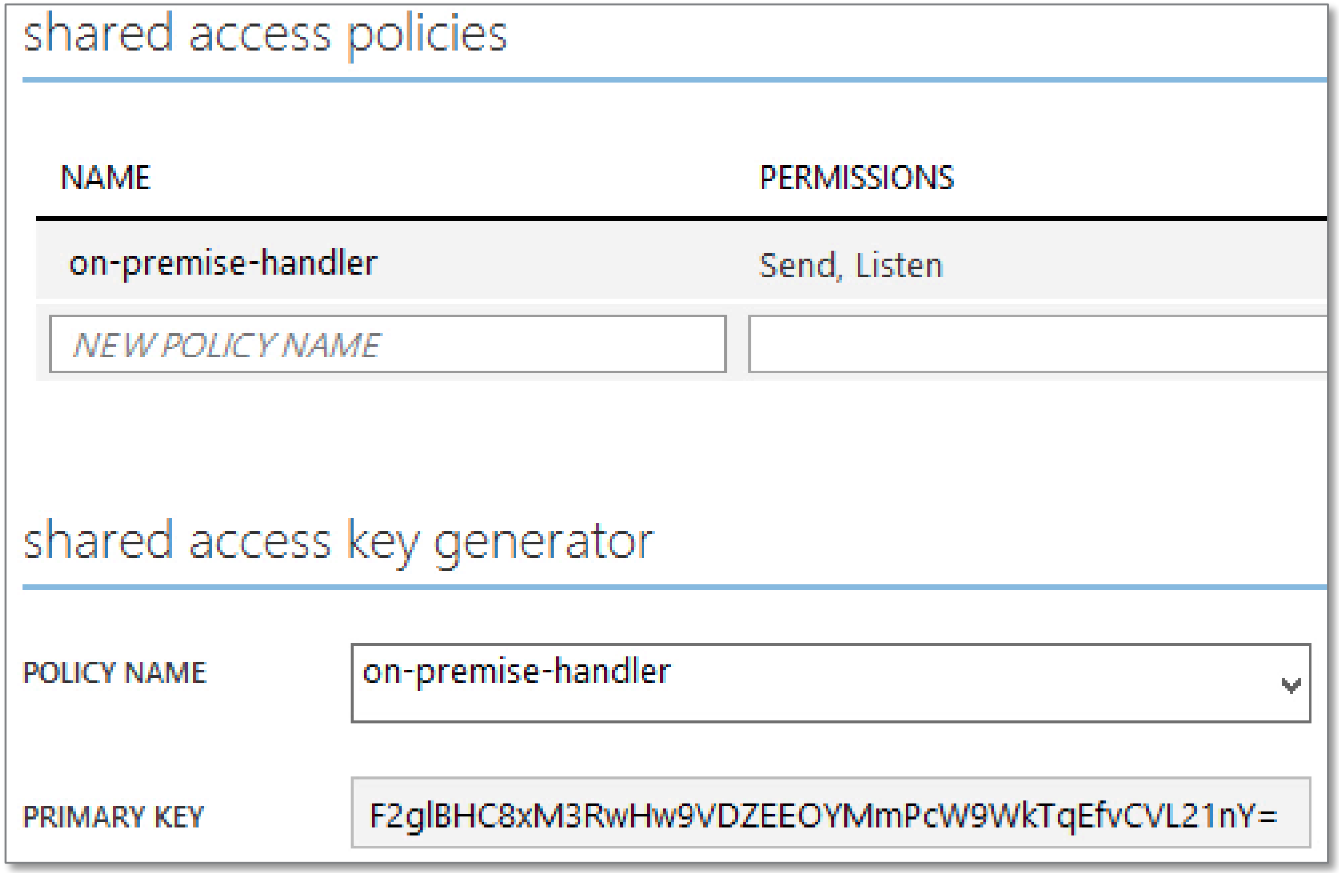


general

DEFAULT MESSAGE TIME TO LIVE	14	days
EXPIRED MESSAGE	☐ MOVE TO THE DEAD-LETTER SUBQUEUE	
DUPLICATE DETECTION HISTORY	10	minutes
LOCK DURATION	30	seconds
MAXIMUM DELIVERY COUNT	10	
QUEUE STATE	Enabled	

Demo 1: Administering Cloud Queues

- **Azure Service Bus – Shared Access Policies**



shared access policies

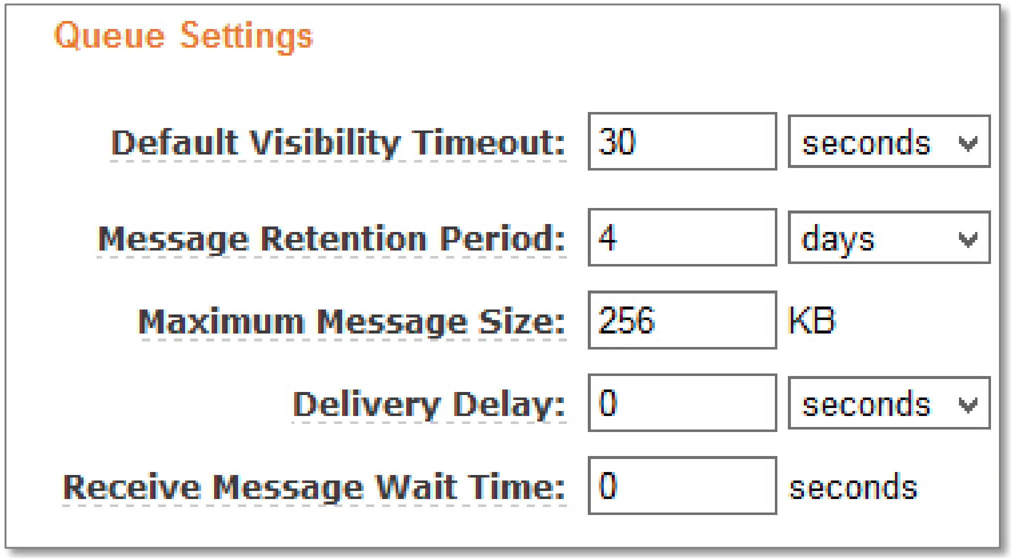
NAME	PERMISSIONS
on-premise-handler	Send, Listen
NEW POLICY NAME	

shared access key generator

POLICY NAME	on-premise-handler
PRIMARY KEY	F2glBHC8xM3RwHw9VDZEEOYMmPcW9WkTqEfvCVL21nY=

Demo 1: Administering Cloud Queues

- AWS Simple Queue Service – settings

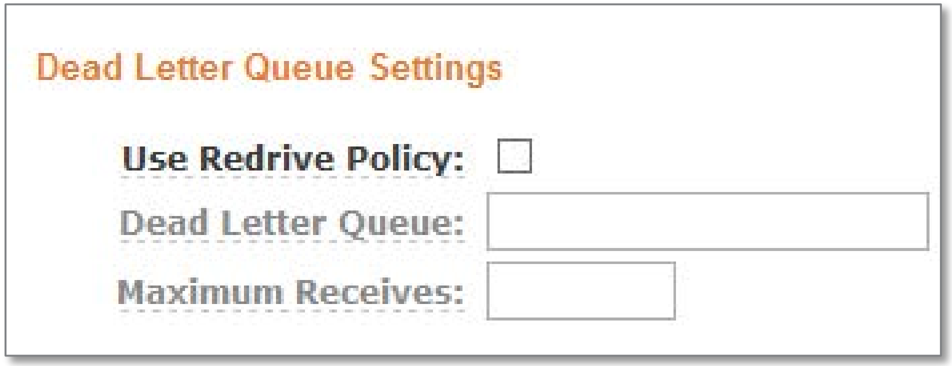


Queue Settings

Default Visibility Timeout:	30	seconds
Message Retention Period:	4	days
Maximum Message Size:	256	KB
Delivery Delay:	0	seconds
Receive Message Wait Time:	0	seconds

Demo 1: Administering Cloud Queues

- **AWS Simple Queue Service – settings**

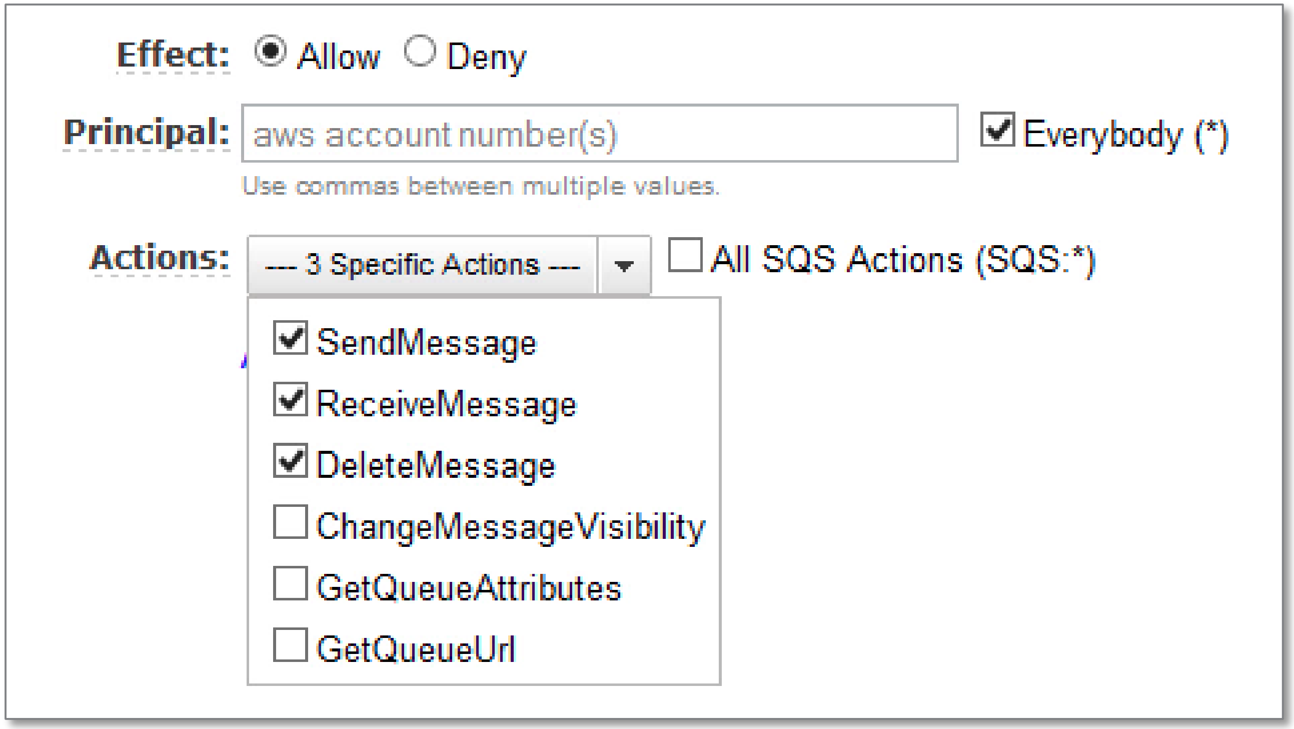


Dead Letter Queue Settings

Use Redrive Policy:	☐
Dead Letter Queue:	
Maximum Receives:	

Demo 1: Administering Cloud Queues

- **AWS Simple Queue Service – permissions**



Effect: [✓ Allow] Deny

Principal: aws account number(s) [✓ Everybody (*)]

Use commas between multiple values.

Actions: --- 3 Specific Actions --- ☐ All SQS Actions (SQS:*)

- ✓ SendMessage
- ✓ ReceiveMessage
- ✓ DeleteMessage
- ☐ ChangeMessageVisibility
- ☐ GetQueueAttributes
- ☐ GetQueueUrl

.NET Client Libraries

Client libraries targeting **multiple platforms** and **OS**

.NET primary client platform (also PHP, Java, Node etc.)

Java primary platform, **feature parity** in .NET (also PHP etc.)

BrokeredMessage

C

Microsoft.Service Bus.Messaging

Message envelope

Typed message body

Runtime properties

Serialization

XML with DataContractSerializer

Domain behaviour

GetBody<T>()

Complete()

QueueClient

C

Microsoft.Service Bus.Messaging

Messaging operations

Created from MessagingFactory

Context-bound to one queue

BrokeredMessage

Send() and Receive()

Message-pump pattern

Allows batch processing

SendBatch() and ReceiveBatch()

Async alternatives

Demo 2: Service Bus Queues with .NET

Feature

Explore messaging with Windows Azure

Task

Send and receive BrokeredMessage objects

Task

Check basic performance for multiple sends

Demo 2: Service Bus Queues with .NET

Demo 2: Service Bus Queues with .NET

- **Send messages**
  - Create queue client from factory

```
var factory = MessagingFactory.CreateFromConnectionString(
        "Endpoint=sb://sc-unsubscribe.servicebus.windows.net/;etc");
var queueClient = factory.CreateQueueClient("test");
```

  - Build and send brokered message

```
var message = new BrokeredMessage("message");
queueClient.Send(message);
```

Demo 2: Service Bus Queues with .NET

- **Receive messages**
  - Create queue client from factory

```
var factory = MessagingFactory.CreateFromConnectionString(
        "Endpoint=sb://sc-unsubscribe.servicebus.windows.net/;etc");
var queueClient = factory.CreateQueueClient("test");
```

  - Receive brokered message and extract body

```
var message = queueClient.Receive();
message.GetBody<string>().Dump("Body");
```

  - Flag as complete

```
message.Complete();
```

AmazonSqsClient

No message envelope

String message body

Manual serialization

SOA-style interface

Methods with Request/Response pairs

SendMessageRequest ->

SendMessageResponse

Receive behaviour

Async operations

Demo 3: AWS SQS with .NET

Feature

Explore messaging with Amazon Simple Queue Service

Task

Send and receive strings with AmazonSqsClient

Task

Check basic performance for multiple sends

Demo 3: AWS SQS with .NET

Demo 3: AWS SQS with .NET

- **Send messages**
  - Create queue client with credentials

```
varvar sqsClient = new AmazonSQSClient("AccessKey", "SecretKey",
                                                   RegionEndpoint.EUWest1);
```

  - Build message request

```
var request = new SendMessageRequest();
request.MessageBody = "message";
request.QueueUrl = "https://sqs.eu-west-1.amazonaws.com/etc";
```

  - Send message

```
var response = sqsClient.SendMessage(request);
```

Demo 3: AWS SQS with .NET

- **Receive messages**
  - Create queue client with credentials

```
varvar sqsClient = new AmazonSQSClient("AccessKey", "SecretKey",
                                                      RegionEndpoint.EUWest1);
```

  - Build message request

```
var request = new ReceiveMessageRequest();
request.QueueUrl = "https://sqs.eu-west-1.amazonaws.com/etc";
var response = sqsClient.ReceiveMessage(request);
```

  - Send message

```
var response = sqsClient.SendMessage(request);
```

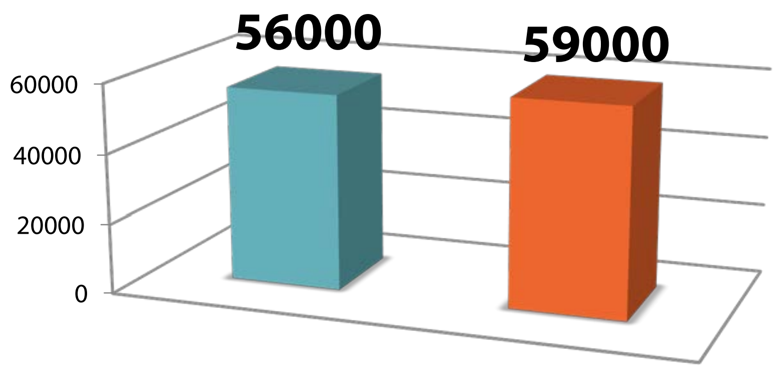
Demo 3: AWS SQS with .NET

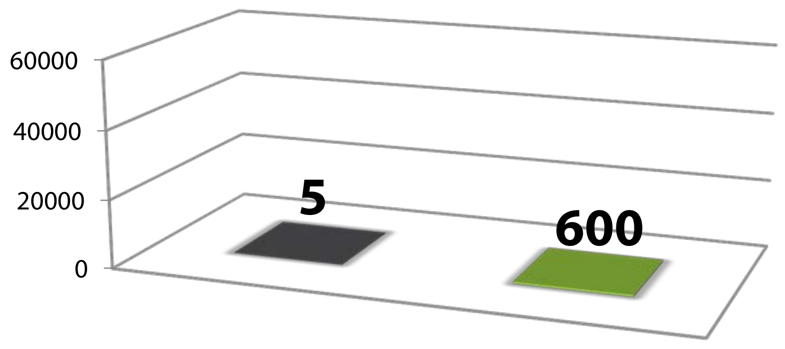
- **Delete received messages**
  - Confirms completion
  - Removes message from queue

```
var deleteRequest = new DeleteMessageRequest();
deleteRequest.QueueUrl = request.QueueUrl;
deleteRequest.ReceiptHandle = response.Messages[0].ReceiptHandle;

var deleteResponse = sqsClient.DeleteMessage(deleteRequest);
```

Cloud Message Queue Performance



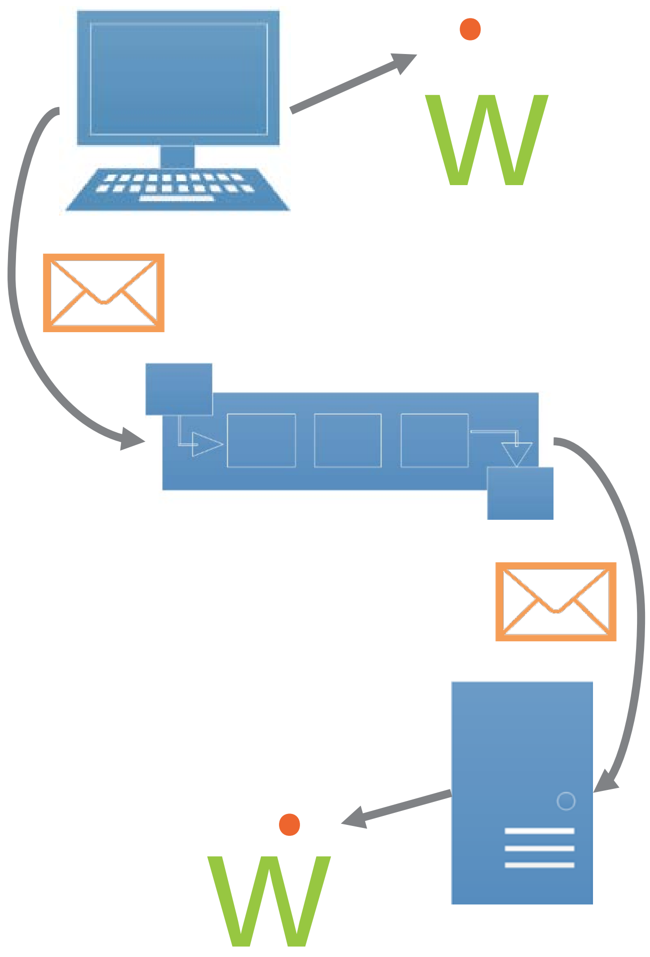


Azure and AWS
sub-minute

ZeroMQ and MSMQ
sub-second

▢ - milliseconds to send 1,000 messages

Messaging Pattern Support

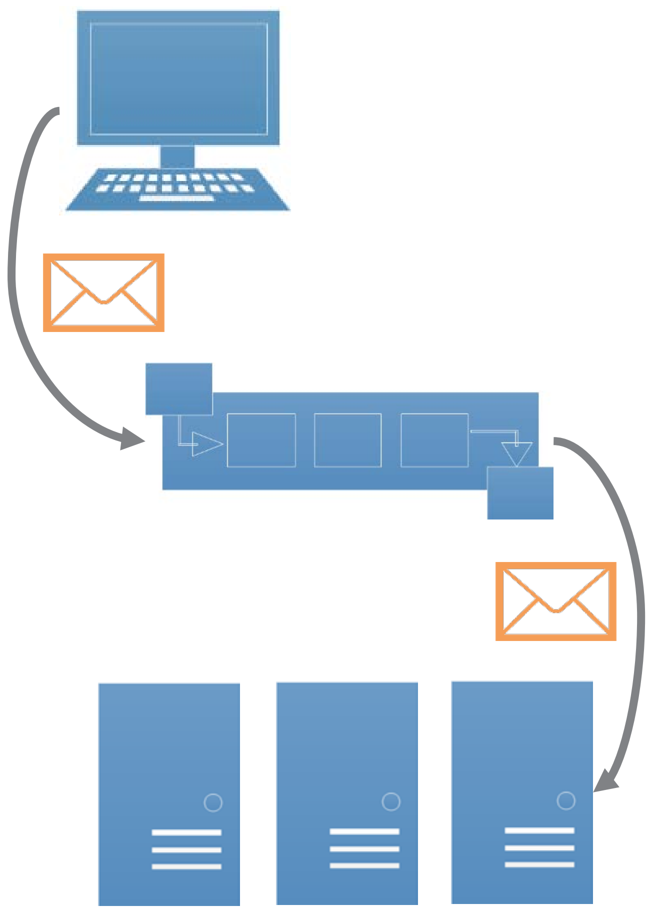


Fire-and-forget

Global access

Scaling to demand

Messaging Pattern Support

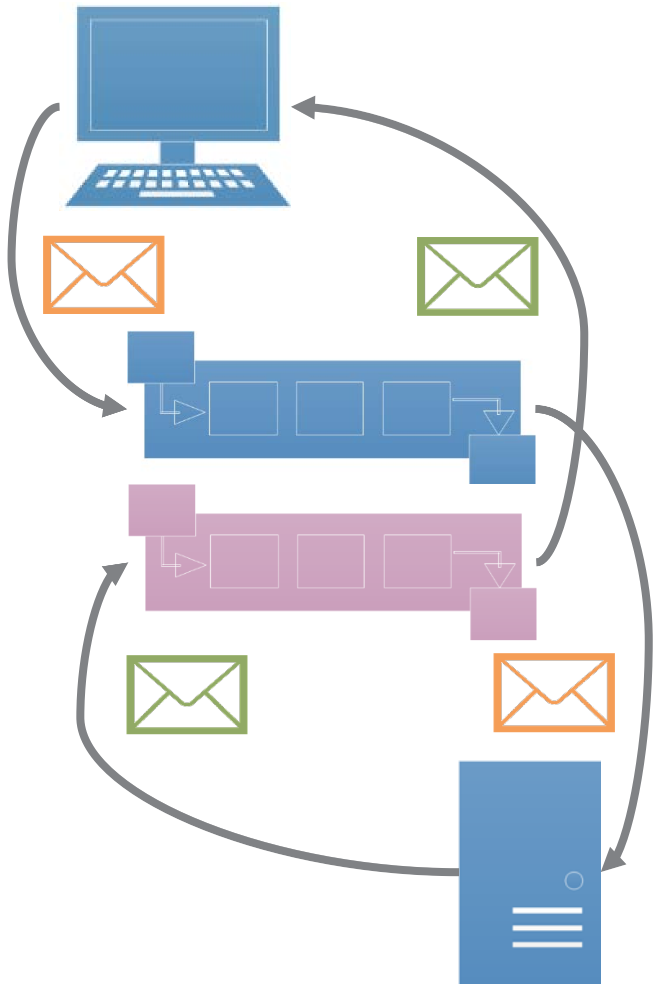


Fire-and-forget

Global access

Scaling to demand

Messaging Pattern Support



Fire-and-forget

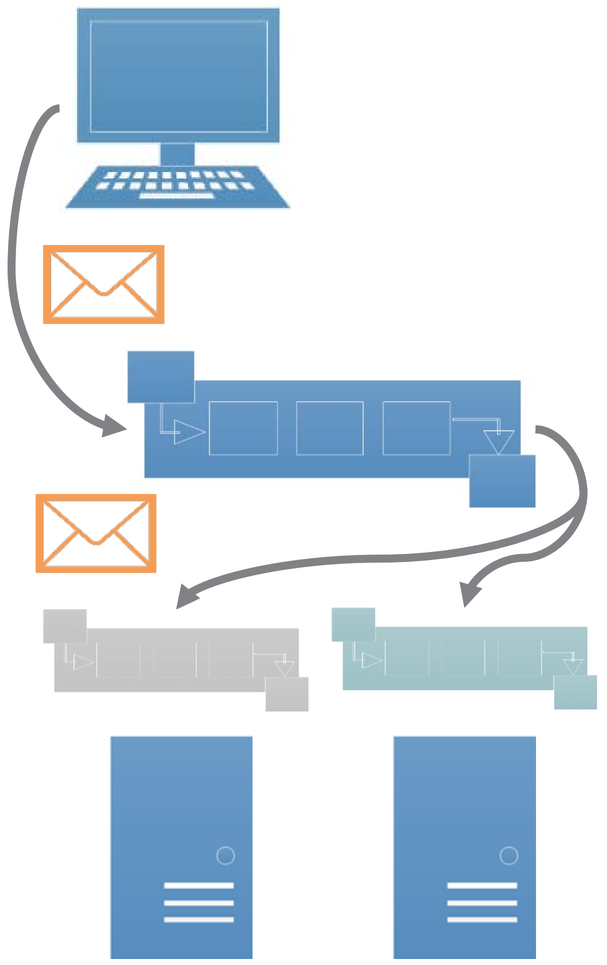
Global access

Scaling to demand

Request-response

Permanent/temporary response queue

Messaging Pattern Support



Fire-and-forget

Global access

Scaling to demand

Request-response

Permanent/temporary response queue

Publish-subscribe

Separate broadcast components

Topics coupled to queue/subscription

Summary

- **Introducing Cloud Messaging** ☑
  - Azure Service Bus Queues (and Topics)
  - AWS Simple Queue Service (and Simple Notification Service)
- **Feature set** ☑
  - Queue access permissions
  - Regional with HA across data centres
  - Limitations – message size & cost
- **Usage** ☑
  - Web administration
  - .NET client libraries

Cloud Message Queues (Azure & AWS)