Microsoft

Module 10: Develop event-based solutions

Microsoft

Topics

- Azure Event Grid
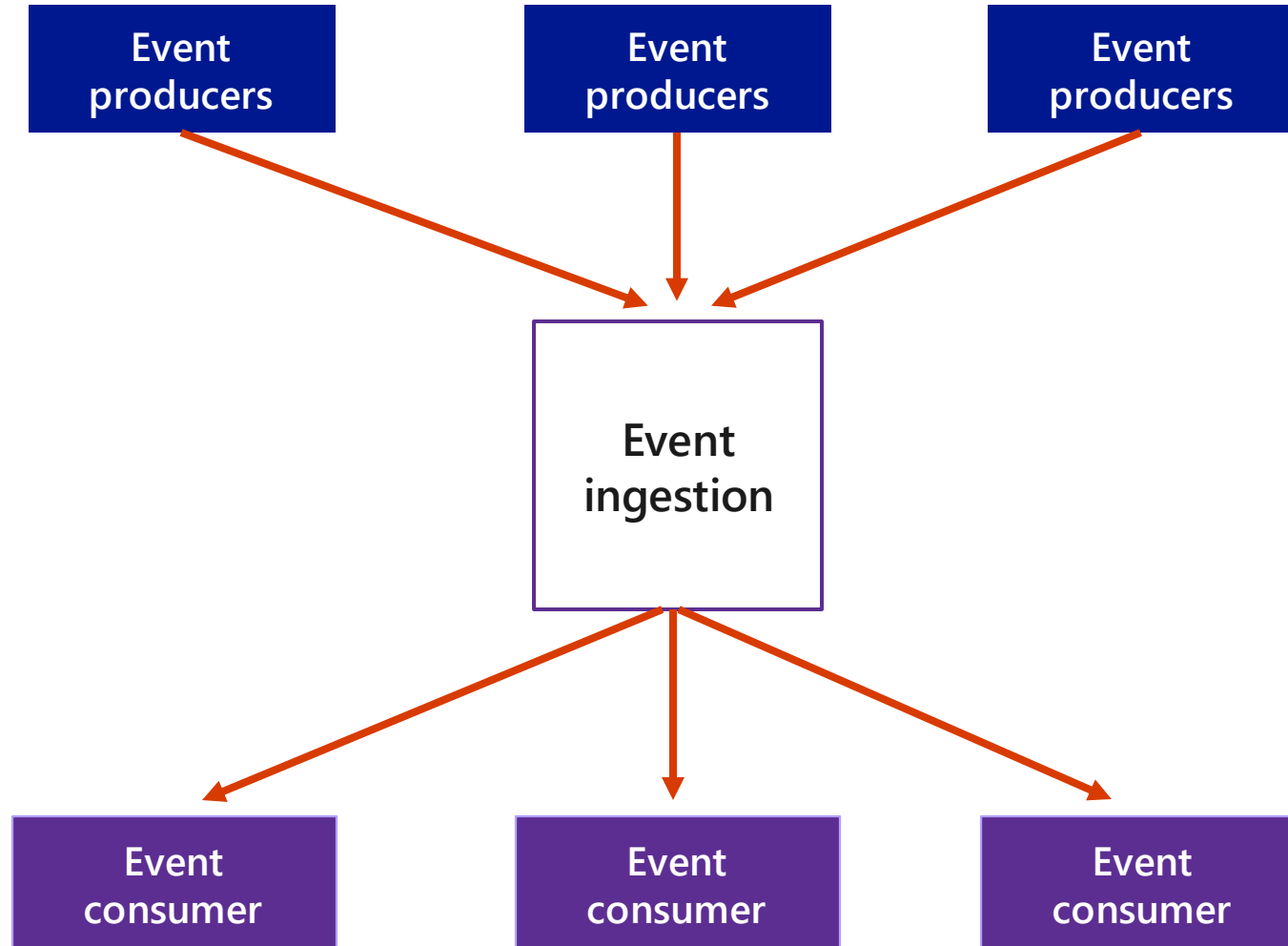- Azure Event Hubs
- Azure Notification Hubs

# Lesson 01: Azure Event Grid

Event-driven architecture

- Consists of event producers that generate a stream of events, and event consumers that listen for the events

- Events are delivered in nearly real time, so that consumers can respond immediately to events as they occur

- Common implementations include:

  - Single-event processing

  - Complex-event processing

  - Event-stream processing

Event-driven architecture (Continued)
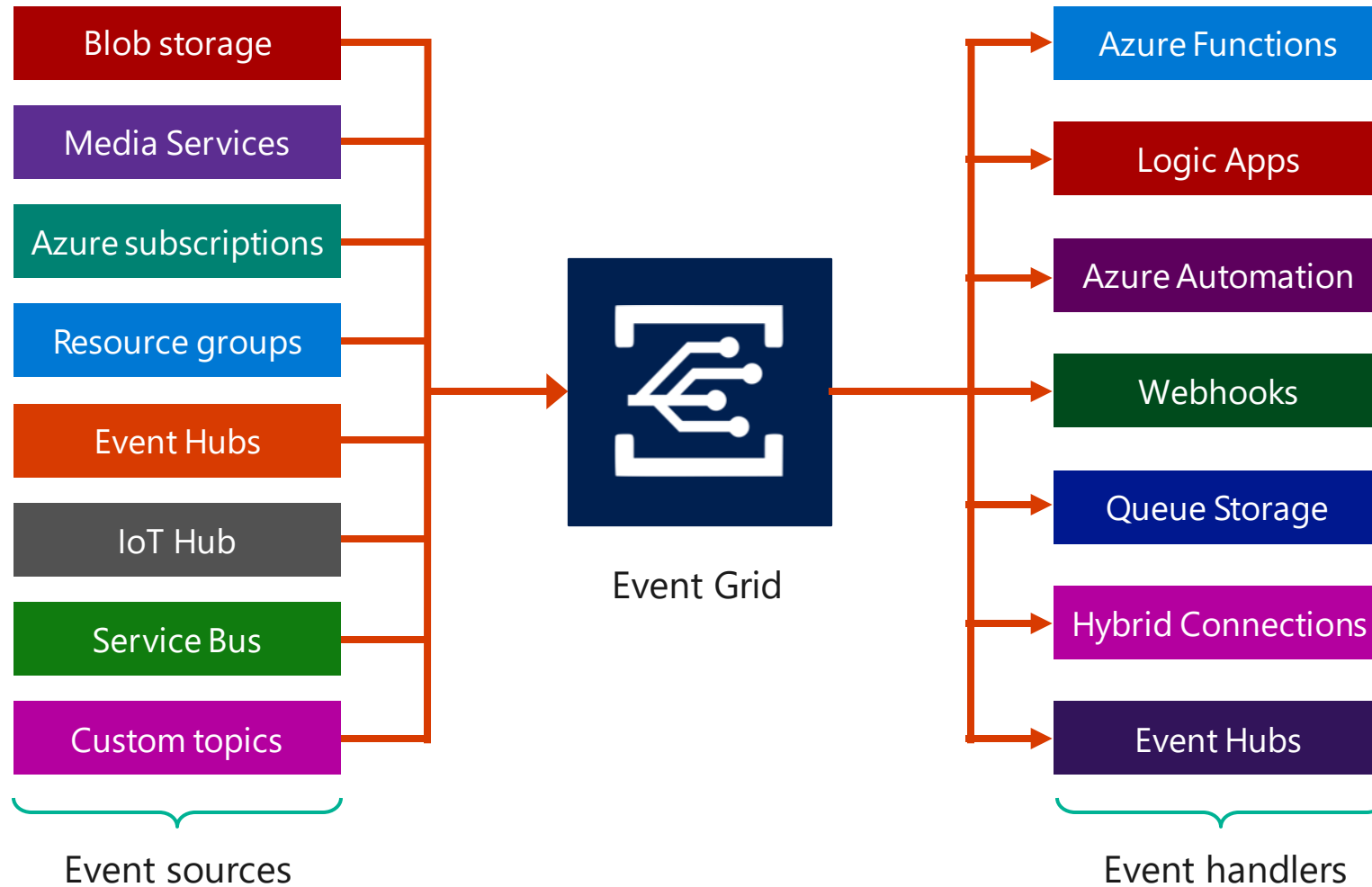
Azure Event Grid

- Allows you to easily build applications with event-based architectures in near-real fashion

- Fully managed event routing service using publish subscribe pattern

- Has built-in support for events coming from Azure services, ie Storage Blobs and Resource Groups

- Can be used for Non-Azur Events
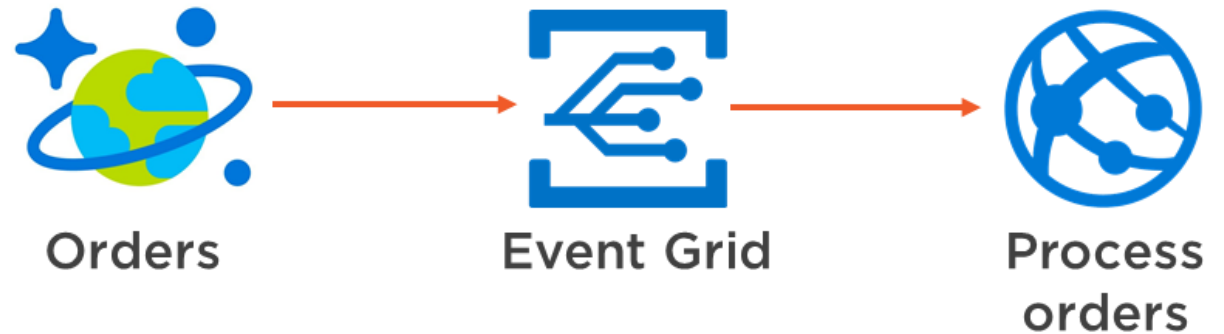
Event Grid concepts

- Events
  - What happened

- Event sources
  - Where the event took place

- Topics
  - The endpoint where publishers send events

- Event subscriptions
  - Subscriptions are also used by handlers to intelligently filter incoming events.

- Event handlers
  - The app or service reacting to the event

Sources and handlers

Event Grid concepts (cont.)
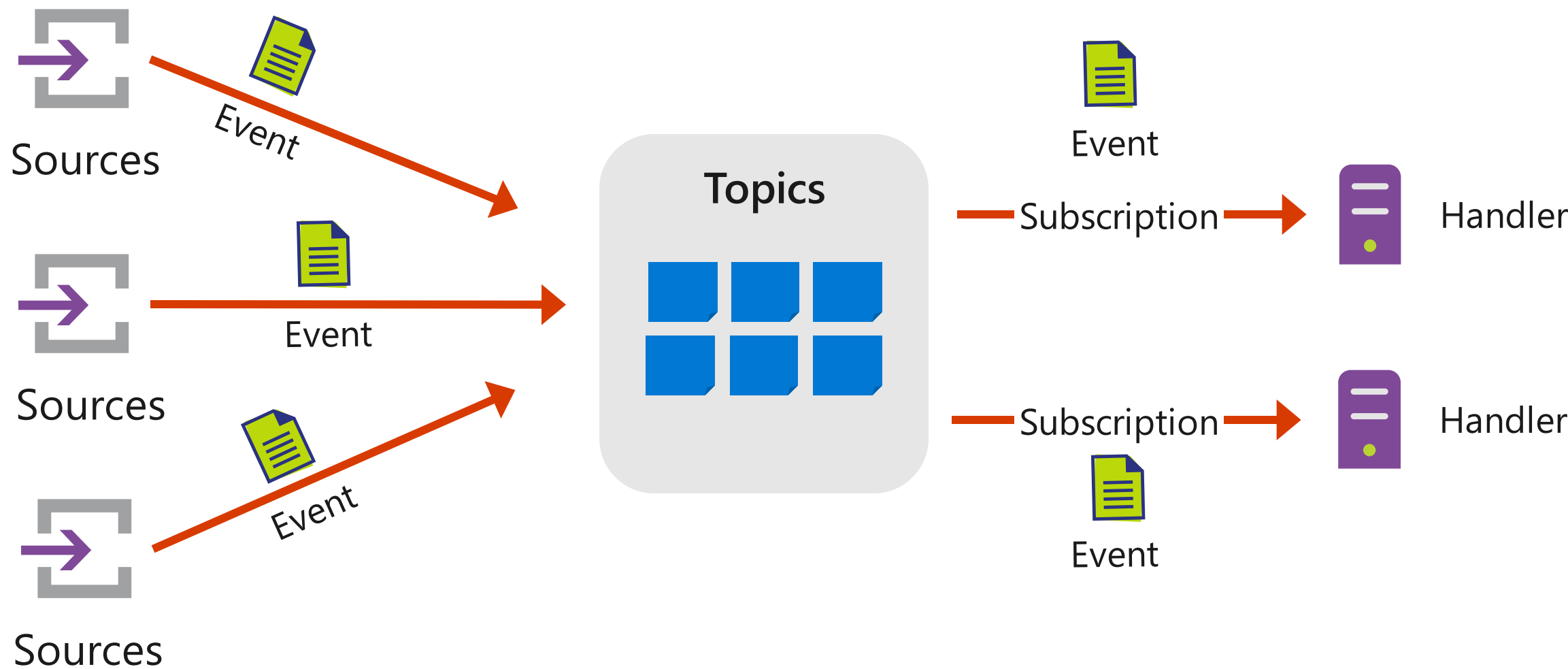
Schema

```json
[
    {
        "topic": string,
        "subject": string,
        "id": string,
        "eventType": string,
        "eventTime": string,
        "data":{
            object-unique-to-each-publisher
        },
        "dataVersion": string,
        "metadataVersion": string
    }
]
```

JSON

Schema – Azure Blob storage event

```
[{
    "topic": "...", "subject": "...",
    "eventType": "Microsoft.Storage.BlobCreated",
    "eventTime": "2017-06-26T18:41:00.9584103Z",
    "id": "831e1650-001e-001b-66ab-eeb76e069631",
    "data": {
        "api": "PutBlockList", "eTag": "0x8D4BCC2E4835CD0",
        "storageDiagnostics": { "batchId": "b68529f3-68cd-4744-baa4-3c0498ec19f0" },
        "clientRequestId": "6d79dbfb-0e37-4fc4-981f-442c9ca65760",
        "requestId": "831e1650-001e-001b-66ab-eeb76e000000",
        "contentType": "application/octet-stream", "contentLength": 524288,
        "blobType": "BlockBlob", "sequencer": "00000000000004420000000000028963",
        "url": "https://test.blob.core.windows.net/container/blob"
    },
    "dataVersion": "", "metadataVersion": "1"
}]
```

JSON

Schema – event properties

Property	Type	Description
topic	string	Full resource path to the event source. This field isn't writeable. Event Grid provides this value.
subject	string	Publisher-defined path to the event subject.
eventType	string	One of the registered event types for this event source.
eventTime	string	The time the event is generated based on the provider's UTC time.
id	string	Unique identifier for the event.
data	object	Event data specific to the resource provider.
dataVersion	string	The schema version of the data object. The publisher defines the schema version.
metadataVersion	string	The schema version of the event metadata. Event Grid defines the schema of the top-level properties. Event Grid provides this value.

Security

Three types of authentication:

- Webhook event delivery

  - Event Grid will send HTTP Post requests to an endpoint of your choice

  - Requires you prove ownership unless it's hosted in Azure Functions, Logic Apps, or Azure Automation

- Event subscriptions

  - Uses role-based access control (RBAC) permissions to ensure that you have access to the resources needed to subscribe to events

  - Must have **Microsoft.EventGrid/EventSubscriptions/Write** permission on the resource that is the event source

- Custom topic publishing

  - Uses SAS or key-based authentication

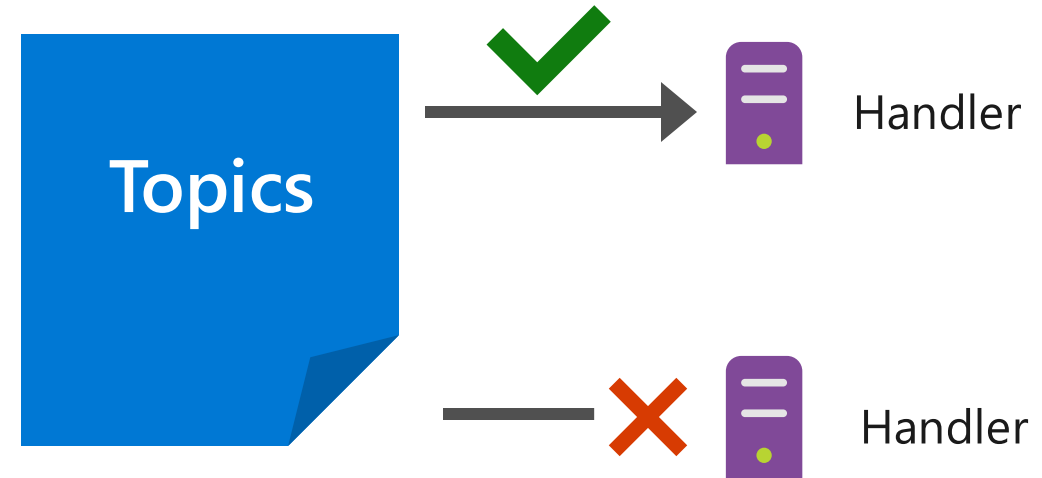# Filtering

Three options for filtering:

Event types

Subject begins with or ends with

Advanced fields and operators

Simplest filtering is by event type:

```
"filter": {
    "includedEventTypes": [
        "Microsoft.Resources.ResourceWriteFailure",
        "Microsoft.Resources.ResourceWriteSuccess"
    ]
}
```

Topics

Handler

Handler

Filtering (continued)

```json
// subject filter
"filter": {
    "subjectBeginsWith": "/blobServices/default/containers/mycontainer/log",
    "subjectEndsWith": ".jpg"
}
// advanced filter
"filter": {
    "advancedFilters": [
        {
            "operatorType": "NumberGreaterThanOrEquals",
            "key": "Data.Key1", "value": 5
        },
        {
            "operatorType": "StringContains",
            "key": "Subject", "values": ["container1", "container2"]
        }
    ]
}
```

Authoring custom events

```
# creates a resource group named gridResourceGroup in the westus2 location
az group create --name gridResourceGroup --location westus2

# register the provider and check the status
az provider register --namespace Microsoft.EventGrid
az provider show --namespace Microsoft.EventGrid --query "registrationState"

# replace <your-topic-name> with a unique name for the topic
topicname=<your-topic-name>

# create the custom topic
az eventgrid topic create --name $topicname -l westus2 -g gridResourceGroup
```
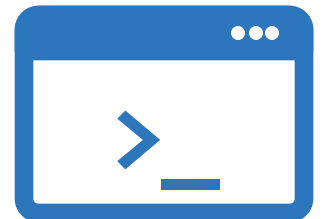
Authoring custom events (continued)

```
# replace <your-site-name> with a unique name for your web app.
sitename=<your-site-name>

# subscribe to the custom topic
az eventgrid event-subscription create \
    -g gridResourceGroup \
    --topic-name $topicname \
    --name demoViewerSub \
    --endpoint https://$sitename.azurewebsites.net/api/updates
```
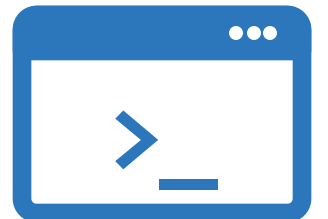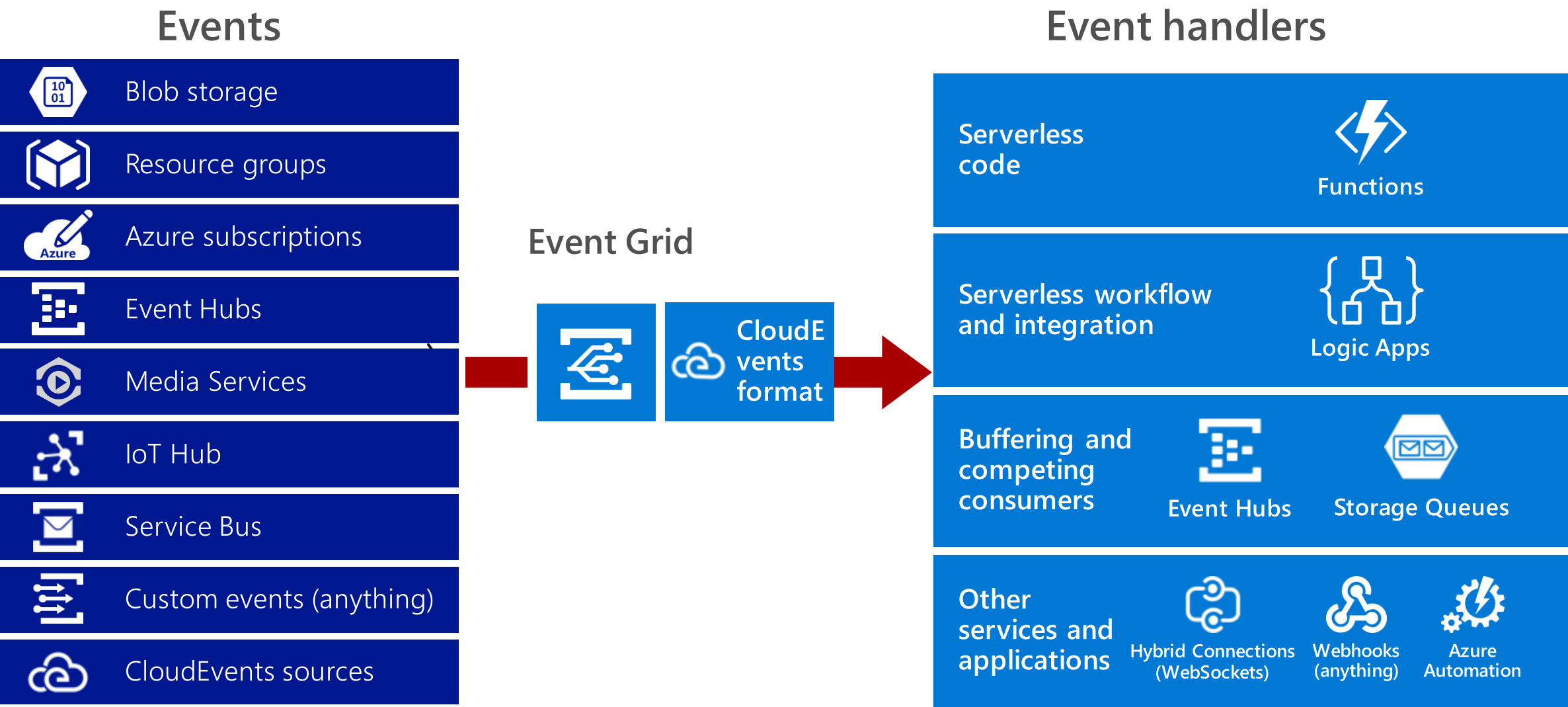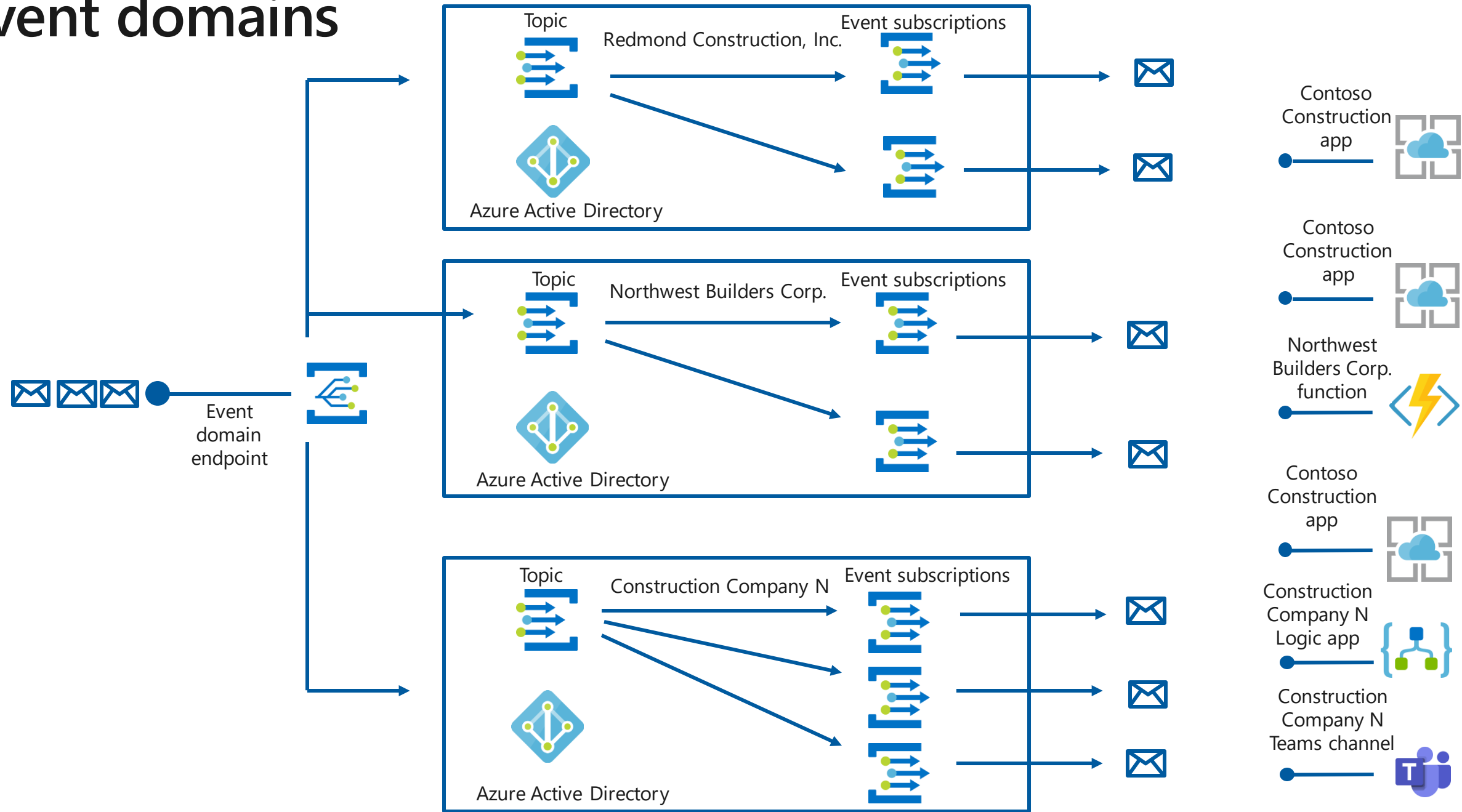
Event handling services

Events

	Blob storage
	Resource groups
	Azure subscriptions
	Event Hubs
	Media Services
	IoT Hub
	Service Bus
	Custom events (anything)
	CloudEvents sources

Event Grid

CloudEvents format

Event handlers

Serverless code
Functions

Serverless workflow and integration
Logic Apps

Buffering and competing consumers
Event Hubs   Storage Queues

Other services and applications
Hybrid Connections (WebSockets)   Webhooks (anything)   Azure Automation

Event domains

# Lesson 02: Azure Event Hubs

Azure Event Hubs

- Can process and store events, data, or telemetry produced by distributed software and devices

    - -> Event Hub is Big Data Event Streaming service

    - -> Scalable, Reliable

- Provide a distributed stream processing platform with low latency, and seamless integration with data and analytics services inside and outside of Azure

- Contain the following key components:

    - Event producers

    - Partitions

    - Consumer groups

    - Throughput units

    - Event receivers

Scenarios for Event Hubs

· Anomaly detection

· Application logging

· Analytics pipelines

· Live Dashboards
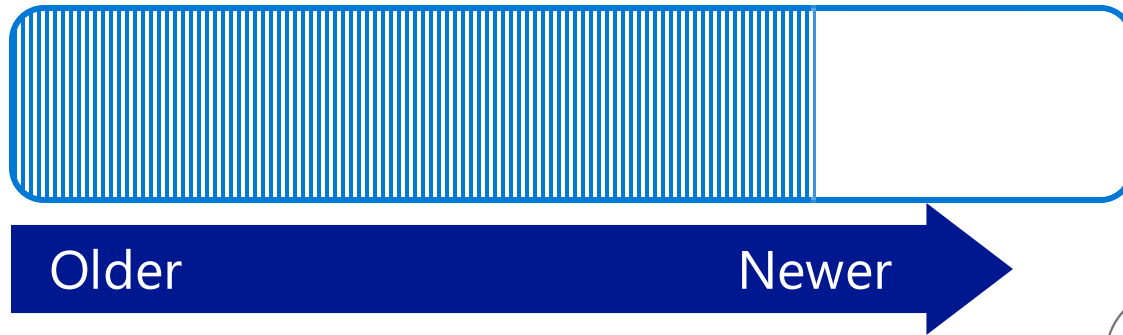
· Transaction, Telemetry processing

Event publishers
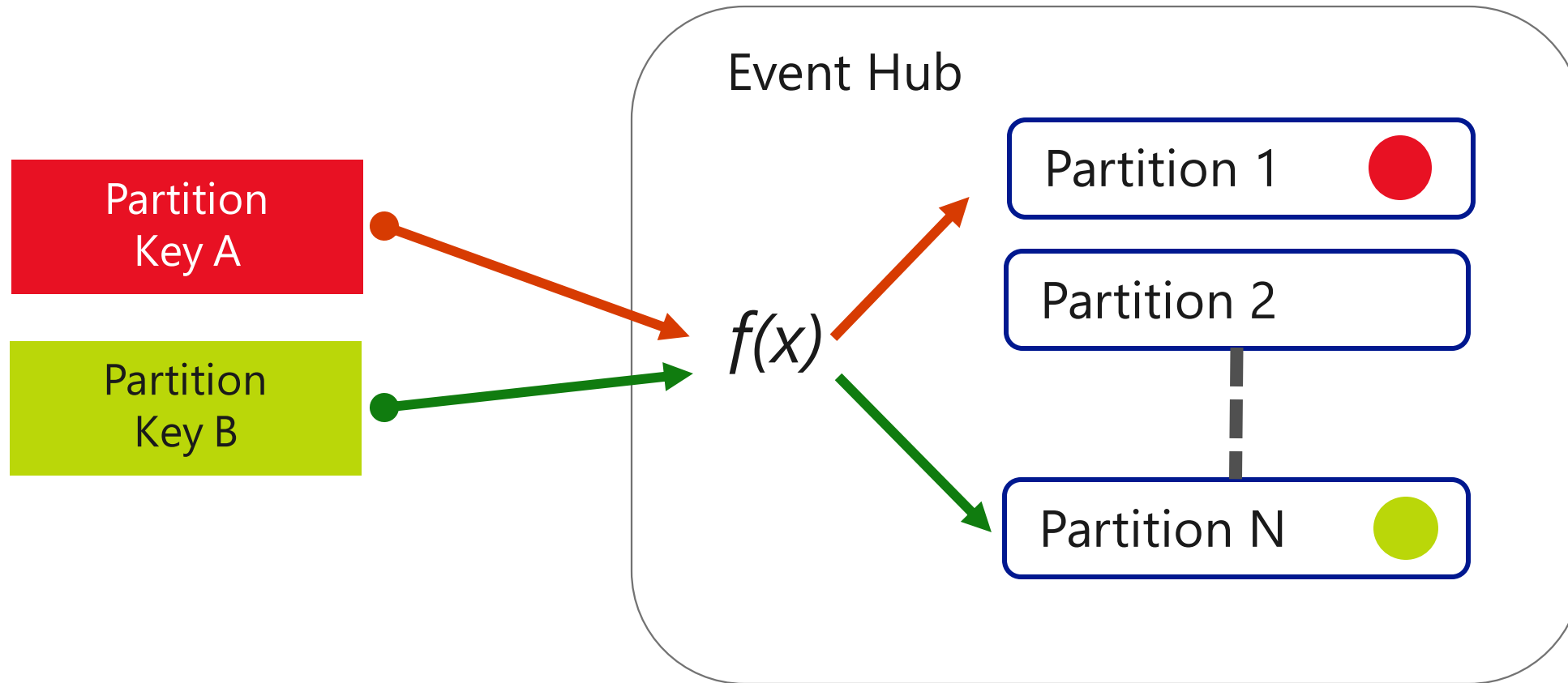
Eventhub Namespaces

- Event Hub represents unique stream of data

- Event Hub Namespace is a collectio of Event Hubs
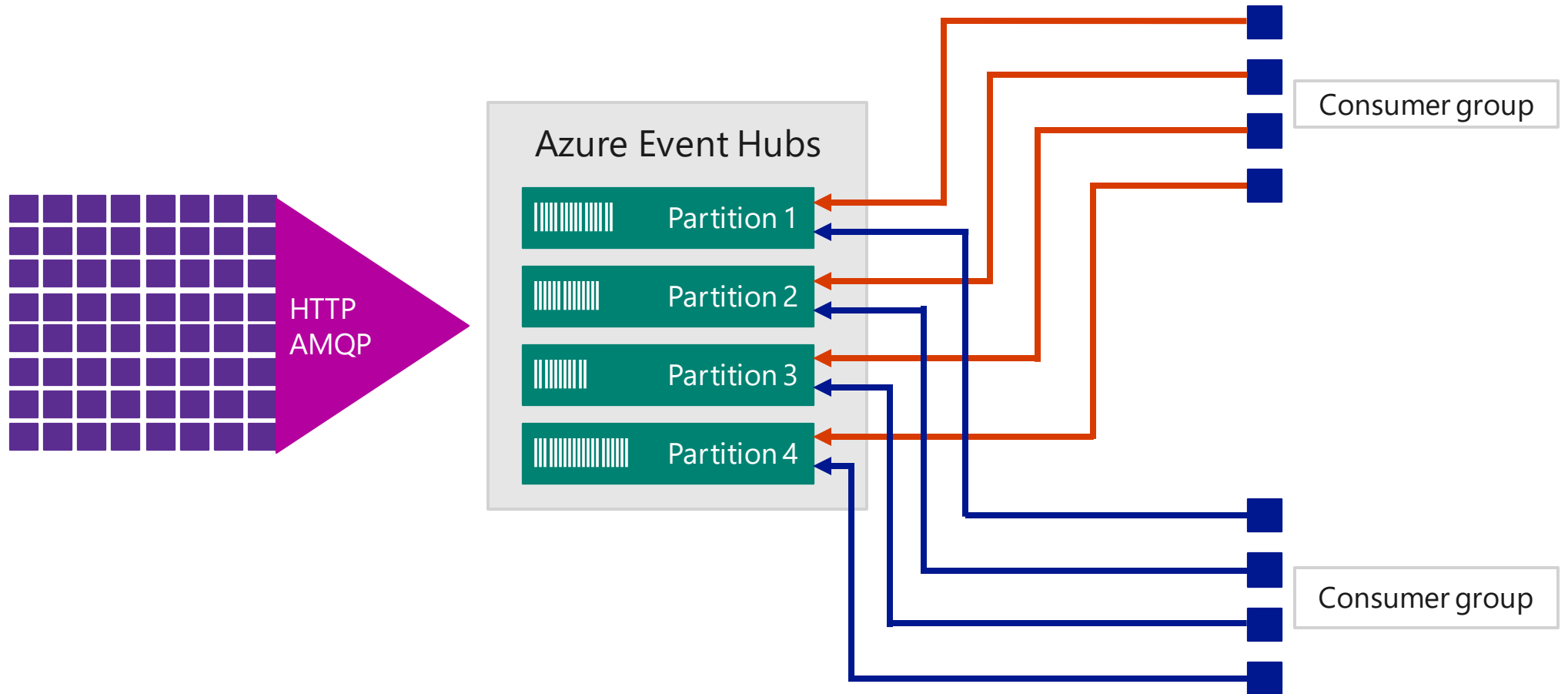   - Scoping Container
    - Shared Properties

Partitions

Older → Newer

A partition is an ordered sequence of events that is held in an event hub. As newer events arrive, they are added to the end of this sequence.

Event Hub

Partition 1

Partition 2

Partition 3
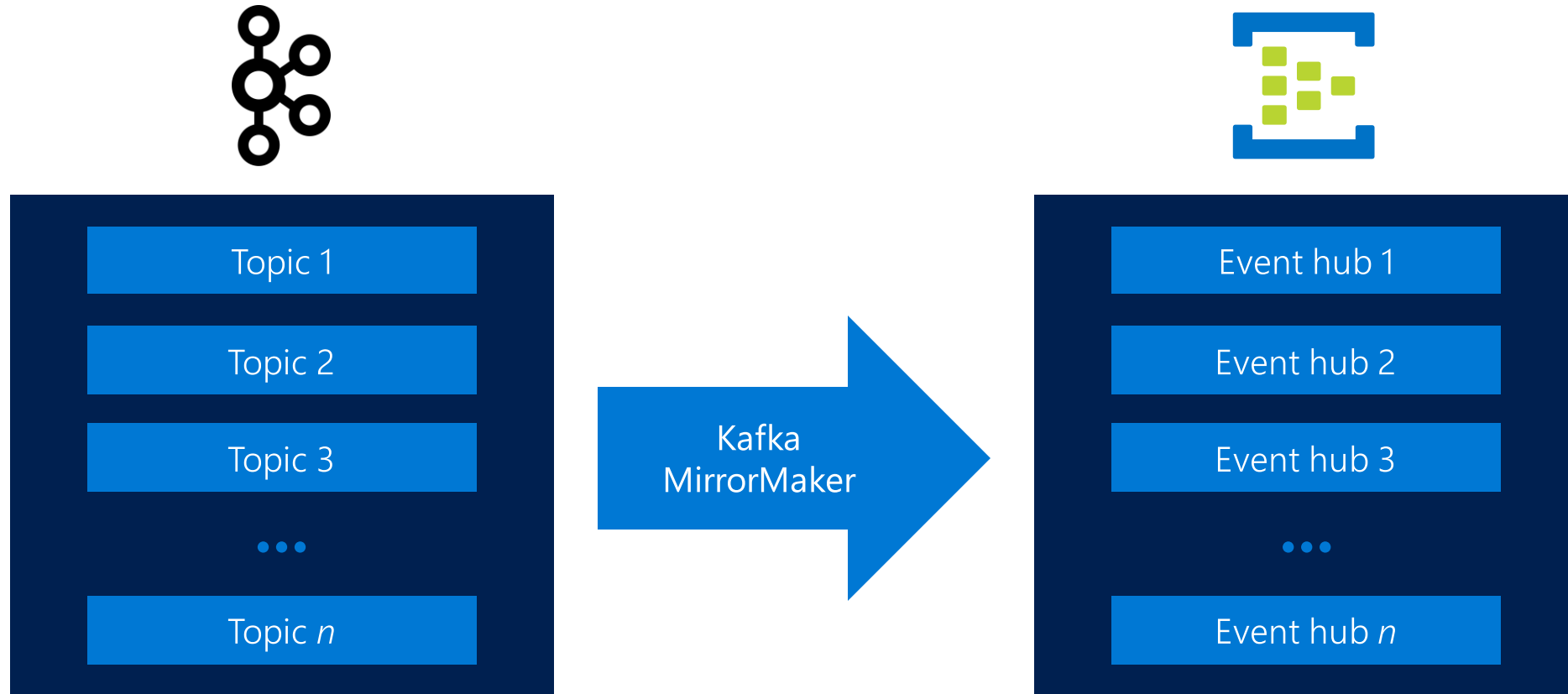
Partition N
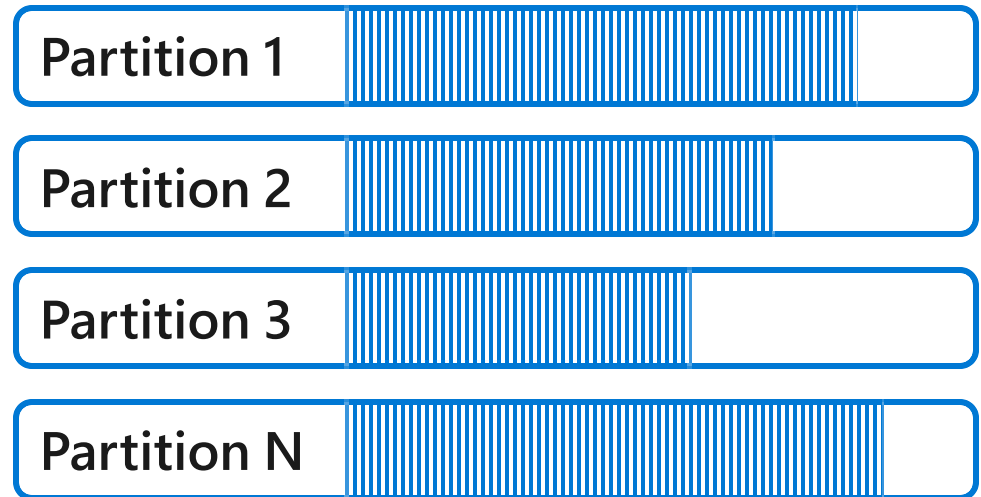
Consumer groups

Capture

- Data can be automatically captured

  - Stored in Azure Blob storage or Azure Data Lake Storage

  - Capture-time or size intervals can be specified

- You can specify a window to control capturing

  - Must specify a minimum size and time configuration

  - First trigger encountered causes a capture operation

- Data is stored by using a naming convention:

  `{Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}`

  `https://mystorageaccount.blob.core.windows.net/mycontainer/mynamespace/myeventhub/0/2017/12/08/03/03/17.avro`

Integration with Apache Kafka

Topic 1		Event hub 1
Topic 2	Kafka MirrorMaker	Event hub 2
Topic 3		Event hub 3
...		...
Topic n		Event hub n

Event Hubs and Apache Kafka mapping

Event Hubs	Kafka
Namespace	Cluster
Event Hub	Topic
Partition	Partition
Consumer Group	Consumer Group
Offset	Offset

Security model

- Only clients that present valid credentials can send data to an event hub

  - Uses a combination of SAS tokens and event publishers (virtual endpoint)

  - Publishers can send, but not receive, messages

- A client cannot impersonate another client

  - Each client has a unique token

- A rogue client can be blocked from sending data to an event hub

  - Clients don't have access to the signing key, thus preventing impersonation of another client

Creating a namespace manager by using the root key

```csharp
// Create namespace manager.
string serviceNamespace = "YOUR_NAMESPACE";
string namespaceManageKeyName = "RootManageSharedAccessKey";
string namespaceManageKey = "YOUR_ROOT_MANAGE_SHARED_ACCESS_KEY";

Uri uri = ServiceBusEnvironment.CreateServiceUri("sb", serviceNamespace, string.Empty);
TokenProvider td = TokenProvider.CreateSharedAccessSignatureTokenProvider(
    namespaceManageKeyName, namespaceManageKey
);

NamespaceManager nm = new NamespaceManager(namespaceUri, namespaceManageTokenProvider);
```

C#

Creating a SAS key

```csharp
// Create event hub with a SAS rule that enables sending to that event hub
EventHubDescription ed = new EventHubDescription("MY_EVENT_HUB") { PartitionCount = 32
};

string eventHubSendKeyName = "EventHubSendKey";
string eventHubSendKey = SharedAccessAuthorizationRule.GenerateRandomKey();

SharedAccessAuthorizationRule eventHubSendRule = new SharedAccessAuthorizationRule(
    eventHubSendKeyName, eventHubSendKey, new[] { AccessRights.Send }
);

ed.Authorization.Add(eventHubSendRule);
nm.CreateEventHub(ed);
```

C#

Creating Event Hubs by using the Azure CLI

```
# create a resource group
az group create --name <resource group name> --location eastus

# create an Event Hubs namespace
az eventhubs namespace create --name <Event Hubs namespace> --resource-group <resource
group name> -l <region, for example: East US>

# create an Event Hub
az eventhubs eventhub create --name <event hub name> --resource-group <resource group
name> --namespace-name <Event Hubs namespace>
```
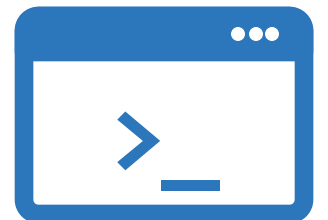
# Lesson 03: Azure Notification Hubs

Notification Hubs

- Automates many of the manual tasks to work with Push Notification Services

- Can tag recipients to send messages only to specific people

- Can create templates to automate converting a standard message into each specific application platform's XML or JSON

- Can send notifications from backend code to hub and then eventually to client devices

Push notifications

- A form of app-to-user communication where users of mobile apps are notified of certain desired information

- Delivered through platform-specific infrastructures called Platform Notification Systems (PNSs)

- Examples include:

  - Apple Push Notification Service (APNs)

  - Firebase Cloud Messaging (FCM)

  - Windows Push Notification Service (WNS)

Push notification service
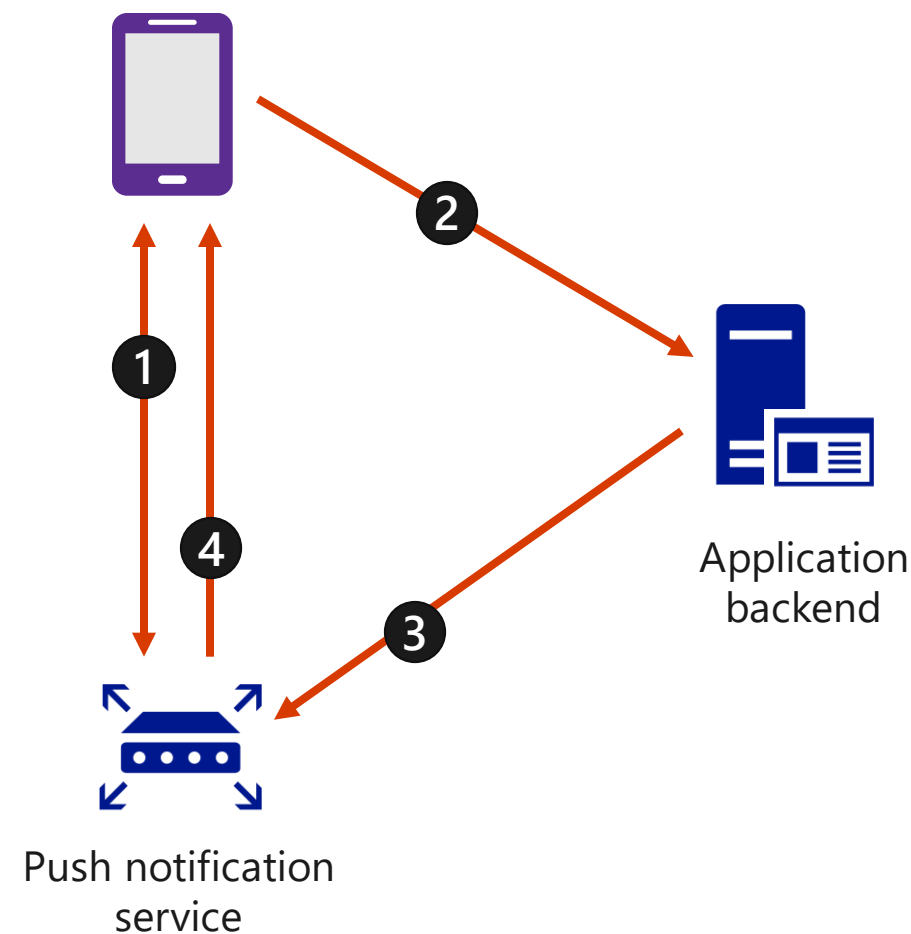
Notification Hubs example
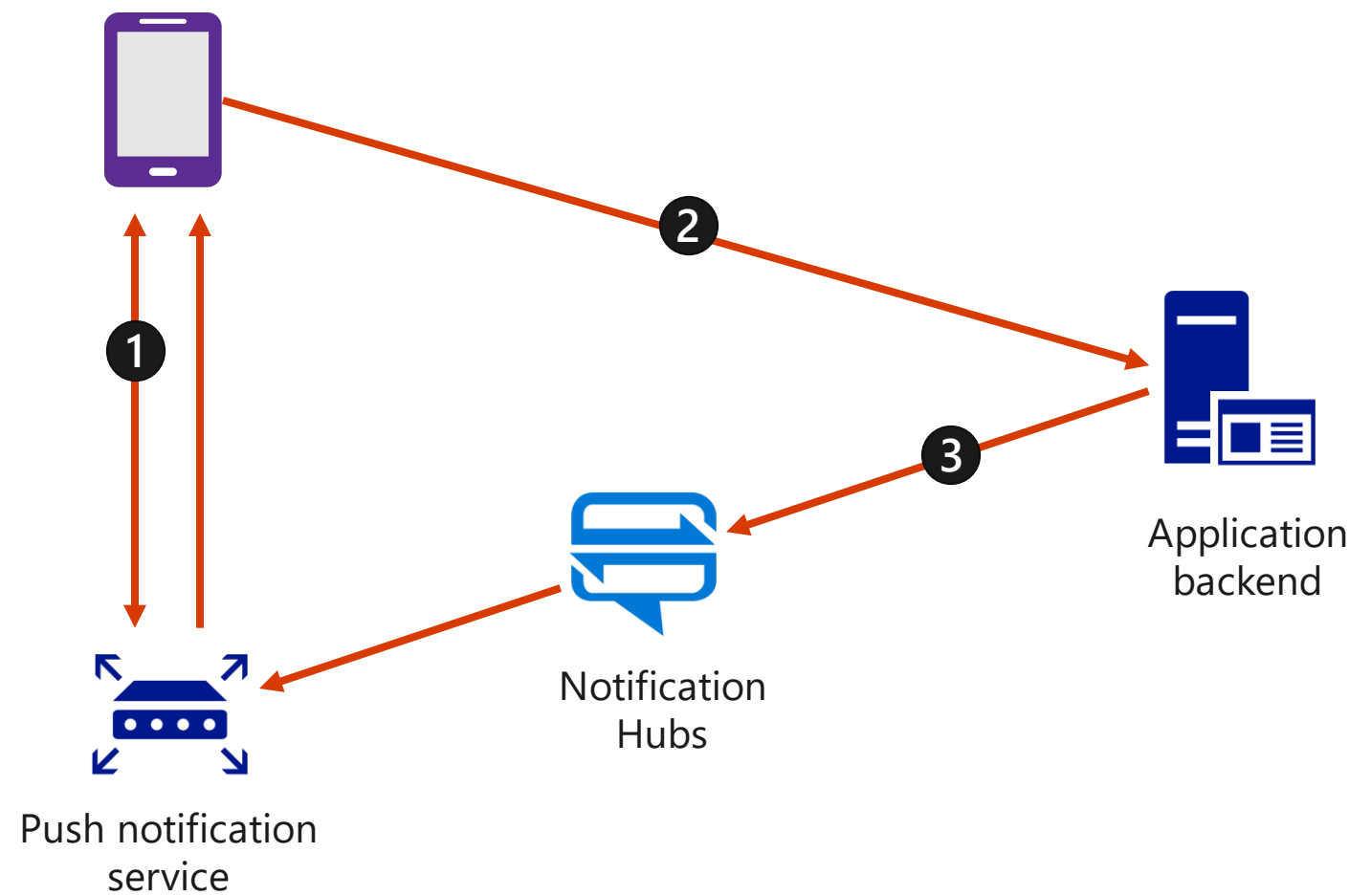
Shared Access Signature (SAS) security

- Messaging entities can declare up to 12 authorization rules in their description that grant rights to the entity

- Each rule contains a name, key value (shared secret), and a set of rights

- Use the key value (shared secret) to access the Notification Hub with least-permissive access

Claim	Description	Operations allowed
Listen	Create/Update, Read, and Delete single registrations	• Create/Update registration • Read registration • Read all registrations for a handle • Delete registration
Send	Send messages to the notification hub	• Send message
Manage	CRUDs on Notification Hubs (including updating PNS credentials, and security keys), and read registrations based on tags	• Create/Read/Update/Delete Notification Hubs • Read registrations by tag

Device registration

- Device registration with a Notification Hub is accomplished by using:

  - Registration

    - Associates the PNS handle for a device directly with tags and a template

    - PNS handle could be a ChannelURI, device token, or Google Cloud Messaging (GCM) registration id

    - Messages are routed to specific devices by using tags

  - Installation

    - Advanced registration that includes a bag of push properties

    - Fully idempotent, so you can retry multiple times without creating duplicate registrations

    - Each installation has a unique ID that can be used to push a message to the specific device

    - You can perform a partial registration update if you want to change some properties of the existing registration without having to rebuild the entire registration

Registration templates

- Templates enable a client application to specify the exact format and content of the notifications it wants to receive

- Templates allow you to create a single message and have it transformed by Notification Hubs to create:

  - A platform-agnostic backend

  - Notifications personalized by device, location, or user

  - Client-version independence

  - Localization of text content

Notification payload example

```
// send Hello! Using APNS (Apple)
{"aps": {"alert" : "Hello!" }}

// send Hello! Using MPNS (Windows)
<toast>
    <visual>
        <binding template=\"ToastText01\">
            <text id=\"1\">Hello!</text>
        </binding>
    </visual>
</toast>
```
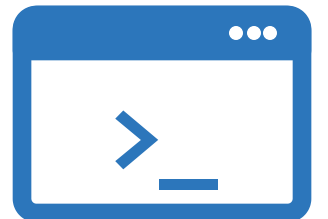
Registration template example

```
// platform-independent message
message = Hello!

// APNS template
{"aps": {"alert": "$(message)"}}

// MPNS template
<toast>
    <visual>
        <binding template=\"ToastText01\">
            <text id=\"1\">$(message)</text>
        </binding>
    </visual>
</toast>
```
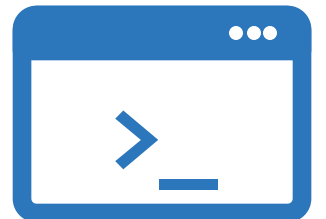
Template expression language

Expression	Description
$(prop)	Reference to an event property with the given name
$(prop, n)	As above, but the text is explicitly clipped at **n** characters
.(prop, n)	As above, but the text is suffixed with three dots as it is clipped
%(prop)	Similar to **$(name)** except that the output is URI-encoded
#(prop)	Used in JSON templates (for example, for iOS and Android templates)
'text' or "text"	A literal. Literals contain arbitrary text enclosed in single or double quotes
expr1 + expr2	The concatenation operator joining two expressions into a single string

Lab: Publishing and subscribing to Event Grid events

Duration

45 minutes

Lab sign-in information

AZ204-SEA-DEV

Username: Admin

Password: Pa55w.rd