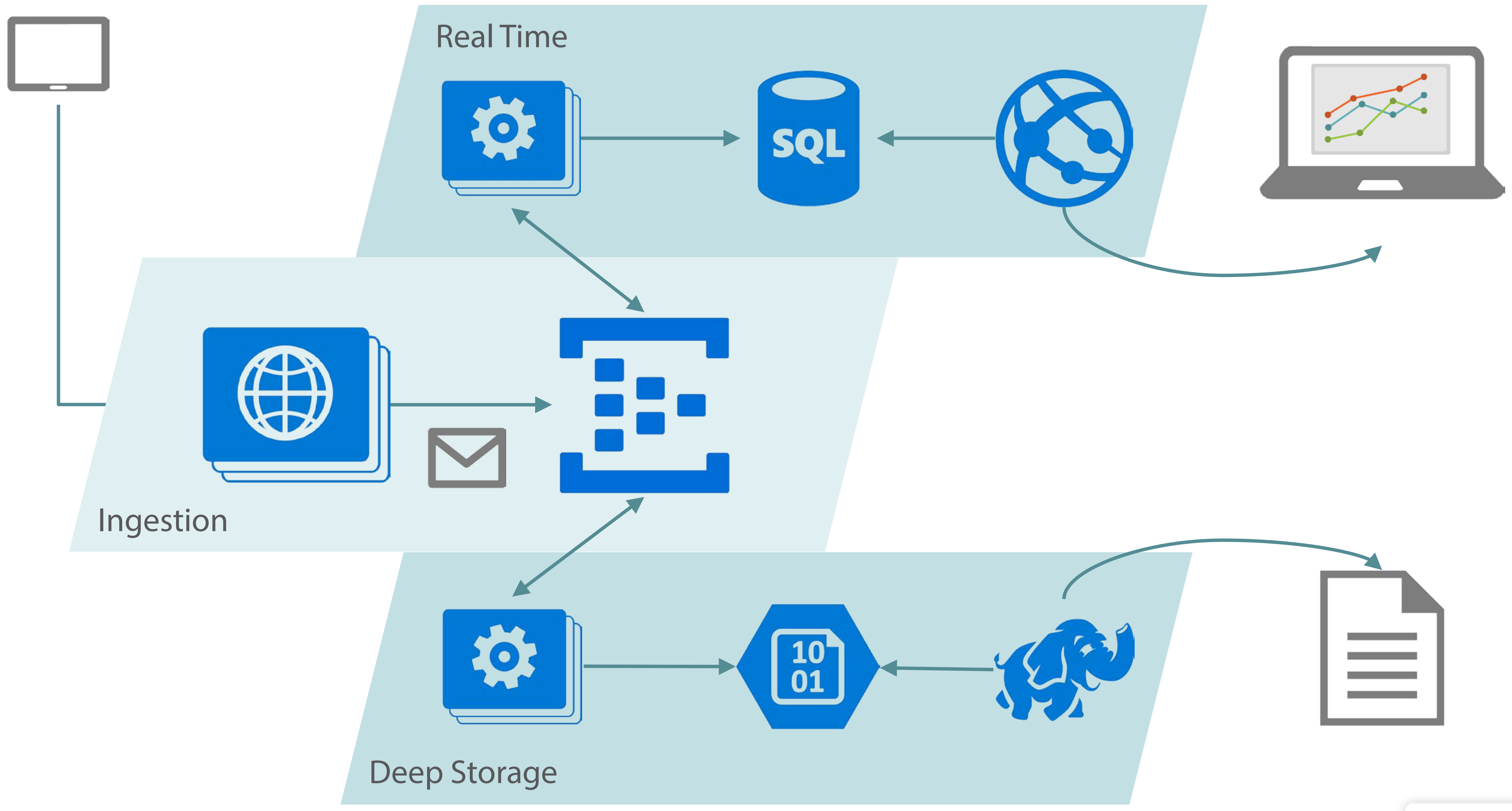
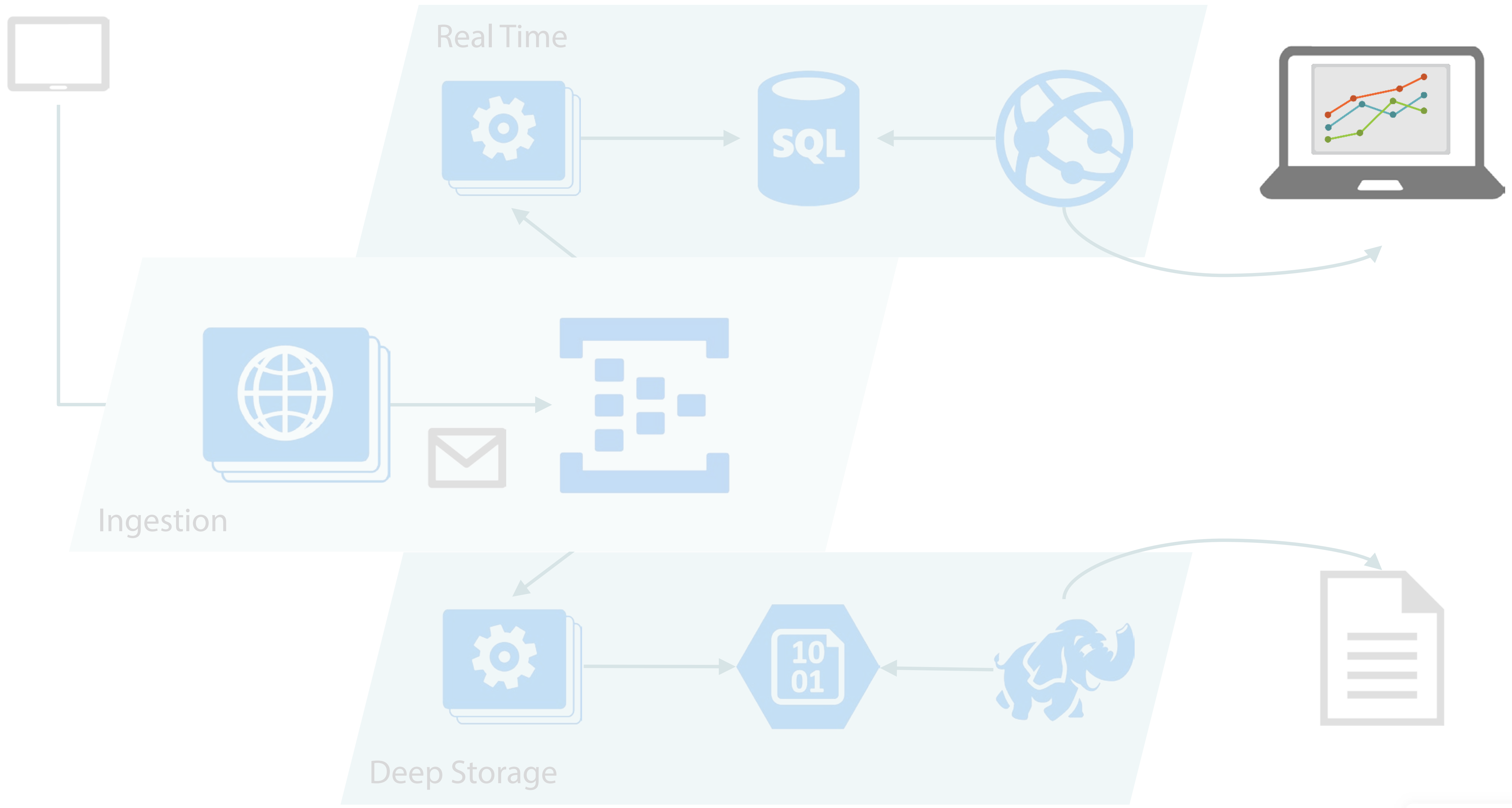
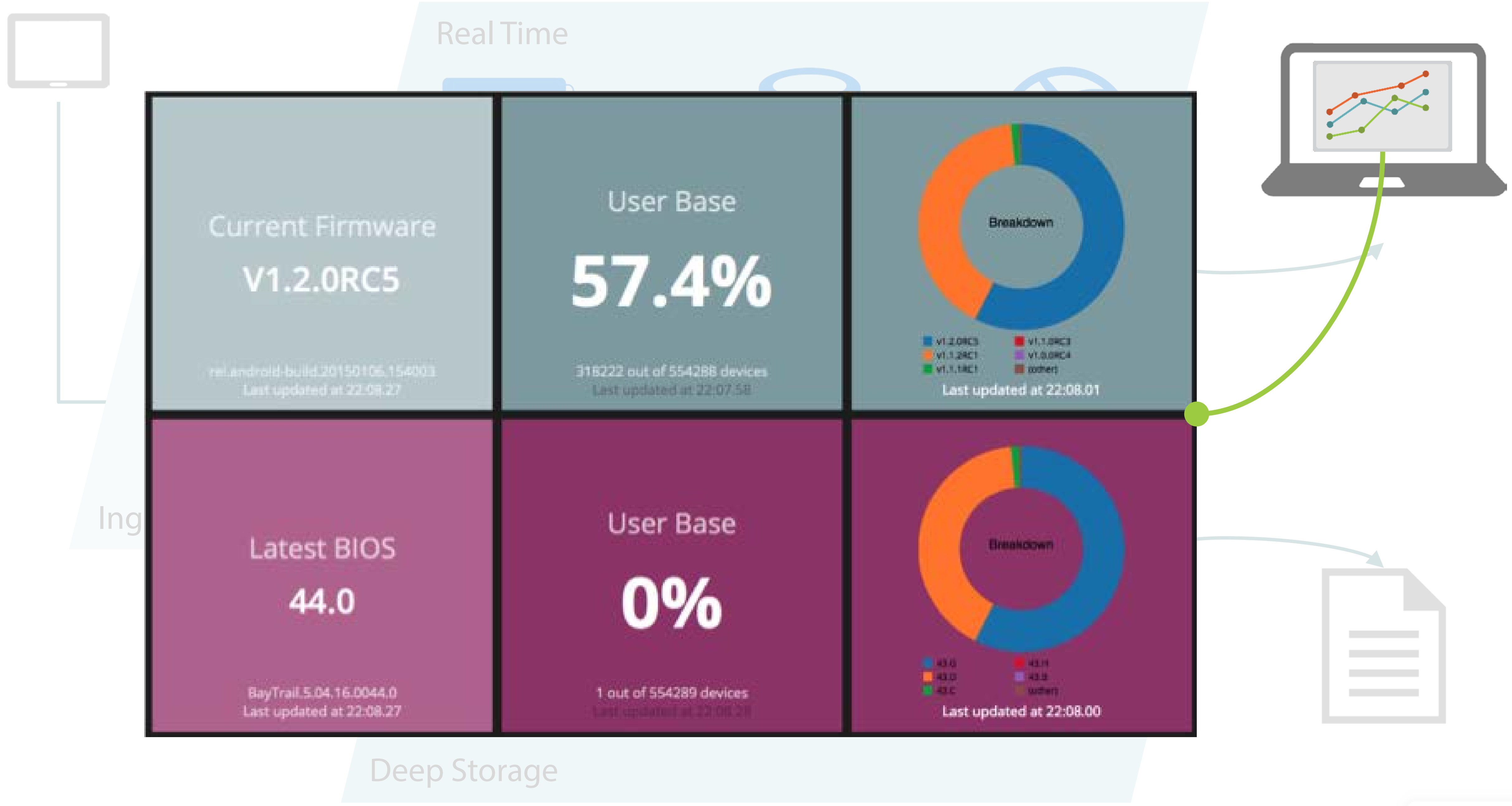
Storing Event Data for Batch Queries

Elton Stoneman

@EltonStoneman | www.geekswithblogs.net/eltonstoneman


Real Time
SQL
Ingestion
Deep Storage
10
01


Real Time
SQL
Ingestion
10
01
Deep Storage



Current Firmware
V1.2.0RC5
rel.android-build.20150106.154003
Last updated at 22:08.27

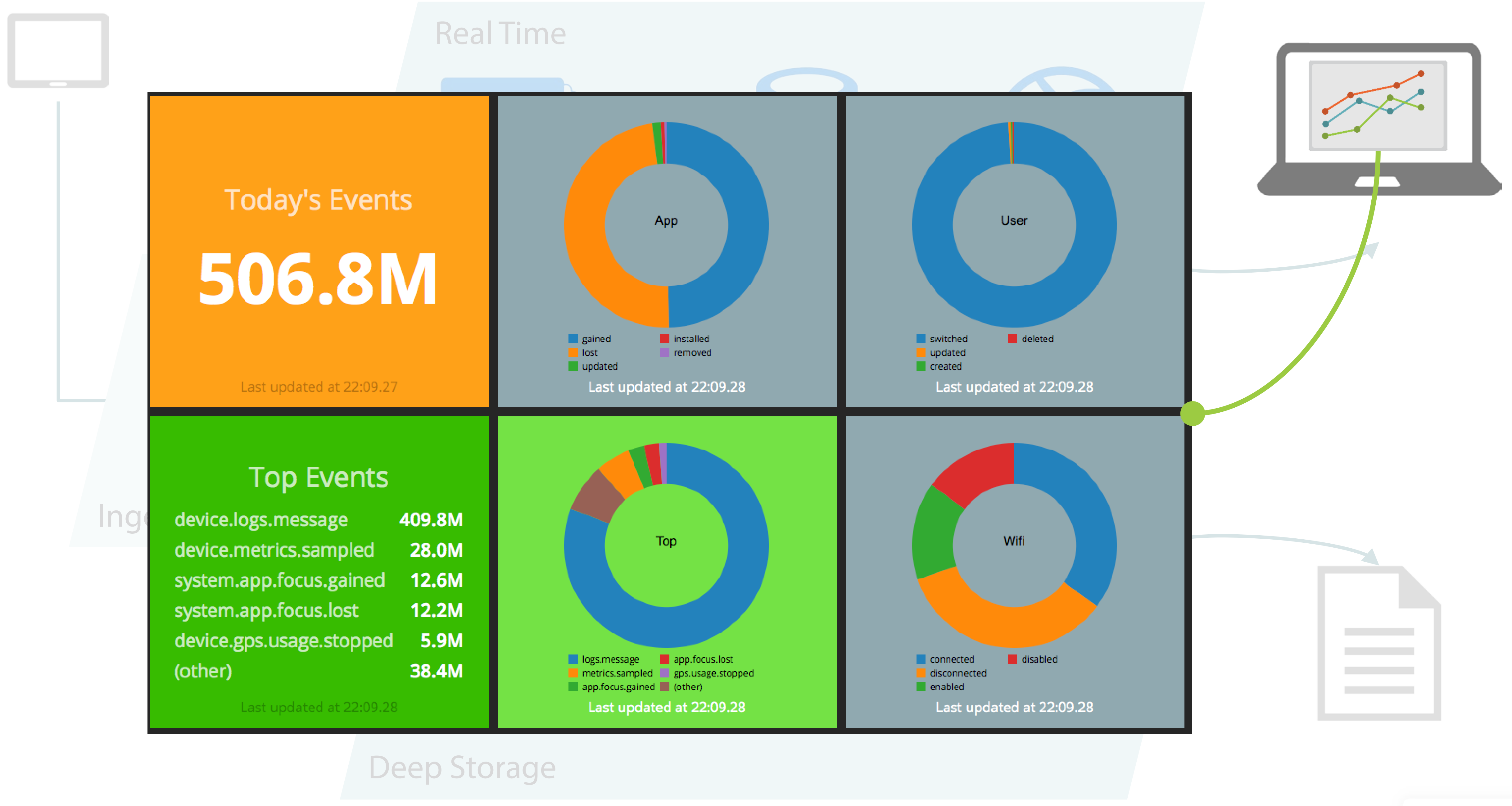
User Base
57.4%
318222 out of 554288 devices
Last updated at 22:07.58

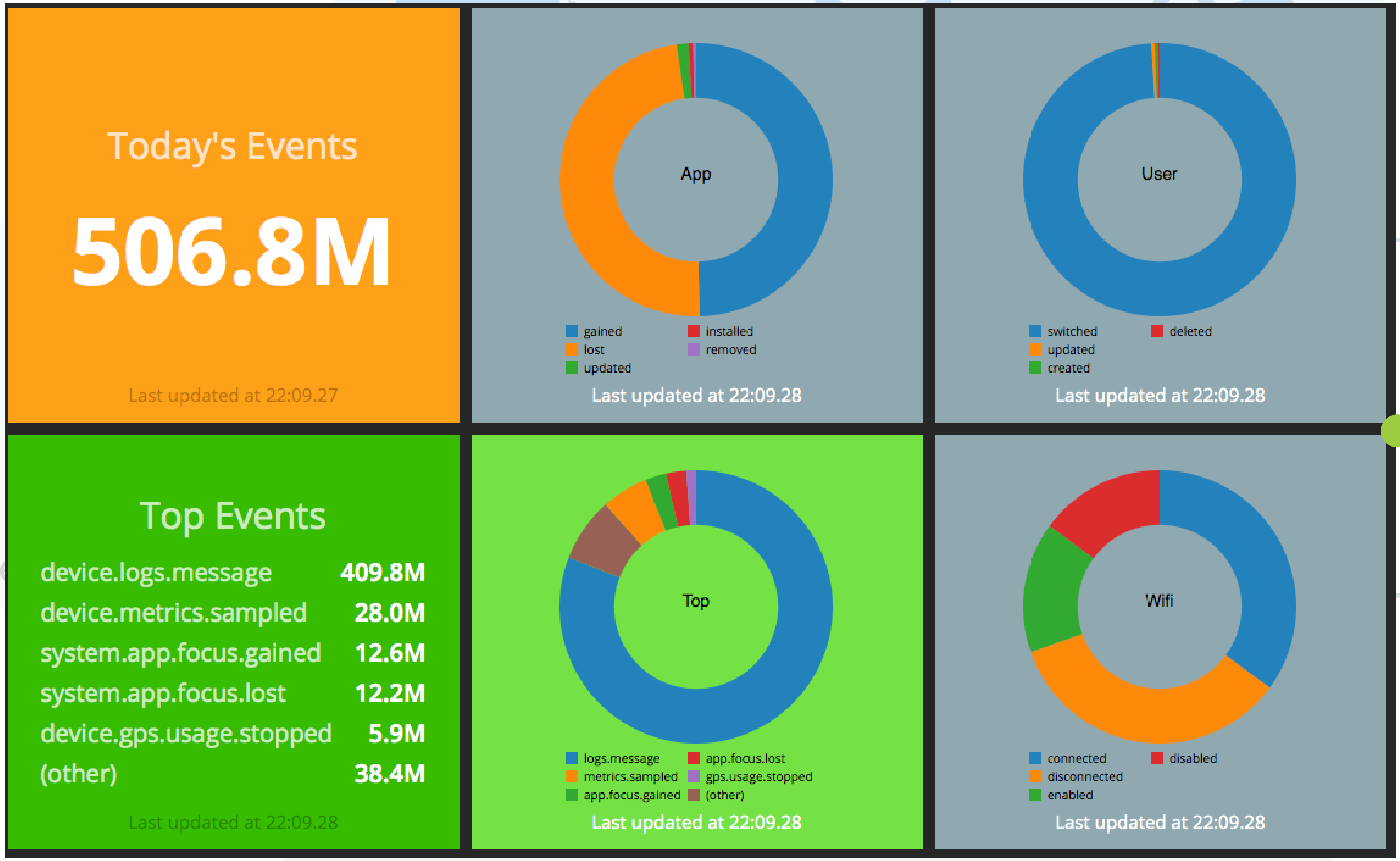
Breakdown
🟦 v1.2.0RC5 🟥 v1.1.0RC3
🟧 v1.1.2RC1 🟪 v1.0.0RC4
🟩 v1.1.1RC1 🟫 (other)
Last updated at 22:08.01

Latest BIOS
44.0
BayTrail.5.04.16.0044.0
Last updated at 22:08.27

User Base
0%
1 out of 554289 devices
Last updated at 22:08.28

Breakdown
🟦 43.0 🟥 43.H
🟧 43.D 🟪 43.3
🟩 43.C 🟫 (other)
Last updated at 22:08.00





Today's Events

506.8M

Last updated at 22:09.27

Last updated at 22:09.28

Last updated at 22:09.28

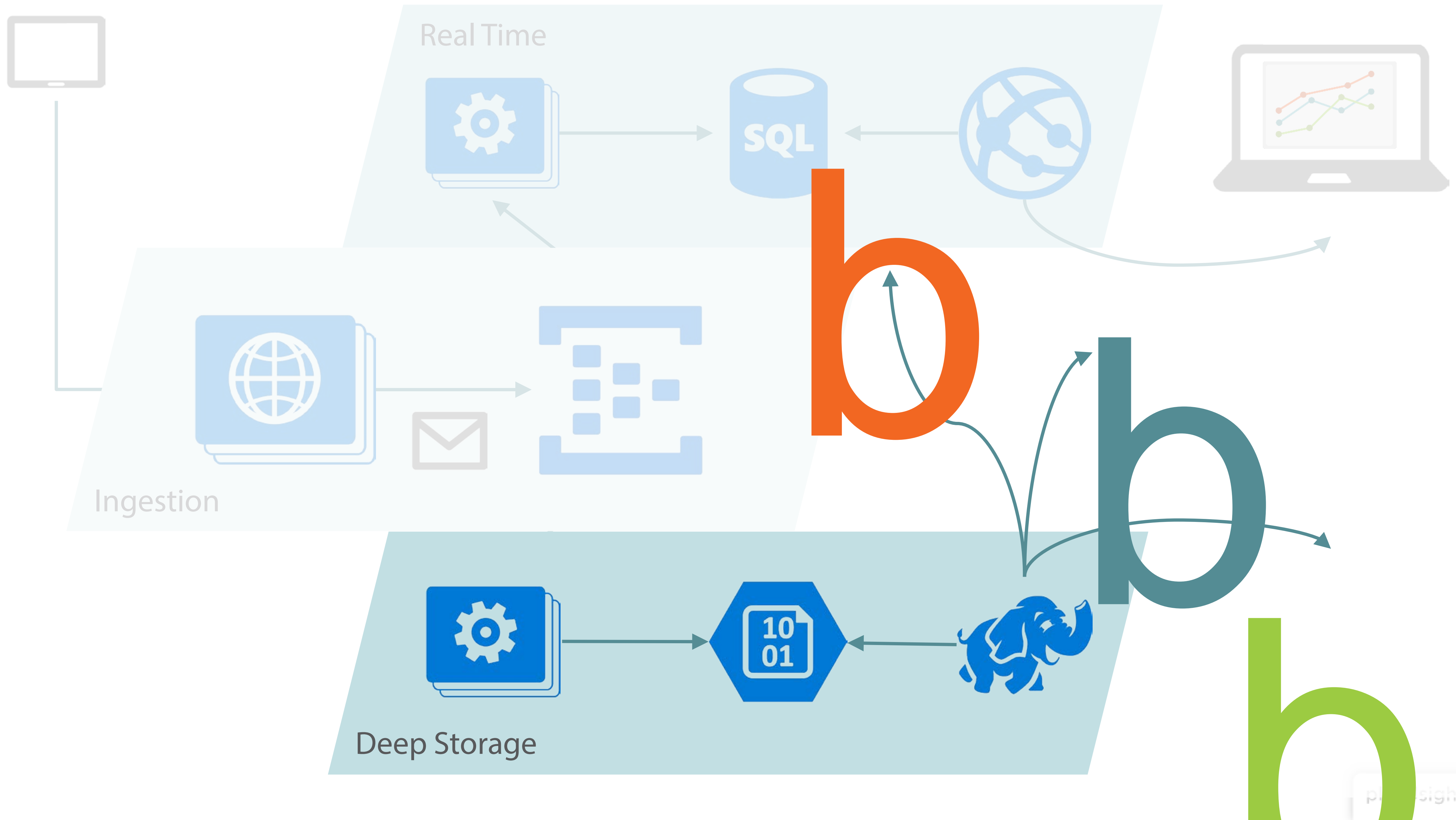
Top Events

device.logs.message	409.8M
device.metrics.sampled	28.0M
system.app.focus.gained	12.6M
system.app.focus.lost	12.2M
device.gps.usage.stopped	5.9M
(other)	38.4M

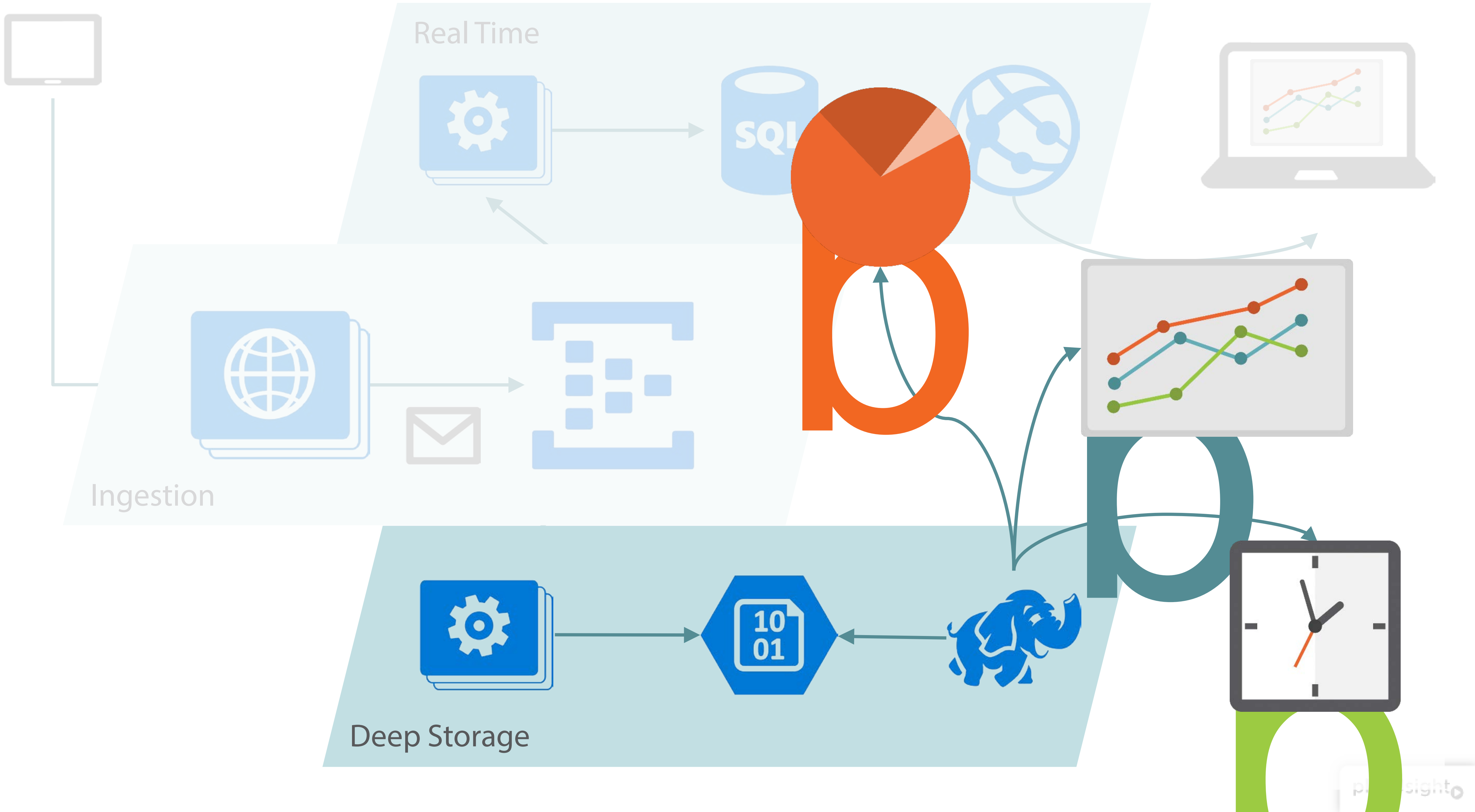
Last updated at 22:09.28

Last updated at 22:09.28

Last updated at 22:09.28



Real Time
SQL
Ingestion
Deep Storage
10
01
b
b
b

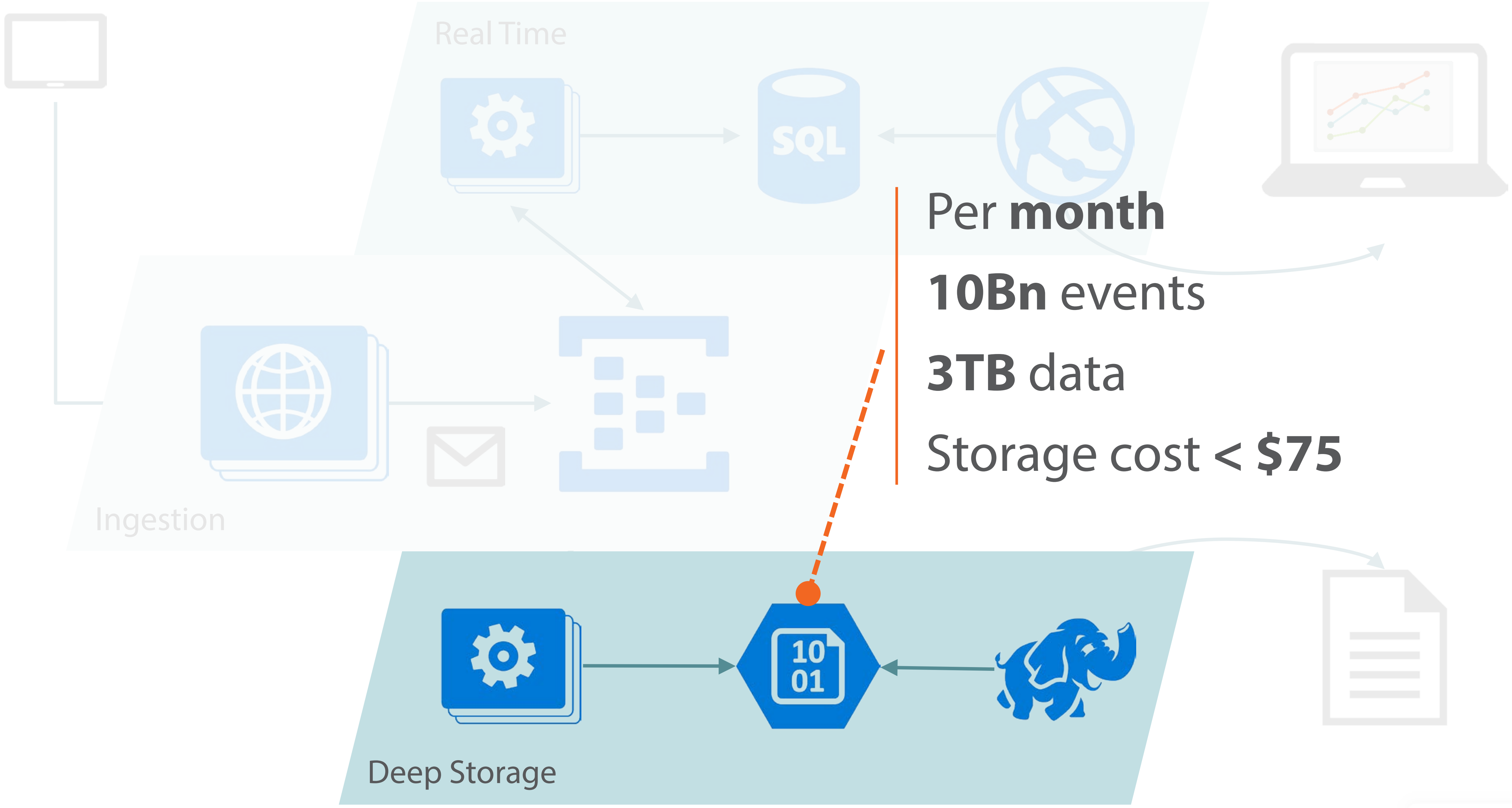

Real Time
SQL
Ingestion
10
01
Deep Storage

Azure Blob Storage

Hadoop HDFS Compatible

Local & Geo-Replication

Billed at < $25 per TB

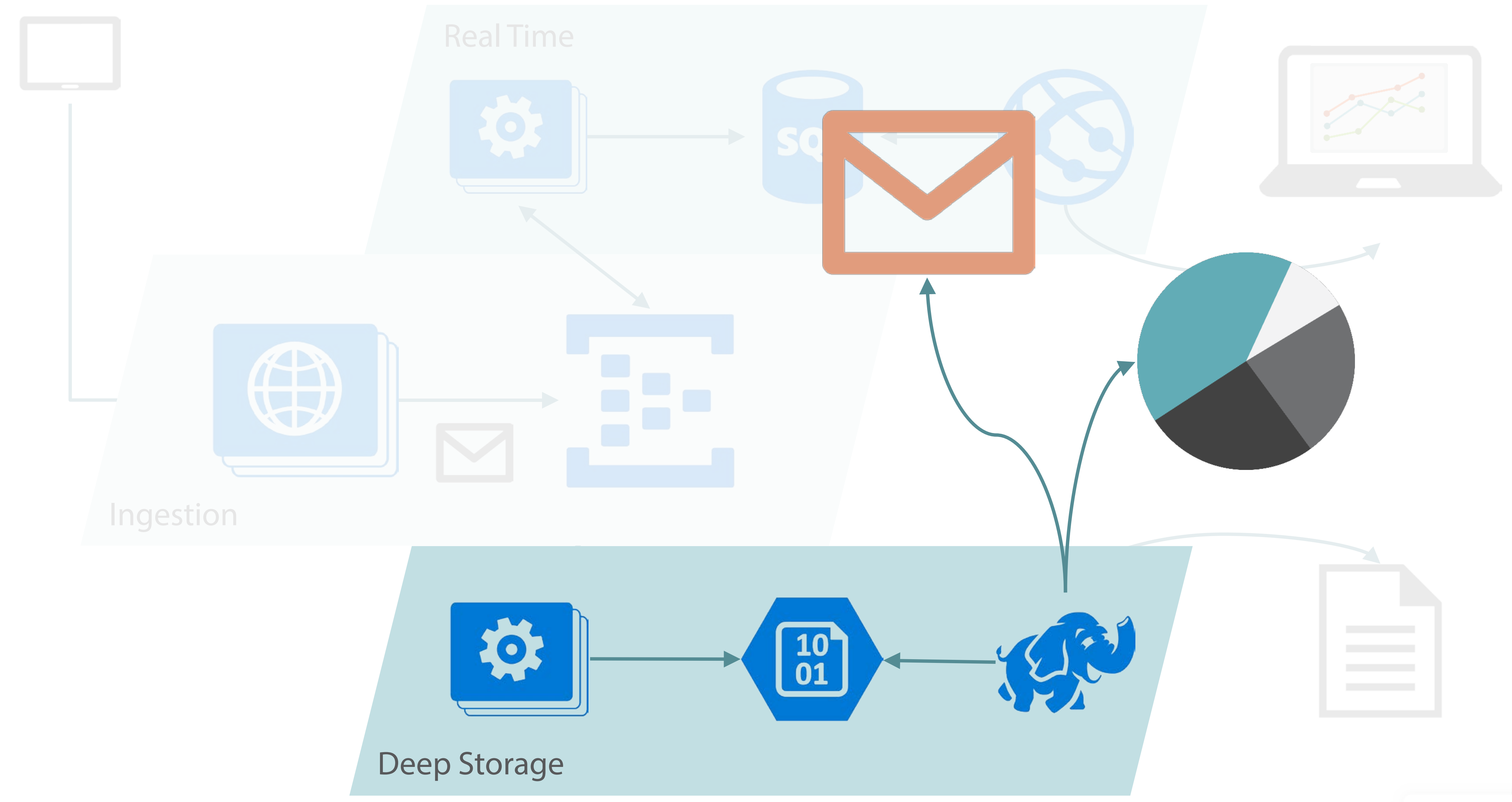


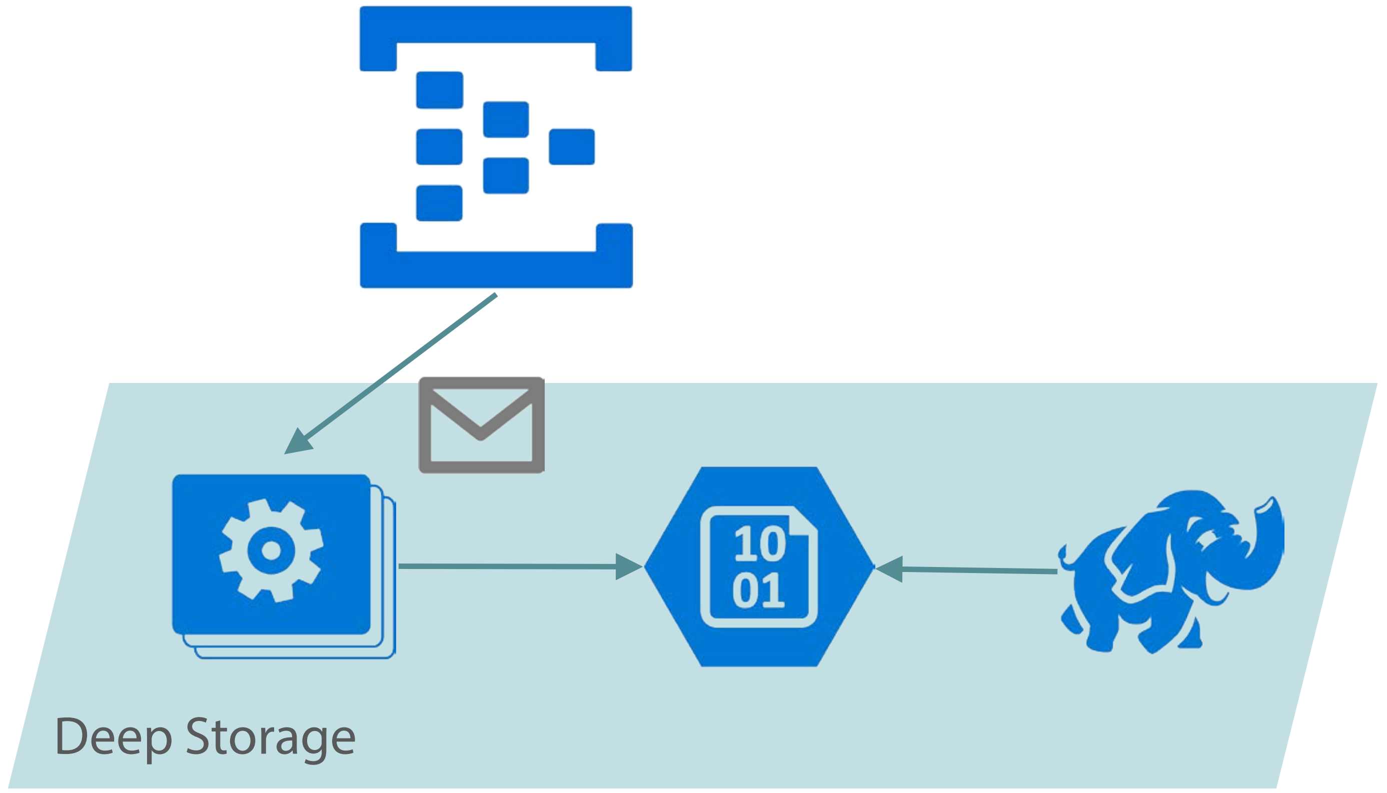
Per **month**

10Bn events

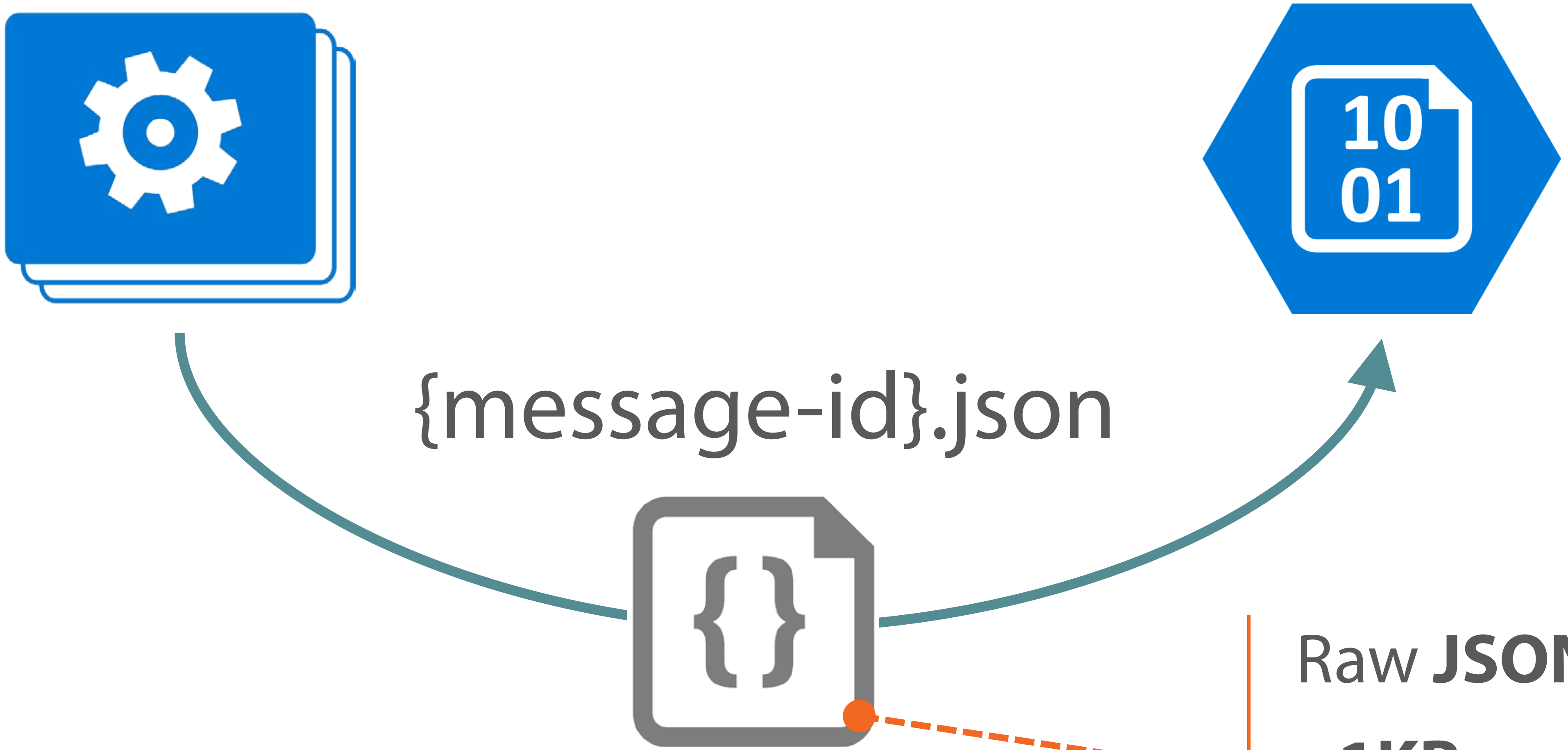
3TB data

Storage cost < **$75**


Real Time
SQ
Ingestion
10
01
Deep Storage



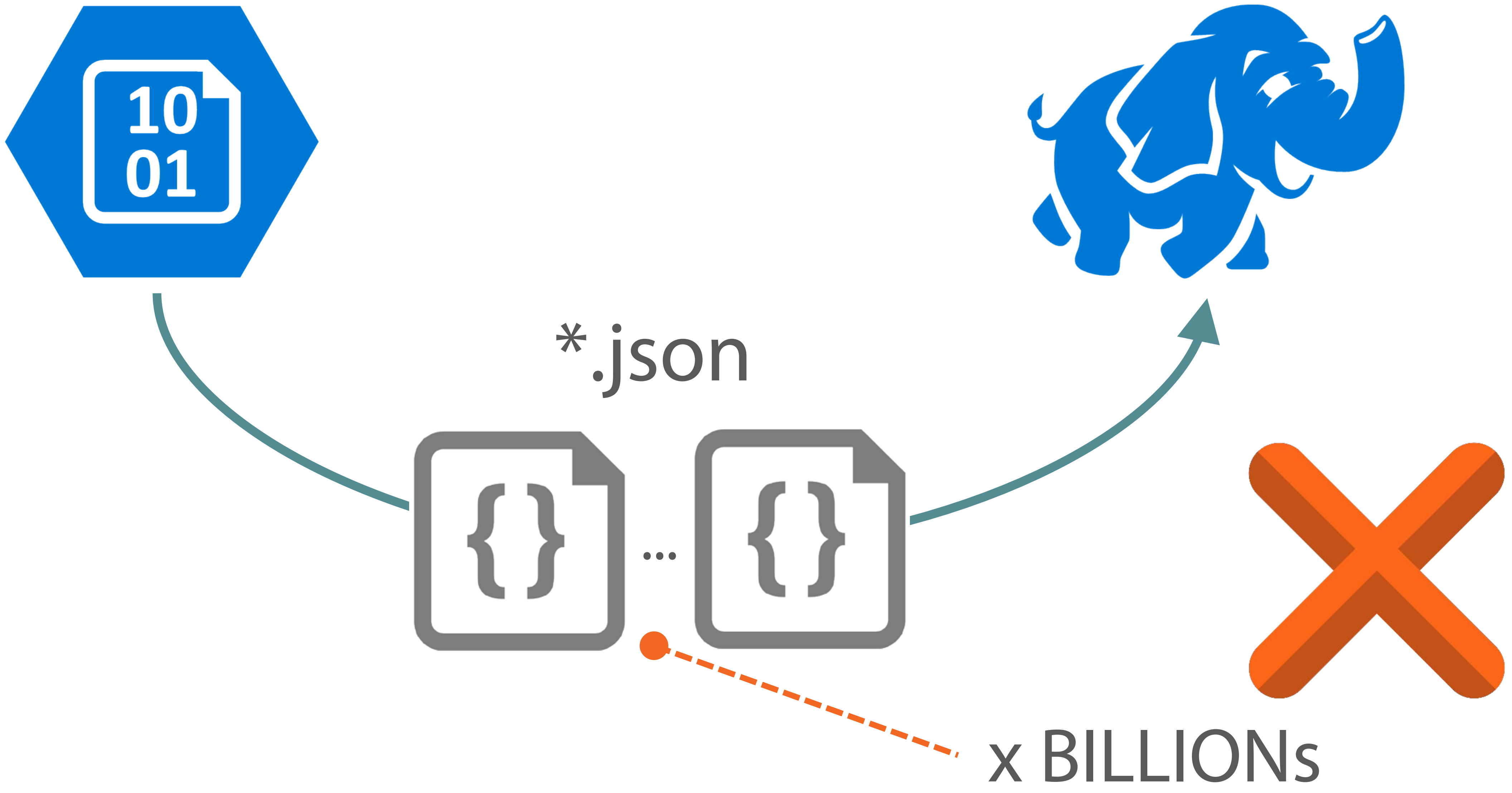
- Efficiency
- Reliability
- Traceability

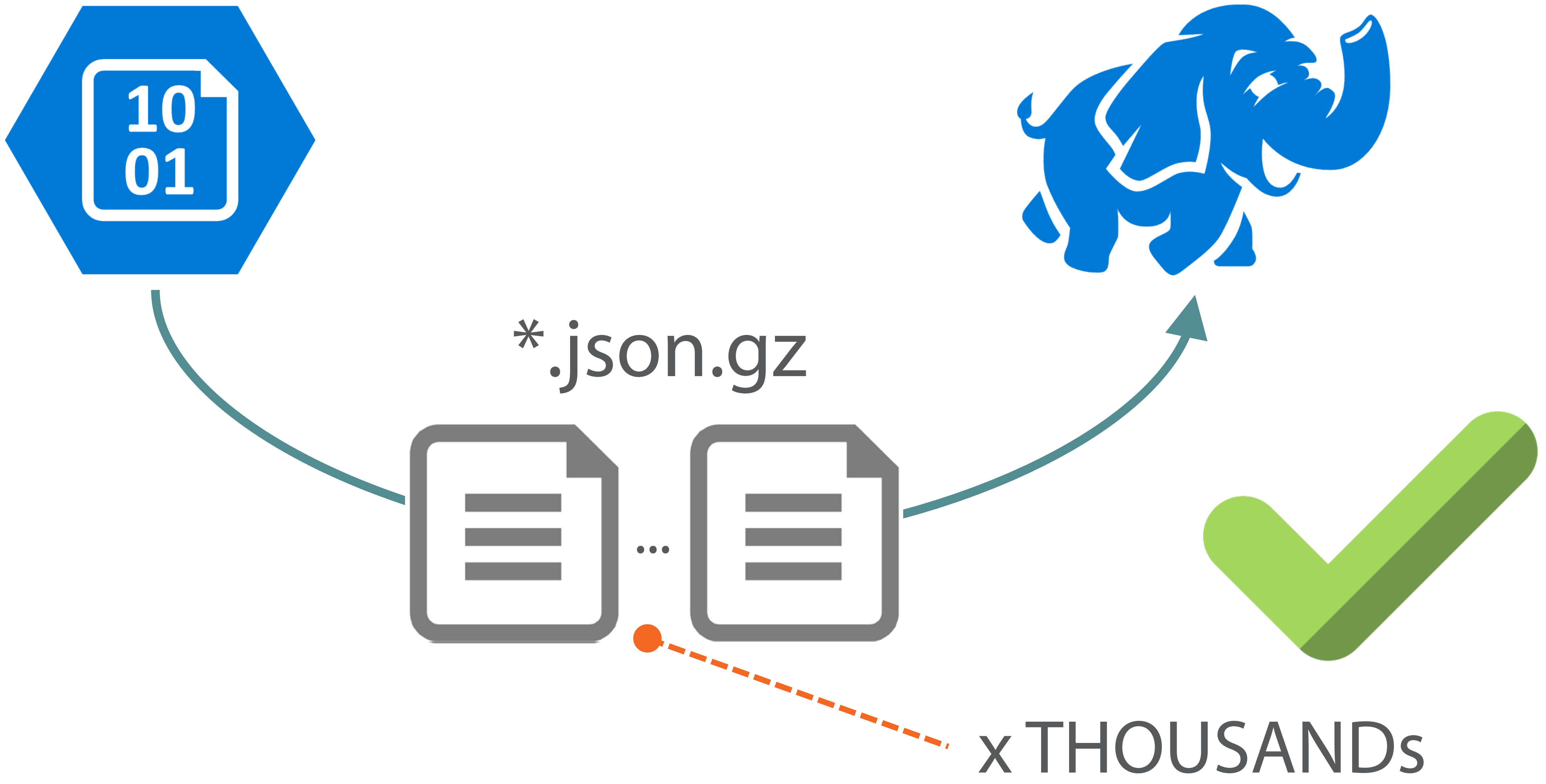


Raw **JSON**

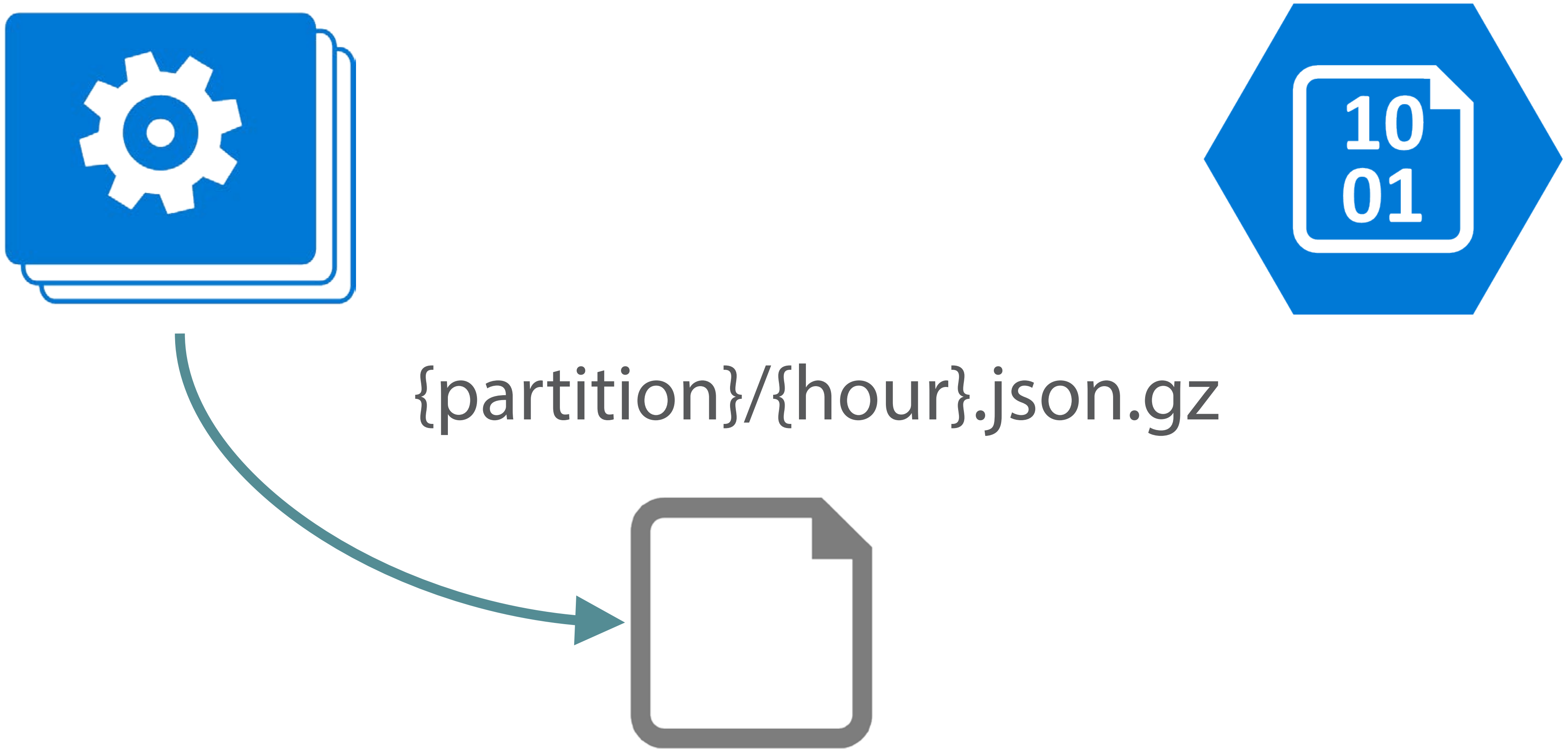
<1KB per blob

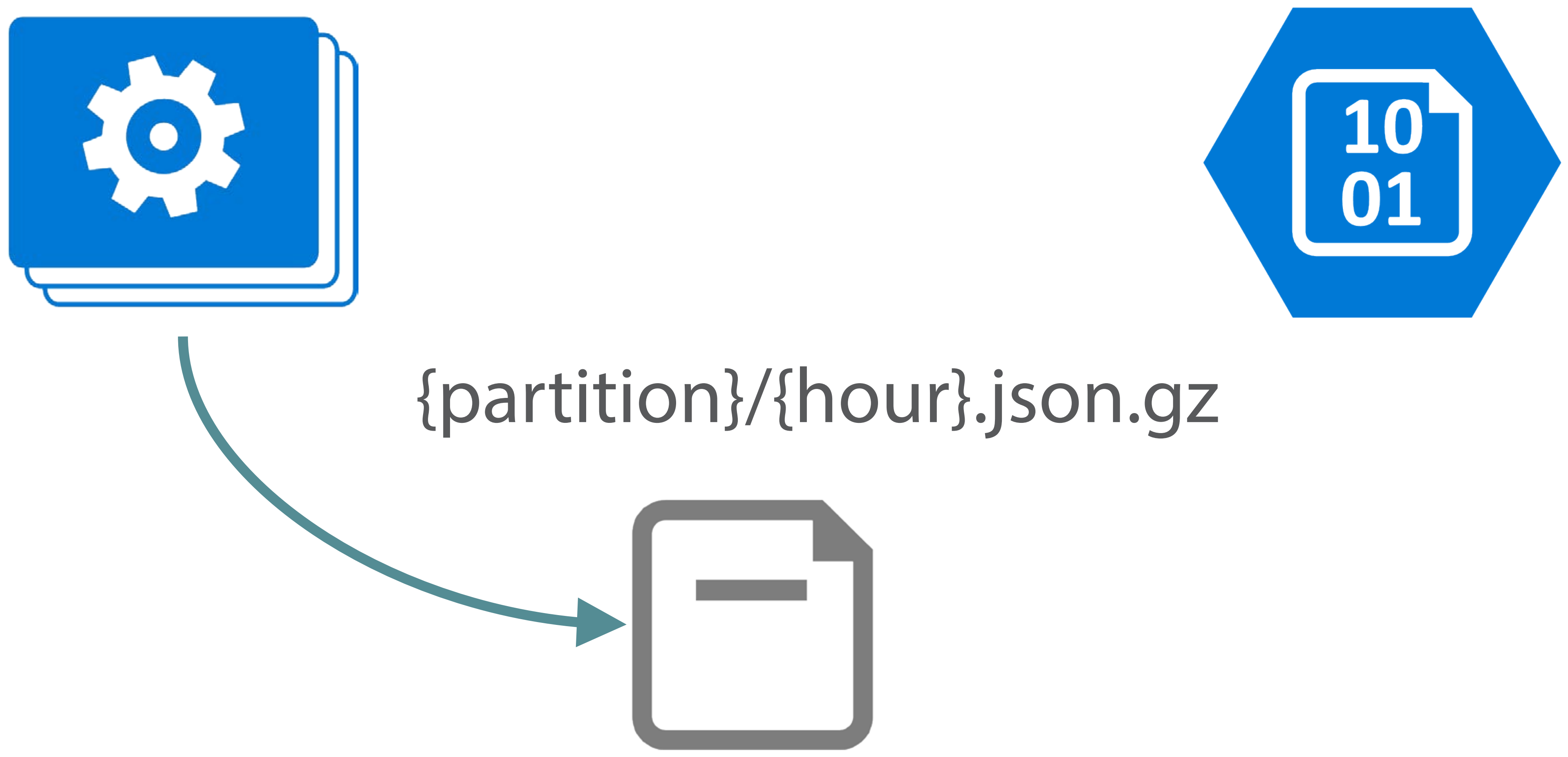
500M per day

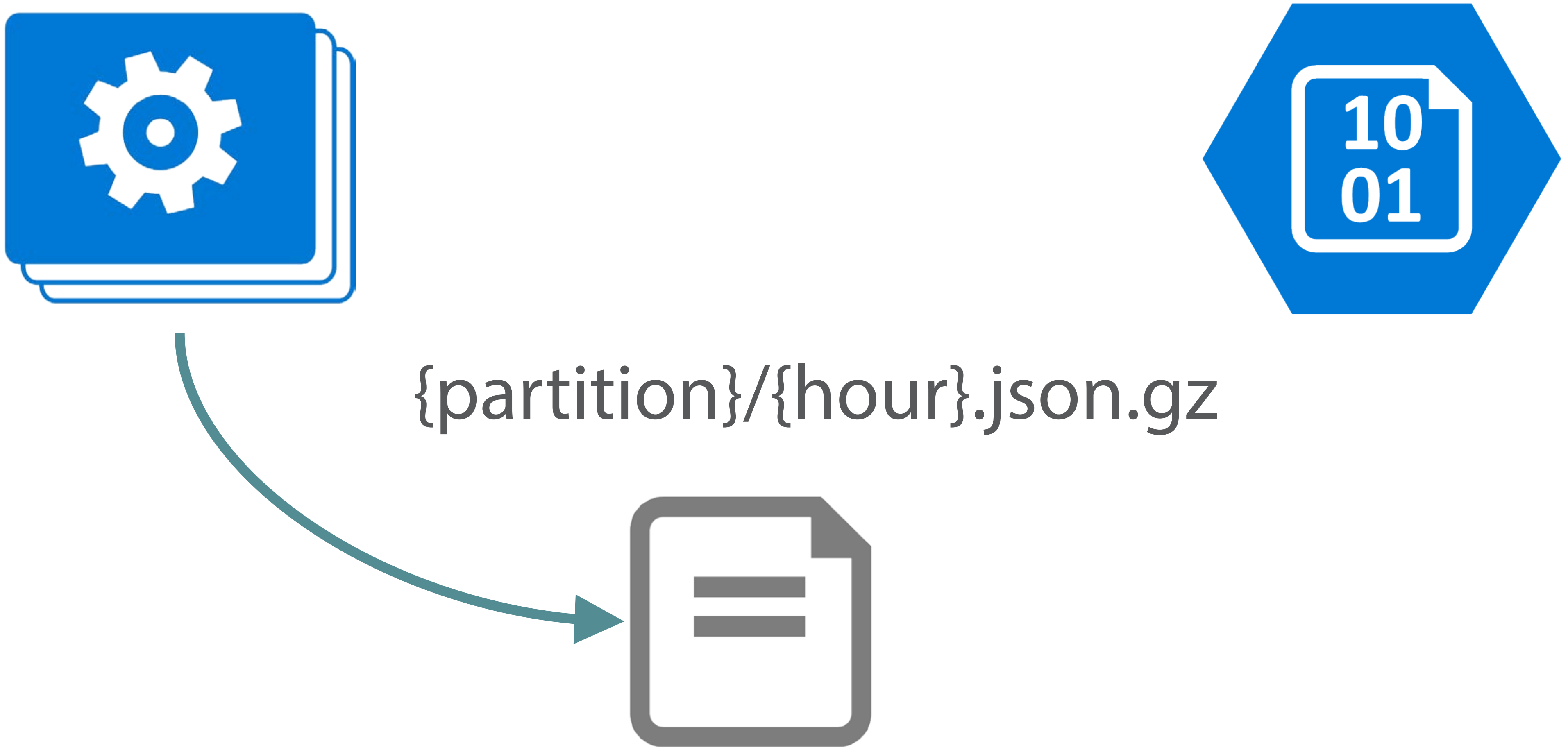

10
01
*.json
{}
...
{}
x BILLIONs

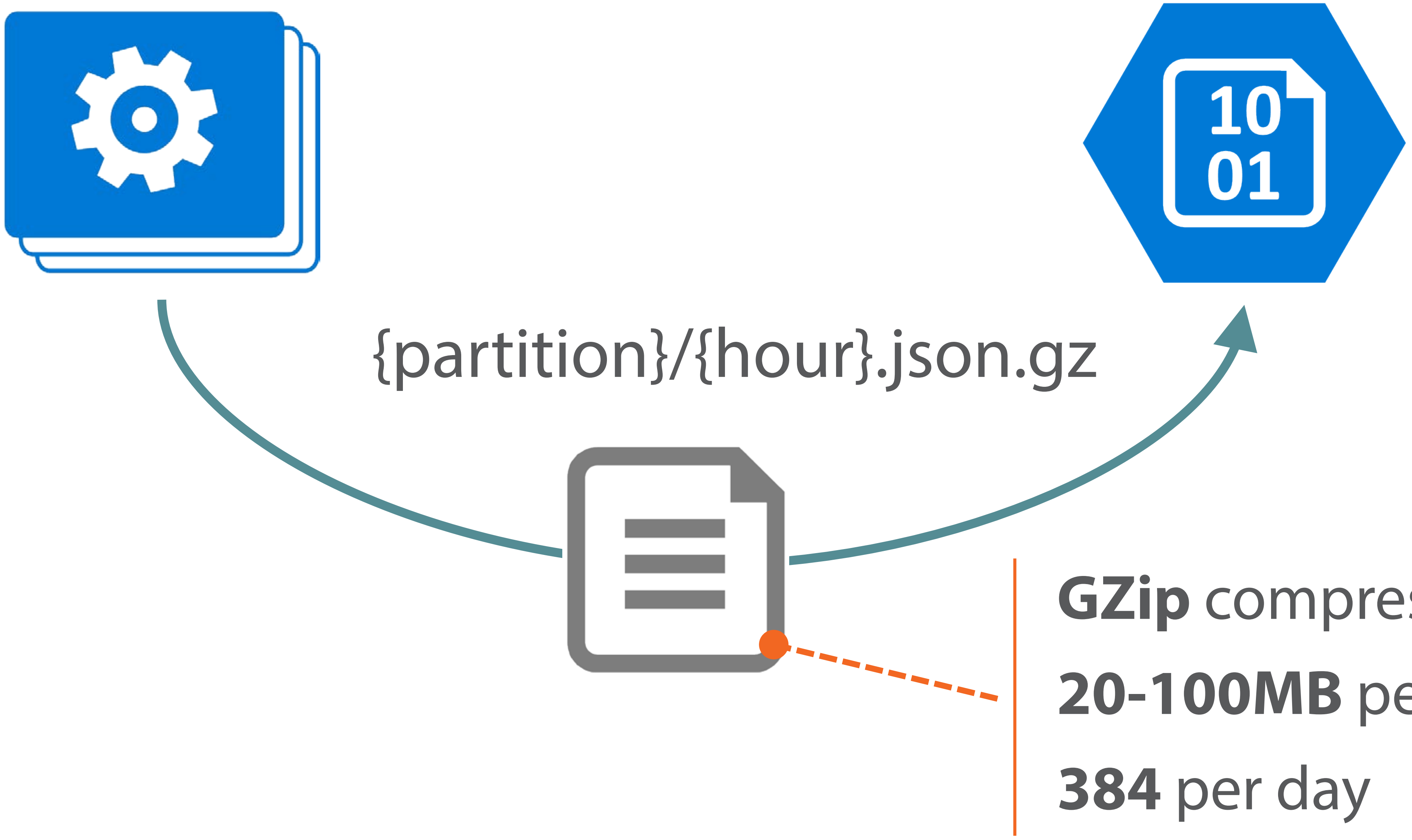


10
01
*.json.gz
...
x THOUSANDs


10
01
{partition}/{hour}.json.gz


10
01
{partition}/{hour}.json.gz

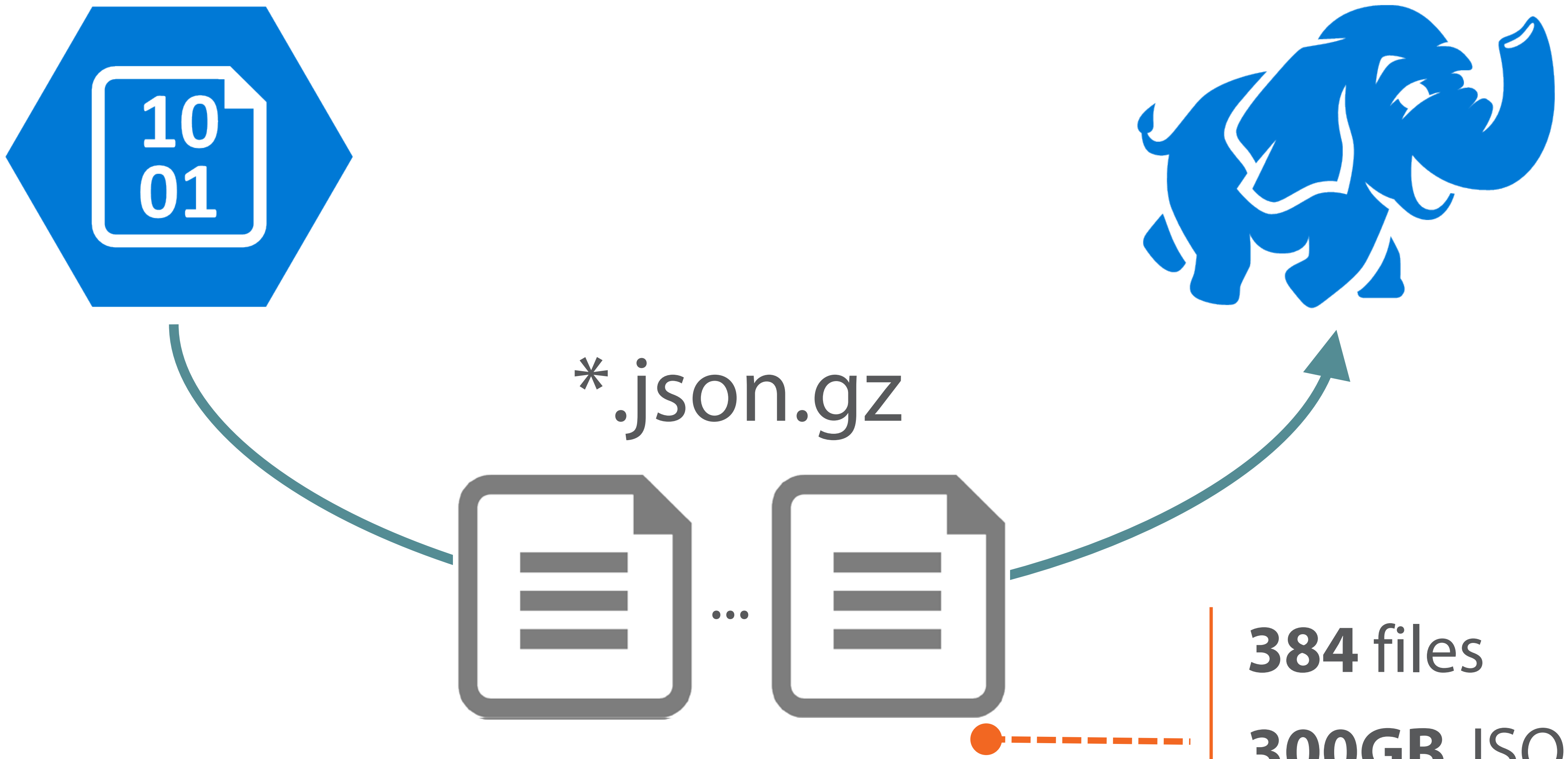

{partition}/{hour}.json.gz
10
01



GZip compressed

20-100MB per blob

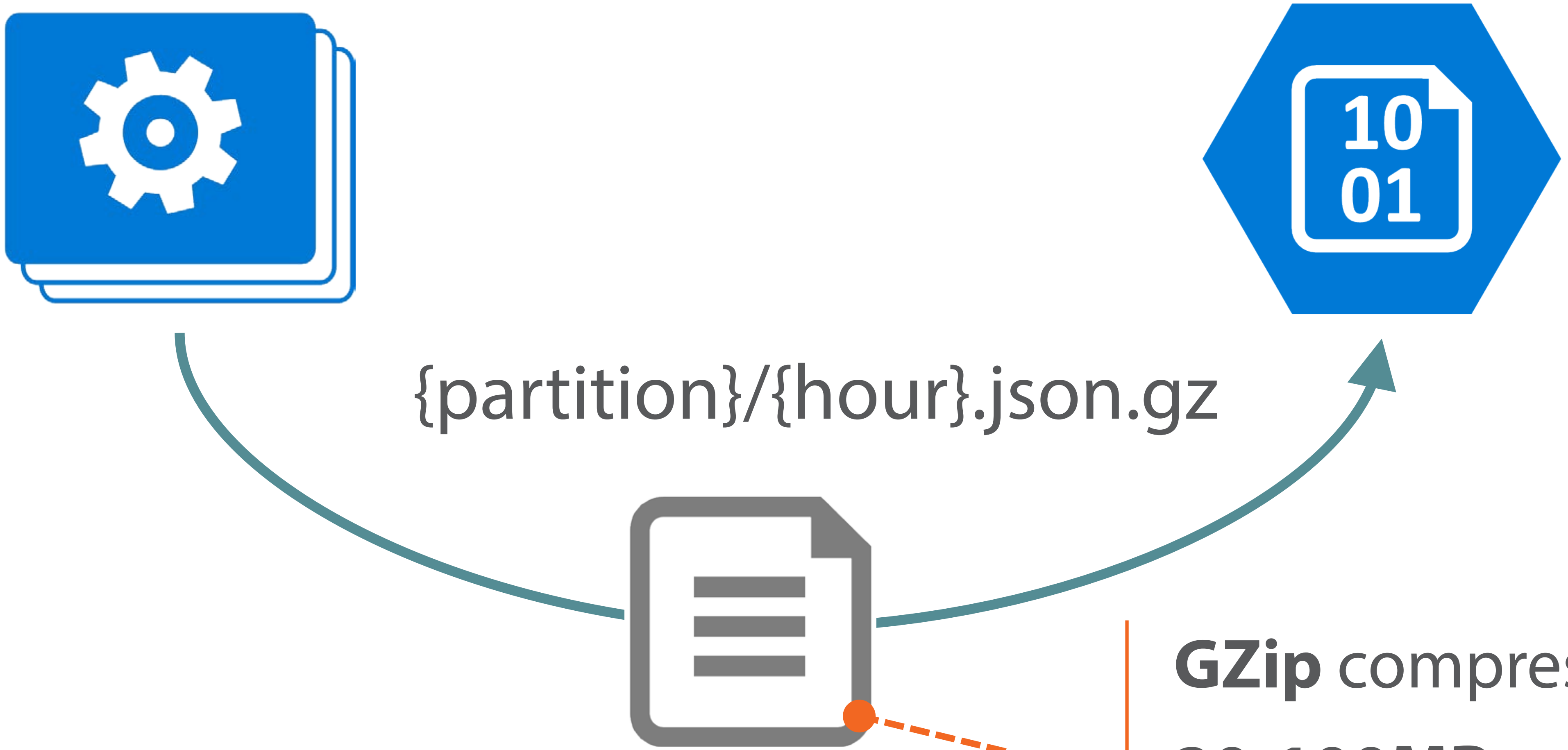
384 per day



384 files

300GB JSON

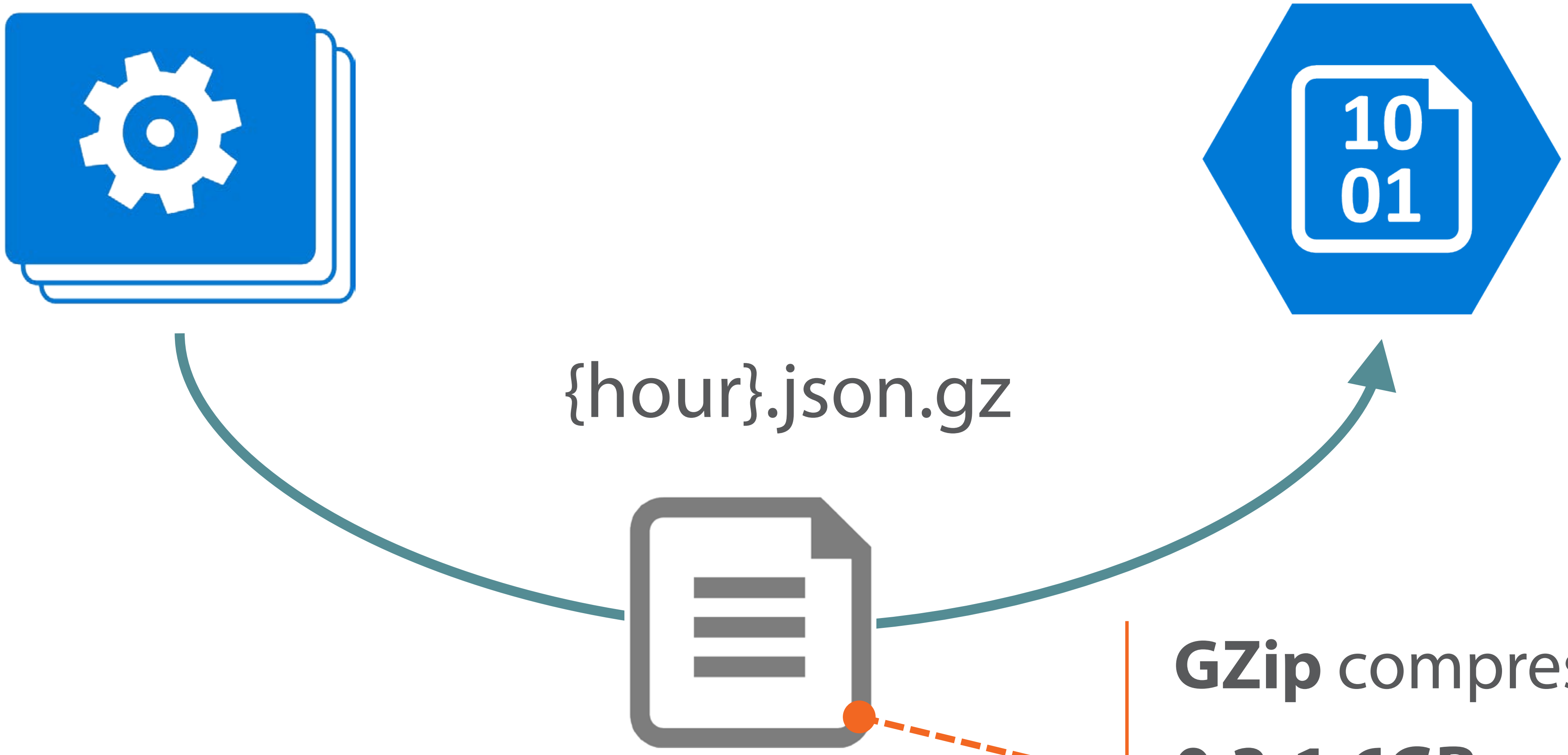
500M events



GZip compressed

20-100MB per blob

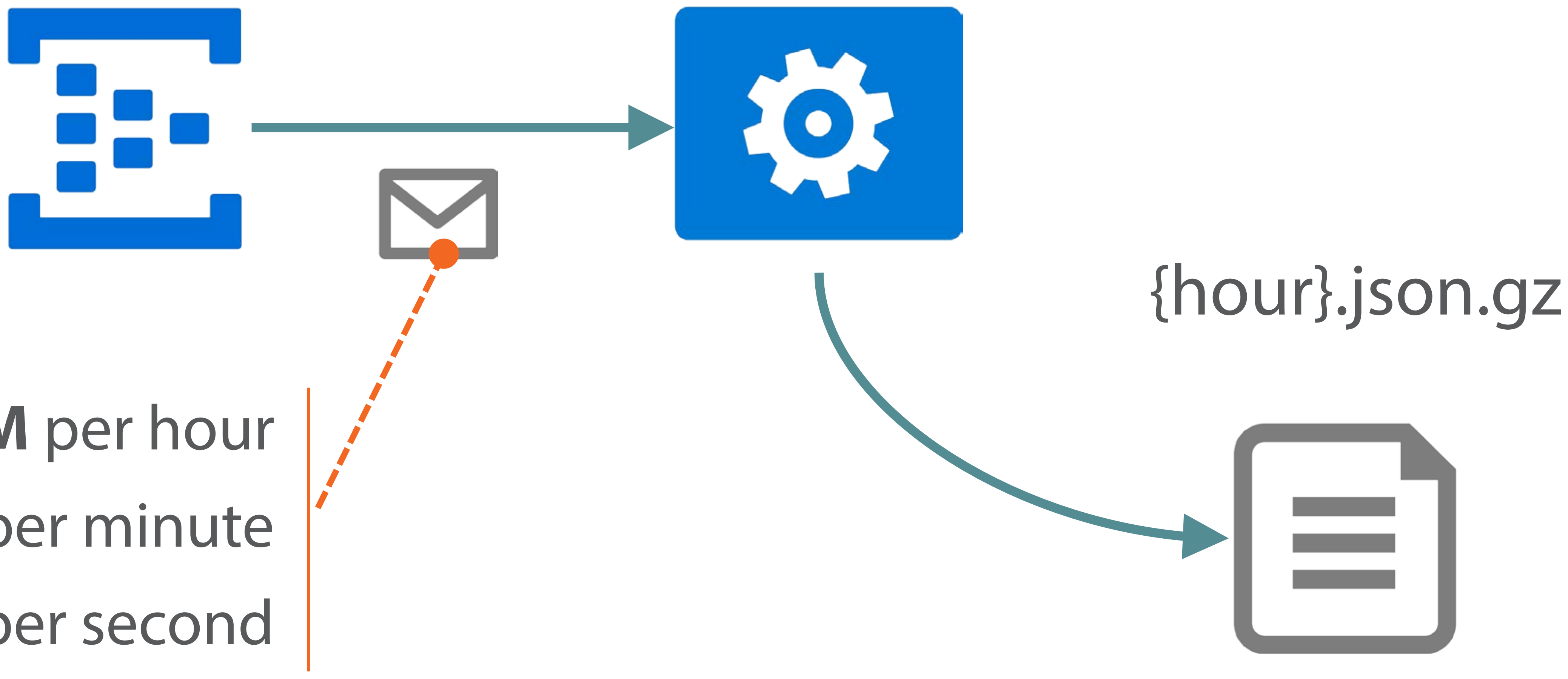
384 per day



GZip compressed

0.3-1.6GB per blob

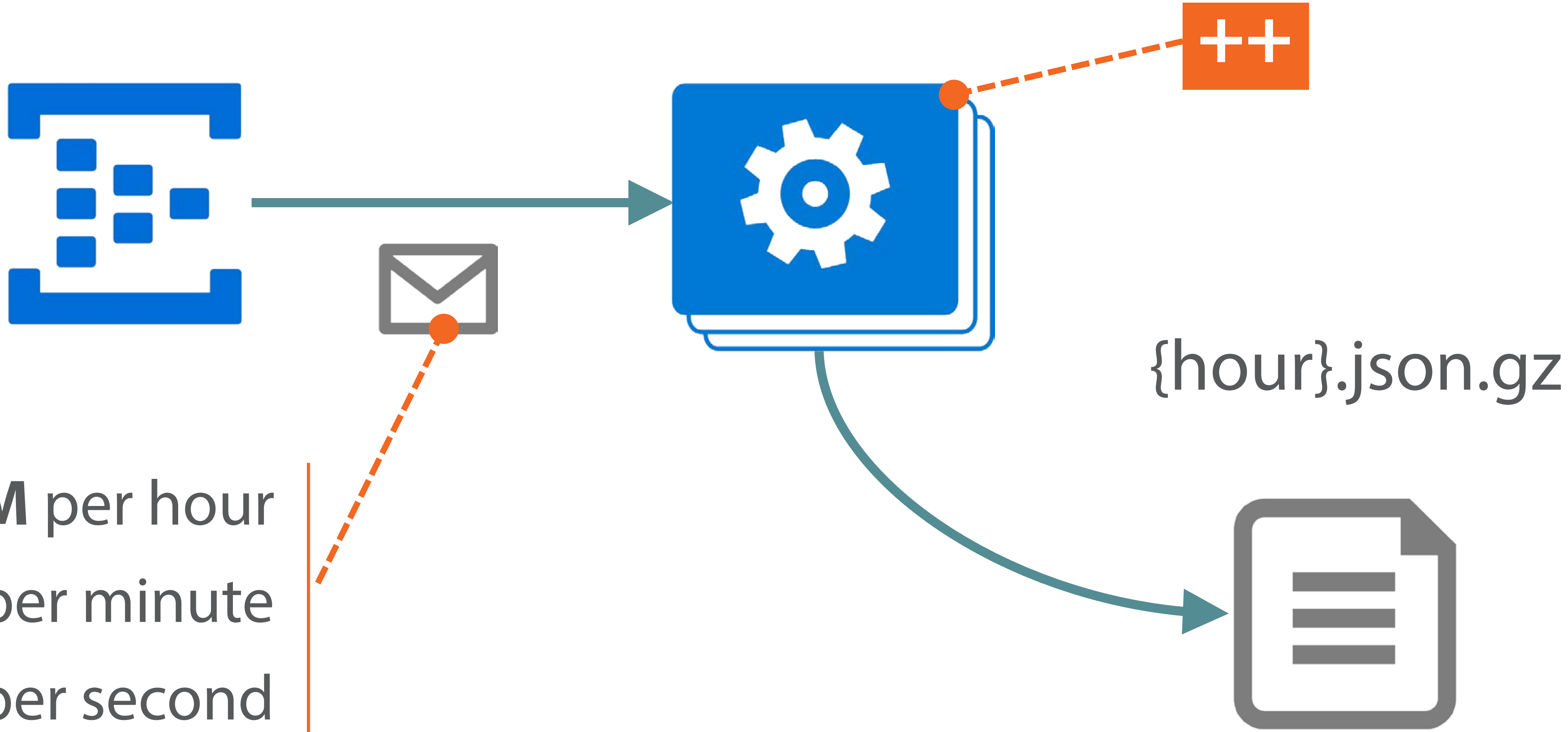
24 per day



Peak **40M** per hour

>600K per minute

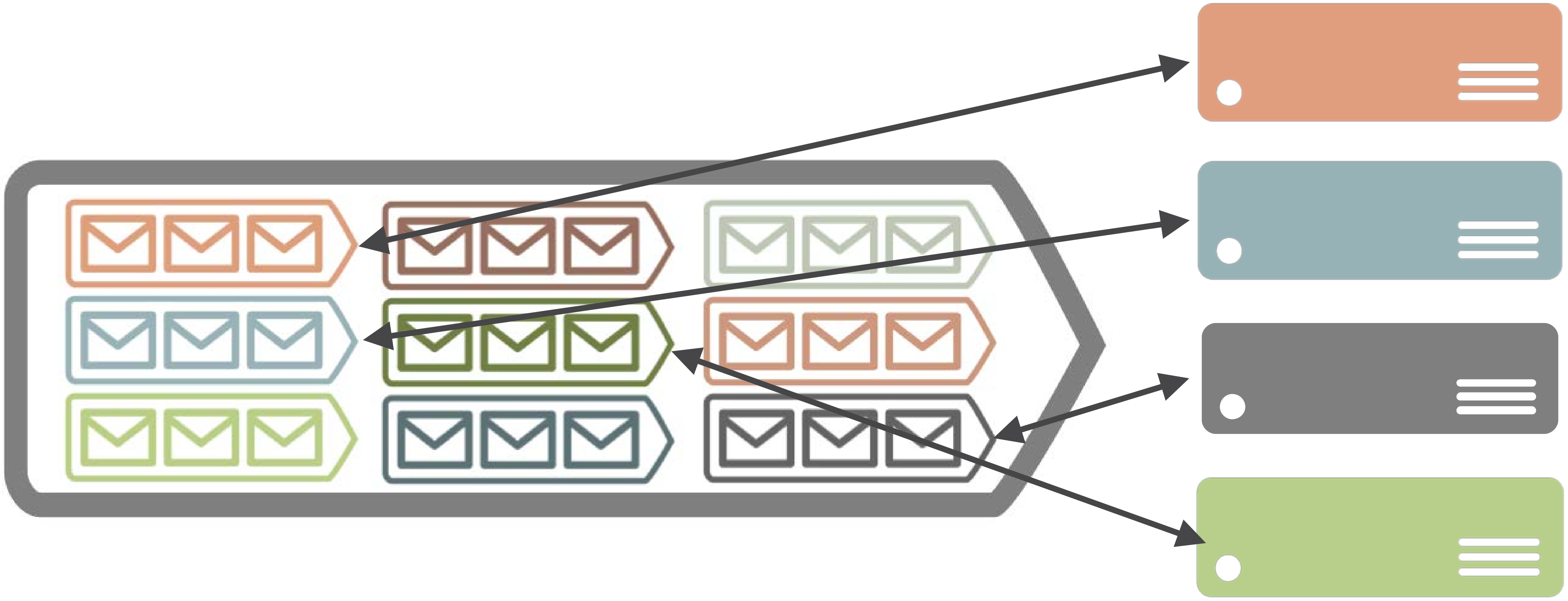
>11K per second

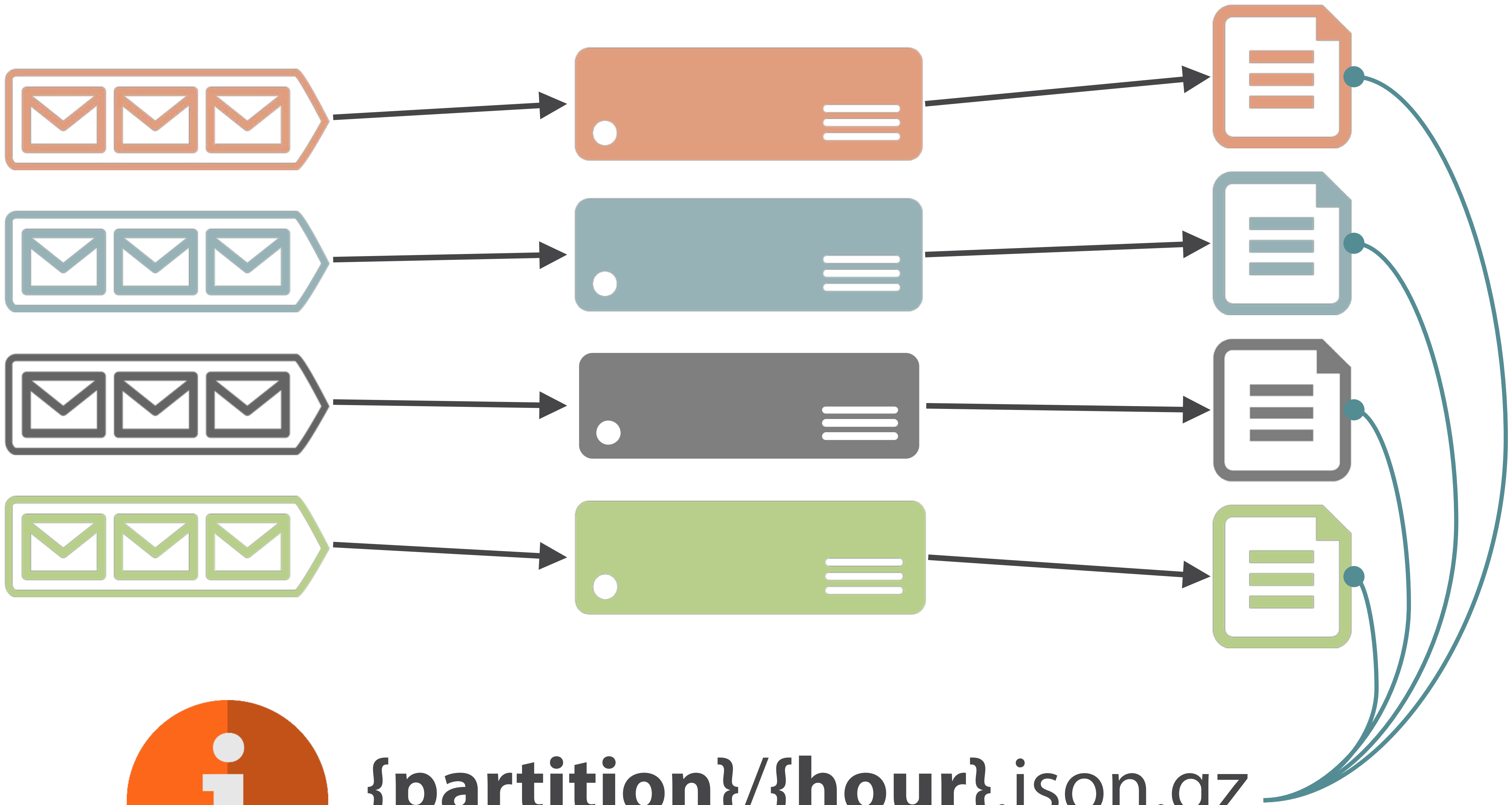


Peak **40M** per hour

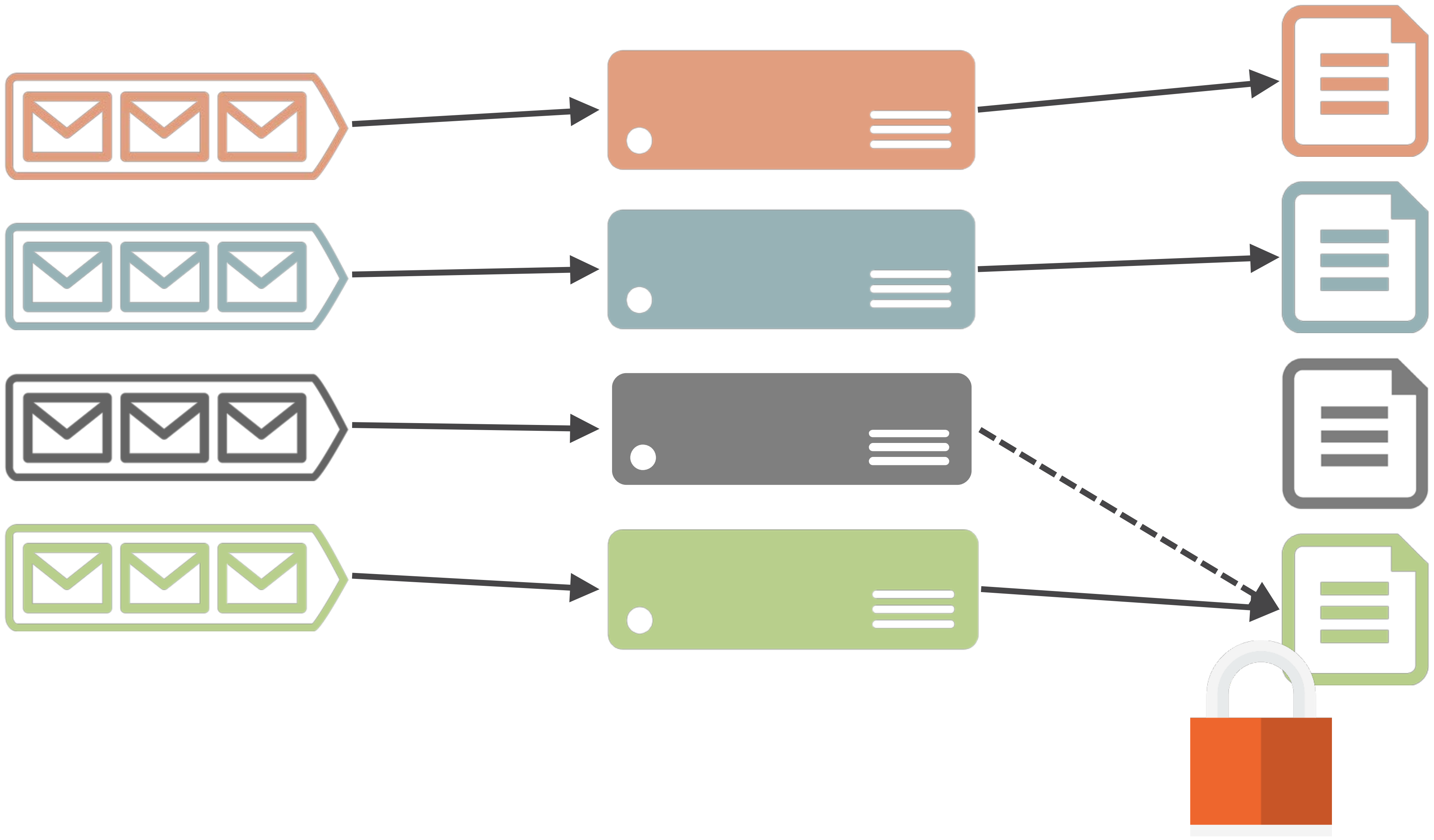
>600K per minute

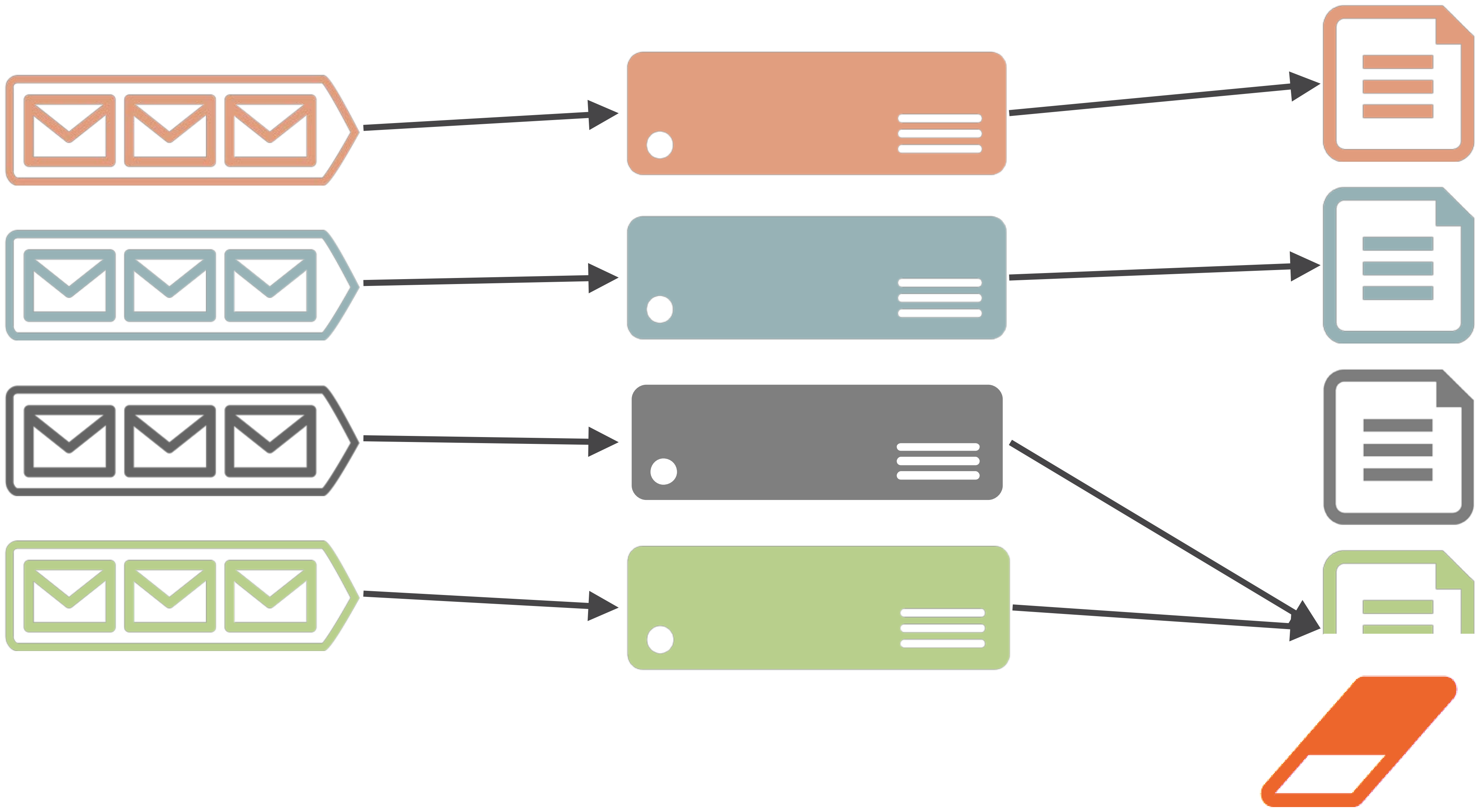
>11K per second

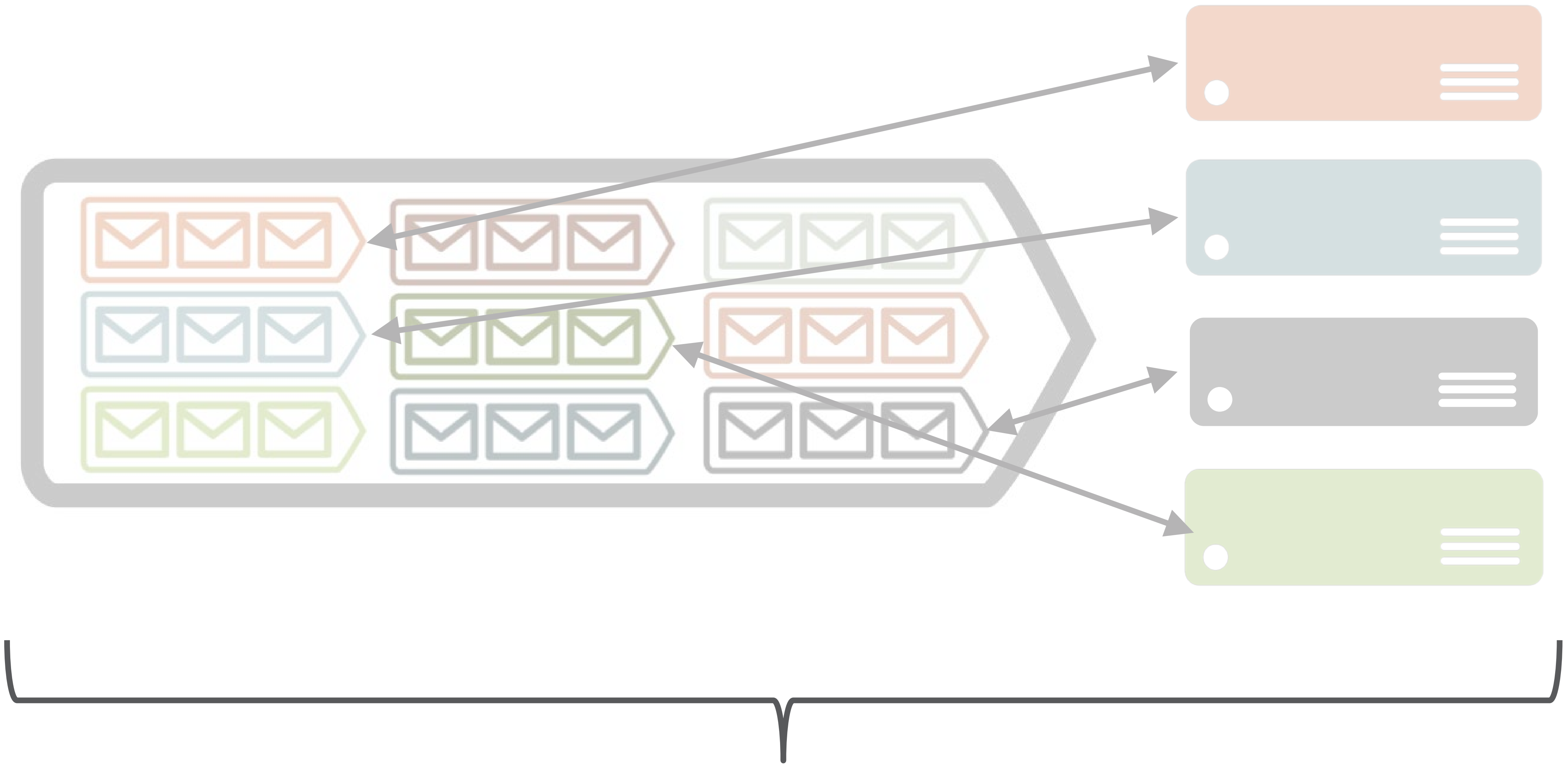




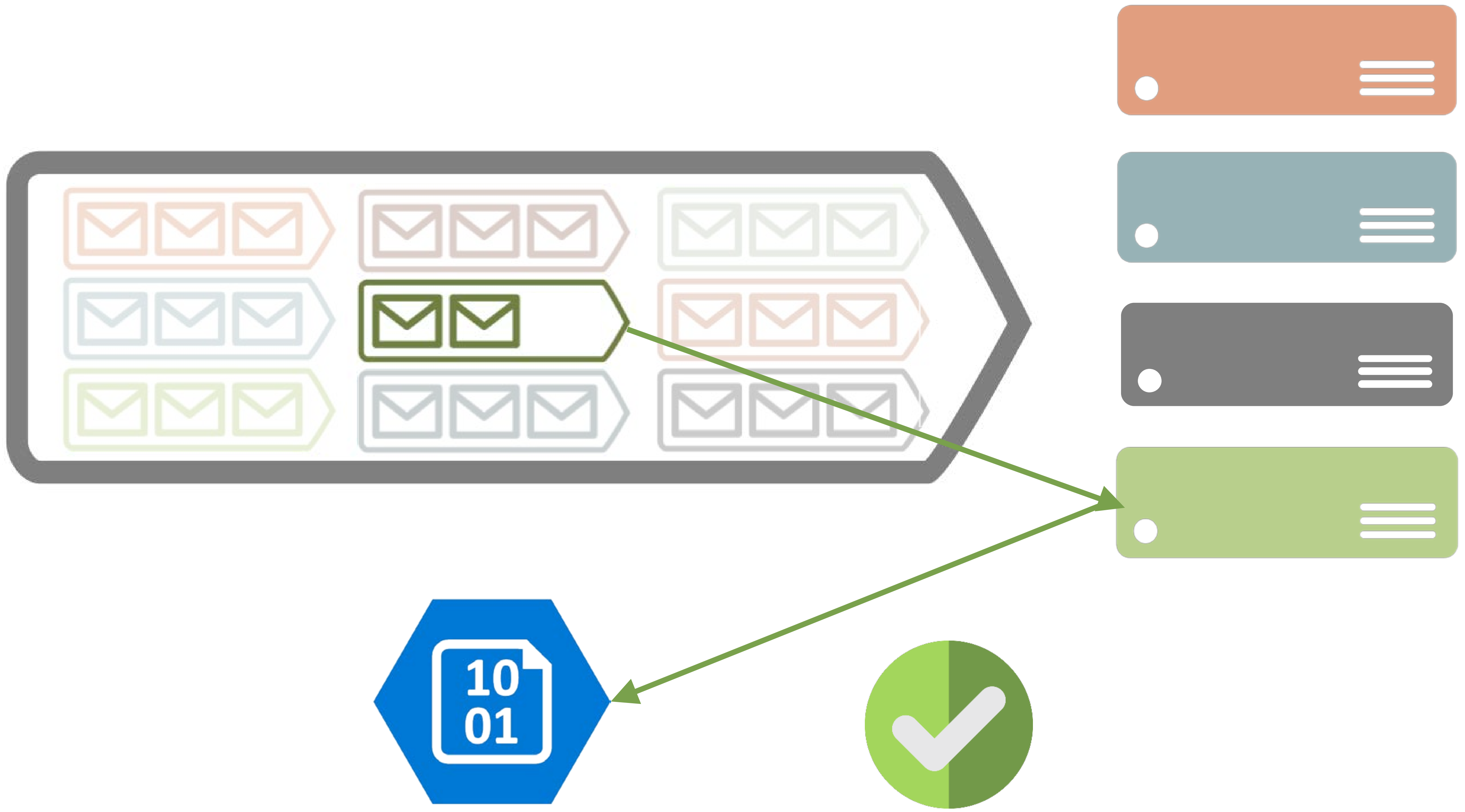
{partition}/{hour}.json.gz

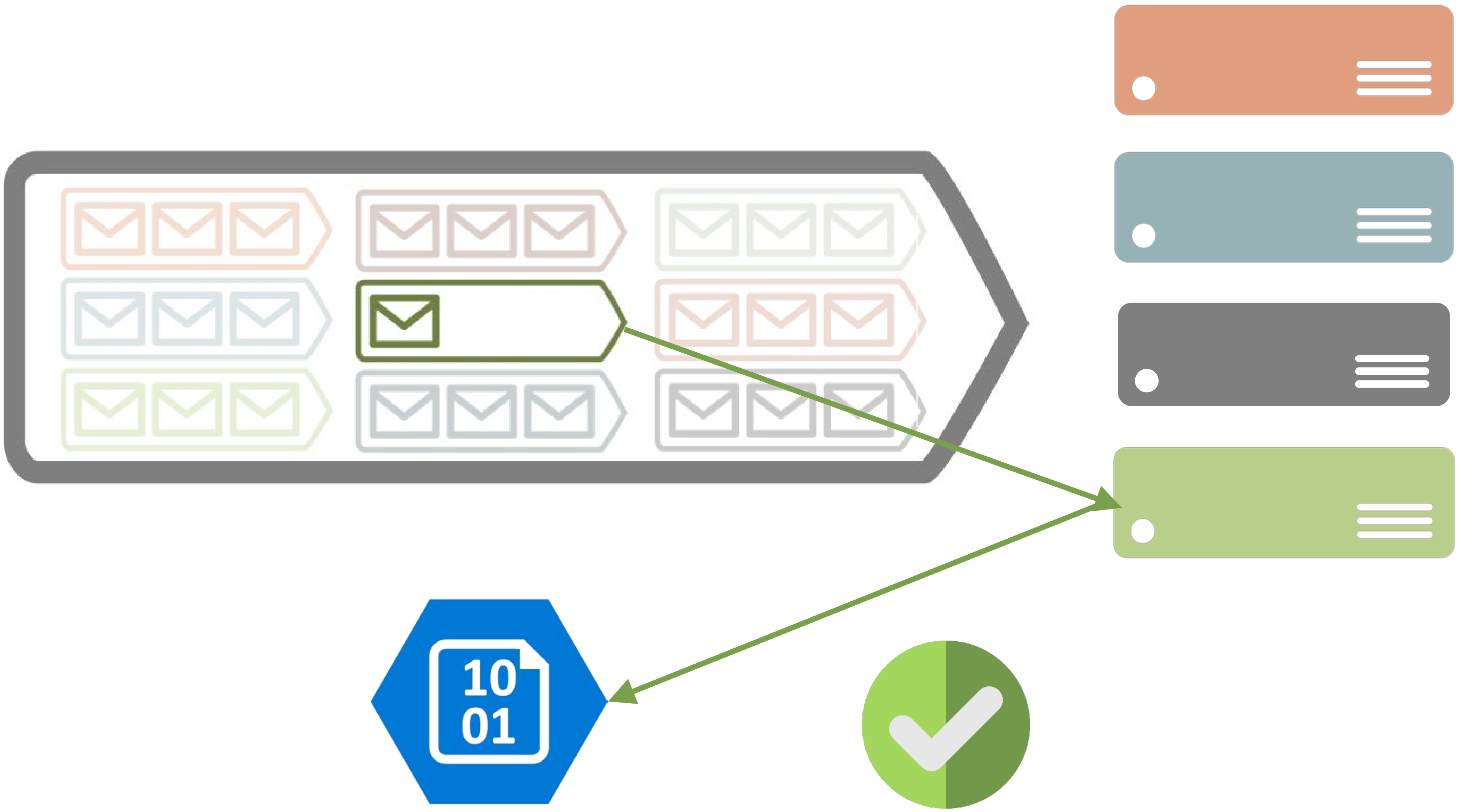


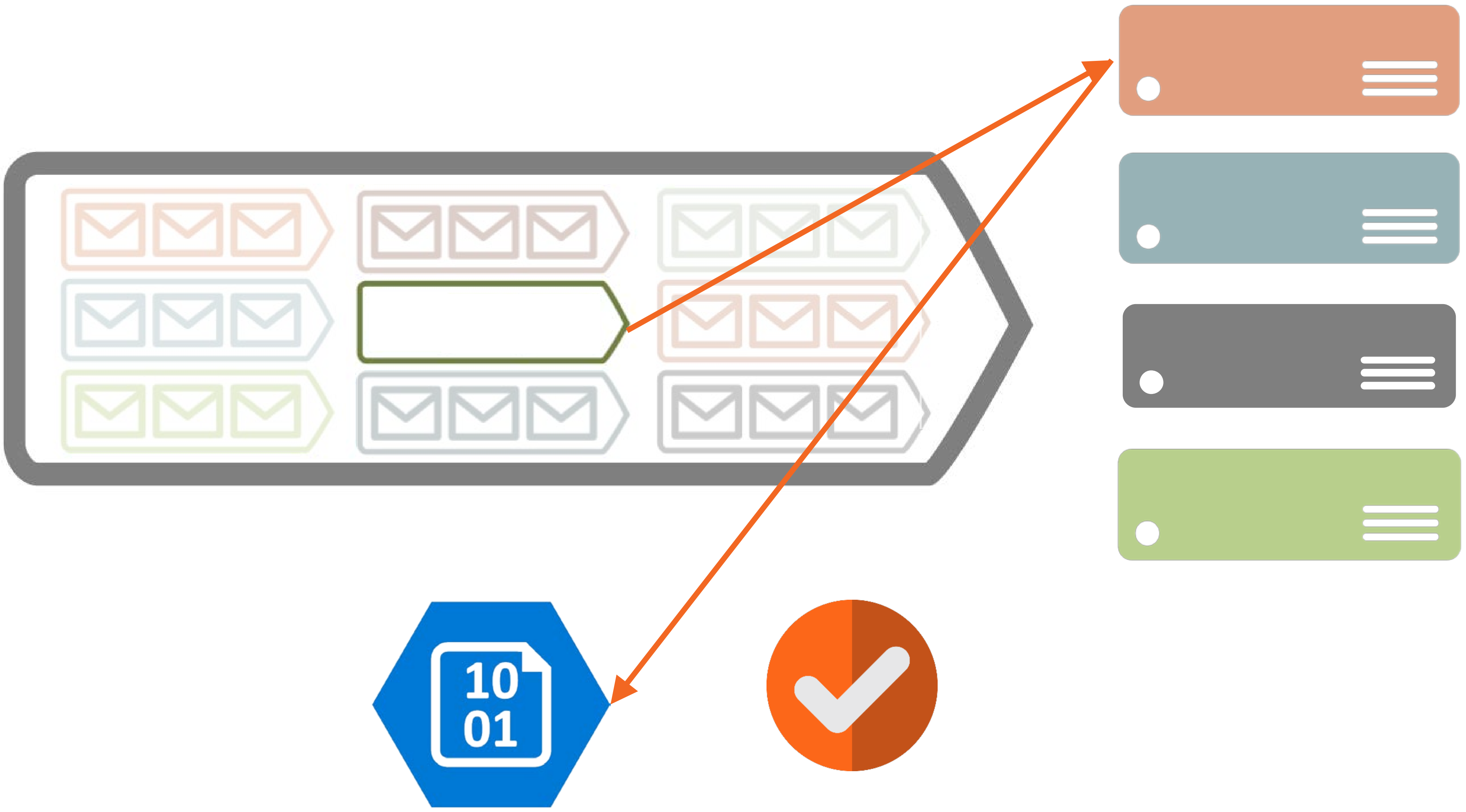




EventProcessorHost


10
01


10
01


10
01

Demo: IEventProcessor

EventReceiver gets events

IEventProcessor handles events

```
_eventProcessorHost = new EventProcessorHost(
    _hostName,_eventHubName, _consumerGroupName,
    _hubConnectionString, _checkpointConnectionString);

await _eventProcessorHost.RegisterEventProcessorAsync
              <DeepStorageEventProcessor>(processorOptions);
```

EventReceiver

Initiaize EventProcessorHost & register IEventProcessor

```
var processorOptions = new EventProcessorOptions
{
    MaxBatchSize = 5000,
    PrefetchCount = 1000
};
```

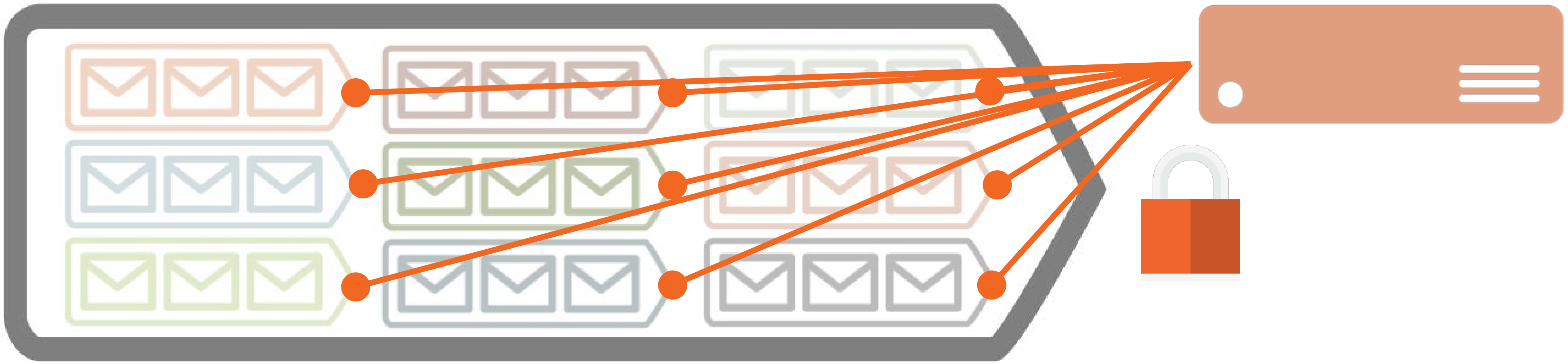
EventProcessorOptions

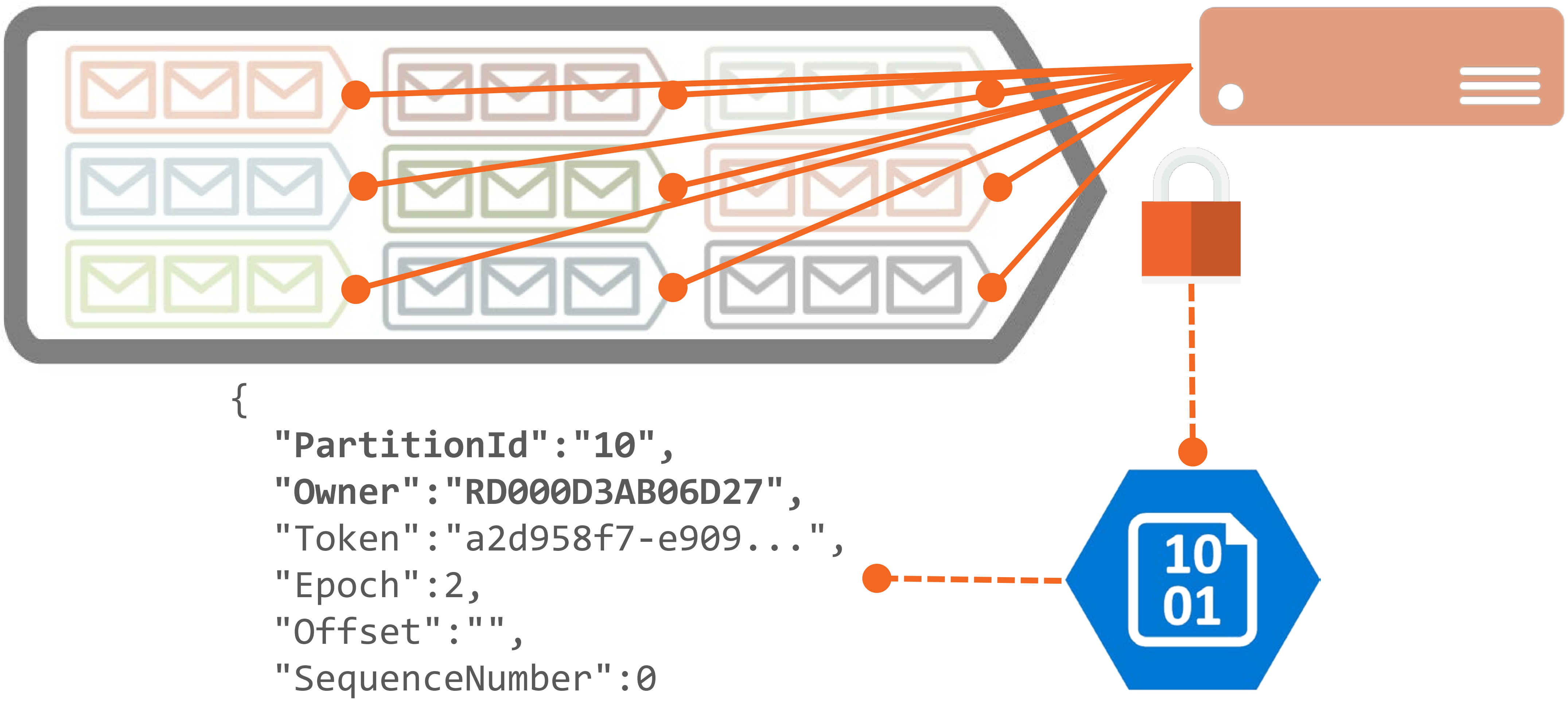
Specify receiver throughput

```
Task OpenAsync(PartitionContext context);
Task CloseAsync(PartitionContext context,
      CloseReason reason);
Task ProcessEventsAsync(PartitionContext context,
      IEnumerable<EventData> messages);
```

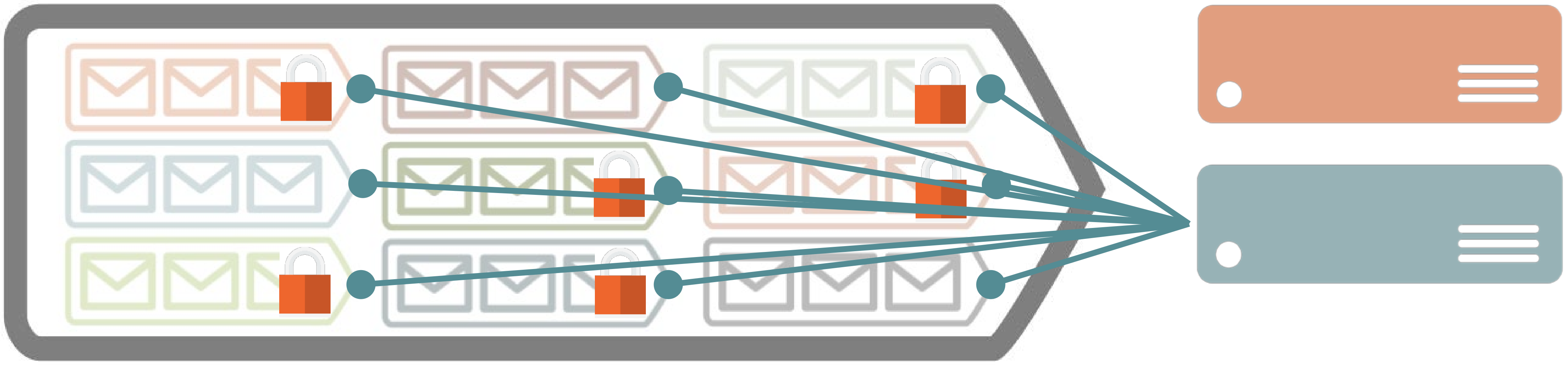
IEventProcessor

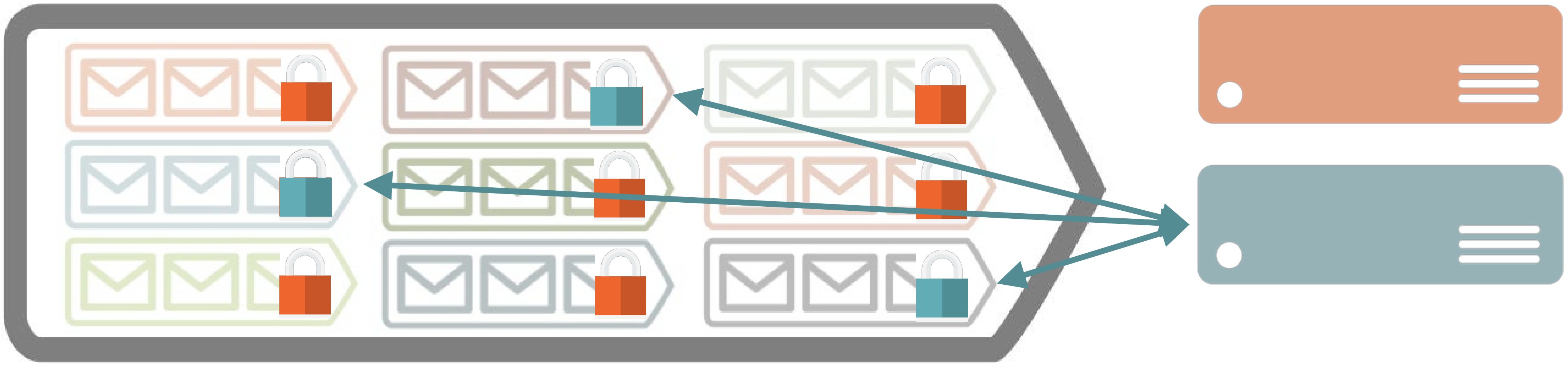
Stateful processor for event batches

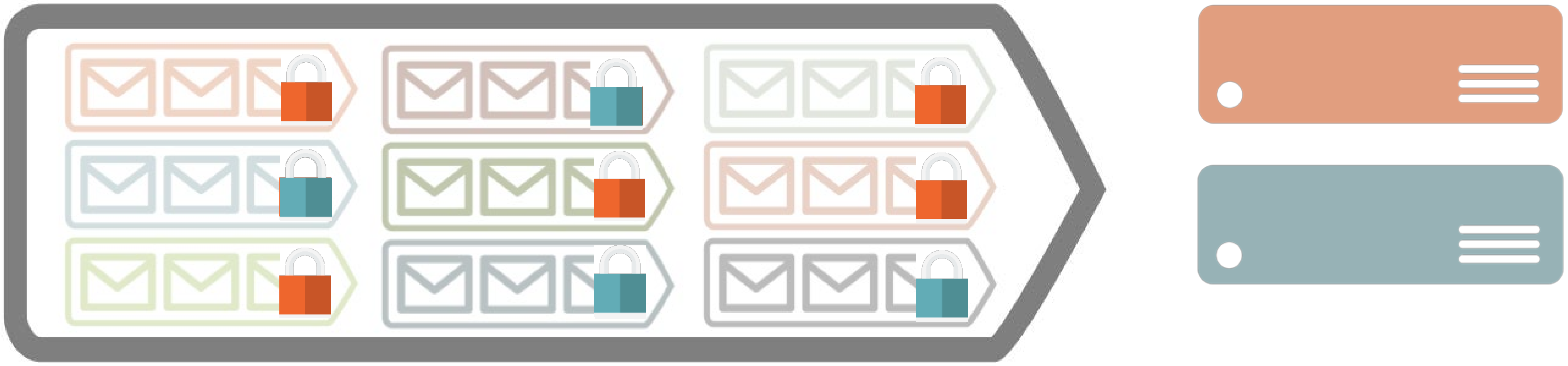


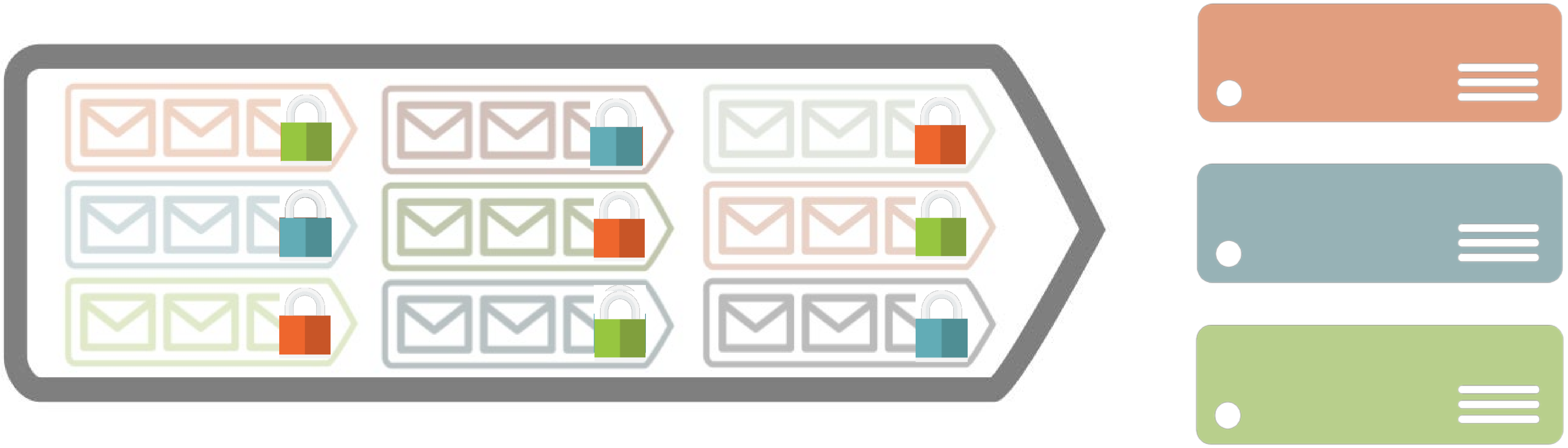


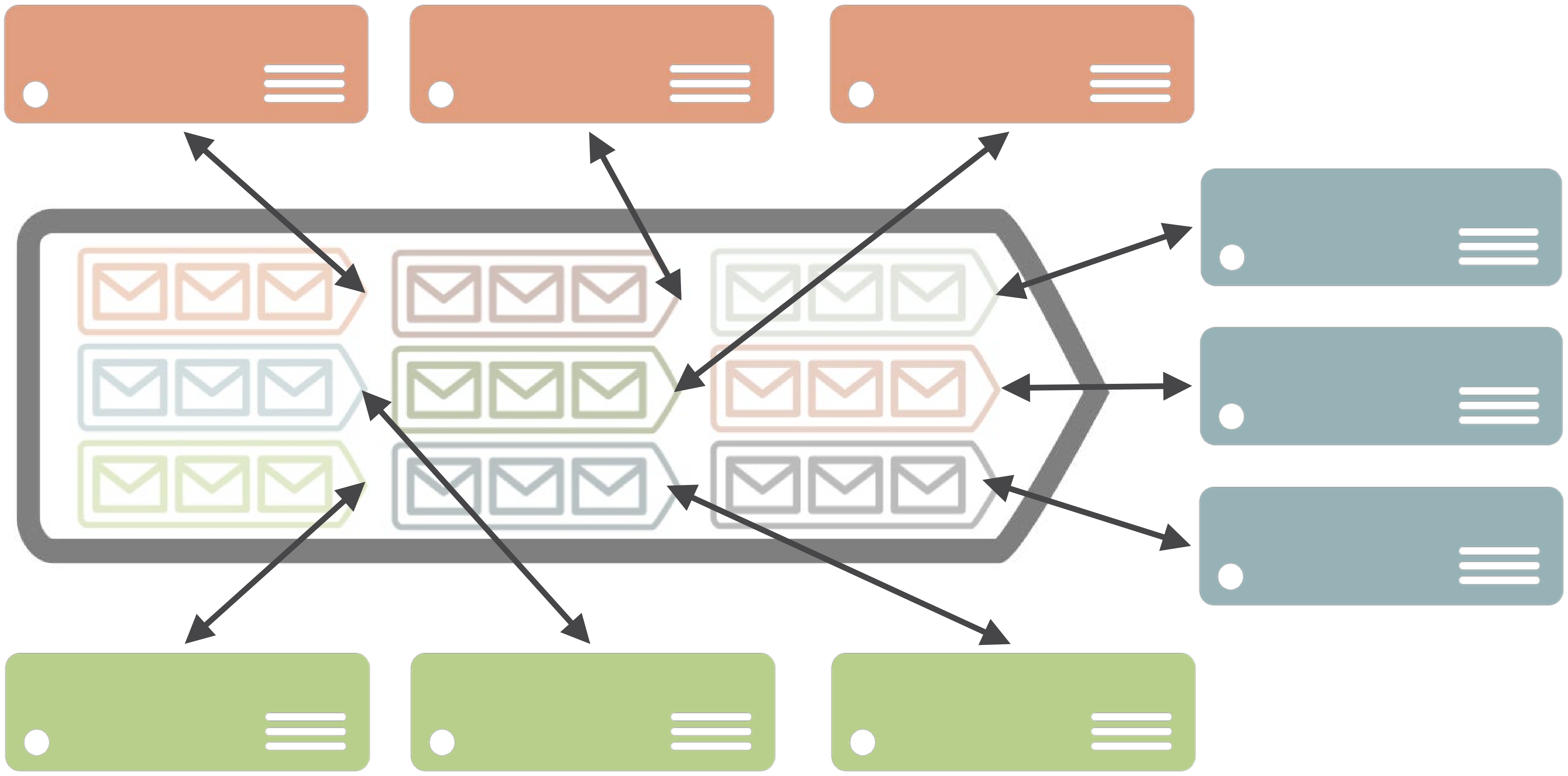
```
{
  "PartitionId":"10",
  "Owner":"RD000D3AB06D27",
  "Token":"a2d958f7-e909...",
  "Epoch":2,
  "Offset":"",
  "SequenceNumber":0
}
```

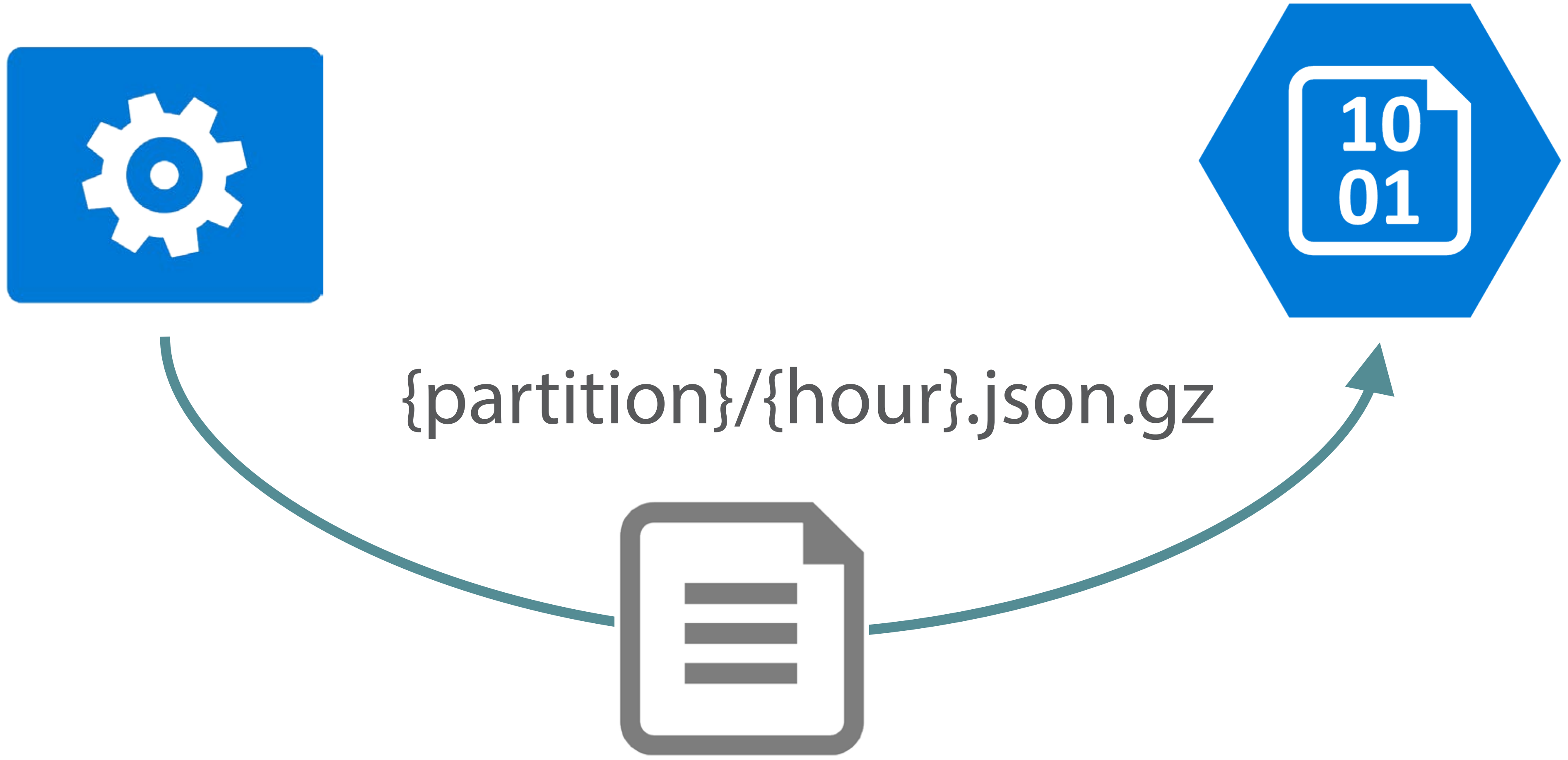


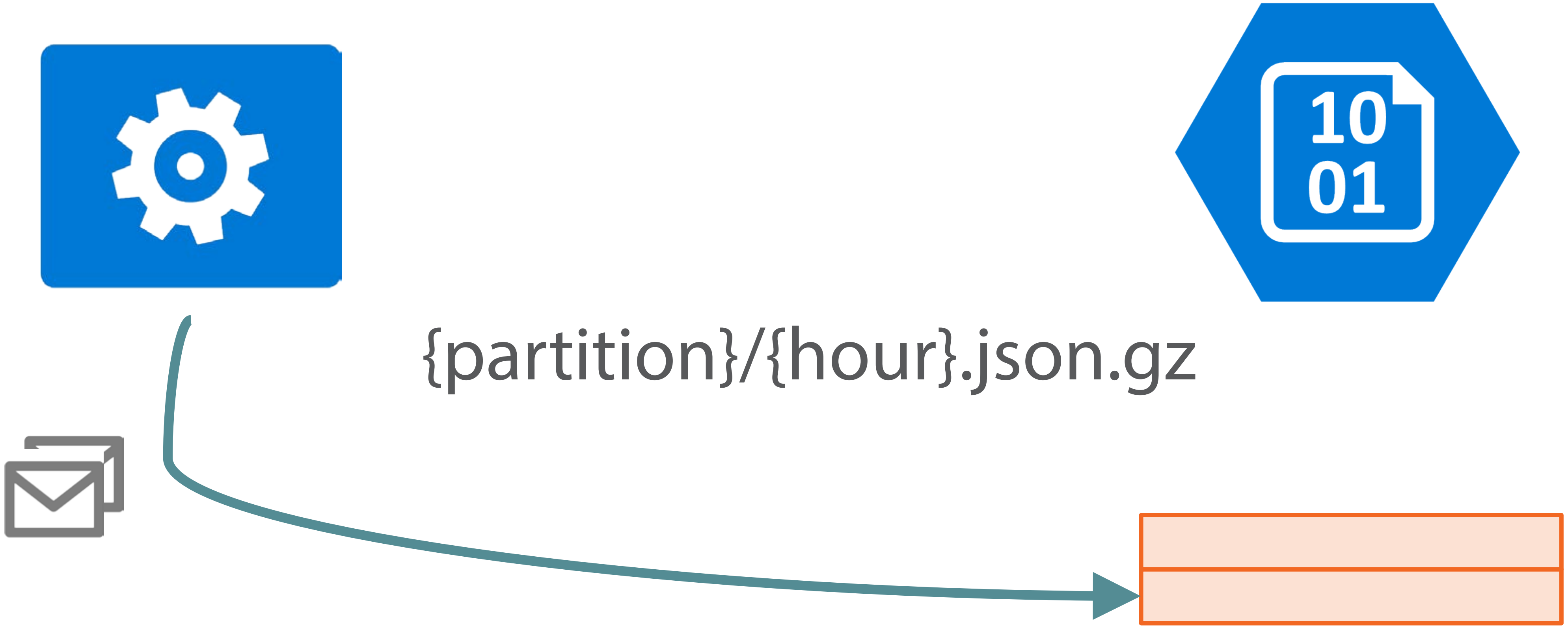


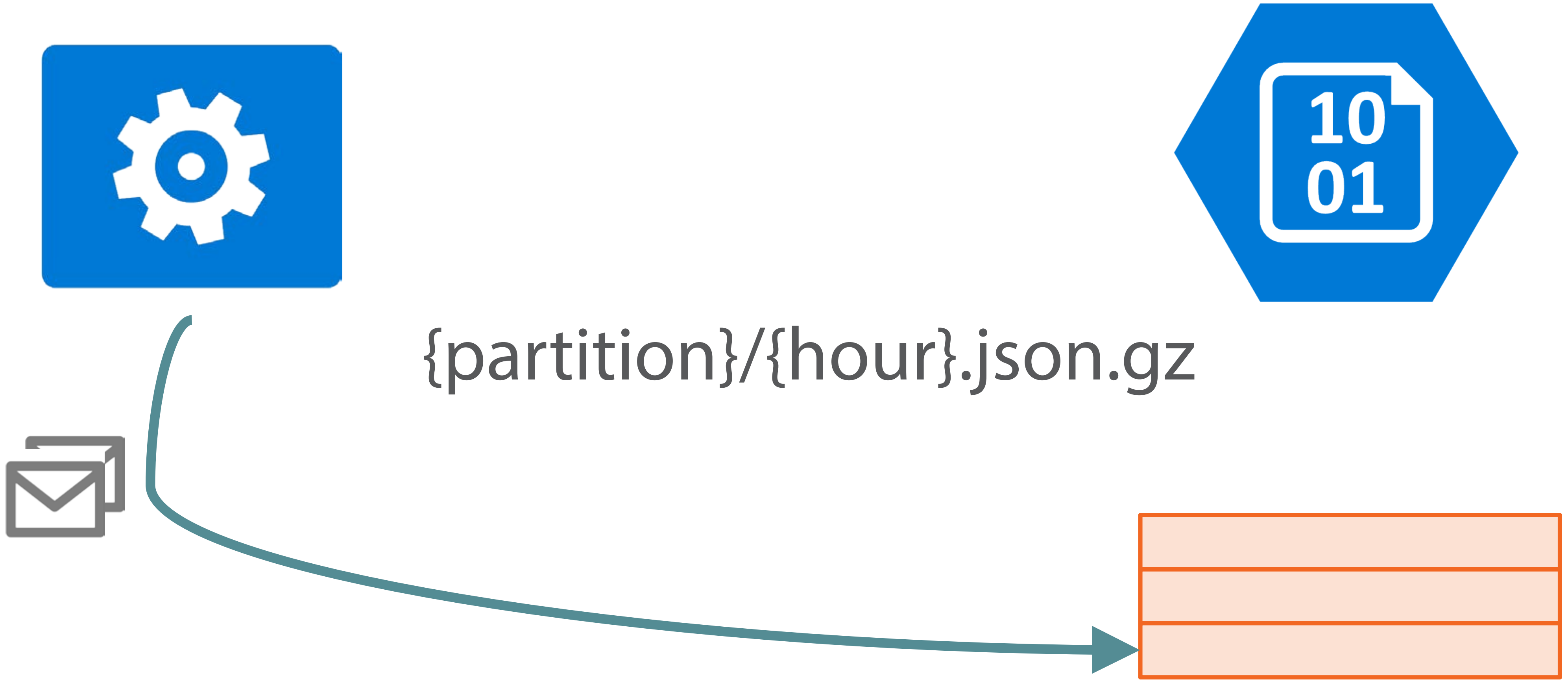


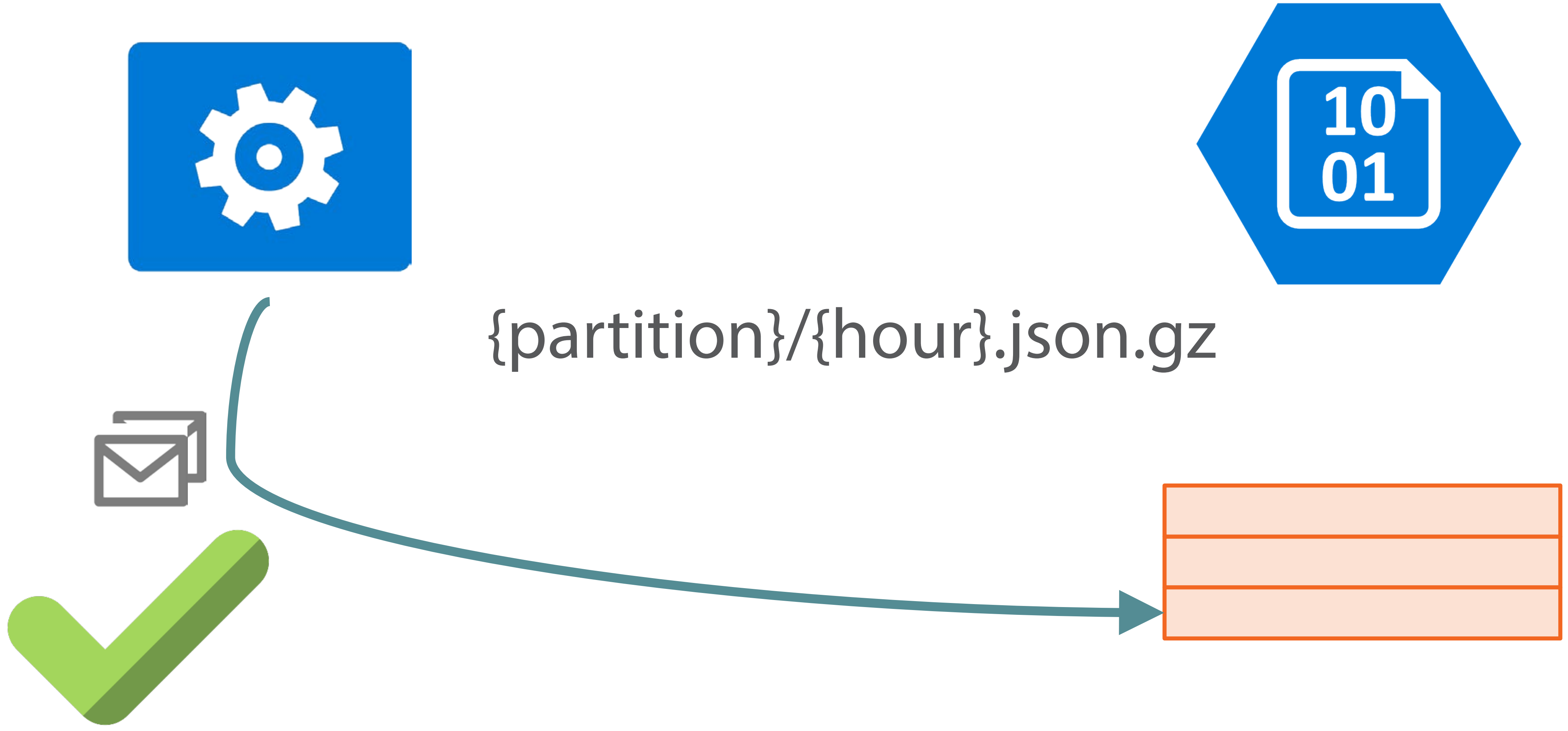




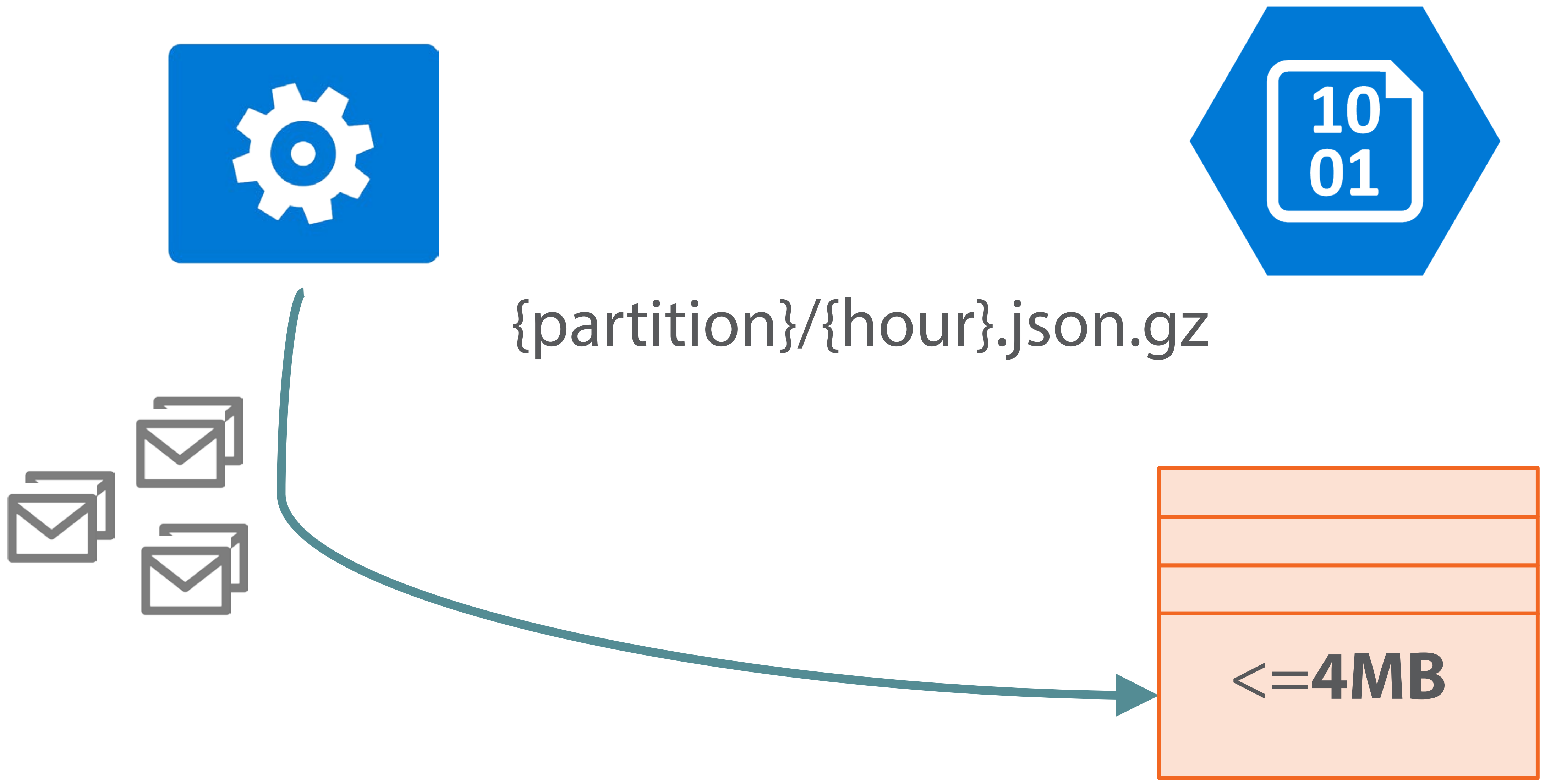

{partition}/{hour}.json.gz
10
01

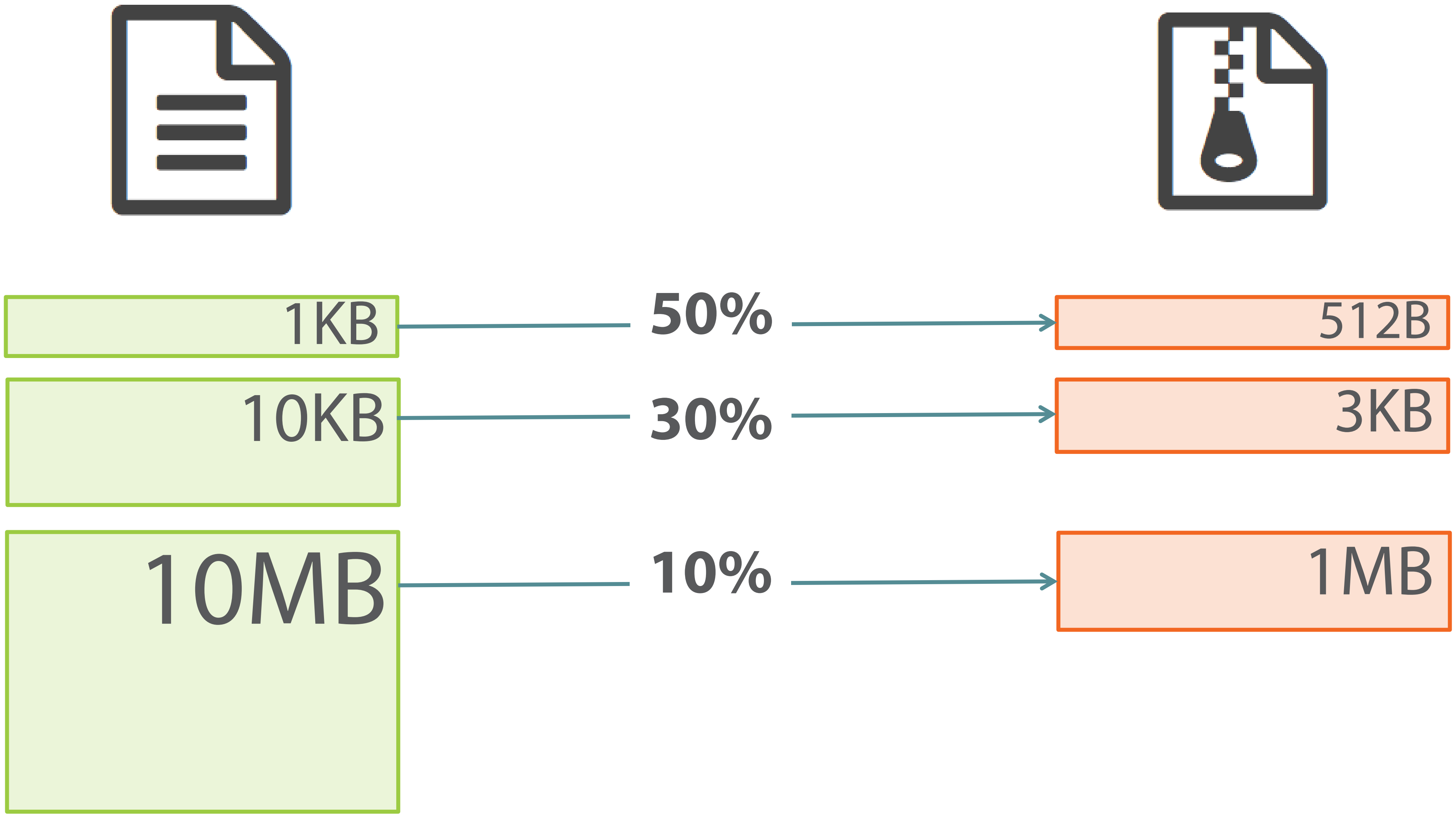

10
01
{partition}/{hour}.json.gz

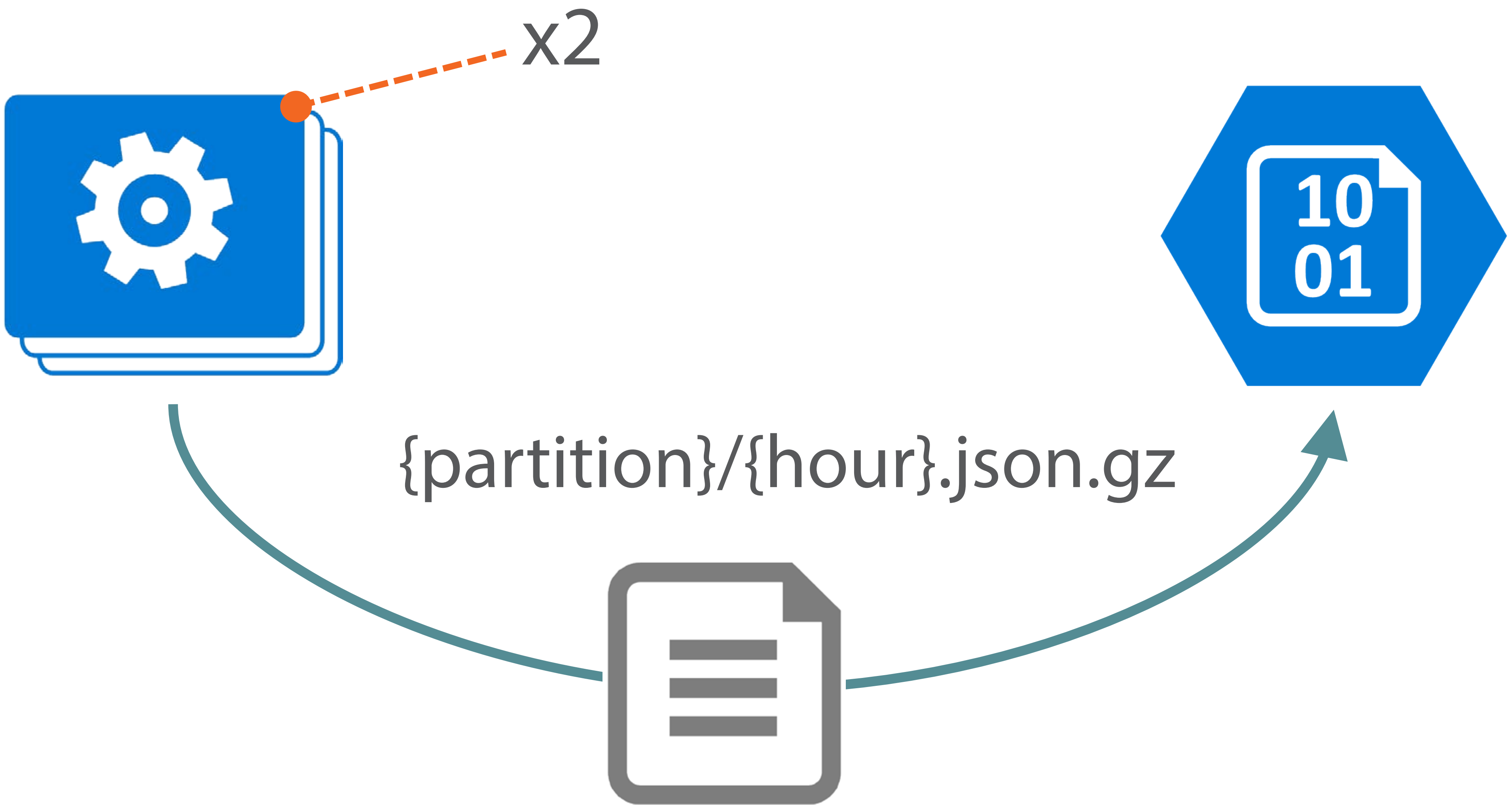

{partition}/{hour}.json.gz

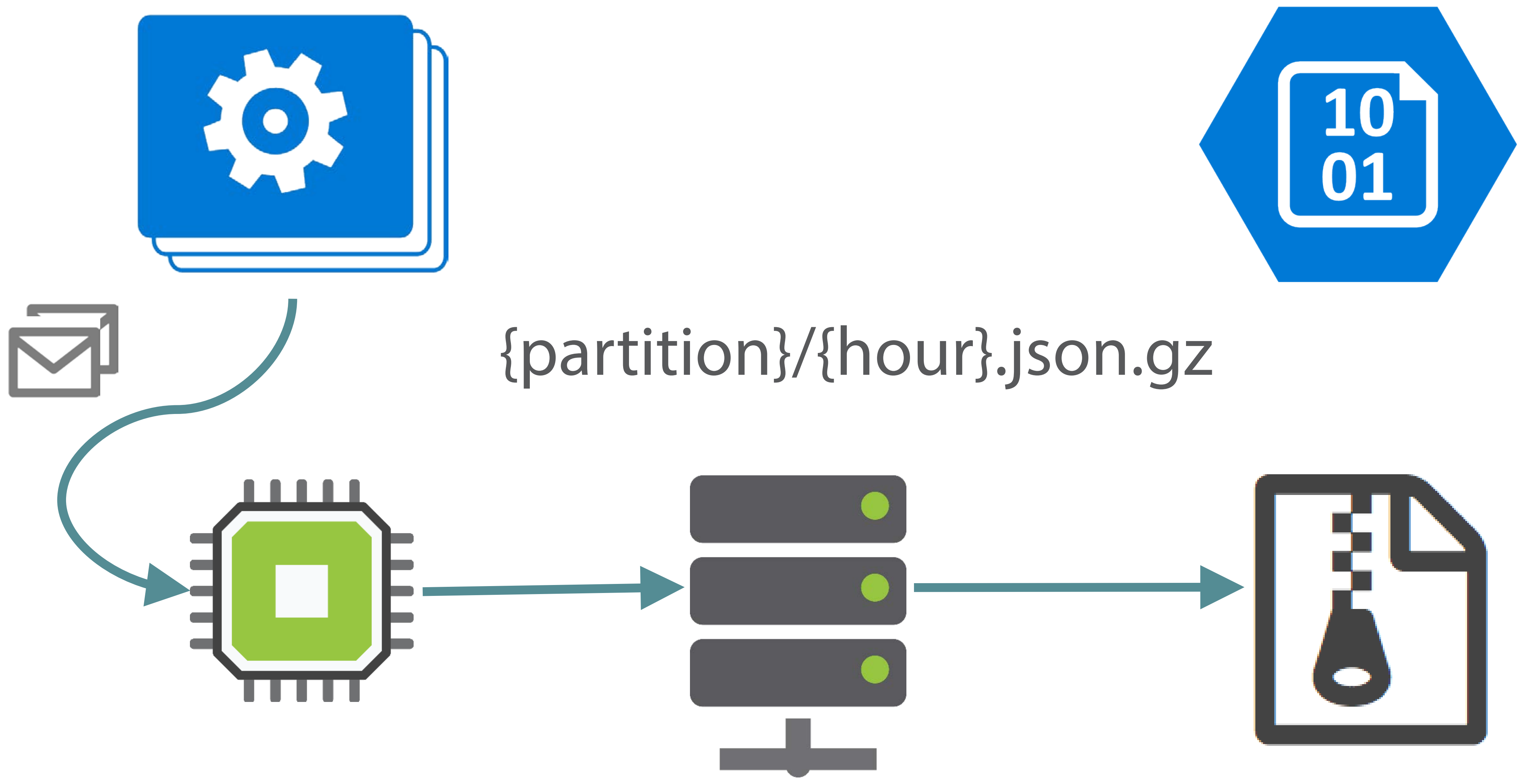

10
01
{partition}/{hour}.json.gz

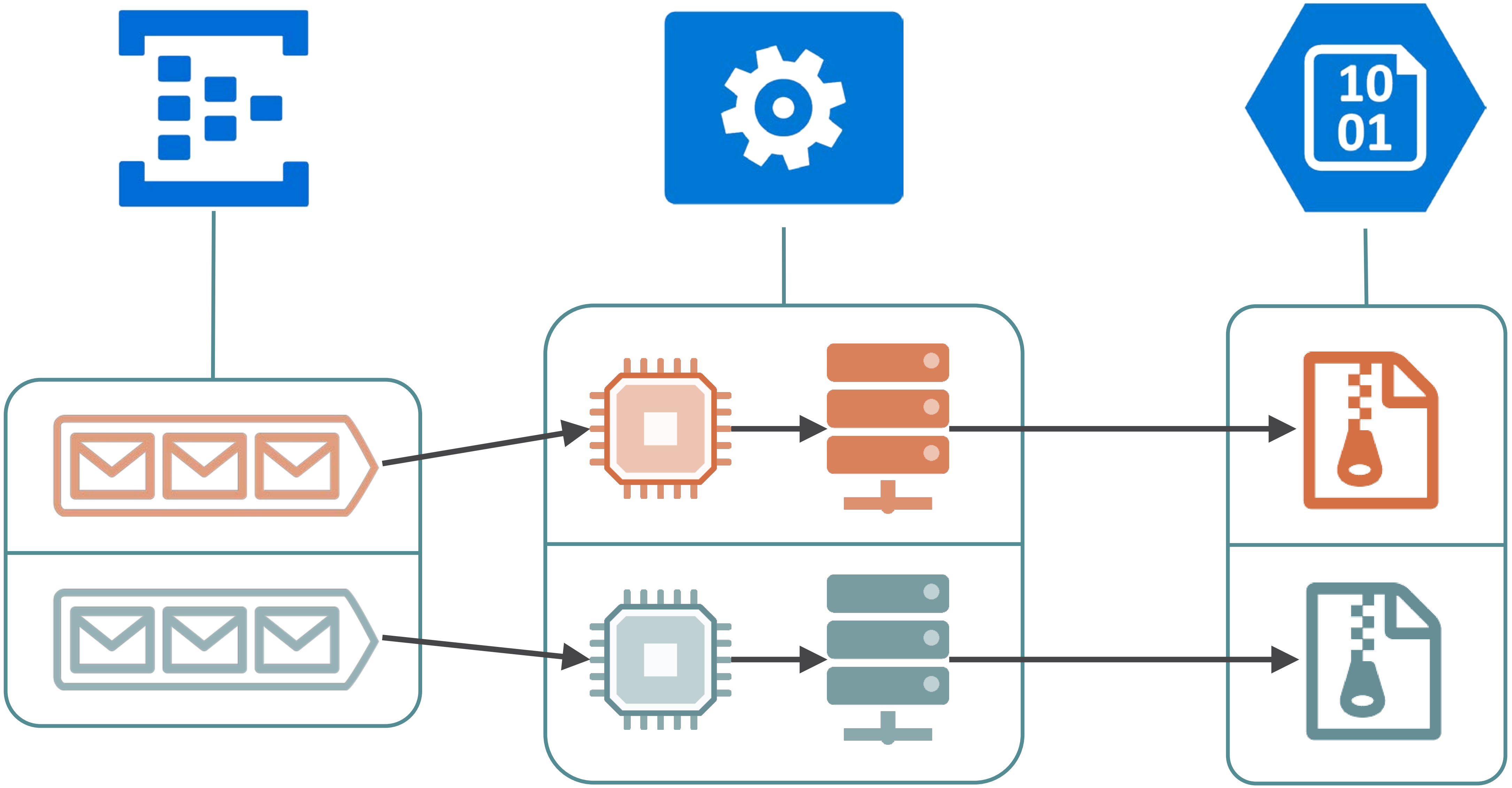
{partition}/{hour}.json.gz

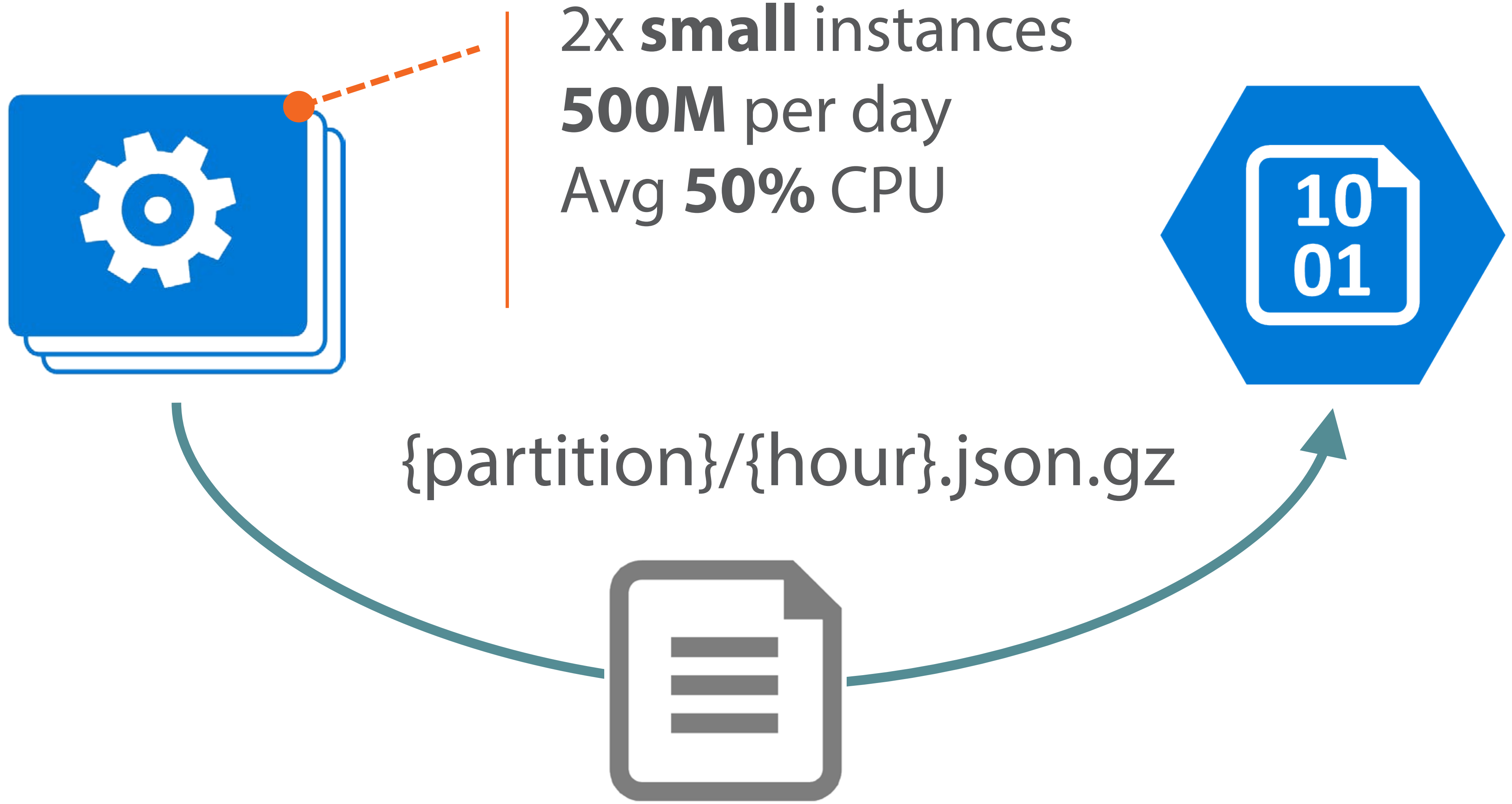

10
01
{partition}/{hour}.json.gz
<=4MB


1KB
50%
512B
10KB
30%
3KB
10MB
10%
1MB

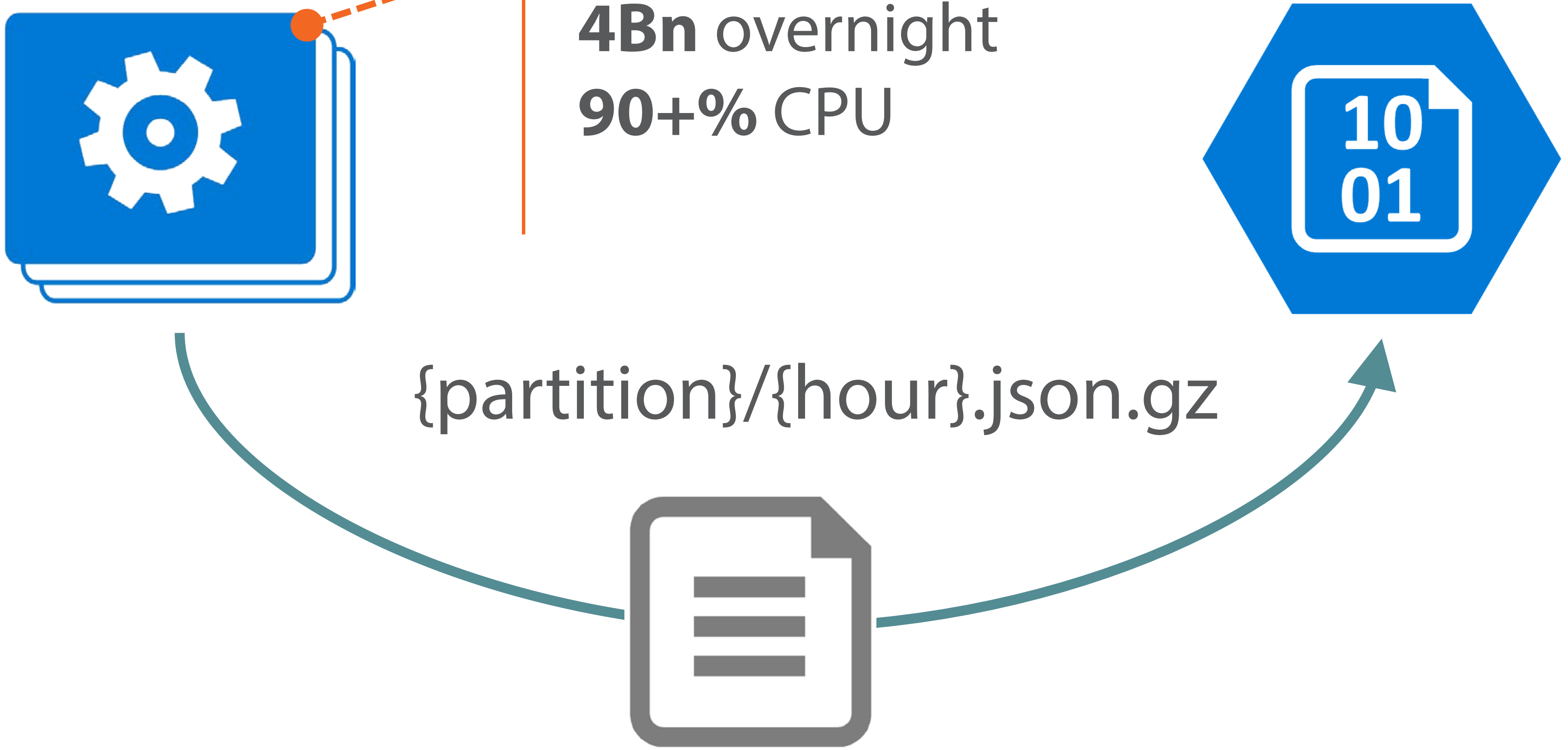

x2
10
01
{partition}/{hour}.json.gz


10
01
{partition}/{hour}.json.gz


10
01


2x **small** instances
500M per day
Avg **50%** CPU
10
01
{partition}/{hour}.json.gz

16x **small** instances
4Bn overnight
90+% CPU



Demo: DeepStorageEventProcessor

IEventStore interface

Multiple buffer abstraction

Event Processor -> IEventStore

```
foreach (EventData eventData in messages)
{
    var store = GetEventStore(eventData, partitionId);
    var bytes = eventData.GetBytes();
    store.Write(bytes);
}
```

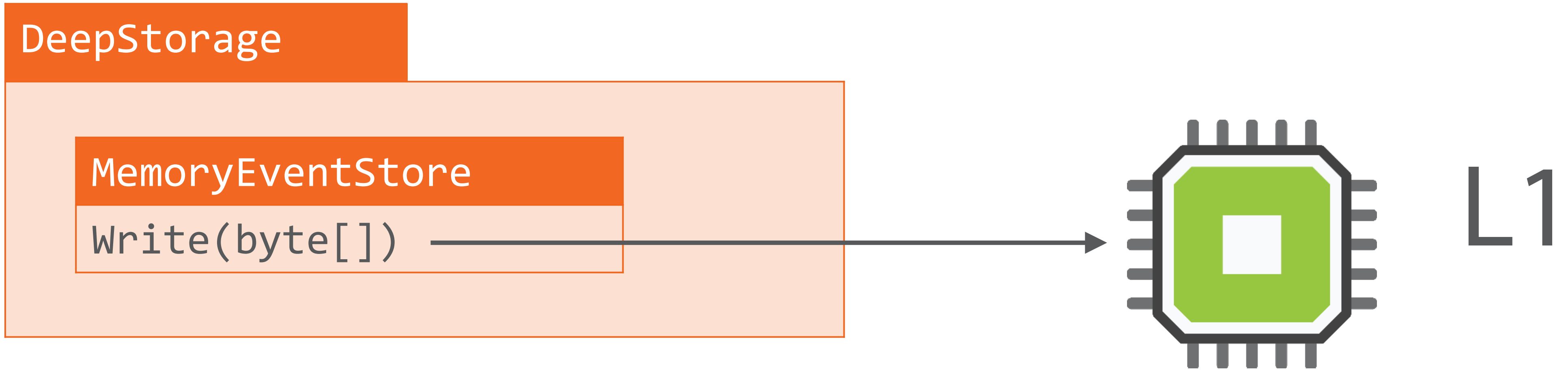
DeepStorageEventProcessor

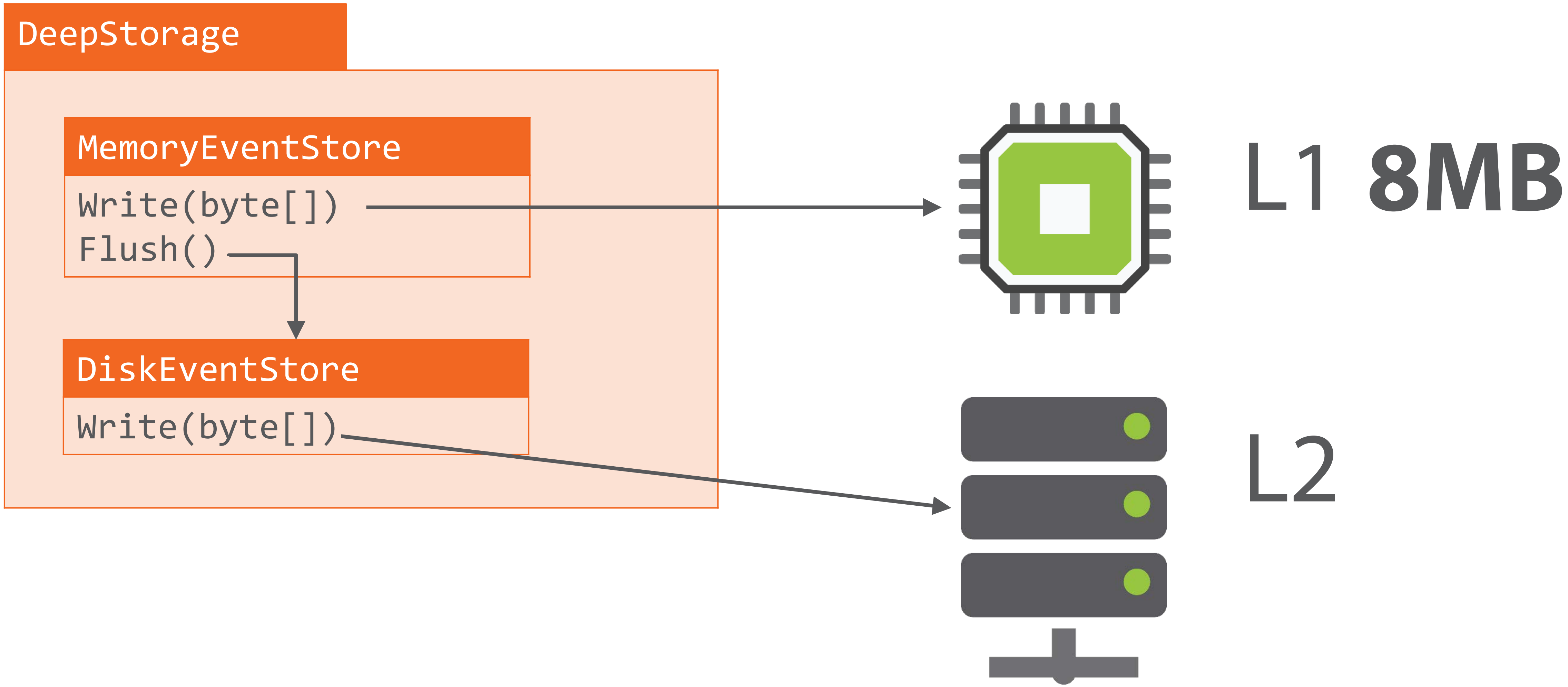
Write events to Level 1 Event Store

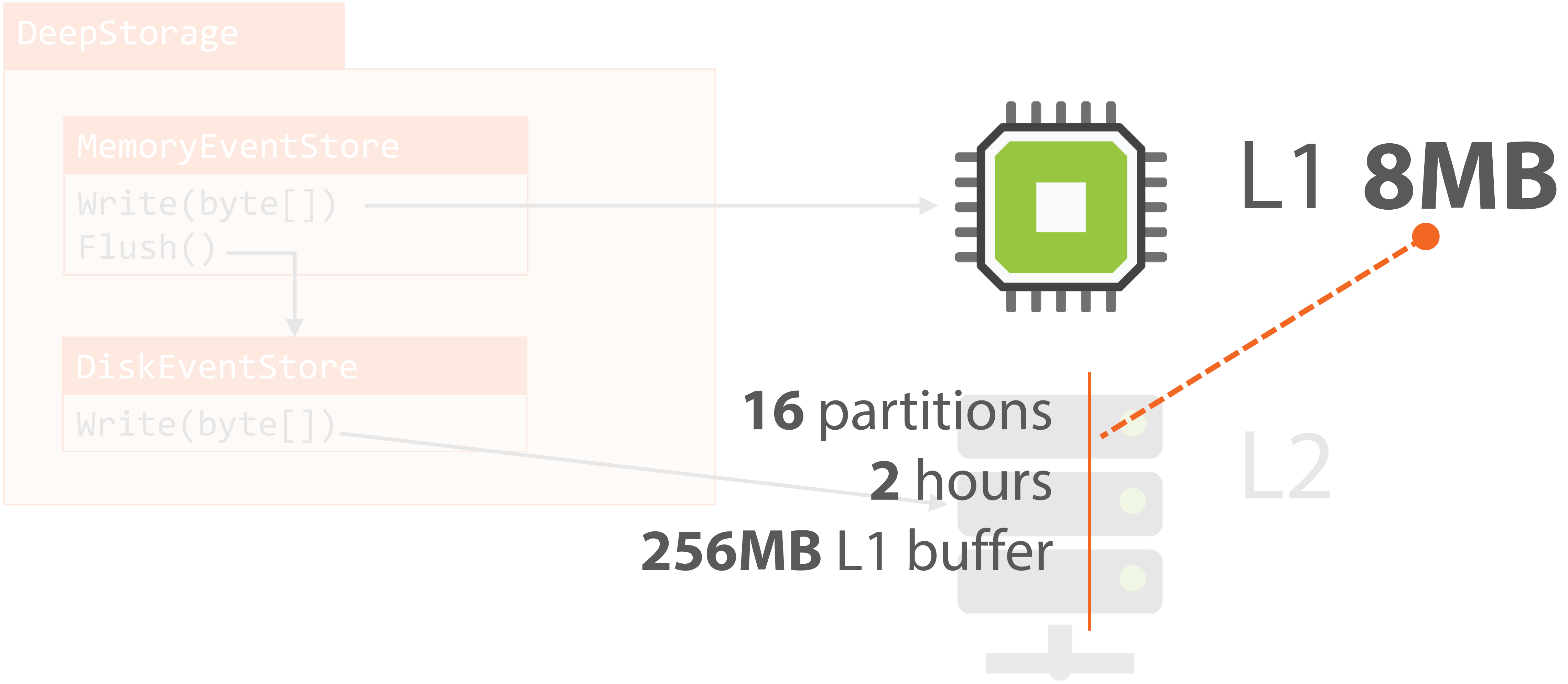
```
var key = string.Format("{0}p{1}", receivedAt, partitionId);
if (!_EventStores.ContainsKey(key)) {
    var store = Container.Instance.Resolve<IEventStore>("1");
    store.Initialise(partitionId, receivedAt);
    _EventStores[key] = store;
}
```

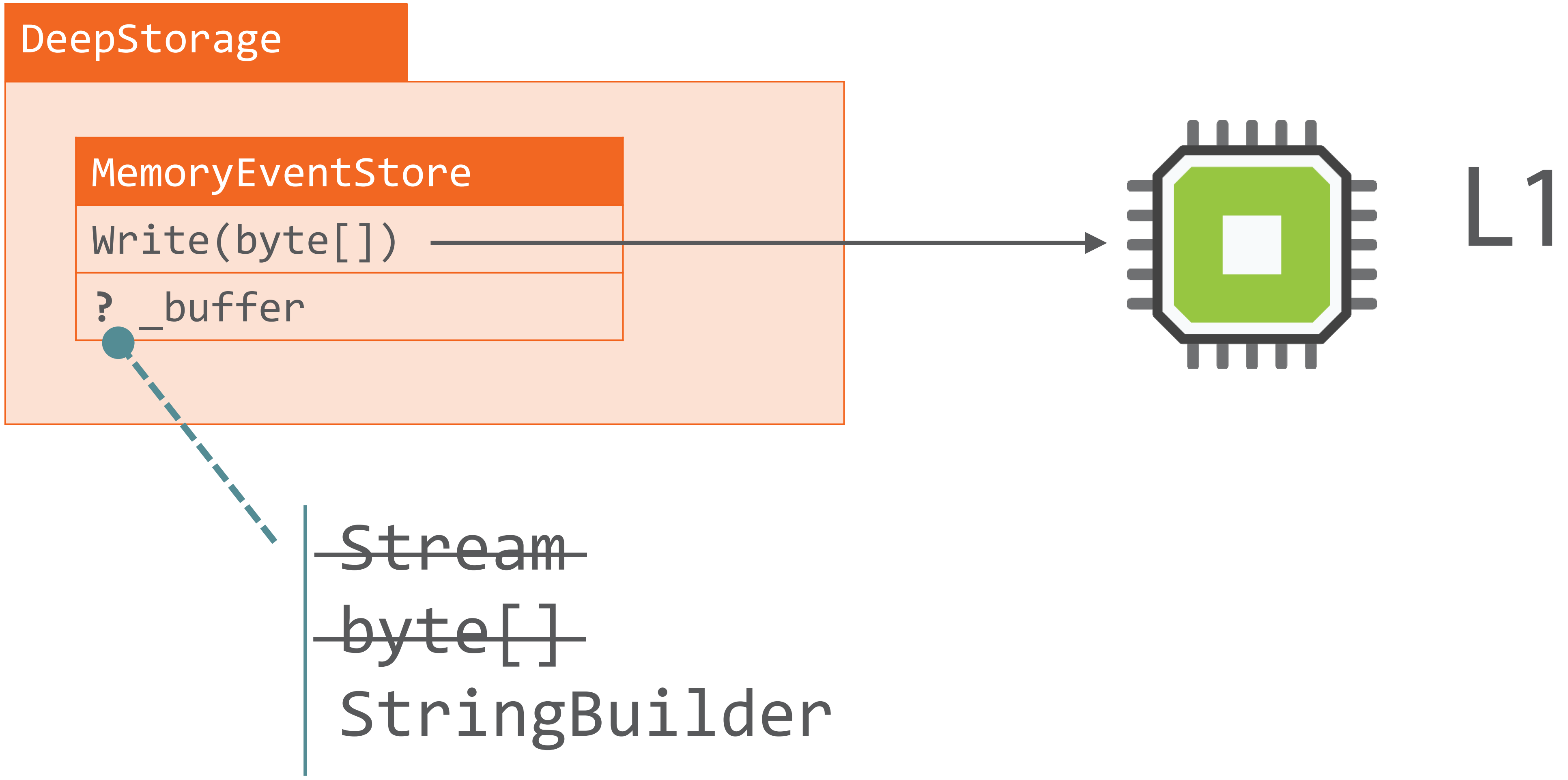
DeepStorageEventProcessor

Event Stores in static ConcurrentDictionary


DeepStorage
MemoryEventStore
Write(byte[])
L1


DeepStorage
MemoryEventStore
Write(byte[])
Flush()
DiskEventStore
Write(byte[])
L1 8MB
L2


DeepStorage
MemoryEventStore
Write(byte[])
Flush()
DiskEventStore
Write(byte[])
L1 **8MB**
16 partitions
2 hours
256MB L1 buffer
L2


DeepStorage
MemoryEventStore
Write(byte[])
? _buffer
L1
~~Stream~~
~~byte[]~~
StringBuilder

Demo: MemoryEventStore

IEventStore implementation

In-memory store

StringBuilder buffer

```
var json = Encoding.UTF8.GetString(value);
try
{
    _lock.Wait();
    _buffer.AppendLine(json);
}
```

MemoryEventStore

Store events in StringBuilder

```
if (_buffer.Length + byteCount > MaxBufferSize)
{
    Flush();
}
```

MemoryEventStore

Flush before buffer size exceeded

```
try
{
    _lock.Wait();
    block = _buffer.ToString();
    _buffer.Clear();
}
```

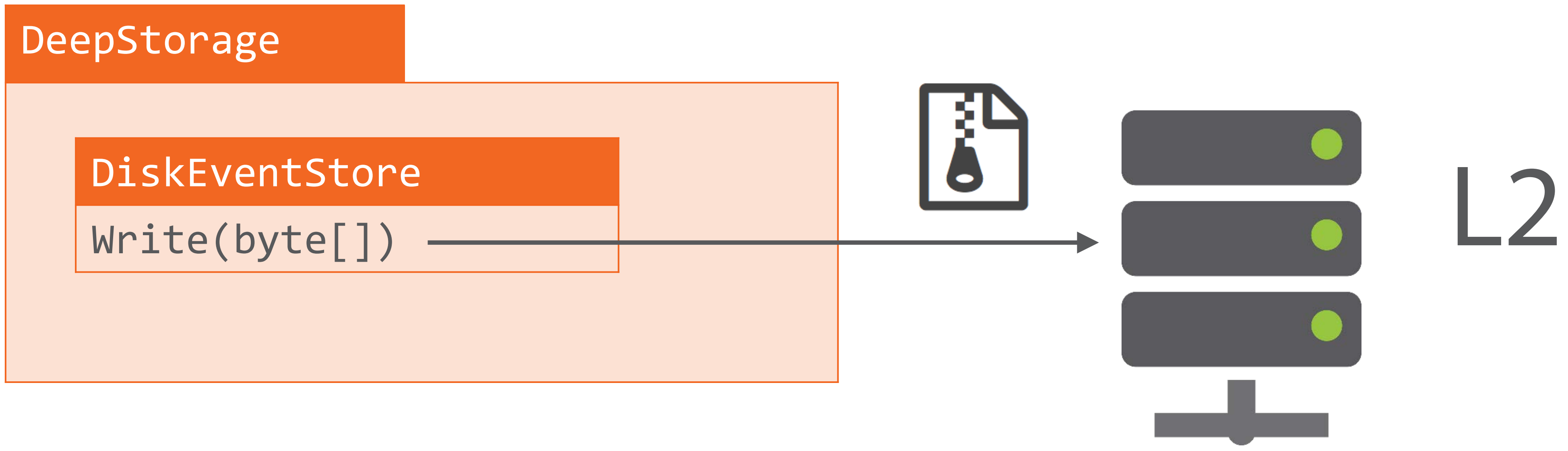
MemoryEventStore

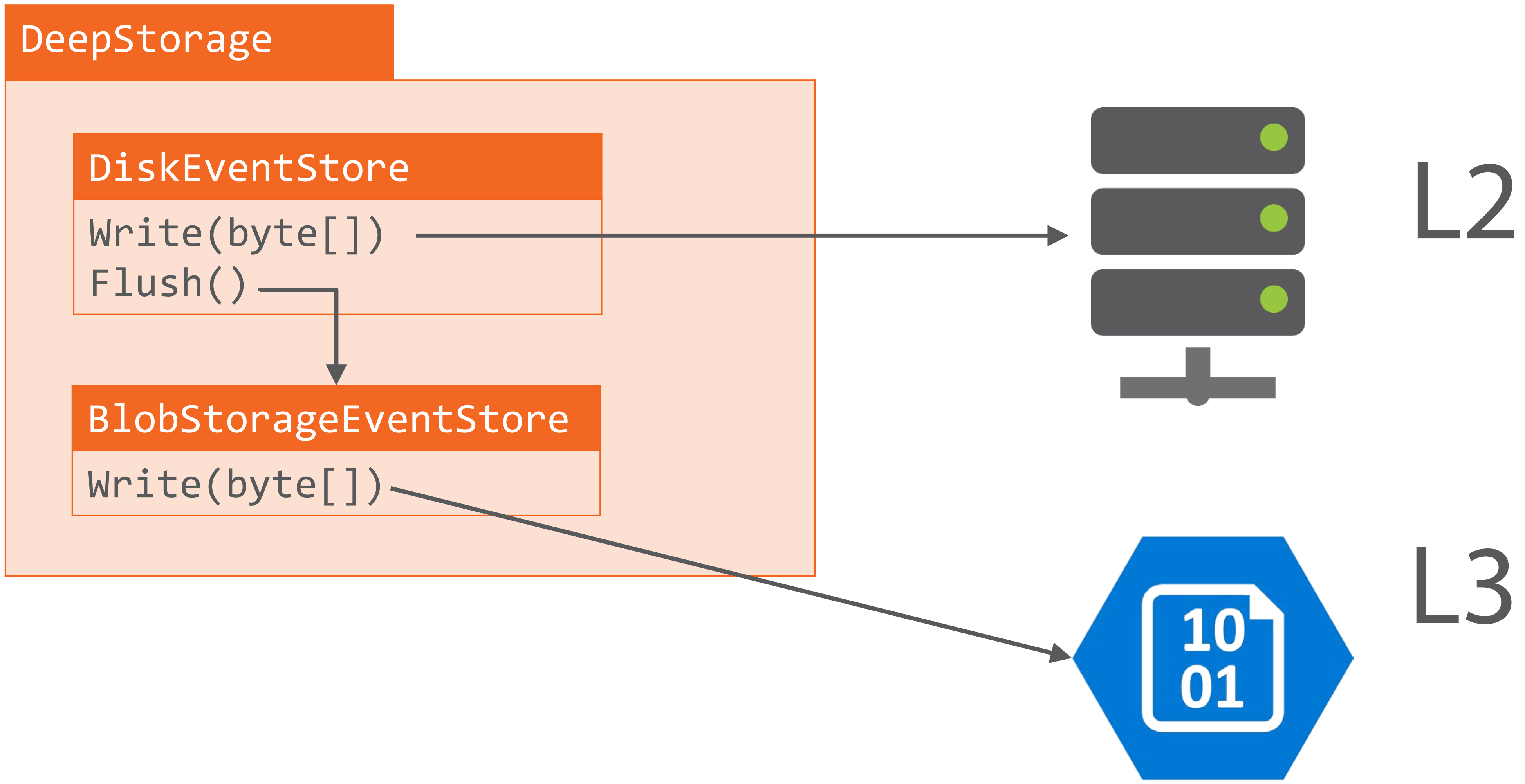
Read whole StringBuilder on flush

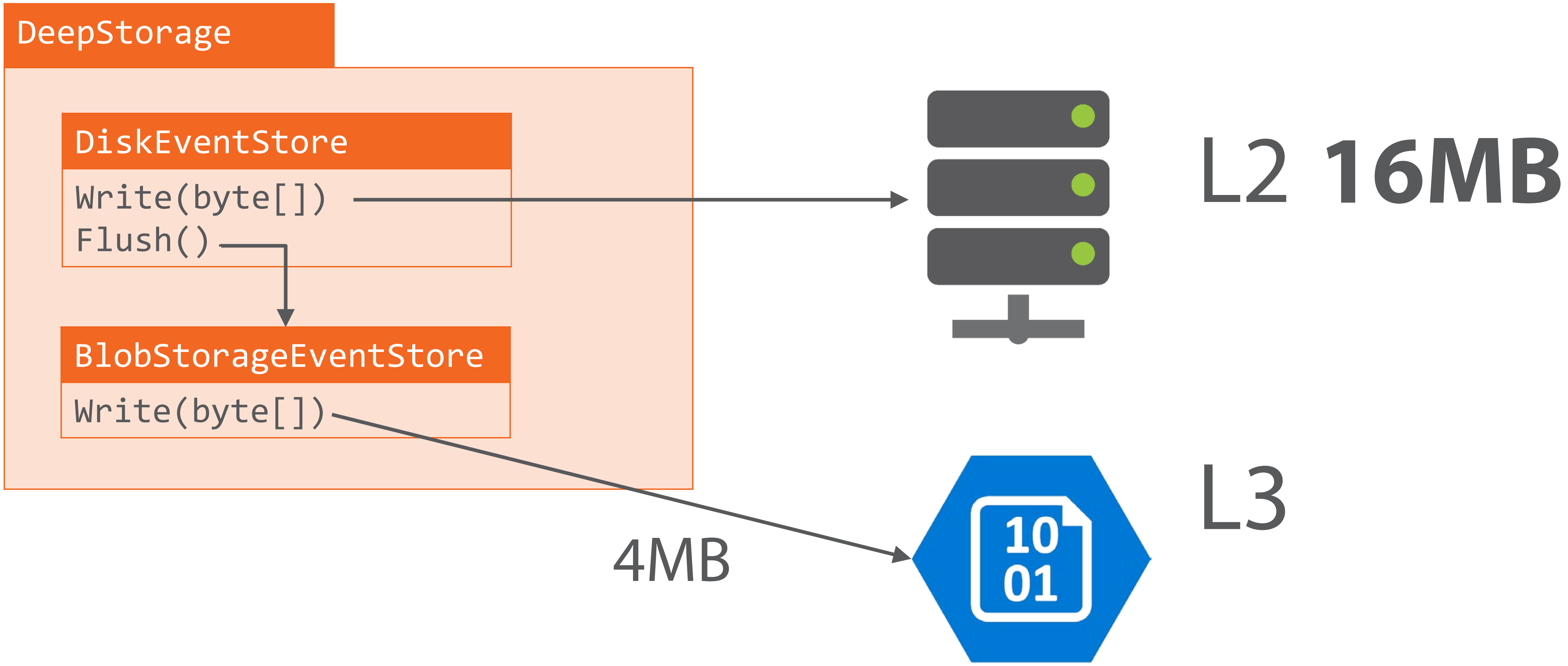
```
if (block.Length > 0)
{
    var data = Encoding.UTF8.GetBytes(block);
    Task.Factory.StartNew(() => _nextStore.Write(data));
}
```

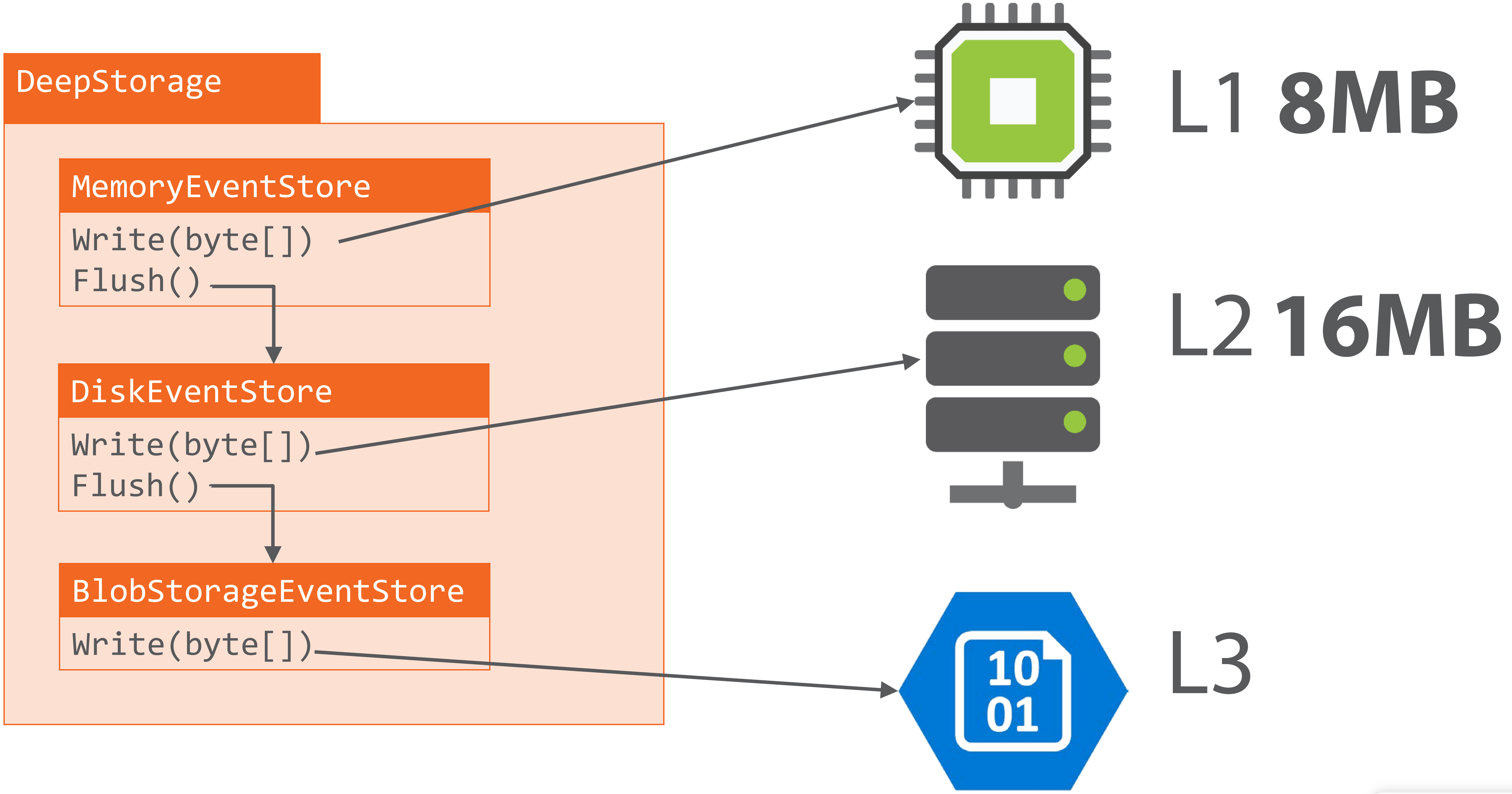
MemoryEventStore

New task, writing to L2 store


DeepStorage
DiskEventStore
Write(byte[])
L2


DeepStorage
DiskEventStore
Write(byte[])
Flush()
BlobStorageEventStore
Write(byte[])
L2
10
01
L3


DeepStorage
DiskEventStore
Write(byte[])
Flush()
BlobStorageEventStore
Write(byte[])
L2 16MB
4MB
L3
10
01


DeepStorage
MemoryEventStore
Write(byte[])
Flush()
DiskEventStore
Write(byte[])
Flush()
BlobStorageEventStore
Write(byte[])
L1 8MB
L2 16MB
10
01
L3

Demo: DiskEventStore

Compresses in memory

Appends GZip to disk

Inherits EventStoreBase

```
public override void Write(byte[] data)
{
    var compressedStream = new MemoryStream();
    using (var inputStream = new MemoryStream(data))
    {
```

DiskEventStore

Write gets flushed events from MemoryEventStore

```
using (var compressionStream = new
GZipOutputStream(compressedStream))
{
    compressionStream.SetLevel(9);
    inputStream.CopyTo(compressionStream);
    compressionStream.Flush();
}
```

DiskEventStore

Compress events in memory

```
_lock.Wait();
using (var outputStream = File.OpenWrite(_filePath))
{
  outputStream.Position = outputStream.Length;
  outputStream.Write(compressedData, 0, compressedData.Length);
  outputStream.Flush();
}
```

DiskEventStore

Append compressed events to file

```
data = File.ReadAllBytes(_filePath);
File.Delete(_filePath);
using (File.Create(_filePath)) { }
```

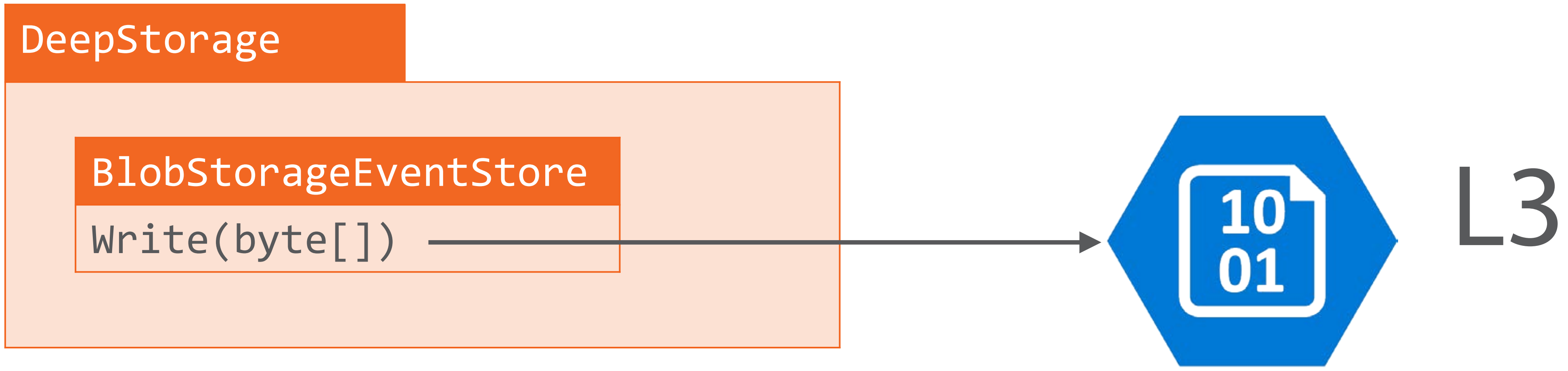
DiskEventStore

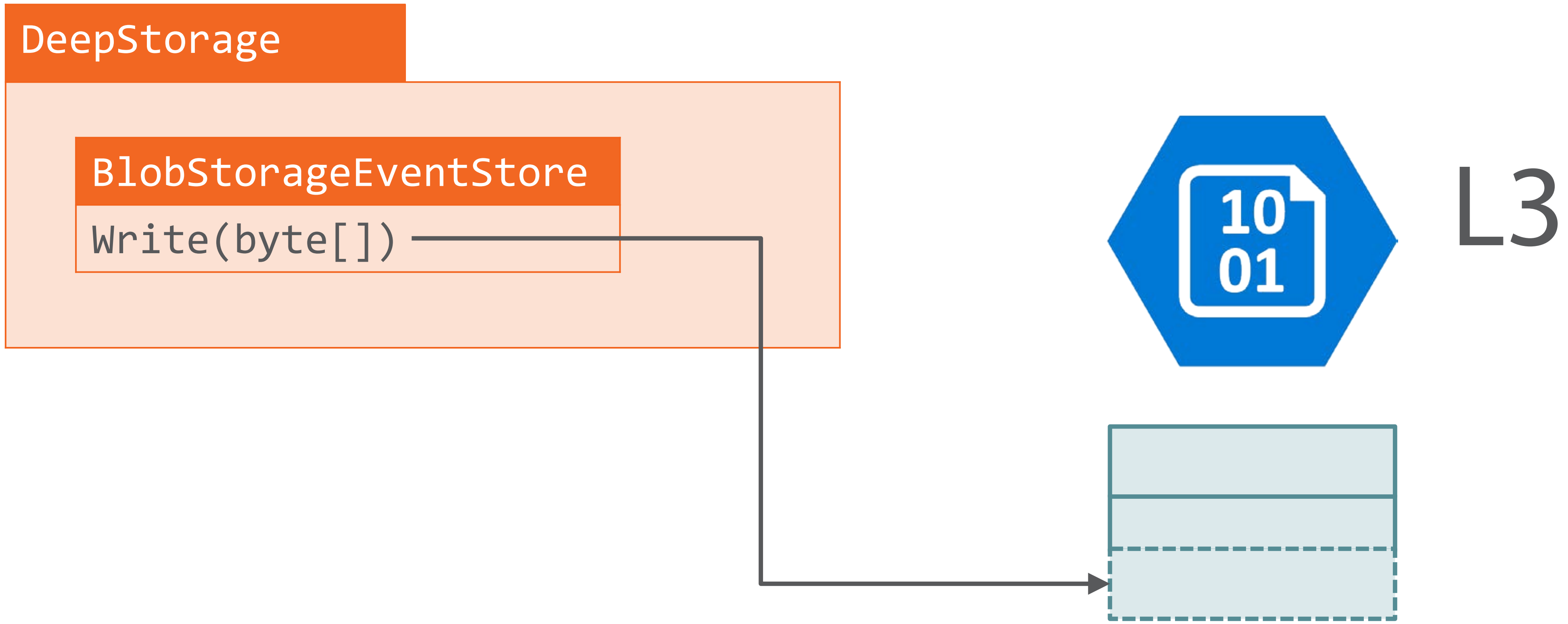
Flush reads & resets file

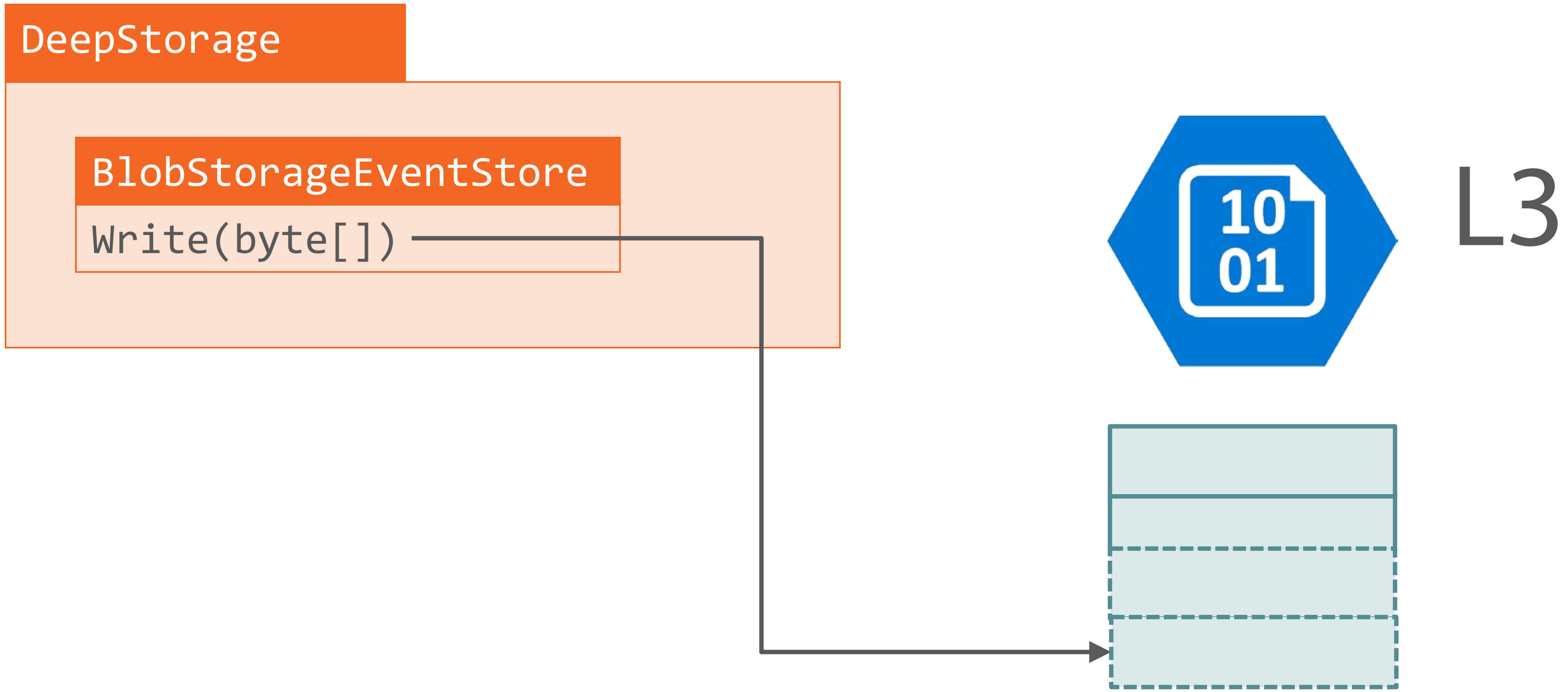
```
if (data.Length > 0)
{
    _startedTasks.Add(Task.Factory.StartNew(
                         () => NextStore.Write(data)));
}
```

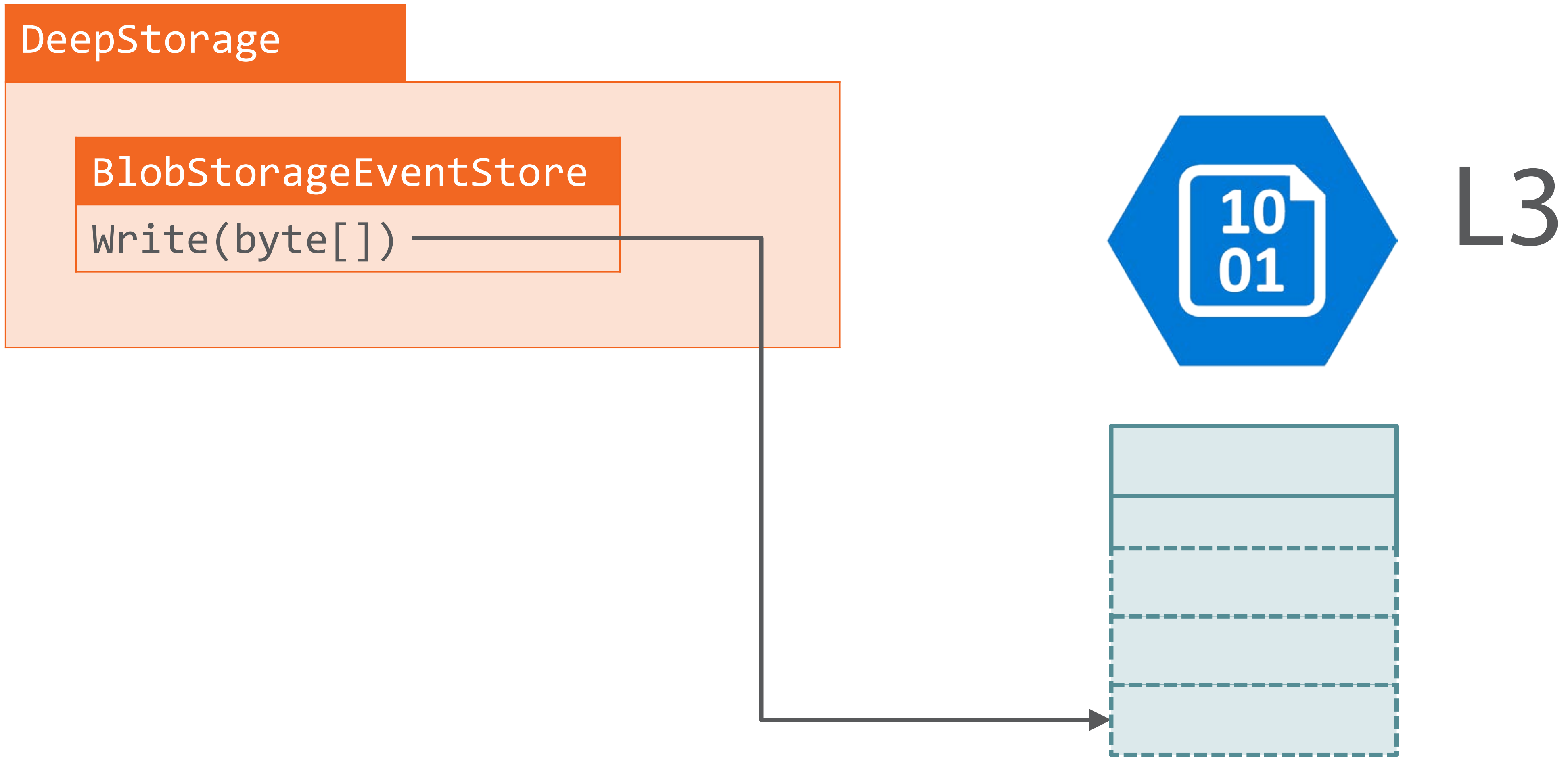
DiskEventStore

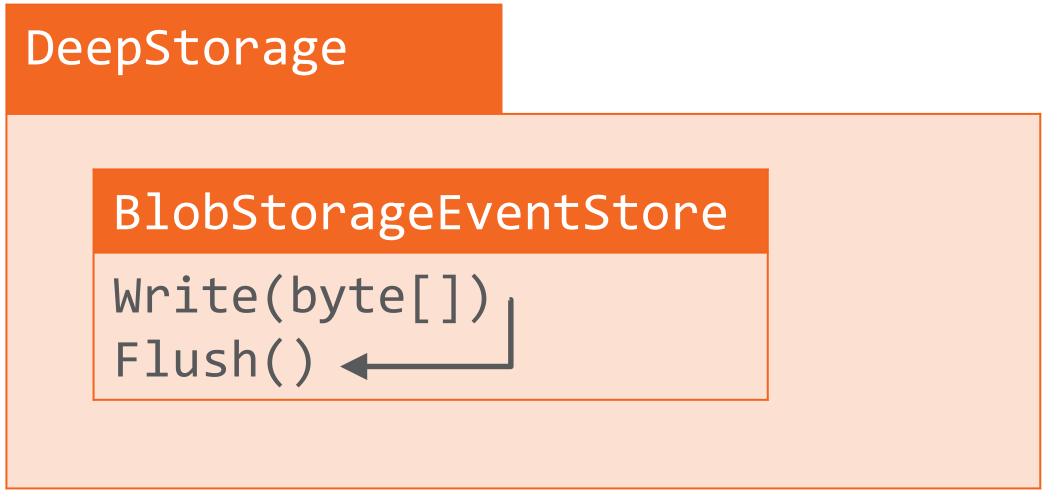
Write to next store asynchronously

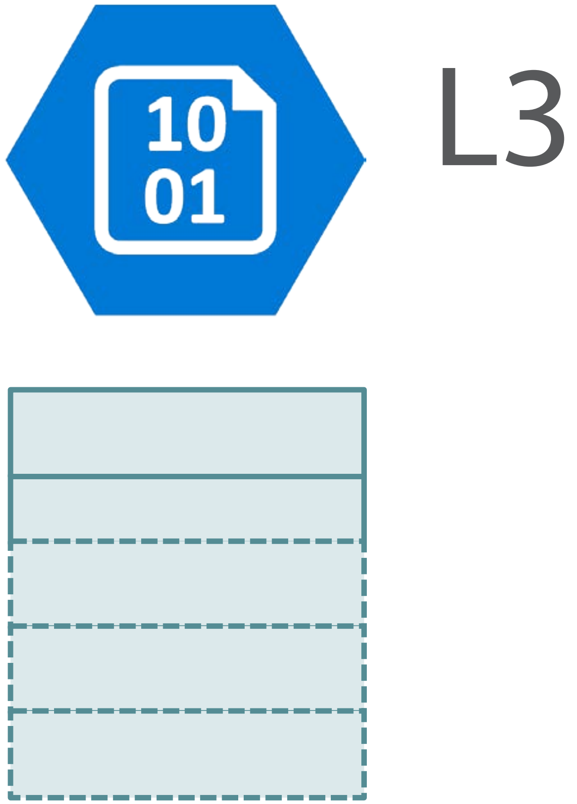

DeepStorage
BlobStorageEventStore
Write(byte[])
10
01
L3

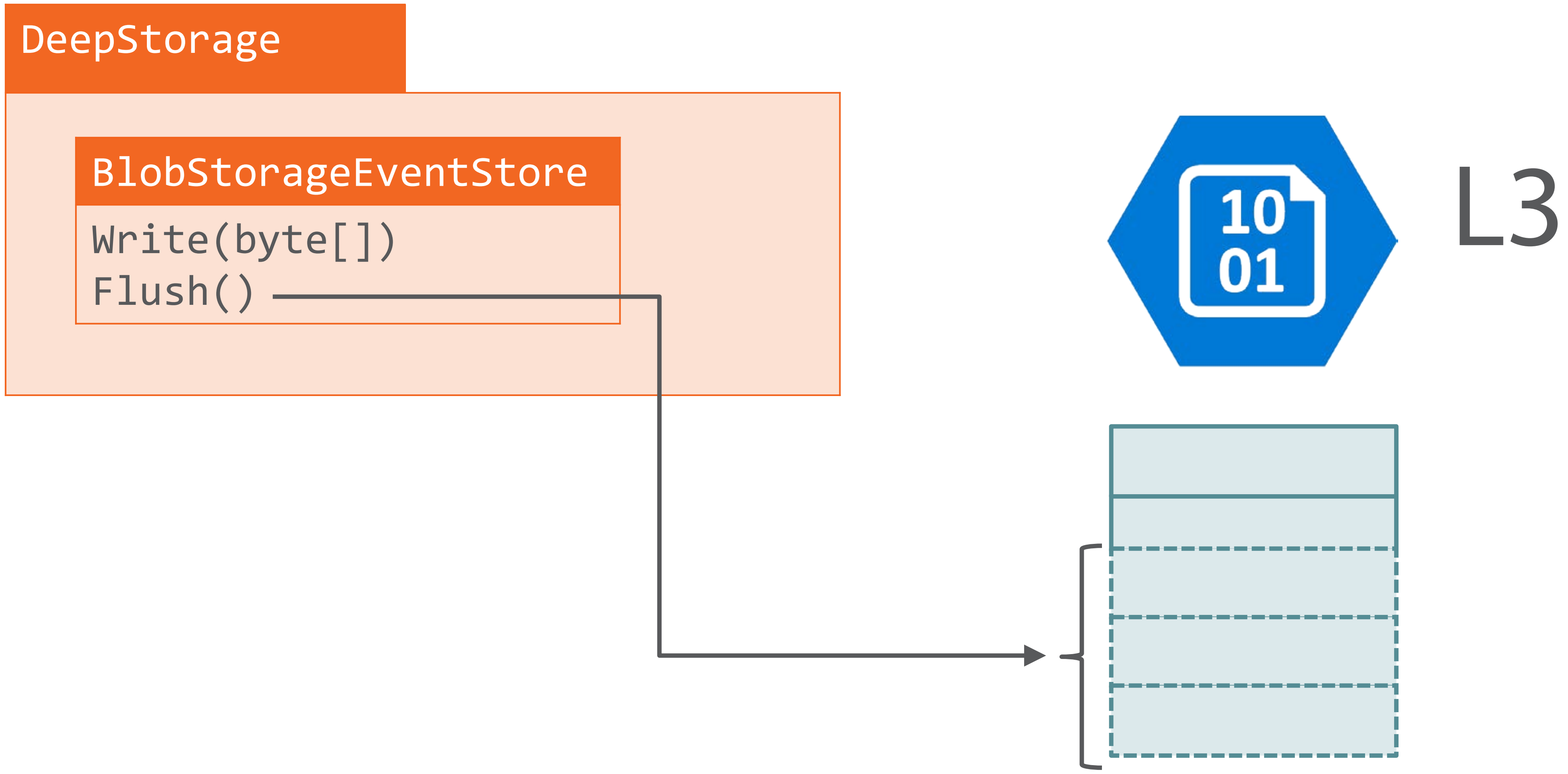

DeepStorage
BlobStorageEventStore
Write(byte[])
10
01
L3

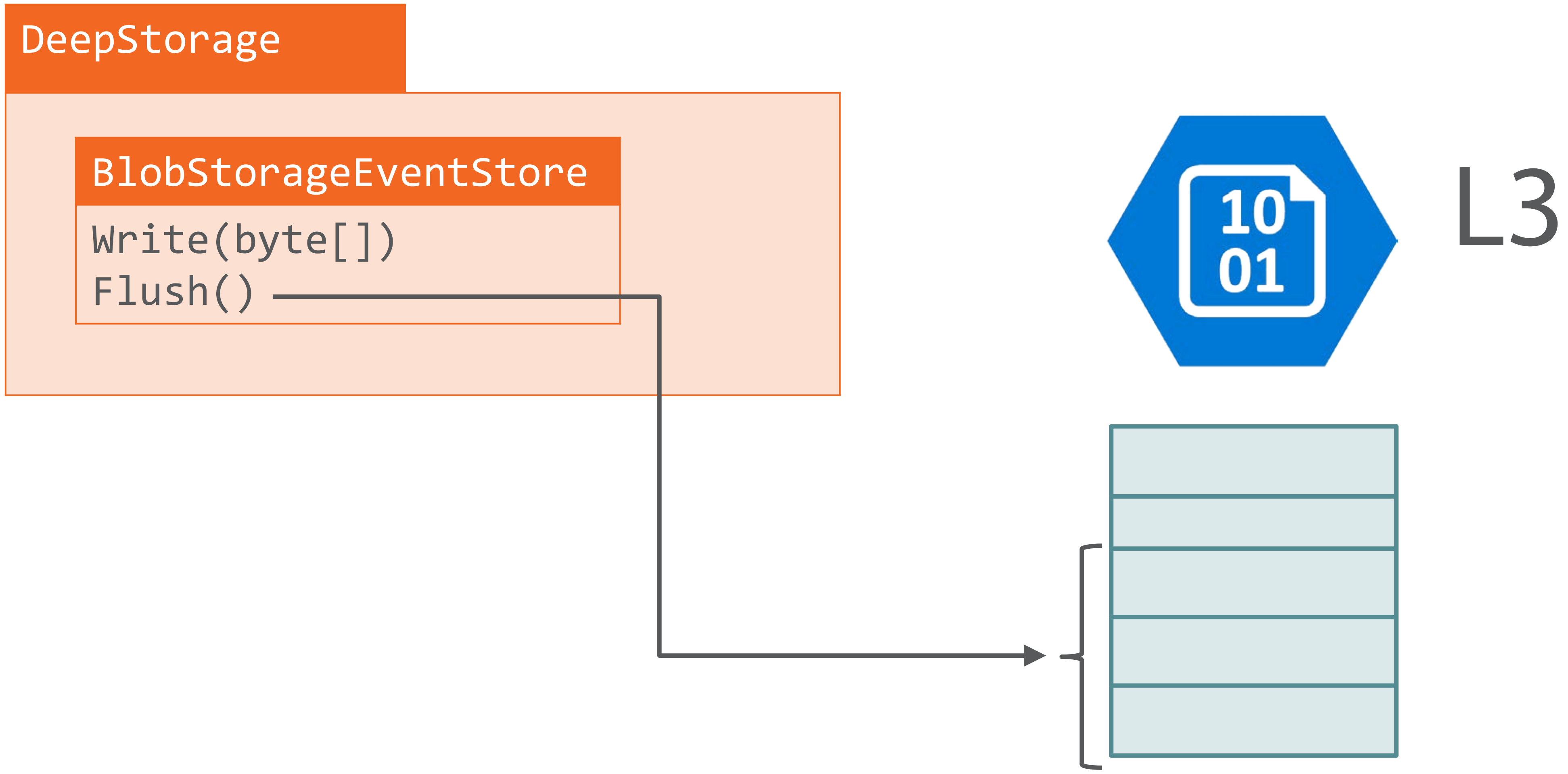

DeepStorage
BlobStorageEventStore
Write(byte[])
10
01
L3


DeepStorage
BlobStorageEventStore
Write(byte[])
10
01
L3


DeepStorage
BlobStorageEventStore
Write(byte[])
Flush()


10
01
L3


DeepStorage
BlobStorageEventStore
Write(byte[])
Flush()
10
01
L3


DeepStorage
BlobStorageEventStore
Write(byte[])
Flush()
10
01
L3

Demo: BlobStorageEventStore

Inherits EventStoreBase

Appends <=4MB blocks

Commits with lease

```
_lock.Wait();
using (var lease = new RenewingBlobLease(_blob))
{
    var offset = 0;
    while (offset < data.Length)
    {
```

BlobStorageEventStore

Write loops data with renewing blob lease

```
var remaining = data.Length - offset;
var length = remaining < MaxBlockSize ? remaining :
                                         MaxBlockSize;
using (var stream = new MemoryStream(data, offset, length))
{
```

BlobStorageEventStore

Extract blocks of 4MB or less

```
var blockId =
        Convert.ToBase64String(Guid.NewGuid().ToByteArray());
_blob.PutBlock(blockId, stream, null,
        AccessCondition.GenerateLeaseCondition(lease.Id));
offset += (int)stream.Length
_blockIds.Add(blockId)
```

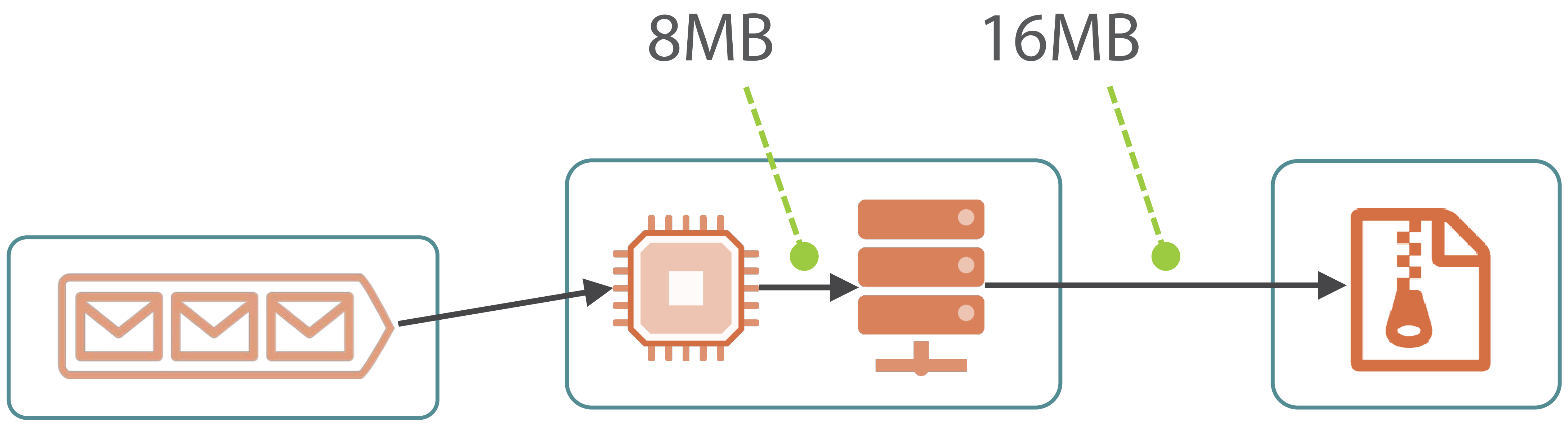
BlobStorageEventStore

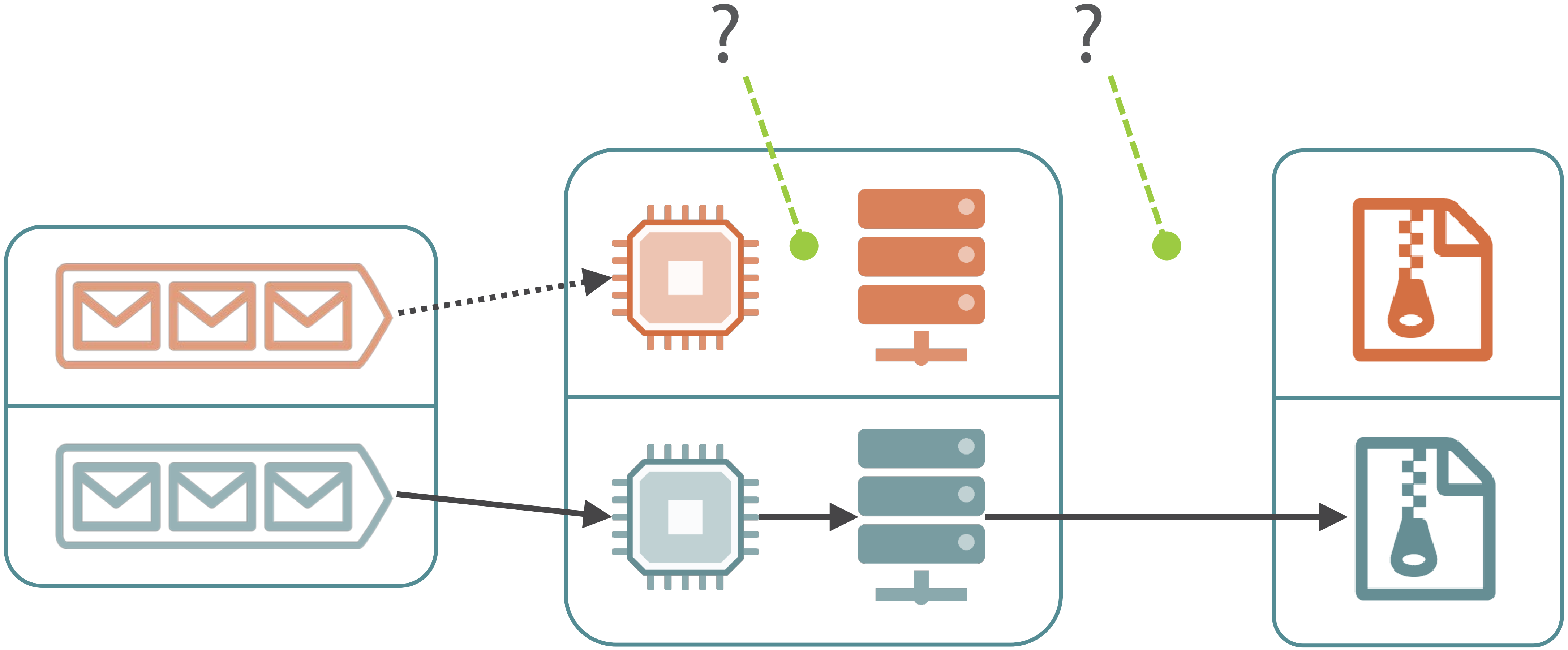
Append block with unique ID

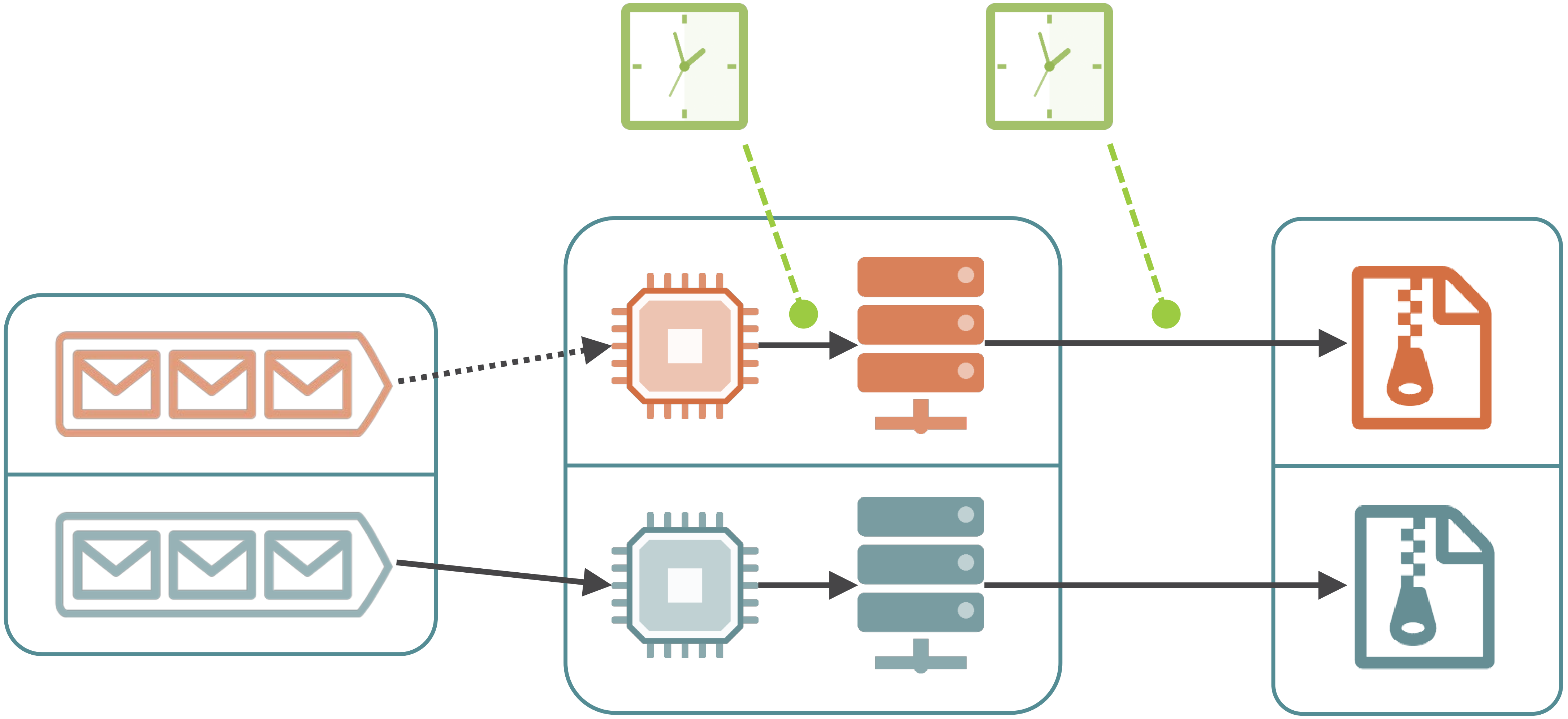
```
var blockList =
      _blob.DownloadBlockList(BlockListingFilter.Committed,
      AccessCondition.GenerateLeaseCondition(lease.Id));
var blockIds = blockList.Select(x => x.Name).ToList();
blockIds.AddRange(_blockIds);
_blob.PutBlockList(blockIds,
      AccessCondition.GenerateLeaseCondition(lease.Id));
```

BlobStorageEventStore

Flush commits new blocks


8MB
16MB





Demo: WorkerRole

Start EventReceiver

Run WorkerRole

View blobs with CloudBerry Explorer

```
Container.Instance.RegisterType<IEventStore,
                                MemoryEventStore>("1");
Container.Instance.RegisterType<IEventStore,
                                DiskEventStore>("2");
Container.Instance.RegisterType<IEventStore,
                                BlobStorageEventStore>("3");
```

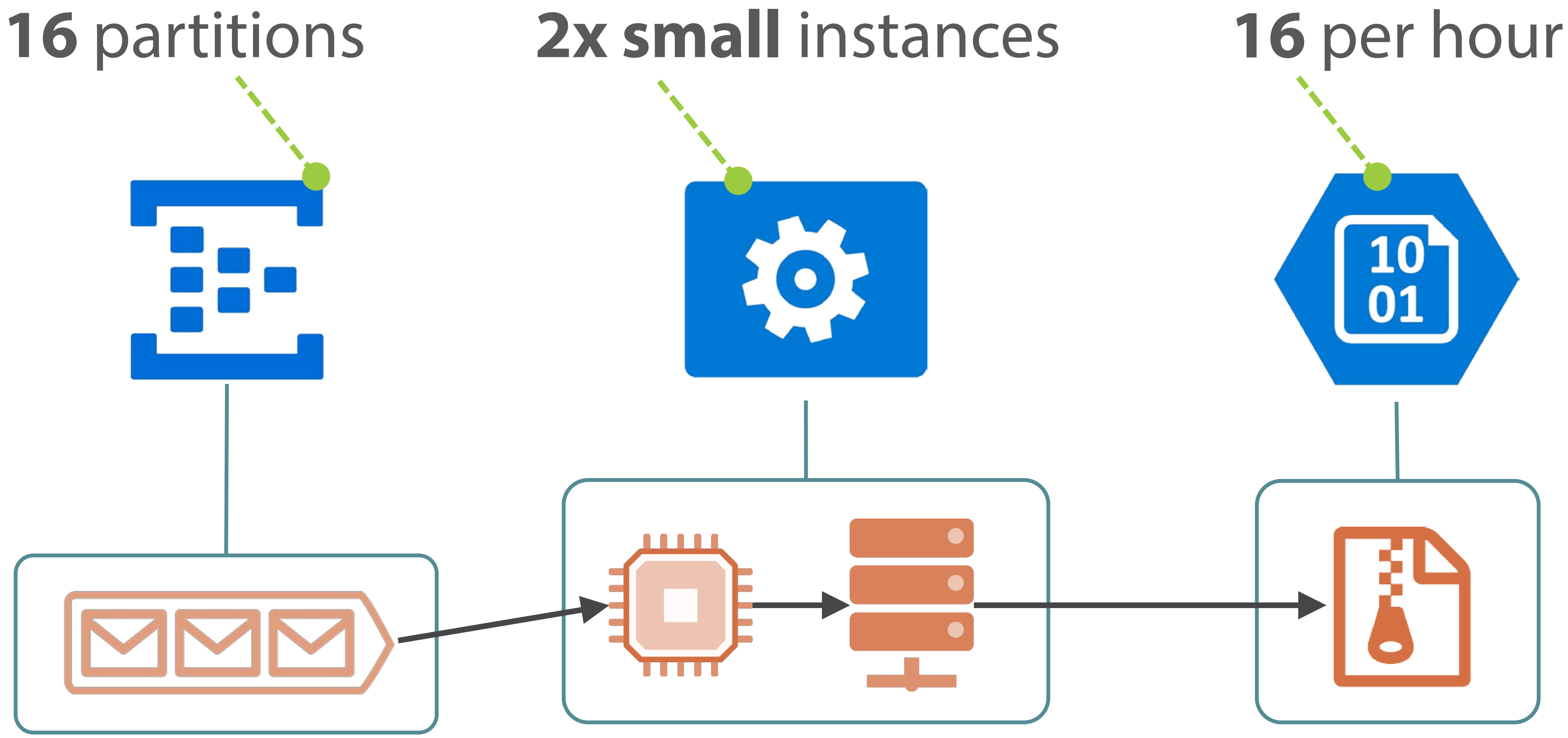
WorkerRole

Register event stores on Start

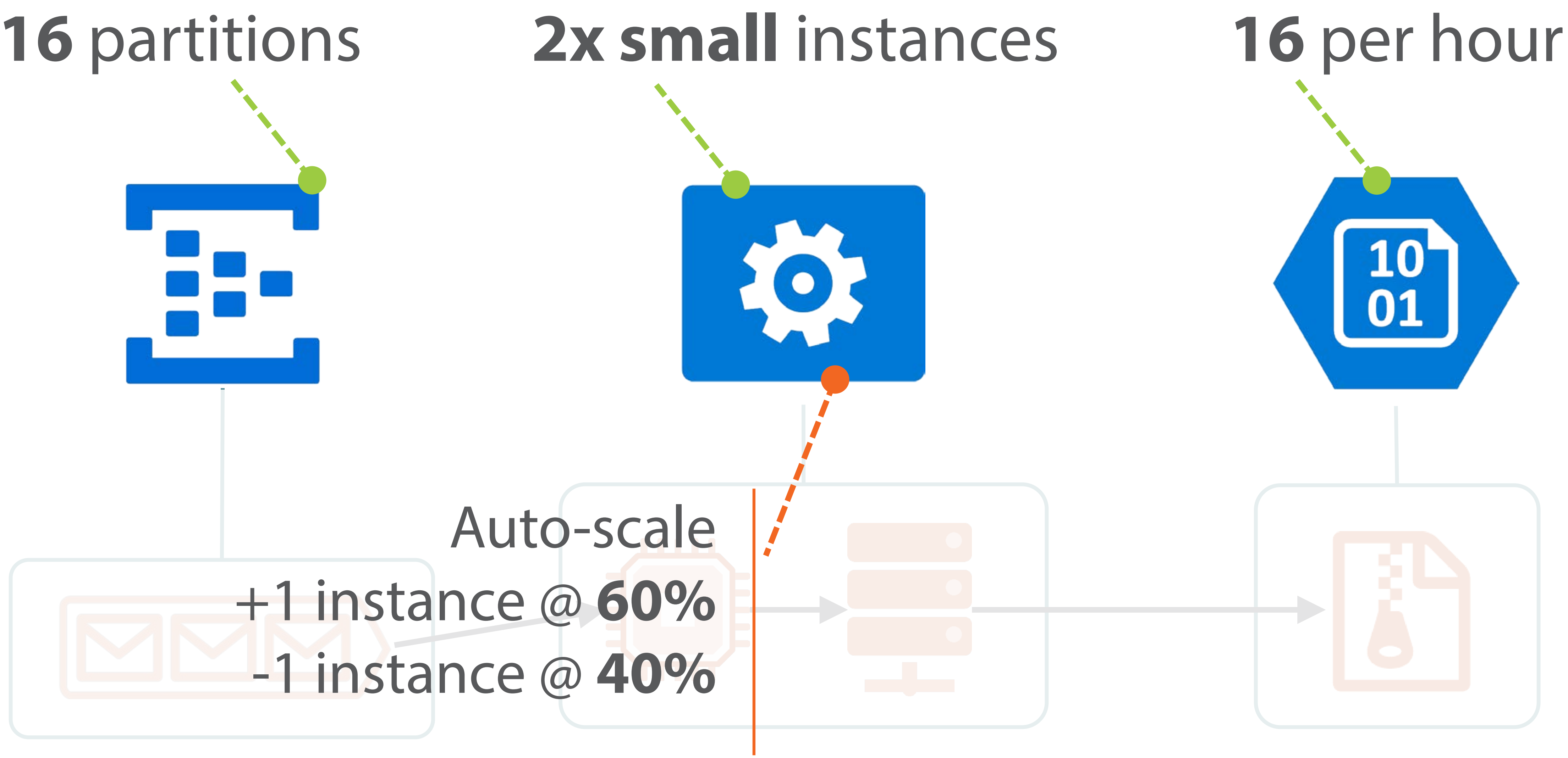
```
public override void Run()
{
    _receiver.RegisterProcessorAsync().Wait();
    CompletedEvent.WaitOne();
}
```

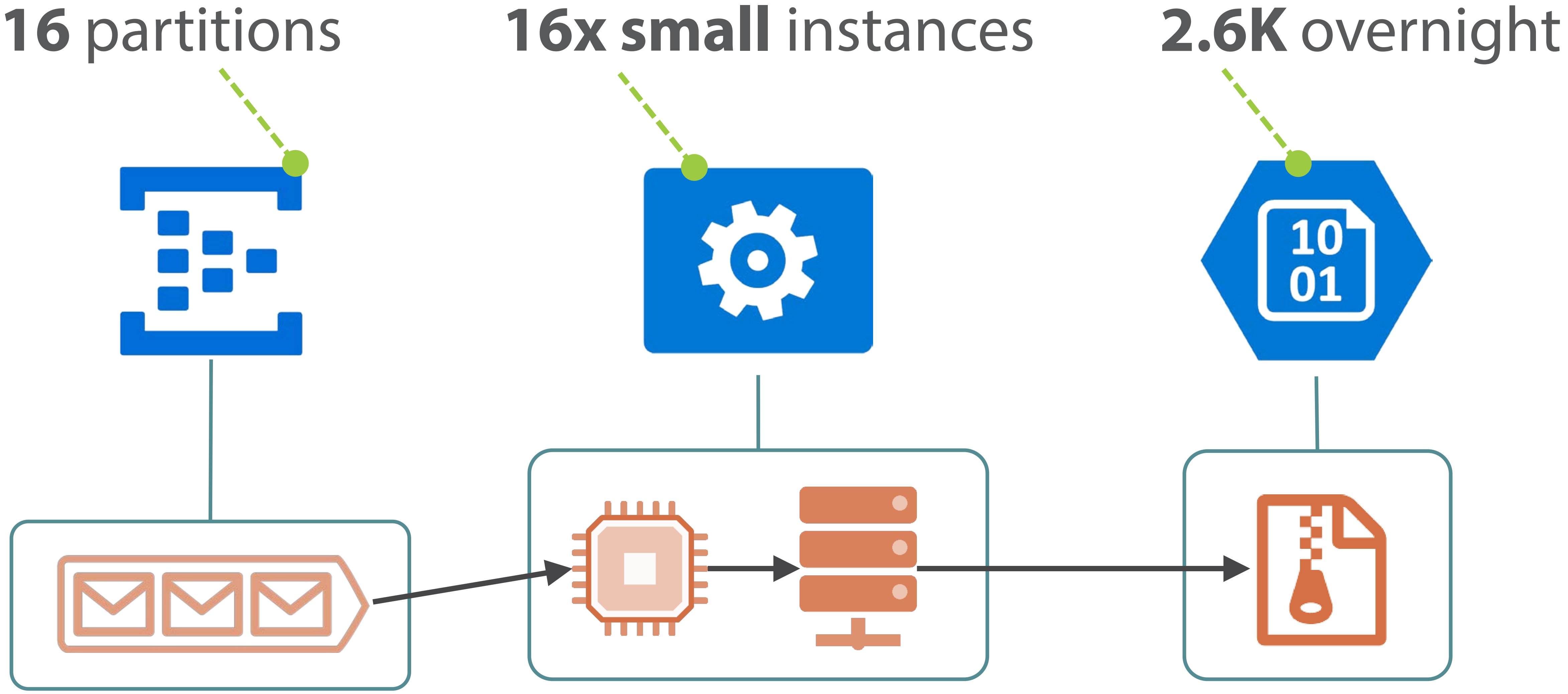
WorkerRole

Register processor & start receiving events on Run



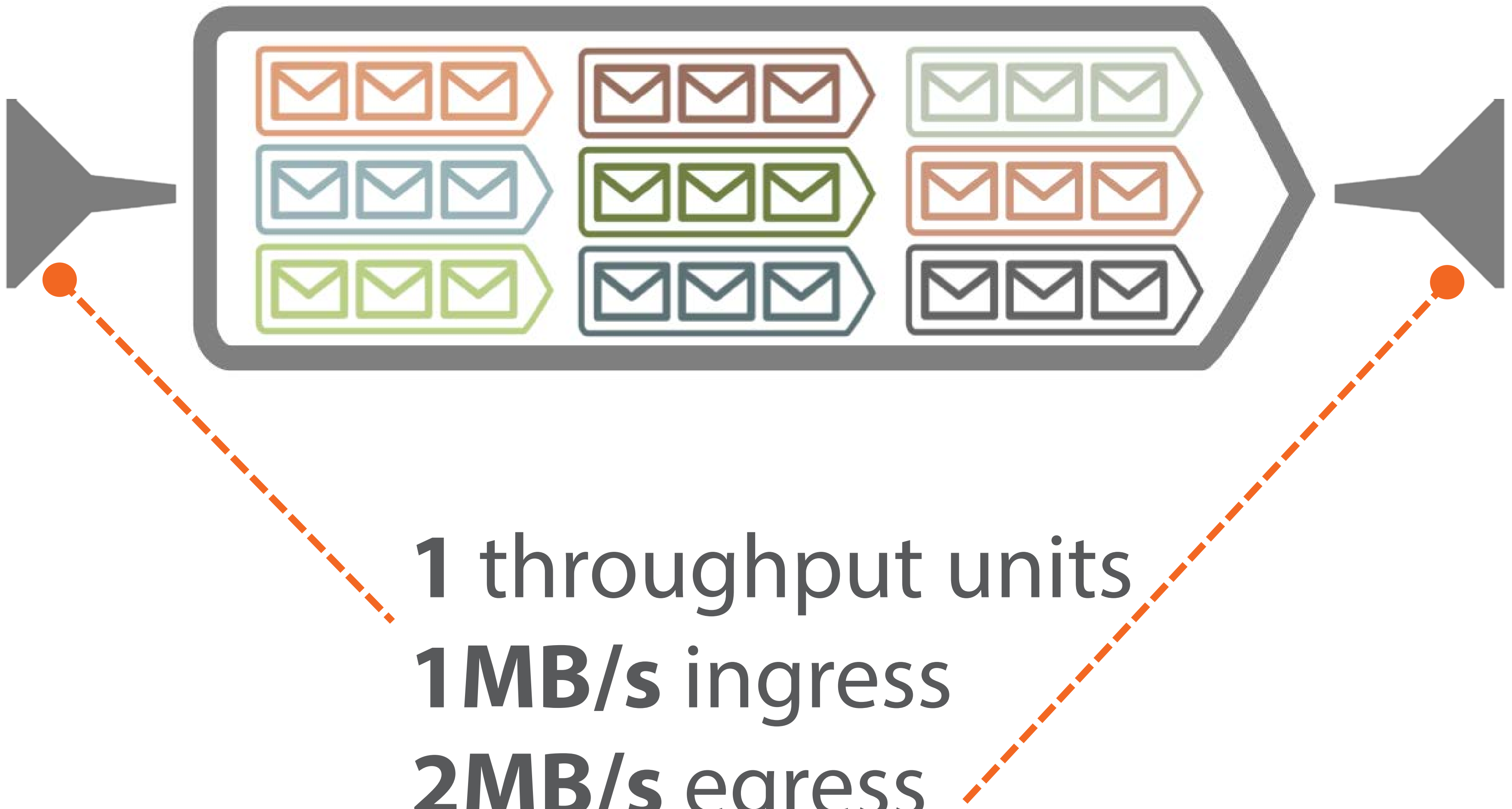
16 partitions
2x small instances
16 per hour
10
01


16 partitions
2x small instances
16 per hour
Auto-scale
+1 instance @ **60%**
-1 instance @ **40%**


16 partitions
16x small instances
2.6K overnight
10
01

1 throughput units

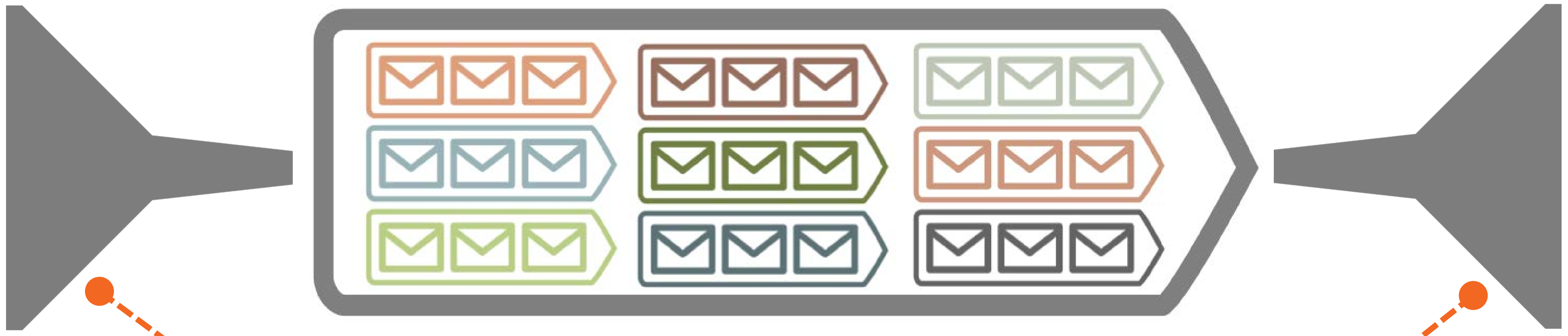
1MB/s ingress



1 throughput units

1MB/s ingress

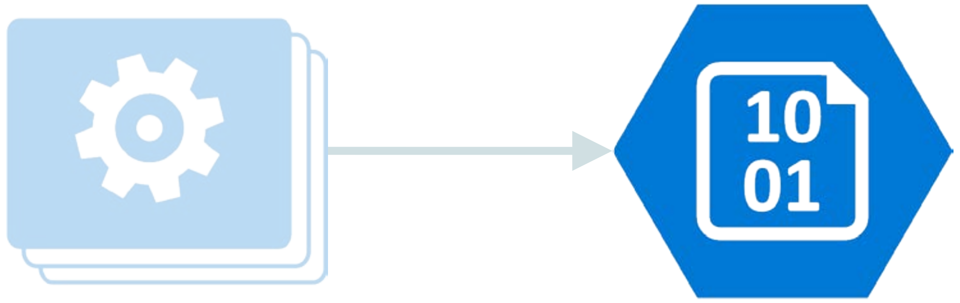
2MB/s egress



5 throughput units

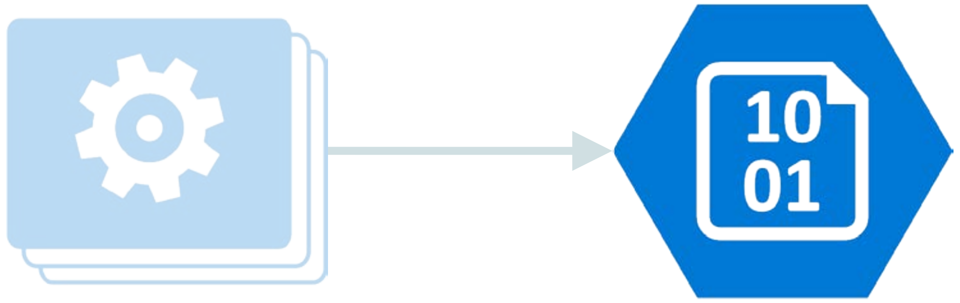
5MB/s ingress

10MB/s egress



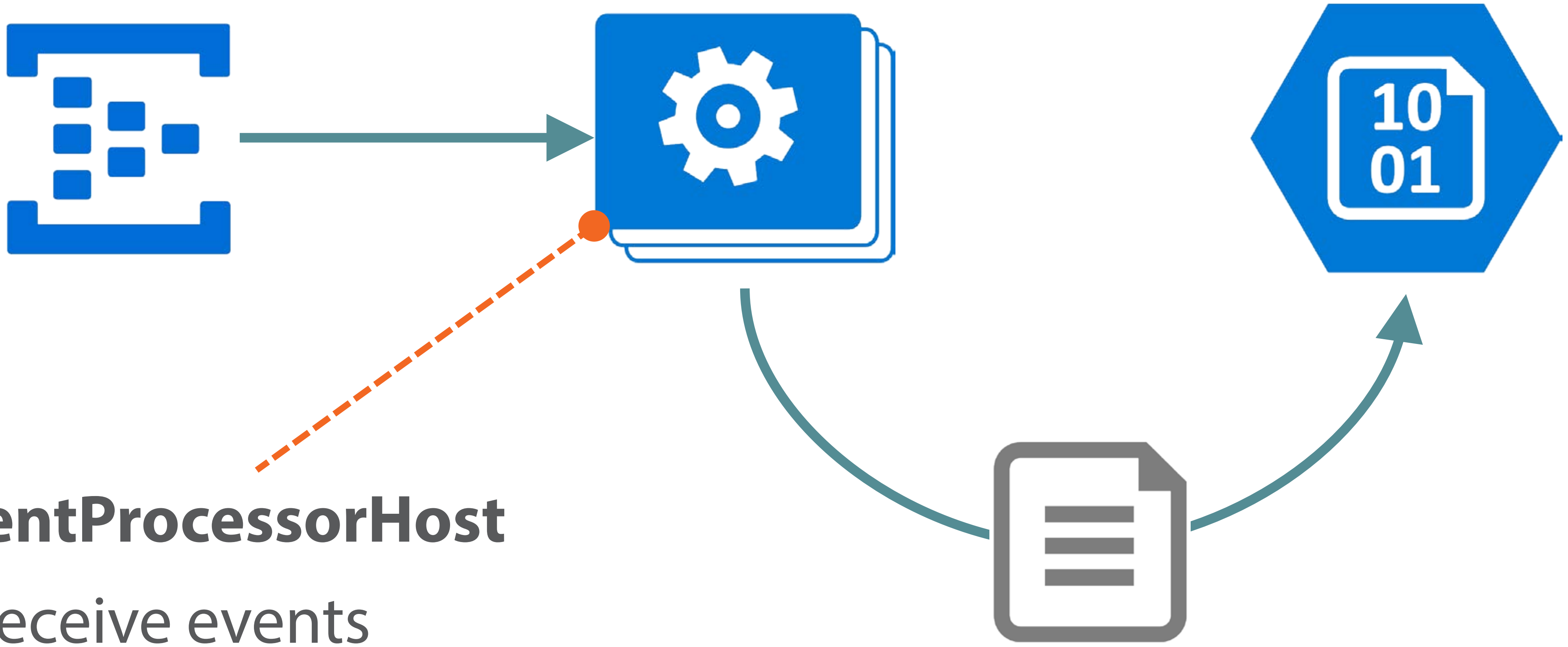
- 500TB storage
- 20K requests /sec
- 20Gb/sec upload
- 30Gb/sec download

Per Storage Account



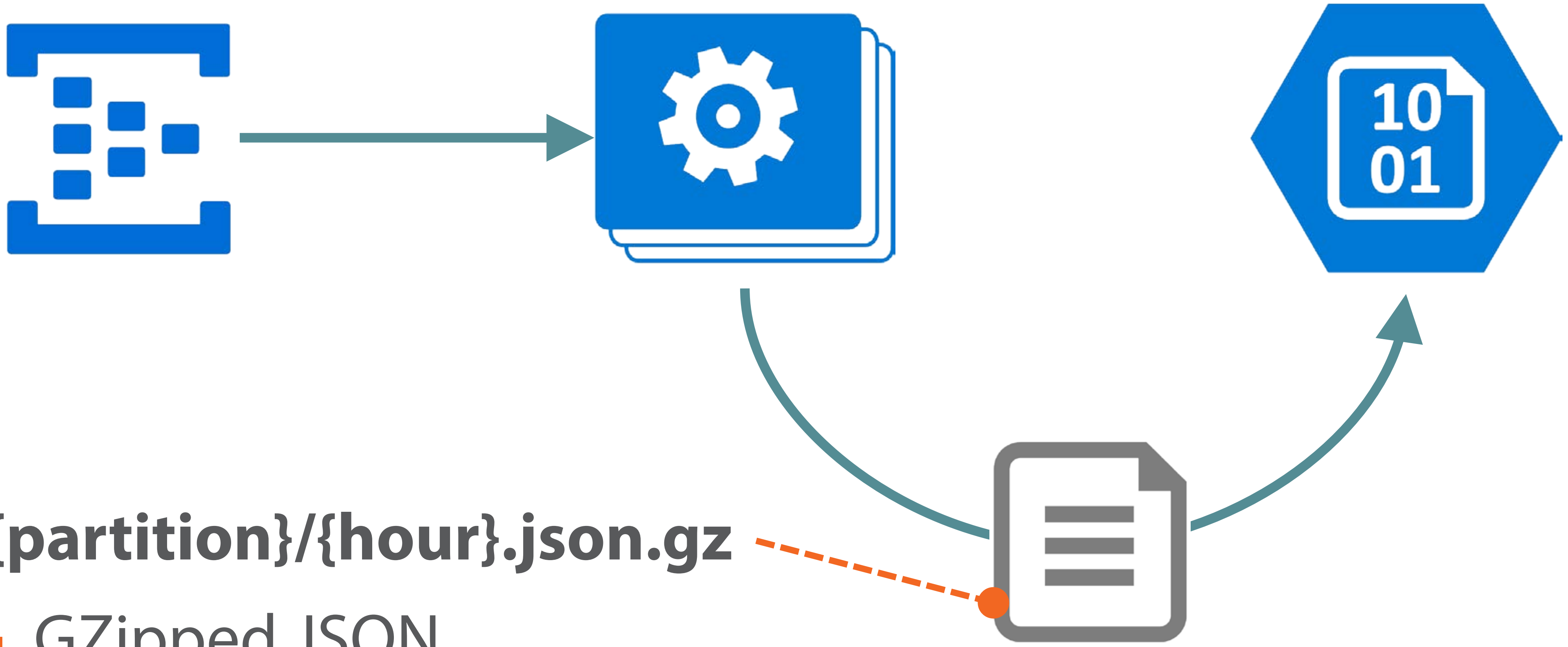
- 500TB storage
- 20K requests /sec
- 20Gb/sec upload
- 30Gb/sec download

<**$25** per TB per month



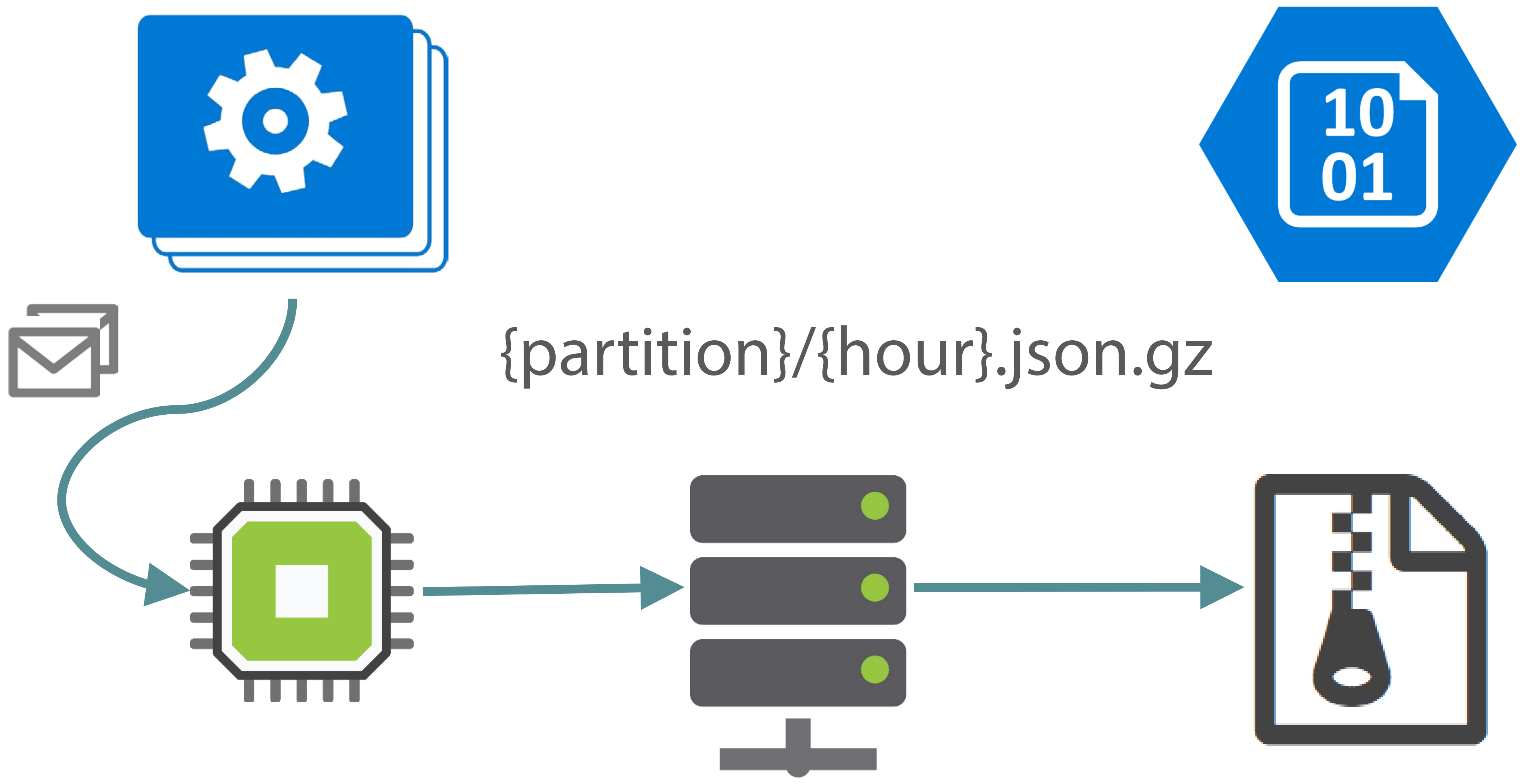
EventProcessorHost

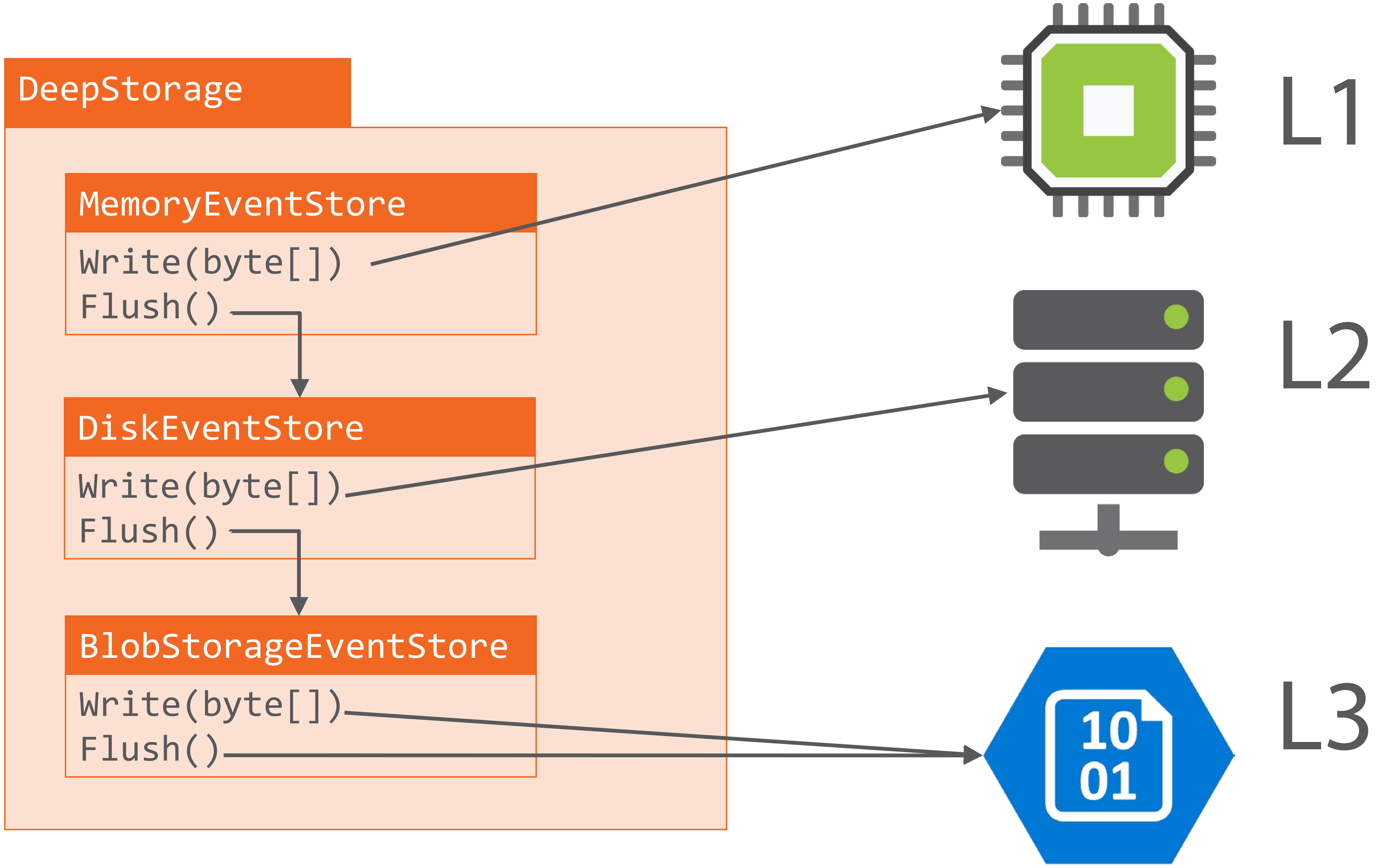
- Receive events
- Lock partitions
- Checkpoint progress

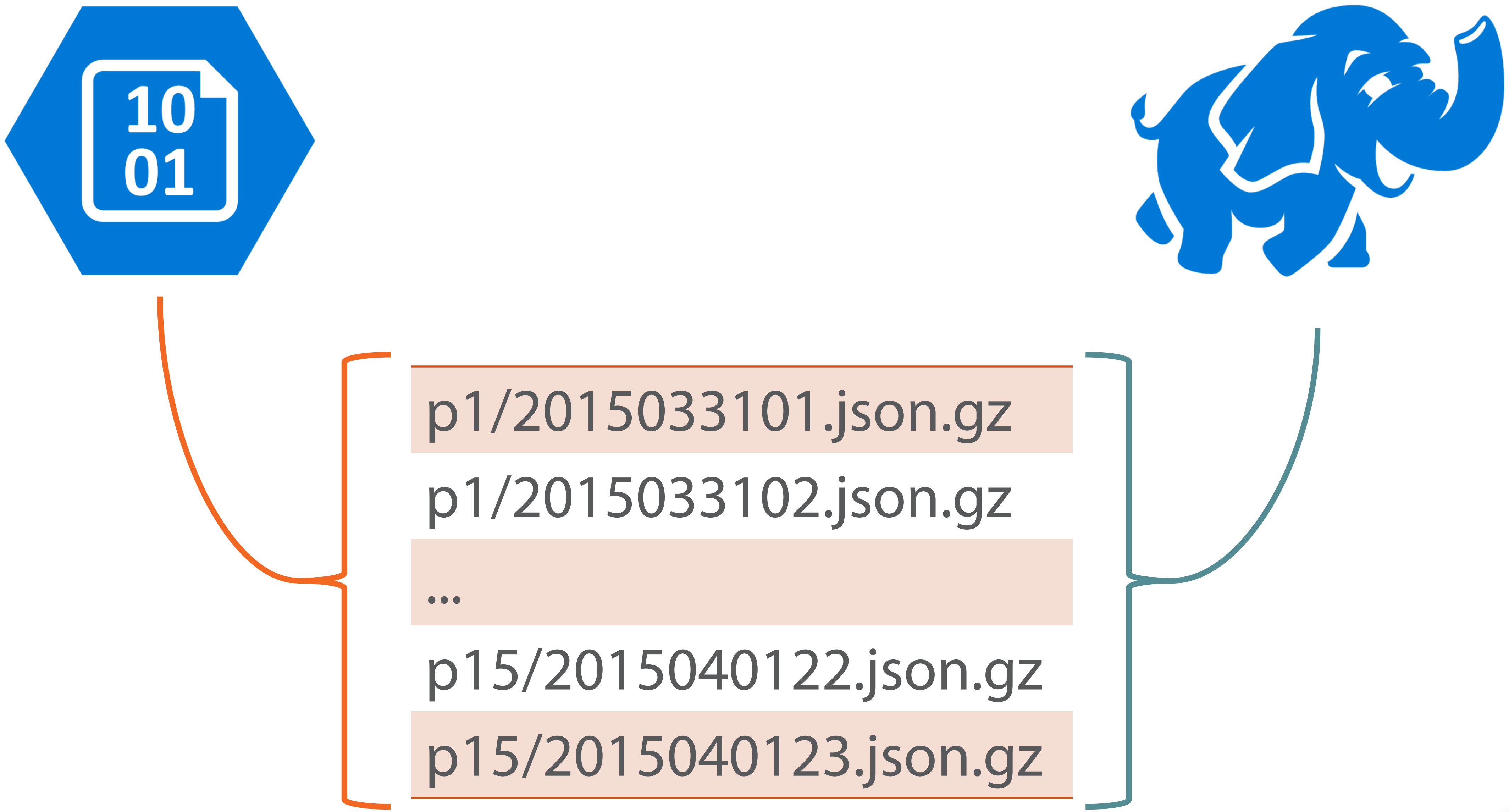


{partition}/{hour}.json.gz

- GZipped JSON
- One partition, one hour
- 384 files per day


10
01
{partition}/{hour}.json.gz


DeepStorage
MemoryEventStore
Write(byte[])
Flush()
DiskEventStore
Write(byte[])
Flush()
BlobStorageEventStore
Write(byte[])
Flush()
L1
L2
10
01
L3



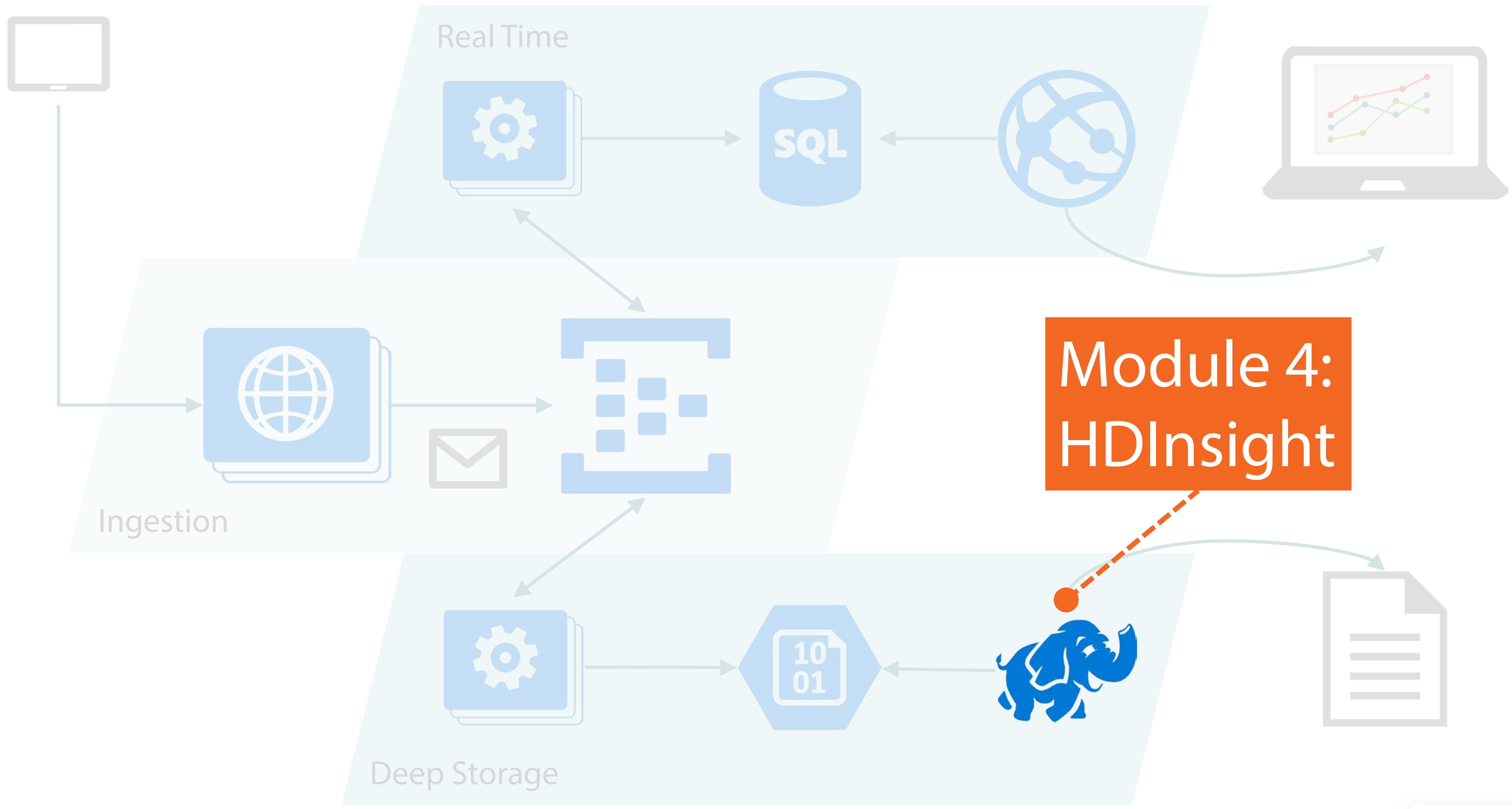
p1/2015033101.json.gz

p1/2015033102.json.gz

...

p15/2015040122.json.gz

p15/2015040123.json.gz


Real Time
SQL
Ingestion
Deep Storage
10
01
Module 4: HDInsight