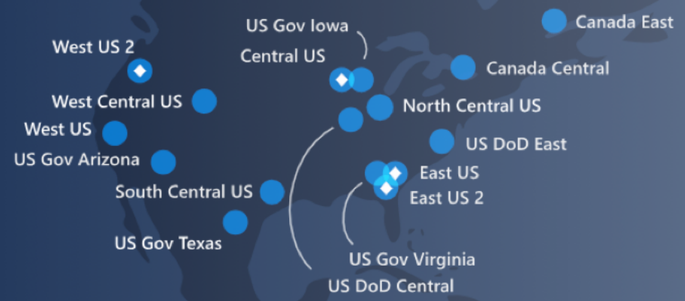
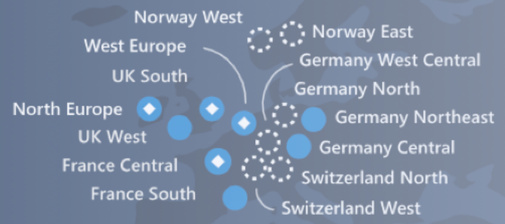
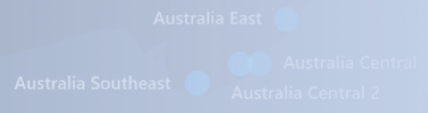




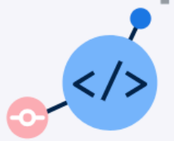
Microsoft Azure ❤️ Open Source Tech



2019

Global Azure BOOTCAMP

Microsoft + Open Source ?

People →  How, when, and where developers build and learn on GitHub Projects →  Popular projects, topics, and programming languages Platform →  The GitHub application ecosystem, Marketplace, and security programs

GitHub Octoverse 2018

Top open source projects

VS Code, React, and Tensorflow once again top our list of open source projects by contributor count. New to the list are projects that manage containerized applications, share Azure documentation, and consolidate TypeScript type definitions: Kubernetes, Azure Docs, and DefinitelyTyped. *

		Contributors
1	Microsoft/vscode	19K
2	facebook/react-native	10K
3	tensorflow/tensorflow	9.3K
4	angular/angular-cli	8.8K
5	MicrosoftDocs/azure-docs	7.8K
6	angular/angular	7.6K
7	ansible/ansible	7.5K
8	kubernetes/kubernetes	6.5K
9	npm/npm	6.1K
10	DefinitelyTyped/DefinitelyTyped	6.0K

GitHub Octoverse 2018

Fastest growing open source projects

Overall, we're seeing trends in growth of projects related to machine learning, gaming, 3D printing, home automation, scientific programming, data analysis, and full stack JavaScript development. *

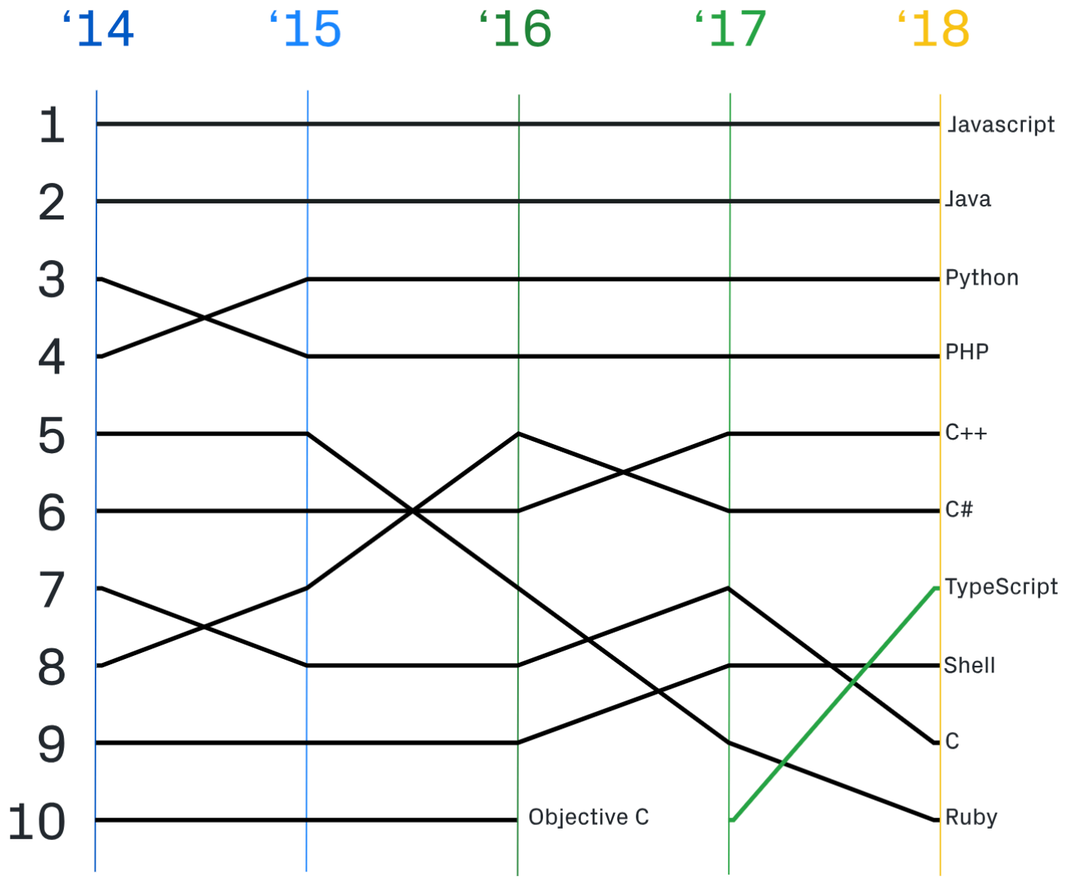
		Change
1	MicrosoftDocs/azure-docs	4.7x
2	pytorch/pytorch	2.8x
3	godotengine/godot	2.2x
4	nuxt/nuxt.js	2.1x
5	ethereum/go-ethereum	2.0x
6	wix/react-native-navigation	1.9x
7	spyder-ide/spyder	1.8x
8	tensorflow/models	1.8x
9	home-assistant/home-assistant	1.6x
10	MarlinFirmware/Marlin	1.6x

GitHub Octoverse 2018

Top languages over time

You're coding on GitHub in hundreds of programming languages, but JavaScript still has the most contributors in public and private repositories, organizations of all sizes, and every region of the world.

This year, TypeScript shot up to #7 among top languages used on the platform overall, after making its way in the top 10 for the first time last year. TypeScript is now in the top 10 most used languages across all regions GitHub contributors come from—and across private, public, and open source repositories. *

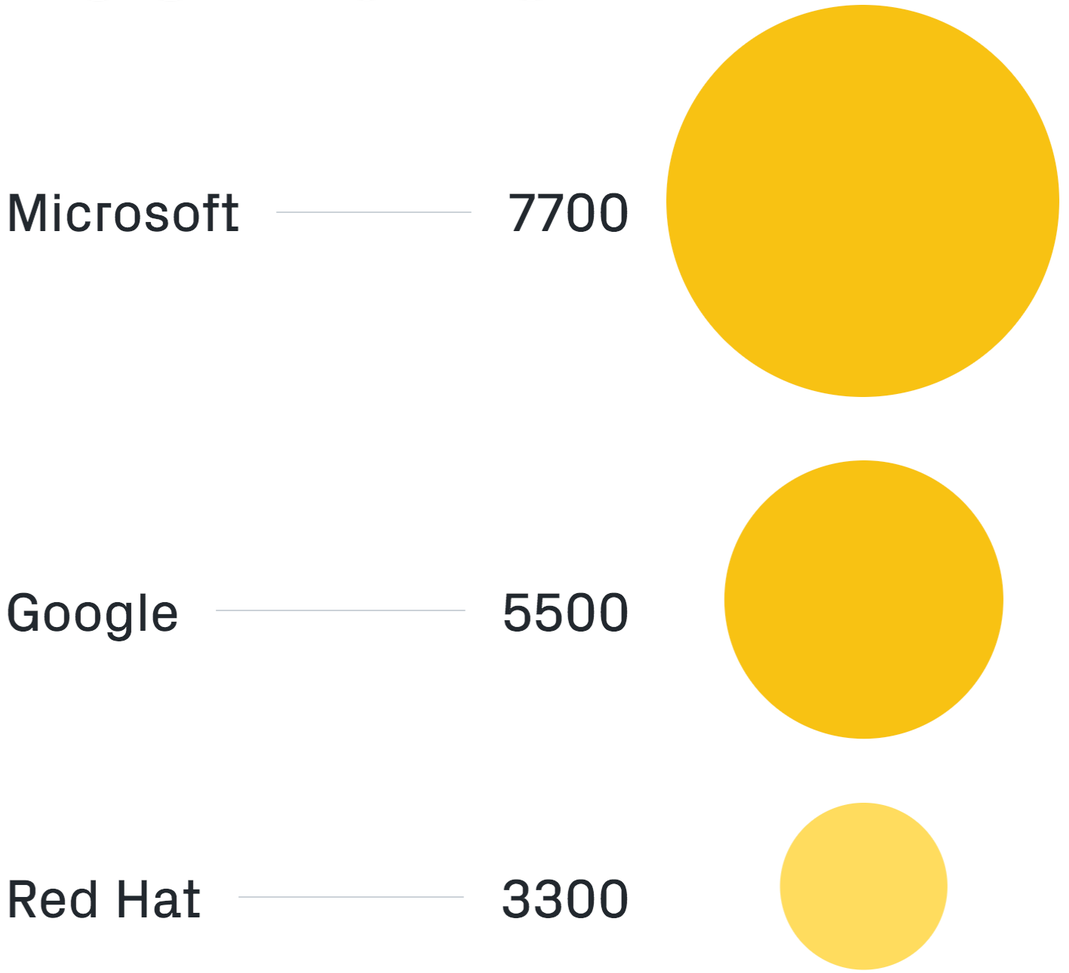


GitHub Octoverse 2018

> ORGANIZATIONS COMMITTING TO OPEN SOURCE

Open source contributions made by employees of different organizations

Open source development is driven by millions of paid and volunteer developers—and many of the organizations that employ them. Microsoft, Google, Red Hat, Intel, and a number of universities top the list of organizations whose employees contribute most to open source. *



HDInsight Services

HDInsight Services

Batch	SQL	NoSQL	Stream	Machine Learning
MapReduce	Apache Hive LLAP	Apache HBase	Apache Kafka	MLib
Apache Pig	Apache Spark SQL		Apache Storm	
Apache Spark	Apache Phoenix		Apache Spark	
Apache Hive				

$ whoami

- Hans-Peter Grahsl
- living & working in Graz 🇦🇹
- technical trainer at NETCONOMY
- independent engineer & consultant
- associate lecturer
- 🗣 irregular conference speaker 📣

Application Needs ?

- "It depends" 🤓
- plenty of pieces & components
- irrespective of concrete use cases
- two architectural pieces...

operational data store

Cosmos DB

messaging platform

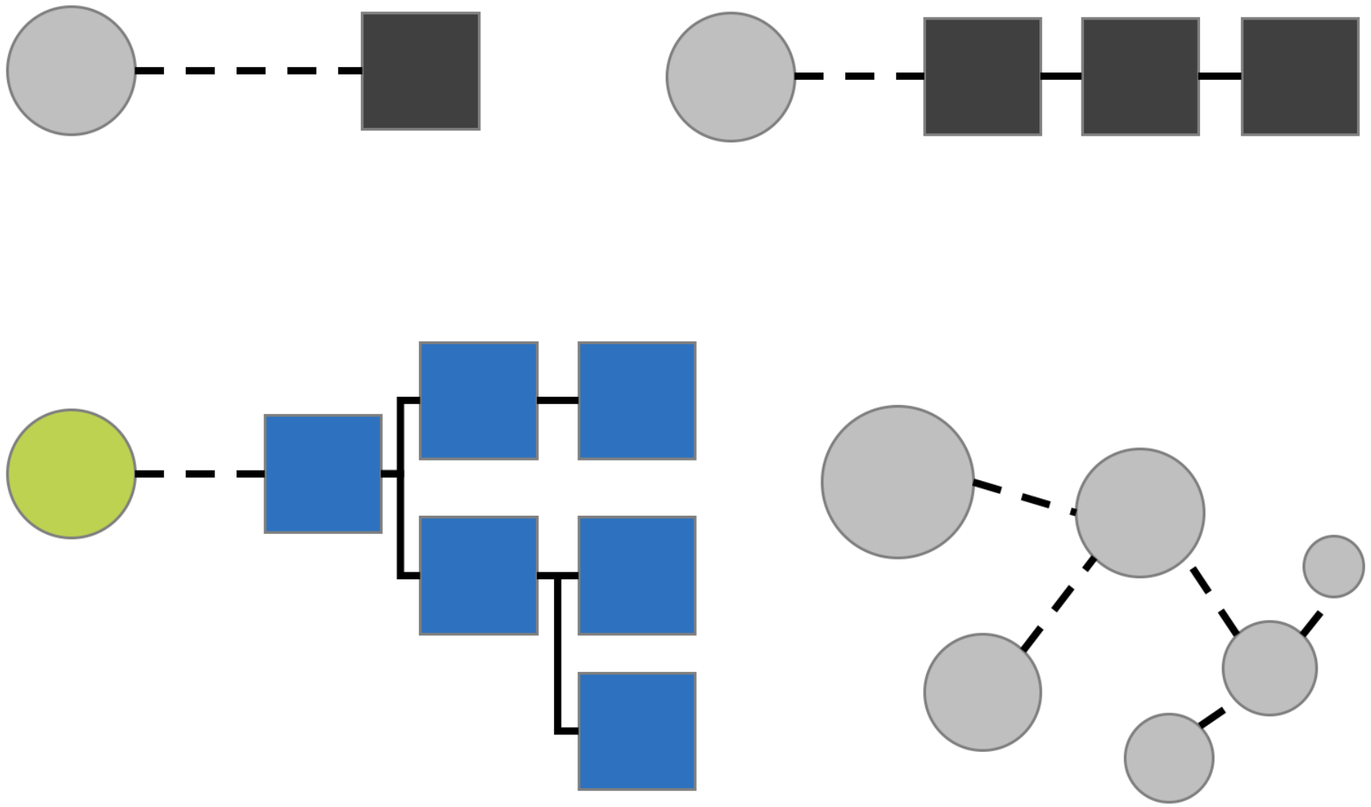
Event Hubs

Cosmos DB

- global distribution
- high availability & elastic scaling
- multi-model & **native NoSQL APIs**
- consistency choices
- leading security, compliance & SLAs

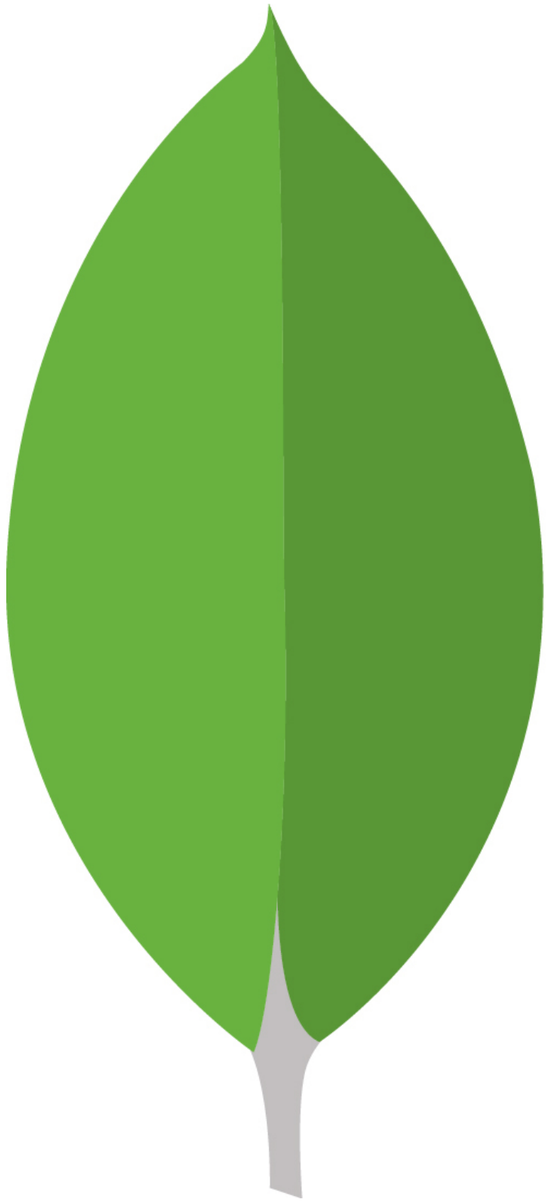
Multi-Model

- keys + values
- **documents**
- column families
- graphs



Native API Support

- SQL
- Table Storage
- **MongoDB**
- Cassandra
- Gremlin



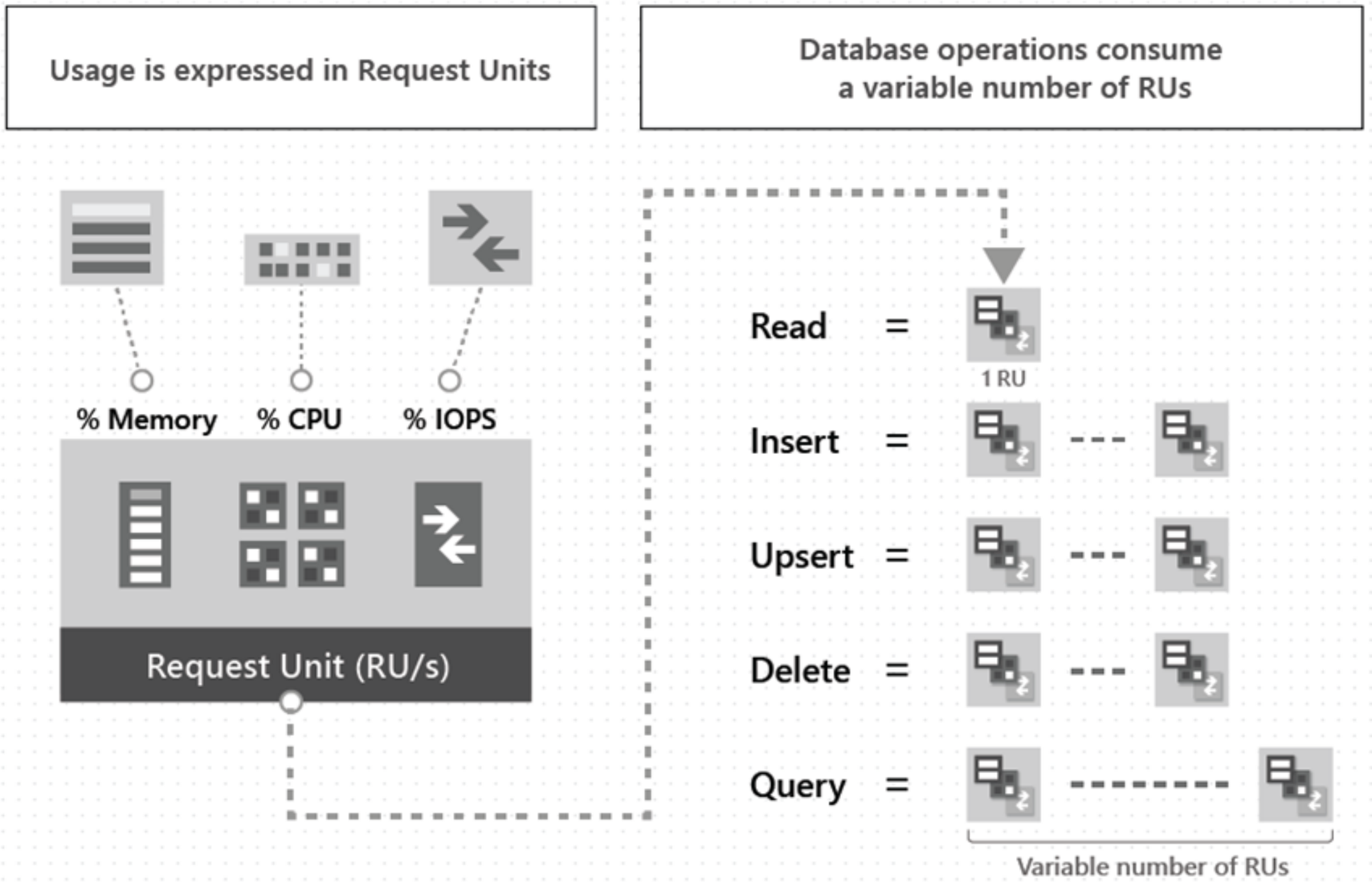
Throughput Provisioning

- **request units (RUs)**
- abstract performance metric
- combination of **CPU + Memory + IOPS**
- backed by SLAs


exciting
...in theory

Azure Docs on RUs

"The **cost of all database operations is normalized** *by Azure Cosmos DB* **and** *is* **expressed in terms of Request Units** *(RUs). The* **cost to read a 1-KB item is 1 Request Unit** *(1 RU) and minimum RUs required to consume 1 GB of storage is 40."*


Usage is expressed in Request Units
Database operations consume a variable number of RUs
% Memory
% CPU
% IOPS
Request Unit (RU/s)
Read = 1 RU
Insert =
Upsert =
Delete =
Query =
Variable number of RUs


daunting
...in practice

```
> db.sensordata.insert({
      "_id": 1234567,
      "serialno": "9C:CC:58:8E:42:56",
      "timestamp": 1549655161262,
      "locationid": "Location_7682",
      "gpscoords": {
          "latitude": 54.45508161071572,
          "longitude": 31.486299060940382
      },
      "data": {
          "sensor1b": false,
          "sensor2b": true,
          "sensor3i": 55,
          "sensor4d": -370.4468858634573,
          "sensor5s": "LOW",
          "sensor6a": [
              "ON",
              "ON",
              "ON"
          ]
      }
  })
Operation consumed 20.19 RUs
```

Measure via API

- **write operation** consumed ??? RUs
- given 1000 provisioned RUs at most
  ☛ 1000 ÷ 20.19 ≅ **49 docs/sec**
- **no parallel activity** considered

Measure via API

- **read operation** consumed ??? RUs
- given 1000 provisioned RUs at most
  ☛ 1000 ÷ 3.11 ≅ **321 reads/sec**
- **no parallel activity** considered

```
> db.sensordata.find({"_id" : 1234567})
Operation consumed 3.11 RUs
{
    "_id" : 1234567,
    "serialno" : "9C:CC:58:8E:42:56",
    "timestamp" : 1549655161262,
    "locationid" : "Location_7682",
    "gpscoords" : {
        "latitude" : 54.45508161071572,
        "longitude" : 31.486299060940382
    },
    "data" : {
        "sensor1b" : false,
        "sensor2b" : true,
        "sensor3i" : 55,
        "sensor4d" : -370.4468858634573,
        "sensor5s" : "LOW",
        "sensor6a" : [
            "ON",
            "ON",
            "ON"
        ]
    }
}
```

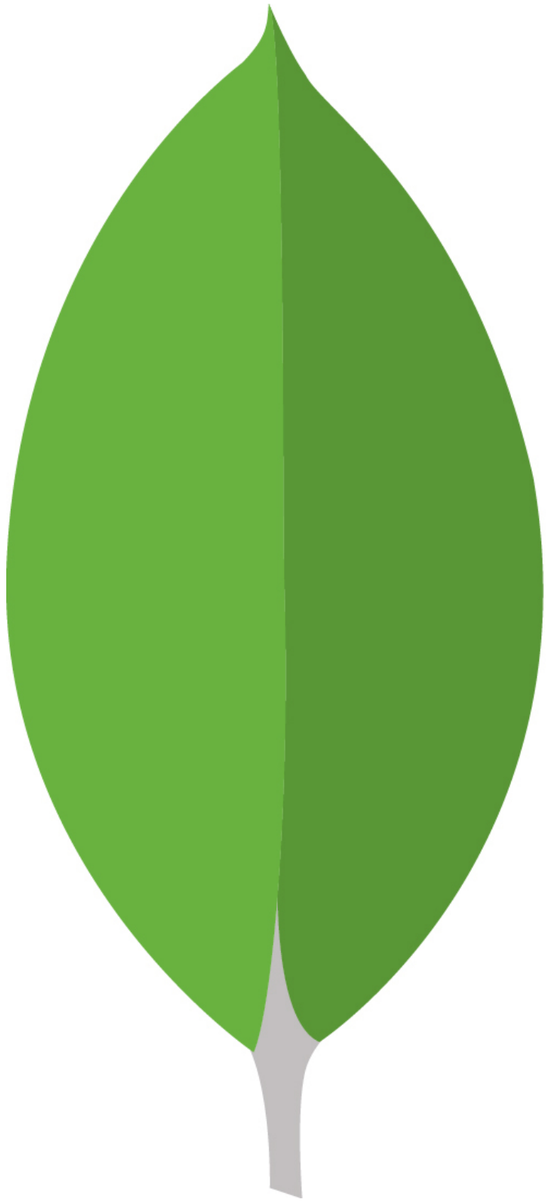
Influences on RUs

- **indexing**
- **property count**
- **consistency level**
- **query patterns**
- **scripts**

...and any other concurrent operation

Cosmos DB for MongoDB API

- **native** implementation
- **wire protocol** compatibility
- transparent **re-use** (code + tools)
- migration benefits: in **both ways**




Showtime!

Important Note

- feature support is limited of course
- carefully check what does (NOT) work
- wire protocol versions: 3.2 GA, 3.4 preview

3.6 ?? 4.0


It's going on right now
and I'm missing it!

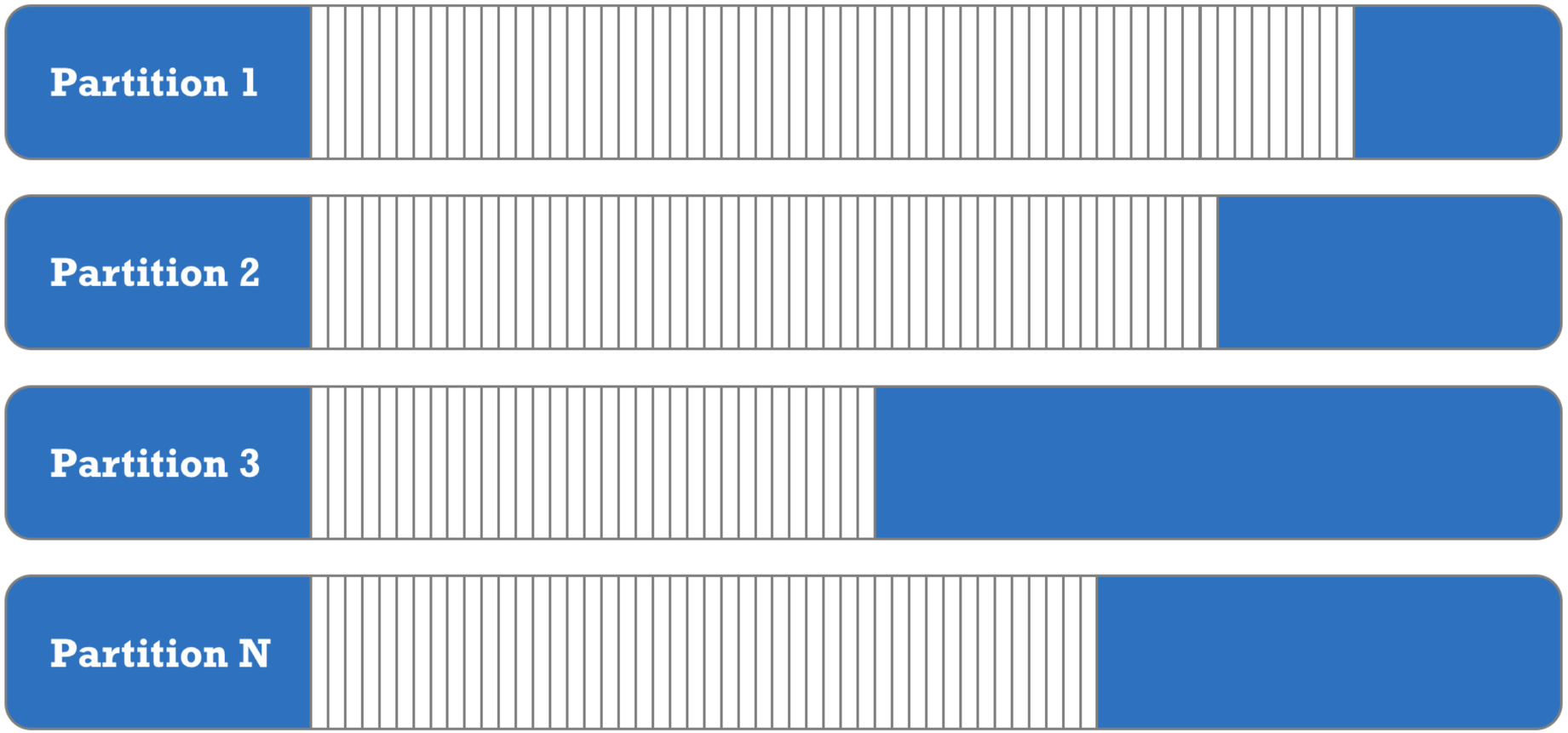
Currently Missing Out On…

- **powerful operators** from 3.2 + 3.4
  - *$graphLookup, $facet, $bucket(Auto)*
  - *$reduce, $zip, $switch, $replaceRoot, ...*
- **great features & major advances** from 3.6 + 4.0
  - views, change streams,
  - type conversions, schema validation,
  - multi-element array updates, array filters,
  - sessions & transactions, ...

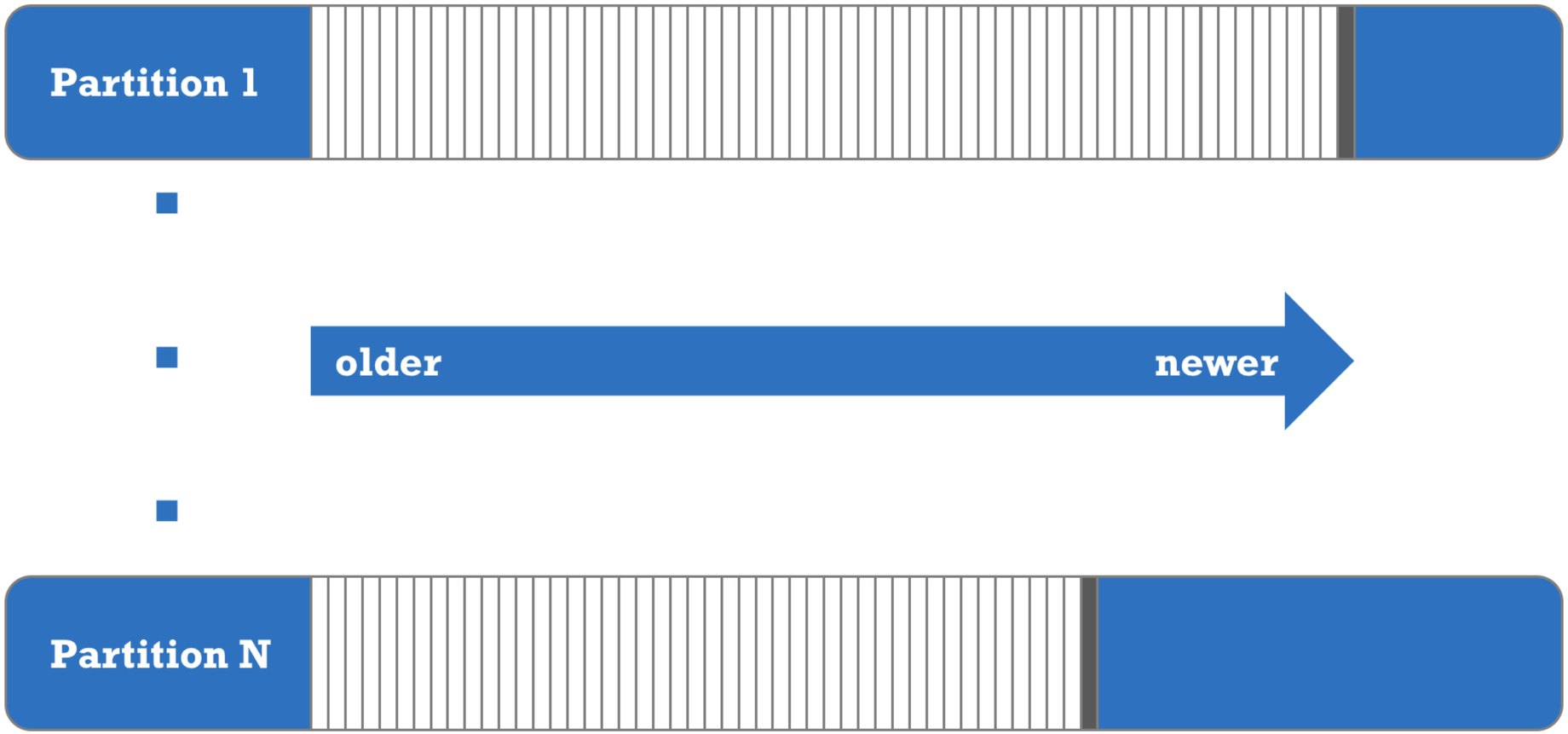
Event Hubs

- distributed **event ingestion** platform
- **decouple** producers ↔ consumers
- source for **stream processing**
- high **availability & scalability**
- **fully-managed**

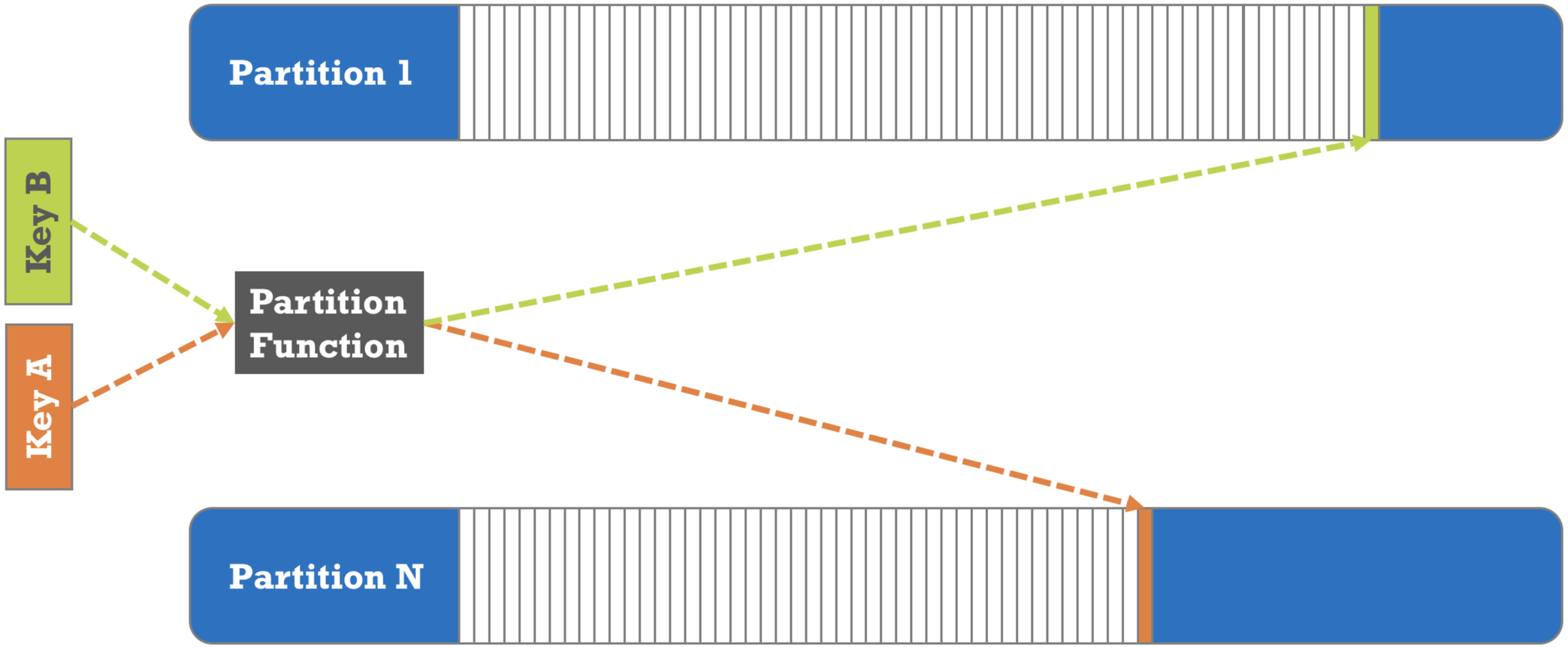
Concepts: Event Hub



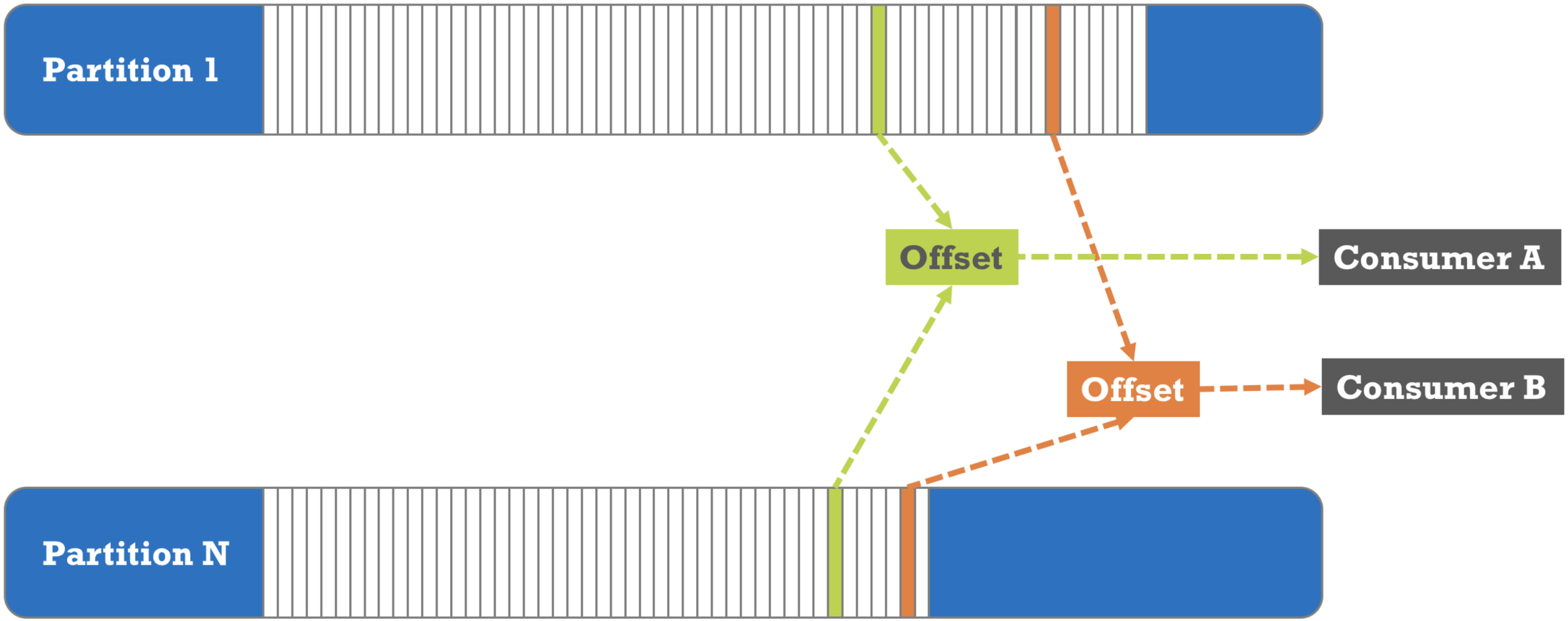
Concepts: Partition



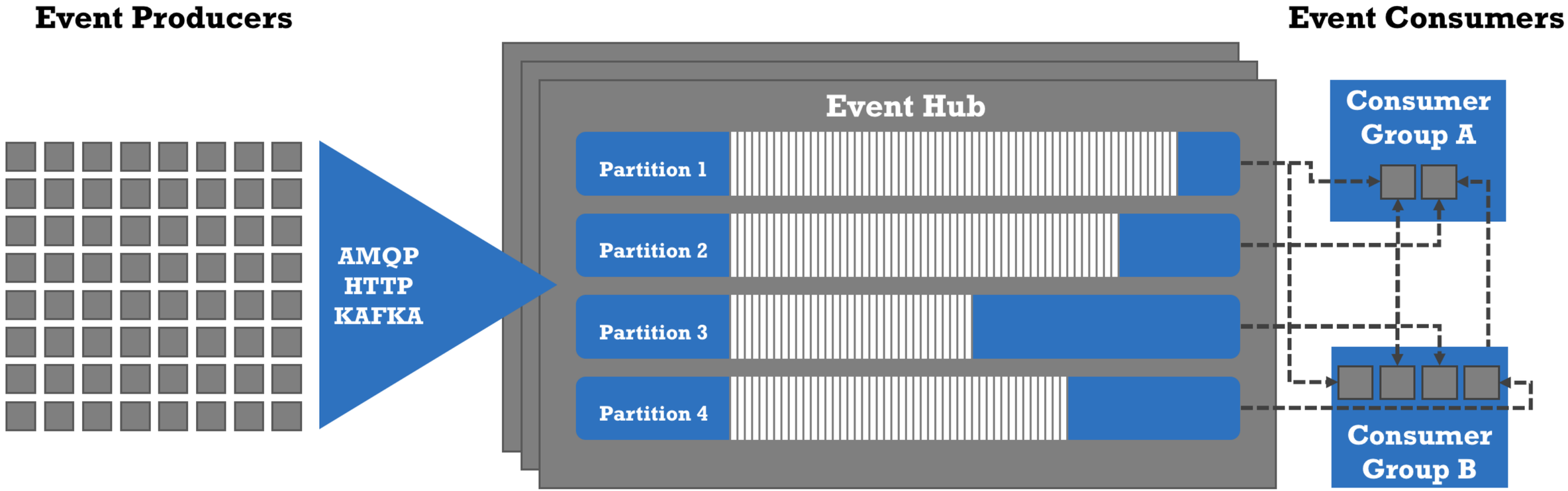
Concepts: Partition Keys



Concepts: Partition Offsets

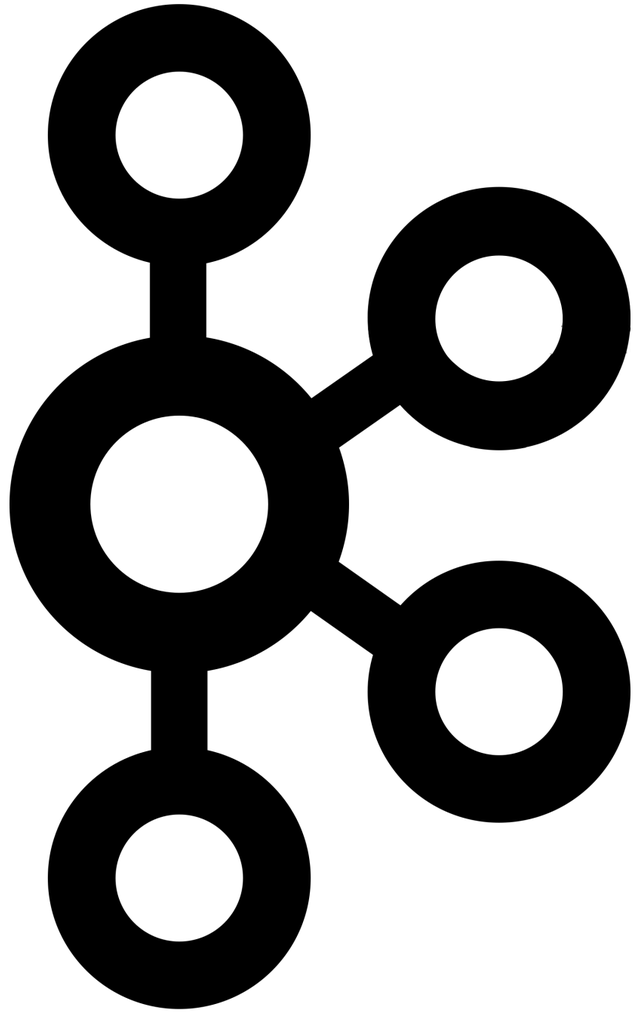


Concepts: Big Picture



Event Hubs for Apache Kafka

- **overlay** on top of Event Hubs
- **binary compatible** with Kafka 1.0+
- transparent **re-use** (code + tools)
- migration benefits: in **both ways**



The Promise…

"You **update the connection string** *in configurations* **to point to the Kafka endpoint exposed by your event hub** *instead of pointing to your Kafka cluster. Then, you can* **start streaming events from your applications that use the Kafka protocol into Event Hubs.**"


Showtime!

Unsupported Kafka Features

- idempotent producers & transactions
- message compression
- size-based retention or log compaction
- parition resize for existing topics
- HTTP Kafka API support
- Kafka Streams & KSQL


The devil is
in the detail

THANK YOU

Q & A ?

https://bit.ly/2W7KGn1


NETCONOMY