Cosmos DB, Graph and Azure Search, building a compelling cloud solution

Steef-Jan Wiggers
Azure Technology Consultant
@SteefJan

codit

Nice to meet you

Steef-Jan Wiggers

Azure Technology Consultant

Codit Netherlands

steefjan.wiggers@codit.eu

+31 653 12 29 57

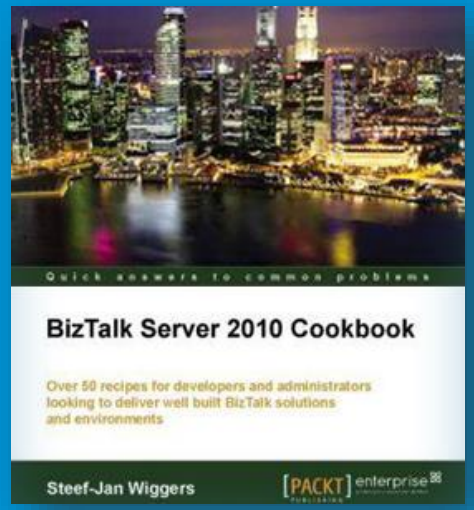
@SteefJan

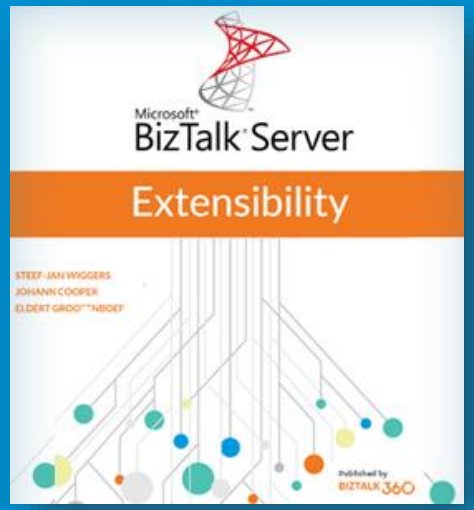
nl.linkedin.com/in/steefjan

Microsoft® MVP Most Valuable Professional

8th year - Azure MVP

InfoQueue





codit

Marathon Man

What can you expected in this session

- Scenario
- Cosmos DB
- Graph model
- Azure Search
- Demo

Scenario

Knowledge Platform

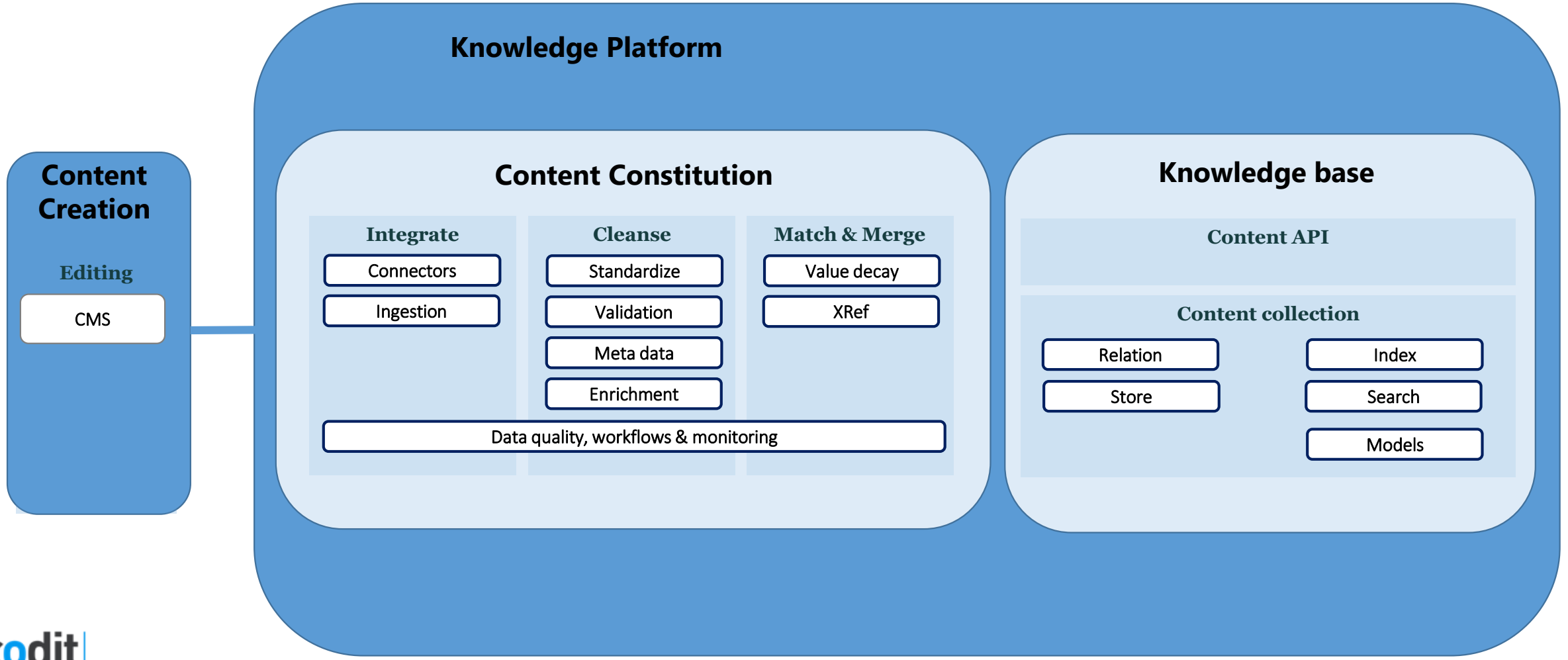
Scenario

- Old knowledge base implementation with SOLR struggled with related content
- Future proof new knowledge base (business case)
  - Completely PaaS
  - Azure (Pay as you go)
  - No IT management support only DevOps

Business Requirements Validation

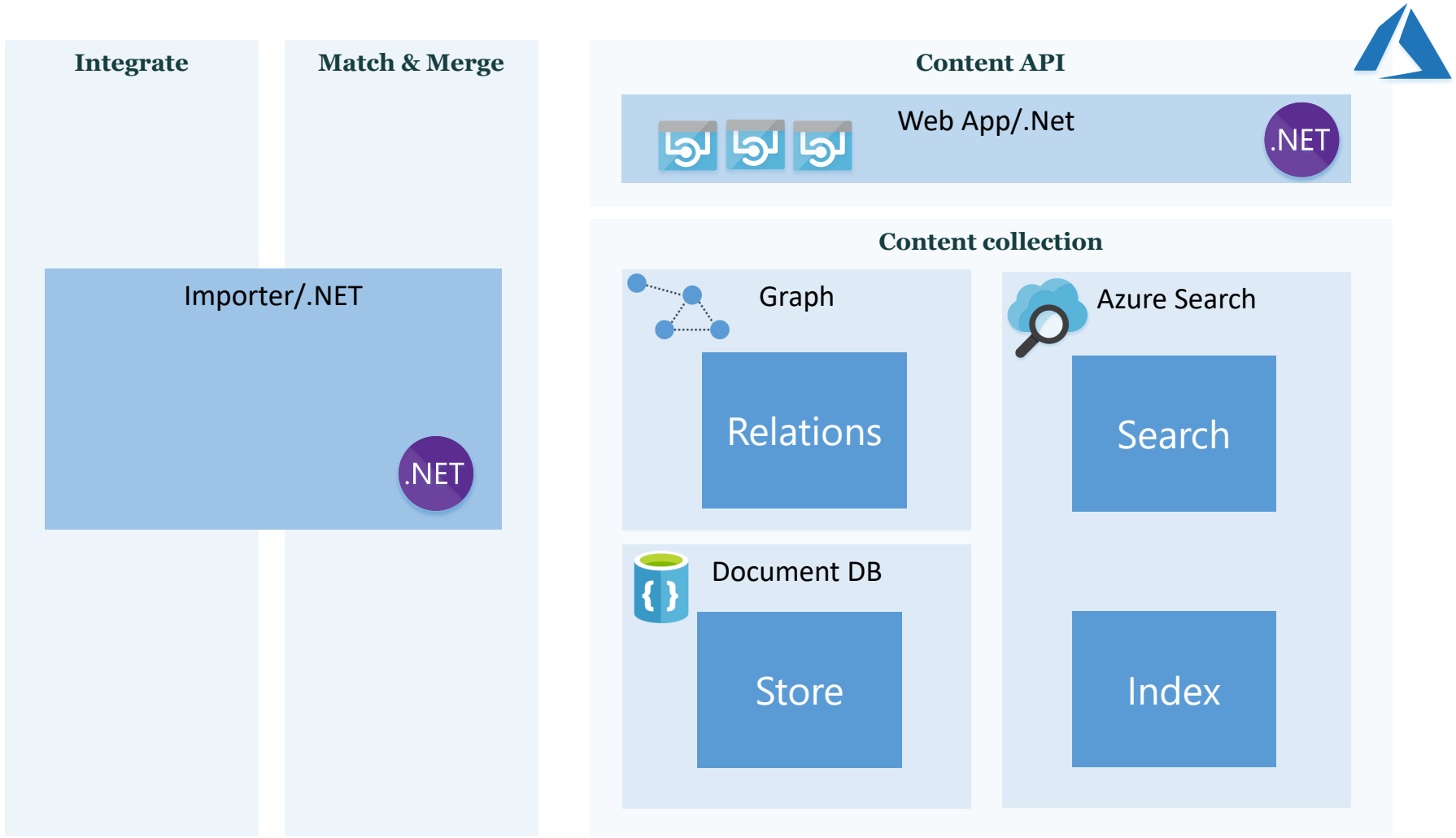
- Increase quality
- Justify investment:
  - Development
  - Deployments
  - Support
- No managed services

High Level Architecture

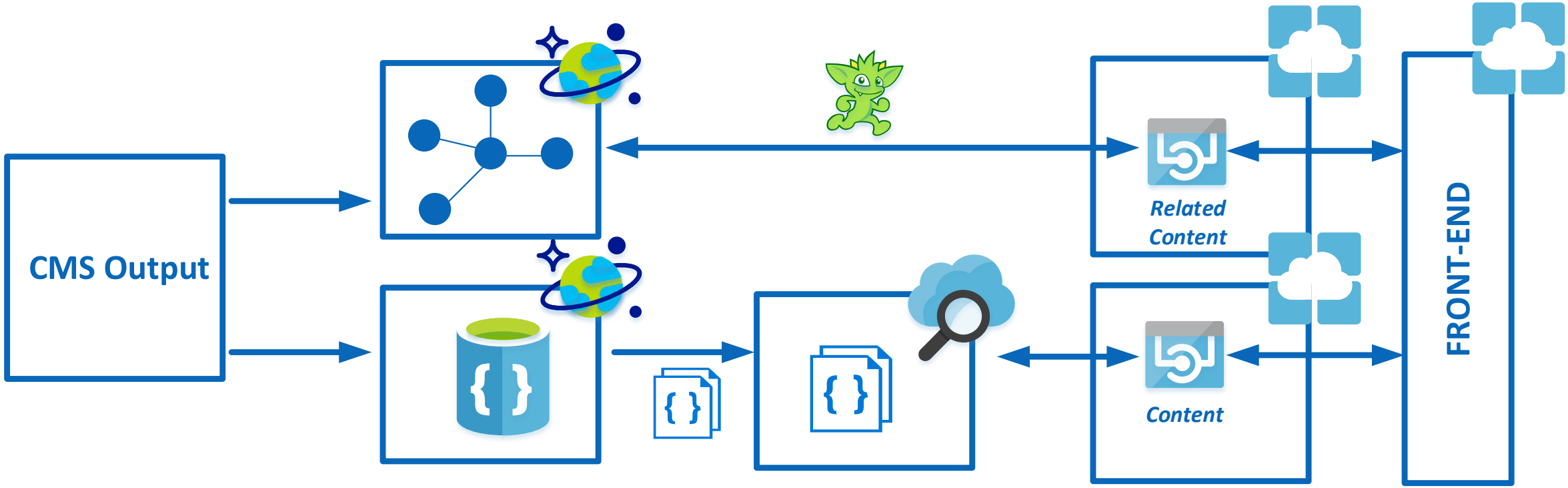


codit

Solution Building Blocks



Solution Architecture



codit

CosmosDB

CosmosDB

A globally distributed, massively scalable, multi-model database service

Table API

MongoDB API

Cassandra API

Key-value

Column-family

Document

Graph

Guaranteed low latency at the 99th percentile

Elastic scale out
of storage & throughput

Five well-defined consistency models

Turnkey global distribution

Comprehensive SLAs

codit

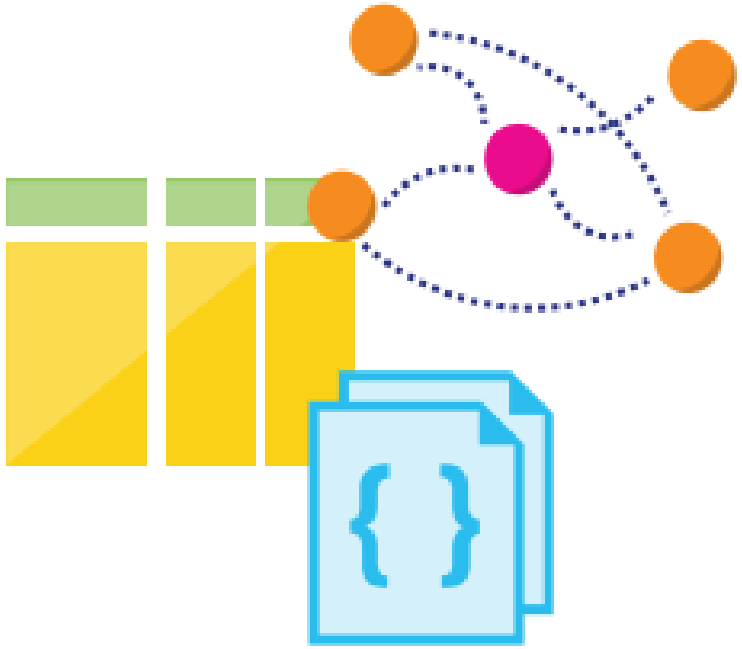
Global Distribution

- Turnkey global distribution
- Multiple datacenters
- Auto replication
- 99,99% Availability
- All resources are horizontally partitioned and vertically distributed
- Replication topology is dynamic based on consistency level and network conditions

codit

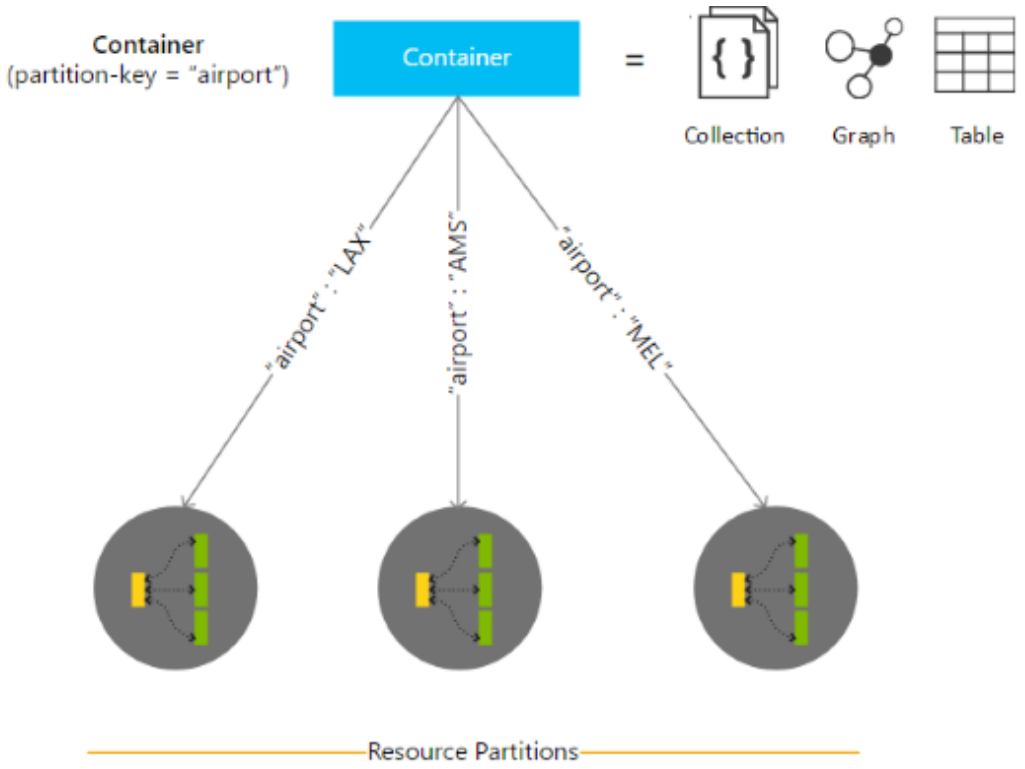
Multi-model + multi-API

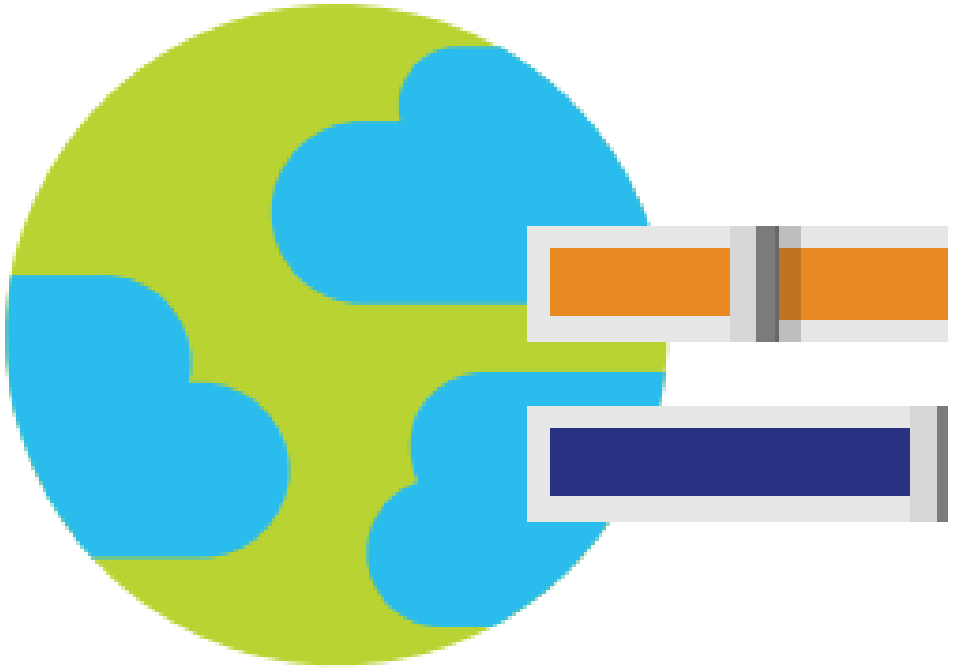
- Different models:
  - Graph
  - Key-Value
  - Document DB
- No schema or index management
- Automatic indexing
- API support:
  - SQL
  - JavaScript
  - Gremlin
  - MongoDB
  - Azure Table Storage
  - Cassandra



Scale

- Pay as go for storage and throughput
- Elastic scale across regions
- Partitions





Consistency

- Five levels of consistency
- Programmatically change at anytime
- Can be overridden on a per-request basis
- Writing correct distribution applications is hard
- Global distribution forces CAP theorem
- Intuitive and practical with clear PACELC tradeoffs

Latency

- Simultaneously read/write
- Latch free and write-optimized
  - 10-ms latency on read
  - 15-ms latency on writes

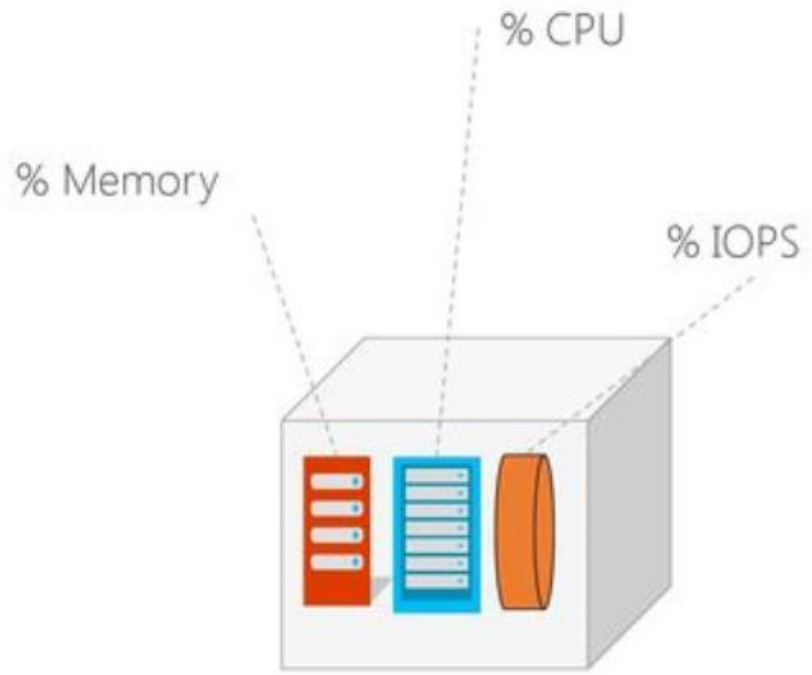
SLA

- Fully managed service
- 99,99% SLA for latency
- Guaranteed throughput, consistency and high availability

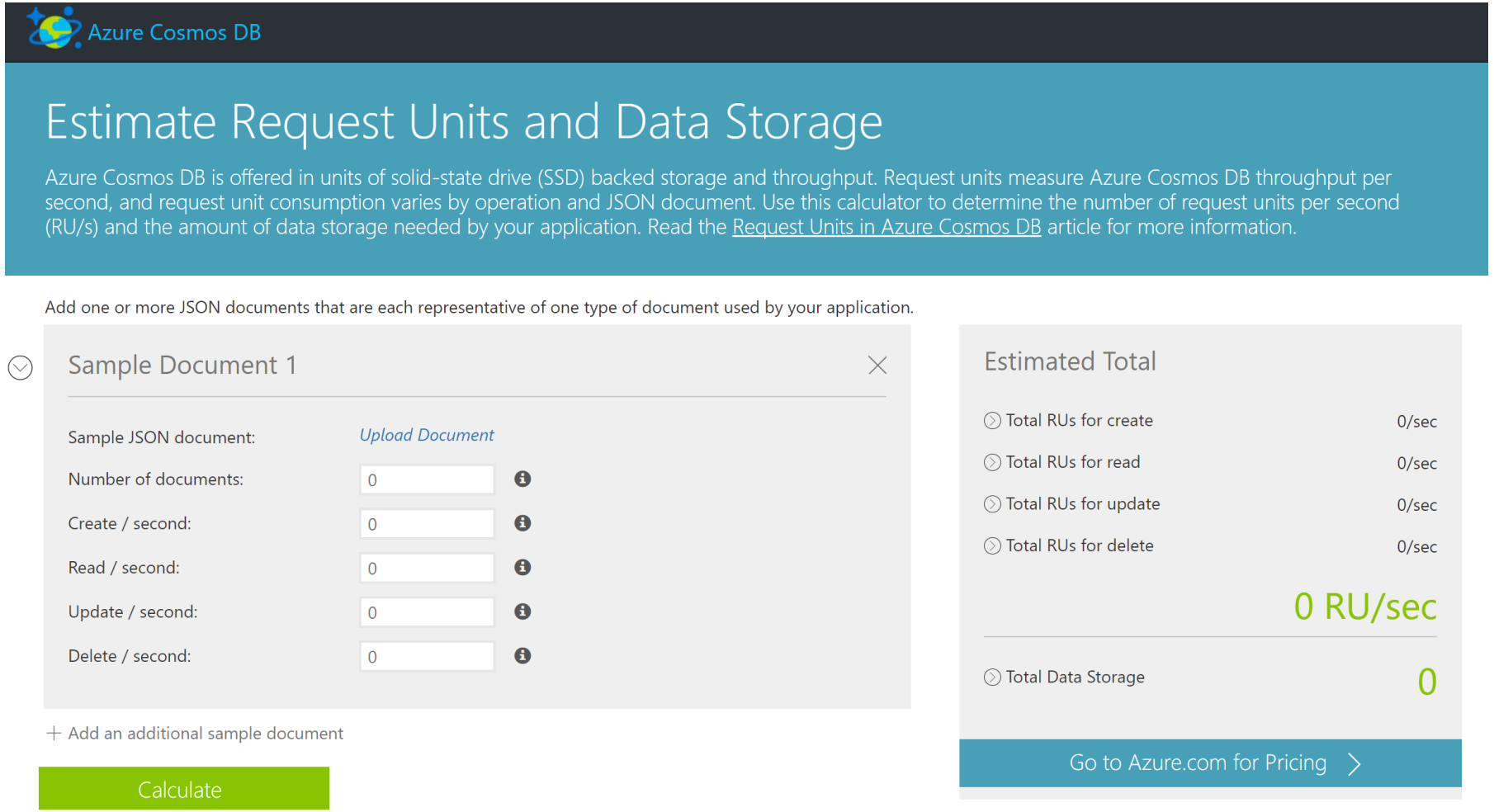
codit

Request Units

- Request Units (RU) is a rate-based currency
- Abstracts physical resources for performing requests
- 1 RU = 1 read of 1 Kb document
- Each request consumes fixed Rus
- Provisioned in terms of RU/sec and RU/min
- Rate limiting based on provisioned throughput
- Can be in- and decreased instantly
- Metered hourly



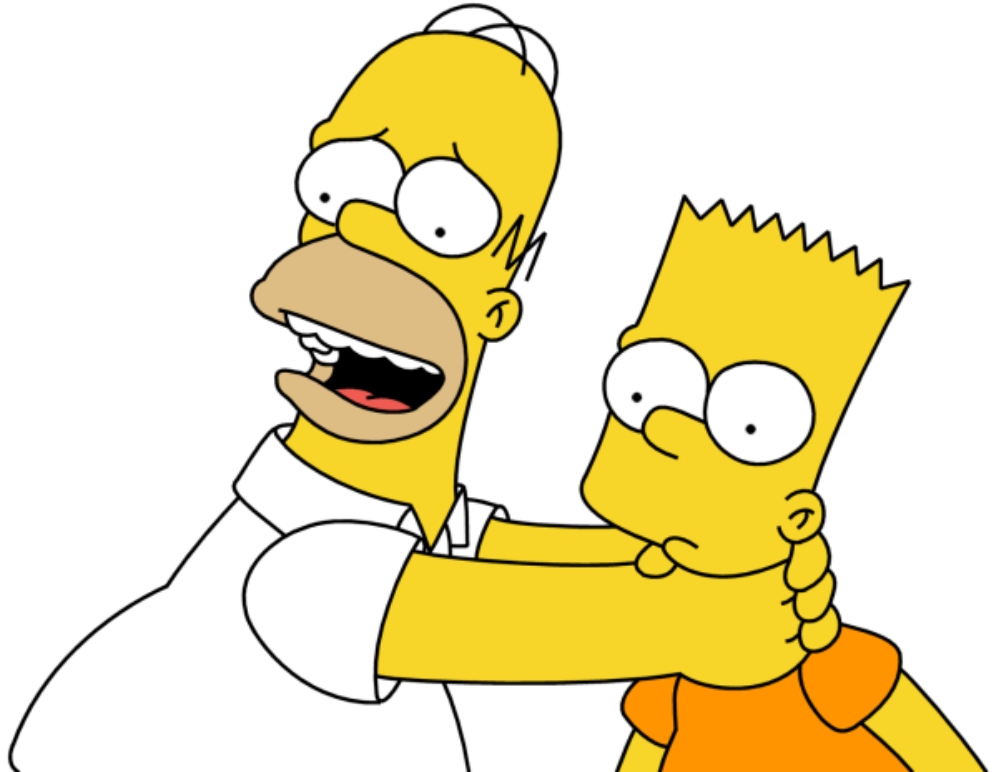
Capacity Planner



codit

Integrations

- Azure Search
- Apache Spark Connector
- Azure Functions



codit

Business Case challenges

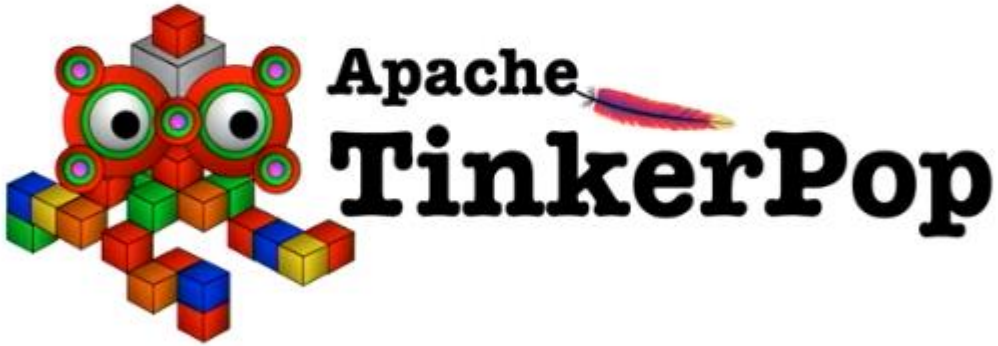
- What are the costs (ROI, TCO)
- Architectural Fit – POA
- Will it work for content and related content (Quality)
- Microsoft Support
- Training

codit

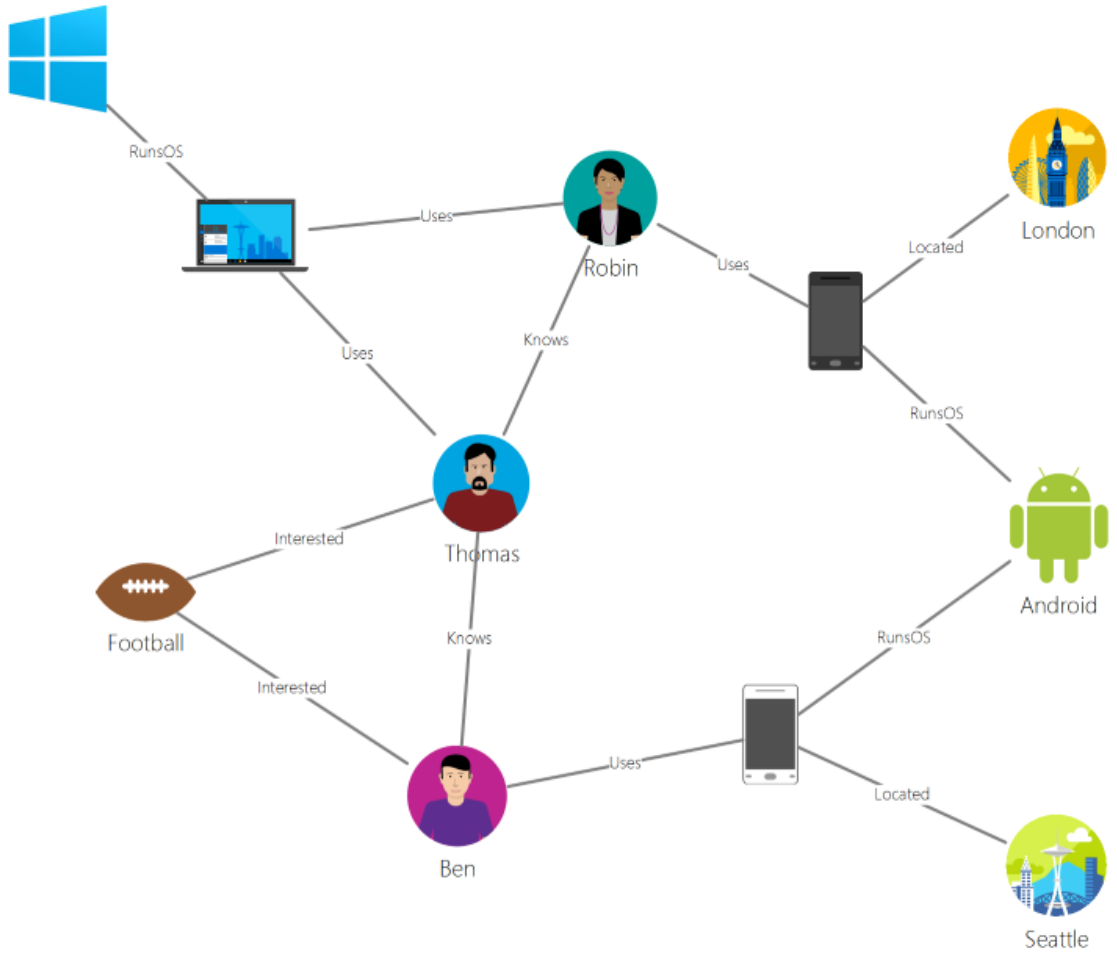
Graph Model

Graph

- TinkerPop is a developer group creating an open-source stack for graphs (http://tinkerpop.apache.org/)
- Graph database and analytics systems



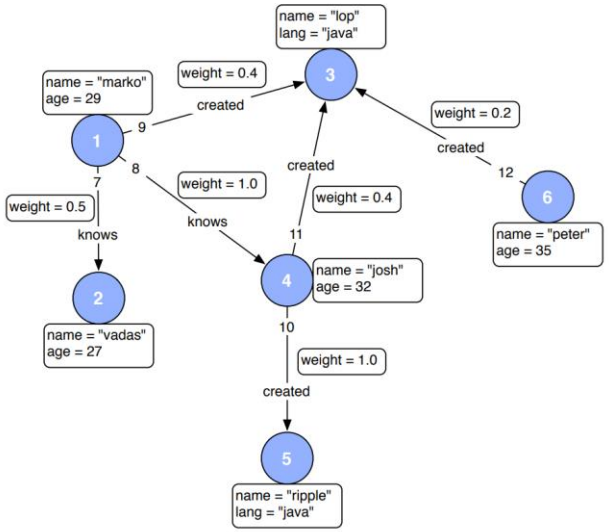
Data is your model, model is your data

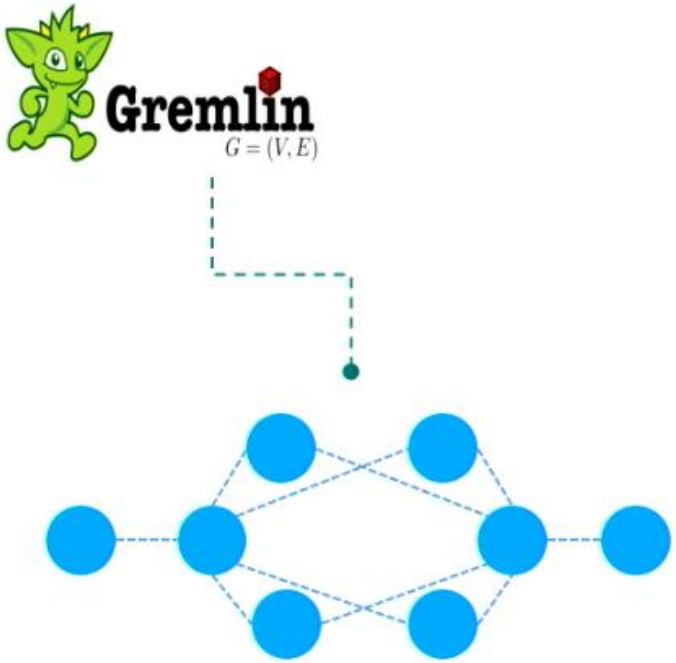


codit

Graph API

- Gremlin is a graph traversal language
- Vertices, Edges, and Properties

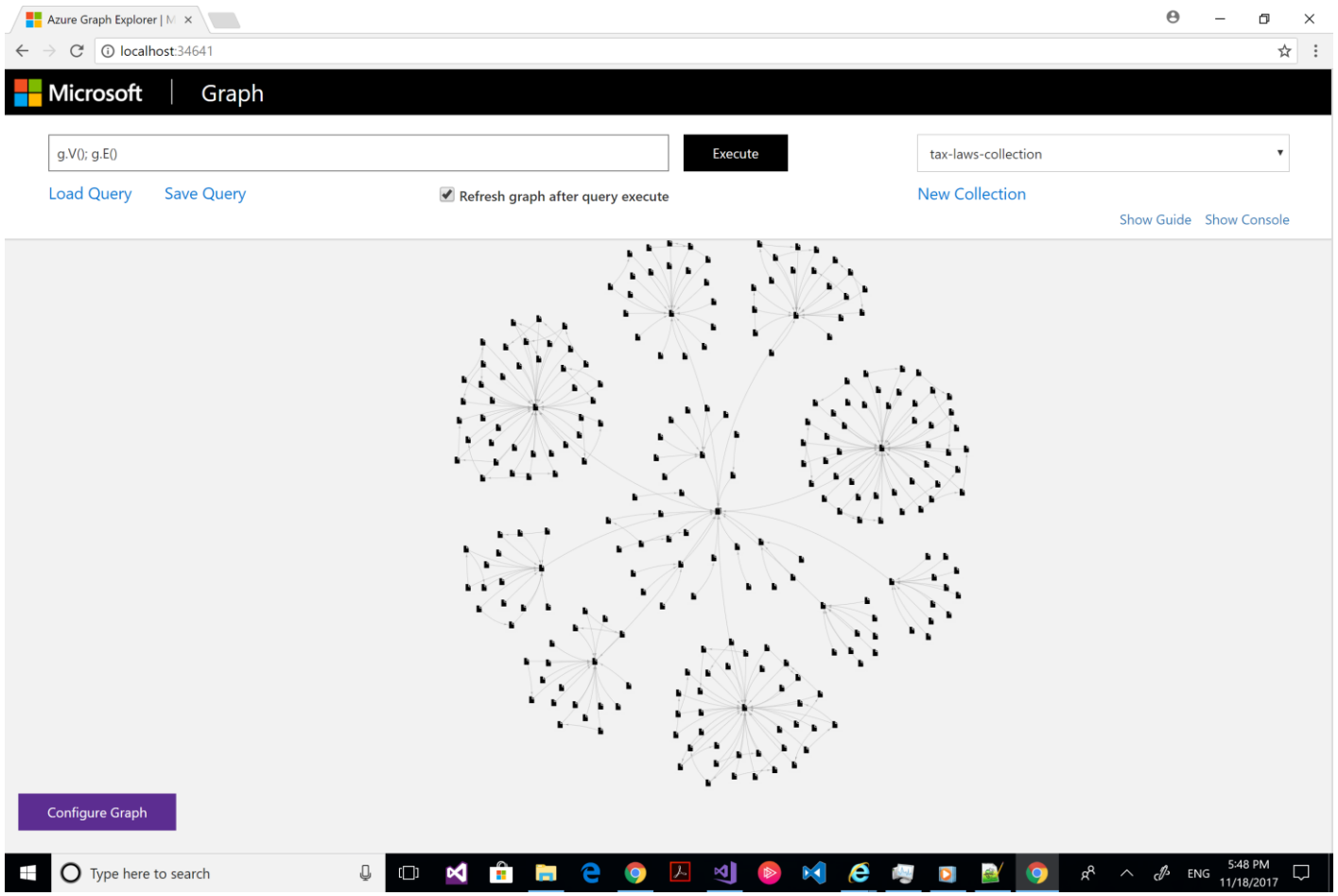




A **graph** is composed of **vertices** (nodes), **edges** (links), and **properties** (keys/values).

codit

Demo Graph Explorer



codit

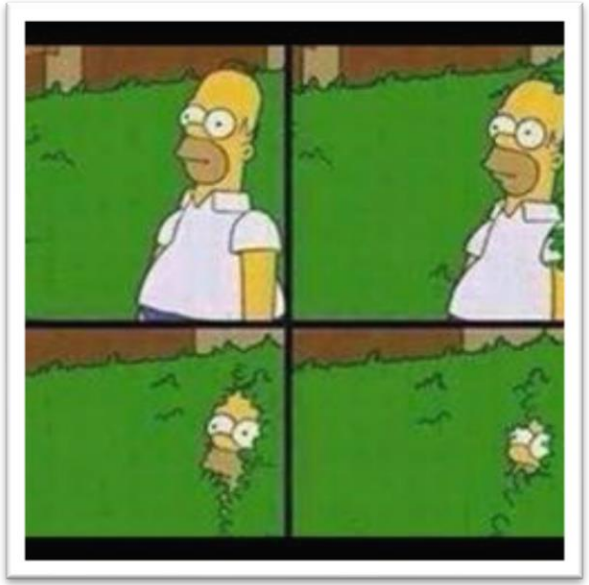
Business Case challenges

- Fit for purpose
- Performance
- Costs
- TCO
- Scale
- Complexity

Azure Search

Azure Search

- Provision service
- Create index
- Index data
- Search



Provisioning

"Search service"

- Scope for capacity
- Bound to a region
- Has keys, indexes, indexers, data sources

Provisioning

- Azure Portal
- Azure resource management API

Elastic scale

- Capacity can be changed dynamically
- Replicas ~ more QPS, HA
- Partitions ~ more documents, write throughput

Index

"Index"

- Container for data, think "table"
- Has schema, CORS options, search options
- Create in portal or during app initialization

Typical schema

- Fields definition: name, type, key

Search specifics

- Field attributes – searchable, facetable, etc.
- Linguistics and analysis
- Suggesters for auto-complete
- Scoring profiles for ranking tuning

Index data

Push - using indexing API

- POST to /indexes/<name>/docs/index
- Up to 1000 actions per batch
- Actions can be upload, merge, delete, etc.
- WebJobs are great for regular execution

Pull - using indexers

- Azure SQL DB and DocumentDB
- Change detection, deletion markers
- Point it at the data source, define policy, done

Search

Search + typical data operations

- Simple search options, + - * () “”
- Filter, sort, project, page over results
- Options work with search and suggest

Search from client or server

- Use query keys when searching from clients
- CORS allows direct calls from browsers

Render from search results

- Include necessary non-searchable data
- E.g. URLs for pictures, keys to main content

Customization

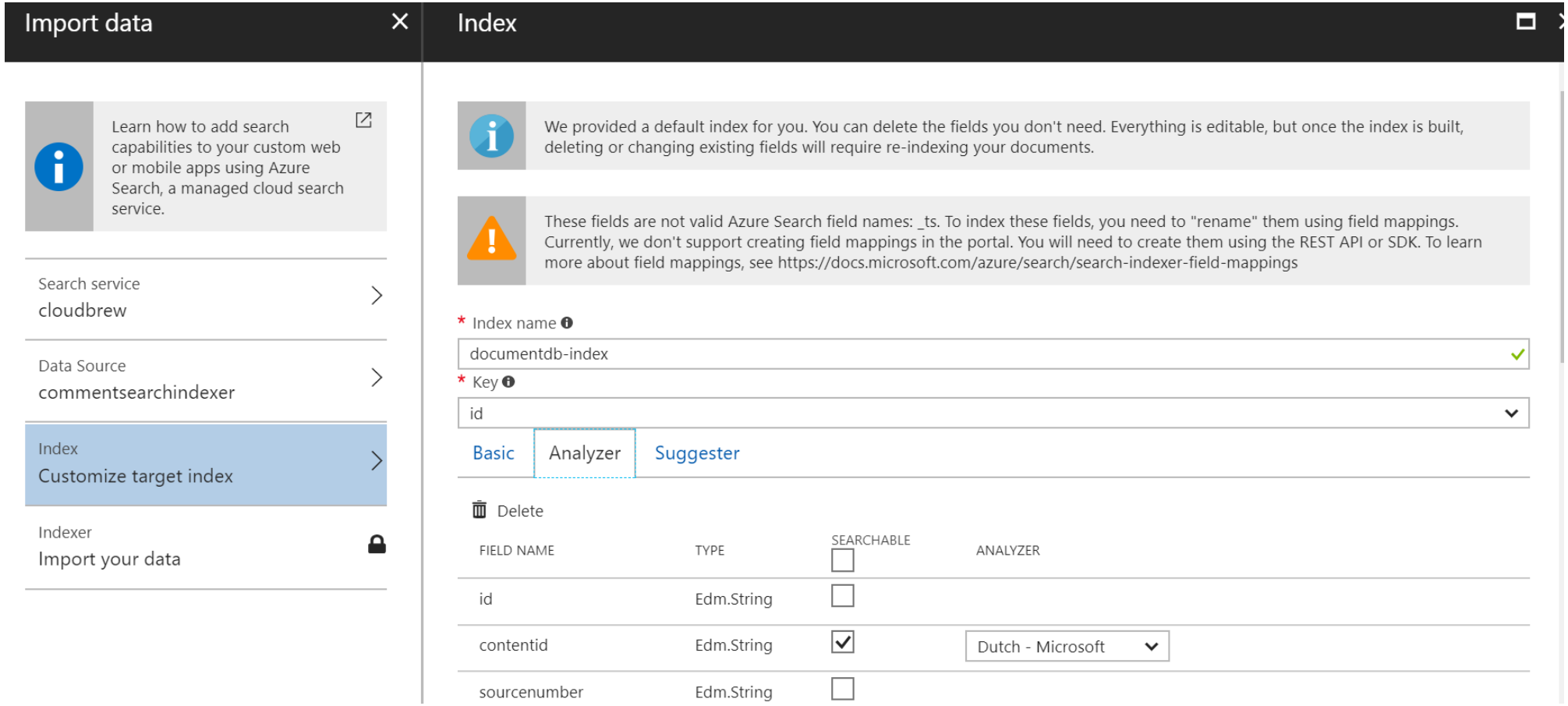
Scoring profiles

- Field weights
- Scoring functions
- magnitude, freshness, distance, tags

3 main patterns

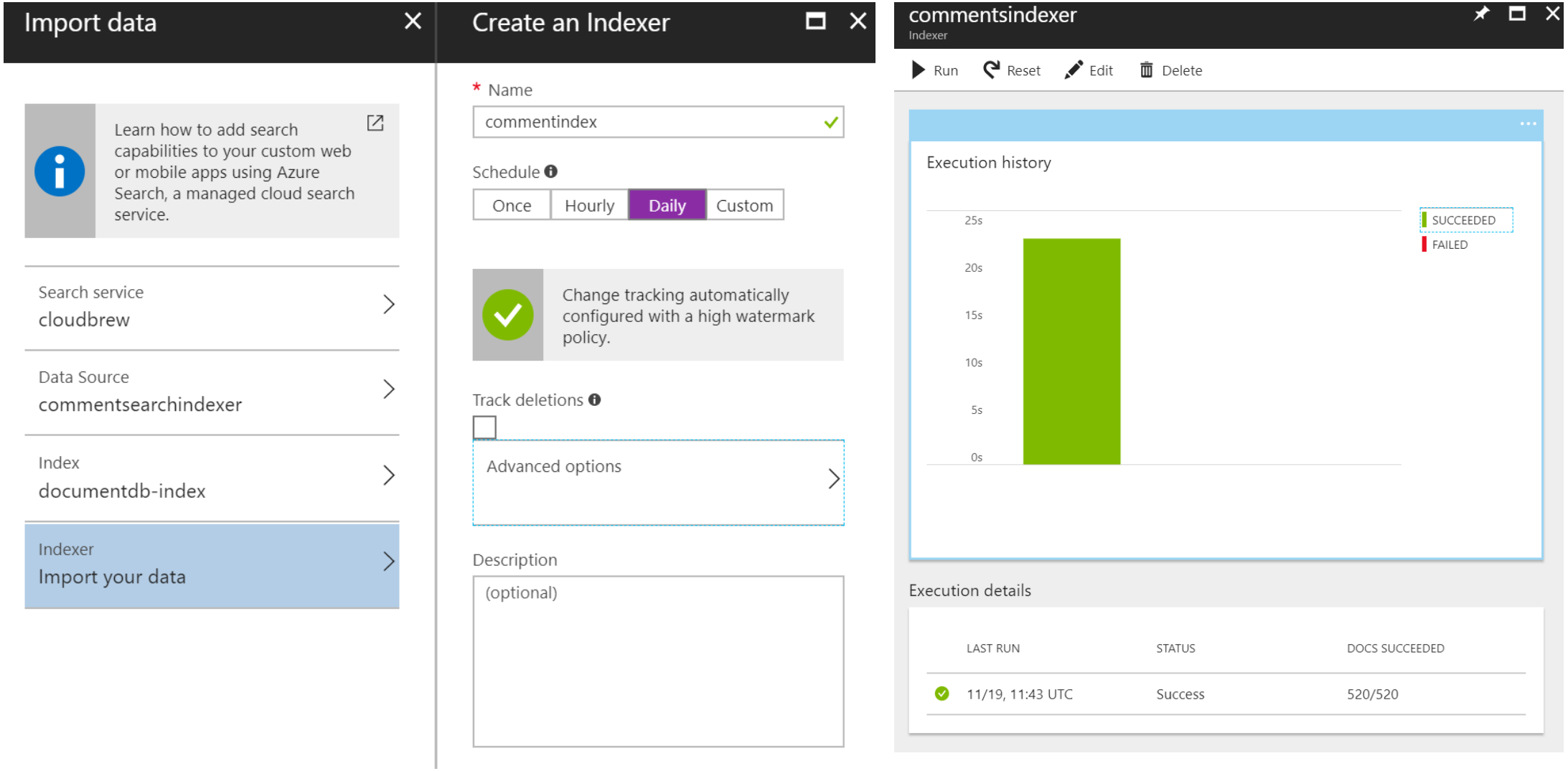
- Known data directly available in the index
- Personalization using tag boosting
- Analytics, compute externally and push to the index

Integrate with Azure Search



codit

Integrate with Azure Search



codit

Demo Search

POST https://cloudbrew.search.windows.net/indexes/documentdb-index/docs/search?api-version=2016-09-01 Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2     "search": "ftid eq 'Dividenduitkeringen'",
3     "filter": "fttypeid eq 16"
4 }
```

Body Cookies Headers (13) Test Results Status: 200 OK Time: 367 ms

Pretty Raw Preview JSON

```
1 {
2     "@odata.context": "https://cloudbrew.search.windows.net/indexes('documentdb-index')/$metadata#docs",
3     "value": [
4         {
5             "@search.score": 1.0133729,
6             "id": "ODgzMmYxMTAtZGQxMy00ZDIwLWFjMWYtZjZkMmU0NjNiNTY10",
7             "contentid": "Dividenduitkeringen",
8             "ftid": "Dividenduitkeringen",
```

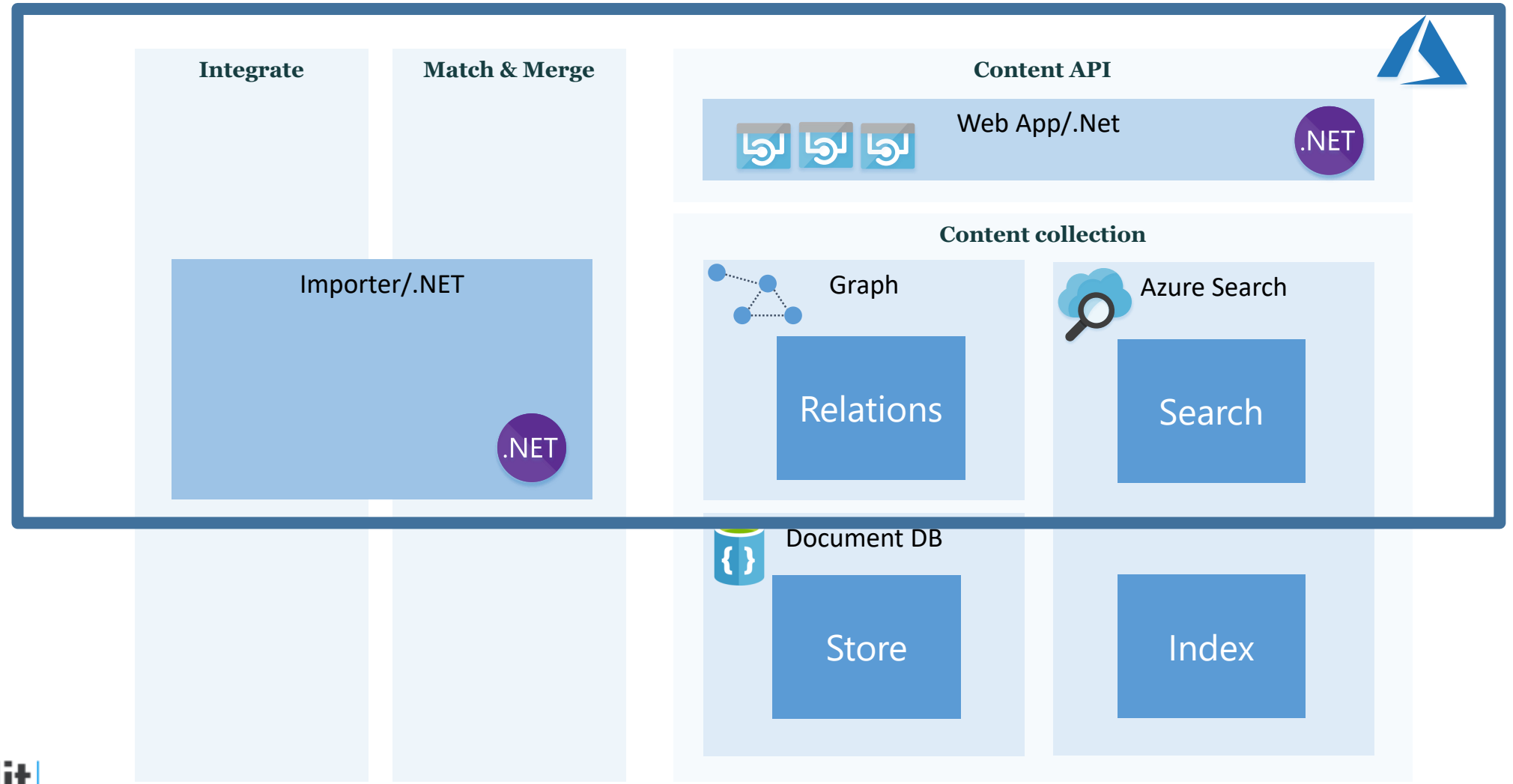
codit

Business Case challenges

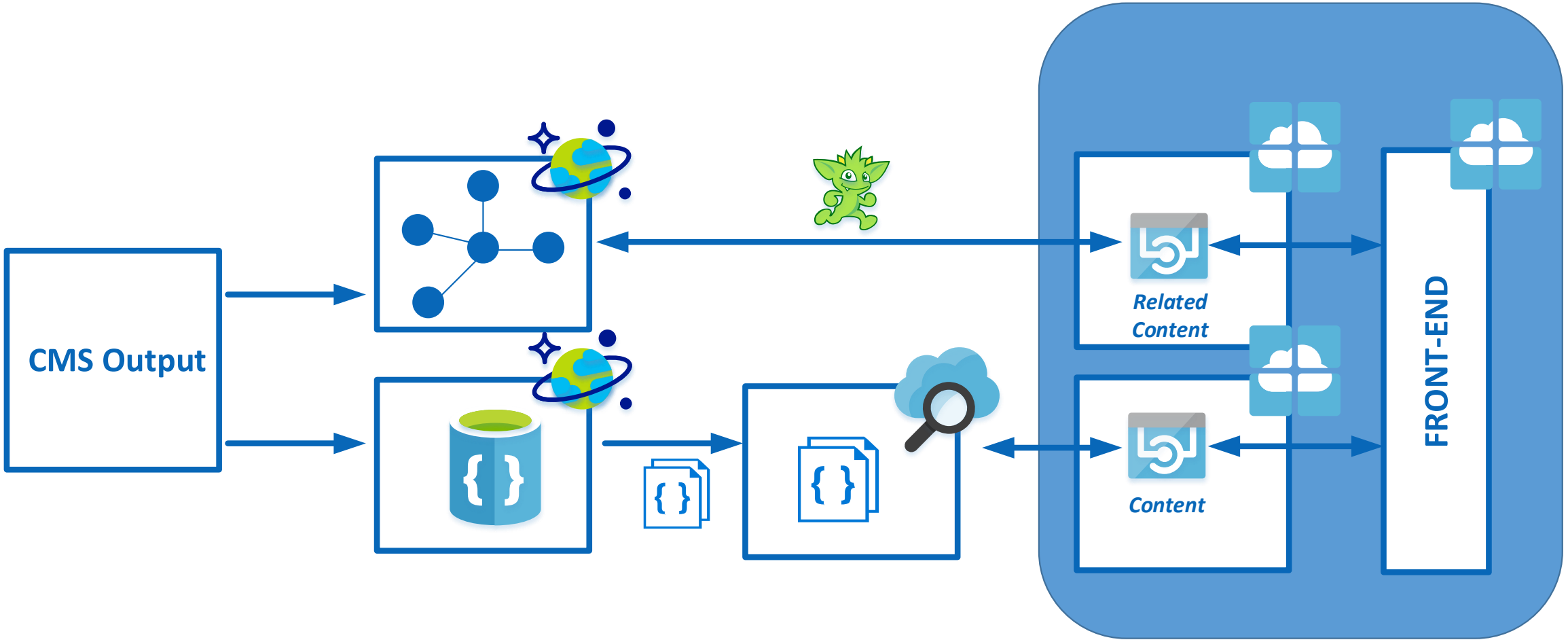
- Meet requirements
- Costs (ROI, TCO)
- Compare with other search solutions

Scenario Revisted

Development

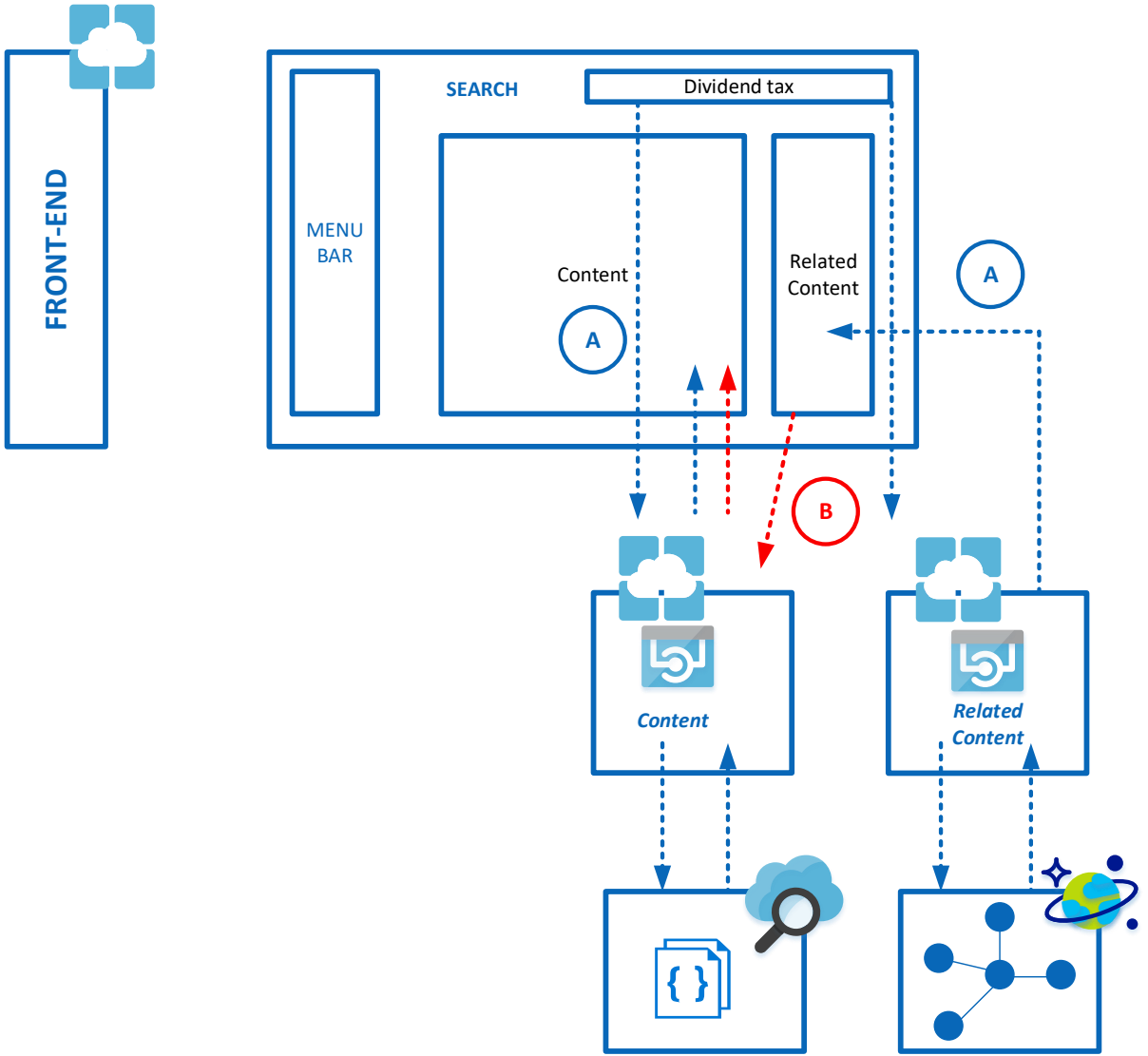


Solution Architecture



codit

Search Content



codit

Demo API

```
public async Task<List<VertexModel>> GetAllContentByContentType(int contenttype)
{
    var vertices = await GraphRepository.RunGremlinQuery<VertexEntity>($"g.V().hasLabel('publication').has('contenttype', {contenttype})");
    return VertexModelFactory.CreateVertexModel<VertexModel>(vertices);
}
```

```
public async Task<IEnumerable<T>> RunGremlinQuery<T>(string gremlin)
{
    Exception exception;

    using (var client = new DocumentClient(new Uri(_graphDatabaseUrl), _graphDatabaseSecret,
        new ConnectionPolicy { ConnectionMode = ConnectionMode.Gateway, ConnectionProtocol = Protocol.Https }))
    {
        try
        {
            await client.OpenAsync();

            var graph = await client.CreateDocumentCollectionIfNotExistsAsync(
                UriFactory.CreateDatabaseUri(_graphDatabase),
                new DocumentCollection {Id = _graphDatabaseCollection},
                new RequestOptions {OfferThroughput = 400});

            var queryString = gremlin;

            var query = client.CreateGremlinQuery<Vertex>(graph, queryString);

            var results = new List<dynamic>();

            while (query.HasMoreResults)
            {
                results.AddRange(await query.ExecuteNextAsync());
            }

            var json = JsonConvert.SerializeObject(results);

            return JsonConvert.DeserializeObject<IEnumerable<T>>(json);
        }
```

codit

Summary

- Cosmos DB Graph fit for related content purpose
- Cosmos DB Document fit for content
- Cosmos DB + Search good combination
- Meets business requirements
- Complete Architecture on PaaS
- Cool eh!

Resources

- CosmosDB
- CosmosDB Graph Explorer
- Azure Search
- Azure Search Resources

Thank you!

Keep in touch.

Call or mail us. Ask us. Happy to help.