CosmosDB: использование в реальном проекте

DODO ENGINEERING Анна Морозова

Dodo Brands – это про IT

Dodo Brands – компания,
которая развивает три бренда:
Додо Пицца, Дринкит
и Донер 42, на базе единой
инфраструктуры и самописной
цифровой платформы Dodo IS.
Dodo Engineering – команда
разработки.

15 стран мира

26 млрд выручки
за 2020

700+ точек питания

16 млн клиентов

600+ человек
в Dodo Brands

Dodo IS

3000 запросов в секунду

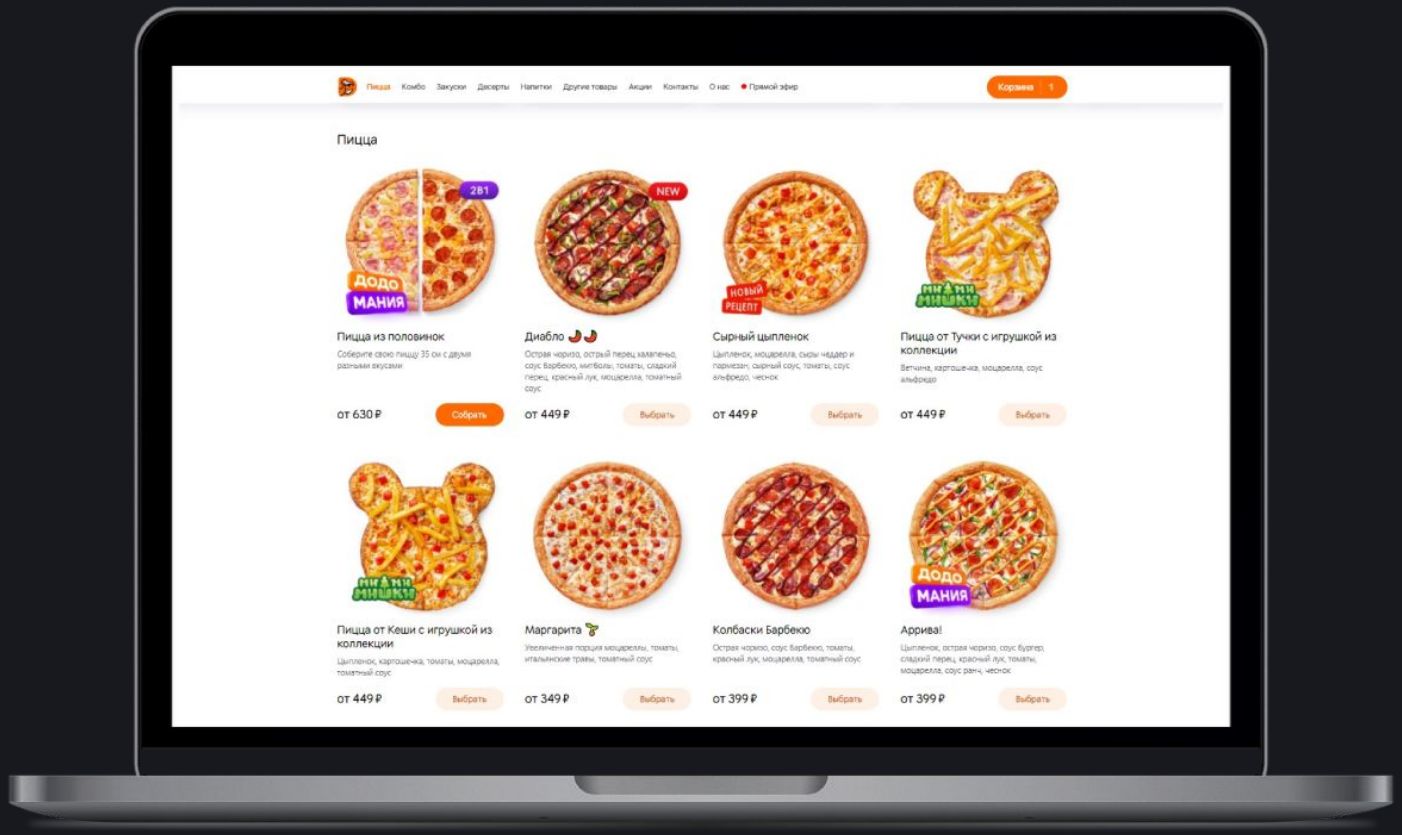
200–250 заказов в минуту

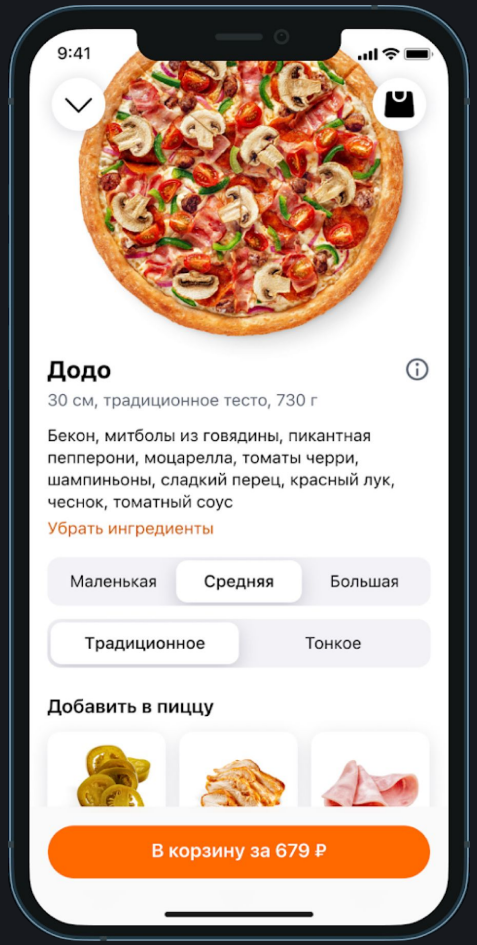
470 – максимальная нагрузка заказов в минуту

2 датацентра

40+ сервисов

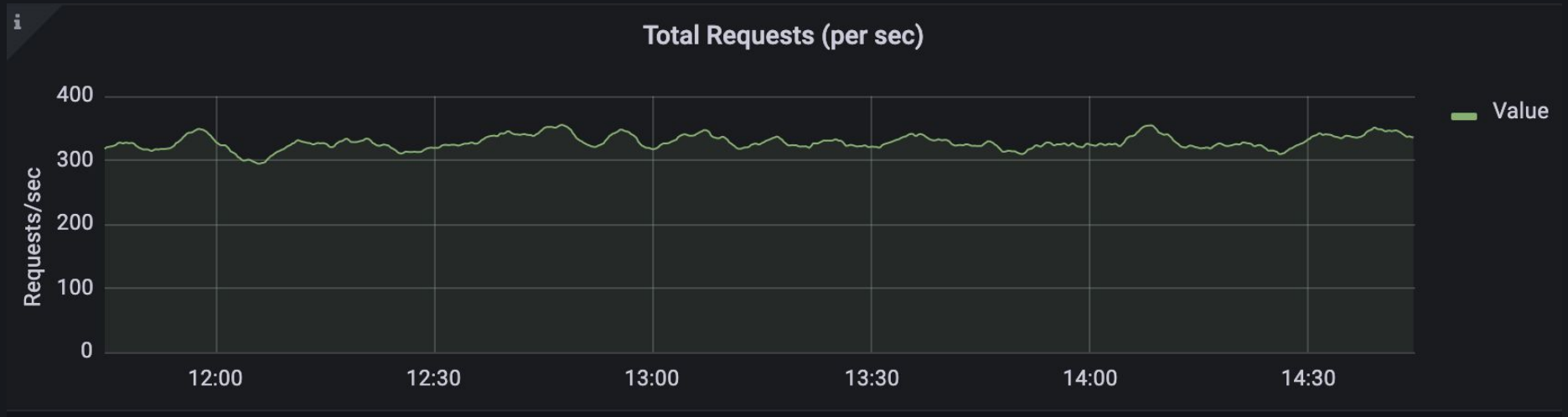
160+ человек в Dodo Engineering


Пицца
Комбо
Закуски
Десерты
Напитки
Другие товары
Акции
Контакты
О нас
Прямой эфир
Корзина 1
Пицца
281
Додо Мания
Пицца из половинок
Соберите свою пиццу 35 см с двумя разными вкусами
от 630 ₽
Собрать
NEW
Диабло
Острая чоризо, острый перец халапеньо, соус Барбекю, мятболы, томаты, сладкий перец, красный лук, моцарелла, томатный соус
от 449 ₽
Выбрать
Новый рецепт
Сырный цыпленок
Цыпленок, моцарелла, сыры чеддер и пармезан, сырный соус, томаты, соус альфредо, чеснок
от 449 ₽
Выбрать
Ники Микки
Пицца от Тучки с игрушкой из коллекции
Ветчина, картошечка, моцарелла, соус альфредо
от 449 ₽
Выбрать
Ники Микки
Пицца от Кеши с игрушкой из коллекции
Цыпленок, картошечка, томаты, моцарелла, томатный соус
от 449 ₽
Выбрать
Маргарита
Увеличенная порция моцареллы, томаты, итальянские травы, томатный соус
от 349 ₽
Выбрать
Колбаски Барбекю
Острая чоризо, соус барбекю, томаты, красный лук, моцарелла, томатный соус
от 399 ₽
Выбрать
Додо Мания
Аррива!
Цыпленок, острая чоризо, соус бургер, сладкий перец, красный лук, томаты, моцарелла, соус ранч, чеснок
от 399 ₽
Выбрать

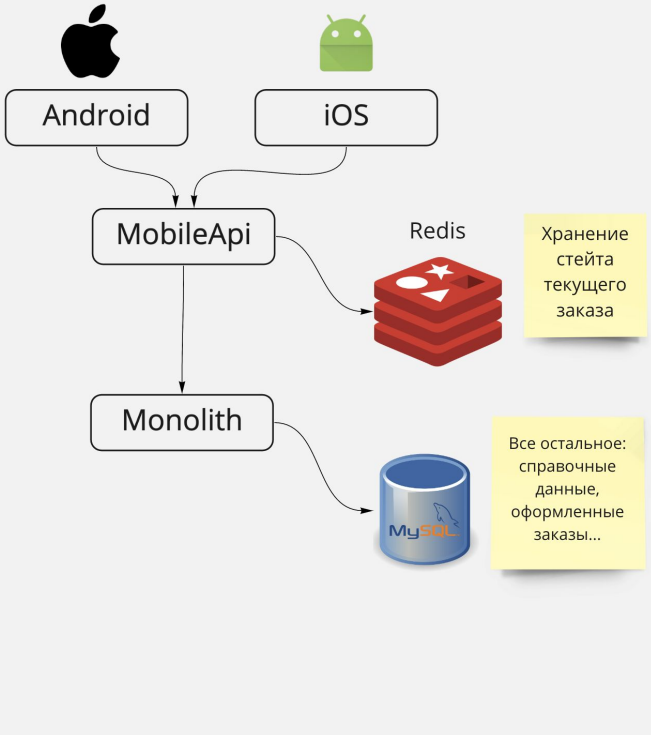


- .NET developer в мобильной команде с 2019
- b2c lead с июня 2021

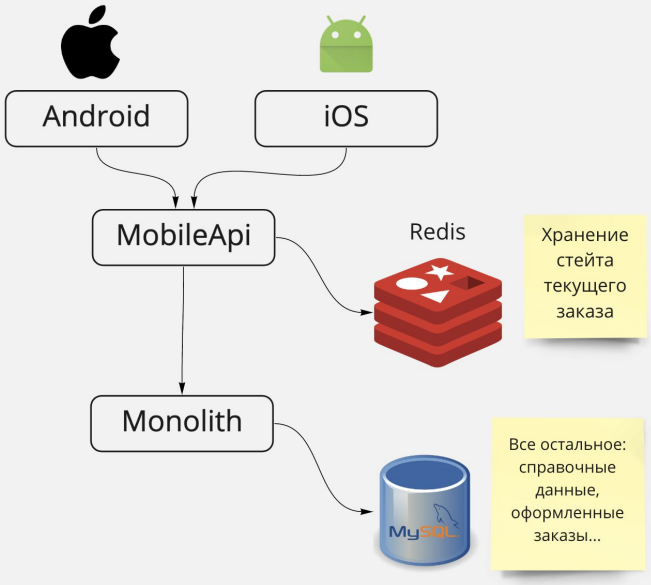
Mobile Api – высоконагруженный сервис



БЫЛО

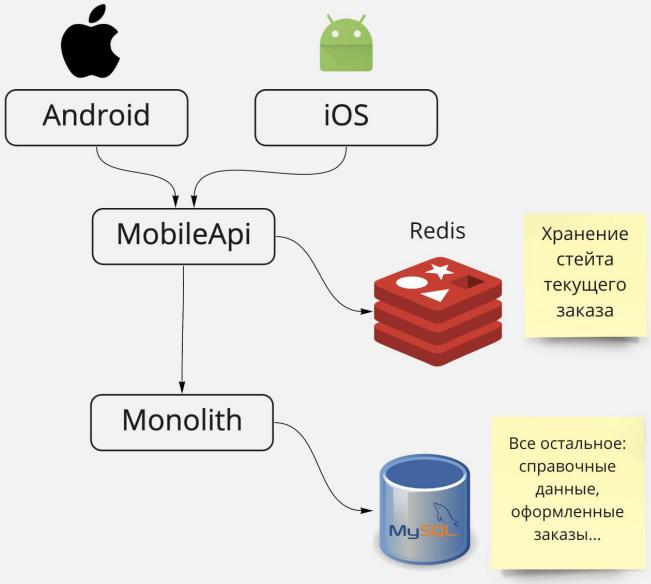

Android
iOS
MobileApi
Redis
Хранение стейта текущего заказа
Monolith
MySQL
Все остальное: справочные данные, оформленные заказы...

БЫЛО



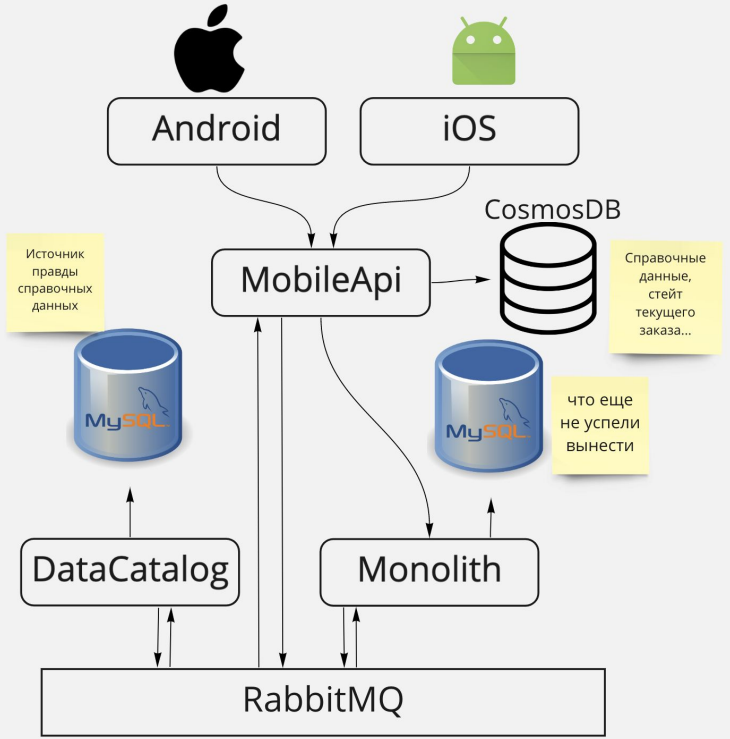
- Один долгий запрос к редису может привести к торможению остальных
- Хождение по grpc к монолиту за данными
- Устаревшие данные из-за сильного кеширования
- Единая точка отказа

БЫЛО



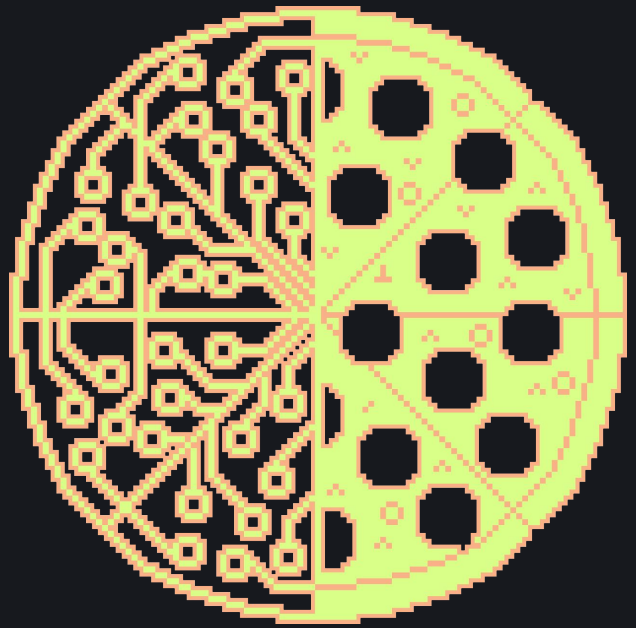
- Один долгий запрос к редису может привести к торможению остальных
- Хождение по grpc к монолиту за данными
- Устаревшие данные из-за сильного кеширования
- Единая точка отказа

СТАЛО



Преимущества

- База данных как сервис
- SLA на доступность 99,999%
- Дешево
- Api на любой вкус и цвет
- Георасределенность
- Есть во всех регионах Azure



Начинаем настраивать

Создаем базу

New Database

* Database id ⓘ

dotnext2021

☑ Provision throughput ⓘ

* Database throughput (400 - unlimited RU/s) ⓘ

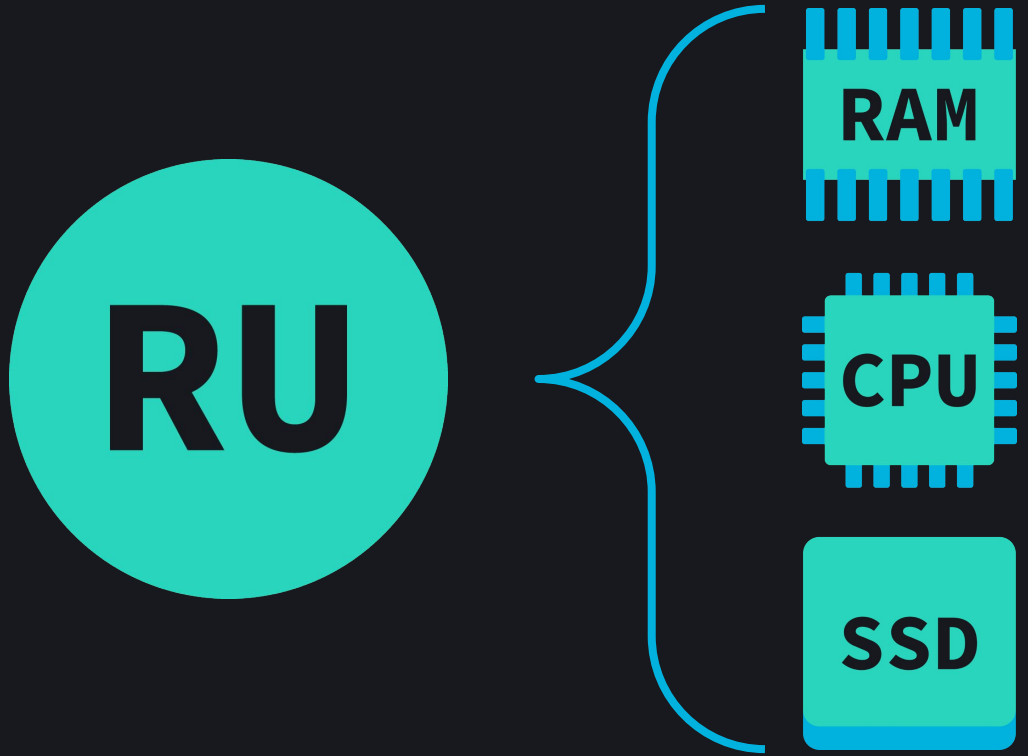
○ Autoscale ◉ Manual

Estimate your required RU/s with capacity calculator.

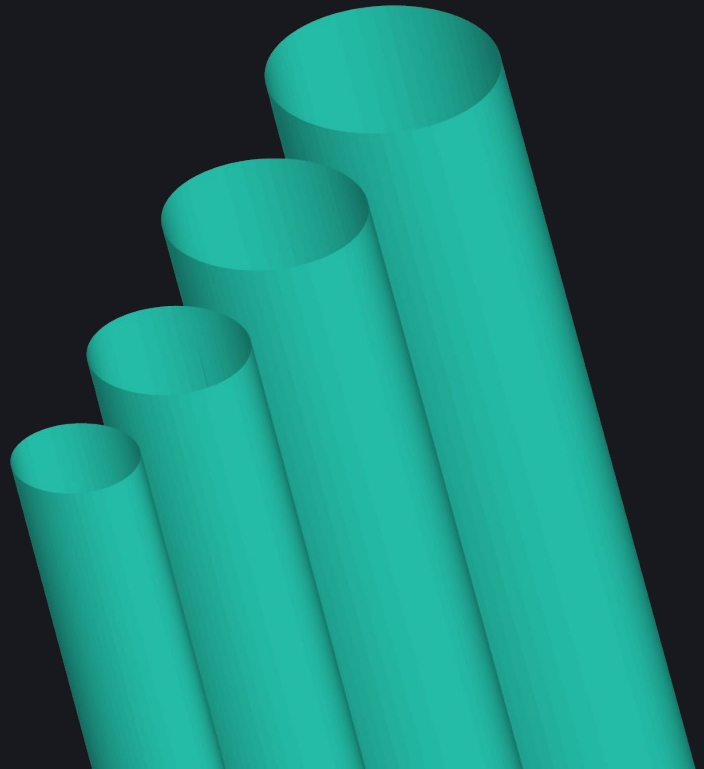
400 *

Estimated cost (USD) ⓘ: **$0.032 hourly / $0.77 daily / $23.36 monthly** (1 region, 400RU/s, $0.00008/RU)

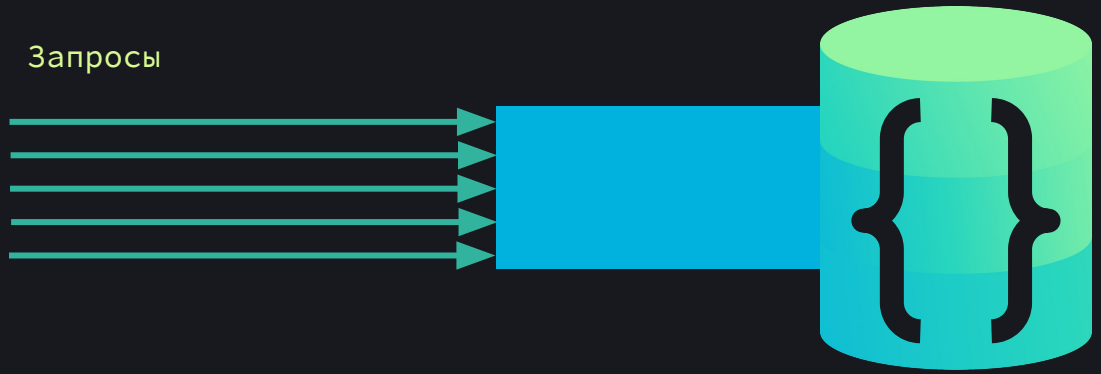
Что такое RU



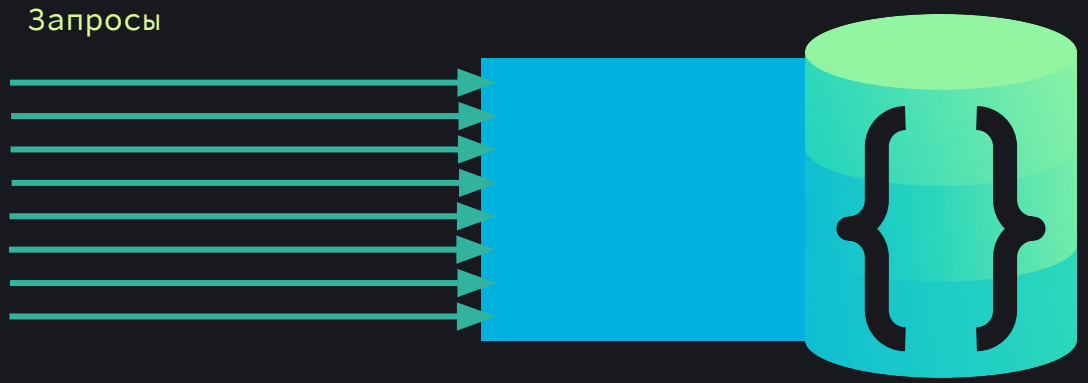
Provisioned throughput



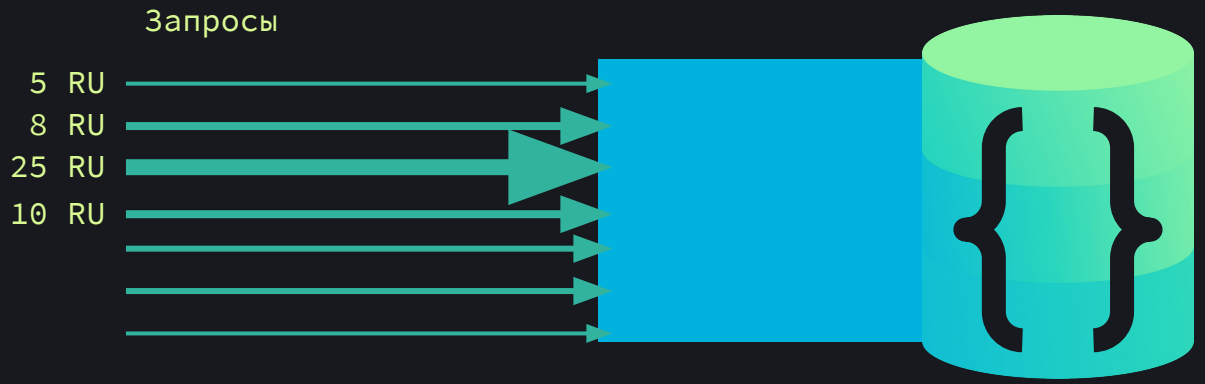
Provisioned throughput



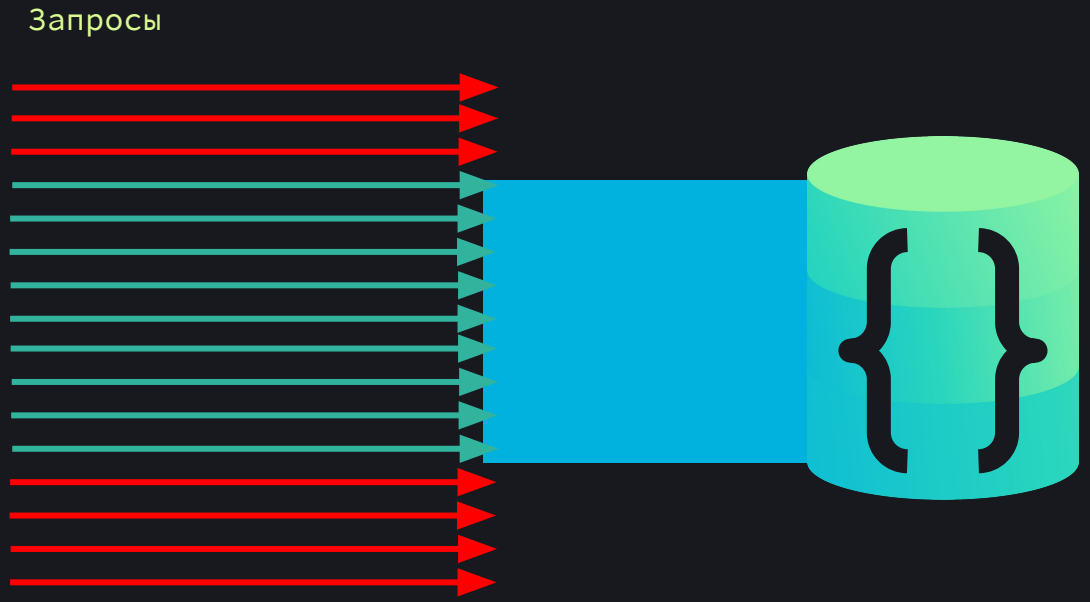
Меняем размер трубы

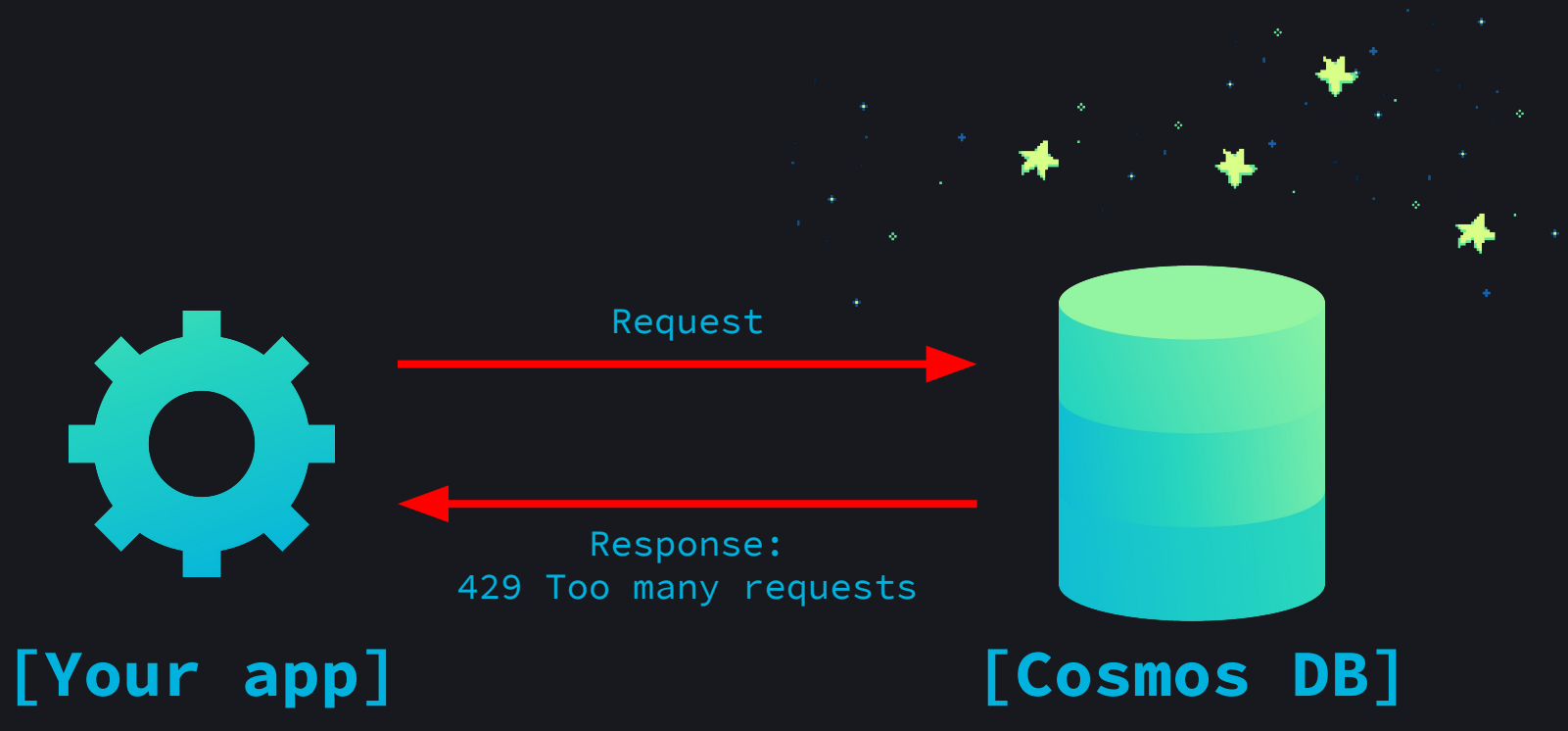


Разные запросы – разный вес



Что если превысить?

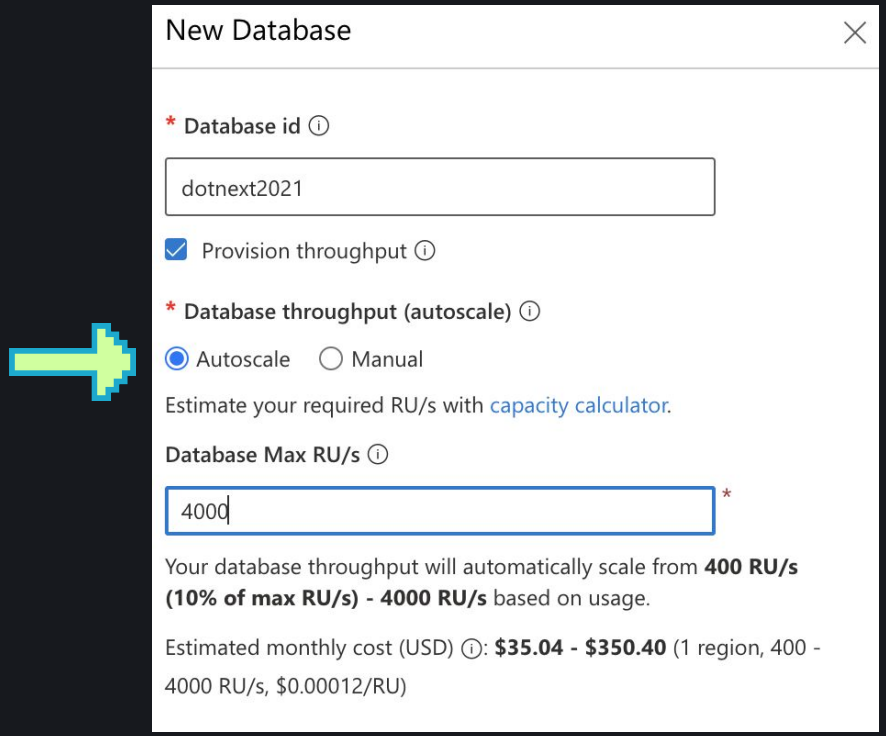



Request
Response:
429 Too many requests
[Your app]
[Cosmos DB]

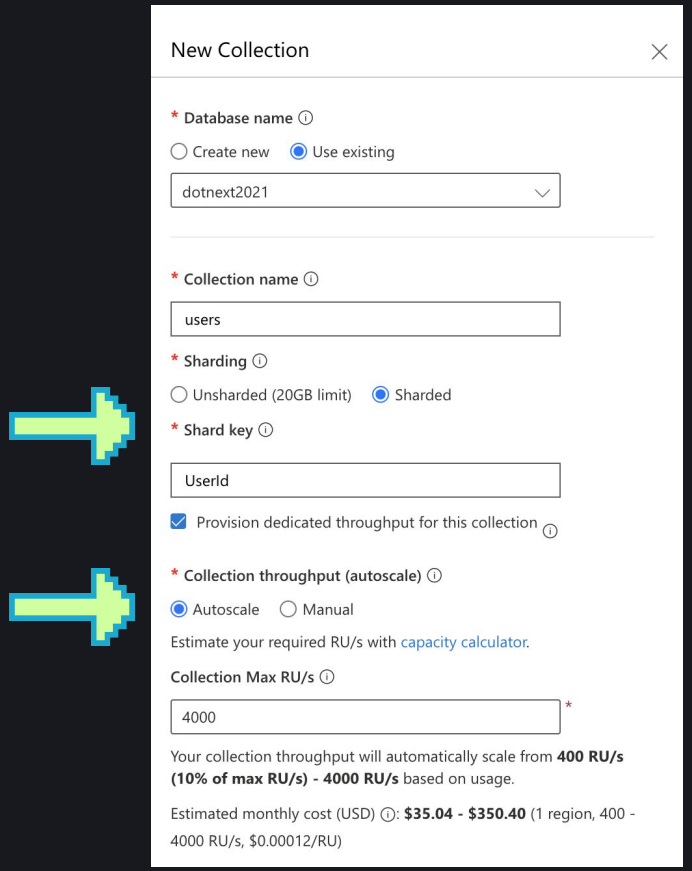
Коды ответа

- 200 OK
- 201 Created
- 400 Bad Request
- 401 Unauthorized
- 429 Too Many Requests
- 500 Internal Server Error
- И другие…

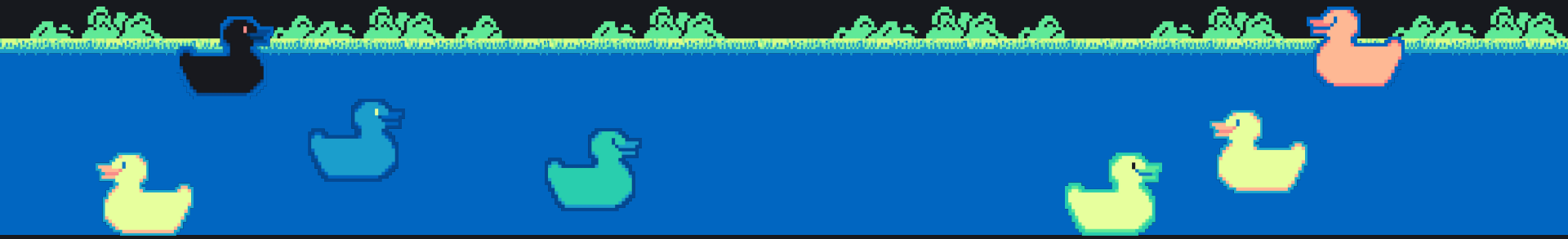
Создаем базу с автопилотом



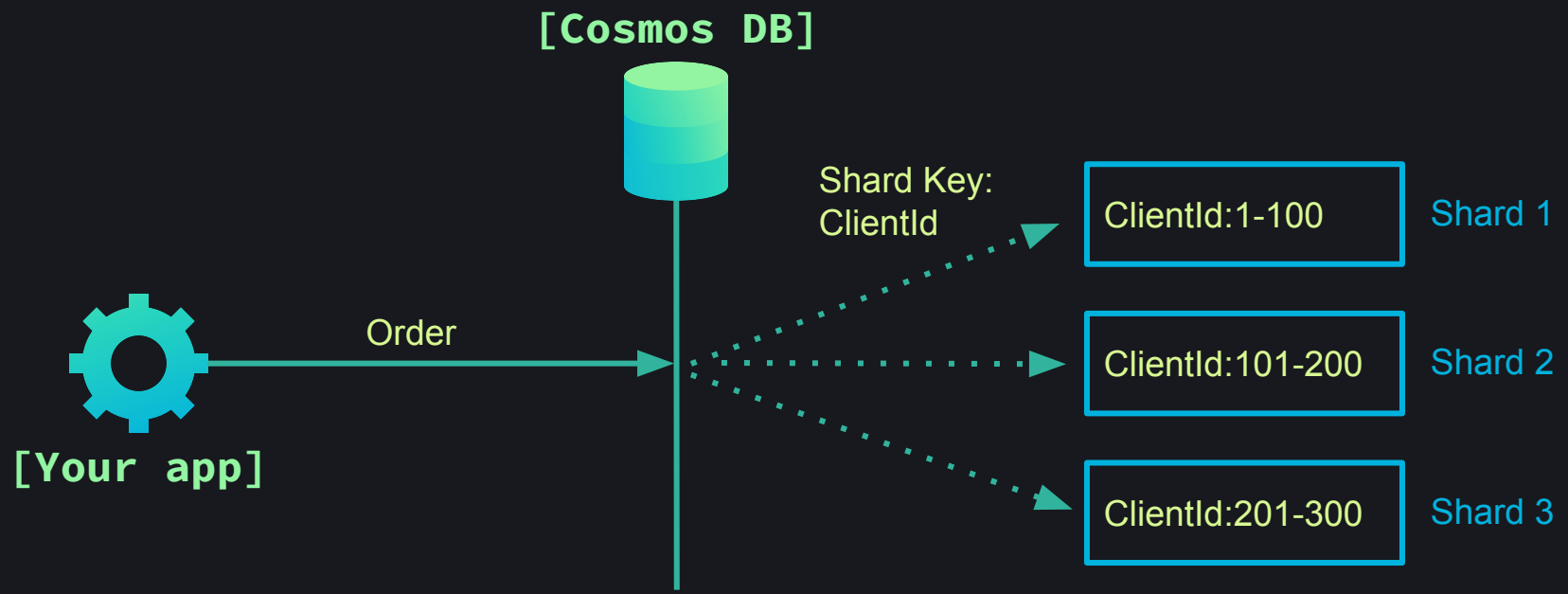
Создаем коллекцию



Шардирование

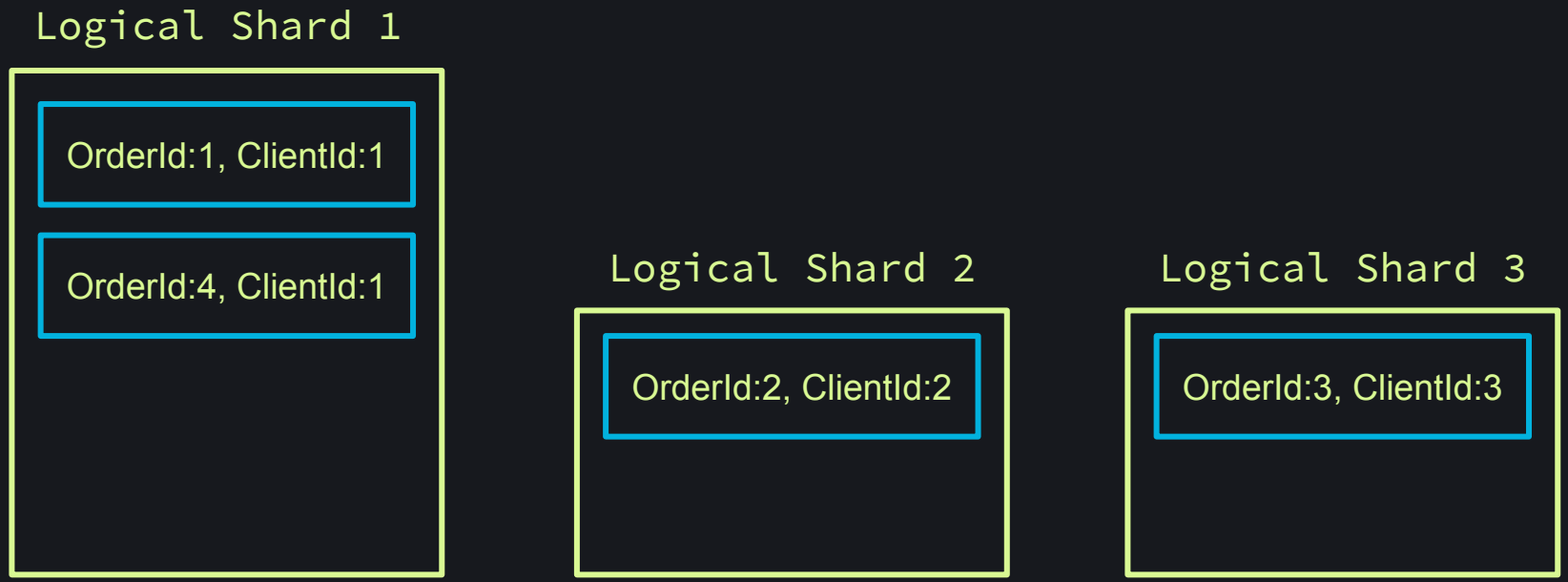


Что такое шардирование?



Логические шарды

ShardKey = ClientId



Как выбрать ShardKey

- Это поле есть при каждом запросе к базе

Как выбрать ShardKey

- Это поле есть при каждом запросе к базе
- Равномерное распределение

Плохой ShardKey

- Id города для данных пиццерии – неравномерное распределение

Плохой ShardKey

- Id города для данных пиццерии - неравномерное распределение
- ClientId для корзины - есть не во всех запросах

Хороший ShardKey

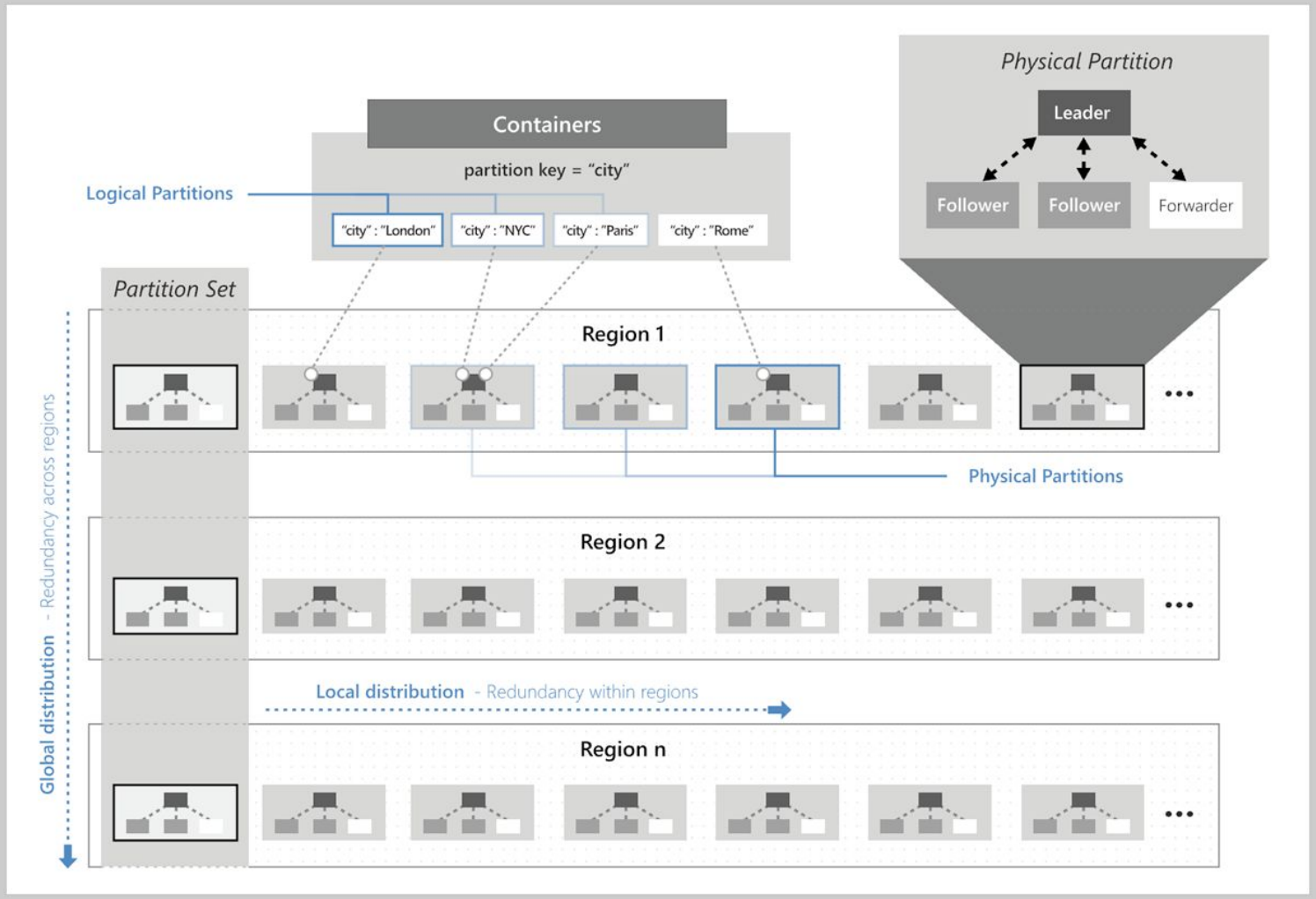
- ShardKey = ClientId для заказов клиента

Хороший ShardKey

- ShardKey = ClientId для заказов клиента
- ShardKey = PizzeriaId для данных пиццерии

Под капотом

- Логическая партиция
- Физическая партиция


Physical Partition
Leader
Follower
Follower
Forwarder
Containers
partition key = "city"
Logical Partitions
"city" : "London"
"city" : "NYC"
"city" : "Paris"
"city" : "Rome"
Partition Set
Region 1
Physical Partitions
Region 2
Local distribution - Redundancy within regions
Region n
Global distribution - Redundancy across regions

Ограничение на шарде

- Max Size - 10 GB
- Max Provisioned throughput-10 000 RU/s

Моделирование

Рекомендации от Microsoft

- Денормализация - не бойтесь дублировать данные
- Помнить о лимите в 2Mb на 1 документ
- Кладите документы в одну коллекцию, если нет обоснованных причин их вынести

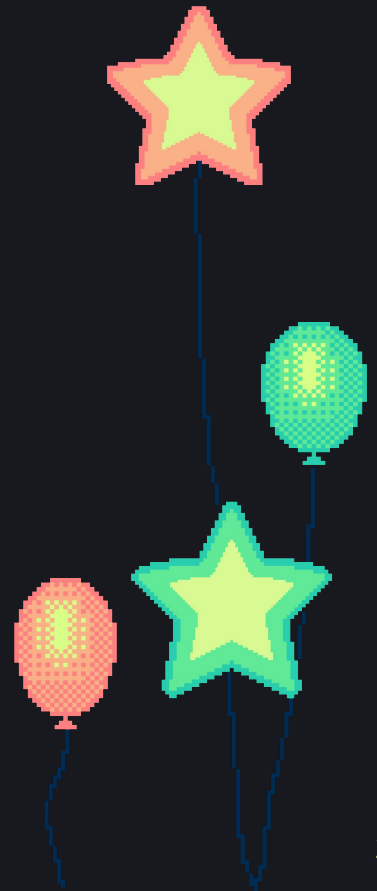
Наши рекомендации

- Разделить документы по категориям - критичный путь и все остальное
- для коллекций, используемых на критическом пути, настройте provisioned throughput
- Не плодить коллекции без выделенного Provisioned throughput

- Базовый документ имеет такую структуру:

```
public class OperationalDataDocument<T>
{
    public ObjectId Id { get; set; }
    31 usages
    public string ShardKey { get; set; }
    69 usages
    public T Value { get; set; }
    27 usages
    public DocumentType Type { get; set; }

    [BsonElement("ttl")]
    7 usages
    public int Ttl { get; set; }
}
```

Mongo API



Создаем коллекцию из кода

```
var createCollectionDocument = new BsonDocument
{
    {"customAction", "CreateCollection"},
    {"collection", collectionName},
    {"shardKey", shardingKey}
};
var shellCommand = new BsonDocumentCommand<BsonDocument>(createCollectionDocument);
var result = await _database.RunCommandAsync(shellCommand, cancellationToken: cancellationToken);

if (result["ok"] == 0)
{
    throw new CreateCosmosCollectionException(result);
}
```

Пишем запрос с помощью MongoApi

```
public Task SavePinCode(string phone, string pin, CancellationToken ct)
{
    var filter = BuildFilter(phone);

    return _collection.UpdateOneAsync(
        filter,
        Builders<PinCodeDocument>.Update
            .Set(_ => _.Value, pin)
            .Set(_ => _.Ttl, _pinCodeExpirationInSeconds),
        new UpdateOptions
        {
            IsUpsert = true
        },
        ct);
}
```

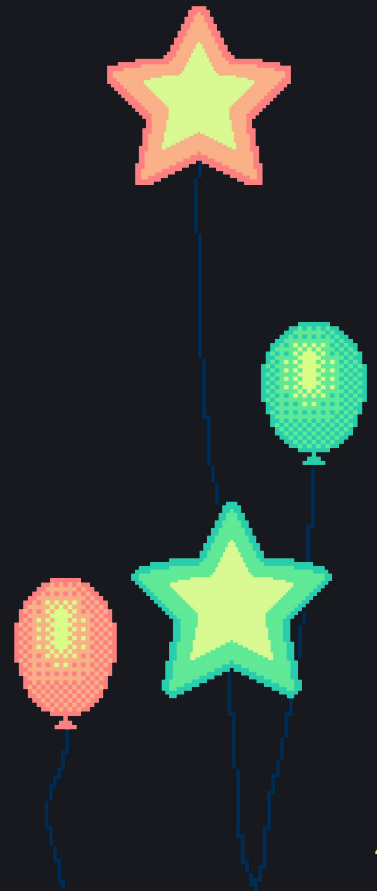
Пишем запрос с помощью MongoApi

```
private static FilterDefinition<PinCodeDocument> BuildFilter(string phone)
{
    var builder = Builders<PinCodeDocument>.Filter;
    var filter = builder.Eq(_ => _.ShardKey, phone) &
                 builder.Eq(_ => _.Type, Contracts.DocumentType.PinCode);
    return filter;
}
```

Пишем запрос с помощью MongoApi

```
public async Task<string> GetPinCode(string phone, CancellationToken ct)
{
    var filter = BuildFilter(phone);

    return await _collection.Find(filter)
                .Project(_ => _.Value)
                .FirstOrDefaultAsync(ct) ??
            string.Empty;
}
```

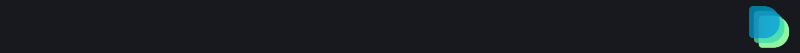
SQL API



Абстрагируемся от sql

```
public QueryDefinition GenerateSql(string partitionKey, string tableName, CosmosQuery query)
{
    _text.Clear();

    Write("SELECT * FROM ");
    Write(_containerId);
    Write(" t WHERE t.");
    Write(_schema.FieldNames.PartitionKey);
    Write("=@key AND t.");
    Write(_schema.FieldNames.Table);
    Write("=@table");

    WriteConditions(query);
    WriteSortings(query);
    WriteLimit(query);

    var queryDefinition = new QueryDefinition(_text.ToString())
        .WithParameter("@key", partitionKey)
        .WithParameter("@table", tableName);

    for (var conditionIndex = 0; conditionIndex < query.Conditions.Count; conditionIndex++)
    {
        var condition = query.Conditions[conditionIndex];
        var value = condition.Range.Value;
        queryDefinition.WithParameter($"@p{conditionIndex}", value);
    }

    return queryDefinition;
}
```

Запрос с помощью обертки



```
public async Task<DomainUnit[]> GetCityUnits(int countryId, Guid cityId, CancellationToken cancellationToken)
{
    return await _unitStore.Query(PartitionKey.From(countryId),
        CosmosQuery
            .Create()
            .Filter(CosmosQueryField.From(nameof(DomainUnit.CityId)), cityId),
        cancellationToken);
}
```

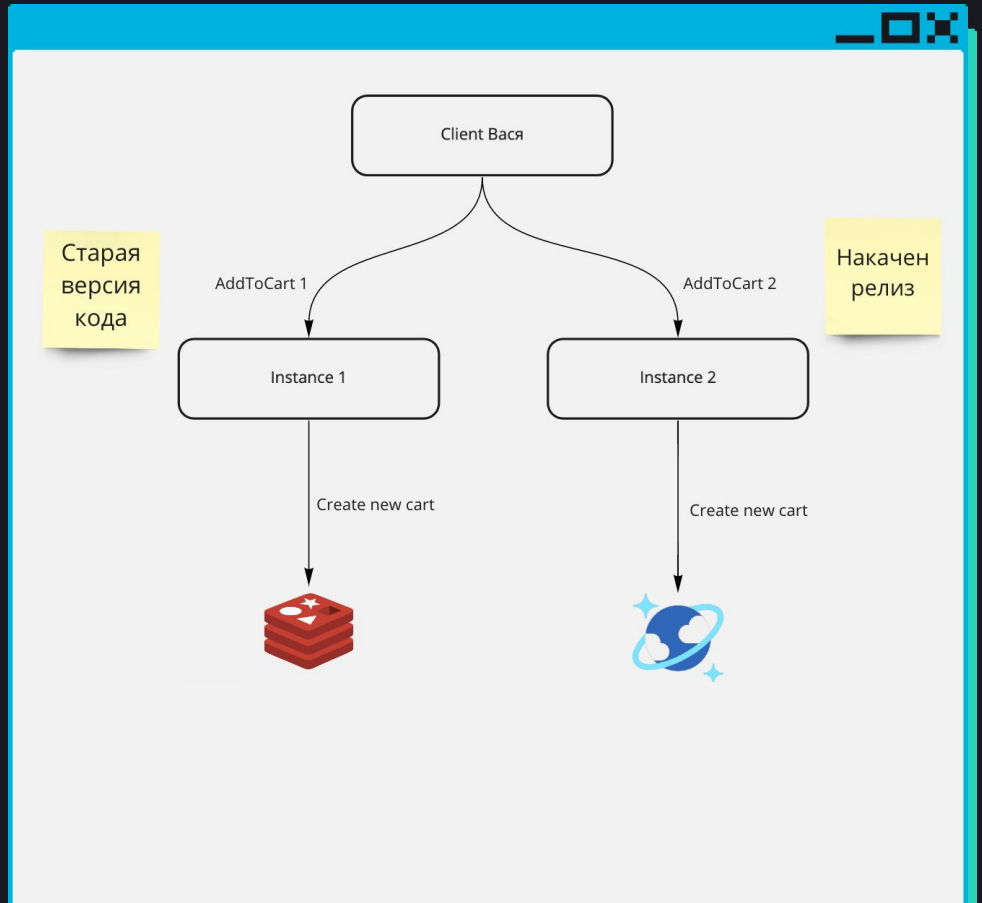
Транзакции

```
public async Task AddOrder(
    DomainOrder order,
    CalculatedOrderDto calculatedOrder,
    DomainRecentOrderList recentOrderList,
    CancellationToken cancellationToken)
{
    if (order == null) throw new ArgumentNullException(nameof(order));

    var transaction = _orderStore
        .CreateTransaction(PartitionKey.From(order.Customer.Id))
        .Create(_orderStore, order.Id, order)
        .Create(_calculatedOrderStore, order.Id, calculatedOrder)
        .Upsert(_recentOrderStore, order.Customer.Id, recentOrderList);

    await transaction.Execute(cancellationToken);
}
```

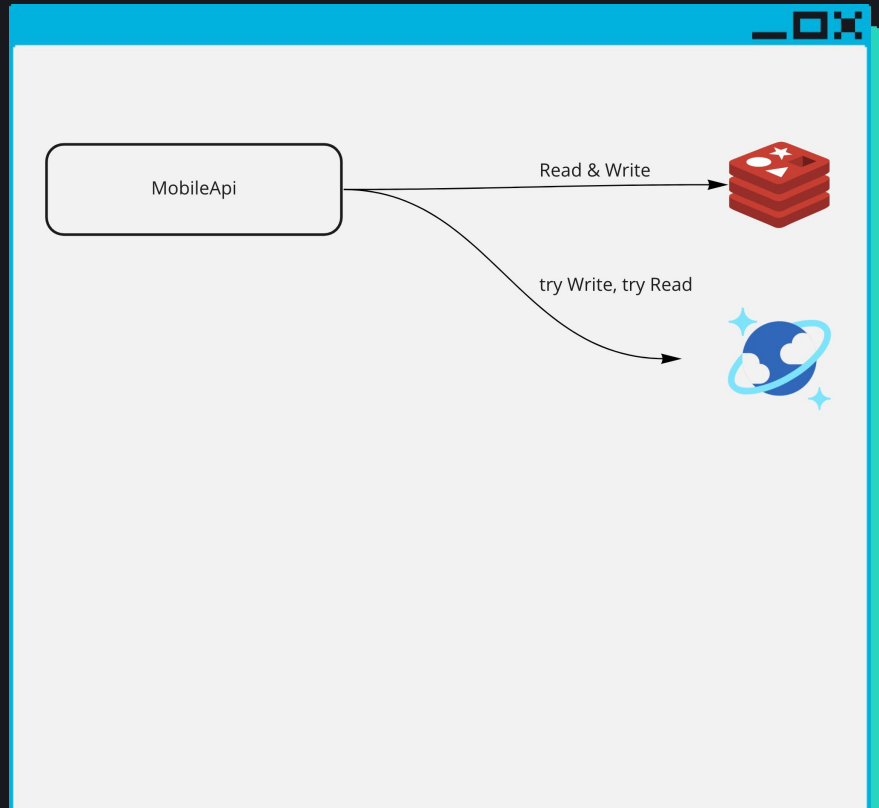
Переезд

Проблема переезда

Потеряем корзины клиентов, которые были в процессе оформления



Продолжаем использовать в Redis. Пробуем писать и читать из CosmosDB

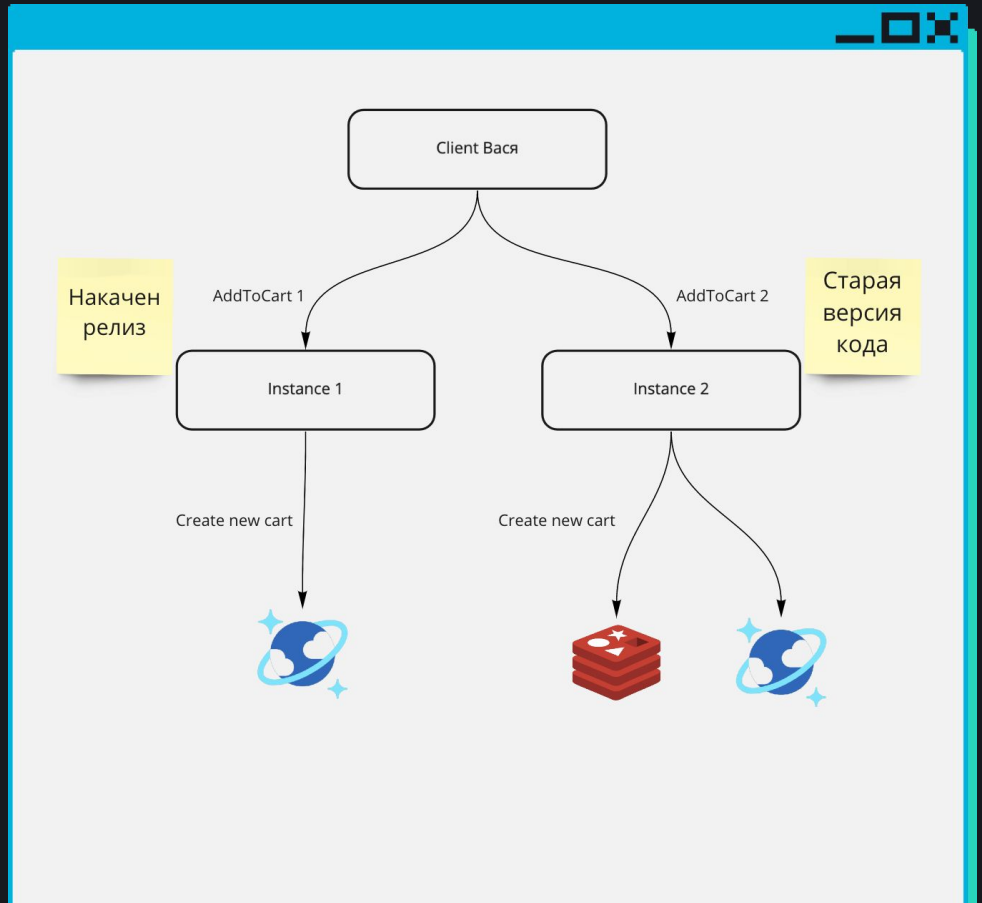



```
public async Task<IOrderWorkflow> GetWorkflowAsync(Guid workflowId)
{
    Contract.RequiresNotEmptyGuid(workflowId, nameof(workflowId));
    var cosmosTask = GetSafelyFromCosmos(workflowId: workflowId);
    var redisTask = _redisStore.GetWorkflowAsync(workflowId: workflowId);
    await Task.WhenAll(cosmosTask, redisTask);

    var workflow = redisTask.Result;
    if (cosmosTask.IsCompletedSuccessfully && cosmosTask.Result != null)
    {
        workflow.ETag = cosmosTask.Result.ETag;
    }

    return redisTask.Result;
}
```

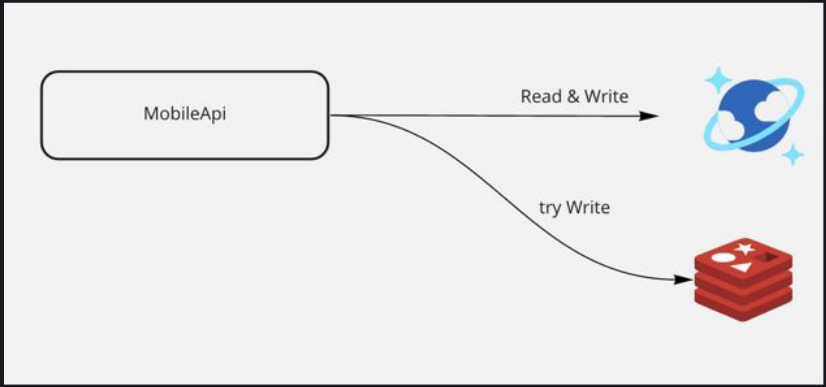
```
public Task SaveWorkflowAsync(IOrderWorkflow workflow)
{
    Contract.RequiresNotNull(workflow, nameof(workflow));
    var cosmosTask = SaveSafelyInCosmos(workflow: workflow);
    var redisTask = _redisStore.SaveWorkflowAsync(workflow: workflow);
    return Task.WhenAll(cosmosTask, redisTask);
}
```

Проблема переезда

Потеряем часть изменений
корзины, которые были
в процессе оформления



Главным становится CosmosDB. Но все еще пишем в Redis




```
public async Task<IOrderWorkflow> GetWorkflowAsync(Guid workflowId)
{
    Contract.RequiresNotEmptyGuid(workflowId, nameof(workflowId));
    return await _cosmosStore.GetWorkflowAsync(workflowId: workflowId);
}
```

```
public async Task SaveWorkflowAsync(IOrderWorkflow workflow)
{
    Contract.RequiresNotNull(workflow, nameof(workflow));
    await _cosmosStore.SaveWorkflowAsync(workflow: workflow);
    await SaveInRedis(workflow: workflow);
}
```

```
private async Task SaveInRedis(IOrderWorkflow workflow)
{
    try
    {
        await Policy
            .TimeoutAsync(TimeSpan.FromMilliseconds(250), TimeoutStrategy.Pessimistic)
            .ExecuteAsync(
                async () =>
                {
                    try
                    {
                        await _redisStore.ForceSave(workflow);
                    }
                    finally
                    {
                        await _redisStore.UnlockWorkflow(workflow);
                    }
                });
```

Pessimistic - в случае таймаута управление возвращается вызвавшему методу, но запрос доделывается в бэкграунде

Change Feed

Что это?

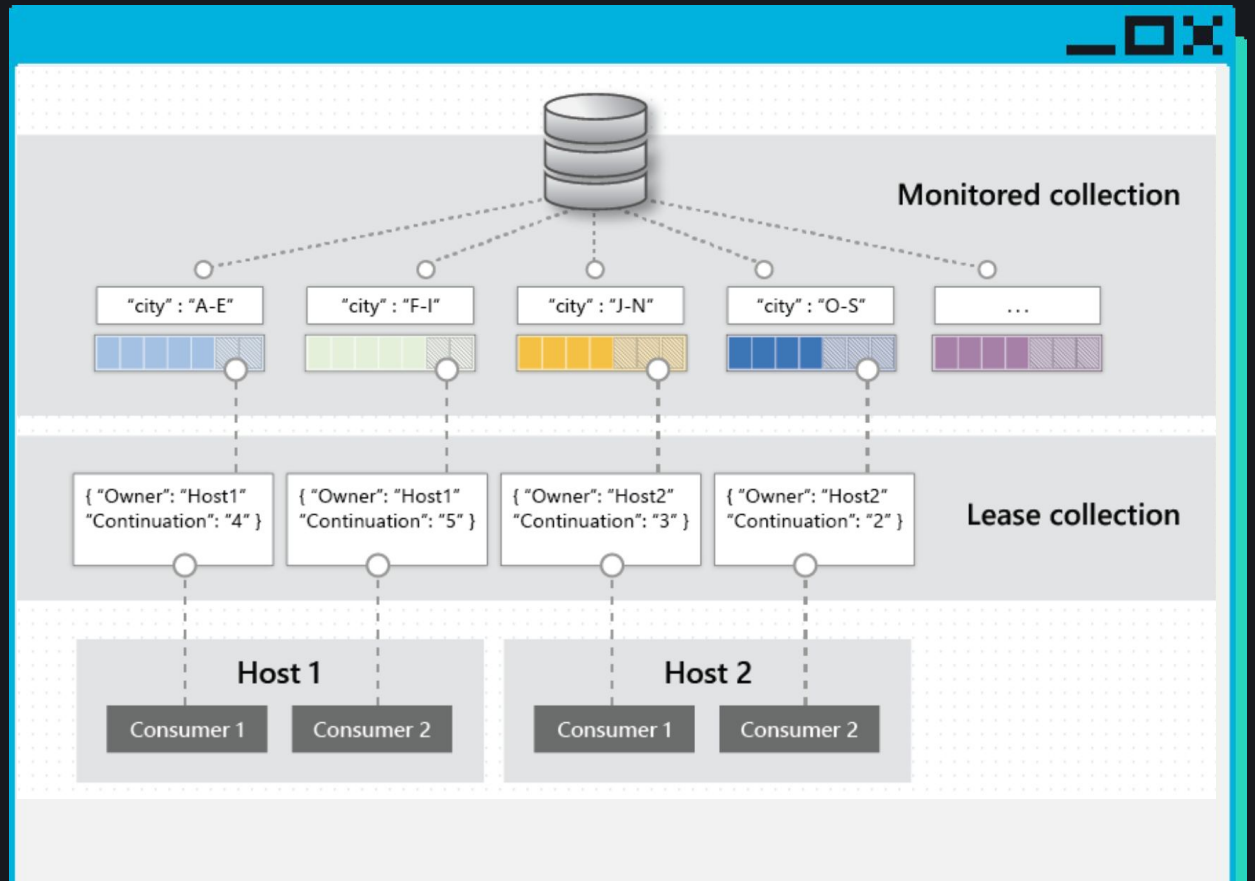
Change Feed в Mongo API

```
protected override async Task ChangeStreamReading(CancellationToken stoppingToken)
{
        using var cursor = _orderCollection.WatchBsonDocument();
        while (await cursor.MoveNextAsync(stoppingToken))
        {
                foreach (var bsonDocument in cursor.Current)
                {
                        var order = bsonDocument.ToChangeStreamResult<Order>().fullDocument;
                        _logger.LogDebug("Entry from recentorder with id {Id} updated", order.Id);
                        _cacheHolder.AddOrUpdate(
                                id: order.Id,
                                addedEntity: order,
                                updateEntity: oldOrder => order);
                }
        }
}
```

Change Feed в Mongo API

```
public static class ChangeStreamExtensions
{
        public static ChangeStreamResult<TDocument> ToChangeStreamResult<TDocument>(this BsonDocument document) =>
                BsonSerializer.Deserialize<ChangeStreamResult<TDocument>>(document);

        public static IChangeStreamCursor<BsonDocument> WatchBsonDocument(
                this IMongoCollection<BsonDocument> collection) =>
                collection.Watch(_pipeline, _options);

        private static ChangeStreamOptions _options = new ChangeStreamOptions
                {FullDocument = ChangeStreamFullDocumentOption.UpdateLookup};

        private static PipelineDefinition<ChangeStreamDocument<BsonDocument>, BsonDocument> _pipeline =
                new EmptyPipelineDefinition<ChangeStreamDocument<BsonDocument>>()
                        .Match(change => change.OperationType == ChangeStreamOperationType.Insert ||
                                          change.OperationType == ChangeStreamOperationType.Update ||
                                          change.OperationType == ChangeStreamOperationType.Replace)
                        .AppendStage<ChangeStreamDocument<BsonDocument>, ChangeStreamDocument<BsonDocument>, BsonDocument>(
                                "{ $project: { '_id': 1, 'fullDocument': 1, 'ns': 1, 'documentKey': 1 }}");
}
```

Как работает для SqlApi?



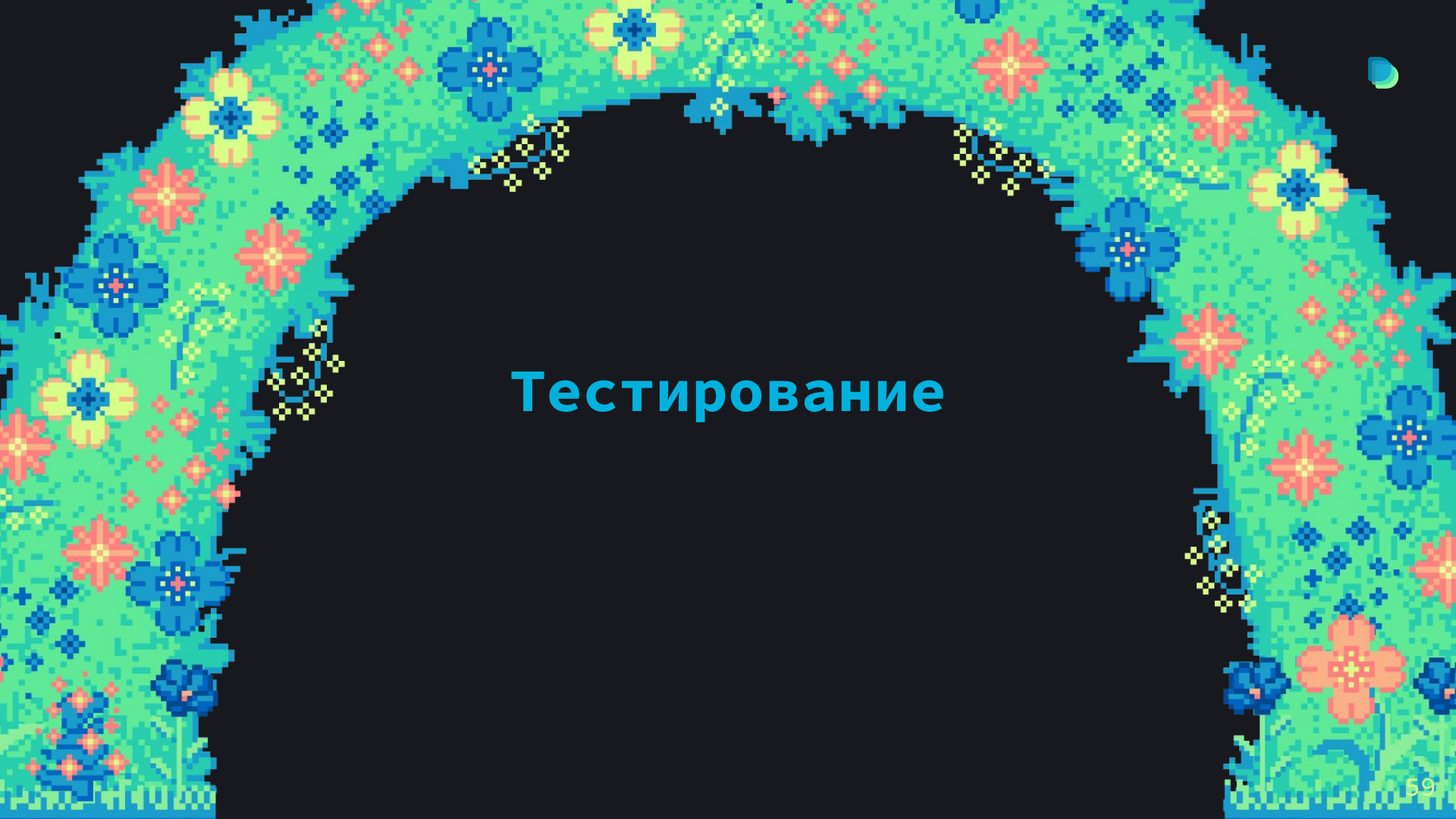
Change Feed в SQL API

```
var leaseContainer = _connection.GetContainer(options.LeaseContainerId);

var builder =
    _container.GetChangeFeedProcessorBuilder(options.ProcessorName, (Container.ChangesHandler<JObject>)HandleChange)
    .WithInstanceName(options.InstanceName)
    .WithLeaseContainer(leaseContainer);

if (options.FromStart)
{
    builder.WithStartTime(DateTime.UnixEpoch);
}

var processor = builder.Build();
await processor.StartAsync();
```

- Change feed processor in Azure Cosmos DB

Change Feed в SQL API

```
async Task HandleChange(IReadOnlyCollection<JObject> documents, CancellationToken cancellationToken)
{
    foreach (var document in documents)
    {
        var id = document.PropertyAsString(_schema.FieldNames.Id);
        var table = document.PropertyAsString(_schema.FieldNames.Table);
        if (table != _schema.TableName)
        {
            continue;
        }

        try
        {
            var entity = _schema.ToEntity(document);
            await changeHandler(entity, cancellationToken);
        }
        catch (Exception exception)
        {
            var item = new CosmosChangeFeedItem(id);
            var resolution = exceptionHandler(item, exception);

            if (resolution == CosmosChangeFeedResolution.Retry)
            {
                throw;
            }
        }
    }
}
```


Тестирование

Тестирование

- Нет образа для докера, но можно использовать не NativeApi (Mongo, SQL, Cassandra, Graph)
- Мы в докере поднимаем монгу и работаем в интеграционных тестах с ней
- Есть отдельные тесты которые проверяют интеграцию с космос

Пример 1

```
[Scenario]

public async Task GivenFeatureEnabledForLocality_ThenFeatureEnabled()
{
    var feature = "feature_1";
    var localityUUId = new UUId();
    var country = CountryCode.RU;

    await RunScenarioAsync(
        _ => _.Given_feature_toggleble_for_localities(feature),
        _ => _.Given_feature_enabled_for_locality(feature, localityUUId),
        _ => _.When_feature_list_requested(country, localityUUId),
        _ => _.Then_ok_received_for(_.FeatureListResponse),
        _ => _.Then_feature_list_contains_feature(feature));
}
```

```
public async Task Given_feature_toggleble_for_localities(string feature)
{
    var filter = BuildTogglebleFeaturesForLocalitiesFilter();
    var addFeatureToList = Builders<TogglebleFeaturesForLocalitiesDocument>
        .Update.AddToSet(_ => _.Value, feature);
    var options = new UpdateOptions() {IsUpsert = true};

    await _countryFeaturesCollection.UpdateOneAsync(filter,
        addFeatureToList,
        options);
}
```

Пример 1


```
if (!_environment.IsLocaltests())
{
    var createCollectionDocument = new BsonDocument
    {
        {"customAction", "CreateCollection"},
        {"collection", collectionName},
        {"shardKey", shardingKey}
    };
    var shellCommand = new BsonDocumentCommand<BsonDocument>(createCollectionDocument);
    var result = await _database.RunCommandAsync(
        shellCommand,
        cancellationToken: cancellationToken);
    if (result["ok"] == 0)
    {
        throw new CreateCosmosCollectionException(result);
    }
}
else
{
    await _database.CreateCollectionAsync(
        collectionName,
        cancellationToken: cancellationToken);
}
```

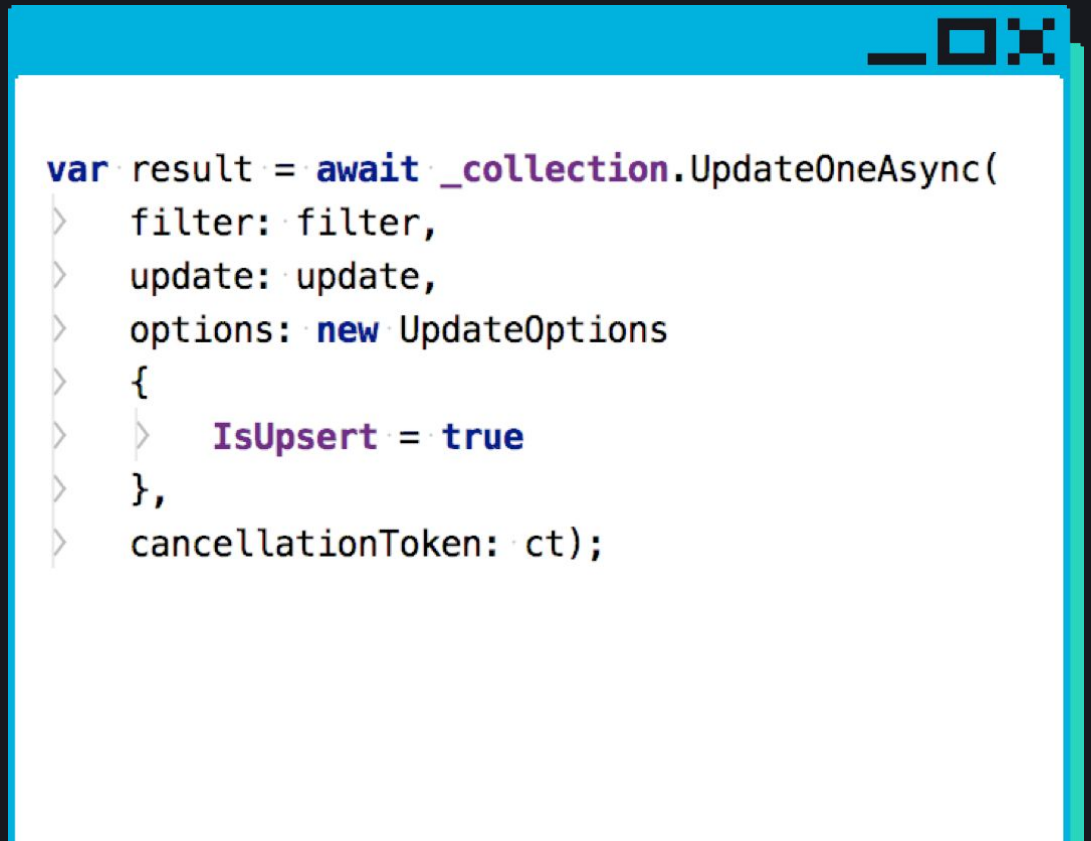
Пример 2

```
public CosmosConnectionFeature()
{
    _configurationProvider = LoadAppSettings();
    _dbCollectionsPrefix = $"{DateTime.Now:ddMM-hhmmss}";
    _settings = _configurationProvider.GetRequiredService<IOptions<Settings>>();
    _cosmosOptions = CreateCosmosOptions(_settings, _dbCollectionsPrefix);
    _mongoDatabase = CreateDatabase(_cosmosOptions);
}

[Scenario]

public async Task CanCreateCollections()
{
    var sut = CreateCollectionCreatorService();
    var delayTask = Task.Delay(TimeSpan.FromSeconds(30));
    var startTask = sut.StartAsync(CancellationToken.None);
    var resultTask = await Task.WhenAny(delayTask, startTask);
    resultTask.Should()
        .NotBeEquivalentTo(delayTask);

    var cursor = await _mongoDatabase.ListCollectionNamesAsync();
    var collections = await cursor.ToListAsync();
    collections.Count(_ => _.Contains(_dbCollectionsPrefix))
        .Should()
        .Be(8);
}
```

```
private CollectionCreationService CreateCollectionCreatorService()
{
    var dbFactory = new DatabaseFactory(_cosmosOptions);
    var collectionFactory = new CollectionFactory(dbFactory, _cosmosOptions);
    return new CollectionCreationService(
        dbFactory,
        collectionFactory,
        _settings,
        NullLogger<CollectionCreationService>.Instance);
}
```

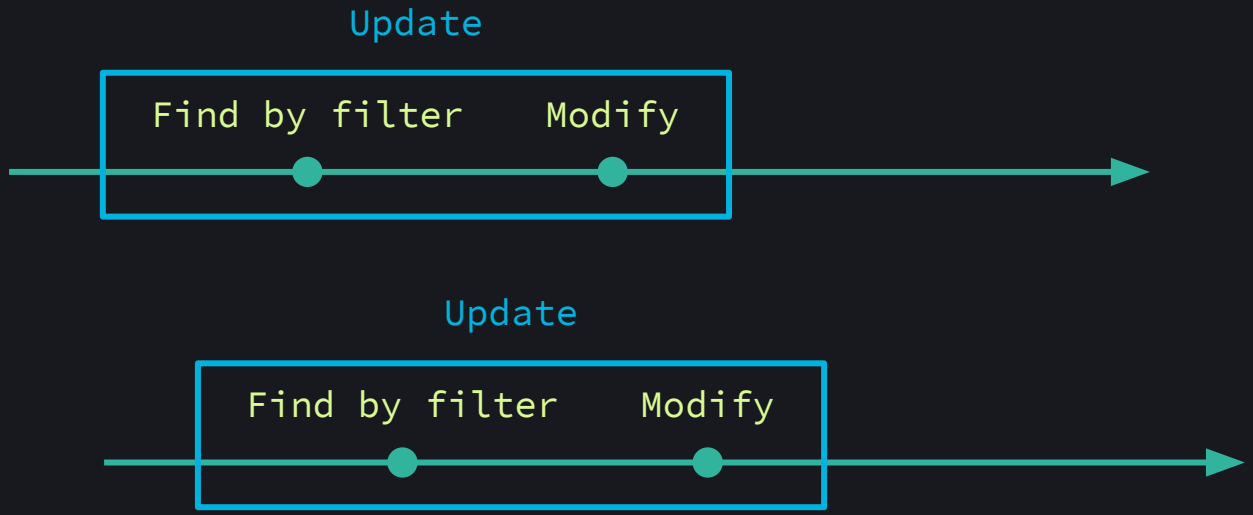
Особенности

Вам не понравится UpdateOne



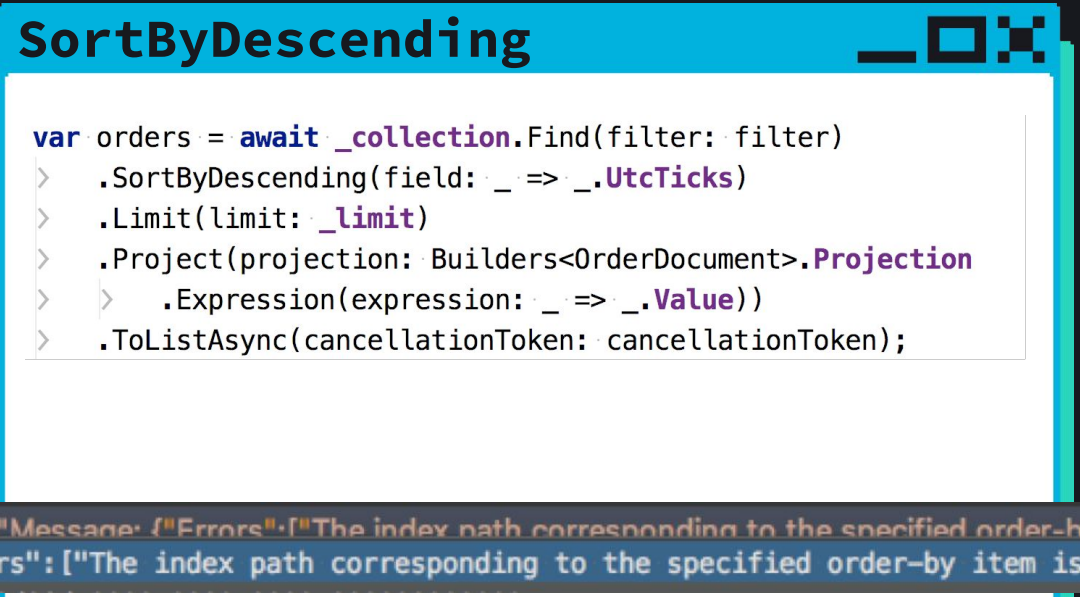
```
var result = await _collection.UpdateOneAsync(
    filter: filter,
    update: update,
    options: new UpdateOptions
    {
        IsUpsert = true
    },
    cancellationToken: ct);
```

Вам не понравится UpdateOne

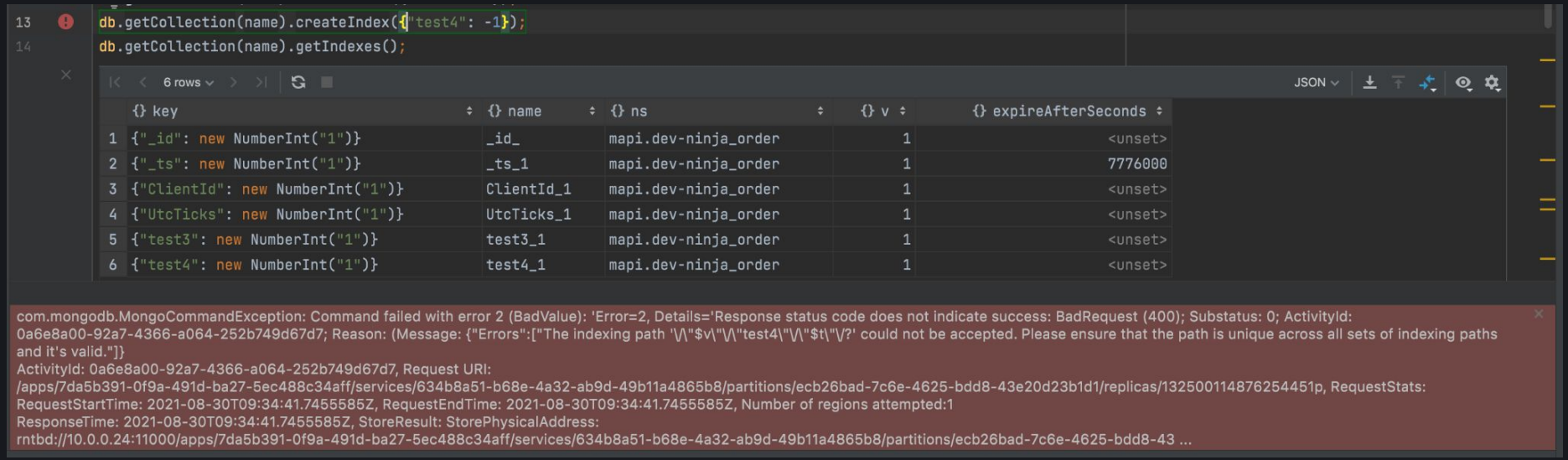


UpdateOne

В монго-апи `UpdateOne` не гарантирует транзакционности исполнения предиката (where) и последующего изменения данных. Это создаёт возможность перетирания более свежих версий документа устаревшими его версиями. `FindOneAndModify` работает правильно

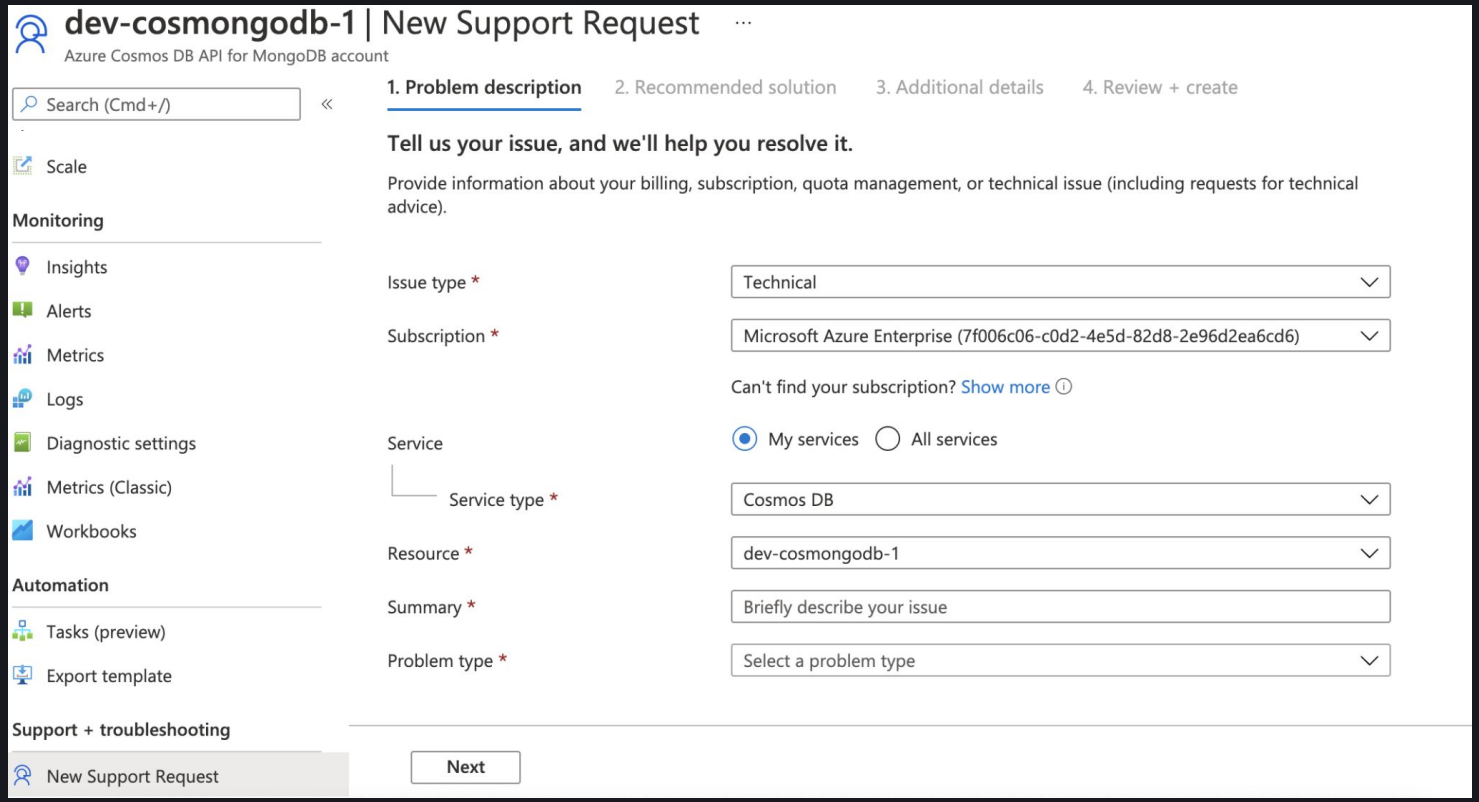

SortByDescending
var orders = await _collection.Find(filter: filter)
.SortByDescending(field: _ => _.UtcTicks)
.Limit(limit: _limit)
.Project(projection: Builders<OrderDocument>.Projection
.Expression(expression: _ => _.Value))
.ToListAsync(cancellationToken: cancellationToken);
Message: {"Errors":["The index path corresponding to the specified order-by item is excluded."]}

Пересоздание индексов для descending

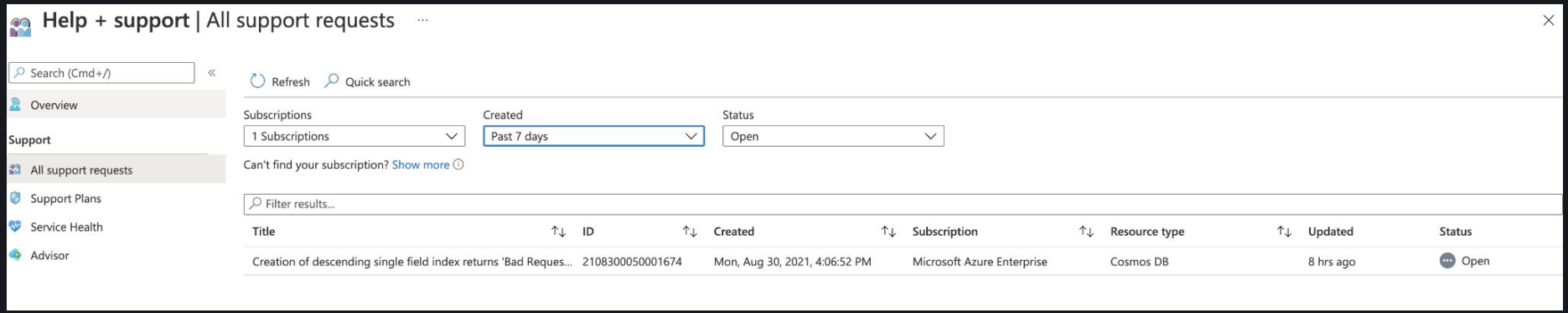


* Для ascending работает нормально

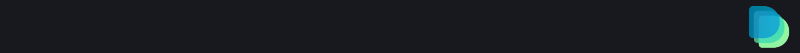
Issue не на github, а в portal.azure.com



Issue не на github, а в portal.azure.com

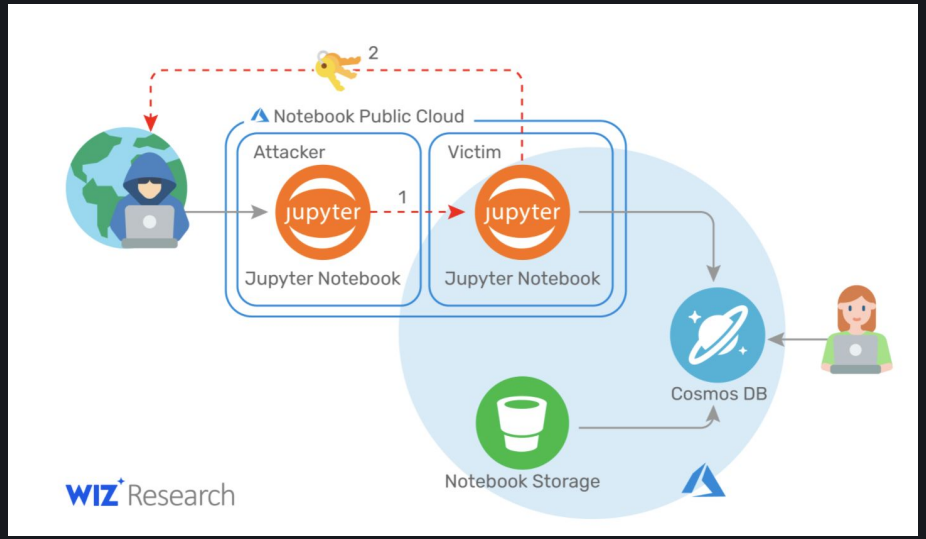


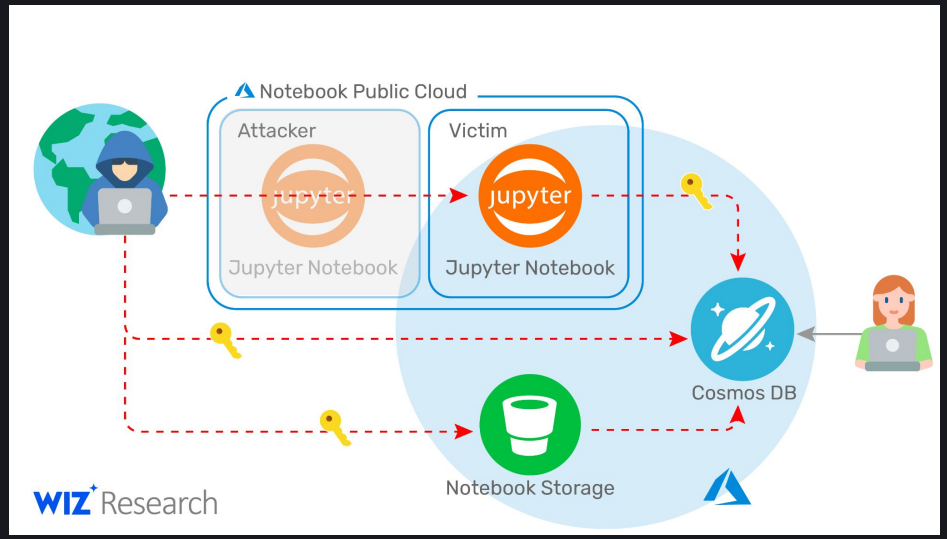
Безопасность не гарантирована



В 2019 году из-за новой фичи по визуализации данных «Jupyter Notebook» появилась уязвимость. Она давала полный доступ к любой CosmosDB.
С февраля 2021 года «Jupyter Notebook» включена по умолчанию. В августе этого года компания «Wiz» обнаружила эту уязвимость. Майкрософт устранили проблему в течении 48 часов. + Заявил что не обнаружил попыток воспользоваться этой уязвимость

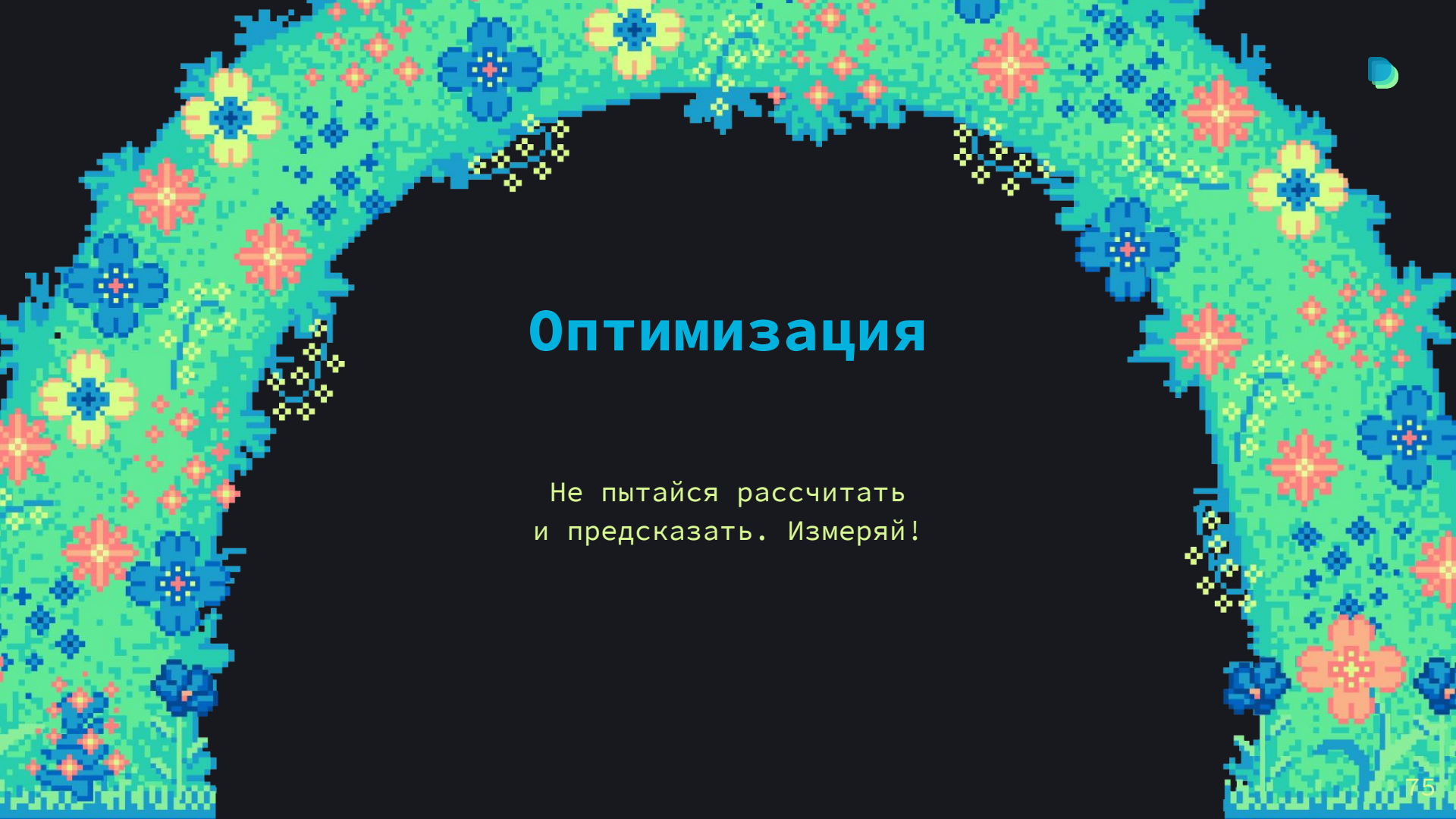
Уязвимость через повышение привилегий





Уровни консистентности с Mongo API

MongoDB и, как следствие, MongoAPI не поддерживают уровень консистентности Session. Соответственно, при его использовании с MongoAPI клиент по факту получает уровень консистентности Consistent prefix


Оптимизация
Не пытайся рассчитать
и предсказать. Измеряй!

Код

```
var requestStatistics :BsonDocument = _database.RunCommand(
    command: new JsonCommand<BsonDocument>(json: "{getLastRequestStatistics: 1}"));
```

Ответ

```
{
  "_t": "GetRequestSatisticsRespose",
  "ok": 1,
  "CommandName": "insert",
  "RequestCharge": 339.62,
  "RequestDurationInMilliSeconds": NumberLong(53)
}
```

Ответ


{
"_t": "GetRequestSatisticsRespose",
"ok": 1,
"CommandName": "insert",
"RequestCharge": 339.62,
"RequestDurationInMilliSeconds": NumberLong(53)
}

Как мы решили проблему?

```
var collection = _database.GetCollection<T>(name: collectionName);
collection.Indexes.DropAll(cancellationToken: CancellationToken.None);
collection.Indexes.CreateMany(models: indexes);
```

Стало

```
{
  "_t": "GetRequestSatisticsRespose",
  "ok": 1,
  "CommandName": "insert",
  "RequestCharge": 26.48,
  "RequestDurationInMilliSeconds": NumberLong(15)
}
```

Как мы теперь анализируем запросы

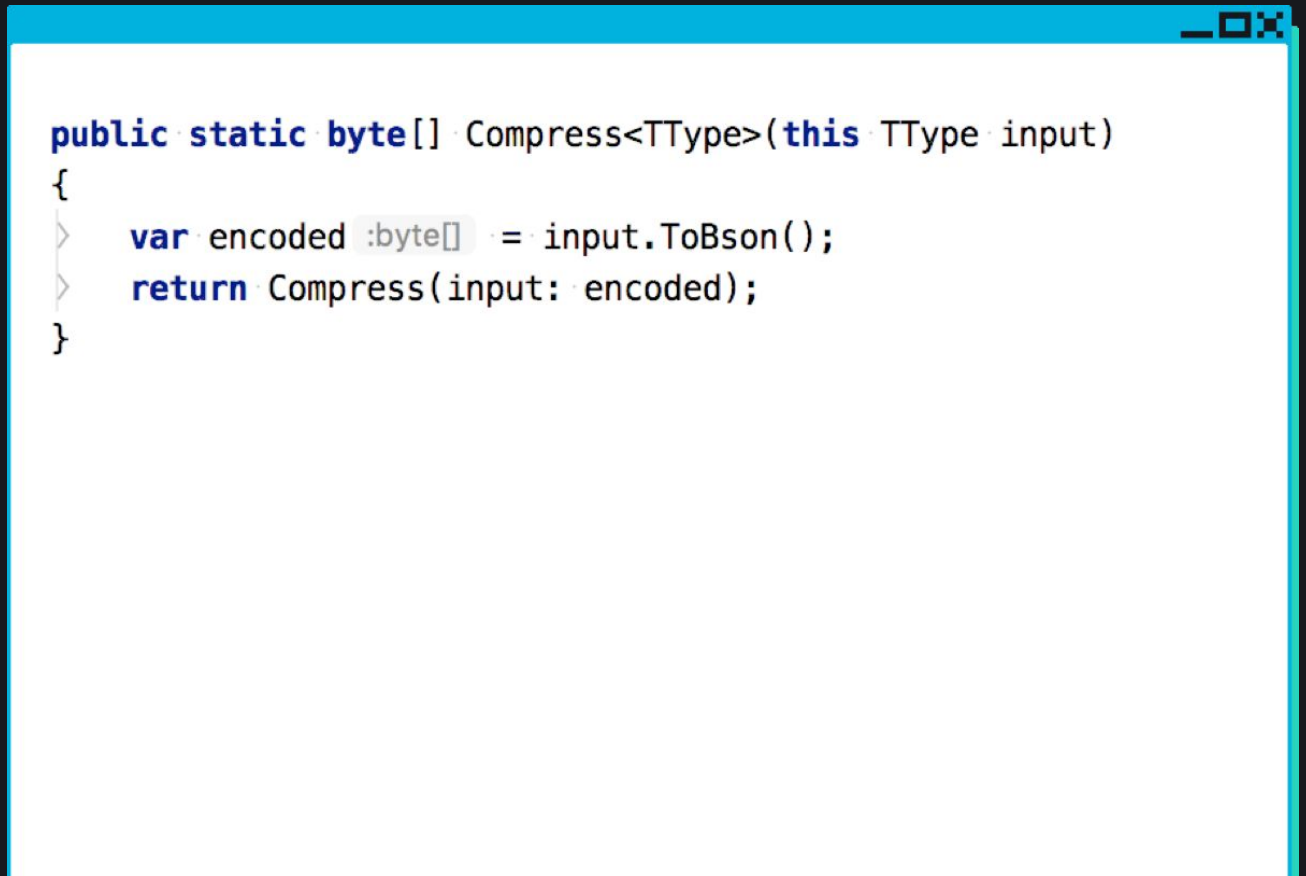
```
public class AddQuery : BaseQuery<ComboTemplateInfo>
{
    private readonly IComboTemplateRepository _repository;


    public AddQuery(
        IComboTemplateRepository repository,
        CollectionFactory factory)
    {
        _repository = repository;
        CollectionParameters = new CollectionParameters(
            factory.Menu,
            nameof(ComboTemplateDocument.ShardKey));
        IndexesFactory = new List<Action<IndexCreator>>
        {
            creator => creator.CreateIndexes(
                CollectionParameters.CollectionName,
                Indexes.ComboTemplate)
        }.AsReadOnly();
    }


    public override Task Setup() =>
        Task.CompletedTask;


    public override Task Execute(int index) =>
        _repository.Save(Rand(), CancellationToken.None);

    2+1 usages
    public override CollectionParameters CollectionParameters { get; }
    1+1 usages
    public override ReadOnlyCollection<Action<IndexCreator>> IndexesFactory { get; }
}
```

Как мы теперь анализируем запросы

```
public async Task Analyze(
    IEnumerable<IQuery> queries,
    int runningCount = 200)
{
    foreach (var query in queries)
    {
        var indexesFactory = query.IndexesFactory;
        for (var i = 0; i < indexesFactory.Count; i++)
        {
            Console.WriteLine($"New Index model {i} for {query.GetType().Name}");

            await _collectionCreator.RecreateCollection(
                query.CollectionParameters,
                indexesFactory[i]);

            await query.Setup();
            for (var j = 0; j < runningCount; j++)
            {
                await Analyze(query, j);
            }
        }
    }
}
```

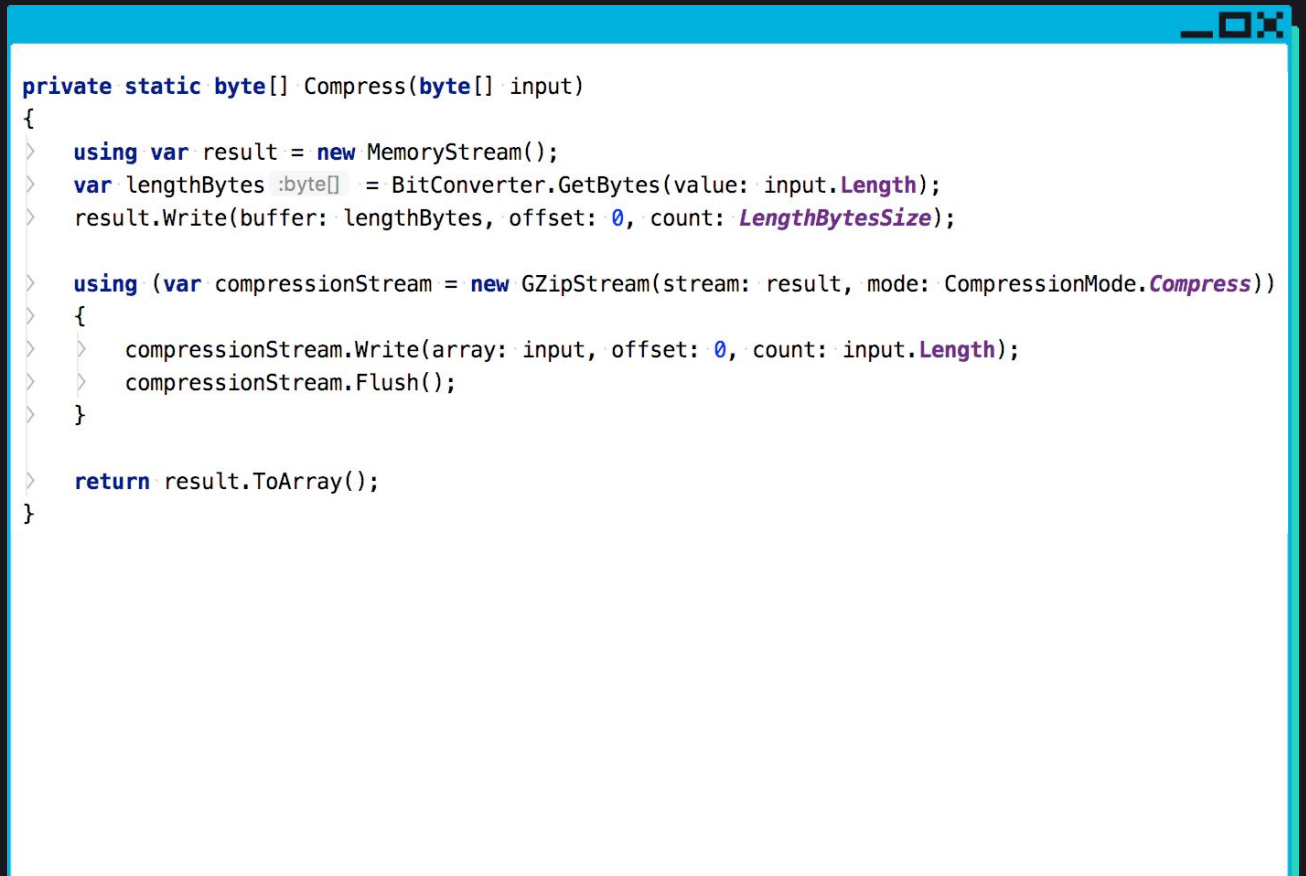
```
private async Task Analyze(IQuery query, int i)
{
    try
    {
        await query.Execute(i);
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
        return;
    }

    var ru = _database.RunCommand(
        new JsonCommand<BsonDocument>("{getLastRequestStatistics: 1}"));
    Console.WriteLine(ru.ToJson() + Environment.NewLine);
}
```

Сжатие больших документов



```
public static byte[] Compress<TType>(this TType input)
{
    var encoded :byte[] = input.ToBson();
    return Compress(input: encoded);
}
```

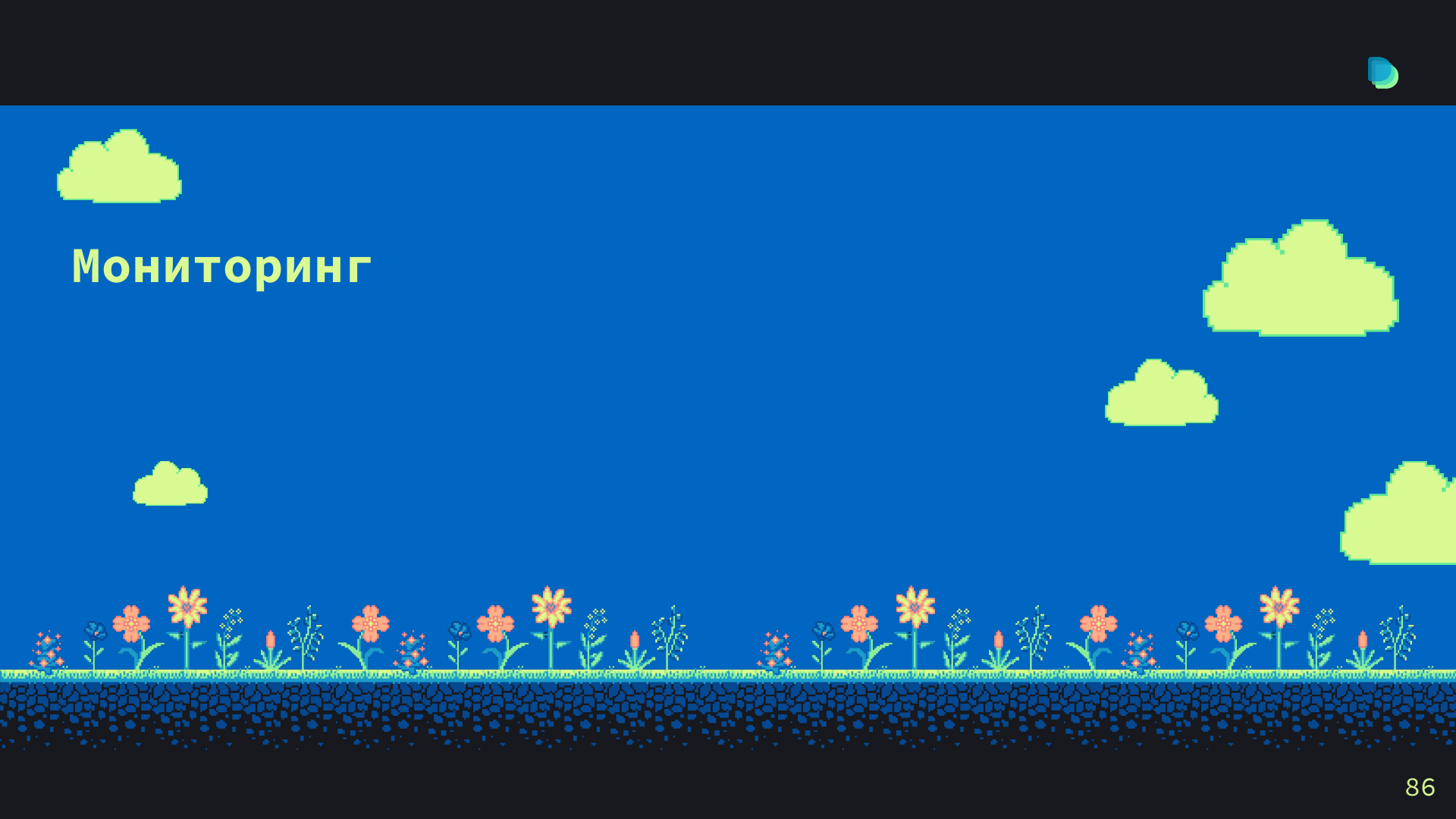
Сжатие больших документов



```
private static byte[] Compress(byte[] input)
{
    using var result = new MemoryStream();
    var lengthBytes :byte[] = BitConverter.GetBytes(value: input.Length);
    result.Write(buffer: lengthBytes, offset: 0, count: LengthBytesSize);

    using (var compressionStream = new GZipStream(stream: result, mode: CompressionMode.Compress))
    {
        compressionStream.Write(array: input, offset: 0, count: input.Length);
        compressionStream.Flush();
    }

    return result.ToArray();
}
```

Советы по оптимизации

- Если делаете запрос с фильтрами, настраивайте индексы
- Не используйте в индексах сильно вложенные поля
- Сжимайте очень большие документы


Мониторинг

Azure Cosmos DB by yesoreyeram

DASHBOARD

https://github.com/yesoreyeram/grafana-azure-dashboards

Last updated: 7 months ago

Downloads: 100

Reviews: 0

Add your review!

Overview | Revisions | Reviews

DocumentDB / Cosmos DB

Get this dashboard:

10539

Copy ID to Clipboard

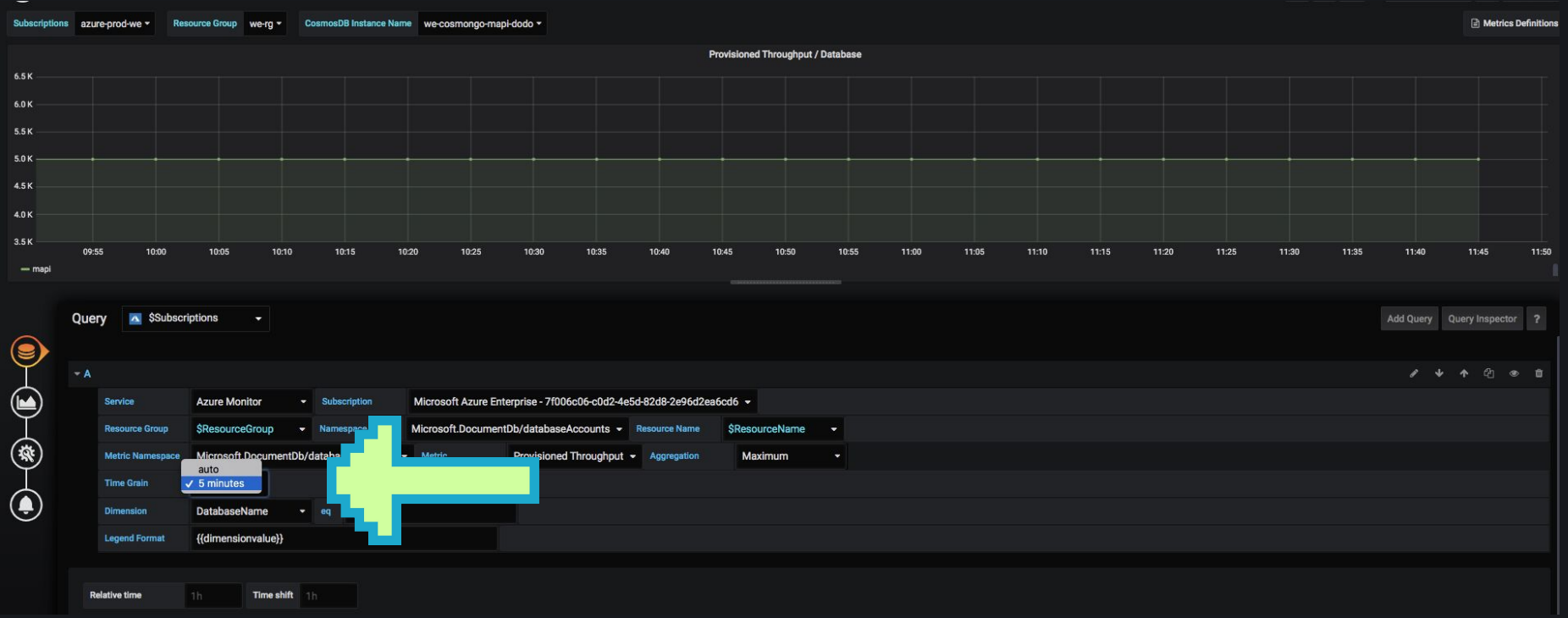
Download JSON

How do I import this dashboard?

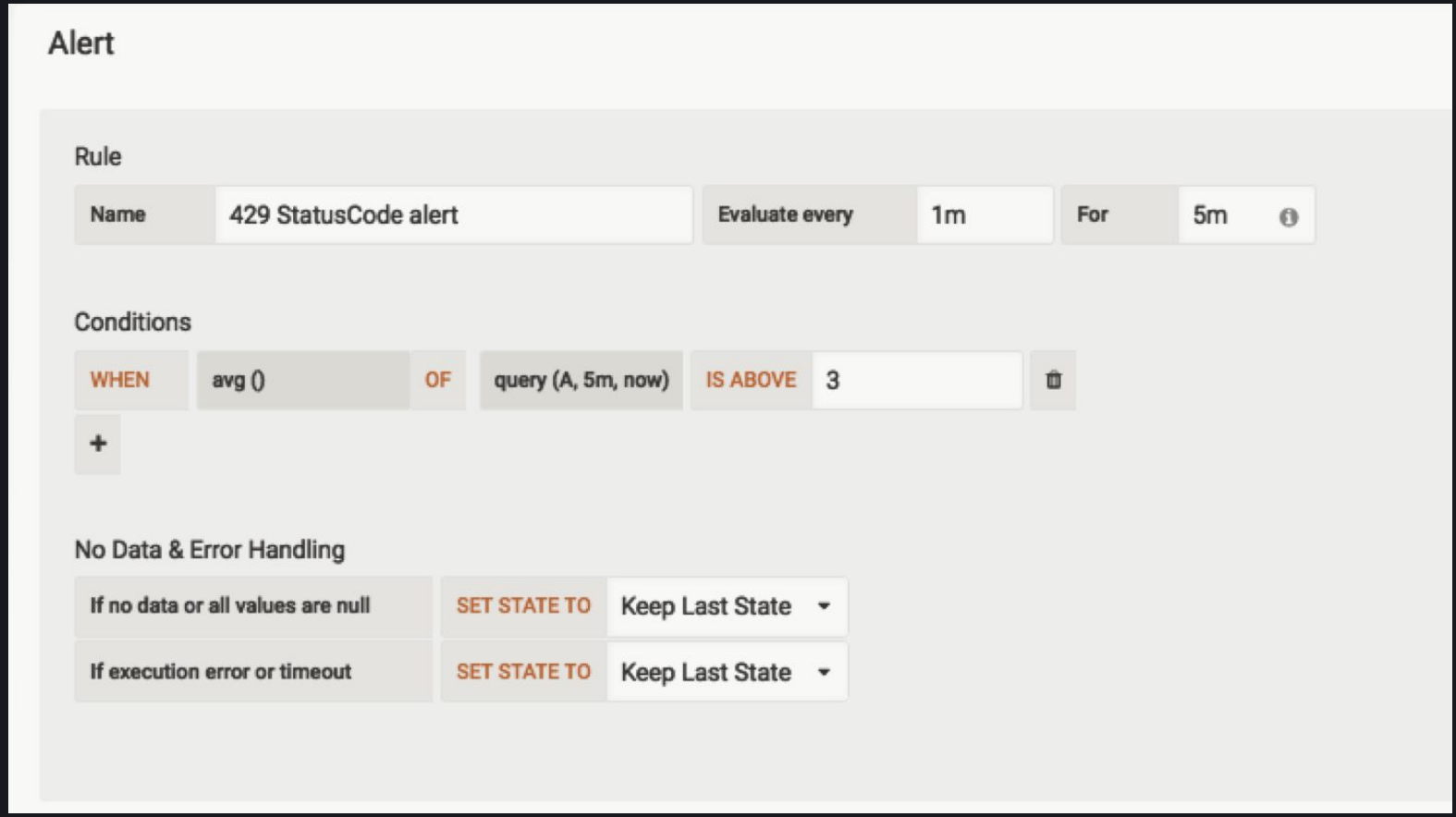
Dependencies:

- GRAFANA 6.1.6
- AZURE MONITOR 0.3.0
- GRAPH
- SINGLESTAT


Azure Cosmos DB 2
Subscriptions
azure-prod-we
Resource Group
we-rg
CosmosDB Instance Name
we-cosmongo-mapi-dodo
Metrics Definitions
Service Availability / Hour
100%
Document Count
2359901
Available Storage
406 GB
Total storage quota
107 GB
Data Usage
8.695 GB
Index Usage
615 MB
Total Requests / min
4643
Total Request Units
28781
Provisoned Throughput / 5 min
0
Number of Request / Status
200
201
409
449
412
Total Request Units
Provisioned Throughput
Total Requests / OperationType
query
Replace
create
read
Total Requests / Region
West Europe
Total Requests / Database
mapi
Provisioned Throughput / Collection
Provisioned Throughput / Database
Availble storage / Collection
Data usage / Collection


Subscriptions
azure-prod-we
Resource Group
we-rg
CosmosDB Instance Name
we-cosmongo-mapi-dodo
Metrics Definitions
Provisioned Throughput / Database
6.5 K
6.0 K
5.5 K
5.0 K
4.5 K
4.0 K
3.5 K
09:55
10:00
10:05
10:10
10:15
10:20
10:25
10:30
10:35
10:40
10:45
10:50
10:55
11:00
11:05
11:10
11:15
11:20
11:25
11:30
11:35
11:40
11:45
11:50
mapi
Query
$Subscriptions
Add Query
Query Inspector
A
Service
Azure Monitor
Subscription
Microsoft Azure Enterprise - 7f006c06-c0d2-4e5d-82d8-2e96d2ea6cd6
Resource Group
$ResourceGroup
Namespace
Microsoft.DocumentDb/databaseAccounts
Resource Name
$ResourceName
Metric Namespace
Metric
Provisioned Throughput
Aggregation
Maximum
auto
5 minutes
Time Grain
Dimension
DatabaseName
eq
Legend Format
{{dimensionvalue}}
Relative time
1h
Time shift
1h


Azure Cosmos DB
Last 2 hours
15m
Subscriptions
azure-prod-we
Resource Group
we-rg
CosmosDB Instance Name
we-cosmongo-mapi-dodo
Metrics Definitions
Service Availability / Hour
100%
Document Count
6058608
Available Storage
2.45 TB
Total storage quota
591 GB
Data Usage
17.3 GB
Index Usage
2.53 GB
Total Requests / min
67480
Total Request Units
290100
Provisoned Throughput / 5 min
5400
Number of Request / Status
70 K
60 K
50 K
40 K
16:00
16:30
17:00
17:30
200
201
204
409
449
412
Total Request Units
350 K
300 K
250 K
200 K
Total Request Units
Provisioned Throughput
5.50 K
5 K
4.50 K
4 K
Provisioned Throughput
Total Requests / OperationType
100 K
75 K
50 K
25 K
0
Query
Replace
Create
DELETE
read
Total Requests / Region
West Europe
Total Requests / Database
mapi
Provisioned Throughput / Collection
6 K
4 K
2 K
0
16:00
16:10
16:20
16:30
16:40
16:50
17:00
17:10
17:20
17:30
17:40
17:50
_Empty
CriticalPath
admin
Provisioned Throughput / Database
mapi
Availble storage / Collection
300 GB
200 GB
Data usage / Collection
60 GB
40 GB


Alert
Rule
Name
429 StatusCode alert
Evaluate every
1m
For
5m
Conditions
WHEN
avg ()
OF
query (A, 5m, now)
IS ABOVE
3
+
No Data & Error Handling
If no data or all values are null
SET STATE TO
Keep Last State
If execution error or timeout
SET STATE TO
Keep Last State

Трейсинг

```
internal static class TracingExtension
{
    1 usage  Andrew Paramonov
    internal static IOpenTracingBuilder AddMongoTracing(this IOpenTracingBuilder builder)
    {
        var applicationLifetime = builder.Services.BuildServiceProvider()
            .GetRequiredService<IHostApplicationLifetime>();
        var subscription = DiagnosticListener.AllListeners.Subscribe(new ListenersObserver(GlobalTracer.Instance.MongoTracing()));
        applicationLifetime.ApplicationStopping.Register(() => subscription.Dispose());
        return builder;
    }
}
```

Выводы

У космоса много плюшек:

- SLA 99,999
- Легко оптимизировать стоимость
- Не нужно быть DevOps-ом чтобы адмить
- Круто с точки зрения оптимизации (RU)
- Change Stream

Есть свои сложности

- Все коллекции с ShardKey
- Баго-фичи

ВОПРОСЫ?