Az COMMUNITY

Conference 2022

Asia's Largest Azure Community Conference

#AzConfDev

SAJEETHARAN SINNATHURAI

Senior Program Manager, Azure Cosmos DB

kokkisajee

#AzConfDev

Unleash the power of Azure Cosmos DB

Best practices, Tips and Tricks

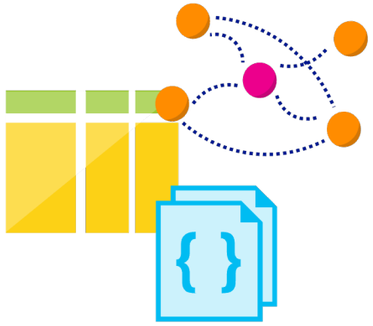
#AzConfDev

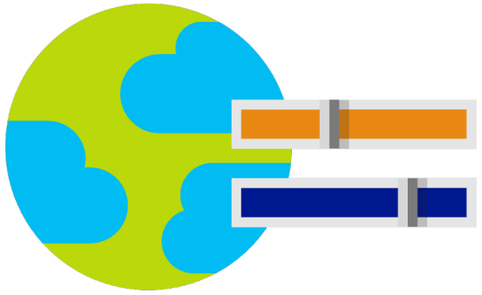
Next 45 minutes

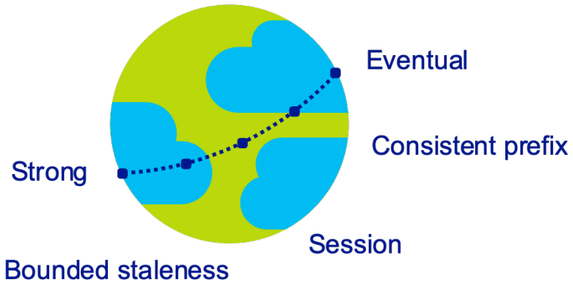
- Why Cosmos DB for developers?
- How can your code talk to Cosmos DB?
- Best Practices / Common challenges with SDKs
- Tools for Azure Cosmos DB
- Q&A

Why Azure Cosmos DB for developers?

- It’s PaaS – No “Infrastucture to manage”
- Multiple Supported OSS APIs, multiple supported languages
- Schema Agnostic
- Global distribution, multi-master, scale, consistency , indexing control
- 99.999% HA with 2+ regions
- Guaranteed throughput / granula SLAs/ low latency
- Data security including BYOK
- Low impedence to storing and querying JSON in Core (NOSQL) API
- Stored procs and UDFs (Written in Javascript)
- Graphql Support to build JAMstack apps







How to talk to Cosmos DB with code?

How can we use Cosmos DB in our code/apps?

- APIS
- Languages
- SDKs
- ORMs
- Drivers

Azure Cosmos DB REST API

Cosmos DB provides REST APIs to manage and query Cosmos DB resources.

Careful – terminology overload: "**API**"

- **Account** APIs = compatibility, storage format; SQL, MongoDB, Cassandra, etc.
- **REST** APIs = programmability from anything that speaks HTTP/REST

Two separate REST APIs:

1. "REST API" for working with resources and data in a Cosmos DB account
2. "Resource Provider API" for working with accounts and keys

Azure Cosmos DB *Data Plane* REST API

- Reference: https://docs.microsoft.com/rest/api/cosmos-db/
- Work with resources ***in*** a Cosmos DB account
  - Databases, Containers, Documents, Sprocs, UDFs, Documents, Queries
- Can NOT create/manage Cosmos DB accounts or keys with this API
- NOSQL(**formerly named SQL API**) only! (Against other APIs = 404/failure)
- Use the account endpoint: https://youraccount.documents.azure.com

- Built on the .NET SDK v2: https://github.com/Azure/azure-cosmos-dotnet-v2
  - Do you need to care? No – the API itself is pure HTTP/REST
    - I.e. the underlying technology doesn't matter to the API consumer

Azure Cosmos DB *Management* REST API

- Reference: https://docs.microsoft.com/rest/api/cosmos-db-resource-provider/
- Work with Azure Cosmos DB accounts and keys
  - For other tasks, use the REST API (previous slide)
- *All* Cosmos DB APIs supported
- Endpoint: the Azure management API itself
  - https://**management.azure.com**/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DocumentDB/databaseAccounts/{accountName}/...?api-version=2021-10-15
- Other layers (SDKs, Azure CLI, Azure Portal etc.) use this RP API

Azure Cosmos DB REST API – should use them?

Yes, if…

- Working in a language without an SDK or other Cosmos DB library/package
- No other library, driver, or ORM available
- Need to avoid using SDK/library/package
- Want to re-implement SDK capabilities
- Want to experiment/learn

No, because…

- Cosmos DB creates SDKs – significant dev/test resources
- SDKs incorporate best practices
- SDKs include cross-cutting concerns like error handling, logging
- SDKs handle retry, parallelization, network traffic optimization, bulk, batch, more
- SDKs heavily tested, optimized, supported
- SDKs are OSS on github – can inspect, file issues, even contribute!

Summary: use SDKs when you can, REST API if you must, or to experiment.

REST API DEMO

Azure Cosmos DB Language Support

.NET SDKs (Framework, Core, Standard)

- V2: still supported but do not use for new work
- V3: current release – notably, includes bulk support which was separate in V2
- V4: preview

Java SDKs

- V2: sync and async versions – still supported but do not use for new work
- V4: current version

Node.js SDK (3.17.1)

Python SDK (4.3.0)

GO SDK (2.2.0)

Azure Cosmos DB ORM Support

ORMs used with RDBMS to solve “object-relational impedance”.

NoSQL/Document databases: easy to store serialized entity → less need for ORMs.

In C#/.NET: object → JSON in one line of code:

```
string myDocument = JsonConvert.SerializeObject(myObject);
```

Then write document to Cosmos DB. No mapping to relational schemas!

(But we may still need to think about embedding vs. referencing.)

Major ORMs are traditionally relational-centric. NoSQL database support is limited.

Azure Cosmos DB Language Support

Mongo API

-Official drivers provided by MongoDB

-13 officially supported libraries including .NET/C#, Java, Go, Node.js, Python, Rust

- CData drivers – specifically for Cosmos DB accounts with MongoDB API

- Mongoose – Node.js object-database mapper (see how-to guide)

The idea: Cosmos DB MongoDB API account can be substituted for a MongoDB instance – change nothing in your code

- Not all Mongo commands are supported; review and test early!!

Cassandra API

- Cassandra drivers from Datastax and gocql
  - Datastax is commercial Cassandra vendor; has Azure MP offerings
  - Java, .NET/C#, Node.js, Python, C++, PHP
  - Gocql is OSS: Cassandra client for Go
- Spring Data
  - Spring is a major Java framework
- Carefully review supported Cassandra data types, CQL functions, and limits!
- Also review differences

Gremlin API

Gremlin API based on Apache Tinkerpop graph database standard.

Uses Gremlin query language to interact with data.

Use Gremlin API for compatibility, or to implement a graph-capable application.

Use a compatible client library from Apache or a third party. (we recommend drivers supported by Apache Tinkerpop.) Can also use Spring Data.

Careful! Review Cosmos DB Gremlin compatibility and limits!

Table API

Table API is for apps written to use Azure Table Storage.

SDKs for .NET/C#, Java, Python, and Node.js.

Recommended to use the Cosmos DB-specific SDKs rather than Azure Storage SDKs.

Carefully review the differences between Azure Table Storage and Cosmos DB Table API, as well as the benefits of Cosmos DB Table API.

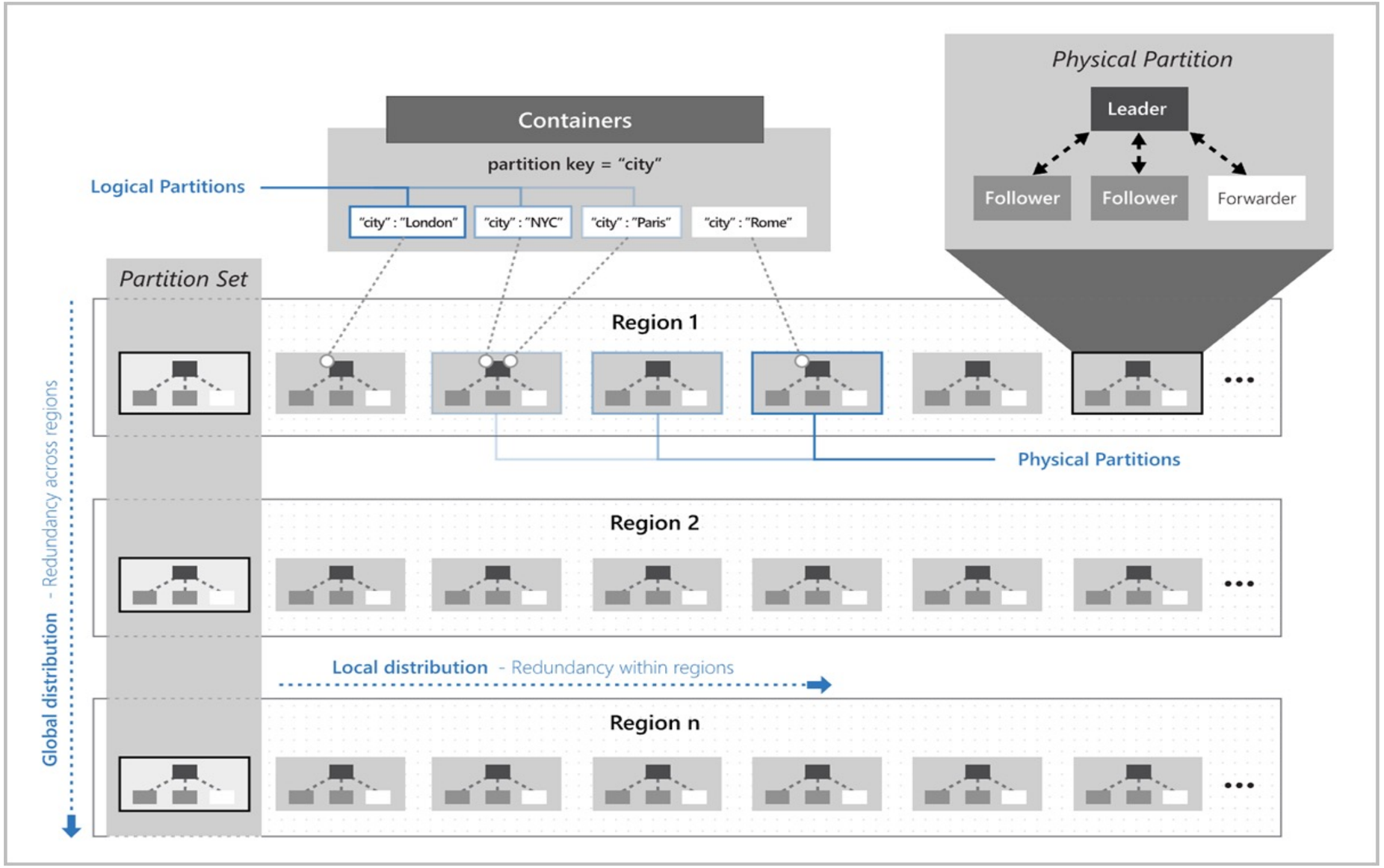
Postgresql API

Regular ORMs that talk to native Postgresql

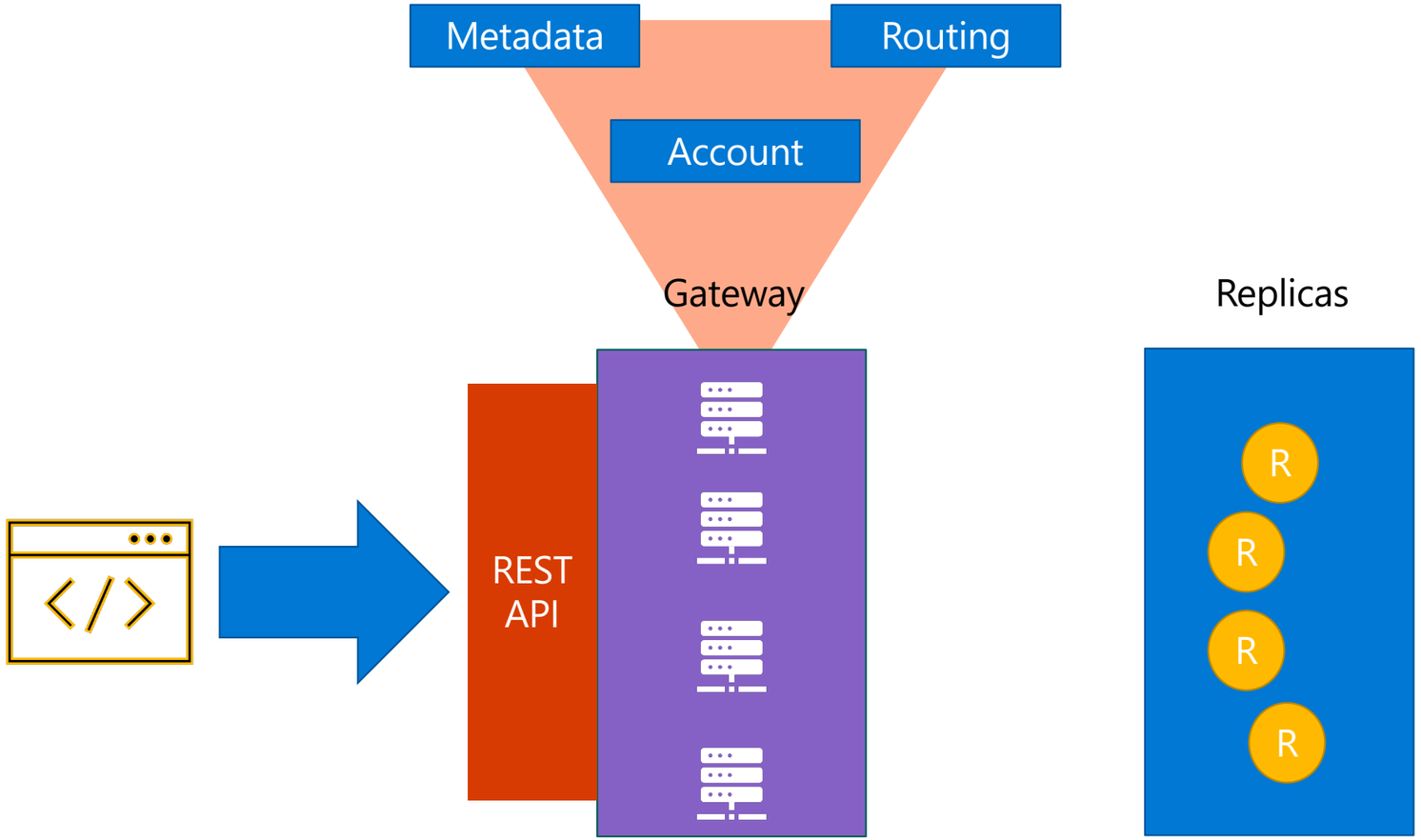
What are the SDK best practices/ Common challenges?

Connectivity modes

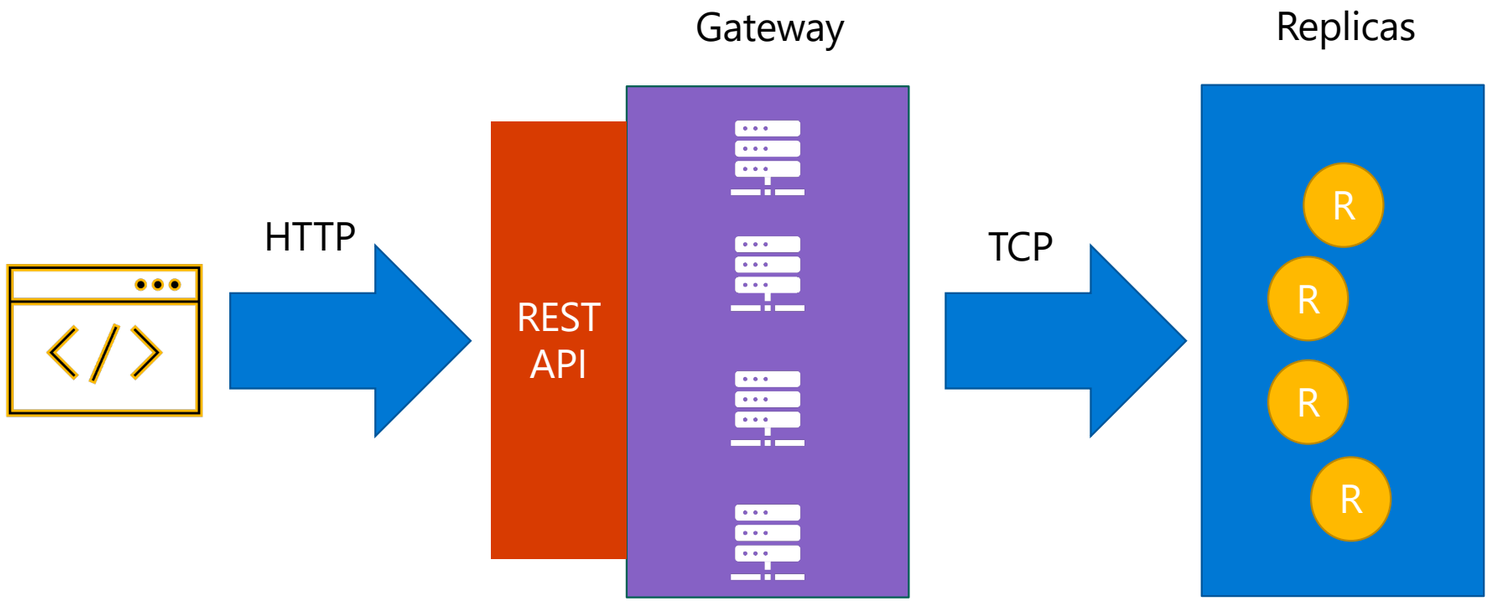
Azure Cosmos DB Topology



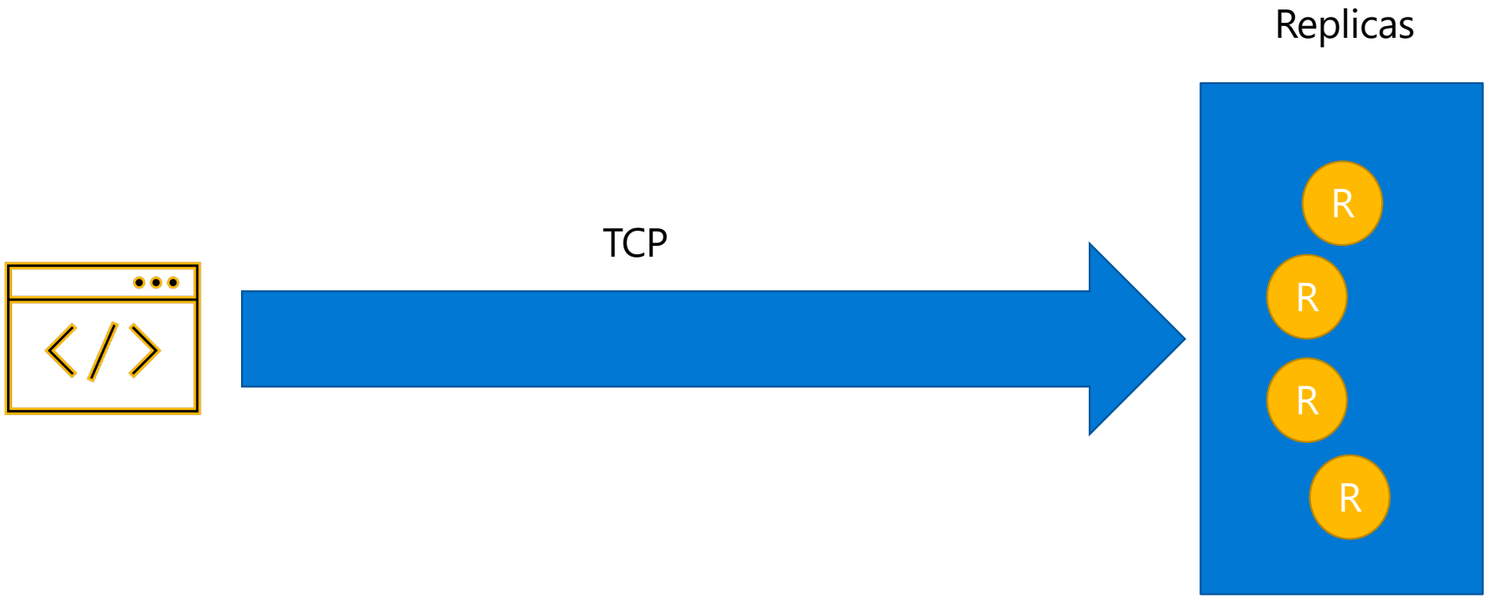
Under the hood



Connectivity mode - Gateway



Connectivity mode - Direct



What are the differences?

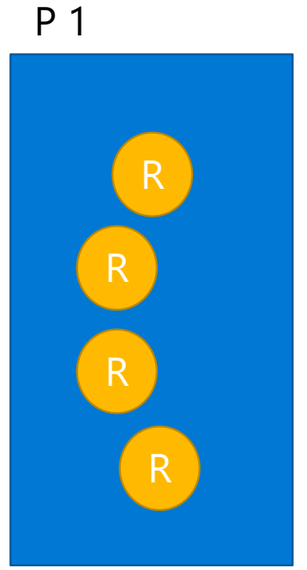
Gateway

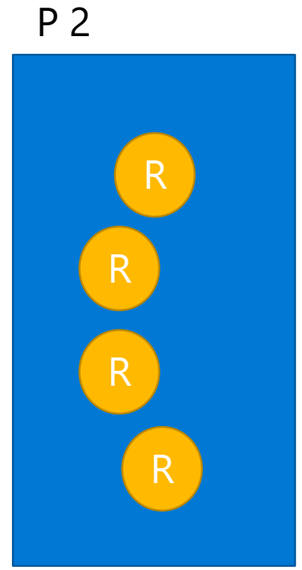
- HTTP protocol
- Compatible with enterprise environment with proxies or hard network rules.
- Single port (443) and known domains.
- Less number of connections.
- Higher latency due to extra network hop

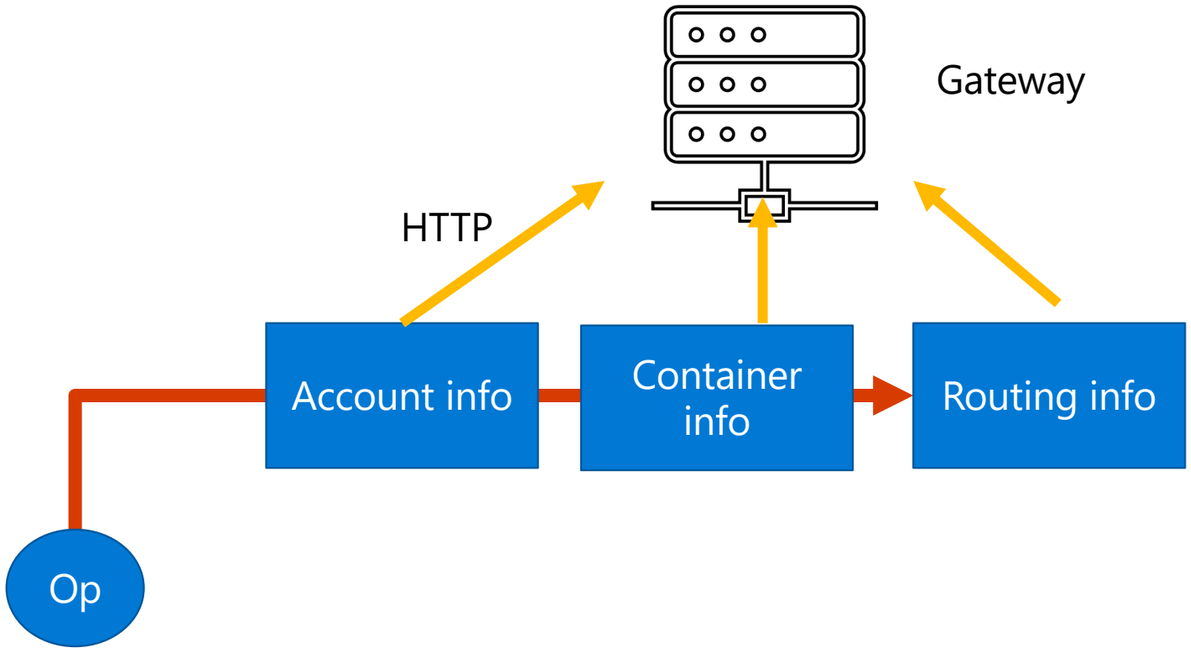
Direct

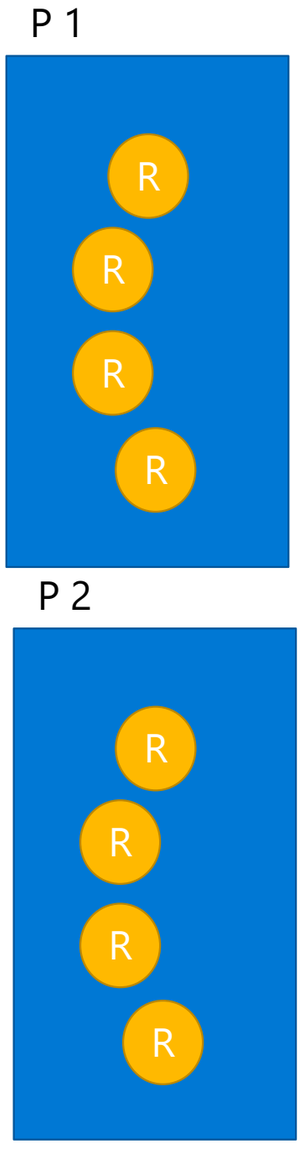
- TCP protocol
- SLA-backed low latency
- Higher port range
- Higher number of connections

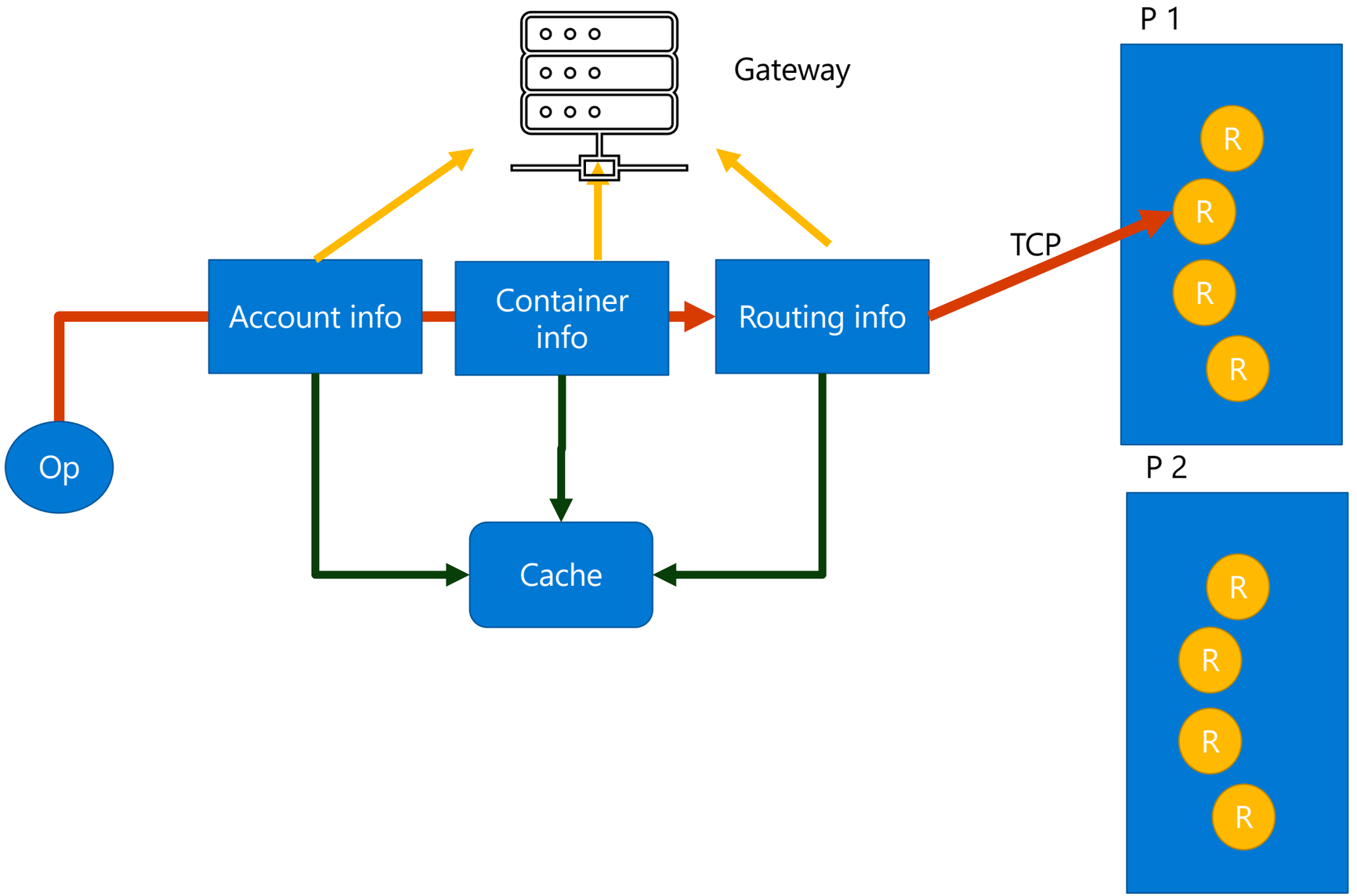

Op

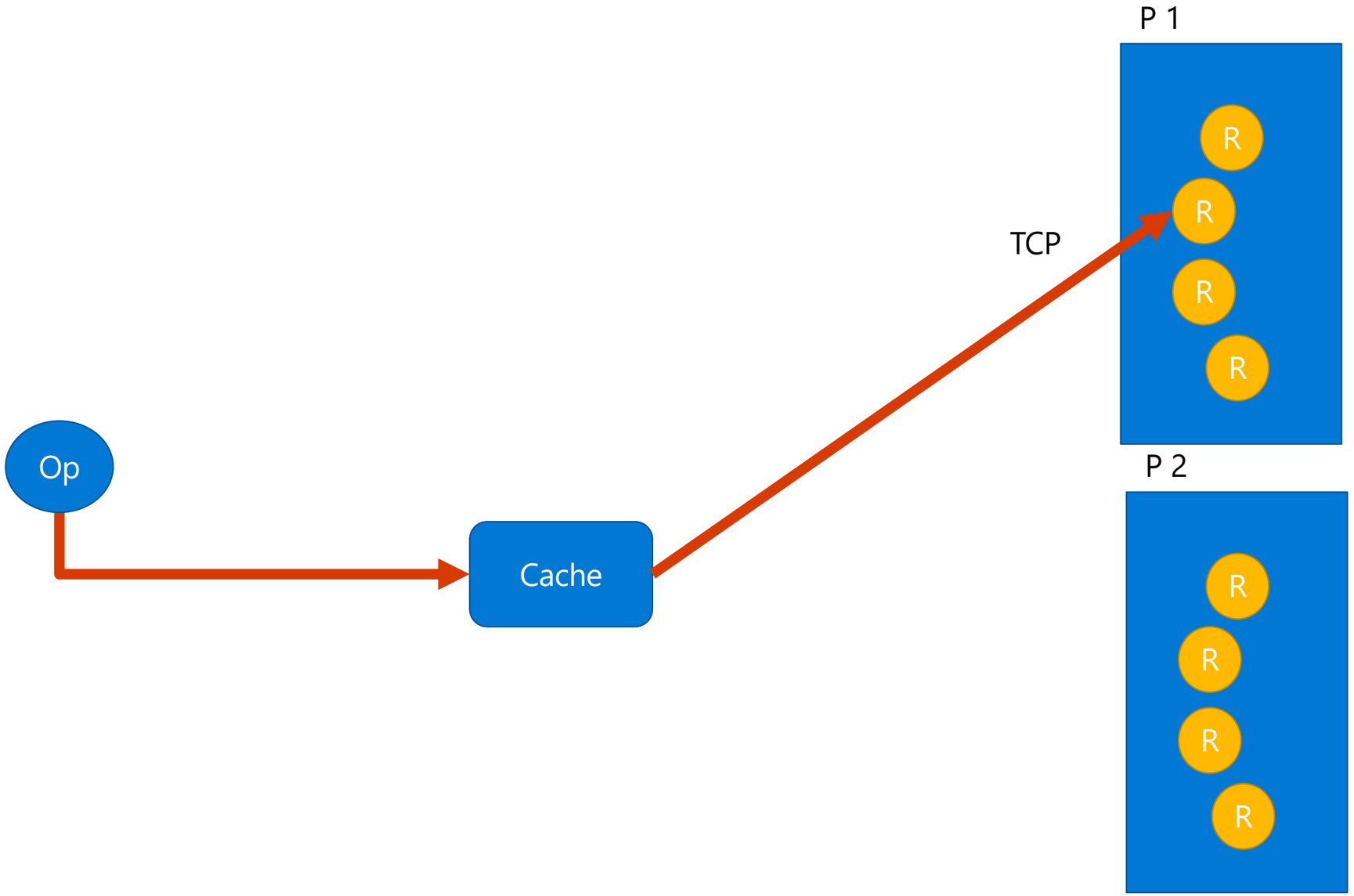

P 1
R
R
R
R

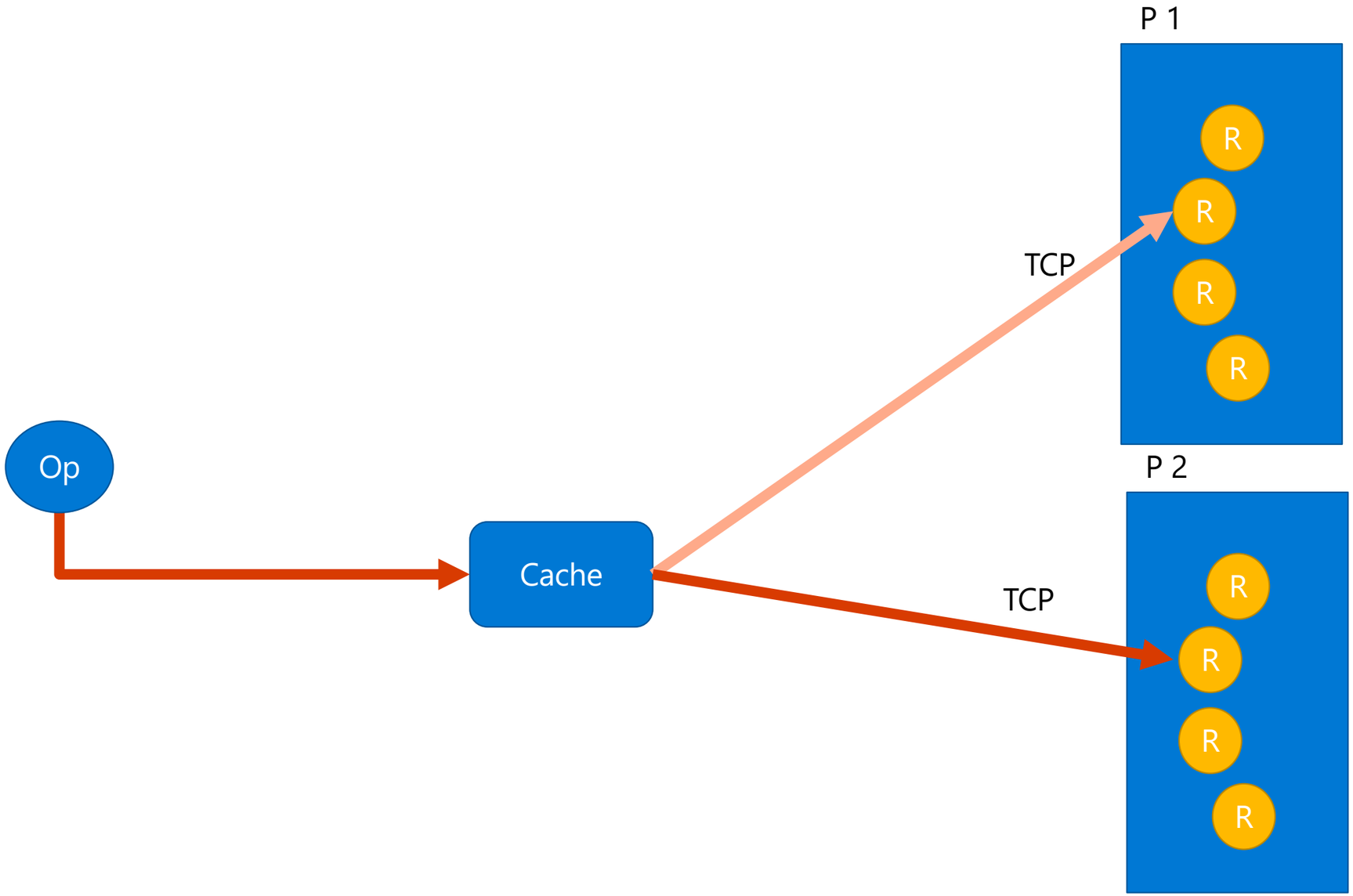

P 2
R
R
R
R


Gateway
HTTP
Account info
Container info
Routing info
Op


P 1
R
R
R
R
P 2
R
R
R
R


Gateway
P 1
R
R
R
R
TCP
Account info
Container info
Routing info
Op
Cache
P 2
R
R
R
R


P 1
R
R
R
R
TCP
Op
Cache
P 2
R
R
R
R


P 1
R
R
R
R
TCP
Op
Cache
P 2
TCP
R
R
R
R

When things don’t go as planned

Diagnostics

Both Java and .NET expose diagnostic information.

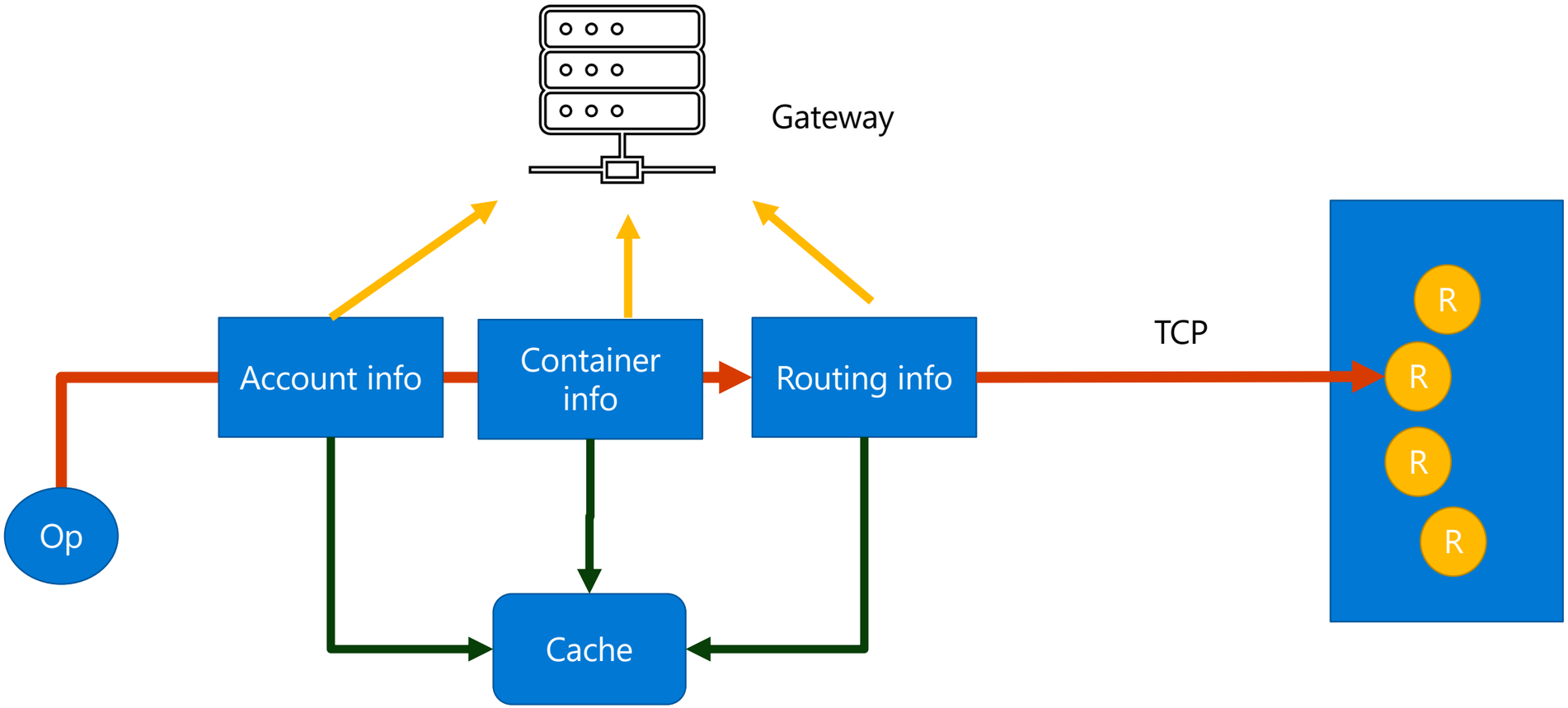
Includes all service interactions, time spent on the SDK (serialization or initialization), server side latency, and connectivity troubleshooting.

.NET

```
try
{
        ItemResponse<MyItem> response = await container.ReadItemAsync<MyItem>(
                partitionKey: new PartitionKey("MyPartitionKey"),
                id: "MyId");
        string diagnostics = response.Diagnostics.ToString();

        TimeSpan elapsedTime = response.Diagnostics.GetClientElapsedTime();

        IReadOnlyList<(string region, Uri uri)> regions = response.Diagnostics.GetContactedRegions();
}
catch (CosmosException cosmosException) {
        string diagnostics = cosmosException.Diagnostics.ToString();

        TimeSpan elapsedTime = cosmosException.Diagnostics.GetClientElapsedTime();

        IReadOnlyList<(string region, Uri uri)> regions = cosmosException.Diagnostics.GetContactedRegions();

    Console.WriteLine(cosmosException.ToString());
}
```

On the Client environment

High latency – cold start



Initialization

The first request pays the cost of initializing the client metadata information.

Either understand that this first request will have high latency, which should not impact the 1h P99 or warm up the client.

.NET

```
List<(string, string)> containers = new List<(string, string)>
{
  ("DatabaseNameForContainer1", "ContainerName1"),
  ("DatabaseNameForContainer2", "ContainerName2")
};

CosmosClientOptions cosmosClientOptions = new CosmosClientOptions
{
    ApplicationName = "MyApplicationName",
   // any other setting I want to customize
};

CosmosClient cosmosClient = await
CosmosClient.CreateAndInitializeAsync(connectionString, containers,
cosmosClientOptions);
```

High latency – regional preference

High latency – regional preference

High latency – regional preference


```
{
    "name": "Microsoft.Azure.Documents.ServerStoreModel Transport Request",
    "id": "0e026cca-15d3-4cf6-bb07-48be02e1e82e",
    "start time": "12:58:20:032",
    "duration in milliseconds": 1638.5957,
    "data": {
      "Client Side Request Stats": {
        "Id": "AggregatedClientSideRequestStatistics",
        "RegionsContacted": [
            "https://maquaran-custrepro-westus2.documents.azure.com/"
        ],
```


```
ItemResponse<MyItem> response = await container.ReadItemAsync<MyItem>(
            partitionKey: new PartitionKey("MyPartitionKey"),
            id: "MyId");

IReadOnlyList<(string region, Uri uri)> regions =
response.Diagnostics.GetContactedRegions();
```

Regional preference

Both Java and .NET SDKs allow the user to set regional preference upon initialization.

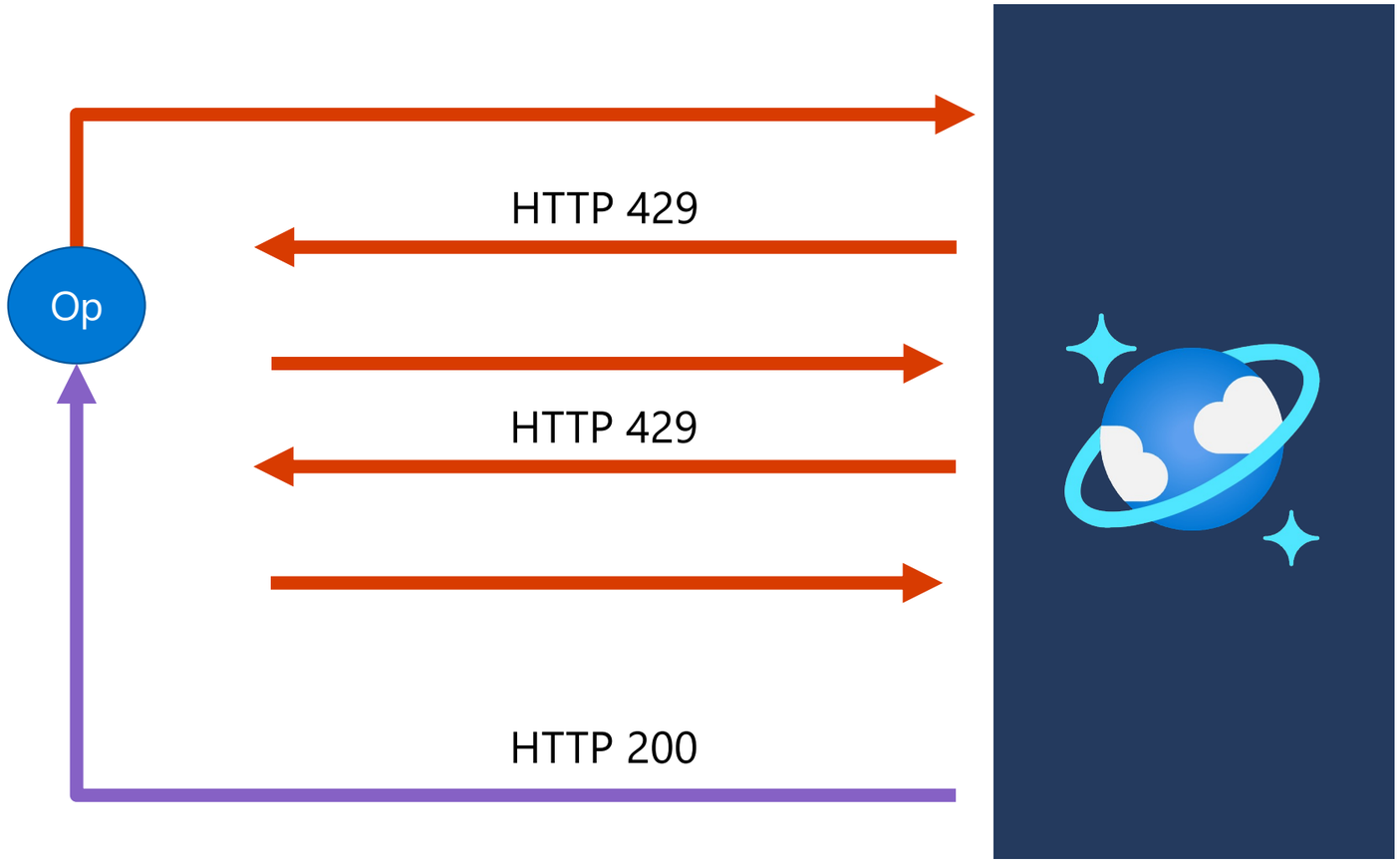
Particularly in .NET, there is an API to indicate on which Azure region the app is running on.

.NET

```
CosmosClientOptions options = new CosmosClientOptions() {
        ApplicationName = “MyApp”,
        ApplicationRegion = Regions.WestUS
};

CosmosClient client = new CosmosClient(“endpoint”, “key”, options);
```

```
CosmosClientOptions options = new CosmosClientOptions() {
        ApplicationName = “MyApp”,
        ApplicationPreferredRegions = new List<string>() {
                Regions.WestUS,
                Regions.CentralUS
        }
};

CosmosClient client = new CosmosClient(“endpoint”, “key”, options);
```

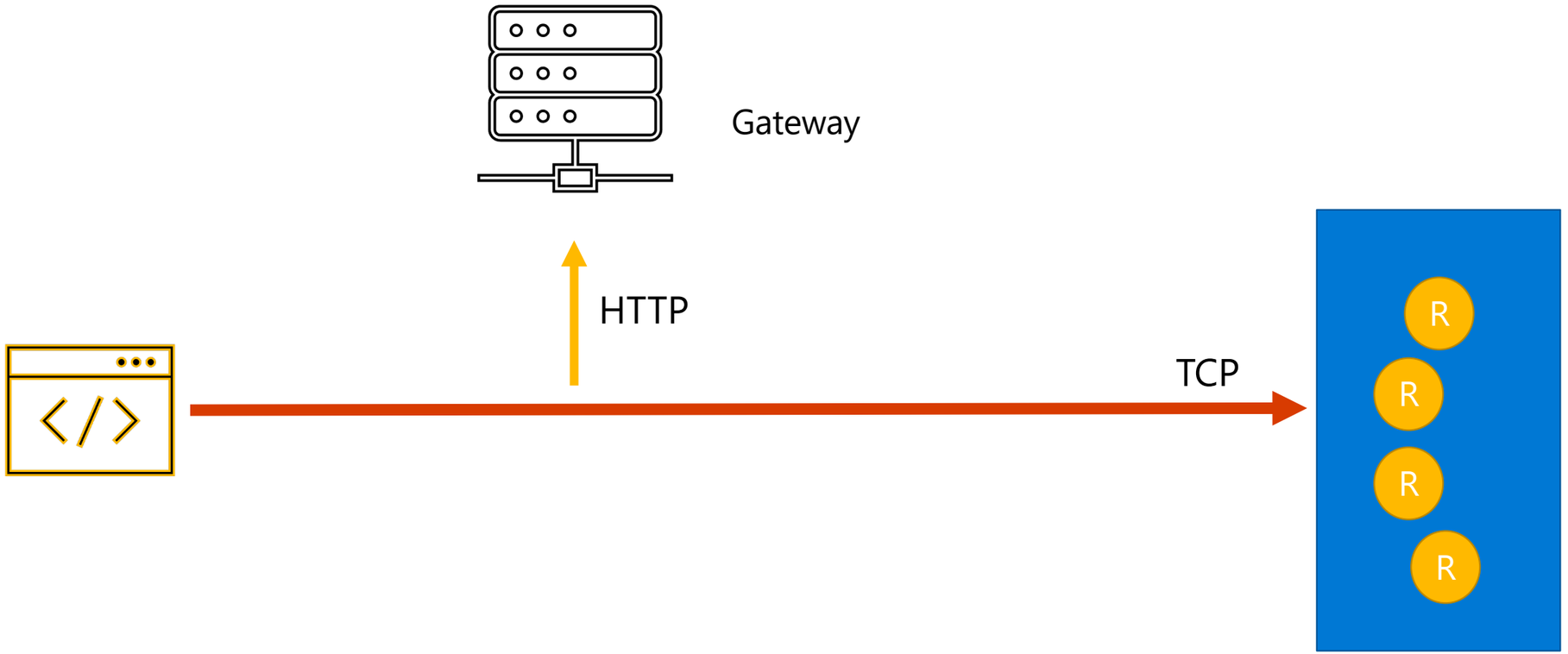
High latency – retries



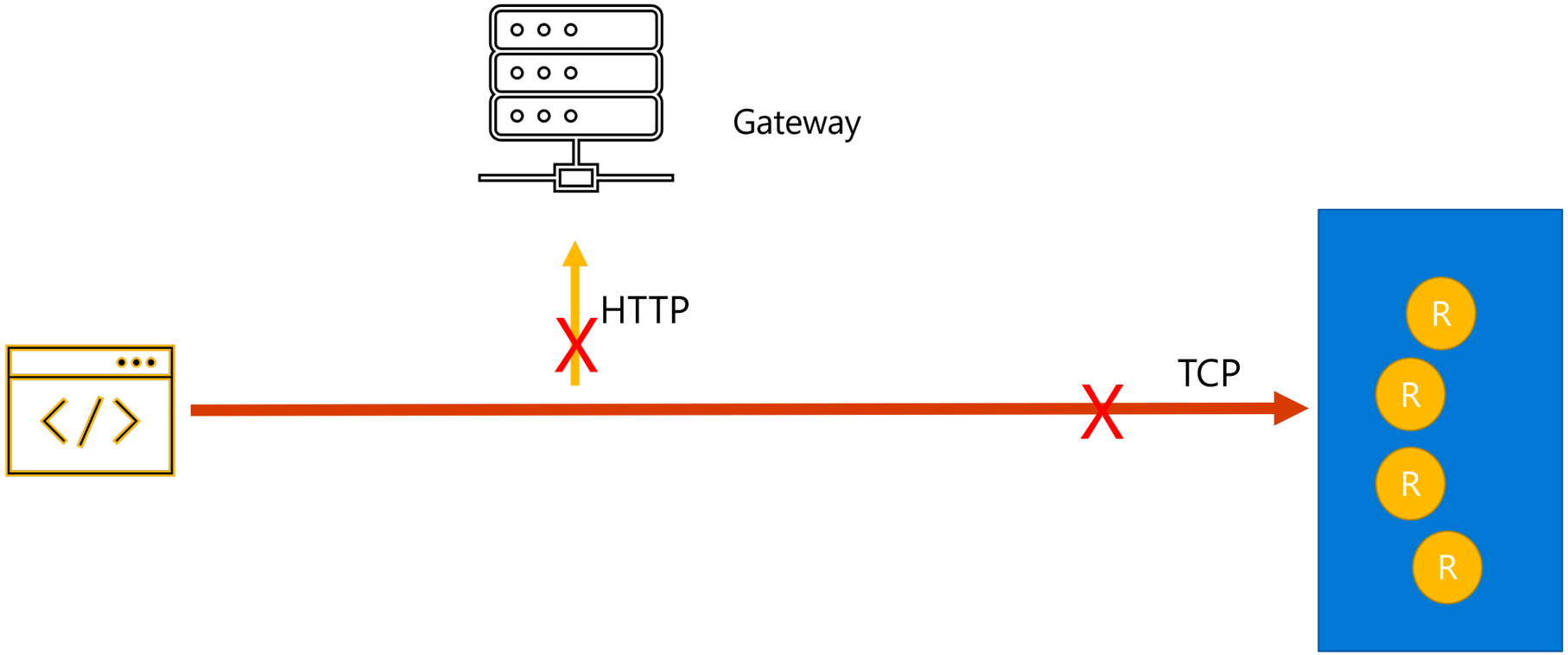
High latency – retries


```
[
        {
                "ResponseTimeUTC": "2021-05-10T20:20:04.3221389Z",
                "ResourceType": "Document",
                "OperationType": "Read",
                "LocationEndpoint": "https://myaccount.documents.azure.com/",
                "StoreResult": {
                        "ActivityId": "d0e640ee-553b-4c3b-ae31-752d947a0b3b",
                        "StatusCode": "TooManyRequests",
                        "BELatencyInMs": null
                }
        },
        {
                "ResponseTimeUTC": "2021-05-10T20:20:04.3221389Z",
                "ResourceType": "Document",
                "OperationType": "Read",
                "LocationEndpoint": "https://myaccount.documents.azure.com/",
                "StoreResult": {
                        "ActivityId": "d0e640ee-553b-4c3b-ae31-752d947a0b3b",
                        "StatusCode": "Ok",
                        "BELatencyInMs": "0.513"
                }
        }
]
```

Timeouts



Timeouts



Timeouts – common client issues

CPU

Connection limiting

Timeouts – common client issues (CPU)

```
CPU history:
    (2021-05-21T00:40:09.1769900Z 60.114),
    (2021-05-21T00:40:19.1763818Z 60.732),
    (2021-05-21T00:40:29.1759235Z 70.000),
    (2021-05-21T00:40:39.1763208Z 100.063),
    (2021-05-21T00:40:49.1767057Z 100.648),
    (2021-05-21T00:40:59.1689401Z 100.137),
CPU count: 8)
```

High CPU

```
CPU history:
    (2021-05-21T00:40:09.1769900Z 5.231),
    (2021-05-21T00:40:59.1763818Z 6.721),
    (2021-05-21T00:41:16.1759235Z 4.000),
    (2021-05-21T00:41:26.1763208Z 17.063),
    (2021-05-21T00:41:48.1767057Z 10.648),
    (2021-05-21T00:42:03.1689401Z 5.137),
CPU count: 8)
```

Thread Starvation

Timeouts – common client issues (connection)


```
{
        "ResponseTimeUTC": "2021-05-10T20:20:04.3221389Z",
        "ResourceType": "Document",
        "OperationType": "Read",
        "LocationEndpoint": "https://myaccount.documents.azure.com/",
        "StoreResult": {
                "ActivityId": "d0e640ee-553b-4c3b-ae31-752d947a0b3b",
                "StatusCode": "Gone",
                "BELatencyInMs": null,
                "TransportException": "A client transport error occurred: The connection
attempt timed out. (Time: 2021-05-10T20:20:04.3221389Z, activity ID: d0e640ee-553b-4c3b-ae31-
752d947a0b3b, error code: ConnectTimeout [0x0006], base error: HRESULT 0x80131500, URI:
rntbd://<somevalue>.documents.azure.com:14155/, connection: <not connected> ->
rntbd://<somevalue>.documents.azure.com:14155/, payload sent: False, CPU history: (2021-05-
10T20:18:56.4258031Z 0.000), (2021-05-10T20:19:06.4265621Z 0.000), (2021-05-
10T20:19:16.4295823Z 0.000), (2021-05-10T20:19:26.4279937Z 0.000), (2021-05-
10T20:19:46.4240876Z 0.000), (2021-05-10T20:19:56.4208742Z 0.000), CPU count: 4)"
        }
}
```

Timeouts – common client issues (connection)

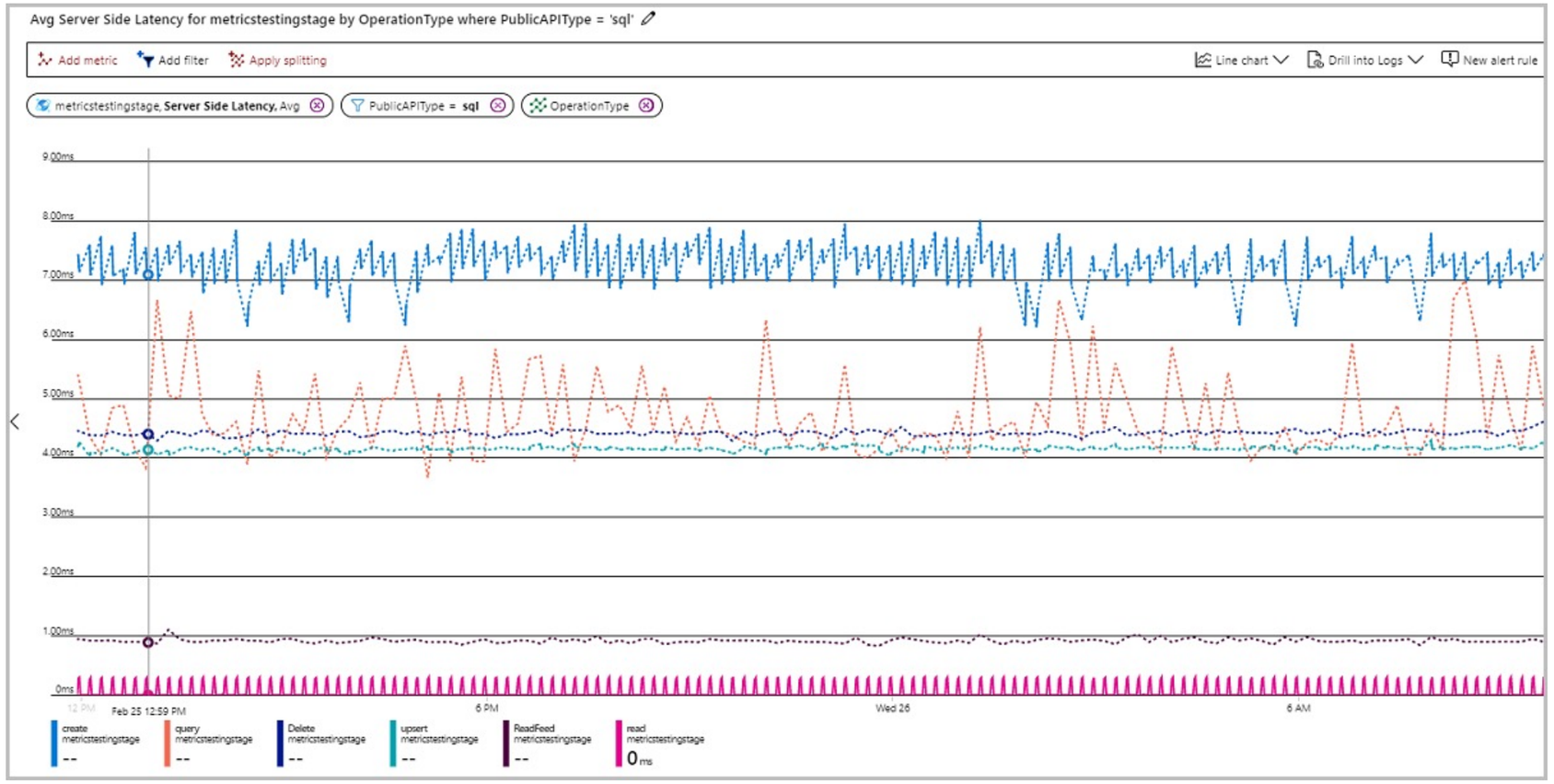
1. Client as a Singleton – one instance reused across the entire application lifetime.
2. Connection limit on the instance – different Azure services might have limits in terms of maximum established connections.
3. Use Gateway mode – if the environment has a low connection limit, like Azure Functions on Consumption Mode.
4. Customize TCP configurations for sporadic loads:
    1. CosmosClientOptions.PortReuseMode = PrivatePortPool
    2. CosmosClientOptions.IdleConnectionTimeout = TimeSpan.FromMinutes(30)

On the service

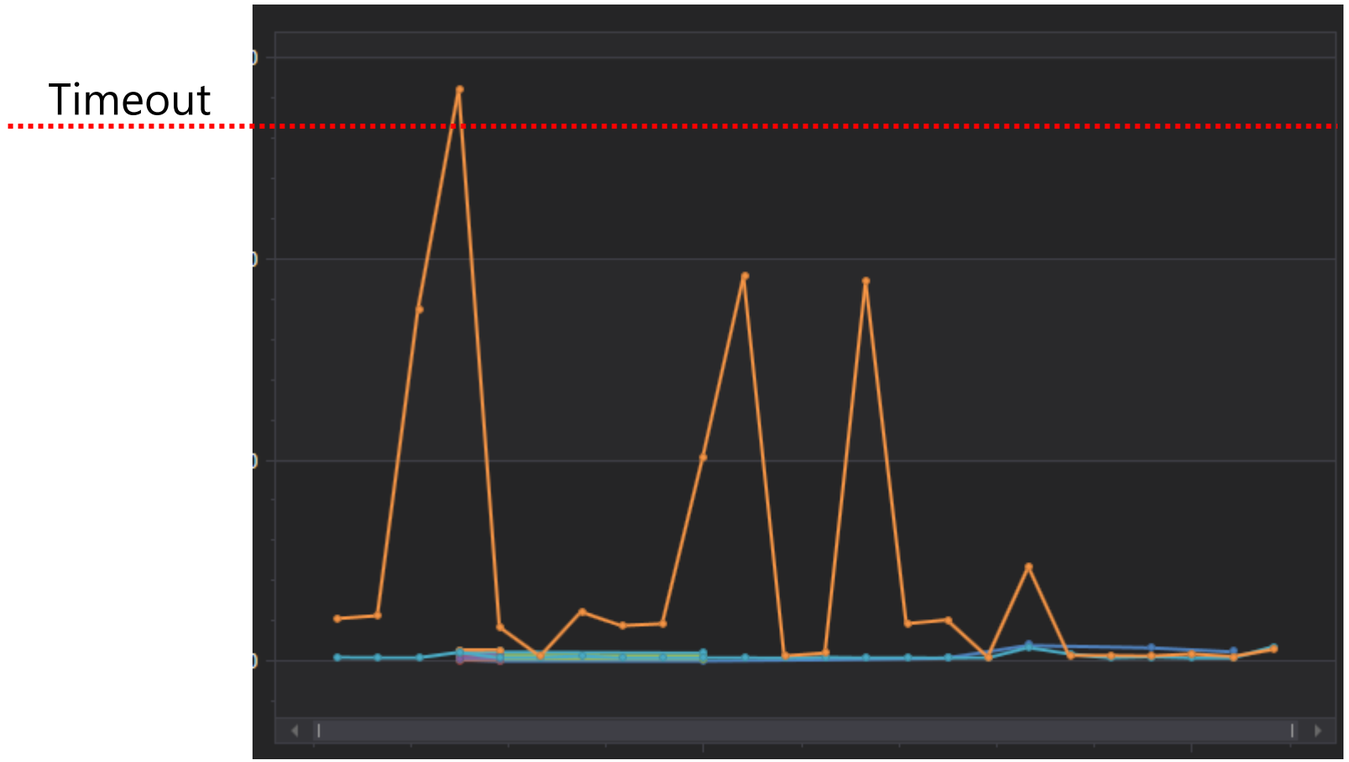
High latency – service side latency


```
{
        "ResponseTimeUTC": "2021-05-21T20:20:04.3221389Z",
        "ResourceType": "Document",
        "OperationType": "Read",
        "LocationEndpoint": "https://myaccount.documents.azure.com/",
        "StoreResult": {
                "ActivityId": "d0e640ee-553b-4c7b-ae31-752d947a0b3b",
                "StatusCode": "Ok",
                "IsValid": true,
                "RequestCharge": 1,
                "BELatencyInMs": "300.366"
        }
}
```

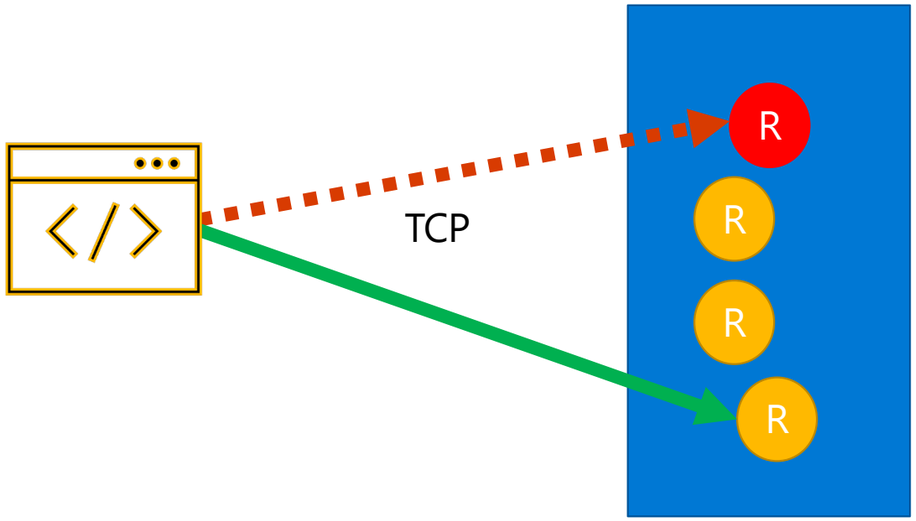
High latency – service side latency



Timeouts – service side latency



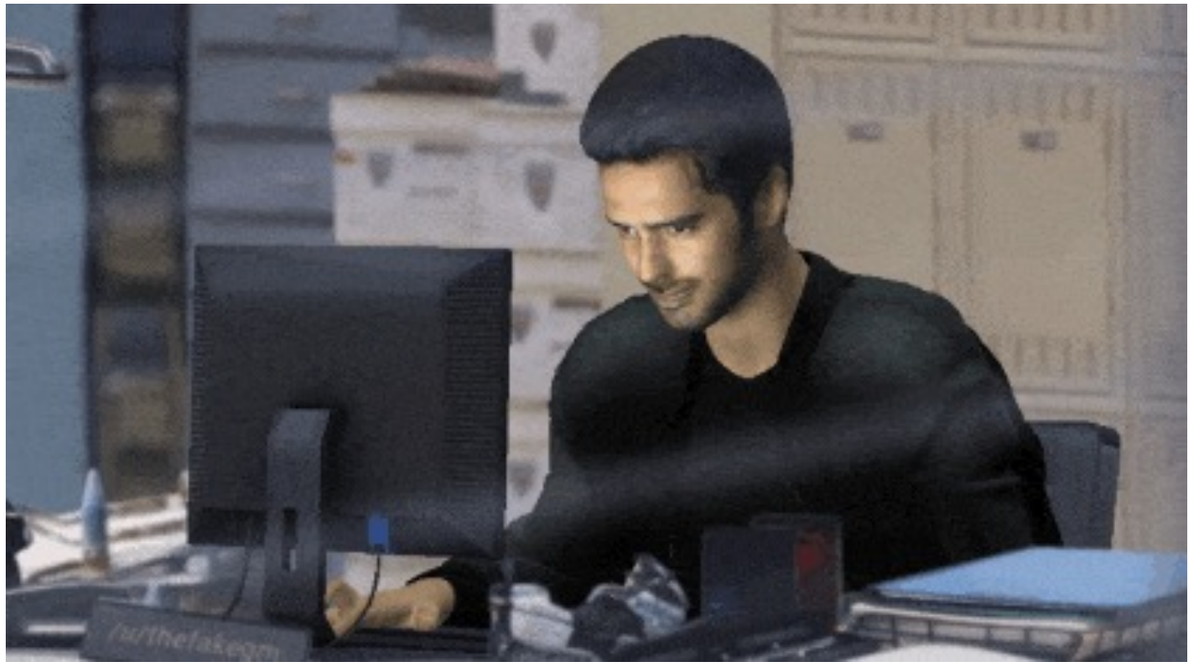
High latency – unavailability on a replica



High latency – unavailability on a replica


```
[
        {
                "ResponseTimeUTC": "2021-05-10T20:20:04.3221389Z",
                "ResourceType": "Document",
                "OperationType": "Read",
                "LocationEndpoint": "https://myaccount.documents.azure.com/",
                "StoreResult": {
                        "ActivityId": "d0e640ee-553b-4c3b-ae31-752d947a0b3b",
                        "StatusCode": "Gone",
                        "BELatencyInMs": null,
                        "TransportException": "A client transport error occurred: The connection attempt timed out.
                        (Time: 2021-05-10T20:20:04.3221389Z, activity ID: d0e640ee-553b-4c3b-ae31-752d947a0b3b, error
                                code: ConnectTimeout …"
        },
        {
                "ResponseTimeUTC": "2021-05-10T20:20:04.3221389Z",
                "ResourceType": "Document",
                "OperationType": "Read",
                "LocationEndpoint": "https://myaccount.documents.azure.com/",
                "StoreResult": {
                        "ActivityId": "d0e640ee-553b-4c3b-ae31-752d947a0b3b",
                        "StatusCode": "Ok",
                        "BELatencyInMs": "0.513"
                }
        }
]
```

Troubleshooting Demo



Common error responses

StatusCode	Meaning	Retry?
400	Invalid content https://aka.ms/CosmosDB/sql/errors/wrong-pk-value https://aka.ms/CosmosDB/sql/errors/missing-id	No
401	Invalid auth. https://aka.ms/cosmosdb-tsg-mac-signature	No
403	Blocked by Access control. https://aka.ms/cosmosdb-tsg-forbidden	Maybe.
404	Resource does not exist. https://aka.ms/cosmosdb-tsg-not-found	No
408	Timeout https://aka.ms/cosmosdb-tsg-request-timeout https://aka.ms/cosmosdb-tsg-request-timeout-java	For reads, the SDK already retries, but further retries are valid. For writes, the operation is idempotent, retry can cause 409.

Common error responses

StatusCode	Meaning	Retry?
409	Conflict	No
412	Optimistic concurrency.	Yes. Requires a read to get a new Etag, apply update, and retry.
413	Max document size reached	No.
429	Throttling	Yes. SDKs already retry.
449	RetryAfter. Multiple concurrent write operations on the same resource.	Yes.
500	Internal server error.	Maybe, 1 retry.
503	Service Unavailable. Multiple timeouts on TCP layer, the client cannot connect. https://aka.ms/cosmosdb-tsg-service-unavailable https://aka.ms/cosmosdb-tsg-service-unavailable-java	Similar to timeouts.

Tools to interact with Azure Cosmos DB

- Visual Studio Code
  - vscode-cosmosdb extension
  - Cosmos DB trigger bindings for Azure Function projects / Visual Studio 2019 has Azure Function project template, deploy/publish

- Cosmos DB Emulator
- Azure DevOps Build task with Cosmos DB Emulator

- Azure portal Data Explorer and https://cosmos.azure.com
- Azure Storage Explorer

How can we learn fast and start fast?

Quickstarts

- Fastest path to “running code”
- “I am a developer who has never worked with Cosmos DB before”
- Quickstarts for each Cosmos DB API

	Core/NOSQL	MongoDB	Cassandra	Gremlin	Table
.NET	Link	Link	Link	Link	Link
Java	Link	Link	Link	Link	Link
Node.js	Link	Link	Link	Link	Link
Python	Link	Link	Link	Link	Link
Go		Link			
PHP				Link	

Platinum Partner

Gold Partner

Silver Partner

Unleash the power of Azure Cosmos DB

Sajeetharan Sinnathurai