# Docker installieren

Install Docker Desktop: <https://hub.docker.com/editions/community/docker-ce-desktop-windows/>

Login Docker

Switch to Linux Containers

## Images holen

|  |
| --- |
| docker login |
| docker pull tinkerpop/gremlin-console |
| docker pull mongo:latest |
| docker pull node:latest |
| docker pull bitnami/cassandra:latest |

# Azure Ressourcen installieren

## Install latest CLI

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>

|  |
| --- |
| Invoke-WebRequest -Uri <https://aka.ms/installazurecliwindows> -OutFile .\AzureCLI.msi; Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'; rm .\AzureCLI.msi |

**Resource Group:** GAB2020

Create CosmosDB Accounts:

* gab2020sql (CORESQL) Notebook Preview ON, NO FREE TIER,

|  |  |
| --- | --- |
| az login |  |
| az account set -s **<yourSubscriptionID>** |  |
| Create Accounts |  |
| az cosmosdb create --resource-group GAB2020 --name **gabus2020sql** --locations regionName=westus | Core SQL |
|  |  |
| az cosmosdb create --resource-group GAB2020 --name **gabus2020cas** --locations regionName=westus --capabilities EnableCassandra | Cassandra |
|  |  |
| az cosmosdb create --resource-group GAB2020 --name **gabus2020graph** --locations regionName=westus --capabilities EnableGremlin | Gremlin |
|  |  |
| az cosmosdb create --resource-group GAB2020 --name **gabus2020mongo** --locations regionName=westus --kind MongoDB | Mongo DB - Wire Protokoll 3.2 |
| **Muss im Portal angelegt werden (keine CLI/ARM Unterstützung zur Zeit) - gabus2020mongo** | [Mongo DB - Wire Protokoll 3.6](https://devblogs.microsoft.com/cosmosdb/azure-cosmos-dbs-api-for-mongodb-now-supports-server-version-3-6/) |
|  |  |

# Git Repository holen

C:  
cd \Source\2020\Events\  
git clone [https://github.com/apollak/GAB2020-CosmosDB-AT.git GAB2020](https://github.com/apollak/GAB2020-CosmosDB-AT.git%20GAB2020)

## Endpoint URL und Passwort für Core SQL setzen

**// SETX setzt das Maschinenweit / Erfordert neue Instanz von Terminal! (nicht nur TAB!)**

|  |
| --- |
| setx EndpointUrl "<https://gabus2020sql.documents.azure.com:443/>" |
| setx PrimaryKey "<your primary key>" |

# Core SQL – Demos

## Basic Demo

### Terminal

|  |
| --- |
| cd C:\Source\2020\Events\GAB2020\CoreSQL\Basic |
| setx EndpointUrl "<Your\_Azure\_Cosmos\_account\_URI>"  setx PrimaryKey "<Your\_Azure\_Cosmos\_account\_PRIMARY\_KEY>" |
| code . |

## Spatial Demo

### Terminal

|  |
| --- |
| cd C:\Source\2020\Events\GAB2020\CoreSQL\Spatial |
| setx EndpointUrl "<Your\_Azure\_Cosmos\_account\_URI>"  setx PrimaryKey "<Your\_Azure\_Cosmos\_account\_PRIMARY\_KEY>" |
| code . |

# Graph-Demo

<https://portal.azure.com>

Select: **gabus2020graph**

DataExplorer

* Create Database 400 RU/s (**graphdb**)
* Create Collection (**thehobbit**) - Partition Key (/p)

|  |
| --- |
| **cd C:\Source\2020\Events\GAB2020\Gremlin\msgraph** |
| setx GraphEndpointUrl "<your graph db account>.gremlin.cosmos.azure.com"  setx GraphPrimaryKey "<your primary key>" |

## Terminal

cd C:\Source\2020\Events\GAB2020\Gremlin

mkdir conf

code remote.yaml

|  |
| --- |
| **remote.yaml** (<https://docs.microsoft.com/en-us/azure/cosmos-db/create-graph-gremlin-console>) |
| hosts: [your\_database\_server.gremlin.cosmos.azure.com]  port: 443  username: /dbs/your\_database\_account/colls/your\_collection  password: your\_primary\_key  connectionPool: {  enableSsl: true  }  serializer: { className: org.apache.tinkerpop.gremlin.driver.ser.GraphSONMessageSerializerV2d0, config: { serializeResultToString: true }} |

## 

## Open Terminal

|  |  |
| --- | --- |
| C:\Source\2020\Events\GAB2020\Gremlin |  |
| docker run -it --volume ${PWD}:/pc --name graph tinkerpop/gremlin-console bash | Beim ersten Aufruf |
| docker start -i graph | Danach |

**Alternativ**: Cd C:\Source\2020\Events\GAB2020\Gremlin\msGraph => Simple REPL verwenden

|  |
| --- |
| **Connect to Graph** |
| :remote connect tinkerpop.server /pc/conf/remote.yaml |
| :remote console |
| g.addV('person').property('name','Bilbo Beutlin').property('p','ring'); |
| g.addV('person').property('name','Gandalf').property('p','ring');  g.addV('person').property('name','Thorin').property('p','ring');  g.addV('person').property('name','Dwalin').property('p','ring');  g.addV('person').property('name','Balin').property('p','ring');  g.addV('person').property('name','Kili').property('p','ring');  g.addV('person').property('name','Fili').property('p','ring');  g.addV('person').property('name','Dori').property('p','ring');  g.addV('person').property('name','Nori').property('p','ring');  g.addV('person').property('name','Ori').property('p','ring');  g.addV('person').property('name','Oin').property('p','ring');  g.addV('person').property('name','Gloin').property('p','ring');  g.addV('person').property('name','Bifur').property('p','ring');  g.addV('person').property('name','Bofur').property('p','ring');  g.addV('person').property('name','Bombur').property('p','ring'); |
| g.V() |
| **Bilbo connecting (pre corona)** |
| g.V().has('name','Bilbo Beutlin').addE('knows').to(g.V().has('name','Gandalf')) |
| g.V().has('name','Bilbo Beutlin').addE('knows').to(g.V().has('name','Thorin')) |
| **Gandalf connects to everybody** |
| g.V().has('name','Gandalf').as('gandalf').V().hasLabel('person').where(neq('gandalf')).as('p').V().has("name","Gandalf").addE("knows").to(select('p')) |
| **Add Places** |
| g.addV('place').property('name','Hobbithöhle').property('p','ring');  g.addV('place').property('name','Gasthaus Zum grünen Drachen').property('p','ring');  g.addV('place').property('name','Troll Lagerplatz').property('p','ring');  g.addV('place').property('name','Trollhöhle').property('p','ring');  g.addV('place').property('name','Rivendell').property('p','ring');  g.addV('place').property('name','Elronds Haus').property('p','ring');  g.addV('place').property('name','Orkhöhle').property('p','ring'); |
| **Add Paths** |
| g.V().has('name','Hobbithöhle').addE('path').to(g.V().has('name','Gasthaus Zum grünen Drachen')).property('weight',2.0);  g.V().has('name','Gasthaus Zum grünen Drachen').addE('path').to(g.V().has('name','Troll Lagerplatz')).property('weight',4.0);  g.V().has('name','Troll Lagerplatz').addE('path').to(g.V().has('name','Trollhöhle')).property('weight',1.0);  g.V().has('name','Troll Lagerplatz').addE('path').to(g.V().has('name','Rivendell')).property('weight',3.0);  g.V().has('name','Rivendell').addE('path').to(g.V().has('name','Elronds Haus')).property('weight',1.0);  g.V().has('name','Elronds Haus').addE('path').to(g.V().has('name','Orkhöhle')).property('weight',5.0);  g.V().has('name','Gasthaus Zum grünen Drachen').addE('path').to(g.V().has('name','Rivendell')).property('weight',4.0);  g.V().has('name','Gasthaus Zum grünen Drachen').addE('path').to(g.V().has('name','Orkhöhle')).property('weight',8.0); |
|  |
| g.V(); g.E(); |
|  |
| g.V().has('name','Hobbithöhle').repeat(outE().inV().simplePath()).until(has('name','Orkhöhle')).path().by(coalesce(values('weight'),constant(0.0))).map(unfold().sum()); |
| g.V().has('name','Hobbithöhle').repeat(outE().inV().simplePath()).until(has('name','Orkhöhle')).path().by(coalesce(values('name'),constant(0.0))); |
| g.V().has('name','Hobbithöhle').repeat(outE().inV().simplePath()).until(has('name','Orkhöhle')).path().by(coalesce(values('name','weight'),constant(0.0))); |
|  |
| :remote console |
| :quit |

# MongoDB

<https://portal.azure.com>

Select: **gabus2020mongo**

Select: **ConnectionString => Primary Key**

**Adapt primary-template.key file and copy to primary.key**

## Terminal

|  |  |
| --- | --- |
| cd C:\Source\2020\Events\GAB2020\Mongo |  |
| docker run -it --volume ${PWD}:/home --name mongo mongo bash | Beim ersten Aufruf |
| docker start -i mongo | Danach |

|  |
| --- |
| **Connect to MongoDB** |
| cd /home |
| . ./primary.key |
| mongo gabus2020mongo.mongo.cosmos.azure.com:10255 -u gabus2020mongo -p $primarykey --tls --tlsAllowInvalidCertificates |

## News

* Azure Cosmos DB’s API for MongoDB unterstützt **Server Version 3.6**
  + Verbesserte Latency bei Group, Count and Skip-Limit
  + Compound indexes
  + ChangeFeed support via ChangeStream API
  + Creating unsharded collections under databases with throughput
  + Aggregation pipeline stages/operators

## Unsharded Collections (Database Throughput does no longer require a partition key)

|  |
| --- |
|  |
| use mydb |
| **Create explicit Throughput** |
| db.runCommand({customAction: "CreateDatabase", offerThroughput: 400}); |
| **Create explicit Throughput on collection (based on DB-Value!)** |
| db.createCollection("democol"); |
| **Create implicit Throughput on collection (uses db Rus)** |
| db.runCommand({customAction: "CreateCollection", collection: "democol2"}); |
| **Create explicit Throughput on collection based on parameter!** |
| db.runCommand({customAction: "CreateCollection", collection: "democol3", offerThroughput: 1000 }); |
| db.democol.insert({a:1}); |
| **Creates implicit throughput on collection (uses db Rus)** |
| db.runCommand({customAction: "CreateCollection", collection: "democol1", shardKey: "a.b" }); |
| db.democol1.insert({a:{ b: 'Hugo' }}); |
| db.democol1.insert({a:{ c: 'Hugo' }}); |
| **Works in previous collection because there is no shard key** |
| db.democol.insert({a:{ c: 'Hugo' }}); |

## Compound Indexes (democol2)

Compound indexes give a huge performance boost for queries with:

* **Multiple properties in the Sort()** definition

* Find() and Sort() where the **property in the Find() definition is part of Sort()**

* Find() with multiple properties filter where at least one property filter is equality.(e.g. db.test.find({a:1, b: {$gte: 5}, b: {$lte: 7}}); )

|  |
| --- |
| db.democol2.insertMany( [  { a:1, b: 5, i:"Hansi" },  { a:1, b: 2, i:"Susi" },  { a:2, b: 10, i:"Maria" },  { a:1, b: 7, i:"Karl" },  { a:1, b: 9, i:"Cornelia" },  ] ); |
| db.democol2.createIndex({a:1, b:1}) |
| db.democol2.find({a:1, b: {$gte: 5}, b: {$lte: 7}}) |
| db.democol2.find({a:1, b: {$gte:5, $lte: 7}}) |

## ChangeFeed via ChangeStream API (democol)

<https://docs.mongodb.com/manual/reference/method/db.collection.watch/#db.collection.watch>

|  |
| --- |
| Watch Changefeed for documents that have property a set! |
| var watchCursor = db.democol.watch(  [  {  $match: {  $and: [  { "fullDocument.a": 1 },  { "operationType": { $in: ["insert", "update", "replace"] } }  ]  }  },  { $project: { "\_id": 1, "fullDocument": 1, "ns": 1, "documentKey": 1} }  ],  { fullDocument: "updateLookup" }); |
| watchCursor.hasNext() |
| db.democol.insert({a:1,b:"Hello Change"}) |
| watchCursor.hasNext() |
| watchCursor.next() |
| watchCursor.close() |
| while (!watchCursor.isExhausted()){  if (watchCursor.hasNext()){  printjson(watchCursor.next());  }  } |

## Faster Aggregation pipeline stages/operators (democol2)

|  |
| --- |
| use mydb |
| Example-Data |
| db.democol2.insertMany( [  { animal:"hedgehog", color:"brown", a:1 },  { animal:"hedgehog", color:"indigo", a:2 },  { animal:"hedgehog", color:"blue", a:3 },  { animal:"sheep", color:"blue", a:5 },  { animal:"sheep", color:"yellow", a:6 },  { animal:"sheep", color:"brown", a:7 },  { animal:"sheep", color:"orange", a:8 },  { animal:"ape", color:"orange", a:4 },  { animal:"ape", color:"black", a:3 }  ] ); |
| Sample Aggregates |
| db.democol2.aggregate({$sortByCount:"$animal"}) |
| db.democol2.aggregate([{$group:{\_id:"$color", count:{$sum:1},a:{$min:"$a"}}},{$sort:{count:-1}}]) |
| Delete items from collection |
| db.democol2.deleteMany({}); |
| exit |

## TTL (Time To Live Support) (democol)

|  |
| --- |
| Default 10 Seconds |
| db.democol.createIndex({"\_ts":1}, {expireAfterSeconds: 10}) |
| db.democol.insertMany( [  { name:"TTL Hansi" },  { name:"TTL Susi", ttl:6 }  ] ); |
| db.democol.find({}); |
| Drop Index |
| db.democol.dropIndex({"\_ts":1}) |
| No Default |
| db.democol.createIndex({"\_ts":1}, {expireAfterSeconds: -1}) |
| db.democol.insertMany( [  { name:"TTL Hansi" },  { name:"TTL Susi", ttl:6 }  ] ); |
| db.democol.find({}); |
| db.democol.find({}); |

# Cassandra - Demo

## Start Node Environment

|  |  |
| --- | --- |
| 2 Terminal Sessions öffnen und in beiden in das nachfolgende Verzeichnis wechseln |  |
| cd C:\Source\2020\Events\GAB2020\Cassandra |  |
| Terminal-1 (Node Environment) |  |
| docker run -it -p 3000:3000 --volume ${PWD}:/home --name nodedev node bash | #Initial run |
| docker start -i nodedev | #restart run |
| docker container rm nodedev | #remove container |

## Cassandra Simple Demo

|  |
| --- |
| Terminal-1 Node Environment |
| cd /home |
| git clone <https://github.com/doanduyhai/Cassandra-NodeJS-Demo.git> |
| git clone <https://github.com/Azure-Samples/azure-cosmos-db-cassandra-nodejs-getting-started.git> |
| cp config.js Cassandra-NodeJS-Demo/config.js |
| cp config.js azure-cosmos-db-cassandra-nodejs-getting-started/config.js |
| cd Cassandra-NodeJS-Demo |
| npm install |
| openssl s\_client -connect gabus2020cas.cassandra.cosmos.azure.com:10350 | openssl x509 -out gabus2020cas.pem |

|  |
| --- |
| Terminal-2 Node Environment |
| code . |
| Show config-Template.js  Navigate to /Cassandra-NodeJS-Demo/src/Ingestionjs |

## Navigate to: <https://portal.azure.com>

Open **gabus2020cas**

Click DataExplorer

New Keyspace: nodejs\_demo

Click **Quickstart** - **NodeJS**

|  |
| --- |
| Change Code: src/Ingestion.js and src/Reading.js |
| Replace the following line of code:  **var** client = new cassandra.Client({contactPoints: ['localhost']}); |
|  |
| **WITHOUT SSL-VALIDATION** |
| var config = require('../config');  process.env.NODE\_TLS\_REJECT\_UNAUTHORIZED = "0";  const authProviderLocalCassandra = new cassandra.auth.PlainTextAuthProvider(config.username, config.password);  const client = new cassandra.Client({contactPoints: [config.contactPoint], authProvider: authProviderLocalCassandra,sslOptions: { rejectUnauthorized: false },localDataCenter: 'West US'}); |
| OPTIONAL: In package.json replace: "cassandra-driver": "datastax/nodejs-driver" with: "cassandra-driver": "^3.3.0" |
|  |
| **WITH SSL-VALIDATION** |
| var ssl\_option = {  cert : fs.readFileSync("/home/Cassandra-NodeJS-Demo/gabus2020cas.pem"),  secureProtocol: 'TLSv1\_2\_method'  };  var config = require('../config');  const authProviderLocalCassandra = new cassandra.auth.PlainTextAuthProvider(config.username, config.password);  const client = new cassandra.Client({contactPoints: [config.contactPoint], authProvider: authProviderLocalCassandra,  sslOptions: ssl\_option,localDataCenter: 'West US'}); |
|  |
| Remove:  ,**function**(nextCall) {  client.execute("TRUNCATE nodejs\_demo.us\_unemployment", [],nextCall)} |
|  |

|  |
| --- |
| Terminal-1 Node Environment |
| node src/Ingestion.js |
| node src/Reading.js |
|  |
| rm -r -f Cassandra-NodeJS-Demo/ |
| rm -r -f azure-cosmos-db-cassandra-nodejs-getting-started/ |