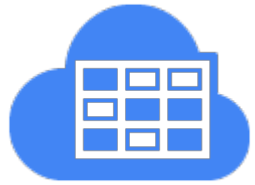

Google
Developers

Getting Started with Google Cloud Datastore



codelab

Johan Euphrosine
Developer Programs Engineer

David Gay
Software Engineer

Takashi Matsuo
Developer Programs Engineer

About us

. Johan Euphrosine, Developer Programs Engineer

. David Gay, Software Engineer

. Takashi Matsuo, Developer Programs Engineer

Prerequisites

. Visit:

goo.gl/whUZZ

. Sign up for:

Getting Started with **Google Cloud Datastore**

. Click on Install SSH

. Click on Start SSH

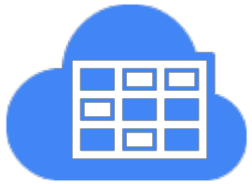
password: **io2013**

Demo

The Google Cloud Datastore

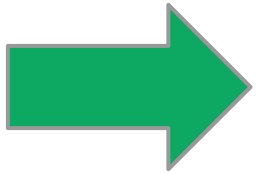
Fully Managed Schemaless Storage

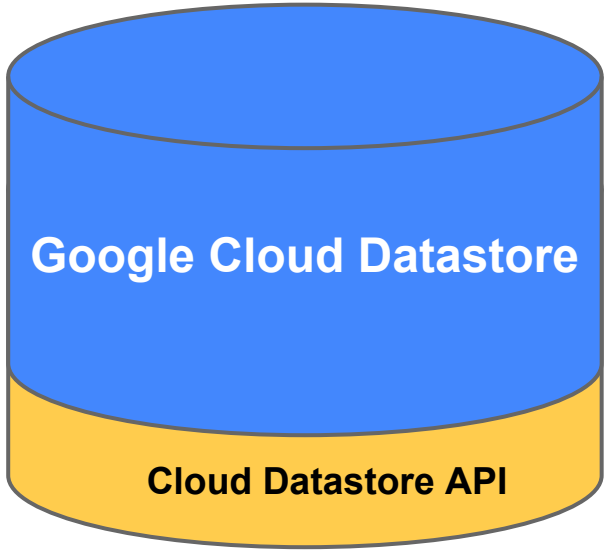
App Engine

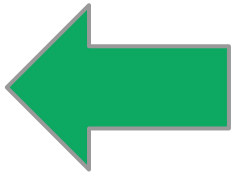


Compute Engine

#golang
#angularjs
Todo webapp







#nodejs
Todo CLI

#socketstream
Todo webapp

Google Cloud Datastore

API Frontend

Cloud Datastore Service

Megastore

BigTable

Colossus

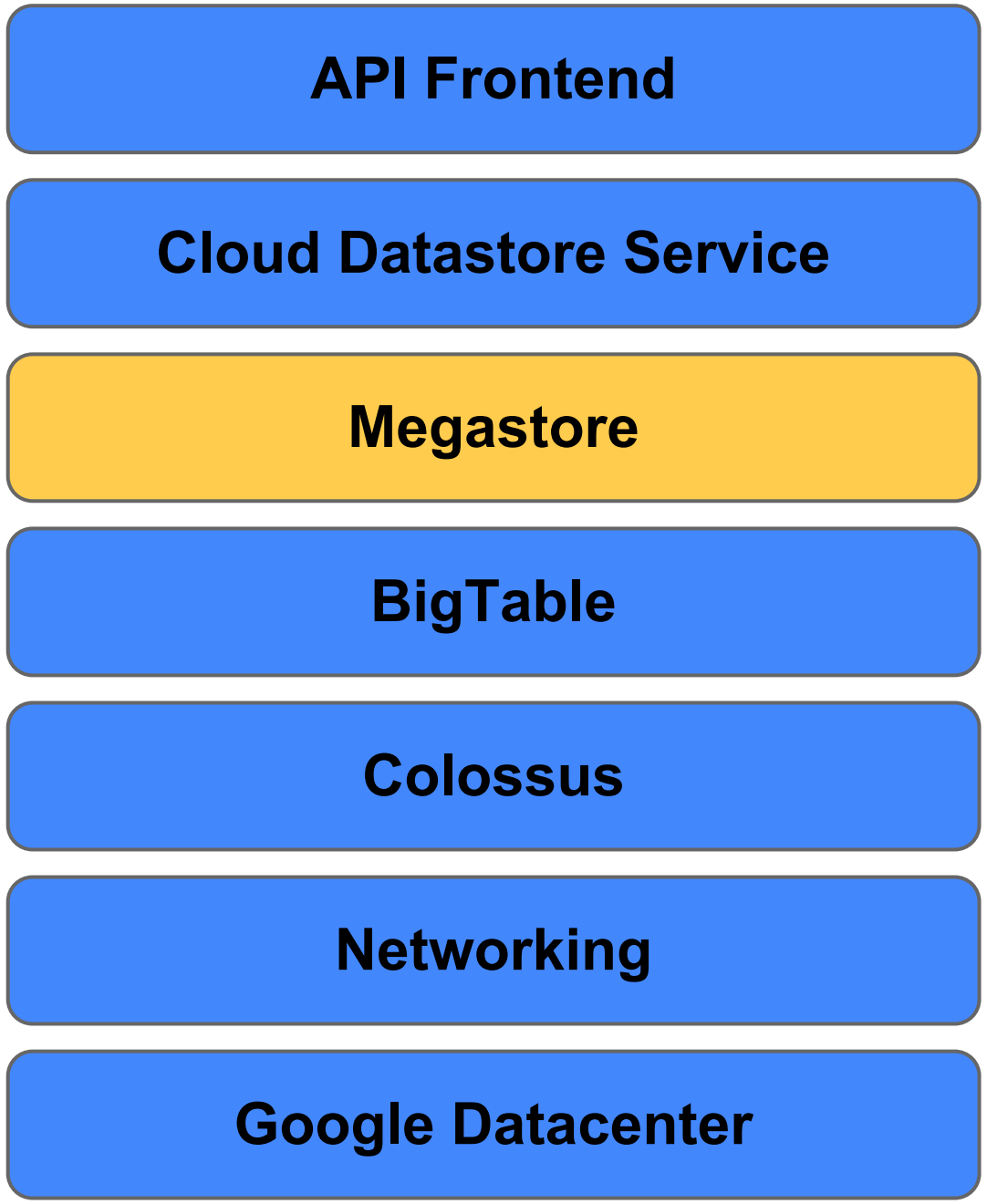
Networking

Google Datacenter

Google Cloud Datastore

High Availability

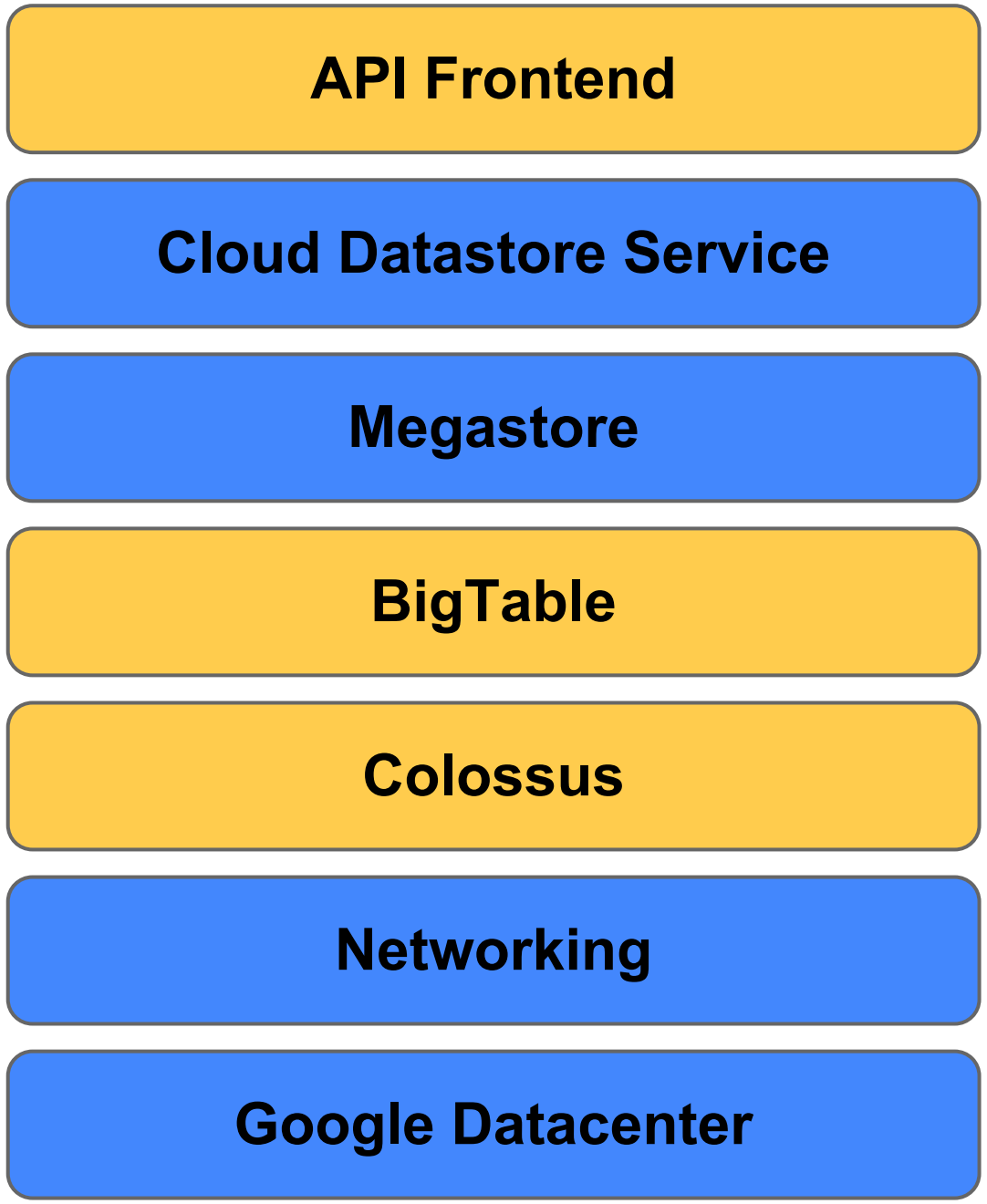
- Auto-replication across multiple datacenters
- Paxos consensus
- Strong and Eventual consistency



Google Cloud Datastore

High Scalability

- Horizontal auto-scaling
- Huge capacity
- High durability



Google Cloud Datastore

Ubiquitous Accessibility

Managed Frontend

App Engine SDK

Unmanaged Backend

Cloud Datastore API

API Frontend

Cloud Datastore Service

Megastore

BigTable

Colossus

Networking

Server Hardware

Quick Datastore Overview

"Non-relational" schemaless database with ACID transactions, flexible queries

Basic concepts:

- entities with single and list-valued properties
- hierarchical keys

Will be covered later:

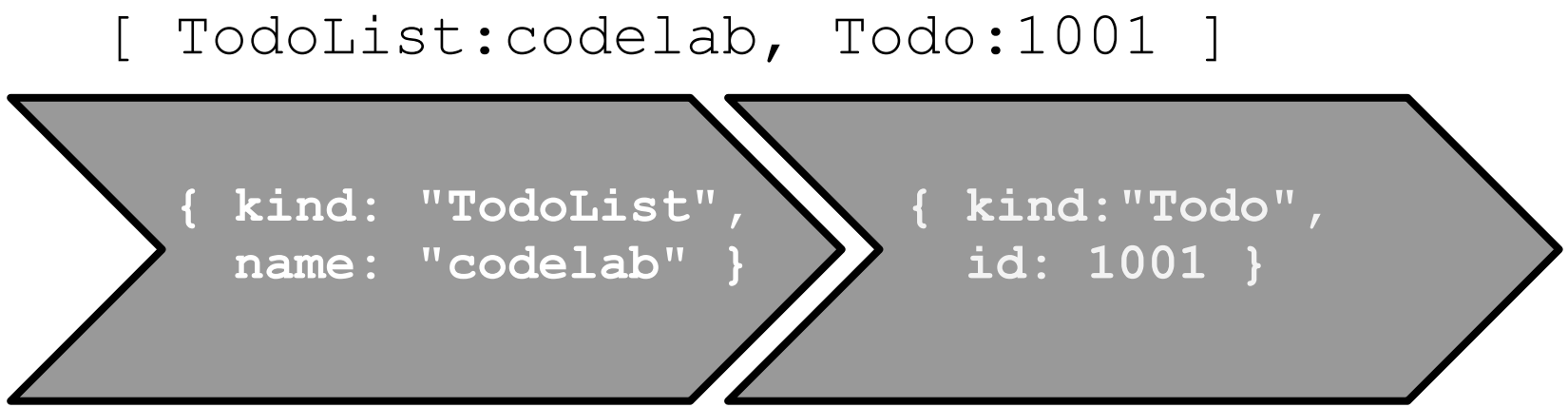
- transactions, queries and entity groups

Keys and Entities

Entity keys:

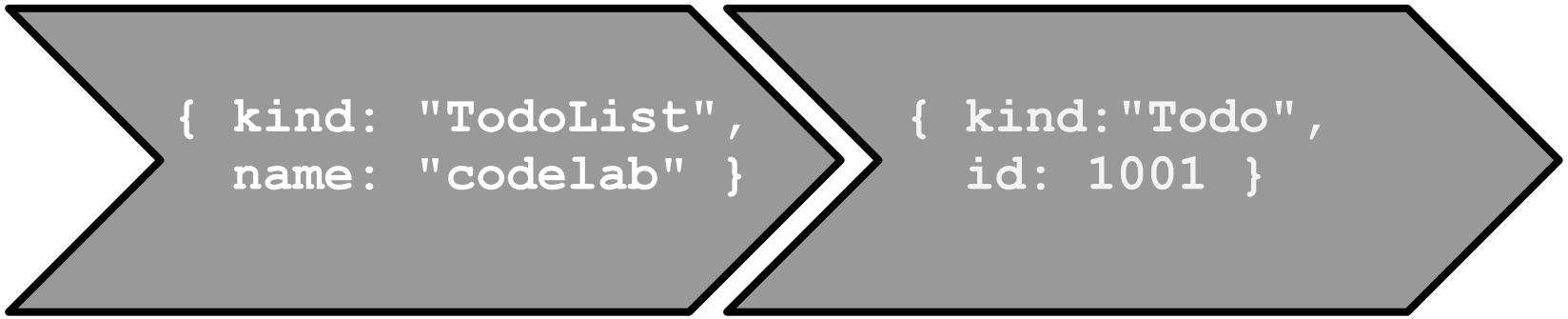
- have a kind and string or numerical identifier
  - numerical identifiers are automatically selected, string identifiers are programmer selected
- have a parent entity (really a parent key, entity may not exist)

Together, these form the entity's *key*



Keys and Entities

```
[ TodoList:codelab, Todo:1001 ]
```



Key	result	title	completed
[MeaningOfLife:1]	{ intValue: 42 }		
[TodoList:codelab, Todo:1001]		{ stringValue: "sit down" }	{ booleanValue: true }
[TodoList:codelab, Todo:2002]		{ stringValue: "register" }	{ booleanValue: true }
[TodoList:codelab, Todo:3003]		{ stringValue: "learn about stuff" }	{ booleanValue: false }

Mutation

insert: [{ key: [Foo:foo] }, properties: { result: "bar" }]

insertAutoId: [{key: [MeaningOfLife] }, properties: { result: 43 }]

delete: [{ key: [MeaningOfLife:1] }]

update: [{ key: [TodoList:codelab, Todo:1003], properties: { title: "learn about stuff", completed: true } }]

Key	result	title	completed
~~[MeaningOfLife:1]~~	~~{ intValue: 42 }~~		
[Foo:foo1]	{ stringValue: "bar" }		
[MeaningOfLife:3]	{ intValue: 43 }		
[TodoList:codelab, Todo:3003]		{ stringValue: "learn about stuff" }	{ booleanValue: true }

RPCs

```
lookup(keys)
blindWrite(mutations)
runQuery(query, kind, filters)
beginTransaction()
commit(transaction, mutations)
rollback(transaction)
allocateIds(keys)
```

JSON API reference

Protocol buffers message definitions

Codelab

Level 0

Send RPCs from Compute Engine

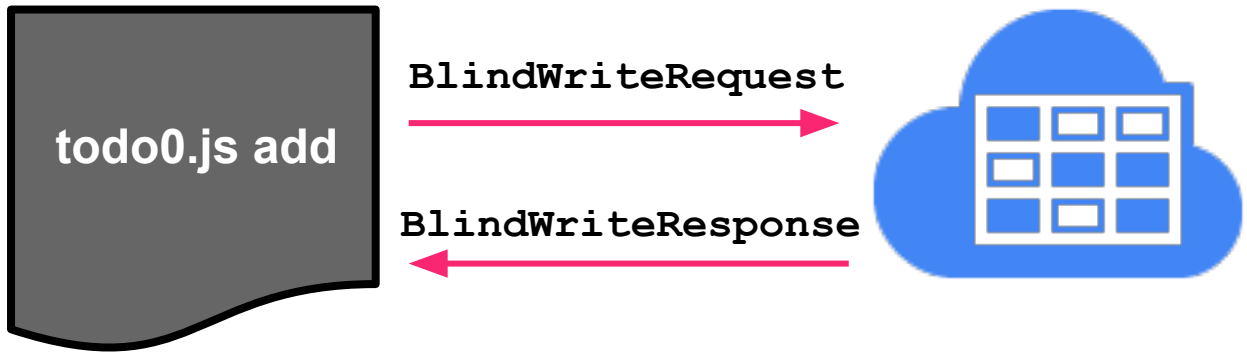
Achievement: Play with `todo0.js` sample app

- `todo0.js <list>` **`add`** `<title>`: adds a new todo entry

```
rpc blindWrite(BlindWriteRequest/Mutation/insertAutoId)-> BlindWriteResponse
```

- `todo0.js <list>` **`get`** `<id>`: reads a todo entry

```
rpc lookup(LookupRequest/Key/id) -> LookupResponse
```



Level 0

Send RPCs from Compute Engine (10 minutes)

1. Go to todo.gcd-codelab.appspot.com, you wil be redirected to todo-gcd.codelab.appspot.com/**uNNNN**
2. Go back to your SSH window (password: **io2013**)

```
$ export LIST=uNNNN
```

```
$ ./todo0.js $LIST add "complete gcd level0"
ID <TODO-ID-0>: complete gcd level0 - TODO
```

3. Go back to todo-gcd.codelab.appspot.com/**uNNNN** and mark your TODO as Done

```
$ ./todo0.js $LIST get <TODO-ID-0>
```

4. Review the implementation in `todo0.js` and **`blindWrite`** and **`lookup`** documentation.

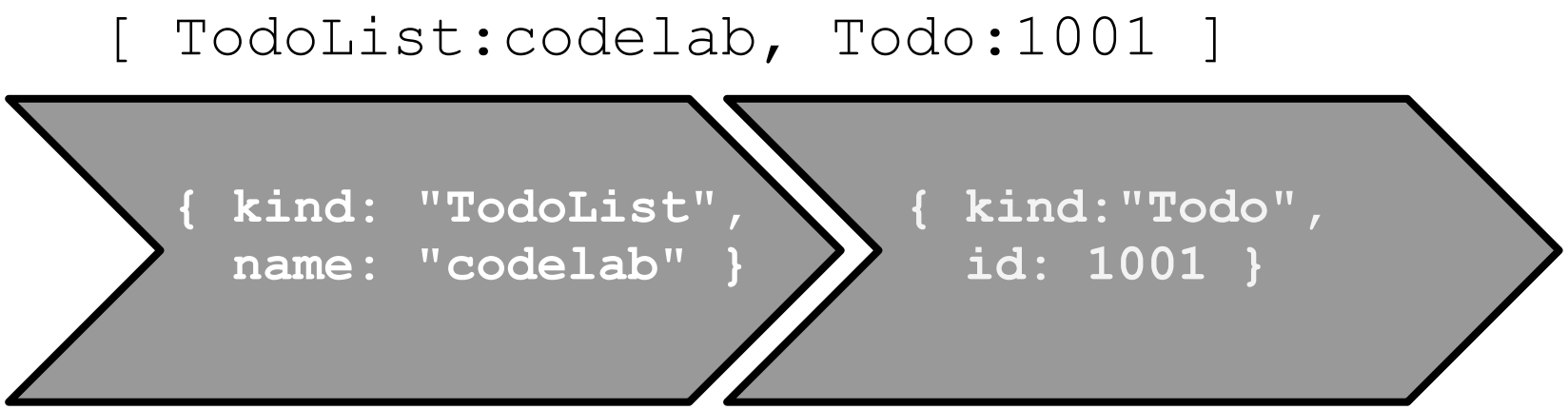
Level 1

Write your first mutation

Achievement: Add a new todo.js command:

- `todo1.js <list> del <id>:` deletes a todo entry

```
rpc blindWrite(BlindWriteRequest/Mutation/delete)-> BlindWriteResponse
```



Level 1

Write your first mutation (15 minutes)

1. Fill the **`FIXMEs`** in `todo1.js` to complete the **`del`** command:
   a. Read **`blindWrite`** rpc request documentation
   b. Review the existing **`get`** command

```
$ ./todo1.js $LIST add "complete gcd level1"
ID <TODO-ID-1>: complete gcd level1 - TODO
```

2. Test your **`del`** command

```
$ ./todo1.js $LIST del <TODO-ID-1>
```

```
$ ./todo1.js $LIST get <TODO-ID-1>
AssertionError: todo <TODO-ID-1> not found
```

```
$ ./test1
PASS
```

3. Review solution in `todo.js`: look for: **`// level 1`**

Level 2

Complete the CLI client

Achievement: Add two new todo.js commands:

- `todo2.js <list> edit <id> <title> <true|false>`: edit a todo entry

```
rpc blindWrite(BlindWriteRequest/Mutation/update)-> BlindWriteResponse
```

Key	result	title	completed
[MeaningOfLife:1]	{ intValue: 42 }		
[TodoList:codelab, Todo:1001]		{ stringValue: "complete level 0" }	{ booleanValue: true }
[TodoList:codelab, Todo:2002]		{ stringValue: "complete level 1" }	{ booleanValue: true }
[TodoList:codelab, Todo:3003]		{ stringValue: "complete level 2" }	{ booleanValue: false }

Level 2

Complete the CLI client

Achievement: Add two new todo.js commands:

- `todo2.js <list>` **`ls`**: lists all todo entries

```
rpc runQuery(RunQueryRequest/kind,filter/ancestor) -> RunQueryResponse
```

  - **`Query: kind Todo`**
  - **`Ancestor filter: entities that match parent key [ TodoList:codelab ]`**

	Kind 1	Name or Id 1	Kind 2	Name or Id 2
Entity 0	MeaningOfLife	1		
Entity 1	TodoList	codelab	Todo	1001
Entity 2	TodoList	codelab	Todo	2001
Entity 3	TodoList	codelab	Todo	3003
Entity 4	TodoList	shopping	Todo	3
Entity 5	TodoList	shopping	Todo	1001

Level 2

Complete the CLI client

Achievement: Add a new todo.js commands:

- `todo2.js <list>` **`ls`**: lists all todo entries

```
rpc runQuery(RunQueryRequest/kind,filter/ancestor) -> RunQueryResponse
```

  - **`Query: kind Todo`**
  - **`Ancestor filter: entities that match parent key [ TodoList:codelab ]`**

	Kind 1	Name or Id 1	Kind 2	Name or Id 2
Entity 0	MeaningOfLife	1		
Entity 1	TodoList	codelab	Todo	1001
Entity 2	TodoList	codelab	Todo	2001
Entity 3	TodoList	codelab	Todo	3003
Entity 4	TodoList	shopping	Todo	3
Entity 5	TodoList	shopping	Todo	1001

Level 2

Complete the CLI client

Achievement: Add a new todo.js commands:

- `todo2.js <list>` **`ls`**: lists all todo entries

```
rpc runQuery(RunQueryRequest/kind,filter/ancestor) -> RunQueryResponse
```

  - **`Query: kind Todo`**
  - **`Ancestor filter: entities that match parent key [ TodoList:codelab ]`**

	Kind 1	Name or Id 1	Kind 2	Name or Id 2
Entity 0	MeaningOfLife	1		
Entity 1	TodoList	codelab	Todo	1001
Entity 2	TodoList	codelab	Todo	2001
Entity 3	TodoList	codelab	Todo	3003
Entity 4	TodoList	shopping	Todo	3
Entity 5	TodoList	shopping	Todo	1001

Level 2

Complete the CLI client (20 minutes)

1. Fill the **`FIXMEs`** in `todo2.js`:
   a. Review the existing **`add`** command
   b. Read **`runQuery`** rpc documentation
2. Test your **`edit`** and **`ls`** commands:

```
$ ./todo2.js $LIST add "complete gcd level2"
ID <TODO-ID-2>: complete gcd level2 - TODO
```

```
$ ./todo2.js $LIST edit <TODO-ID-2> "complete gcd level2" true
```

```
$ ./todo2.js $LIST ls
```

```
$ ./test2
PASS
```

3. Review solution in `todo.js`: look for: **`// level 2`**

Level 3

Add consistency and transactions

Add a new todo.js command:

- `./todo3.js <list> archive`: delete all completed todo entries

Needs transactions and consistent queries to avoid problems with simultaneous uses of **edit** and **archive**:

```
$ ./todo3.js list codelab
ID 3003: complete level3 - DONE
gcduser1 $ ./todo3.js codelab edit 3003 "complete gcd level3" false
ID 3003: complete gcd level 3 - TODO
gcduser2 $ ./todo3.js codelab archive
ID 3003: ARCHIVED
gcduser1 $ todo.js list codelab
gcduser1 $
```

Level 3

Add consistency and transactions

```
gcduser1 $ ./todo3.js codelab edit 3003 "complete gcd level3" false
blindWrite(update, completed=false)
gcduser2 $ ./todo3.js codelab archive
runQuery(completed=true),blindWrite(delete)

2013-05-17 04:44:20.342 runQuery(completed=true)              result := [3003: true]
2013-05-17 04:44:20.351 blindWrite(update, completed=false) [3003: false]
2013-05-17 04:44:20.366 blindWrite(delete)                    delete(3003)
```

Transactional Interlude

By David Gay

Transactions in the Google Cloud Datastore

Google Cloud Datastore transaction guarantees ("ACID"):

- durable
- provide a consistent view of the datastore for the duration of the transaction
- prevent concurrent updates
- choice of snapshot vs serializable isolation

Google Cloud Datastore transaction limitations:

- all mutations must be performed at the end
- must be scoped to a small number (5) of *entity groups*
- limits on the write-rate *per entity group*

Transactions in the Google Cloud Datastore

Google Cloud Datastore transactions and entity groups:

- all operations must be scoped to a small number (5) of *entity groups*

An *entity group* is a set of entities with a common top-level ancestor:

- same entity group: `[ TodoList:codelab, Todo:2002 ]` and `[ TodoList:codelab, SomethingElse:fun ]`
- different entity group: `[ MeaningOfLife:1 ]` and `[ TodoList:shopping, Todo:1002 ]`

	Kind 1	Name or Id 1	Kind 2	Name or Id 2
Entity 0	MeaningOfLife	1		
Entity 1	**TodoList**	**codelab**	Todo	1001
Entity 2	**TodoList**	**codelab**	Todo	2001
Entity 3	**TodoList**	**codelab**	Todo	3003
Entity 4	TodoList	shopping	Todo	3
Entity 5	TodoList	shopping	Todo	1001

Transactions: Scoping Operations, Performance

Scoping reads:

- lookup specifies entity keys: automatically scoped to keys' entity groups
- runQuery: ancestor filter scopes query to ancestor key's entity group
  -> runQuery without ancestor filter is **not allowed** in a transaction
  Aside: non-ancestor queries outside transactions are eventually consistent

Scoping mutations (sent with commit RPC **only**):

- specify entities (with keys) or keys (to delete): automatically scoped to keys' entity groups

Performance:

- read-only transactions: no scaling limits
- transactions with mutations: limit of one (1) transaction per entity-group per second

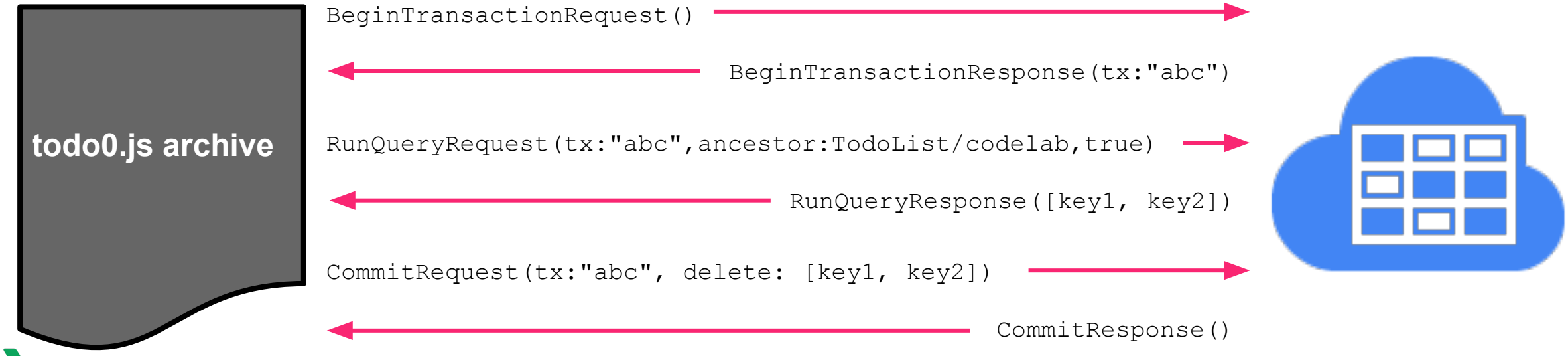
Level 3

Add consistency and transactions

Add a new todo.js command:

- `todo3 <list> archive`: delete all completed todo entries

```
rpc beginTransaction(BeginTransactionRequest) -> BeginTransactionResponse
rpc runQuery(RunQueryRequest/kind,filter/ancestor) -> RunQueryResponse
rpc commit(CommitRequest/mutation) -> CommitResponse
```



Level 3

Add consistency and transactions (25 minutes)

1. Fill the **`FIXMEs`** in `todo3.js`:
   a. Read **`beginTransaction`** and **`runQuery`** rpc documentation
   b. Read **`commit`** rpc documentation
2. Test your **archive** command:

```
$ ./todo3.js $LIST add "complete gcd level3"
ID <TODO-ID-3>: complete gcd level3 - TODO
$ ./todo3.js $LIST edit <TODO-ID-3> "complete gcd level3" true
```

```
$ ./todo3.js $LIST archive
```

```
$ ./todo3.js $LIST ls
```

```
$ ./test3
PASS
```

3. Review solution in `todo.js`: look for: `// level 3`

Level 4

Use a sample helper library

Rewrite the get command to use a sample helper library.

Sample helper implementation is in `todo4_solution.js`.

With the helper, code becomes

- less fragile
- more readable
- more fun to work with

Level 4

Use a sample helper library

With the helper, the code below:

```
key: {
  path: [{ kind: 'TodoList', name: todoListName },
         { kind: 'Todo' }]
}
```

Level 4

Use a sample helper library

becomes a little bit more readable:

```
key: new Key('TodoList', todoListName, 'Todo'),
```

Level 4

Use a sample helper library

And the code below:

```
      mutation: {
        insertAutoId: [{
          key: {
            path: [{ kind: 'TodoList', name: todoListName },
                   { kind: 'Todo' }]
          },
          properties: {
            title: { values: [{ stringValue: title }] },
            completed: { values: [{ booleanValue: false }] }
          }
        }]
      }
```

Level 4

Use a sample helper library

becomes more readable and less fragile:

```
var mutation = new MutationBuilder()
    .insertAutoId(
      new Key('TodoList', todoListName, 'Todo'),
      {title: {stringValue: title}},
      {completed: [{booleanValue: false}]}
    ).build();
```

Level 4

Use a sample helper library

The helper allows methods to be chained:

```
var mutation = new MutationBuilder()
    .insertAutoId(
      new Key('TodoList', todoListName, 'Todo'),
      {title: {stringValue: title}},
      {completed: [{booleanValue: false}]}
    ).insertAutoId(
      new Key('TodoList', todoListName, 'Todo'),
      {title: {stringValue: anotherTitle}},
      {completed: [{booleanValue: false}]}
    ).delete(
      new Key('TodoList', todoListName, 'Todo', 2)
    ).build();
```

Level 4

Use a sample helper library (10 mins)

Rewrite the get command to use a sample helper library.

1. Fill **`FIXME`** in `todo4.js` to complete the **`Key`** helper implementation
2. Test the revised **`add`** command:

```
$ ./todo4.js $LIST add "complete gcd level4"
ID <TODO-ID-4>: complete gcd level4 - TODO
```

3. Fill **`FIXME`** in `todo4.js` to complete the get command with the **`Key`** helper
4. Test the revised **get** command:

```
$ ./todo4.js $LIST get <TODO-ID-4>
```

5. Review the solutions in `todo4_solution.js`. Note that **edit** and **del** command are also rewritten with the helper.

Level 4 - Homework

Build your own!

Get inspired by the helper?
Please build your own helper/client library on top of the low level JSON API

Bonus

Go live on GCE w/ socketstream (20 minutes)

Fork TodoMVC socketstream demo and add a GCD backend

In `~/gcd-codelab/bonus`

1. Run the app on your todo list:

```
$ npm install
```

```
$ node app.js $LIST
```

2. Fill the **`FIXMEs`** in `server/rpc/todos.js`
3. Restart the app between edits:

```
$ node app.js $LIST
listening on http://173.255.120.243:5XXXX
```

4. Review solution in `server/rpc/solution.js`

Take away

RTFM developers.google.com/datastore

FORK github.com/.../google-cloud-datastore

github.com/.../datastore-codelab-nodejs

JOIN gcd-discuss@

{ key: "Thank", name: "You" }

developers.google.com/datastore

github.com/GoogleCloudPlatform/google-cloud-datastore

gcd-discuss@googlegroups.com


Google
Developers