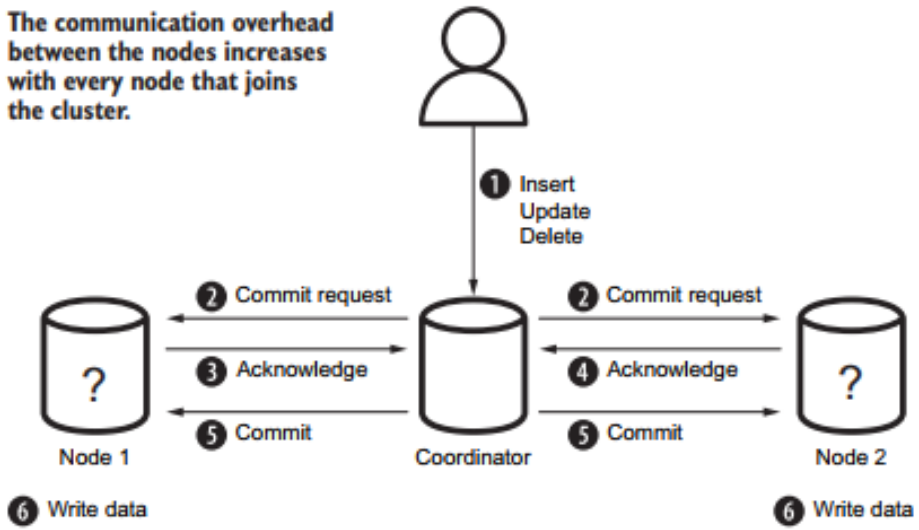
Lesson 9 DynamoDB

Michael Yang

Why NoSQL?

- *Vertically*—You can use faster hardware for your database machine; for example, you can add memory or replace the CPU with a more powerful model. *Horizontally*—You can add a second database machine. Both machines then form a *database cluster*.
- Relationanl Database, two phase commit



What is DynamoDB

- There are four types of NoSQL databases—document, graph, columnar, and key-value store—each with its own uses and applications. Amazon provides a NoSQL database service called *DynamoDB*, a key-value store.
- DynamoDB is a fully managed, proprietary, closed source key-value store with document support. In other words, DynamoDB persists objects identified by a unique key, which you might know from the concept of a hash table.
- DynamoDB is highly available and highly durable. You can scale from one item to billions and from one request per second to tens of thousands of requests per second.

Use Cases

- Some typical use cases for DynamoDB follow:

1. When building systems that need to deal with a massive amount of requests or spiky workloads, the ability to scale horizontally is a game changer. We have used DynamoDB to track client-side errors from a web application, for example.

2. When building small applications with a simple data structure, the pay-perrequest pricing model and the simplicity of a fully managed service are good reasons to go with DynamoDB. For example, we used DynamoDB to track the progress of batch jobs.

Example – A To-do-list Application

- “nodetodo” uses DynamoDB as a database and comes with the following features:

1 Creates and deletes users
2 Creates and deletes tasks
3 Marks tasks as done
4 Gets a list of all tasks with various filters

- You’ll implement these commands for “nodetodo” app. This listing shows the full CLI description of all the commands, including parameters.

```
nodetodo

Usage:
  nodetodo user-add <uid> <email> <phone>
  nodetodo user-rm <uid>
  nodetodo user-ls [--limit=<limit>] [--next=<id>]
  nodetodo user <uid>
  nodetodo task-add <uid> <description> \
➥ [<category>] [--dueat=<yyyymmdd>]
  nodetodo task-rm <uid> <tid>
  nodetodo task-ls <uid> [<category>] \
➥ [--overdue|--due|--withoutdue|--futuredue]
  nodetodo task-la <category> \
➥ [--overdue|--due|--withoutdue|--futuredue]
  nodetodo task-done <uid> <tid>
  nodetodo -h | --help
  nodetodo --version

Options:
  -h --help         Show this screen.
  --version         Show version.
```

The named parameters limit and next are optional.

The category parameter is optional.

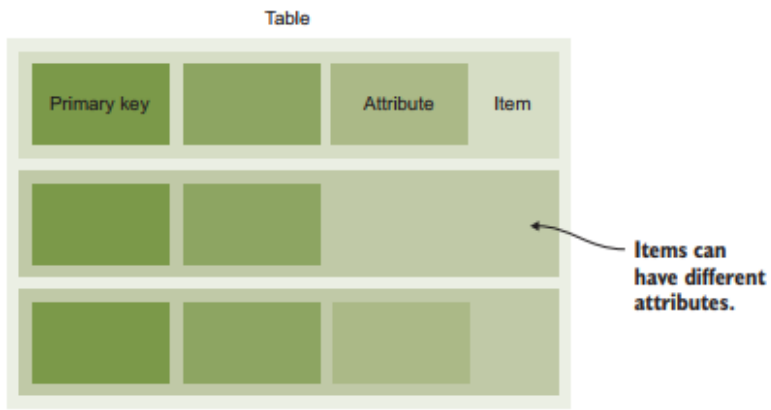
Pipe indicates either/or.

Version information

help prints information about how to use nodetodo.

Example- Creating Tables

- An *item* is a collection of attributes
- an *attribute* is a name-value pair. The attribute value can be scalar (number, string, binary, Boolean), multivalued (number set, string set, binary set), or a JSON document (object, array). Items in a table aren't required to have the same attributes; there is no enforced schema.
- DynamoDB doesn't need a static schema like a relational database does, but you must define the attributes that are used as the primary key in your table. In other words, you must define the table's primary key schema.



DynamoDB tables store items consisting of attributes identified by a primary key.

“Users” Table

- Table name is “todo-user”
- A primary key consists of **one or two** attributes. A primary key is unique within a table and identifies an item.
- When using a single attribute as primary key, DynamoDB calls this the ***partition key*** of the table.

Create DynamoDB table

- The `aws dynamodb create-table` command has the following four mandatory options:

`table-name`—Name of the table (can't be changed)

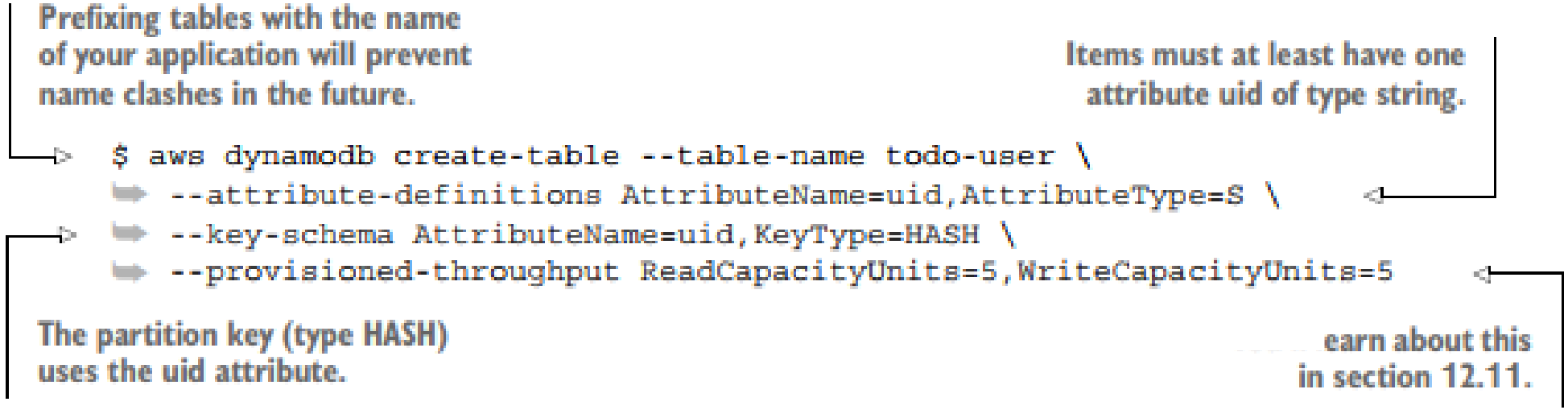
`attribute-definitions`—Name and type of attributes used as the primary key. Multiple definitions can be given using the syntax `AttributeName=attr1,AttributeType=S`, separated by a space character. Valid types are `S` (string), `N` (number), and `B` (binary).

`key-schema`—Name of attributes that are part of the primary key (can't be changed). Contains a single entry using the syntax `AttributeName=attr1, KeyType=HASH` for a partition key, or two entries separated by spaces for a partition key and sort key. Valid types are `HASH` and `RANGE`.

`provisioned-throughput`—Performance settings for this table defined as `ReadCapacityUnits=5,WriteCapacityUnits=5` (you'll learn about this later).

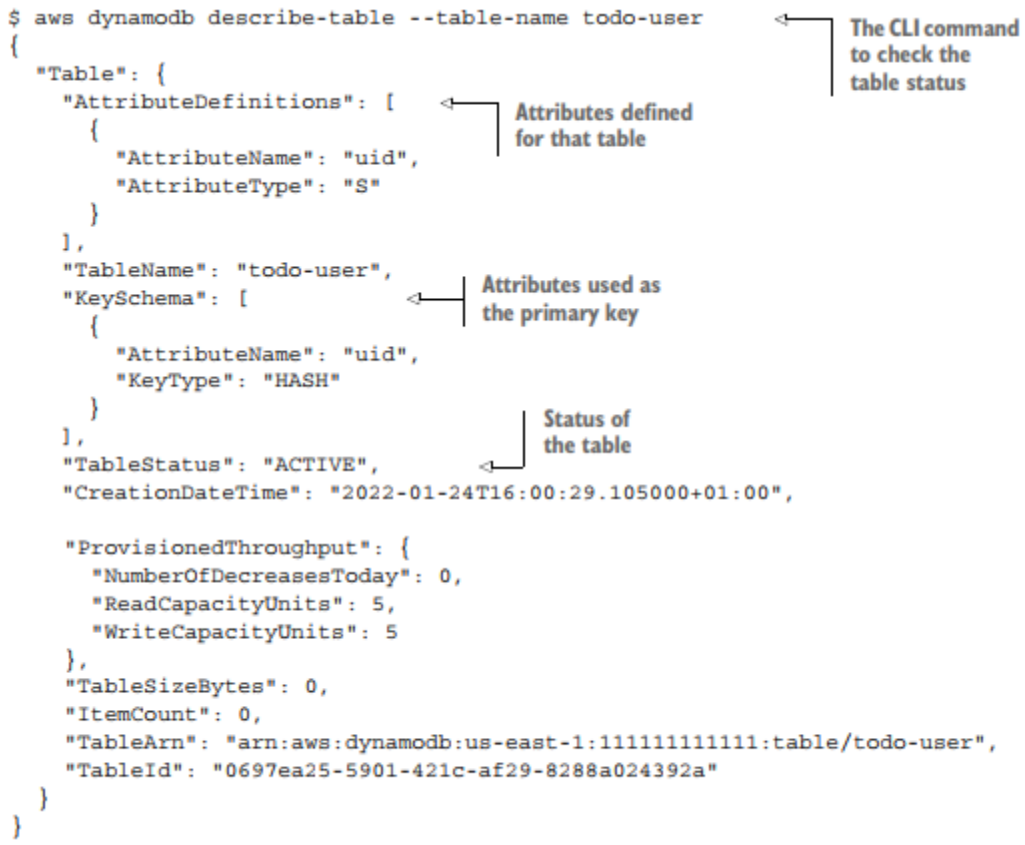
AWS CLI – create-table

- Execute the following command to create the `todo-user` table with the `uid` attribute as the partition key:

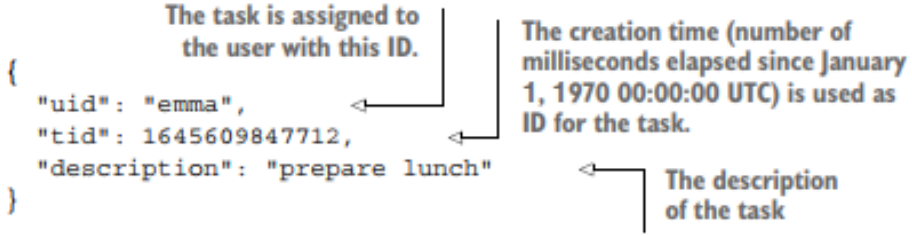


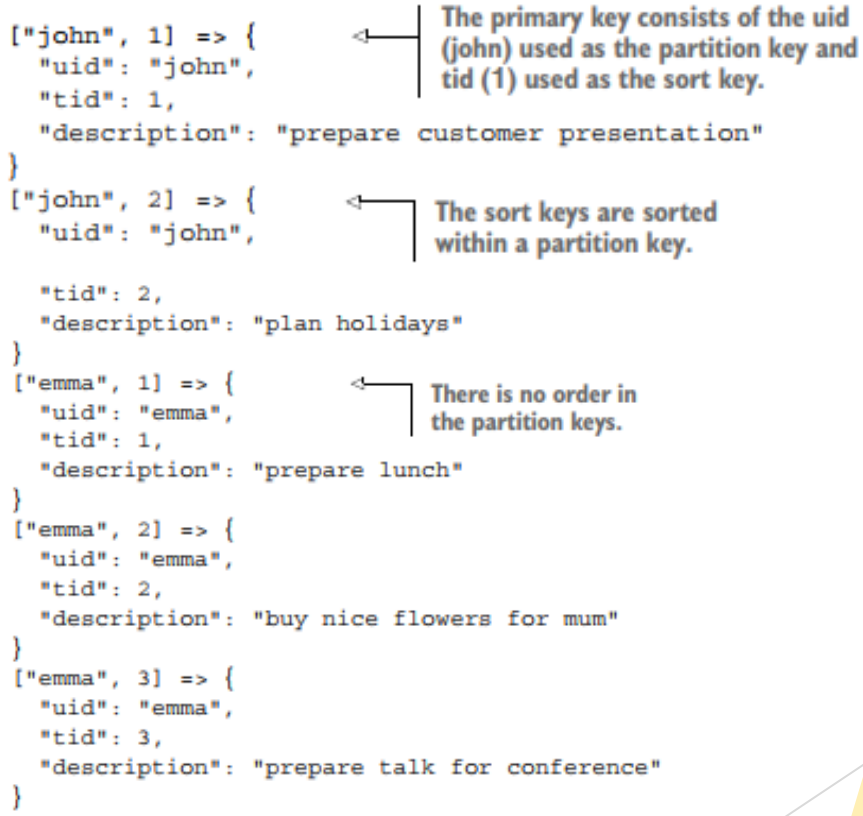
AWS CLI – check status

- Creating a table takes some time. Wait until the status changes to `ACTIVE`. You can check the status of a table as follows



“Tasks” Table


The task is assigned to the user with this ID.
The creation time (number of milliseconds elapsed since January 1, 1970 00:00:00 UTC) is used as ID for the task.
The description of the task
{
"uid": "emma",
"tid": 1645609847712,
"description": "prepare lunch"
}


The primary key consists of the uid (john) used as the partition key and tid (1) used as the sort key.
The sort keys are sorted within a partition key.
There is no order in the partition keys.
["john", 1] => {
"uid": "john",
"tid": 1,
"description": "prepare customer presentation"
}
["john", 2] => {
"uid": "john",
"tid": 2,
"description": "plan holidays"
}
["emma", 1] => {
"uid": "emma",
"tid": 1,
"description": "prepare lunch"
}
["emma", 2] => {
"uid": "emma",
"tid": 2,
"description": "buy nice flowers for mum"
}
["emma", 3] => {
"uid": "emma",
"tid": 3,
"description": "prepare talk for conference"
}

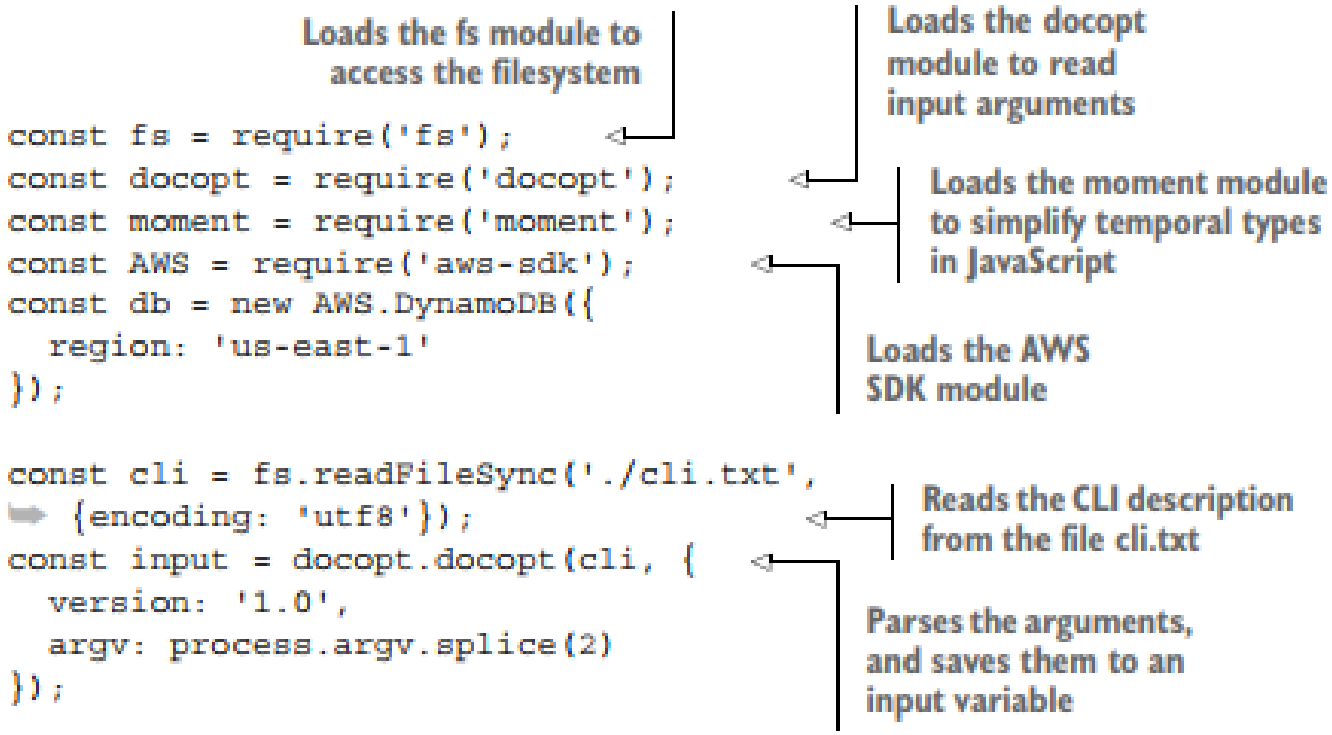
AWS CLI – create table

At least two attributes are needed for a partition key and sort key.

```
$ aws dynamodb create-table --table-name todo-task \
    --attribute-definitions AttributeName=uid,AttributeType=S \
    AttributeName=tid,AttributeType=N \
    --key-schema AttributeName=uid,KeyType=HASH \
    AttributeName=tid,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```

The tid attribute is the sort key.

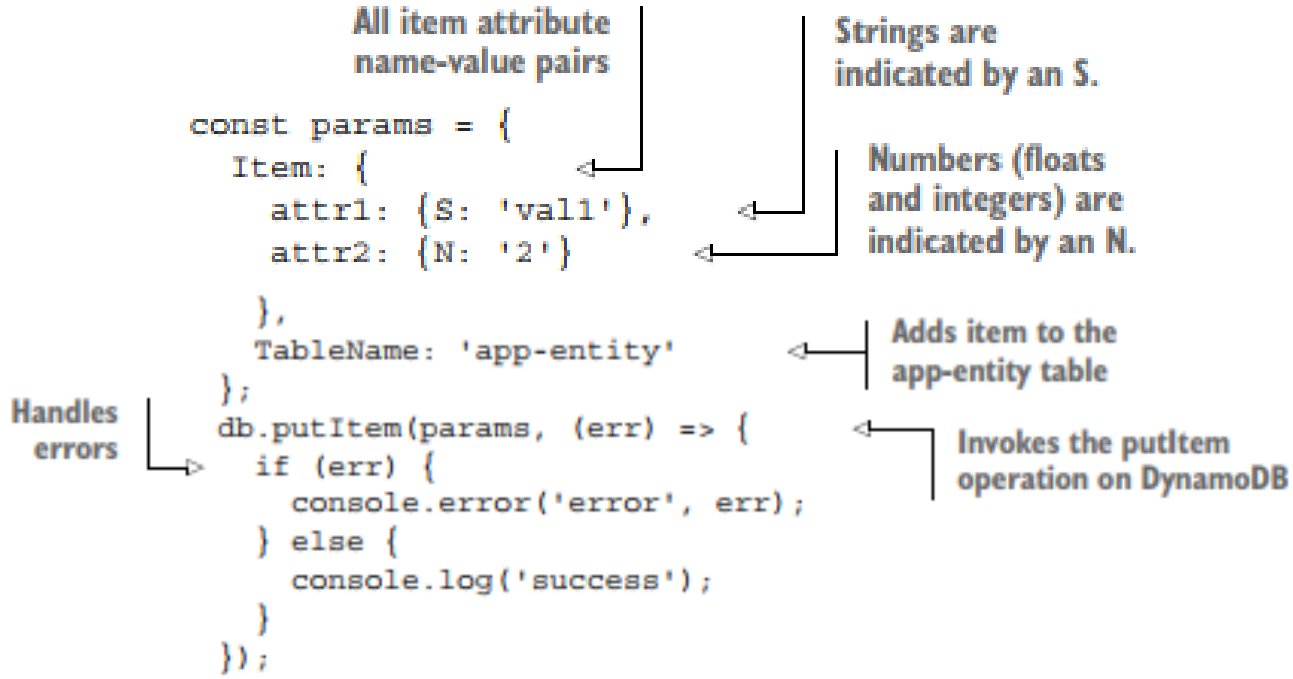
Programming to add items



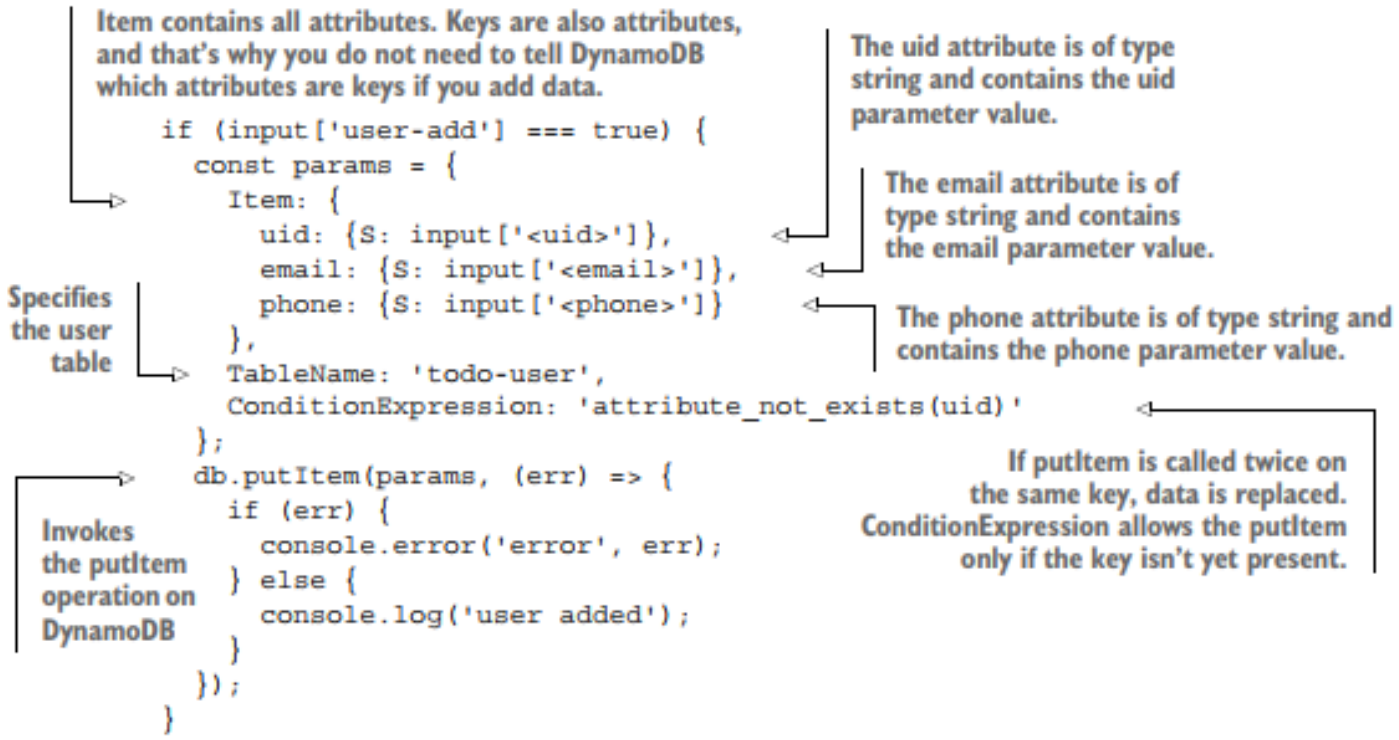

```
const fs = require('fs');
const docopt = require('docopt');
const moment = require('moment');
const AWS = require('aws-sdk');
const db = new AWS.DynamoDB({
  region: 'us-east-1'
});

const cli = fs.readFileSync('./cli.txt',
➥ {encoding: 'utf8'});
const input = docopt.docopt(cli, {
  version: '1.0',
  argv: process.argv.splice(2)
});
```

AWS SDK – How to add items



AWS SDK – Add items to “users” table

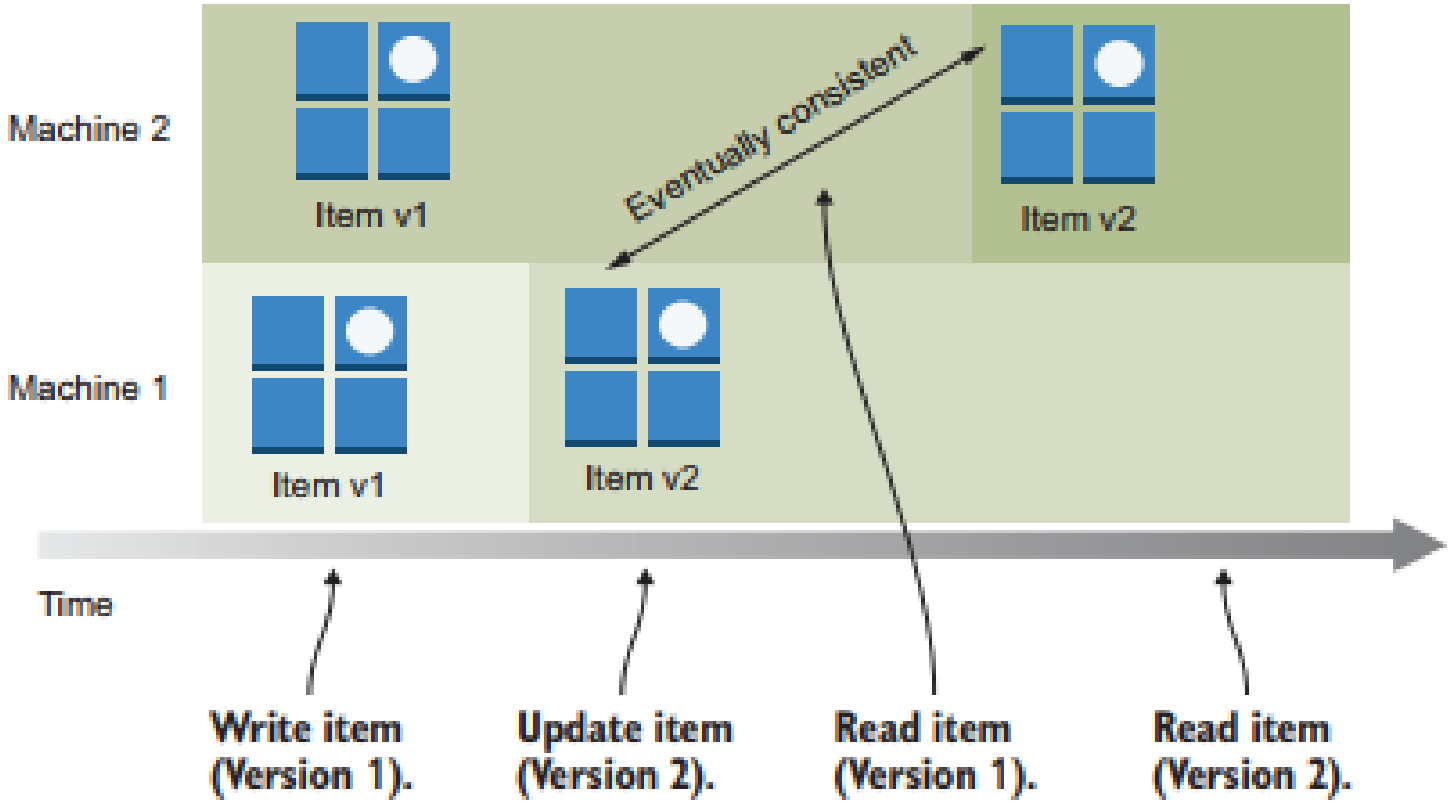



```
if (input['user-add'] === true) {
  const params = {
    Item: {
      uid: {S: input['<uid>']},
      email: {S: input['<email>']},
      phone: {S: input['<phone>']}
    },
    TableName: 'todo-user',
    ConditionExpression: 'attribute_not_exists(uid)'
  };
  db.putItem(params, (err) => {
    if (err) {
      console.error('error', err);
    } else {
      console.log('user added');
    }
  });
}
```

Execute the following commands to add two users:

```
node index.js user-add john john@widdix.de +11111111
node index.js user-add emma emma@widdix.de +22222222
```

Eventually consistent



Eventually consistent reads can return old values after a write operation until the change is propagated to all machines.

PartiQL

The command execute-statement supports PartiQL statements as well.

```
$ aws dynamo3db execute-statement \
➥ --statement "SELECT * FROM \""todo-task\""
```

A simple SELECT statement to fetch all attributes of all items from table todo-task. The escaped " is required because the table name includes a hyphen.

```
$ aws dynamodb execute-statement --statement \
➥ "SELECT * FROM \""todo-task\".\"category-index\""
➥ WHERE category = 'shopping'"
```

```
aws dynamodb execute-statement --statement \
➥ "Update \"todo-user\" SET phone='+33333333' WHERE uid='emma'"
```

DynamoDB Local

- Don’t run DynamoDB Local in production! It’s only made for development purposes and provides the same functionality as DynamoDB, but it uses a different implementation: only the API is the same.

- If you looking for a graphical user interface to interact with DynamoDB, Check out NoSQL Workbench for DynamoDB at http://mng.bz/mJ5P. The tool allows you to create data models, analyze data, and import and export data.

Operating

- DynamoDB isn’t software you can download. Instead, it’s a NoSQL database as a service.
- DynamoDB runs on a fleet of machines operated by AWS.
- DynamoDB replicates your data among multiple machines and across multiple data centers.

Scaling

- a DynamoDB table has two different read/write capacity modes:

On-demand mode adapts the read and write capacity automatically.

Provisioned mode requires you to configure the read and write capacity upfront.

Comparing pricing of DynamoDB on-demand and provisioned mode

Throughput	On-demand mode	Provisioned mode
10 writes per second	$32.85 per month	$4.68 per month
100 reads per second	$32.85 per month	$4.68 per month

Scaling

- Capacity Units

```
$ aws dynamodb get-item --table-name todo-user \
   --key '{"uid": {"S": "emma"}}' \
   --return-consumed-capacity TOTAL \
   --query "ConsumedCapacity"
{
    "CapacityUnits": 0.5,
    "TableName": "todo-user"
}
```

Tells DynamoDB to return the used capacity units

getItem requires 0.5 capacity units.

```
$ aws dynamodb get-item --table-name todo-user \
   --key '{"uid": {"S": "emma"}}' \
   --consistent-read --return-consumed-capacity TOTAL \
   --query "ConsumedCapacity"
{
    "CapacityUnits": 1.0,
    "TableName": "todo-user"
}
```

A consistent read ...

... needs twice as many capacity units.

Networking

- DynamoDB does not run in your VPC. It is accessible via an API. You need internet connectivity to reach the DynamoDB API.