BPA用户指南MongoDB升级v5.1

简介

BPA v4.1.2包括升级到MongoDB v7，因为MongoDB v5即将停产。本文档提供有关在微服务中升级MongoDB版本的详细信息。

更新版本

在microservices中更新以下包版本：

- "mongodb":"^3.1.13"到"mongodb":"^6.8.0"
- “mongoose”：“5.8.7”到“mongoose”："^8.5.1"

折旧变量

MongoDB v7中不建议使用以下变量：

- ssl
- useNewUrlParser
- 使用统一拓扑
- useFindAndModify
- useCreateIndex
- sslValidate

将ssl替换为tls，如下例所示：

```
if (process.env.MONGO_SSL == "true") {
  //options.ssl = true;
  options.tls = true;
  options.tlsAllowInvalidCertificates = true;
}
```

删除回叫

MongoDB查询中的回调在MongoDB v7中已弃用。更新的查询以绿色显示在下方。

```
    deviceCredentials.find(query, ['name', 'credential_id', 'credential_type', 'description'], function (error, credentials) {
    deviceCredentials.find(query, ['name', 'credential_id', 'credential_type', 'description']).then(credentials => {
        let result = [];
```

更新的查询

更新删除查询

执行删除查询后，响应变量应使用deletedCount代替n(例如，用result.deletedCount替换result.n)。

```
  if (!result.n) {
  if (!result.deletedCount) {
    return res.status(404).json(errorHandlerObject.errorHandler(false, false, errorString, 404)).end('');
  }
```

替换result.n

更新查询更改

执行更新操作后，响应变量应使用modifiedCount代替n/nModified(例如，按如下所示将devices.n / devices.nModified替换为devices.modifiedCount)。

```
  if (devices && devices.n && devices.n > 0) {
  if (devices && devices.modifiedCount && devices.modifiedCount > 0) {
    response.success.push({ name: element.name, message: UPDATE_ASSETS.SUCCESS });
  } else {
```

替换device.n或devices.n已修改

更新db.addUser()函数

用db.command(createUse:“username”)，如下例所示。

```
migration_db.addUser(process.env.MONGODB_USER, process.env.MONGODB_PASSWORD, {
  roles: [{
    role: 'readWrite',
    db: database_name
  }]
migration_db.command({ // in mongodb 6.x addUser function removed
  createUser: process.env.MONGODB_USER,
  pwd: process.env.MONGODB_PASSWORD,
  roles: [ { role: 'readWrite', db: database_name } ]
}). then( (result) => {
```

替换db.addUser()

更新方法

如下所述，更新MongoDB v7中不建议使用的以下方法。

已弃用的方法	新方法
update（更新）	updateOne或updateMany
删除	delete或delete许多
计数	计数文档
findOneAndRemove	findOneAndDelete

更新GridFS

在MongoDB v7中，GridFSBucket类用于与GridFS交互，GridFS是用于存储和检索大型文件的规范。GridFSBucket类提供在GridFS中上载、下载和管理文件的方法。

通过从MongoDB导入GridFSBucket替换gridfs-stream。请参阅以下图像查看更改。

```
const Gridfs = require('gridfs-stream');
const { GridFSBucket } = require('mongodb');
```

替换gridfs-stream

```
let dbConn = mongoose.connection.db;
let dbDriver = mongoose.mongo;
let gridFs = new Gridfs(dbConn, dbDriver);
let readStream = gridFs.createReadStream({ _id: pdfrefId });
let gridFs = new GridFSBucket(dbConn);
let fileId = new mongoose.Types.ObjectId(pdfrefId);
let readStream = gridFs.openDownloadStream(fileId);
```

示例 1

```
let dbConn = mongoose.connection.db;
let dbDriver = mongoose.mongo;
let gridFs = new Gridfs(dbConn, dbDriver);
let gridFs = new GridFSBucket(dbConn);

let writeStream = gridFs.createWriteStream({
    filename: fileName,
    mode: 'w',
    content_type: contentType
let writeStream = gridFs.openUploadStream(fileName, {
    contentType: contentType
});
fs.createReadStream(fileName).pipe(writeStream);
writeStream.on('close', async (file) => {

writeStream.on('finish', async (file) => {

if (storeType === 'pdf') {
    reportObj.pdfrefId = file._id;
    reportObj.pdfrefId = writeStream.id;
    reportObj.pdfGenerationState = PDF_GENERATE_STATE.COMPLETED;
} else reportObj.htmlrefId = file._id;
} else reportObj.htmlrefId = writeStream.id;

});
```

示例 2

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。