# Running a MongoDB Database in Kubernetes with StatefulSets

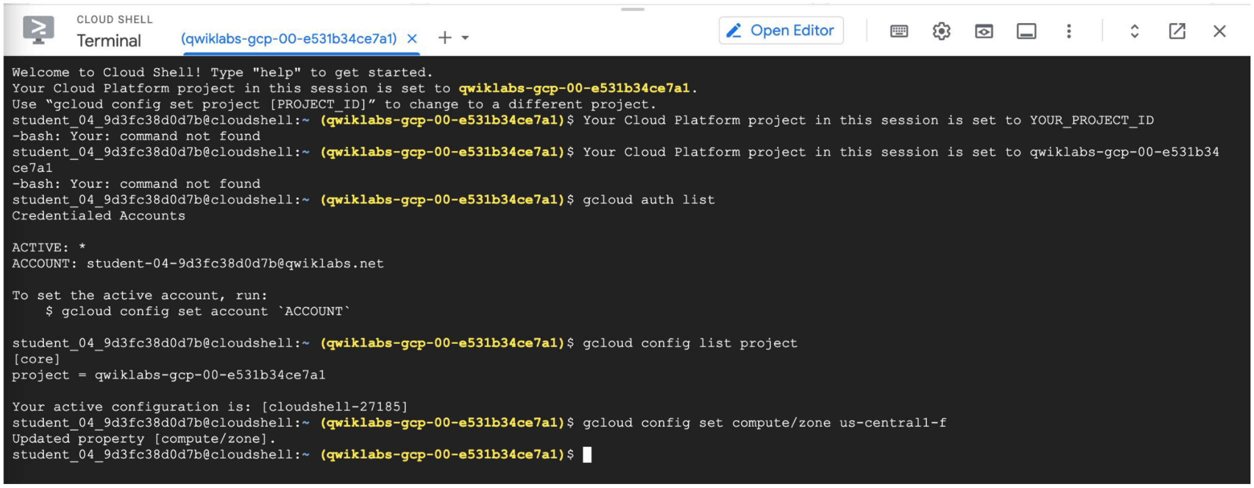
MongoDB is a widely used document-oriented NoSQL database that can be deployed in a containerized environment like Kubernetes. In this report, we will discuss the steps involved in running a MongoDB database in Kubernetes using StatefulSets in Qwiklab.

Steps:
Qwiklab provides a cloud-based platform to learn and practice various technical skills. To run a MongoDB database in Kubernetes using StatefulSets, we used the following steps:
Created a Kubernetes cluster using the Google Cloud Platform (GCP) Console and connected to it using the Cloud Shell.
Deployed a single-node MongoDB replica set using Kubernetes StatefulSets and Services.
Configured the MongoDB database for authentication and created a user with admin privileges.
Tested the MongoDB database by connecting to it using the mongo shell.

We were able to successfully deploy a MongoDB database in Kubernetes using StatefulSets and Services in Qwiklab. We configured the database for authentication and created a user with admin privileges. We also tested the database by connecting to it using the mongo shell.

Conclusion:
Running a MongoDB database in Kubernetes using StatefulSets is a useful skill for anyone working in a containerized environment. Qwiklab provides an excellent platform to learn and practice these skills, and we were able to successfully deploy a MongoDB database in Kubernetes using StatefulSets and Services.

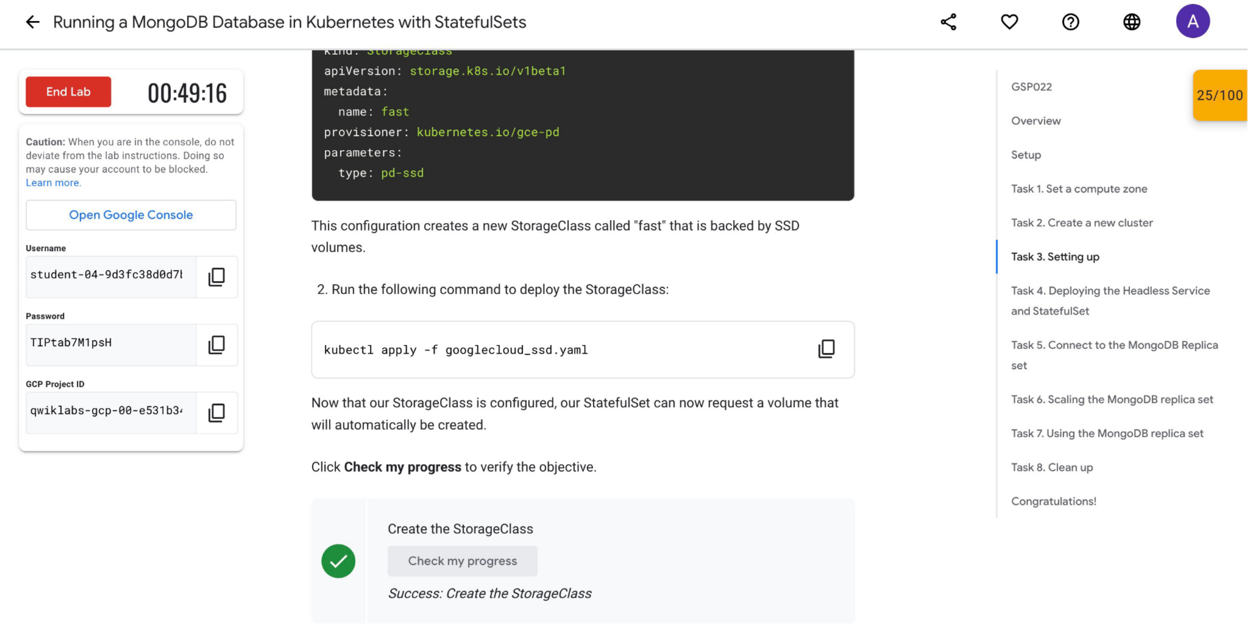
Task 1: Set a compute zone



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-00-e531b34ce7a1.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
student_04_9d3fc38d0d7b@cloudshell:~ (qwiklabs-gcp-00-e531b34ce7a1)$ Your Cloud Platform project in this session is set to YOUR_PROJECT_ID
-bash: Your: command not found
student_04_9d3fc38d0d7b@cloudshell:~ (qwiklabs-gcp-00-e531b34ce7a1)$ Your Cloud Platform project in this session is set to qwiklabs-gcp-00-e531b34
ce7a1
-bash: Your: command not found
student_04_9d3fc38d0d7b@cloudshell:~ (qwiklabs-gcp-00-e531b34ce7a1)$ gcloud auth list
Credentialed Accounts

ACTIVE: *
ACCOUNT: student-04-9d3fc38d0d7b@qwiklabs.net

To set the active account, run:
    $ gcloud config set account `ACCOUNT`

student_04_9d3fc38d0d7b@cloudshell:~ (qwiklabs-gcp-00-e531b34ce7a1)$ gcloud config list project
[core]
project = qwiklabs-gcp-00-e531b34ce7a1

Your active configuration is: [cloudshell-27185]
student_04_9d3fc38d0d7b@cloudshell:~ (qwiklabs-gcp-00-e531b34ce7a1)$ gcloud config set compute/zone us-central1-f
Updated property [compute/zone].
student_04_9d3fc38d0d7b@cloudshell:~ (qwiklabs-gcp-00-e531b34ce7a1)$
```

Task 2: Create a new cluster

## Task 3: Setting up


Running a MongoDB Database in Kubernetes with StatefulSets
A
End Lab
00:49:16
Open Google Console
GSP022
25/100
Overview
Setup
Task 1. Set a compute zone
Task 2. Create a new cluster
Task 3. Setting up
Task 4. Deploying the Headless Service and StatefulSet
Task 5. Connect to the MongoDB Replica set
Task 6. Scaling the MongoDB replica set
Task 7. Using the MongoDB replica set
Task 8. Clean up
Congratulations!

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username: student-04-9d3fc38d0d7l

Password: TIPtab7M1psH

GCP Project ID: qwiklabs-gcp-00-e531b3

```
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```

This configuration creates a new StorageClass called "fast" that is backed by SSD volumes.

2. Run the following command to deploy the StorageClass:

```
kubectl apply -f googlecloud_ssd.yaml
```

Now that our StorageClass is configured, our StatefulSet can now request a volume that will automatically be created.

Click **Check my progress** to verify the objective.

Create the StorageClass

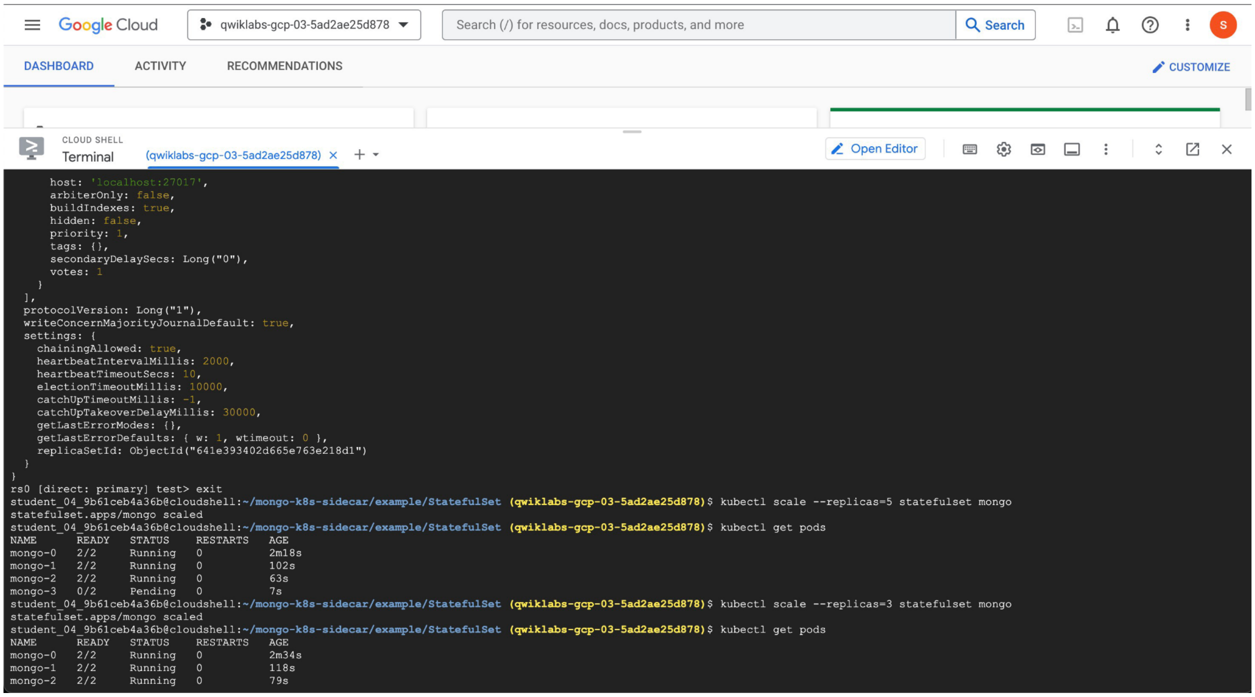
Check my progress

*Success: Create the StorageClass*

## Task 4: Deploying the headless Service and StatefulSet

## Task 5: Connect to the MongoDB Replica set

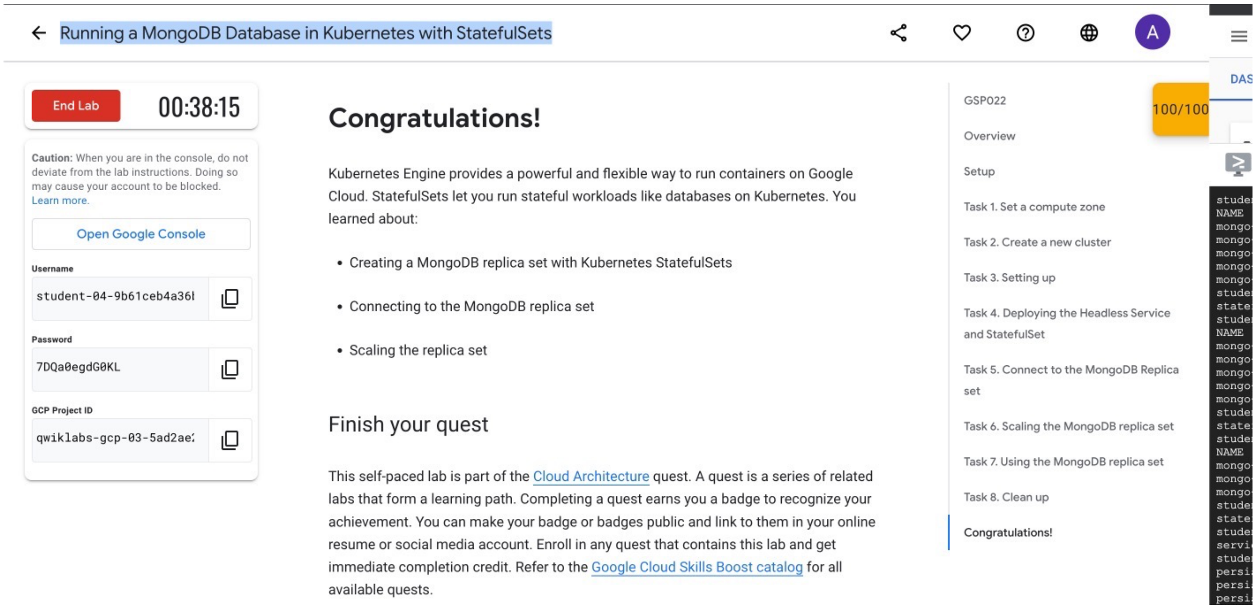
## Task 6: Scaling the MongoDB replica set


Google Cloud
qwiklabs-gcp-03-5ad2ae25d878
Search (/) for resources, docs, products, and more
Search
S
DASHBOARD
ACTIVITY
RECOMMENDATIONS
CUSTOMIZE
CLOUD SHELL
Terminal
(qwiklabs-gcp-03-5ad2ae25d878)
Open Editor

```
      host: 'localhost:27017',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    }
  ],
  protocolVersion: Long("1"),
  writeConcernMajorityJournalDefault: true,
  settings: {
    chainingAllowed: true,
    heartbeatIntervalMillis: 2000,
    heartbeatTimeoutSecs: 10,
    electionTimeoutMillis: 10000,
    catchUpTimeoutMillis: -1,
    catchUpTakeoverDelayMillis: 30000,
    getLastErrorModes: {},
    getLastErrorDefaults: { w: 1, wtimeout: 0 },
    replicaSetId: ObjectId("641e393402d665e763e218d1")
  }
}
rs0 [direct: primary] test> exit
student_04_9b61ceb4a36b@cloudshell:~/mongo-k8s-sidecar/example/StatefulSet (qwiklabs-gcp-03-5ad2ae25d878)$ kubectl scale --replicas=5 statefulset mongo
statefulset.apps/mongo scaled
student_04_9b61ceb4a36b@cloudshell:~/mongo-k8s-sidecar/example/StatefulSet (qwiklabs-gcp-03-5ad2ae25d878)$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
mongo-0   2/2     Running   0          2m18s
mongo-1   2/2     Running   0          102s
mongo-2   2/2     Running   0          63s
mongo-3   0/2     Pending   0          7s
student_04_9b61ceb4a36b@cloudshell:~/mongo-k8s-sidecar/example/StatefulSet (qwiklabs-gcp-03-5ad2ae25d878)$ kubectl scale --replicas=3 statefulset mongo
statefulset.apps/mongo scaled
student_04_9b61ceb4a36b@cloudshell:~/mongo-k8s-sidecar/example/StatefulSet (qwiklabs-gcp-03-5ad2ae25d878)$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
mongo-0   2/2     Running   0          2m34s
mongo-1   2/2     Running   0          118s
mongo-2   2/2     Running   0          79s
```

## Task 7: Using MongoDB replica set

Task 8: Clean Up
Congratulations:



Running a MongoDB Database in Kubernetes with StatefulSets

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username: student-04-9b61ceb4a36l

Password: 7DQa0egdG0KL

GCP Project ID: qwiklabs-gcp-03-5ad2ae:

## Congratulations!

Kubernetes Engine provides a powerful and flexible way to run containers on Google Cloud. StatefulSets let you run stateful workloads like databases on Kubernetes. You learned about:

- Creating a MongoDB replica set with Kubernetes StatefulSets
- Connecting to the MongoDB replica set
- Scaling the replica set

### Finish your quest

This self-paced lab is part of the Cloud Architecture quest. A quest is a series of related labs that form a learning path. Completing a quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. Enroll in any quest that contains this lab and get immediate completion credit. Refer to the Google Cloud Skills Boost catalog for all available quests.

GSP022
100/100
Overview
Setup
Task 1. Set a compute zone
Task 2. Create a new cluster
Task 3. Setting up
Task 4. Deploying the Headless Service and StatefulSet
Task 5. Connect to the MongoDB Replica set
Task 6. Scaling the MongoDB replica set
Task 7. Using the MongoDB replica set
Task 8. Clean up
Congratulations!
DAS

```
stude
NAME
mongo
mongo
mongo
mongo
mongo
stude
state
stude
NAME
mongo
mongo
mongo
mongo
mongo
stude
state
stude
NAME
mongo
mongo
mongo
stude
state
stude
servi
stude
persi
persi
persi
```

**For MongoDB, include your understanding of a headless service, StatefulSet, and why running a DB on a container is not a contradictory task**

In Kubernetes, a headless service is one that does not have a cluster IP allocated to it. Instead, it returns the IP addresses of the pods with which it is related. Headless services are utilized in StatefulSets to offer pods with stable network identities. A headless service is utilized in the case of a MongoDB deployment utilizing StatefulSets to ensure that each pod in the StatefulSet has a unique DNS entry that remains stable during the pod's lifespan.

StatefulSets are a sort of Kubernetes workload that assures the ordering and uniqueness of newly produced pods. They are helpful for stateful applications like databases because they ensure that each pod has a unique network identity that does not change even if the pod is rescheduled or removed.

Running a database in a container is not an incompatible task because containers offer a lightweight, portable, and scalable way to package and deploy applications. Databases, such as MongoDB, may be packaged as container images and deployed in a containerized environment such as Kubernetes. Containerization has various advantages, including isolation, repeatability, and scalability. Furthermore, operating a database in a containerized environment like Kubernetes enables more effective resource use and simplifies database infrastructure management as code.

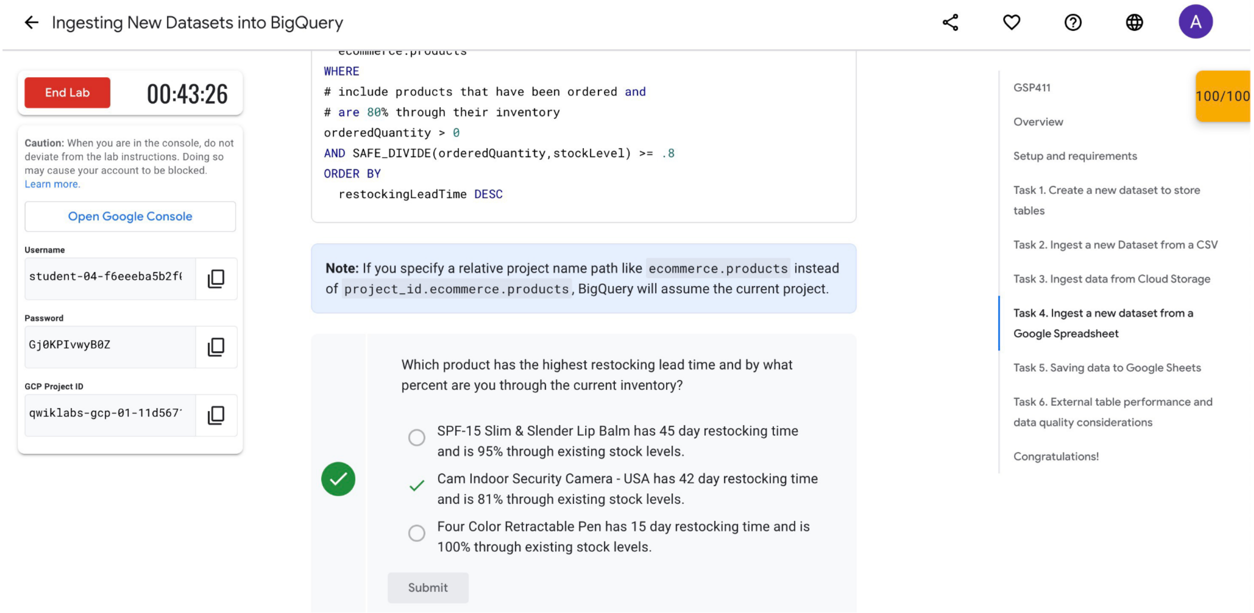
# Ingesting New Datasets into BigQuery

Task 1. Create a new dataset to store tables

Task 2. Ingest a new Dataset from a CSV

Task 3. Ingest data from Cloud Storage

Task 4. Ingest a new dataset from a Google Spreadsheet



Ingesting New Datasets into BigQuery
A
End Lab
00:43:26
Open Google Console
GSP411
100/100
Overview
Setup and requirements
Task 1. Create a new dataset to store tables
Task 2. Ingest a new Dataset from a CSV
Task 3. Ingest data from Cloud Storage
Task 4. Ingest a new dataset from a Google Spreadsheet
Task 5. Saving data to Google Sheets
Task 6. External table performance and data quality considerations
Congratulations!
Submit

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username: `student-04-f6eeeba5b2f(`

Password: `Gj0KPIvwyB0Z`

GCP Project ID: `qwiklabs-gcp-01-11d567`

```
  ecommerce.products
WHERE
# include products that have been ordered and
# are 80% through their inventory
orderedQuantity > 0
AND SAFE_DIVIDE(orderedQuantity,stockLevel) >= .8
ORDER BY
  restockingLeadTime DESC
```

**Note:** If you specify a relative project name path like `ecommerce.products` instead of `project_id.ecommerce.products`, BigQuery will assume the current project.

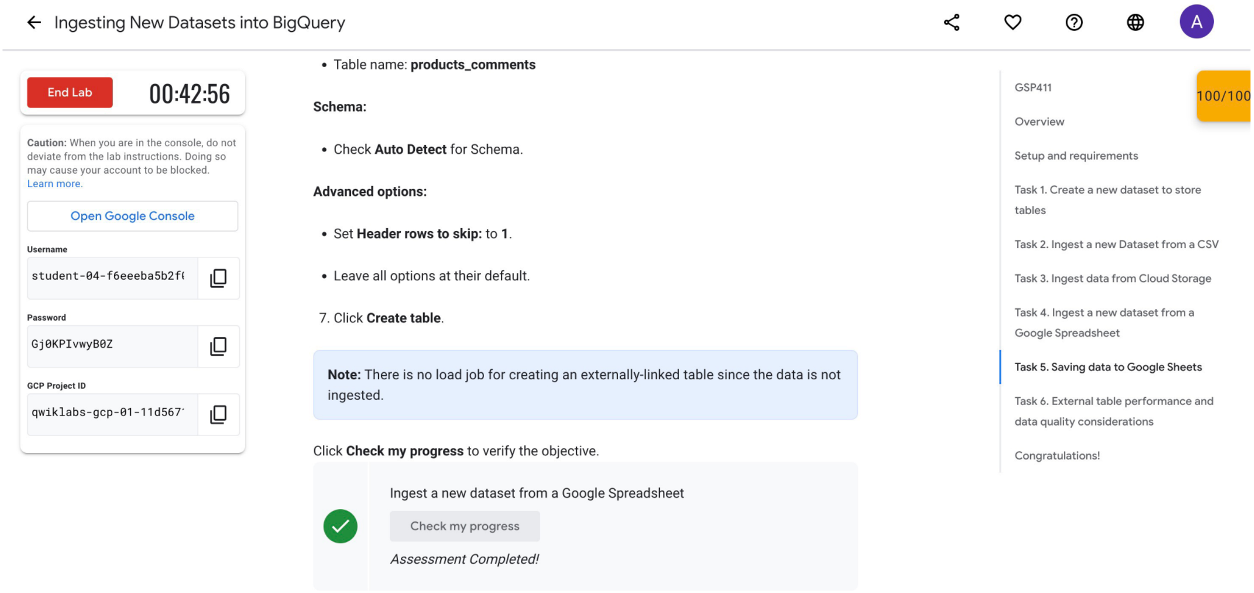
Which product has the highest restocking lead time and by what percent are you through the current inventory?

○ SPF-15 Slim & Slender Lip Balm has 45 day restocking time and is 95% through existing stock levels.

✓ Cam Indoor Security Camera - USA has 42 day restocking time and is 81% through existing stock levels.

○ Four Color Retractable Pen has 15 day restocking time and is 100% through existing stock levels.

Task 5. Saving data to Google Sheets



Ingesting New Datasets into BigQuery
A
End Lab
00:42:56
Open Google Console
GSP411
100/100
Overview
Setup and requirements
Task 1. Create a new dataset to store tables
Task 2. Ingest a new Dataset from a CSV
Task 3. Ingest data from Cloud Storage
Task 4. Ingest a new dataset from a Google Spreadsheet
Task 5. Saving data to Google Sheets
Task 6. External table performance and data quality considerations
Congratulations!
Check my progress

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username: `student-04-f6eeeba5b2f(`

Password: `Gj0KPIvwyB0Z`

GCP Project ID: `qwiklabs-gcp-01-11d567`

- Table name: **products_comments**

**Schema:**

- Check **Auto Detect** for Schema.

**Advanced options:**

- Set **Header rows to skip:** to **1**.
- Leave all options at their default.

7. Click **Create table**.

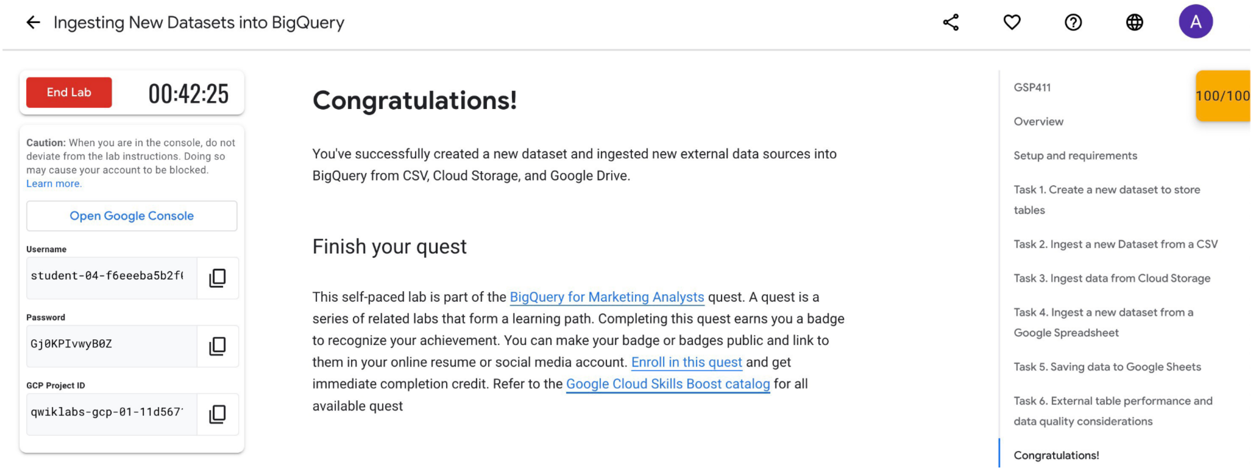
**Note:** There is no load job for creating an externally-linked table since the data is not ingested.

Click **Check my progress** to verify the objective.

Ingest a new dataset from a Google Spreadsheet

✓ *Assessment Completed!*

Task 6. External table performance and data quality considerations
Congratulations!


Ingesting New Datasets into BigQuery
End Lab
00:42:25
Open Google Console
A
100/100
GSP411
Overview
Setup and requirements
Task 1. Create a new dataset to store tables
Task 2. Ingest a new Dataset from a CSV
Task 3. Ingest data from Cloud Storage
Task 4. Ingest a new dataset from a Google Spreadsheet
Task 5. Saving data to Google Sheets
Task 6. External table performance and data quality considerations
Congratulations!

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username
student-04-f6eeeba5b2f

Password
Gj0KPIvwyB0Z

GCP Project ID
qwiklabs-gcp-01-11d567

## Congratulations!

You've successfully created a new dataset and ingested new external data sources into BigQuery from CSV, Cloud Storage, and Google Drive.

### Finish your quest

This self-paced lab is part of the BigQuery for Marketing Analysts quest. A quest is a series of related labs that form a learning path. Completing this quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. Enroll in this quest and get immediate completion credit. Refer to the Google Cloud Skills Boost catalog for all available quest

Qwiklabs is a cloud-based learning platform that provides hands-on training and practice labs on various cloud computing technologies. One of the labs offered by Qwiklabs is "Ingesting New Datasets into BigQuery," which is aimed at teaching users how to load new datasets into Google's BigQuery data warehouse.

In this lab, users are provided with a dataset in CSV format and are guided through the process of uploading the data into a new table in BigQuery. The lab includes step-by-step instructions on how to create a new dataset, create a new table, and load the data into the table using the Google Cloud Console.

The lab also covers various options for loading data, including loading data from a local file or a Google Cloud Storage bucket. Additionally, users learn how to configure the load job to handle various data formats, including CSV, JSON, and Avro.

**Pros and Cons of Big Query**
Pros:

1. BigQuery is built to handle petabyte-scale data and can manage enormous data intake and querying.
2. BigQuery is geared for speed, and queries may be executed in seconds, even on very huge datasets.
3. BigQuery is a pay-as-you-go service, which means you only pay for the data you consume and the processing power you require. Also, storage expenses are inexpensive, and there are no upfront expenditures.
4. BigQuery works nicely with other Google Cloud services as well as external tools like Tableau and Data Studio.
5. BigQuery has robust security features like encryption at rest and in transit, access control, and audit tracking.

Cons:

1. BigQuery is user-friendly, but there is a learning curve to fully understand its capabilities and how to use it effectively.
2. While BigQuery supports the majority of SQL functions, some are still unavailable, which might be a constraint for specific use cases.
3. Because moving data into and out of BigQuery can be expensive, it is critical to design your data storage and access strategy properly.
4. While BigQuery is generally inexpensive, conducting complicated searches or queries on extremely big datasets might incur considerable fees.
5. Access control is limited to the project- and dataset-level permissions in BigQuery, which may not be sufficient for some businesses with sophisticated data access requirements.