Effortless Relational Scalability with Amazon Aurora Serverless v2

**SPL-TF-200-DBASV2-1 - Version 1.0.4**

© 2023 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

Amazon Aurora Serverless is an on-demand auto scaling configuration for Amazon Aurora. With Aurora Serverless, you can scale your database capacity up or down based on your application’s need.

The lab demonstrates how Aurora Serverless v2 scales in response to changes in application demand. You will use an Aurora Serverless v2 PostgreSQL-Compatible Edition cluster to run a generated workload using pgbench. You will then monitor the database performance using Amazon CloudWatch metrics in the Amazon Relational Database Service (Amazon RDS) console, Amazon RDS Performance Insights tool, and CloudWatch dashboard.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

 You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

TOPICS COVERED

By the end of this lab, you will be able to do the following:

* Configure Aurora Serverless v2 for PostgreSQL.
* Use custom pgbench scripts to simulate workload activity.
* Monitor key CloudWatch metrics using dashboard.
* Use Performance Insights to monitor the query performance.

PREREQUISITES

This lab requires the following:

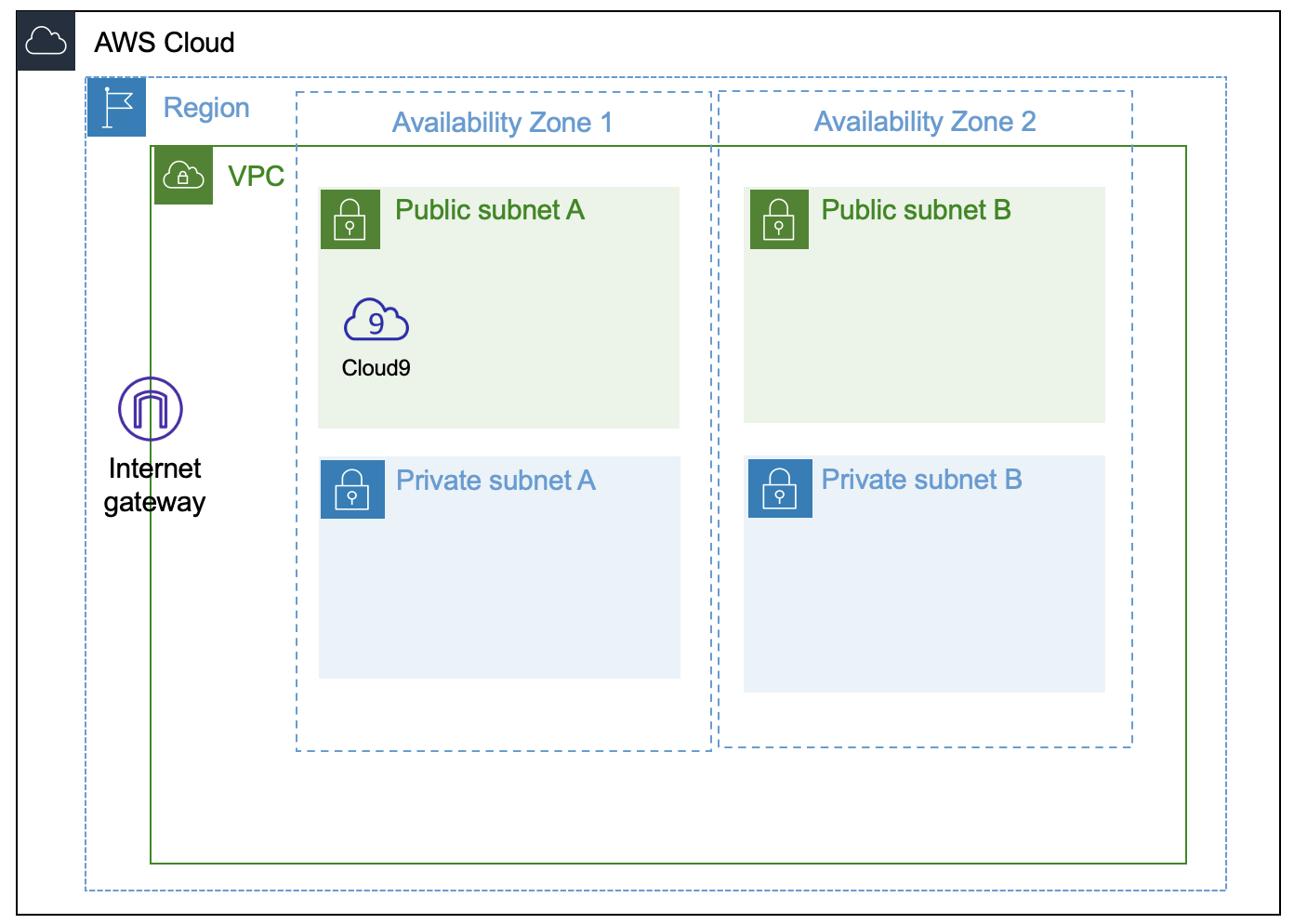
* Access to a computer with Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)
* A modern internet browser such as Chrome or Firefox
* A working knowledge of Amazon RDS, AWS Cloud9, CloudWatch, and Performance Insights and an understanding of database concepts, including database structures, data types, and structured query language (SQL)

ICON KEY

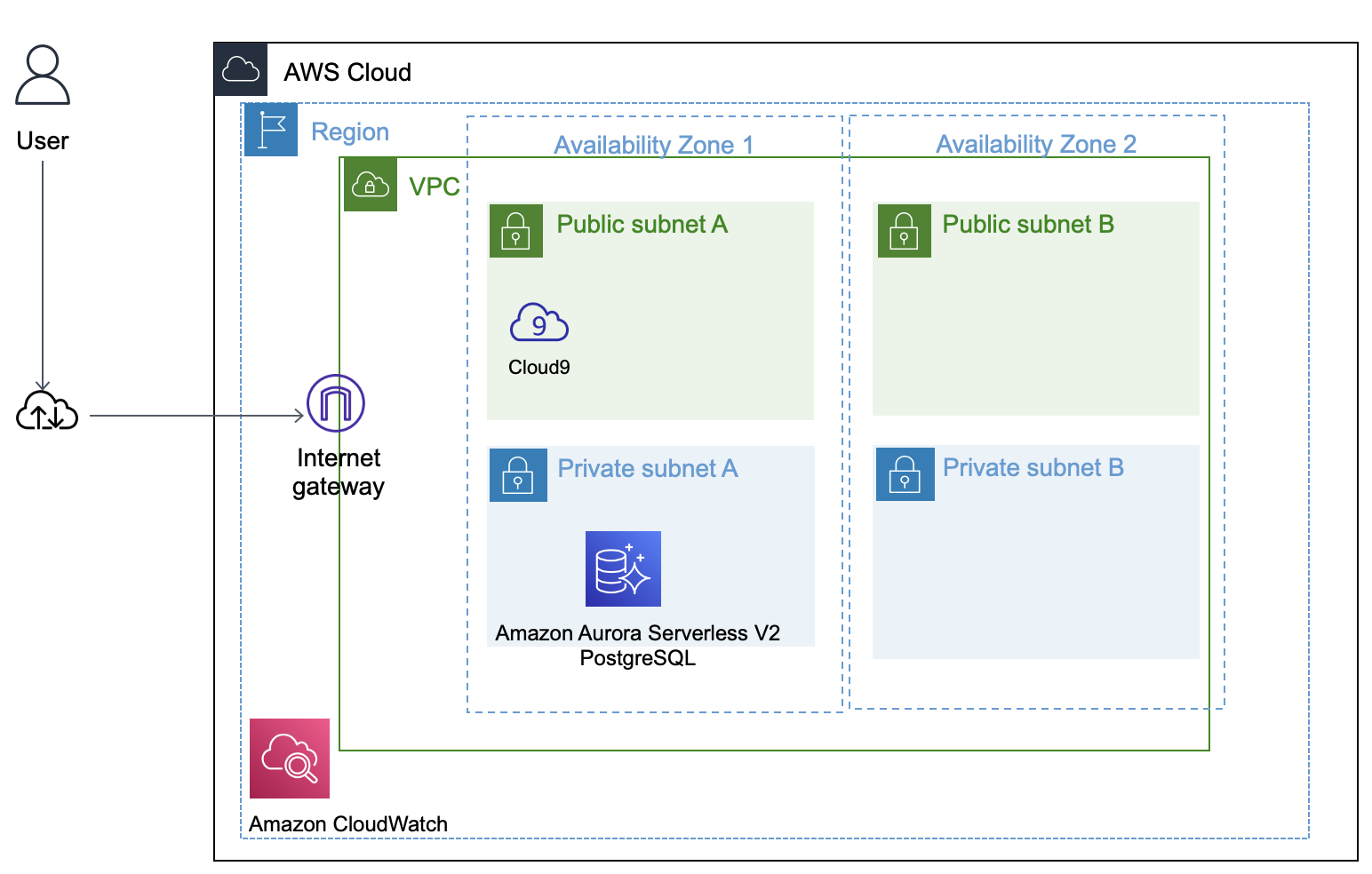
Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run
* **Expected output:** A sample output that you can use to verify the output of a command or edited file
* **Note:** A hint, tip, or important guidance
* **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand
* CAUTION: Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).

Start with the following architecture and proceed to build other components according to the instructions.



Your final architecture has additional components as shown in the following diagram.



**Task 1: Inspect the environment**

In this task, you examine various resources already provisioned for you when the lab was started.

1. At the top of the AWS Management Console, in the search box, search for and choose

CloudFormation

.

* + From the list of stacks displayed, find the stack where the *Description* field contains *Aurora Serverless V2*. Choose the blue name link for this stack to explore its details.
  + Under the Resources tab, many resources are displayed. Confirm the presence of the following resources mentioned in the table. These resources will be used in the steps that follow. There is no need to change any of these resources at this time.

| **Logical ID** | **Description** |
| --- | --- |
| MainVPC | VPC created for the database cluster and AWS Cloud9 |
| SecretPgDbMasterUser | Secret created in AWS Secrets Manager to hold database username and password |
| dbSecGroupCluster | Database security group |
| roleEnhancedMonitoring | Enhanced monitoring role to be assigned to database cluster |
| Cloud9Instance | AWS Cloud9 terminal to run commands |
| DBSubnetGroup | Subnet group for database instances |

**Note:** You can choose the link under **Physical ID** of any resource to explore its details. In the following steps, resources are referenced using Tag/Name which may be different from Logical ID mentioned in the table.

**Congratulations!** In this task, you examined the various resources created for you and you’re ready to begin making changes to this architecture.

**Task 2: Create an Aurora Serverless v2 cluster**

In this task, you create an Aurora Serverless v2 cluster for PostgreSQL.

1. At the top of the AWS Management Console, in the search box, search for and choose

RDS

.

1. In the left navigation pane, choose **Databases**.

**Note:** If **Databases** is not visible, choose the  **Navigation** icon on the left, and then choose **Databases**.

1. Choose **Create database**, and then configure the following:

Under the section **Create database**, configure the following:

* **Choose a database creation method:** Choose  **Standard create**.
* **Engine type:** Choose  **Amazon Aurora**.
* **Edition:** Choose  **Amazon Aurora PostgreSQL-Compatible Edition**.
* **Engine version**
  + From the *Available versions* dropdown menu, choose the version **Aurora PostgreSQL (Compatible with PostgreSQL 13.7)**
* **Templates:** Choose  **Dev/Test**.
* In the **Settings** section, configure the following:
  + **DB cluster identifier:** Enter

lab-db

.

* + **Master username:** Enter

student

.

* + **Master password:** Enter

Lab-password1

.

* + **Confirm password:** Enter

Lab-password1

.

* In the **Instance configuration** section, configure the following:
  + **DB instance class:** Choose  **Serverless**.
  + **Capacity range:** Keep the Minimum ACUs at

2

 and change the Maximum ACUs to

32

.

* In the **Availability & durability** section, configure the following:
  + **Multi-AZ deployment:** Verify  **Don’t create an Aurora Replica** is selected.
* In the **Connectivity** section, configure the following:
  + **Virtual Private Cloud (VPC):** Choose **Lab VPC** from the dropdown menu.
  + **Public access:** Choose  **No**.
  + **Existing VPC security groups:**
    - Choose the X to remove **default**.
    - Select **DbSecGroup** from the dropdown menu.
* In the **Monitoring** section, configure the following:
  + **Performance Insights**: Verify  **Turn on Performance Insights** is turned on.
    - Expand  **Additional configuration: Enhanced Monitoring** at the bottom of this section.
    - Verify  **Enable Enhanced monitoring** is turned on.
    - **Monitoring Role:** Choose **rds-monitoring-role** from the dropdown menu.

**Note:** Recall that this role was created for you in this lab environment.

* + Expand  **Additional configuration**, then configure the following:
    - **Initial database name:** Enter

LabDatabase

1. Keep all other values at defaults, go to the bottom of the screen, and then choose **Create database**.

**Note:** The page shows a message about your newly launched Amazon RDS instance. The RDS instance takes about **8~10** minutes to create.

When you see a message at the top *Successfully created database lab-db*, do the following:

* Choose the link *lab-db* from the message.

The database writer instance *lab-db-instance-1* is displayed with status as *Creating*.

* Refresh the browser to get the latest status. You can go to next task after the status is **Available** .

**Note:** Keep this browser open and continue to the next task. You will return to it later.

**Note:** While waiting for database creation, familiarize yourself with the [pgbench](https://www.postgresql.org/docs/current/pgbench.html) benchmark testing tool used in the next task. A documentation link is also provided at the end of these lab instructions.

**Congratulations!** In this task, you configured an Amazon Aurora Serverless v2 cluster with a single-AZ database instance. Notice that it was not necessary to specify the number or type of database servers needed to support this environment.

**Task 3: Configure load testing scripts in AWS Cloud9**

In this task, you configure the AWS Cloud9 environment and pgbench to run the load tests.

1. Copy the **Cloud9url** from the left of these instructions, paste in a new browser tab, and then press Enter.
2. On the AWS Cloud9 console, in the **Welcome** tab near the top, close the Welcome section by choosing the X.
3. Choose  in the lower right to maximize the AWS Cloud9 terminal window.
4. Before you proceed, ensure that the database is **Available**, as mentioned in the previous task.
5. **Command:** Run a set of commands to configure the **AWS Cloud9** environment to run pgbench.

* To do so, copy the following code and paste it into the AWS Cloud9 terminal, immediately after the text that ends in

~/environment $

.

**Note:** Choose the small button in the upper corner of the code to easily copy the entire text.

# Configure Cloud9 environment to run load testing scripts using Pgbench

# Run the following commands after the DataBase Endpoint is known

# Do not change any values in this script

sudo yum install -y jq

sudo amazon-linux-extras install -y postgresql13

sudo yum install -y postgresql-contrib sysbench

AWSREGION=`aws configure get region`

DBENDP=`aws rds describe-db-clusters \

--db-cluster-identifier lab-db \

--region $AWSREGION \

--query 'DBClusters[\*].Endpoint' \

--output text`

# This assumes you named the secret to be SecretPgDb

# so pick the most recently created one

# If error, you should manually set the SECRETARN variable

SECRETARN=`aws secretsmanager list-secrets \

--query 'sort\_by(SecretList[?contains(Name, \`SecretPgDb\`) == \`true\`],&CreatedDate)[-1].ARN' \

--output text`

CREDS=`aws secretsmanager get-secret-value \

--secret-id $SECRETARN \

--region $AWSREGION | jq -r '.SecretString'`

export DBUSER="`echo $CREDS | jq -r '.username'`"

export DBPASS="`echo $CREDS | jq -r '.password'`"

export DBENDP

echo DBENDP: $DBENDP

echo DBUSER: $DBUSER

##

export PGHOST=$DBENDP

export PGUSER=$DBUSER

export PGPASSWORD="$DBPASS"

echo "export DBPASS=\"$DBPASS\"" >> /home/ec2-user/.bashrc

echo "export DBUSER=$DBUSER" >> /home/ec2-user/.bashrc

echo "export DBENDP=$DBENDP" >> /home/ec2-user/.bashrc

echo "export AWSREGION=$AWSREGION" >> /home/ec2-user/.bashrc

echo "export PGUSER=$DBUSER" >> /home/ec2-user/.bashrc

echo "export PGPASSWORD=\"$DBPASS\"" >> /home/ec2-user/.bashrc

echo "export PGHOST=$DBENDP" >> /home/ec2-user/.bashrc

#Verify DB Instance

echo $DBUSER

psql LabDatabase

select current\_user, current\_database();

\q

**Expected output:** After several installation messages, you should see the following output at the end indicating successful completion of the script.

dbasv2=> select current\_user, current\_database();

current\_user | current\_database

--------------+------------------

student | LabDatabase

(1 row)

LabDatabase=> \q

Admin:~/environment $

**Command:** If you see an error message like this: ***psql: error: could not connect to server: Connection refused***, retry the following command after a few minutes. These commands cannot complete until the writer instance changes its state from *Creating* to *Available*.

psql LabDatabase

select current\_user, current\_database();

\q

**Congratulations!** In this task, you configured the AWS Cloud9 environment to run the load tests and verified connectivity to the Aurora Serverless cluster environment.

**Task 4: Create a database activity for a regular workload**

In this task, you run pgbench to create database tables and run load scripts to generate load on the database. Then you monitor various CloudWatch metrics through the dashboard to observe the scaling behavior.

1. **Command:** Initialize the PostgreSQL database to run pgbench scripts by running the following command:

pgbench -i --fillfactor=90 --scale=500 --host=$DBENDP --username $DBUSER LabDatabase

**Note:** This action creates four tables inside the Aurora database to be used by this lab: pgbench\_accounts, pgbench\_branches, pgbench\_history, and pgbench\_tellers. It then inserts sample data into these tables for demonstration purposes. This takes a few minutes. To understand the details on the working of pgbench, refer to the documentation link provided at the end of this lab.

**Expected output:** Here is the portion of sample output from the pgbench initialization process.

creating tables...

generating data (client-side)...

50000000 of 50000000 tuples (100%) done (elapsed 61.62 s, remaining 0.00 s)

vacuuming...

creating primary keys...

done in 102.58 s (drop tables 0.00 s, create tables 0.07 s, client-side generate 63.26 s, vacuum 21.65 s, primary keys 17.60 s).

You have prepared the database for running various load tests. Now, run your first testing script to generate some load on database.

1. **Command:** Run the following command on the AWS Cloud9 terminal. This runs for about 5 minutes, generating load on your database. You will see little to no activity on the screen during this time. Read the following notes and continue to the next task while this benchmarking test completes. Do not close this window.

#Database Load (Select, Insert, Update)

pgbench LabDatabase \

--host $DBENDP \

--username student \

--protocol=prepared -P 60 \

--time=300 \

--client=500 \

--jobs=500

**Note:** This script performs **SELECT**, **INSERT**, and **UPDATE** SQL commands across client sessions. For details, refer to the pgbench link provided at the end of this lab.

**Expected output:** Here is the expected output after the command completion.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\***\*\* Your OUTPUT will be similar to below. **\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**dropping old tables...**

**NOTICE: table "pgbench*\_accounts" does not exist, skipping***

***NOTICE: table "pgbench\_*branches" does not exist, skipping**

**NOTICE: table "pgbench*\_history" does not exist, skipping***

***NOTICE: table "pgbench\_*tellers" does not exist, skipping**

**creating tables...**

**generating data (client-side)...**

**50000000 of 50000000 tuples (100%) done (elapsed 66.63 s, remaining 0.00 s)**

**vacuuming...**

**creating primary keys...**

**done in 111.88 s (drop tables 0.01 s, create tables 0.11 s, client-side generate 71.08 s, vacuum 21.07 s, primary keys 19.61 s).**

**Congratulations!** In this task, you configured pgbench and created the resources for benchmark testing. Leave this window active and proceed to the next task. You will return to this AWS Cloud9 environment in a later task.

**Task 5: Create a dashboard for monitoring database metrics**

In this task, you create a dashboard with different widgets showing various database metrics.

1. At the top of the AWS Management Console, in the search box, search for and choose

CloudWatch

.

1. In the left navigation pane, choose **Dashboards**.

* Choose **Create dashboard**.
  + Provide a Dashboard name

Scaling-metrics

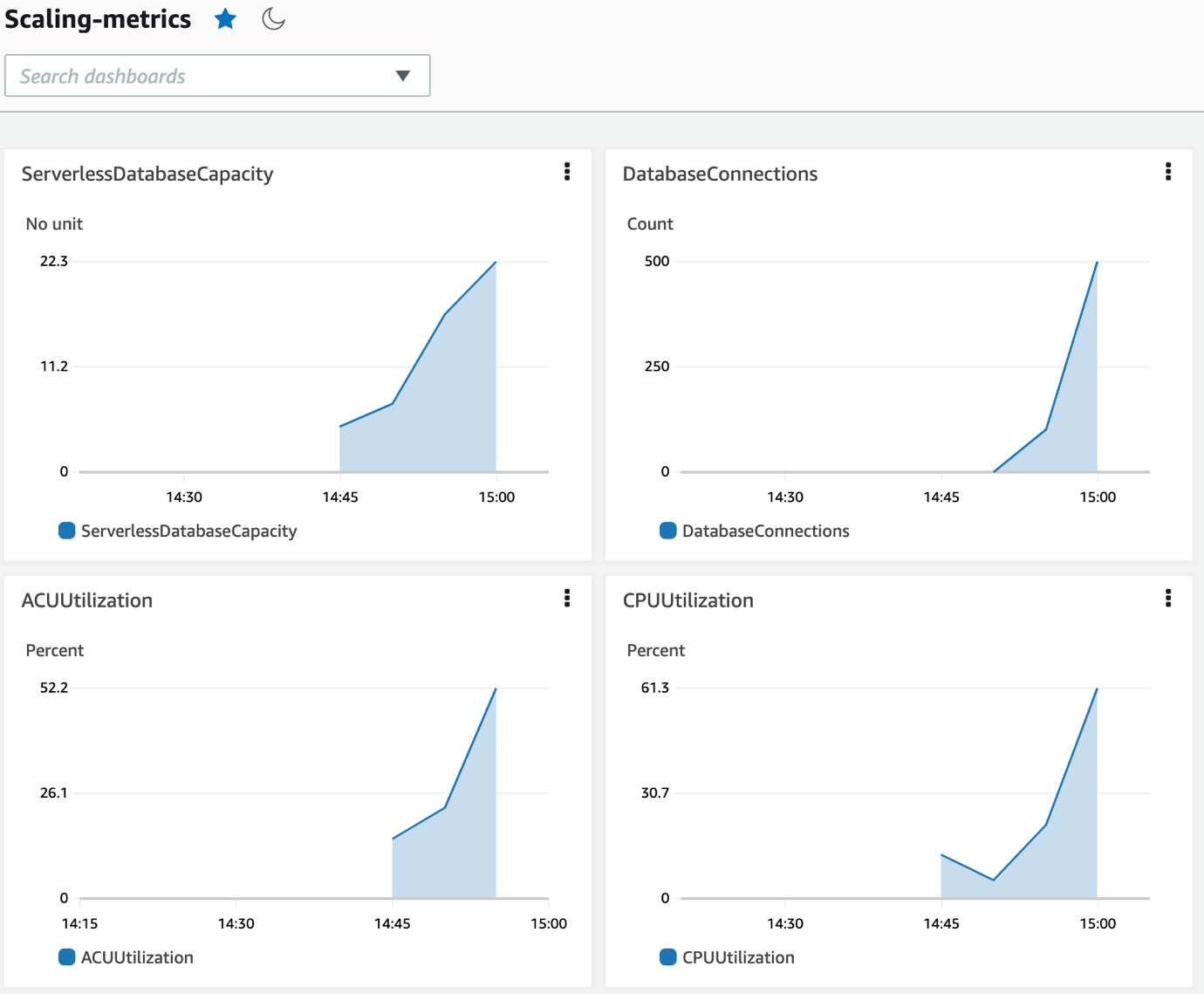
, and then choose **Create dashboard**.

* If the **Add Widget** window does not automatically appear, choose the **Add Widget** button on the top-right side of the page.
  + Choose **Stacked-area**.
  + Under **From which data source would you like to create the widget?**, choose **Metrics**.
  + Scroll down to the **Browse** tab and under the **Metrics** section, find and choose **RDS**.
    - Choose **Per-Database-Metrics**.
    - From the list of Metrics, choose the checkbox for the following metric name for

lab-db-instance-1

:

* + - * **ServerlessDatabaseCapacity**
    - Choose **Create widget**.
  + Repeat the previous steps to add the following three metrics to the dashboard as **Stacked-area widgets** using the **+** (plus) button located to the right of the screen.
    - ACUUtilization
    - DatabaseConnections
    - CPUUtilization
  + Align the widgets in the window as shown (hold the widget and move around as needed).



* The dashboard starts showing data for each metric after a short time.
  + Use the **Custom** option at the top to set the rolling time window to 45 minutes, to get a better view of the data.
  + Choose **Save dashboard**.

 These metrics have been chosen to illustrate scaling activity of the database. Feel free to use any other metrics you think are useful for this lab.

**Congratulations!** In this task, you configured the CloudWatch dashboard to monitor key database metrics.

**Note:** Leave this browser tab open and proceed to the next task. You will need to return to it in later steps.

**Task 6: Create a database activity for variable read-only workloads**

In this task, you run **read only queries** on the database and observe scaling metrics. This is a light *read only* load.

1. Return to the AWS Cloud9 terminal (log in to reconnect if prompted).

**Note:** The earlier task has completed. Take a moment to observe the results of that task.

**Consider:** Discuss with your instructor as needed.

1. **Command:** Run the following command in the AWS Cloud9 terminal.

#Database Readonly Load Test (Select)

pgbench LabDatabase \

--host $DBENDP \

--username student \

--time=300 \

--client=500 \

--select-only \

--jobs=500

**Expected output:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\***\*\* Your OUTPUT will be similar to below. **\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**starting vacuum...end.**

**transaction type: <builtin: select only>**

**scaling factor: 500**

**query mode: simple**

**number of clients: 500**

**number of threads: 500**

**duration: 300 s**

**number of transactions actually processed: 1765998**

**latency average = 85.080 ms**

**tps = 5876.832358 (including connections establishing)**

**tps = 6184.647161 (excluding connections establishing)**

1. After a few minutes, return to the CloudWatch dashboard.
2. Choose **Refresh** to see the values on your dashboard changing in response to the increased demand on the server.
3. While the script is running, switch to **RDS Management Console**.
4. From the menu on the left-hand side, choose **Performance insights**.

* On the **Performance insights** console, from the dropdown menu, choose the database **labdb-instance-instance-1**.
* In the first section, **Counter metrics** is displayed by default for *Commits/s*. If there are no metrics available by default, visit the **Manage metrics** section and observe that you can add your choice of metrics. No particular metrics are required for this part of the lab.
* In the following **Database load** panel, observe the **Average active sessions AAS** metrics along with other database parameters on the graph.
* Scroll down to the **TOP SQL** panel.
  + Note the **Load by waits (AAS)** metrics. These are the most load-generating queries being run against the database at the present time. This is a handy resource for spotting poorly designed queries that should be optimized by a database developer.
  + Choose the radio button  next to the top row and expand using the **+** sign to see the individual query performance (similar to the following).
  + Explore a few of these queries to see how they performed.

​ PI-Top_query

**Note:** These metrics are expected to remain green due to auto scaling of the database.

In the next sub-step, you will increase the load on the database by starting

2000

 connections and observe the scaling.

1. **Command:** From the AWS Cloud9 terminal, run the following command (log in to reconnect if prompted),

#Database Readonly Load Test (Select)

pgbench LabDatabase \

--host $DBENDP \

--username student \

--time=200 \

--client=2000 \

--select-only \

--jobs=100

**Expected output:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\***\*\* Your OUTPUT will be similar to the following. **\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**starting vacuum...end.**

**transaction type: <builtin: select only>**

**scaling factor: 500**

**query mode: simple**

**number of clients: 2000**

**number of threads: 100**

**duration: 200 s**

**number of transactions actually processed: 824354**

**latency average = 488.106 ms**

**tps = 4097.471435 (including connections establishing)**

**tps = 4143.402441 (excluding connections establishing)**

1. Return to the **CloudWatch console**.

* Choose **Refresh** to see the latest metrics. The system is under four times the previous load.
* Go to **Actions** > **Period override** > **30 seconds**.
  + This provides a granular view of scaling behavior.

Finally, you put even higher load on the database by starting 5,000 client connections and observe scaling on dashboard.

1. **Command:** From the AWS Cloud9 terminal, run the following command (log in to reconnect if prompted).

#Database Readonly Load Test (Select)

pgbench LabDatabase \

--host $DBENDP \

--username student \

--time=200 \

--client=5000 \

--select-only \

--jobs=1000

 Ignore warnings similar to *connection limit exceeded for non-superusers* if you encounter.

**Expected output:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\***\*\* Your OUTPUT will be similar to the following. **\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**starting vacuum...end.**

**transaction type: <builtin: select only>**

**scaling factor: 500**

**query mode: simple**

**number of clients: 5000**

**number of threads: 1000**

**duration: 200 s**

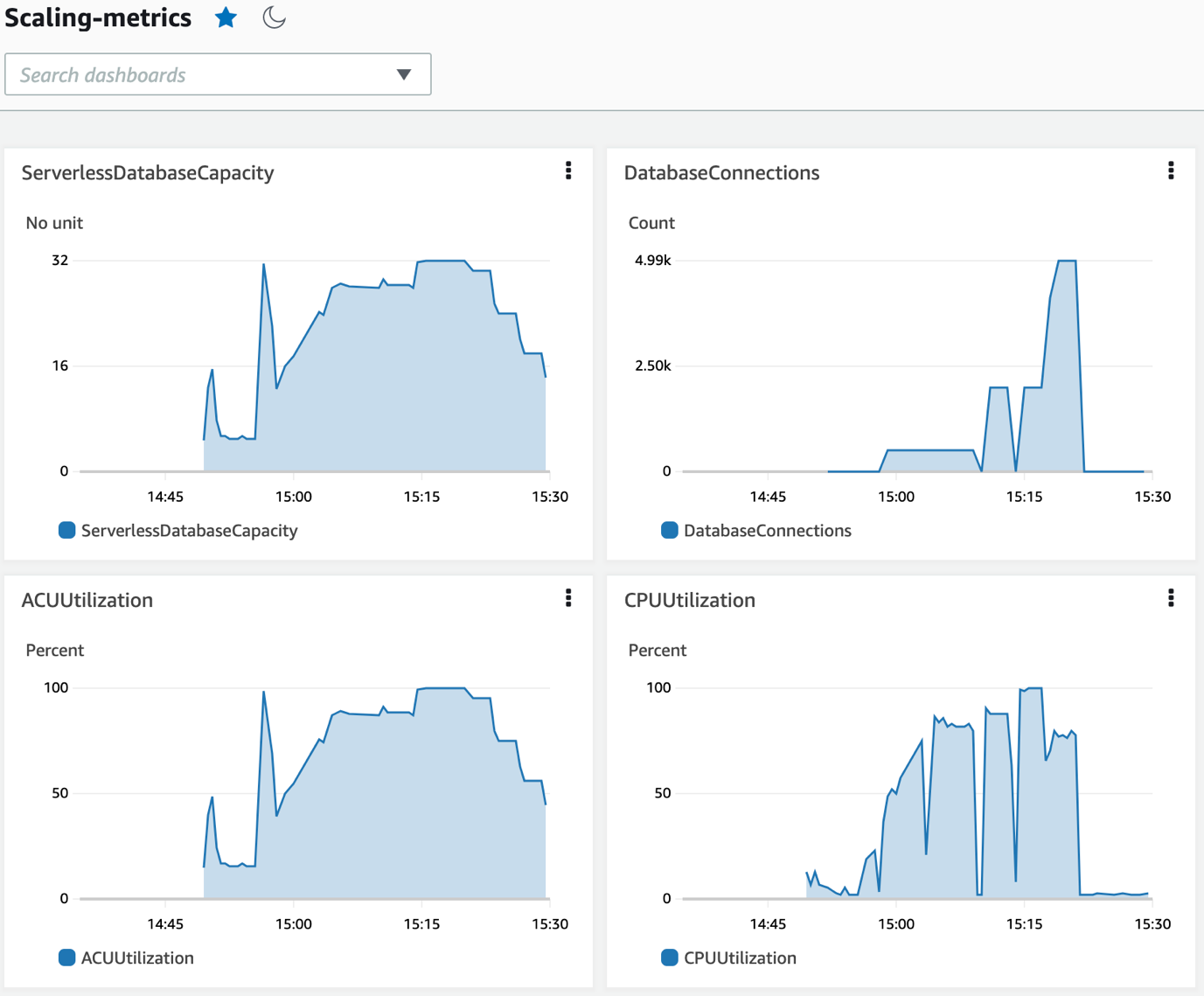
**number of transactions actually processed: 1988798**

**latency average = 531.585 ms**

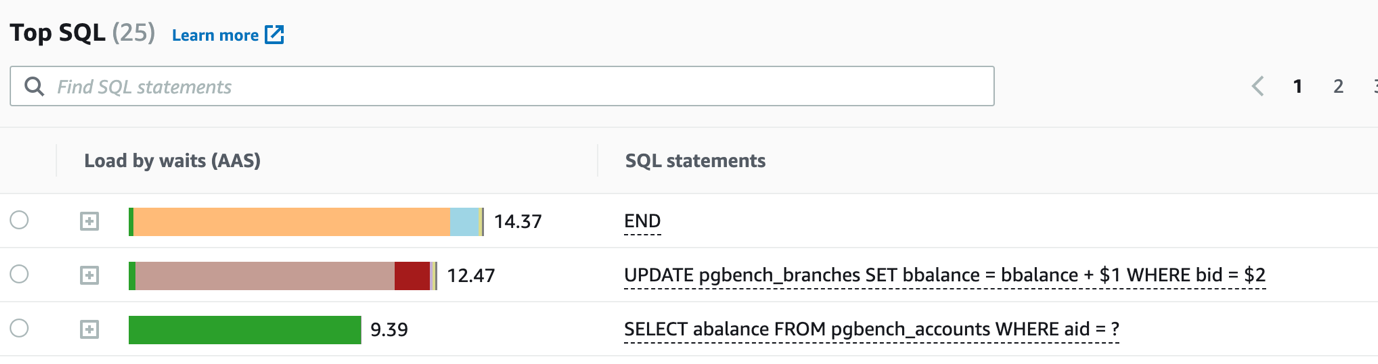
**tps = 9405.838128 (including connections establishing)**

**tps = 11538.034841 (excluding connections establishing)**

The CloudWatch dashboard from the last few test runs should appear similar to the following. Notice that Aurora Serverless v2 responds to demanding workload in a timely manner, scaling the capacity as needed. The system is now under 10 times the load of the original test, and the serverless environment is seamlessly adding capacity behind the scenes to accommodate the load. No adjustments to numbers of servers or server capacity are required.



Performance Insights data for the duration of these tests show consistent read performance as Aurora Serverless v2 scales to the demanding workload.



**Congratulations!** You have successfully tested Aurora Serverless v2 for performant activity under a variety of read-only workloads.

**Conclusion**

**Congratulations!** You now have successfully done the following:

* Configured Aurora Serverless v2 for PostgreSQL.
* Used custom pgbench scripts to simulate workload activity.
* Monitored key CloudWatch metrics using dashboard.
* Used Performance Insights to monitor the query performance.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional resources**

* For more information about how to use Amazon Aurora Serverless v2, see [Amazon Aurora Serverless V2](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless-v2.html).
* For more information about how to use pgbench, see [pgbench](https://www.postgresql.org/docs/current/pgbench.html).
* See the performance monitoring workshop at [Monitoring workshop](https://catalog.us-east-1.prod.workshops.aws/workshops/31babd91-aa9a-4415-8ebf-ce0a6556a216/en-US).
* Refer to [the AWS News blog](https://aws.amazon.com/blogs/aws/amazon-aurora-serverless-v2-is-generally-available-instant-scaling-for-demanding-workloads/) to understand Aurora Serverless scaling.

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).