



Custom Coded Workflow Actions - Workshop

Connect to a MySQL Database

Jack Coldrick, Senior Solutions Engineer @ HubSpot

Overview

- What is a custom coded workflow action?
- What will we be doing in this workshop?
- Setting up a developer account
- Practical:
 - Step 1: Setup a developer account
 - Step 2: Create the Database Instance
 - Step 3: Configure the Database Instance
 - Step 4: Create the Workflow
 - Step 5: Test the Workflow
- Useful Resources

60
minutes



What will we be doing in this workshop?

MySQL is a popular open source relational database. In this workshop we will be creating a database using Amazon Web Services (AWS) and building a HubSpot custom coded action that will check to see if a contact exists in the database when they're created in the CRM. If they do exist, no action is taken. If they don't exist we'll add a new row to the database.

[View Video Walkthrough](#)



What is a custom coded workflow action?

Custom coded workflow actions are a feature included with HubSpot's Operations Hub Professional and allow you to write code in Javascript within a Node JS Runtime environment to solve for specific use cases and processes relevant to your business.

[Learn More](#)[Use Cases](#)[Blog Post](#)

Step 1: Setup a Developer Account

This is an important step, setting up a developer account. You can do this by clicking on the link below. Once you've set this up you will have a special HubSpot portal where you can create up to 10 test portals that will allow you to try and test functionality in a controlled environment. If you already have a developer account feel free to skip this section!

[Create a developer account](#)[Step 1 Video Walkthrough](#)

Step 1: Setup a Developer Account

An overview of the developer account

When you finish setting up your developer account you will see something similar to the screenshot on the left. There are a couple of items in the navigation bar:

- Apps - Register applications and manage scopes so that it can be installed in different portals.
- Testing - Create test portals to try/test HubSpot functionality. Each lasts 90 days and can be renewed manually for a further 90 days.
- App Marketplace - Manage your app listing
- Docs - Links to the developer documentation
- Forums - Links to the developer community, a great place to ask questions and converse with like minded individuals.



Developer Home

Welcome to the HubSpot platform

Get started by building your first app, learn more about the platform, or explore popular features.



Create an app

Ready to build an app? Start here.



Create a test account

Create a test HubSpot account to try out APIs and integrations.

[Not sure where to start?](#)

Connect your app to HubSpot accounts using OAuth

Already built an app and need to connect it to an account? Not sure what OAuth is all about?

[Connecting apps with OAuth](#)

Ready to list your app?

Have you built an app or integration that you want to distribute to HubSpot customers? Join our App Partner Program and get your app listed in our Marketplace.

[Manage app listings](#) [Learn more about the App Marketplace](#)

Still not sure where to start?

Not quite ready to create an app? That's OK. You can also read more about the platform, explore code examples, or learn about the App Marketplace to help you get started.

[Explore the developer community](#)

[Check out the HubSpot API docs](#)

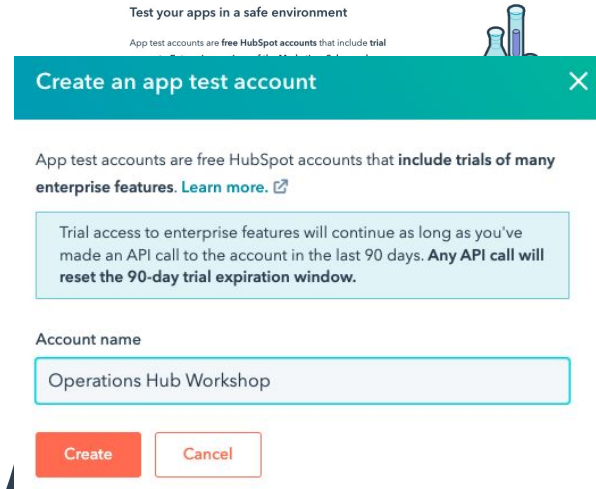


Step 1: Setup a Developer Account

Create a test account

[Test accounts](#) are a great way to try/test functionality in an isolated environment. They provide access to the enterprise suite of HubSpot. You can create a maximum of 10 test portals, each of which lasts for a total of 90 days. They can be manually renewed for a further 90 if desired.

1. Click on "Testing" and in the window that appears click "Create app test account"
2. In the pop up give your account a name and click "Ok"
3. You should see the account in the list, click into it to access your enterprise developer environment. This is where we'll build our custom coded workflow actions.



<input type="checkbox"/>	ACCOUNT DETAILS	EXPIRATION DATE	CONNECTED APPS	STATUS
<input type="checkbox"/>	Operations Hub Workshop ↗ ID 25276890	13 Feb 2022 07:01		Active

#1

#2

#3



Step 2: Create the Database Instance

In this section we're going to setup the database that we'll be querying using our custom coded workflow action. As mentioned previously I'll be using Amazon Web Services RDB to set this up. However you're free to use whatever provider you'd like. So long as you've a database setup you are good to go.

[Create AWS Free Account](#)

[Step 2 Video Walkthrough](#)



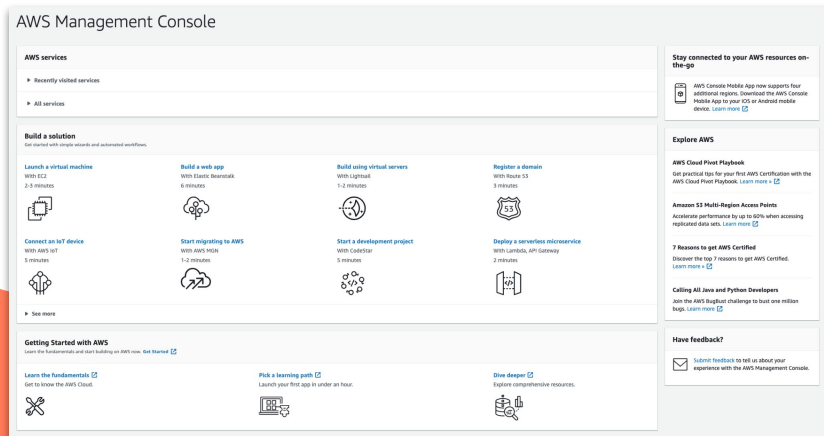
Step 2: Create the Database Instance

Setup an AWS Account

If you don't already have one an important step is to setup an AWS account. You can do this by clicking on the link below:

<https://aws.amazon.com/free/>

Once you've done this you should have access to your AWS management console:



What is AWS Free?

The AWS Free Tier provides Amazon customers the ability to explore and try out AWS services free of charge up to specified limits for each service. The Free Tier is comprised of three different types of offerings, a 12-month Free Tier, an Always Free offer, and short term trials. Services with a 12-month Free Tier allow customers to use the product for free up to specified limits for one year from the date the account was created. Services with an Always Free offer allow customers to use the product for free up to specified limits as long as they are an AWS customer. Services with a short term trial are free to use for a specified period of time or up to a one-time limit depending on the service selected. Details on the limits and services provided for free are detailed in each card on the Free Tier page. If your application use exceeds the free tier limits, you simply pay standard, pay-as-you-go service rates (see each service page for full pricing details). Restrictions apply; see offer terms for more details.



Step 2: Create the Database Instance

Navigate to the RDS tool

Now that your AWS account is setup and you have access to the management console you are ready to begin setting up your database. First click on “All services” and navigate to to “RDS” under the heading “Database”:

Click here

- ▼ All services
 - Compute
 - EC2
 - Lightsail
 - Lambda
 - Batch
 - Elastic Beanstalk
 - Serverless Application Repository
 - AWS Outposts
 - EC2 Image Builder
 - AWS App Runner
 - Containers
 - Elastic Container Registry
 - Elastic Container Service
 - Elastic Kubernetes Service
 - Red Hat OpenShift Service on AWS
 - Storage
 - S3
 - EFS
 - FSx
 - S3 Glacier
 - Storage Gateway
 - AWS Backup
 - Database
 - RDS**
 - DynamoDB
 - ElastiCache
 - Neptune
 - Amazon QLDB
 - Amazon DocumentDB
 - Amazon Keyspaces
 - Amazon Timestream
 - Amazon MemoryDB for Redis
 - Developer Tools
 - CodeStar
 - CodeCommit
 - CodeArtifact
 - CodeBuild
 - CodeDeploy
 - CodePipeline
 - Cloud9
 - CloudShell
 - X-Ray
 - AWS FIS
 - Customer Enablement
 - AWS IQ
 - Support
 - Managed Services
 - Activate for Startups
 - Robotics
 - AWS RoboMaker
 - Blockchain
 - Amazon Managed Blockchain
 - Satellite
 - Ground Station
 - Quantum Technologies
 - Amazon Braket
 - Management & Governance
 - AWS Organizations
 - CloudWatch
 - AWS Auto Scaling
 - CloudFormation
 - CloudTrail
 - Config
 - Machine Learning
 - Amazon SageMaker
 - Amazon Augmented AI
 - Amazon CodeGuru
 - Amazon DevOps Guru
 - Amazon Comprehend
 - Amazon Forecast
 - Amazon Fraud Detector
 - Amazon Kendra
 - Amazon Lex
 - Amazon Personalize
 - Amazon Polly
 - Amazon Rekognition
 - Amazon Textract
 - Support
 - Amazon Transcribe
 - Amazon Translate
 - AWS DeepComposer
 - AWS DeepLens
 - AWS RoboRacer
 - AWS Panorama
 - Amazon Monitron
 - Amazon HealthLake
 - Amazon Lookout for Vision
 - Amazon Lookout for Equipment
 - Amazon Lookout for Metrics
 - Analytics
 - Athena
 - Amazon Redshift
 - EMR
 - CloudSearch
 - Amazon OpenSearch Service (successor to Amazon Elasticsearch Service)
 - Kinesis
 - QuickSight
 - AWS Cost Management
 - AWS Cost Explorer
 - AWS Budgets
 - AWS Marketplace Subscriptions
 - AWS Application Cost Profiler
 - Front-end Web & Mobile
 - AWS Amplify
 - Mobile Hub
 - AWS AppSync
 - Device Farm
 - Amazon Location Service
 - AR & VR
 - Amazon Sumerian
 - Application Integration
 - Step Functions
 - Amazon AppFlow
 - Amazon EventBridge
 - Amazon MQ
 - Simple Notification Service
 - Simple Queue Service
 - SWF
 - Managed Apache Airflow
 - Business Applications
 - Amazon Connect
 - Amazon Pinpoint
 - Amazon Honeycode
 - Amazon Chime
 - Amazon Simple Email Service
 - Amazon WorkDocs
 - Amazon WorkMail
 - Alexa for Business
 - End User Computing

What is Amazon RDS?

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

[LEARN MORE HERE](#)



Step 2: Create the Database Instance

Create database

Once you've clicked into the RDS tool you should see an interface similar to the one below. From here we will click on the option to "Create database"

Once you click on the "Create database" option you will be brought into the database creation wizard. This is where we will configure our database and publish. There are a couple of steps involved here but don't worry we'll address them on the next few slides!

The screenshot displays the Amazon RDS console interface. On the left is a navigation sidebar with the following menu items: Dashboard, Databases, Query Editor, Performance Insights, Snapshots, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Events, Event subscriptions, Recommendations (1), and Certificate update. The main content area is titled 'Resources' and shows a summary of Amazon RDS resources in the EU (Ireland) region. Below this is the 'Create database' section, which includes a 'Restore from 53' button and a highlighted 'Create database' button. A note states: 'Note: your DB instances will launch in the EU (Ireland) region'. The 'Service health' section at the bottom shows the current status of Amazon RDS in Ireland as 'Service is operating normally'.

Amazon RDS ×

Resources

You are using the following Amazon RDS resources in the EU (Ireland) region (used/quota)

- [DB Instances \(5/40\)](#)
 - [Allocated storage \(0.06 TB/100 TB\)](#)
[Click here to increase DB instances limit](#)
- [DB Clusters \(1/40\)](#)
- [Reserved instances \(0/40\)](#)
- [Snapshots \(6\)](#)
 - [Manual \(0/100\)](#)
 - [Automated \(6\)](#)
- [Recent events \(38\)](#)
- [Event subscriptions \(0/20\)](#)

- [Parameter groups \(3\)](#)
 - [Default \(3\)](#)
 - [Custom \(0/100\)](#)
- [Option groups \(2\)](#)
 - [Default \(2\)](#)
 - [Custom \(0/20\)](#)
- [Subnet groups \(1/50\)](#)
- [Supported platforms VPC](#)
- [Default network: vpc-031b69f9cb30ef1e1](#)

Create database

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

[Restore from 53](#) **Create database**

Note: your DB instances will launch in the EU (Ireland) region

Service health

Current status	Details
Amazon Relational Database Service (Ireland)	Service is operating normally



Step 2: Create the Database Instance

Choose a database creation method

There are two options here, keep the default selected "Standard create".

Choose a database creation method [Info](#)

<input checked="" type="radio"/> Standard create You set all of the configuration options, including ones for availability, security, backups, and maintenance.	<input type="radio"/> Easy create Use recommended best-practice configurations. Some configuration options can be changed after the database is created.
---	--



Step 2: Create the Database Instance

Engine Options

There are a number of different types of engines that power databases. AWS gives you the option to define which engine to use. We're going to be using "Amazon Aurora" which is essentially a cloud based MySQL database.

Ensure you have the settings configured as shown on the right. If you'd like to learn more about Amazon Aurora you can do so [here](#).

The screenshot shows the 'Engine options' configuration page in the AWS console. It is divided into several sections:

- Engine type**: A grid of six options with radio buttons. 'Amazon Aurora' is selected and highlighted in blue. The other options are MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server.
- Edition**: Two options with radio buttons. 'Amazon Aurora MySQL-Compatible Edition' is selected.
- Capacity type**: Two options with radio buttons. 'Provisioned' is selected. A note explains that this option allows provisioning and managing server instance sizes. 'Serverless' is also available, with a note that it scales capacity based on database load.
- Replication features**: A section with a right-pointing arrow and the text 'Single-master replication is currently selected'.
- Engine version**: A section with a right-pointing arrow and the text 'View the engine versions that support the following database features.' Below this is a 'Show filters' button.
- Available versions**: A dropdown menu showing 'Aurora (MySQL 5.7) 2.07.2'.
- Warning**: A blue box at the bottom contains an information icon and the text: 'Aurora MySQL engine versions earlier than 2.09.2 don't support the newest r6g generation instance classes.'



Step 2: Create the Database Instance

Templates

Keep the default option of “Production” checked.

Templates

Choose a sample template to meet your use case.

Production

Use defaults for high availability and fast, consistent performance.

Dev/Test

This instance is intended for development use outside of a production environment.



Step 2: Create the Database Instance

Settings

In this section you can give your database a name, it can be anything that you want. You will also define a username and a password to access the database.

We will be covering access to the database later on when we look at connecting via MySQL Workbench and also when we build out the custom coded workflow action.

Settings

DB cluster identifier [Info](#)

Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.

The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. First character must be a letter.

Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm password [Info](#)



Step 2: Create the Database Instance

DB instance class

The DB instance class is essentially how powerful you need the database to be. In other words it defines the speed and memory capacity of the database.

There are several different types and some come at a cost. We can keep the default settings as we don't need anything to powerful and we're using the AWS Free Tier.

If you are interested in learning more about DB instance classes check out this [article](#).

DB instance class

DB instance class [Info](#)

Memory optimized classes (includes r classes)

Burstable classes (includes t classes)

db.r5.large

2 vCPUs 16 GiB RAM Network: 4,750 Mbps

Include previous generation classes



Step 2: Create the Database Instance

Availability & durability

One of the great things about AWS is that it is all in the cloud and as a result there is built in redundancy in the event your database was to go down.

What this setting ensures is that there is a replica database in a different availability zone that is kept in sync with our primary database.

You don't have to do this but it's always good to factor in redundancy and failovers into any solution. More information [here](#).

Availability & durability

Multi-AZ deployment [Info](#)

- Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.
- Don't create an Aurora Replica



Step 2: Create the Database Instance

Connectivity

Quite a lot of settings here but the most important is “Public access” make sure you set this to “Yes” otherwise you will be unable to connect to the database from HubSpot.

Connectivity

Virtual private cloud (VPC) info
VPC that defines the virtual networking environment for this DB cluster.
Default VPC (vpc-031b69f9cb30ef1e1)

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

Subnet group info
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.
default-vpc-031b69f9cb30ef1e1

Public access info

Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

Choose existing
Choose existing VPC security groups

Create new
Create new VPC security group

Existing VPC security groups
Choose VPC security groups
default

Additional configuration

Database port info
TCP/IP port that the database will use for application connections.
3306



Step 2: Create the Database Instance

Database authentication

Two options here, we're only concerned with "Password authentication" - in other words to access the database you need to provide a password.

The other option is using IAM which stands for Identity Access Management and rather than using passwords to access a database you use a token provisioned for users.

Database authentication

Database authentication options [Info](#)

- Password authentication
Authenticates using database passwords.
- Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.



Additional Configuration

There are a couple of options available in this section but for the purposes of this workshop we do not need to make any changes.

► **Additional configuration**

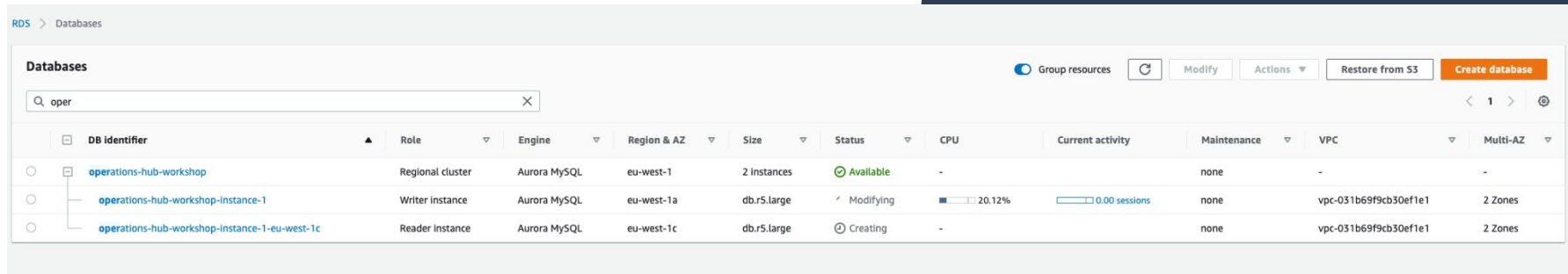
Database options, encryption enabled, failover, backup enabled, backtrack disabled, Performance Insights enabled, Enhanced Monitoring enabled, maintenance, CloudWatch Logs, delete protection enabled.



Step 2: Create the Database Instance

Database Overview

On the next screen you should see a table similar to the below. It might take a couple of minutes (potentially up to 20 minutes) for your database to become "Available" - don't worry that is expected!



The screenshot shows the AWS RDS Databases console. At the top, there is a search bar with the text "oper". Below the search bar is a table with the following columns: DB identifier, Role, Engine, Region & AZ, Size, Status, CPU, Current activity, Maintenance, VPC, and Multi-AZ. The table contains three rows of data:

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity	Maintenance	VPC	Multi-AZ
operations-hub-workshop	Regional cluster	Aurora MySQL	eu-west-1	2 Instances	Available	-	-	none	-	-
operations-hub-workshop-instance-1	Writer instance	Aurora MySQL	eu-west-1a	db.r5.large	Modifying	20.12%	0.00 sessions	none	vpc-031b69f9cb30ef1e1	2 Zones
operations-hub-workshop-instance-1-eu-west-1c	Reader instance	Aurora MySQL	eu-west-1c	db.r5.large	Creating	-	-	none	vpc-031b69f9cb30ef1e1	2 Zones



Step 2: Create the Database Instance

Database Overview

If you select a database you'll notice the section labelled "Connectivity & Security" this information is very important as we'll need it to connect and ultimately configure the database. The most important parts are:

- Endpoint
- Port
- Username
- Password

The screenshot displays the AWS Management Console interface for an Amazon RDS database instance. The breadcrumb navigation shows the path: RDS > Databases > operations-hub-workshop > operations-hub-workshop-instance-1. The main heading is 'operations-hub-workshop-instance-1'. Below this, there is a 'Related' section with a search bar containing 'oper' and a table listing database instances. The table has columns for DB Identifier, Role, Engine, Region & AZ, Size, Status, CPU, Current activity, and Maintenance. The instance 'operations-hub-workshop-instance-1' is selected, showing it is a 'Writer instance' of 'Aurora MySQL' in the 'eu-west-1a' region, with a size of 'db.r5.large', status of 'Modifying', and 20.12% CPU usage. Below the table, there are tabs for 'Connectivity & security', 'Monitoring', 'Logs & events', 'Configuration', 'Maintenance', and 'Tags'. The 'Connectivity & security' tab is active, showing three sections: 'Endpoint & port', 'Networking', and 'Security'. The 'Endpoint & port' section shows the endpoint 'operations-hub-workshop-instance-1.cnv5jtculgzo.eu-west-1.rds.amazonaws.com' and port '3306'. The 'Networking' section shows the availability zone 'eu-west-1a', VPC 'vpc-031b69f9cb30ef1e1', subnet group 'default-vpc-031b69f9cb30ef1e1', and subnets 'subnet-0cb525a3e16055b62', 'subnet-0d83487f69afce4ec', and 'subnet-0d0c0df8b370e488a'. The 'Security' section shows the VPC security group 'default (sg-0790de851dae8b45a)' is active, publicly accessible, and has a certificate authority 'rds-ca-2019' with an authority date of 'August 22, 2024, 06:08 (UTC+08)'. Below this, there is a 'Security group rules (5)' section with a search bar and a table listing security group rules. The table has columns for Security group, Type, and Rule. Three rules are listed: 'default (sg-0790de851dae8b45a)' with type 'CIDR/IP - Inbound' and rule '0.0.0.0/0', 'default (sg-0790de851dae8b45a)' with type 'EC2 Security Group - Inbound' and rule 'sg-0790de851dae8b45a', and 'default (sg-0790de851dae8b45a)' with type 'CIDR/IP - Outbound' and rule '0.0.0.0/0'.

DB Identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity	Maintenance
operations-hub-workshop	Regional cluster	Aurora MySQL	eu-west-1	2 Instances	Available	-	-	none
operations-hub-workshop-instance-1	Writer instance	Aurora MySQL	eu-west-1a	db.r5.large	Modifying	20.12%	0.00 sessions	none
operations-hub-workshop-instance-1-eu-west-1c	Reader instance	Aurora MySQL	eu-west-1c	db.r5.large	Creating	-	-	none

Endpoint & port	Networking	Security
Endpoint operations-hub-workshop-instance-1.cnv5jtculgzo.eu-west-1.rds.amazonaws.com	Availability Zone eu-west-1a	VPC security groups default (sg-0790de851dae8b45a) Active
Port 3306	VPC vpc-031b69f9cb30ef1e1	Publicly accessible Yes
	Subnet group default-vpc-031b69f9cb30ef1e1	Certificate authority rds-ca-2019
	Subnets subnet-0cb525a3e16055b62 subnet-0d83487f69afce4ec subnet-0d0c0df8b370e488a	Certificate authority date August 22, 2024, 06:08 (UTC+08)

Security group	Type	Rule
default (sg-0790de851dae8b45a)	CIDR/IP - Inbound	0.0.0.0/0
default (sg-0790de851dae8b45a)	EC2 Security Group - Inbound	sg-0790de851dae8b45a
default (sg-0790de851dae8b45a)	CIDR/IP - Outbound	0.0.0.0/0




Step 3: Configure the Database Instance

Now that our database has been setup we're ready to configure it. What we mean by configuration is setting up a table to house the data. For this portion of the workshop we'll be using MySQL Workbench a popular client used for managing MySQL databases.

[Download MySQL Workbench](#)[Get the Code](#)[Step 3 Video Walkthrough](#)

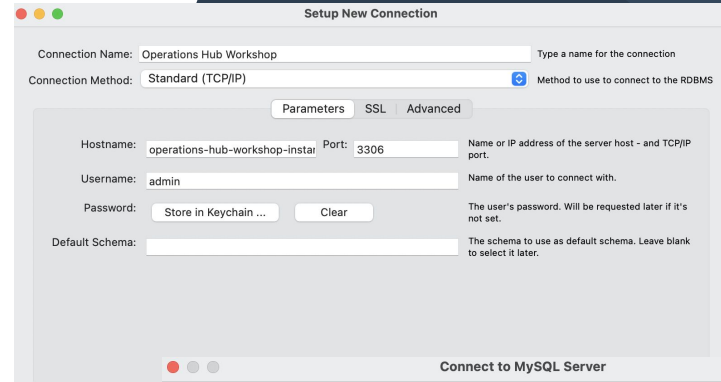
Step 3: Configure the Database Instance

Connect to the instance

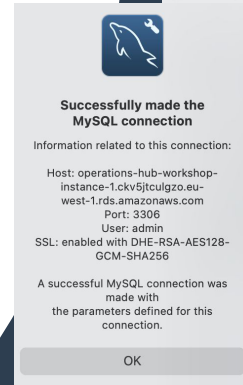
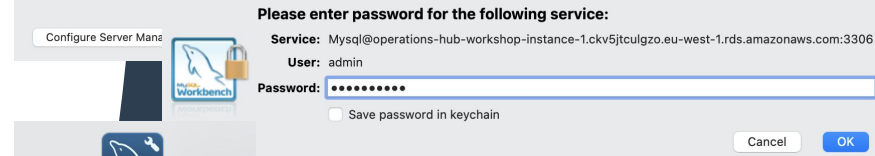
Open MySQL Workbench and click on the  icon. This will open up the wizard to connect to the database of choice.

1. Give it whatever "Connection Name" you'd like and leave "Connection Method" to "Standard (TCP/IP)". Next, enter the "Hostname". This can be found under "Endpoint & port" when viewing your AWS Database.
2. Then enter your username and click "Test connection". In the pop up that appears enter your password.
3. You should see another pop up showing whether or not the connection was successful.

#1



#2



#3



Step 3: Configure the Database Instance

Issue Connecting

If you are having trouble connecting to the database instance double check your username and password. If you are still having issues you will need to add an inbound rule to your security group.

Inbound rules are ways of managing incoming traffic to your AWS instance. Make sure you have one that allows for "All traffic", see below. More information on inbound rules can be found [here](#).

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>		
sgr-070207518c036b5c5	All traffic	All	All	Custom	Q	<input type="text"/>	Delete
sgr-06b37d9ce8f940d6b	MYSQL/Aurora	TCP	3306	Custom	Q	<input type="text"/>	Delete

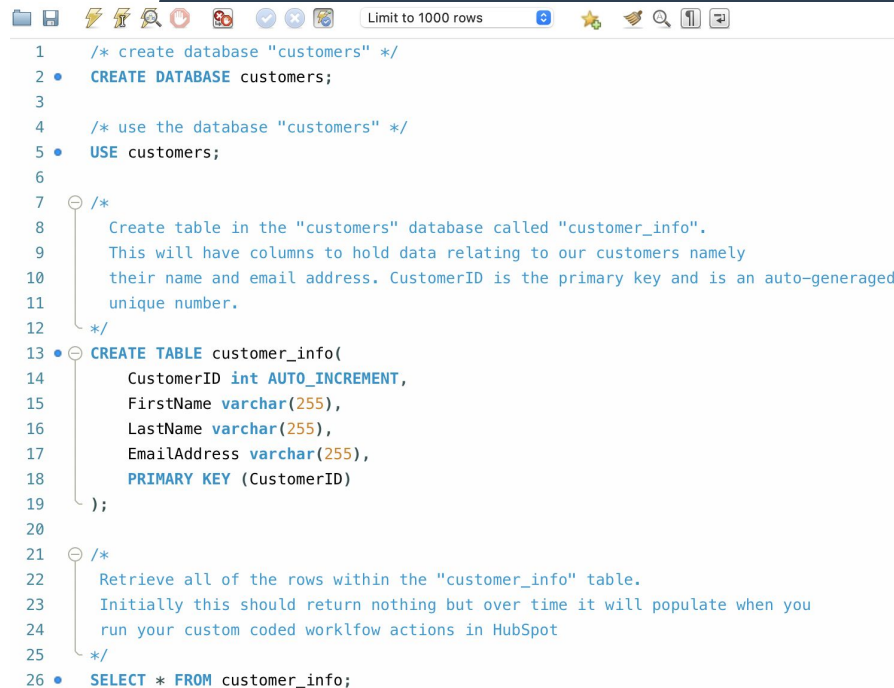


Step 3: Configure the Database Instance

Create the database

Now that the connection has been setup, we can easily connect to our AWS database instance and configure as needed. We are going to “Create a database” and create a table in that database to hold “customer information”. This is done using MySQL commands, you can copy the below into the MySQL query window and click ⚡ to execute.

```
CREATE DATABASE customers;
USE customers;
CREATE TABLE customer_info(
  CustomerID int AUTO_INCREMENT,
  FirstName varchar(255),
  LastName varchar(255),
  EmailAddress varchar(255),
  PRIMARY KEY(CustomerID)
)
SELECT * FROM customer_info
```



The screenshot shows a MySQL query editor window with a toolbar at the top. The toolbar includes icons for file operations, search, and execution. A dropdown menu is open, showing a 'Limit to 1000 rows' option. The main area contains the following SQL code:

```
1  /* create database "customers" */
2  • CREATE DATABASE customers;
3
4  /* use the database "customers" */
5  • USE customers;
6
7  /*
8   Create table in the "customers" database called "customer_info".
9   This will have columns to hold data relating to our customers namely
10  their name and email address. CustomerID is the primary key and is an auto-generated
11  unique number.
12  */
13  • CREATE TABLE customer_info(
14    CustomerID int AUTO_INCREMENT,
15    FirstName varchar(255),
16    LastName varchar(255),
17    EmailAddress varchar(255),
18    PRIMARY KEY (CustomerID)
19  );
20
21  /*
22  Retrieve all of the rows within the "customer_info" table.
23  Initially this should return nothing but over time it will populate when you
24  run your custom coded workflow actions in HubSpot
25  */
26  • SELECT * FROM customer_info;
```



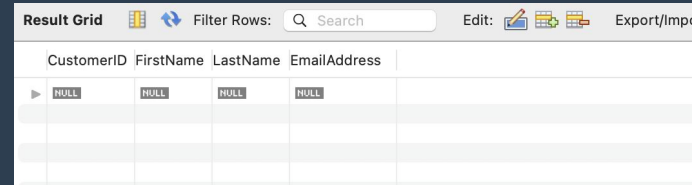
Step 3: Configure the Database Instance

Query the database

From within MySQL workbench you can query the table to see if it contains any data. Right now it shouldn't as we've yet to build our workflow but worth noting at any stage you can use the below command to query

```
SELECT * FROM customer_info
```

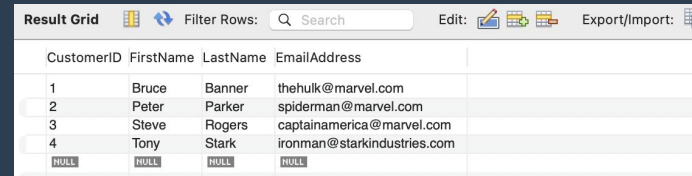
Before



The screenshot shows the MySQL Workbench interface with the 'Result Grid' window. The table has four columns: CustomerID, FirstName, LastName, and EmailAddress. All cells in the table are empty, indicating no data is currently present.

CustomerID	FirstName	LastName	EmailAddress

After*



The screenshot shows the MySQL Workbench interface with the 'Result Grid' window. The table now contains four rows of data, representing customer information.

CustomerID	FirstName	LastName	EmailAddress
1	Bruce	Banner	thehulk@marvel.com
2	Peter	Parker	spiderman@marvel.com
3	Steve	Rogers	captainamerica@marvel.com
4	Tony	Stark	ironman@starkindustries.com

**Results shown after 4 contacts were enrolled into the workflow*



Step 4: Create the Workflow

Now that the database is setup, it is time to create our workflow within HubSpot. The workflow will enroll any contacts who have been created within the CRM, use a custom coded workflow action to query the database and what happens next depends on whether or not a result was found.

[Get the Code](#)[Step 4 Video Walkthrough](#)

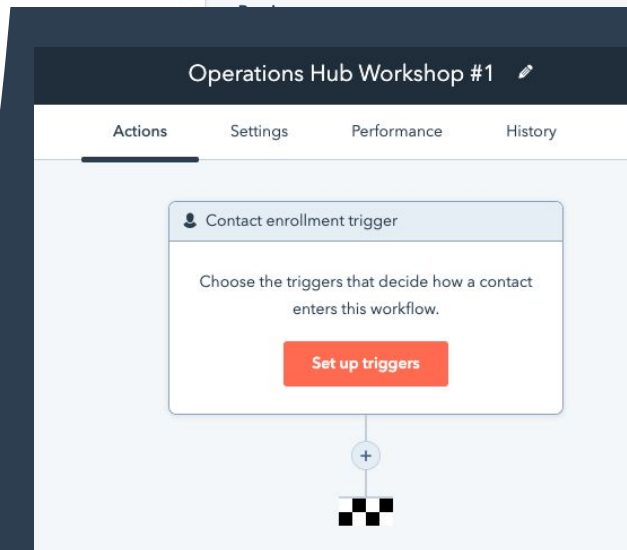
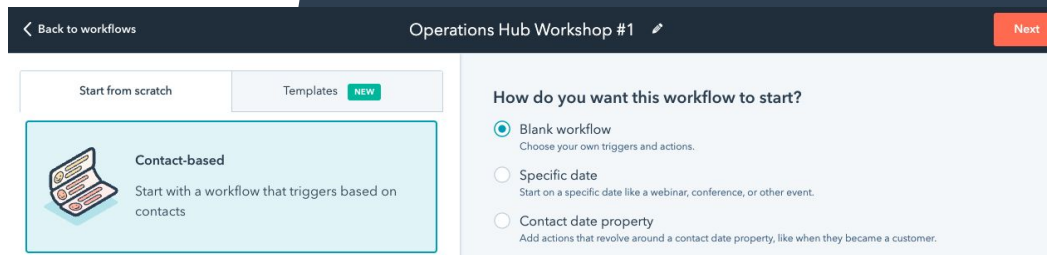
Step 4: Create the Workflow

Create Workflow

Within your test account hover over “Automation” and select “Workflows”. You are now within HubSpot's automation tool.

From here select “Create workflow”. You will be prompted to select a template, instead click on the “start from scratch” tab and ensure “Contact-based” is selected.

Once this is done click on “next” and you’ll be taken into the workflow builder screen.



Step 4: Create the Workflow

Define enrollment Criteria

Now we must define our enrollment criteria, in theory you could set this to anything you like but for the purposes of this workshop we're simply going to use "Create date is known". In other words, any time a contact is created in the CRM this workflow will be triggered.

Operations Hub Workshop #1

Actions Settings Performance History

Contact enrollment trigger

Choose the triggers that decide how a contact enters this workflow.

Set up triggers

Enrollment triggers

Trigger Re-enrollment

Contacts can always be enrolled manually.

Trigger workflow when: Test criteria

Clone Delete

Create date is known

AND

Enrollment triggers

Trigger Re-enrollment

- Allow contacts who meet the trigger criteria to re-enroll when any one of the following occurs. [Learn more about re-enrollment](#).
- They are manually enrolled
- ★ Create date is known

The following enrollment conditions can't be used for re-enrollment. [Learn more.](#)



Step 4: Create the Workflow

Choose an action

There are many [different types of actions](#) you can choose to execute within a workflow. For the purposes of this workshop we're going to be using "[Custom Coded Action](#)" - exclusive to Operations Hub Professional.

Worth noting Operations Hub also gives you the "[Trigger webhook](#)" action and "[Format data](#)" action. They're not going to be leveraged in this workshop.

Operations Hub Workshop #1

Actions Settings Performance History

Choose an action

Available actions Connect an app

Search actions

Delay

- Delay for a set amount of time
- Delay until a day or time
- Delay until event happens

Workflow

- Enroll in another workflow
- Trigger webhook
- Custom code**
- Format data

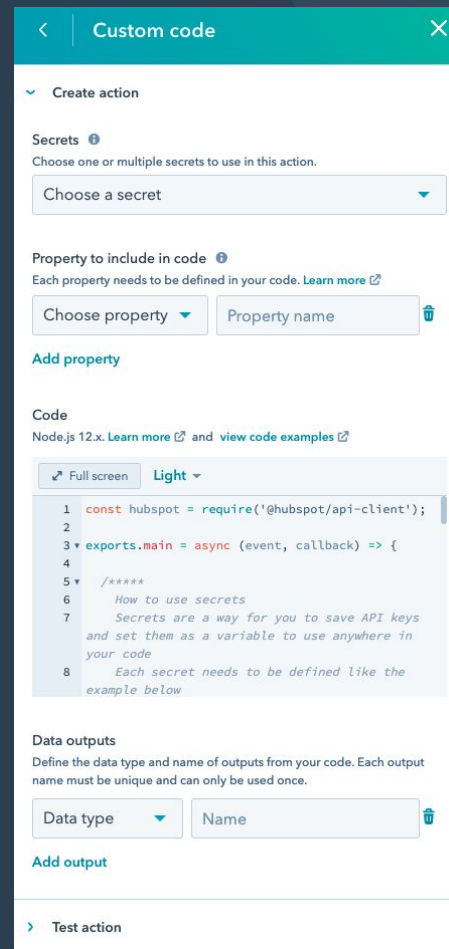


Step 4: Create the Workflow

Create custom coded action

When you click on the “Custom code” action a sidebar similar to the one shown on the left will appear. Couple of things to point out...

- Secrets - define any sort of variables like usernames, passwords that you'd like to reference in your code but keep private.
- Property to include in code - define any properties relating to the enrolled object that you'd like to reference in your code.
- Code - Your javascript, we provide a template by default.
- Data outputs - define any data you'd like the custom code action to pass back to the workflow to reference at a later step.



The screenshot shows the 'Custom code' sidebar in a workflow editor. It has a teal header with a back arrow, 'Custom code', and a close button. The main content is white with teal accents. It includes sections for 'Create action', 'Secrets', 'Property to include in code', 'Code', and 'Data outputs'. The 'Code' section is expanded, showing a JavaScript template for a HubSpot API client. The 'Data outputs' section is partially visible at the bottom.

Custom code

▼ Create action

Secrets ⓘ
Choose one or multiple secrets to use in this action.

Choose a secret ▼

Property to include in code ⓘ
Each property needs to be defined in your code. [Learn more](#) ⓘ

Choose property ▼ Property name ⓘ

[Add property](#)

Code
Node.js 12.x. [Learn more](#) ⓘ and [view code examples](#) ⓘ

Full screen Light ▼

```
1 const hubspot = require('@hubspot/api-client');
2
3 exports.main = async (event, callback) => {
4
5   /****
6    How to use secrets
7    Secrets are a way for you to save API keys
8    and set them as a variable to use anywhere in
9    your code
10   Each secret needs to be defined like the
11   example below
```

Data outputs
Define the data type and name of outputs from your code. Each output name must be unique and can only be used once.

Data type ▼ Name ⓘ

[Add output](#)

▶ Test action



Step 4: Create the Workflow

The finished product

The screenshot displays the 'Operations Hub Workshop #1' interface. The main workspace shows a workflow diagram with the following steps:

- Contact enrollment trigger**: A trigger box containing the text 'Create date is known'.
- 1. Custom code**: An action box labeled 'Custom code'.
- 2. Value equals branch**: A branch action box with the text 'Branch on the value "resultFound" from "1. Custom code"'. It has three outgoing paths: 'Yes', 'No', and 'None met', each leading to a checkered icon.

The right-hand panel is titled '1. Custom code' and contains the following configuration:

- Secrets**: A section for defining secrets. It includes a dropdown menu with selected items: 'MYSQL_PASSWORD x', 'MYSQL_USER x', 'MYSQL_DB x', and 'MYSQL_HOST x'. Below this is a note: 'Use secrets in your code with environment variables. For example, a secret with the name "MYSQL_PASSWORD" is available as process.env.MYSQL_PASSWORD.'
- Property to include in code**: A section for defining properties. It includes three dropdown menus: 'Email' (selected 'email'), 'First name' (selected 'firstname'), and 'Last name' (selected 'lastname').
- Code**: A section for defining the code. It includes a 'Full screen' button, a 'Dark' theme toggle, and a code editor with the following JavaScript code:

```
1 const mysql = require('mysql');
2
3 exports.main = async (event, callback) => {
4
5   const email = event.inputFields['email'];
6   const firstname =
7     event.inputFields['firstname'];
8   const lastname =
9     event.inputFields['lastname'];
10  let resultFound = false;
11 }
```
- Data outputs**: A section for defining data outputs. It includes an 'Enumeration' type with the name 'hs_execution_state' and a list of options: 'Succeeded', 'Failed, object removed fr...', 'Failed, object continued t...', 'Skipped, action deleted o...', and 'Partially succeeded, some ...'. It also includes a 'Boolean' type with the name 'resultFound'.



Step 5: Test the Workflow

Providing you've now setup the database and the workflow it is time to test to ensure all is working as expected. Create a contact within the CRM, they should be enrolled into the workflow.

[Step 5 Video Walkthrough](#)



Step 4: Test the Workflow

Test the Workflow

All custom coded workflow actions can be tested prior to turning the workflow on. Simply click on the "Test action" title in the sidebar. Enter the name of the contact to enroll and click "Test". You should see something similar to the screenshot on the slide.

Also feel free to run "SELECT * FROM customer_info" in MySQL Workbench, you should see rows being added to your database.

If you are happy with the results you can turn your workflow on and any time you add contacts to the CRM it should sync to your database.

```
26 • SELECT * FROM customer_info;
27
```

100% 3:19

Result Grid Filter Rows: Search Edit: Export/Import:

CustomerID	FirstName	LastName	EmailAddress
1	Bruce	Banner	thehulk@marvel.com
2	Peter	Parker	spiderman@marvel.com
3	Steve	Rogers	captainamerica@marvel.com
4	Tony	Stark	ironman@starkindustries.com
5	Carol	Danvers	captainmarvel@marvel.com
NULL	NULL	NULL	NULL

Test action

Changes will be applied to your contact. If you don't want to edit existing contacts try making a test contact.

Contact

Carol Danvers (captainmarvel@marvel.com)

Test View contact

Status

Success

Data outputs

These will be available to use as inputs in supported actions later on in this workflow. Outputs must be defined in both the code and the data outputs form. Only outputs defined in the form will be displayed here. Learn more

NAME	VALUE
hs_execution_state	SUCCESS
resultFound	false

Logs

2021-11-15T14:26:43.723Z	INFO	CONNECTE
2021-11-15T14:26:43.763Z	INFO	NO RESUL
2021-11-15T14:26:43.806Z	INFO	1 RECORD

Memory: 65/128 MB
Runtime: 1656.92 ms



Congratulations!

You've successfully created a custom coded workflow action that can connect and query a MySQL database hosted on AWS.



Resources

- [3 Ways to Use Custom Coded Workflow Actions - Developer blog post](#)
- [Use Custom Coded Actions in Workflows - Knowledge base article](#)
- [Custom Coded Workflow Actions - Developer Documentation](#)
- [Programmable Automation Use Case Library](#)
- [Operations Hub Product Page](#)
- [Amazon relational Database Service \(RDS\) User Guide](#)
- Community Workshops
 - [Workshop #1](#)
 - [Workshop #2](#)
 - [Workshop #3](#)

There are a ton of useful resources online relating to operations hub and custom coded workflow actions specifically. On the left are some of the resources I would recommend reviewing to learn more. I'd also highly recommend checking out my blog post below

3 Ways to Use Custom Coded Workflow Actions

WITH HUBSPOT'S OPERATIONS HUB

HubSpot Developers

