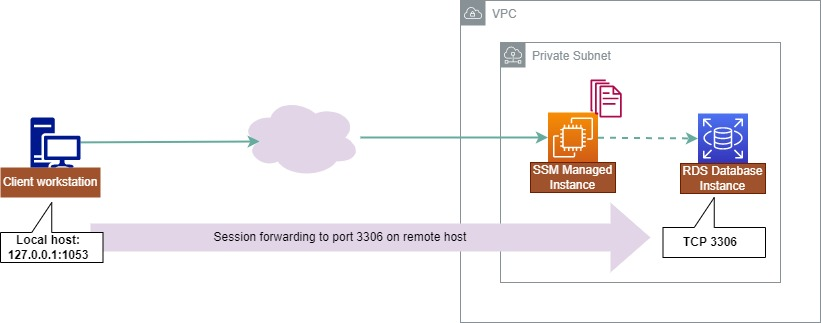
Connecting to RDS   
using the SSM Agent

# Introduction

This guide aims to help you connect to an RDS database in a private subnet using a public bastion instance with SSM agent as opposed to traditional SSH tunnelling. This enables developers to access the database remotely on their local machines using a database client without having to manage PEM files.



# Prerequisites

* A bastion EC2 instance set up in a public subnet with no inbound access and full outbound access. It must be managed by Systems Manager and have the correct permissions to use it.
* An RDS database set up in a private subnet with the security group of the bastion instance allowed in to access it on the port required by the database, e.g. 3306
* [AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html) installed on your local machine
* [Session Manager plugin](https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-working-with-install-plugin.html) installed for the AWS CLI
* A database client tool of your choosing ([DBeaver](https://dbeaver.io/download/) used here)

# Gather IDs

Next step is to go into the console and grab a few IDs.

Start by going to the EC2 console and selecting ‘Instances’ on the side. From here, find your bastion instance and note the ID of it - this will be a string in the format i-0123456789abcdef.

# Next, navigate to the RDS console and select ‘Databases’ from the left hand side, then find the database you want to connect to. Under the ‘Connectivity & Security’ tab, there will be an Endpoint which ends in rds.amazonaws.com, note this down as well.

# Authentication

Once you have the AWS CLI installed, you need to authenticate yourself before you can begin. This can be accomplished with normal authentication like running and inputting your access key and secret access key.

aws configure

# Testing the connection

In your terminal, run the following command to test that you can reach the bastion host:

aws ssm start-session --target {{instance\_id}}

where instance\_id is the instance ID noted in the first step. You should see a message that says

Starting session with SessionId: botocore-session-XXX

and then you are presented with a terminal. You can then run exit to quit this, as your connection is working as expected.

# Establishing the tunnel

Next, in the same terminal, run the following command:

aws ssm start-session \

--target {{instance\_id}} \

--document-name AWS-StartPortForwardingSessionToRemoteHost \

--parameters '{"portNumber":["{{port\_number}}"],"localPortNumber":["{{port\_number}}"],"host":["{{database\_host\_name}}"]}'

You should see the same message as before, followed with

Port 3306 opened for sessionId botocore-session-XXX

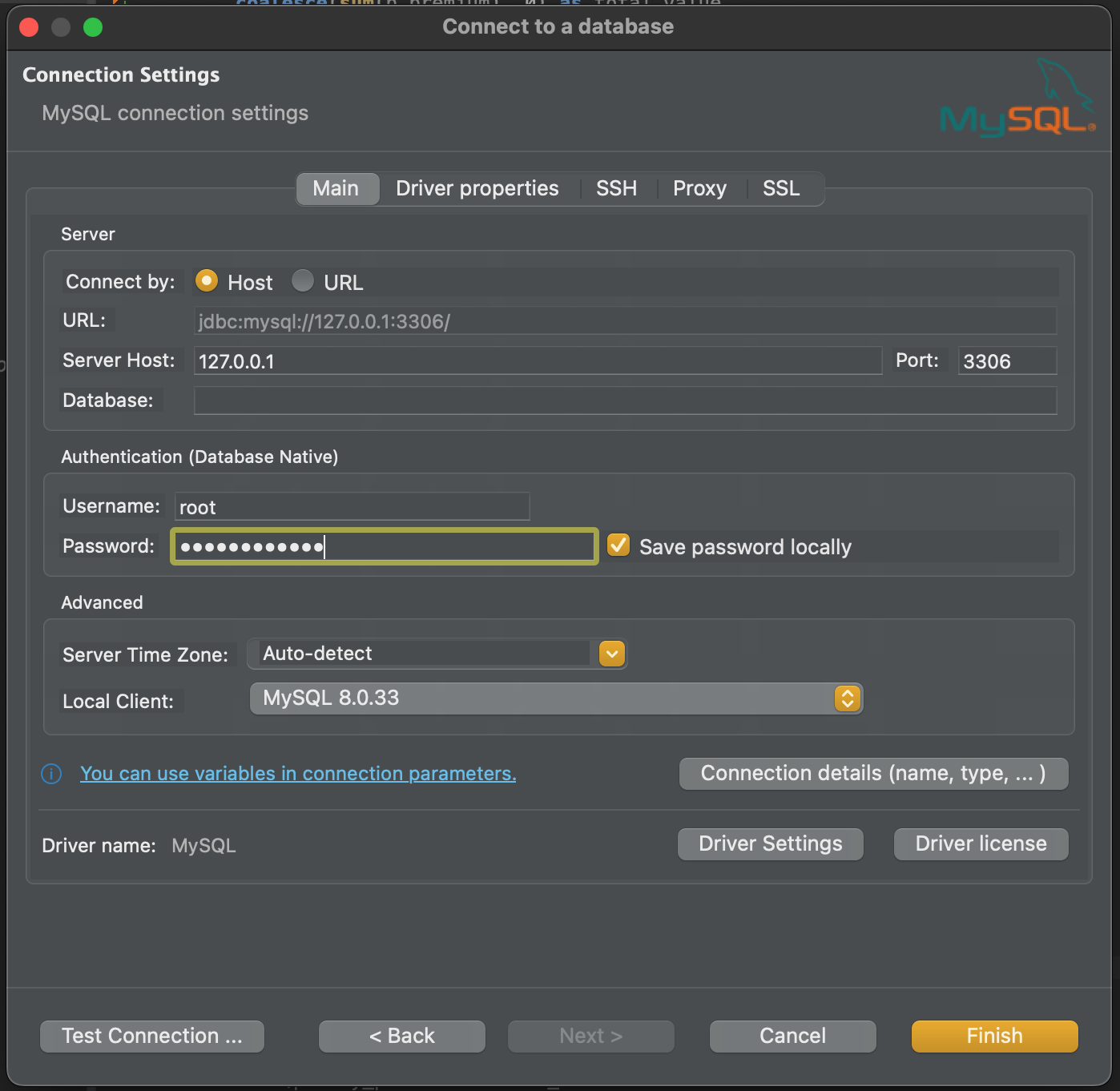
Waiting for connections...

Your tunnel is now up and running, ready to connect!

# Connecting to the database

In your database client, add a new connection with the following details:

* host = 127.0.0.1
* port = the port you forwarded to on your local machine
* username = your username
* password = your password



You can then test the connection and save it.

From here, you can interact with the database as usual.

Future connections will require

* Port forwarding started with

aws ssm start-session \

--target {{instance\_id}} \

--document-name AWS-StartPortForwardingSessionToRemoteHost \

--parameters '{"portNumber":["{{port\_number}}"],"localPortNumber":["{{port\_number}}"],"host":["{{database\_host\_name}}"]}'

This can be bundled up in a bash script to save time if desired.