## What is Lens:

Lens Allows us to visualize and interact with Kubernetes clusters and dockers easily, which helps us accelerate development, operations, and learning. The lens is available for Linux, Mac, and Windows so that you can choose which OS you want to use.

## Prerequisites For Getting started with Lens:

1. Lens Package Installed on your Machine.
2. AWS CLI  and CMD Administrator Access.

## Install Lens on Mac:

First and Foremost Step to get started with a lens is to download the lens IDE, Here is the download link for the lens.

Download Lens For Windows/Mac/Linux  From Here:

to Work With Lens you Should Interact with AWS CLI which is the Command-Line tool for AWS Access. So Now Let's go with the Installation and setting up the AWS CLI on the go.

## Install Lens from CMD on Mac:

1. There is One brew cask command which will do all the work for you.

```
> brew cask install lens
```

2. Make the Lens Directory and Download Binary File for the same.

```
> mkdir lens
> cd lens
> wget
https://github.com/lensapp/lens/releases/download/v3.5.2/Lens-3.5.2.AppImag
e
> chmod +x Lens-3.5.2.AppImage
> ./Lens-3.5.2.AppImage
```

### Install AWS CLI For Mac:

1. Download a CLI Package for AWS. From AWS CLI  For Mac: (It Will Download the package Instantly after Clicking the link).

2. Create the Symlink for AWS CLI. You should hit the command from the same folder you installed the package.

```
> sudo ln -s aws-cli/aws /usr/local/bin/aws
> sudo ln -s aws-cli/aws_completer /usr/local/bin/aws_completer
```

3. Verify the AWS-CLI Installation and make it work for you..

```
> which aws
> aws --version
```

## Install AWS CLI For Linux:

download the .zip file and then unzip the file, make sure that you have the unzip package installed in your Linux machine.

```
> curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
> unzip awscliv2.zip
> sudo ./aws/install
```

These 3 Commands will Download and unzip the aws cli package. So Now let's get started to install the aws cli.

```
> ./aws/install -i /usr/local/aws-cli -b /usr/local/bin
```

Now after Successful Installation, Let's check it with two commands.

```
> which aws
> aws --version
```

## Install AWS CLI For Windows:

Download AWS CLI  For Windows From Here:

After Downloading the MSI package for Windows Just got to the cmd and hit the Command.

```
> aws --version
```

It will Return the AWS Version and you are good to go with the AWS CLI In Windows.

# Configure AWS CLI Credentials:

**** we have shared the Developer credentials with you in the mail ****

There are 3 commands that will do the work for you.

```
> aws configure --profile <Profile name>/<project name>
```

Here Give the Credentials Assigned to you by DevOps Team.


```
> export AWS_PROFILE=<Your Profilename>/<project name>
```

**For Windows there are some changes here:**

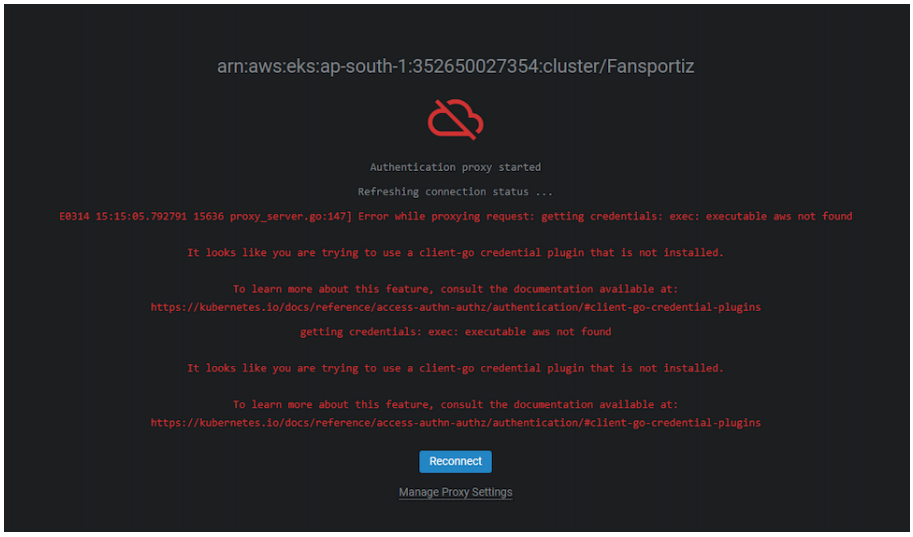
```
> set AWS_PROFILE=<Your Profilename>/<project name>
```

Now Run the AWS Command to Work With Our Cluster.

```
> aws eks --region ap-south-1 update-kubeconfig --name fantasy
```

# Troubleshoot Windows Related issues:

While you are working with the Windows system, there might be the same error as this screenshot, so here is the solution.




For Windows, You Should need to Install IAM Authenticator to configure the AWS eks. So For Windows, if you are getting issues related to IAM and permissions then you should [Follow This Document to Install aws-iam-authenticator]

## After Setting up the Lens:

In order to give access to other IAM Users in this section

- Clusters Master Access
- Read-Only Access ( for Developers )

**Create IAM User:**

[Creating an IAM user in your AWS account - AWS Identity and Access Management]

Give necessary permissions you want to give as per the access required by the user

**But in order to access EKS we must need to give this permission**

- **eksdescribecluster**
- **AmazonEKSClusterPolicy**

### Clusters Master Access:

**NOTE: if you modify this file or modify the existing content from this file then it can affect the cluster.**

In order to give clusters access to anyone, you have to create the **aws-auth-cm.yaml configmap in the Kube-system Namespace.**
Also in **order to set up the configmap from scratch**, you can follow the following steps mentioned in the document.
[Enabling IAM user and role access to your cluster - Amazon EKS]

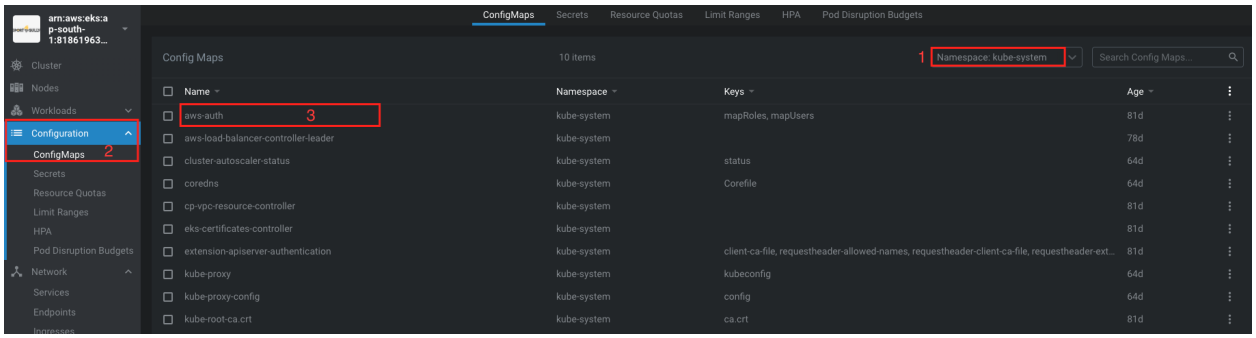
**Note:** In the fantasy cluster, we have already set up the **aws-auth-cm.yaml** configmap so you don't need to set up it.

In lens go to Kube-system namespace -> kube-system
Then go to **Configurations -> ConfigMaps ->** select the **aws-auth** configmap

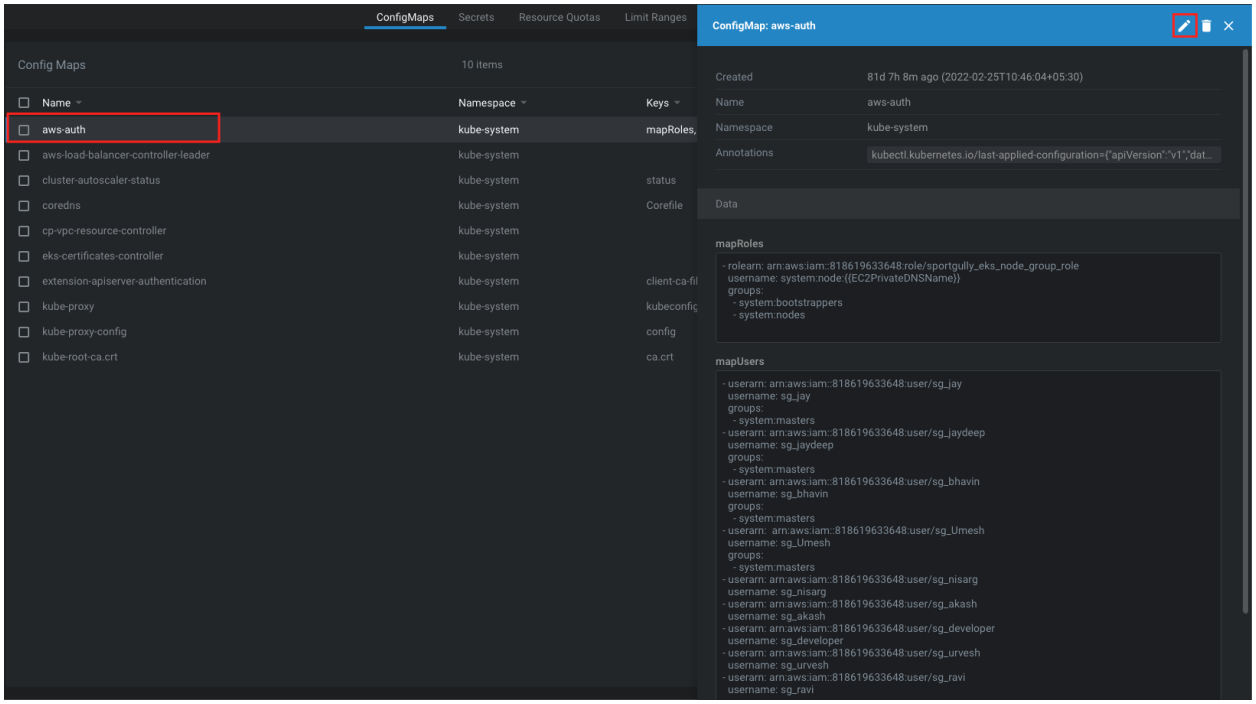
After selecting configmap and **click on that pen ( edit ) button to edit the configmap** and give them access.

As shown in the below picture.

1 - Select the namespace - **kube-system**
2 - After selecting the namespace go to the **Configurations -> ConfigMaps**
3 - Select the **aws-auth** configmap




Now as shown below image click on that pen button on the rightmost side to edit the file.




Here we can give 2 types of Access to clusters

- **Master Access ( Super Users Access )**
- **Access on basis of Cluster Role and Role.**

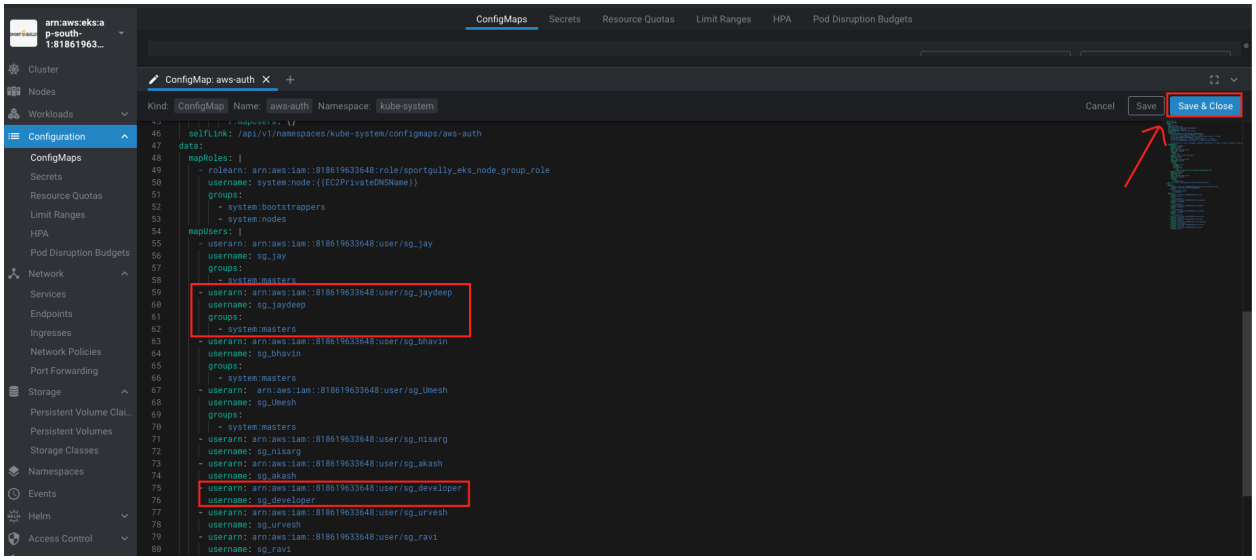
### Master Access

In order to give master access we have to include the following parameters.

```
- userarn: arn of the IAM Username
    username: IAM-username
    groups:
      - system:masters
```

### Access on basis of Cluster Role and Role

```
- userarn: arn of the IAM Username
    username: IAM-username
```




# Port forwarding from Lens and DB access:

**We can port forward using 2 ways**

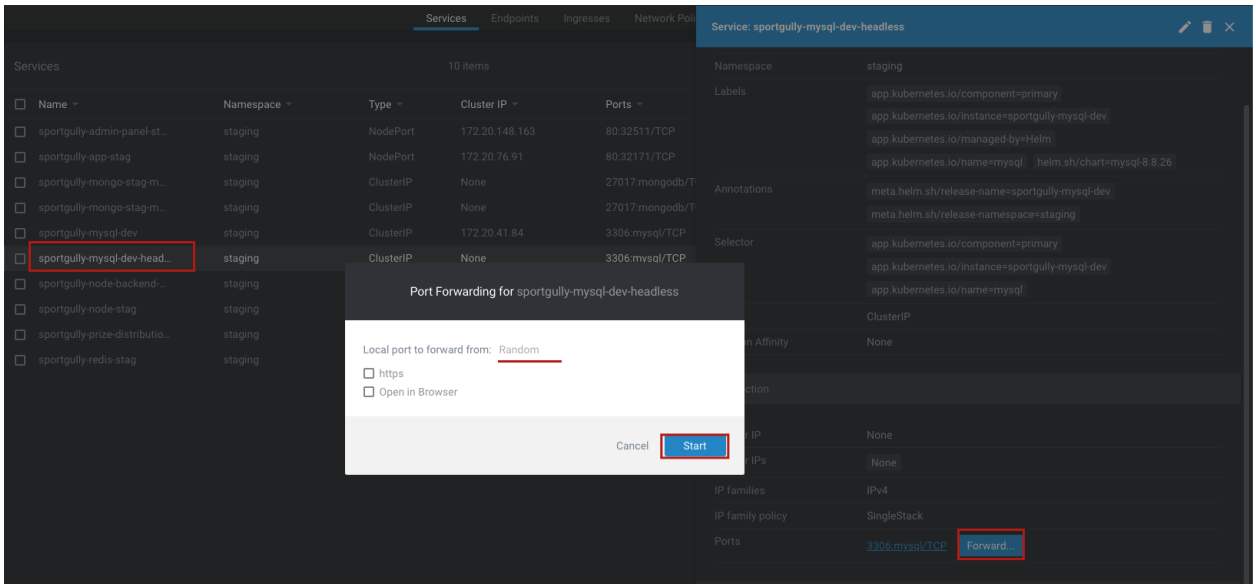
**1 - headless service**
**2 - Pods port forwarding**

## Headless service:

**Step: 1**

**Go to lens -> expand the network section -> services -> click on fantasy-mysql-dev-headless**

As seen in the below picture select the headless service and port forward that service.



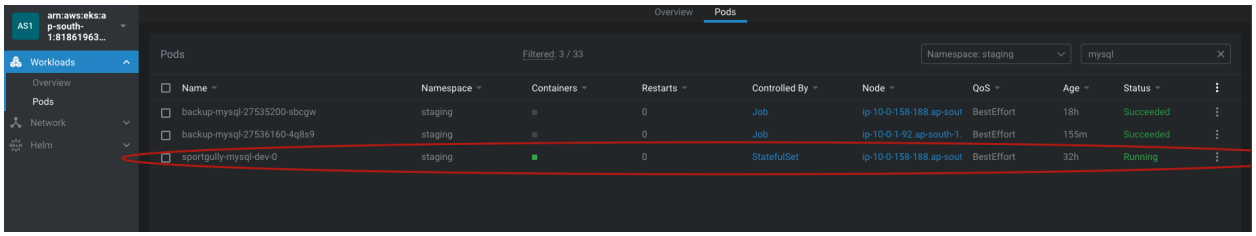

**And do the same process as mentioned in Step2**

**PODS Port Forwarding:**

If you want to access the staging DB from the Local machine then please follow the following steps.

**Step: 1**

**Go to lens -> expand the workload section -> Pods -> click on fantasy-mysql-dev-0 pod**

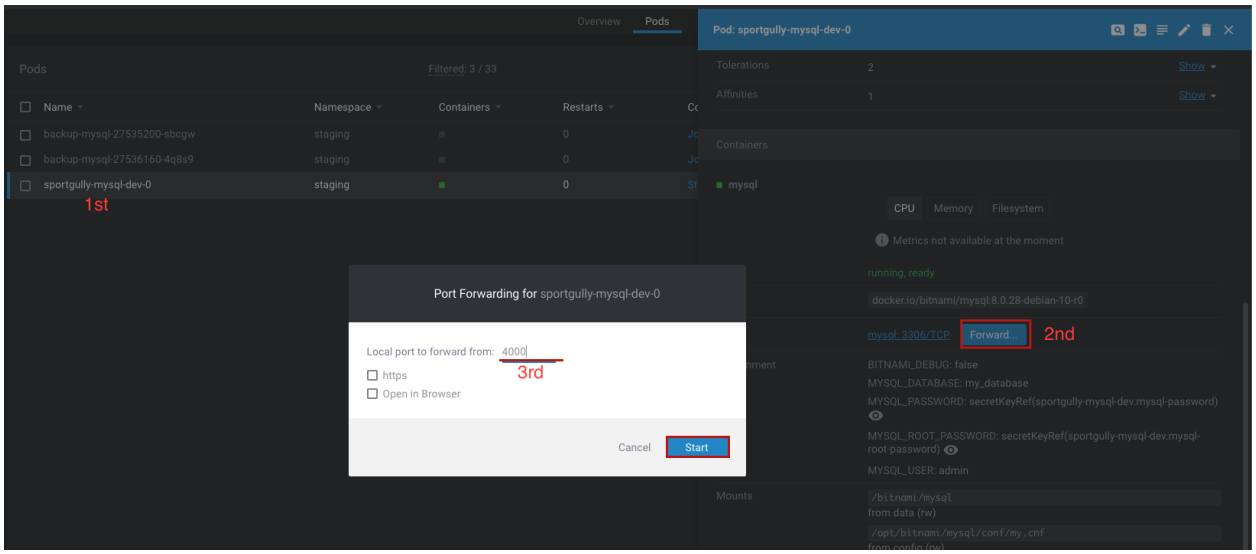



**Step: 2**

**As shown in the below picture**

**1st** - Click on that pod
**2nd** - Scroll a bit to the bottom and click on the forward button
**3rd** - Then it will open the dialogue box like this and you have mentioned the **Local Port number on which you want to connect the DB.**

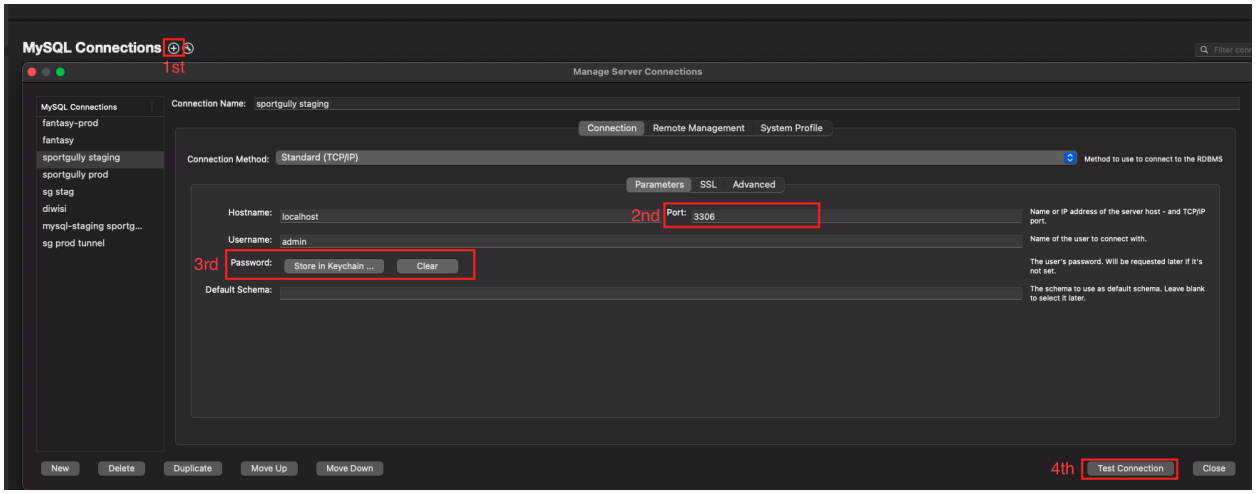
**Credentials**:
We mailed the credentials to you please check them once and if not received then contact your administration.



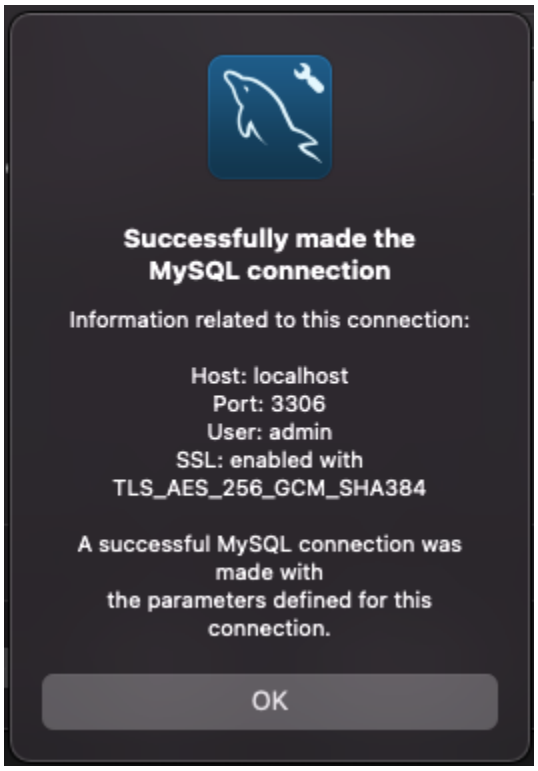

Then in order to connect with the given credentials open **Mysql Workbench software** in order to connect and see the DB Records.

**Download**
https://dev.mysql.com/downloads/

After installing the MySQL workbench software.




**1st** - Click on the plus button to generate the connection.
**2nd** - enter the port number you have forwarded to.
**3rd** - enter the password you have received in the mail.
**4rd** - After entering all the details click on the Test Connection button.

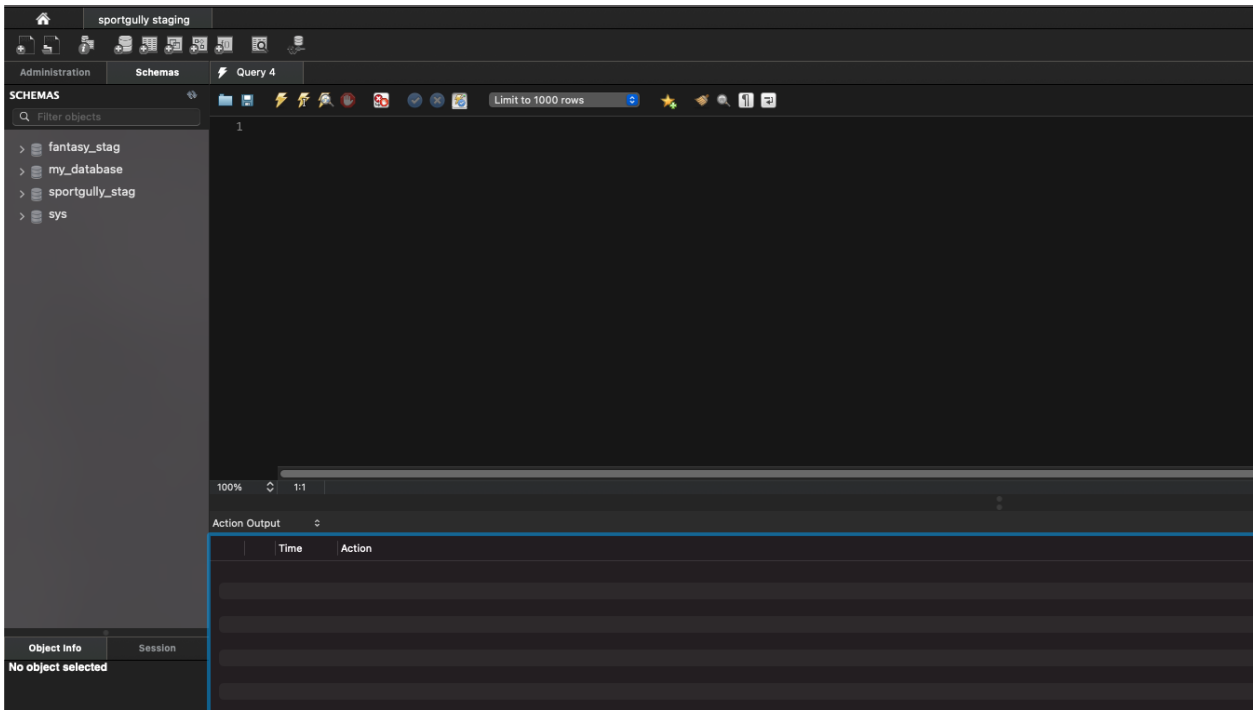


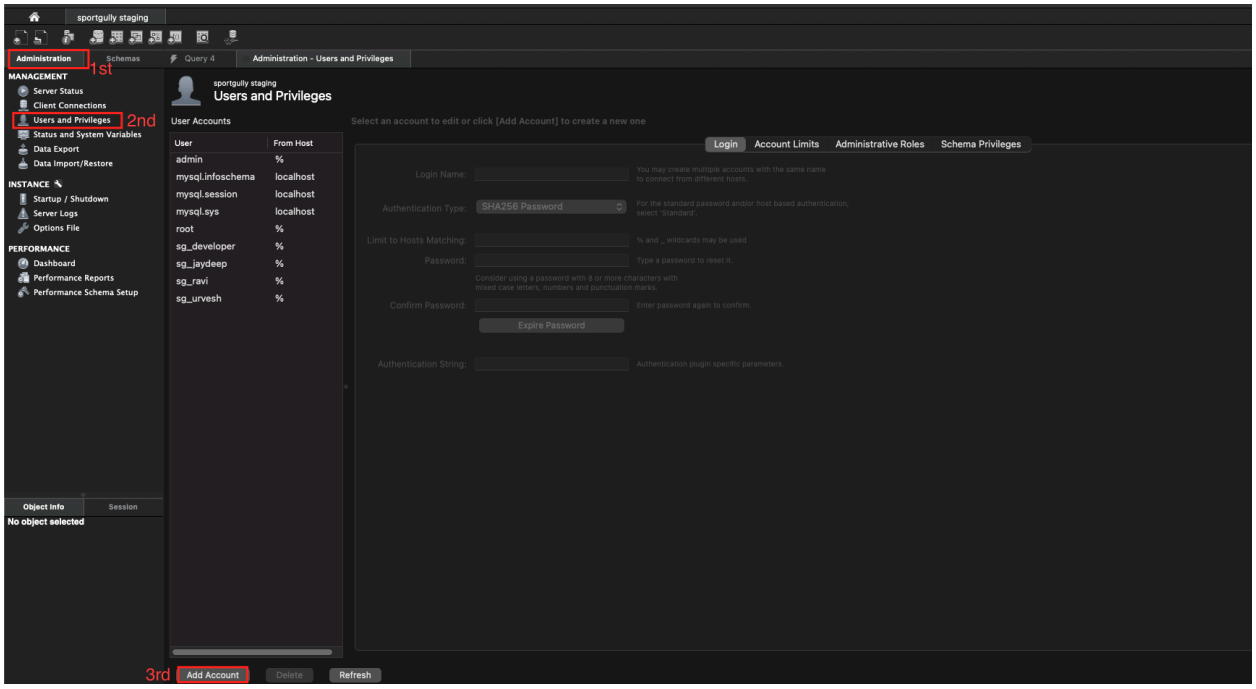

Your connection is successfully made with DB.

# Give MySQL access

After a successful connection with DB

Connect to the Database As shown in the below image type of window will open.






**1st** - After connecting, go to the Administration panel.
**2nd** - Then in order to give others access go to **Users and Privileges.**
**3rd** - Click on **Add Account** button
**4th** - After clicking on Add Account you can follow this doc to give access.
[Create new user using MySQL Workbench - Dot Net Tutorials]

## MongoDB

### Credentials

We mailed the credentials to you please check them once and if not received then contact your administration.

Follow the steps mentioned above in MySql DB for port forwarding and forward the port of pod - **fantasy-mongo-stag-mongodb-0.**

Now after forwarding the port you can **access the DB by MongoDB URI**

mongodb://username:**<password>**@localhost:**<port-you-have-forwarded>**/**<DB-name>**?authSource=admin&readPreference=primary&directConnection=true&ssl=false.

**Note: In this example, I am using the Robo3T tool for connecting mongoDB and giving access.**

In query section fire this query

```
db.createUser({
  user: '<<username>>',
  pwd: '<<password>>',
  roles: [
      { role: 'readWrite', db: 'fantasy_stag_users' },
      { role: 'readWrite', db: 'fantasy_stag_leagues' },
      { role: 'readWrite', db: 'fantasy_stag_notifications' },
      { role: 'readWrite', db: 'fantasy_stag_statistics' },
      { role: 'readWrite', db: 'fantasy_stag_banners' },
      { role: 'readWrite', db: 'fantasy_stag_complains' },
      { role: 'readWrite', db: 'fantasy_stag_tips' },
      { role: 'readWrite', db: 'fantasy_stag_promocodes' },
      { role: 'readWrite', db: 'fantasy_stag_admins' },
      { role: 'readWrite', db: 'fantasy_stag_geo' },
      { role: 'readWrite', db: 'fantasy_stag_game' },
      { role: 'readWrite', db: 'fantasy_stag_match' },
      { role: 'readWrite', db: 'fantasy_stag_teams' },
      { role: 'readWrite', db: 'fantasy_stag_report' },
      { role: 'readWrite', db: 'fantasy_stag_seriesLB' }

  ]
})
```

Replace the `<<username>> and <<password>>`

And after that you have created the **user.**

**Now we are moving towards access and giving creds for Production DB.**