

ORACLE

# MySQL Router

## REST API

---

Frédéric Descamps

Community Manager

MySQL

February 2021



# Who am I ?

 [about.me/lefred](https://about.me/lefred)



# Frédéric Descamps

- @lefred
- MySQL Evangelist
- Managing MySQL since 3.20
- devops believer
- living in Belgium 🇧🇪
- <https://lefred.be>



# What is it ?

 MySQL Router

# MySQL Router

MySQL Router is a building block for high availability (HA) solutions. It simplifies application development by intelligently routing connections to MySQL servers for increased performance and reliability.

MySQL Router is part of MySQL InnoDB Cluster and MySQL InnoDB ReplicaSet.

MySQL Router is written in C++ and is part of MySQL's trunk.





# MySQL Router

  
REST API

# MySQL Router REST API - Why ?

When a problem occurs, it's not always obvious to understand why ? For example if for a given route, the amount of `max_connections` is reached, it's important to know it and to know that value before we reach it.

MySQL Router exposes data (statistics, settings, ..) as REST endpoints via HTTP methods as JSON payload.

This is explained in WL#8965.



# MySQL Router REST API - How ?

Since MySQL 8.0.16, MySQL Router has the possibility to launch an internal http server. At that time it could only serve static files.

Then, in MySQL 8.0.17 we added the **REST API** to MySQL Router.

With MySQL 8.0.20 the authentication credentials to access the REST API could also be stored on MySQL in a metadata table, before it was only using a file. See WL#12952.

And finally with MySQL 8.0.22, bootstrapping a MySQL Router also configures the REST API functionality into the generated `mysqlrouter.conf` configuration file.



# Test it and get the paths

To query `swagger.json` no authentication is required:

```
[fred@imac ~] $ curl -s -k https://[REDACTED]:8443/api/20190715/swagger.json | jq '.paths | keys'
```

```
[  
  "/metadata",  
  "/metadata/{metadataName}/config",  
  "/metadata/{metadataName}/status",  
  "/router/status",  
  "/routes",  
  "/routes/{routeName}/blockedHosts",  
  "/routes/{routeName}/config",  
  "/routes/{routeName}/connections",  
  "/routes/{routeName}/destinations",  
  "/routes/{routeName}/health",  
  "/routes/{routeName}/status"  
]
```

# MySQL Router REST API - Authentication

MySQL Router uses realms for authentication. The backend can be a file or a record in a metadata table:

```
[http_auth_backend:default_auth_backend]  
backend=metadata_cache
```

If you use a file, the `mysqlrouter_passwd` command-line utility must be used to generate and manage the users.



# MySQL Router REST API - Authentication (metadata)

So there is a table in the metadata that we can use to connect to the REST API. Let's have a look...





# MySQL Router REST API - Authentication (metadata)

So there is a table in the metadata that we can use to connect to the REST API. Let's have a look...

```
MySQL single-mysql:33060+ mysql_innodb_cluster_metadata 2021-01-07 12:26:53
SQL desc router_rest_accounts;
```

Field	Type	Null	Key	Default	Extra
cluster_id	char(36)	NO	PRI	NULL	
user	varchar(256)	NO	PRI	NULL	
authentication_method	varchar(64)	NO		modular_crypt_format	
authentication_string	text	YES		NULL	
description	varchar(255)	YES		NULL	
privileges	json	YES		NULL	
attributes	json	YES		NULL	

```
7 rows in set (0.0017 sec)
```



# MySQL Router REST API - Authentication (metadata)

So there is a table in the metadata that we can use to connect to the REST API. Let's have a look...

```
MySQL > single-mysql:33060+ mysql_innodb_cluster_metadata 2021-01-07 12:26:53
SQL > desc router_rest_accounts;
```

Field	Type	Null	Key	Default	Extra
cluster_id	char(36)	NO	PRI	NULL	
user	varchar(256)	NO	PRI	NULL	
authentication_method	varchar(64)	NO		modular_crypt_format	
authentication_string	text	YES		NULL	
description	varchar(255)	YES		NULL	
privileges	json	YES		NULL	
attributes	json	YES		NULL	

7 rows in set (0.0017 sec)

So when we bootstrap a Router against a MySQL InnoDB Cluster or a MySQL InnoDB ReplicaSet, we need to add a user in this table.

# My journey to add a user in *router\_rest\_accounts*

Let me share with you my journey to insert a user into that table...





# My journey to add a user in *router\_rest\_accounts*

Let me share with you my journey to insert a user into that table...



# Adding a user in *router\_rest\_accounts* - Attempt #1

Let's try:

```
INSERT INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefred", "modular_crypt_format", "fosdem", NULL, NULL, NULL);
```



# Adding a user in *router\_rest\_accounts* - Attempt #1

Let's try:

```
INSERT INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefred", "modular_crypt_format", "fosdem", NULL, NULL, NULL);
```

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem https://[REDACTED]:8443/api/20190715/routes | jq  
[fred@imac ~] $
```



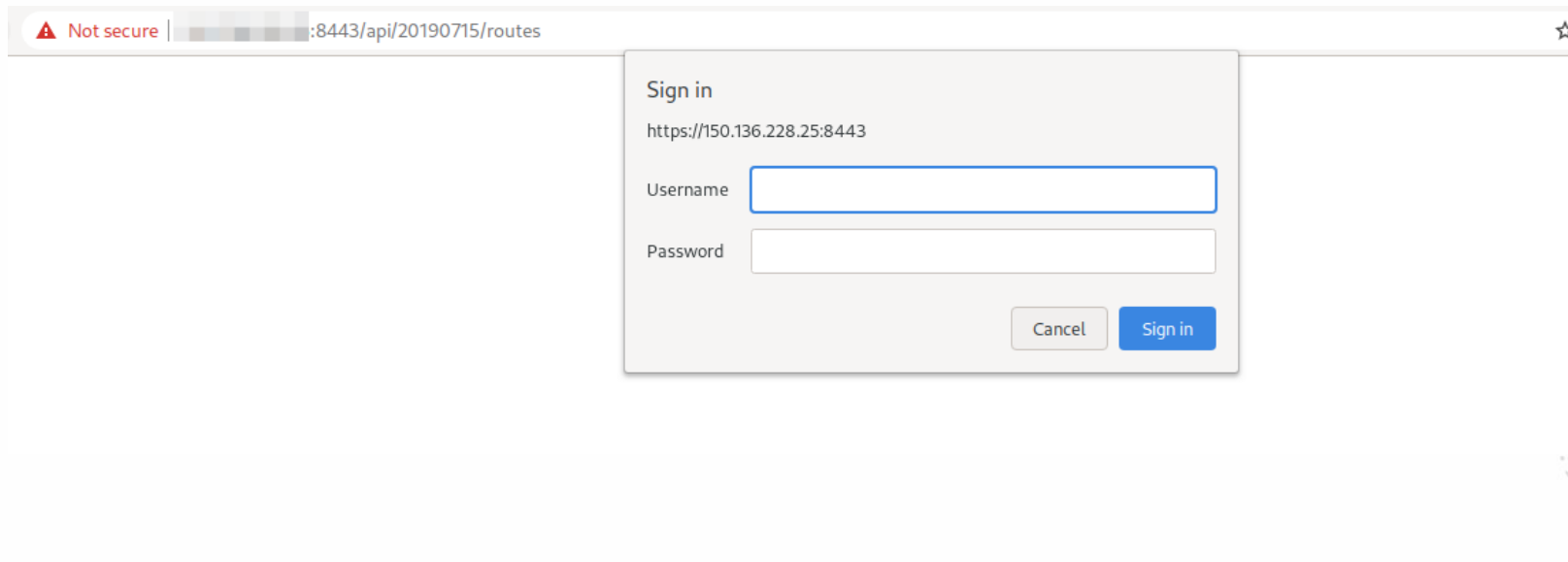


# Adding a user in *router\_rest\_accounts* - Attempt #1

Let's try:

```
INSERT INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefred", "modular_crypt_format", "fosdem", NULL, NULL, NULL);
```

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem https://[REDACTED]:8443/api/20190715/routes | jq  
[fred@imac ~] $
```



# lefred - security : 0 - 1



# Adding a user in *router\_rest\_accounts* - Attempt #2

For my second attempt, I asked to the router dev team for an example of string I could use.

The reply was simple 'copy' the value of your `authentication_string` column in the `mysql.user`:

```
SQL select user, plugin, authentication_string from mysql.user where user='clusteradmin';
```

user	plugin	authentication_string
clusteradmin	caching_sha2_password	\$A\$005\$ct/_qpOY7 gsp"KZGFtOg/guQTNURfbRT1w59C//WGO1xDwYpNaEjYT62



# Adding a user in *router\_rest\_accounts* - Attempt #2

For my second attempt, I asked to the router dev team for an example of string I could use.

The reply was simple 'copy' the value of your `authentication_string` column in the `mysql.user`:

```
SQL select user, plugin, authentication_string from mysql.user where user='clusteradmin';
```

user	plugin	authentication_string
clusteradmin	caching_sha2_password	\$A\$005\$ct/_qpOY7 gsp"KZGFtOg/guQTNURfbRT1w59C//WGO1xDwYpNaEjYT62

```
REPLACE INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefired", "modular_crypt_format",  
  (SELECT authentication_string FROM mysql.user WHERE user='clusteradmin'),  
  NULL, NULL, NULL);
```

# Adding a user in *router\_rest\_accounts* - Attempt #2

Let's test it:

```
[fred@imac ~] $ curl -s -k -u lefred:password https://[REDACTED]:8443/api/20190715/routes | jq '.items'
```

```
[
  {
    "name": "mycluster_ro"
  },
  {
    "name": "mycluster_rw"
  },
  {
    "name": "mycluster_x_ro"
  },
  {
    "name": "mycluster_x_rw"
  }
]
```

# Adding a user in *router\_rest\_accounts* - Attempt #2

Let's test it:

```
[fred@imac ~] $ curl -s -k -u lefred:password https://[REDACTED]:8443/api/20190715/routes | jq '.items'
```

```
[
  {
    "name": "mycluster_ro"
  },
  {
    "name": "mycluster_rw"
  },
  {
    "name": "mycluster_x_ro"
  },
  {
    "name": "mycluster_x_rw"
  }
]
```

It works... but I don't want to use a MySQL user to monitor Router!

# Adding a user in *router\_rest\_accounts* - Attempt #2

Let's test it:

```
[fred@imac ~] $ curl -s -k -u lefred:password https://[REDACTED]:8443/api/20190715/routes | jq '.items'
```

```
[
  {
    "name": "mycluster_ro"
  },
  {
    "name": "mycluster_rw"
  },
  {
    "name": "mycluster_x_ro"
  },
  {
    "name": "mycluster_x_rw"
  }
]
```

It works... but I don't want to use a **MySQL** user to monitor **Router** !

It's important to provide accurate information and context when you ask something...



# lfred - developer : 0 - 1



# Adding a user in *router\_rest\_accounts* - Attempt #3

My goal was to manage the credential for the REST API using a [MySQL Shell](#) Plugin...

I've been said that I should use the same string as a [MySQL](#) user... let's generate it !



# Adding a user in *router\_rest\_accounts* - Attempt #3

My goal was to manage the credential for the REST API using a [MySQL Shell](#) Plugin...

I've been said that I should use the same string as a [MySQL](#) user... let's generate it !

I created a program that does it:

```
[fred@imac ~/workspace/genauth_string] (main) $ ./myauth fosdem
MySQL caching_sha2_password authentication_string generator
=====
The authentication string to use is:
$A$005$zwM4!A72%s)](-ed8FN-euE7cwRygTogySEBOTQY46UbIU483McwXJYiovtbML2
```

# Adding a user in *router\_rest\_accounts* - Attempt #3

My goal was to manage the credential for the REST API using a [MySQL Shell](#) Plugin...

I've been said that I should use the same string as a [MySQL](#) user... let's generate it !

I created a program that does it:

```
[fred@imac ~/workspace/genauth_string] (main) $ ./myauth fosdem
MySQL caching_sha2_password authentication_string generator
=====
The authentication string to use is:
$A$005$zwM4!A72%s)](-ed8FN-euE7cwRygTogySEBOTQY46UbIU483McwXJYiovTbML2
```

```
REPLACE INTO router_rest_accounts VALUES (
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),
  "lefred", "modular_crypt_format",
  '$A$005$zwM4!A72%s)](-ed8FN-euE7cwRygTogySEBOTQY46UbIU483McwXJYiovTbML2',
  NULL, NULL, NULL);
```



# Adding a user in *router\_rest\_accounts* - Attempt #3

Let's test it:

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem https://[REDACTED]:8443/api/20190715/router/status | jq
{
  "processId": 17446,
  "productEdition": "MySQL Community - GPL",
  "timeStarted": "2021-01-07T13:37:53.982719Z",
  "version": "8.0.22",
  "hostname": "single-mysql"
}
```



# Adding a user in *router\_rest\_accounts* - Attempt #3

Let's test it:

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem https://[REDACTED]:8443/api/20190715/router/status | jq
{
  "processId": 17446,
  "productEdition": "MySQL Community - GPL",
  "timeStarted": "2021-01-07T13:37:53.982719Z",
  "version": "8.0.22",
  "hostname": "single-mysql"
}
```

It works... but I don't want to use an external program in my plugin, that's not very portable.



# lefred - usability : 0 - 1



# Adding a user in *router\_rest\_accounts* - Attempt #4

Mmm... let's try something else...





# Adding a user in *router\_rest\_accounts* - Attempt #4

Mmm... let's try something else...

A component !



# Adding a user in *router\_rest\_accounts* - Attempt #4

Mmm... let's try something else...

A component !

```
SQL select generate_auth_string('fosdem2',1);
```

```
+-----+  
| generate_auth_string('fosdem2',1) |  
+-----+  
| $A$005$op^'U#+}b)!n*~~3E<{+PzrzS/Rg9jU0.b4iMY56vET9E3rt2bws/BVw43zpTW8 |  
+-----+
```



# Adding a user in *router\_rest\_accounts* - Attempt #4

Mmm... let's try something else...

A component !

```
SQL select generate_auth_string('fosdem2',1);
```

```
+-----+  
| generate_auth_string('fosdem2',1) |  
+-----+  
| $A$005$op^'U#+}b)!n*~~3E<{+PzrzS/Rg9jUO.b4iMY56vET9E3rt2bws/BVw43zpTW8 |  
+-----+
```

```
REPLACE INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefred", "modular_crypt_format",  
  "$A$005$op^'U#+}b)!n*~~3E<{+PzrzS/Rg9jUO.b4iMY56vET9E3rt2bws/BVw43zpTW8",  
  NULL, NULL, NULL);
```

# Adding a user in *router\_rest\_accounts* - Attempt #4

Let's test it again:

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem2 https://[REDACTED]:8443/api/20190715/router/status | jq
{
  "processId": 17446,
  "productEdition": "MySQL Community - GPL",
  "timeStarted": "2021-01-07T13:37:53.982719Z",
  "version": "8.0.22",
  "hostname": "single-mysql"
}
```



# Adding a user in *router\_rest\_accounts* - Attempt #4

Let's test it again:

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem2 https://[REDACTED]:8443/api/20190715/router/status | jq
{
  "processId": 17446,
  "productEdition": "MySQL Community - GPL",
  "timeStarted": "2021-01-07T13:37:53.982719Z",
  "version": "8.0.22",
  "hostname": "single-mysql"
}
```

It works... but again, another component needs to be installed and maintained...

```
SQL select * from component;
```

component_id	component_group_id	component_urn
1	1	file://component_gen_auth



# lefred - usability : 0 - 2



# Adding a user in *router\_rest\_accounts* - Attempt #5

What else ?



# Adding a user in *router\_rest\_accounts* - Attempt #5

What else ?

Let me ask again with more context to the [MySQL Router](#) Development Team...



# Adding a user in *router\_rest\_accounts* - Attempt #5

What else ?

Let me ask again with more context to the [MySQL Router](#) Development Team...

Summary of the answer: we do support [MySQL](#) 8.0's default authentication string but also modular\_crypt\_format for MCF style password hashes as specified in the WL...



# Adding a user in *router\_rest\_accounts* - Attempt #5

What else ?

Let me ask again with more context to the [MySQL Router](#) Development Team...

Summary of the answer: we do support [MySQL](#) 8.0's default authentication string but also modular\_crypt\_format for MCF style password hashes as specified in the WL...

Oups...



# Adding a user in *router\_rest\_accounts* - Attempt #5

What else ?

Let me ask again with more context to the [MySQL Router](#) Development Team...

Summary of the answer: we do support [MySQL](#) 8.0's default authentication string but also modular\_crypt\_format for MCF style password hashes as specified in the WL...

Oups...

This means that the standard Python `crypt` module is what I need !

```
>>> import crypt
>>> crypt.crypt("fosdem3", crypt.mksalt(method=crypt.METHOD_SHA256))
'$5$tzRKNfEyehq1B5/Q$/PLKjs6PCEjNFjSDtdVZV2aL666SZrvKvrMWFYkIO82'
```



# lefred - developer : 0 - 2



# Adding a user in *router\_rest\_accounts* - Attempt #5

```
REPLACE INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefred", "modular_crypt_format",  
  "$5$tzRKNfEyehq1B5/Q$/PlKjs6PCEjNFiSDtdVZV2aL666SZrvKvrMWfYkIO82",  
  NULL, NULL, NULL);
```



# Adding a user in *router\_rest\_accounts* - Attempt #5

```
REPLACE INTO router_rest_accounts VALUES (  
  (SELECT cluster_id FROM mysql_innodb_cluster_metadata.v2_clusters LIMIT 1),  
  "lefred", "modular_crypt_format",  
  "$5$tzRKNfEyehq1B5/Q$/PlKjs6PCEjNFiSDtdVZV2aL666SZrvKvrMWfYkIO82",  
  NULL, NULL, NULL);
```

```
[fred@imac ~] $ curl -s -k -u lefred:fosdem3 https://[REDACTED]:8443/api/20190715/router/status | jq  
{  
  "processId": 17446,  
  "productEdition": "MySQL Community - GPL",  
  "timeStarted": "2021-01-07T13:37:53.982719Z",  
  "version": "8.0.22",  
  "hostname": "single-mysql"  
}
```

Woohooo \o/



# Examples

 MySQL Router REST API

# Some examples

We can now use the REST API with curl and include that in any monitoring tool like Sensu, Icinga...

```
[fred@lmac ~] $ curl -s -k -u lefred:fosdem3 https://[redacted]:8443/api/20190715/routes/mycluster_rw/connections | jq
{
  "items": [
    {
      "bytesFromServer": 22189,
      "bytesToServer": 771,
      "sourceAddress": "10.0.1.2:53108",
      "destinationAddress": "10.0.0.2:3306",
      "timeStarted": "2021-01-07T15:35:12.786968Z",
      "timeConnectedToServer": "2021-01-07T15:35:12.786979Z",
      "timeLastSentToServer": "2021-01-07T15:35:19.868770Z",
      "timeLastReceivedFromServer": "2021-01-07T15:35:19.869411Z"
    },
    {
      "bytesFromServer": 3445,
      "bytesToServer": 1018,
      "sourceAddress": "127.0.0.1:38154",
      "destinationAddress": "10.0.0.2:3306",
      "timeStarted": "2021-01-07T15:30:17.023359Z",
      "timeConnectedToServer": "2021-01-07T15:30:17.023385Z",
      "timeLastSentToServer": "2021-01-07T15:30:55.499357Z",
      "timeLastReceivedFromServer": "2021-01-07T15:30:55.500142Z"
    },
    {
      "bytesFromServer": 26607,
      "bytesToServer": 3824,
      "sourceAddress": "127.0.0.1:38130",
      "destinationAddress": "10.0.0.2:3306",
      "timeStarted": "2021-01-07T15:29:03.941722Z",
      "timeConnectedToServer": "2021-01-07T15:29:03.941761Z",
      "timeLastSentToServer": "2021-01-07T15:29:20.056723Z",
      "timeLastReceivedFromServer": "2021-01-07T15:29:20.058388Z"
    }
  ]
}
```



# MySQL Router & MySQL Shell

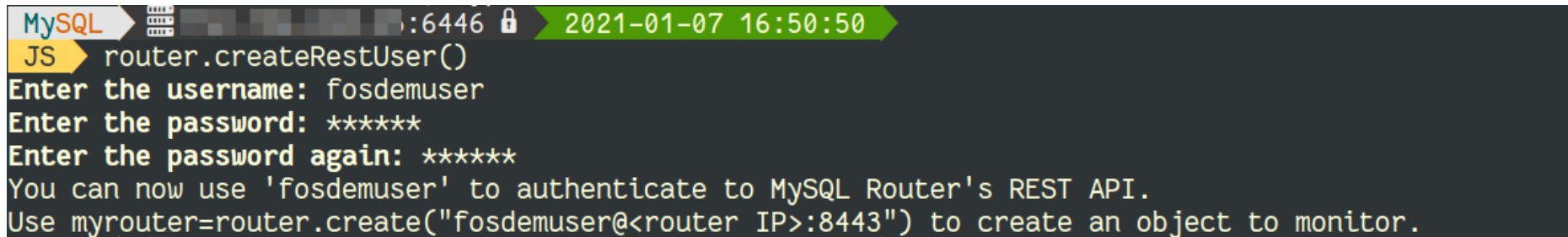
With MySQL Shell's router plugin (<https://github.com/lefred/mysqlshell-plugins>), you can take benefit of all this.



# MySQL Router & MySQL Shell

With **MySQL Shell**'s router plugin (<https://github.com/lefred/mysqlshell-plugins>), you can take benefit of all this.

Creating a user for the REST API:

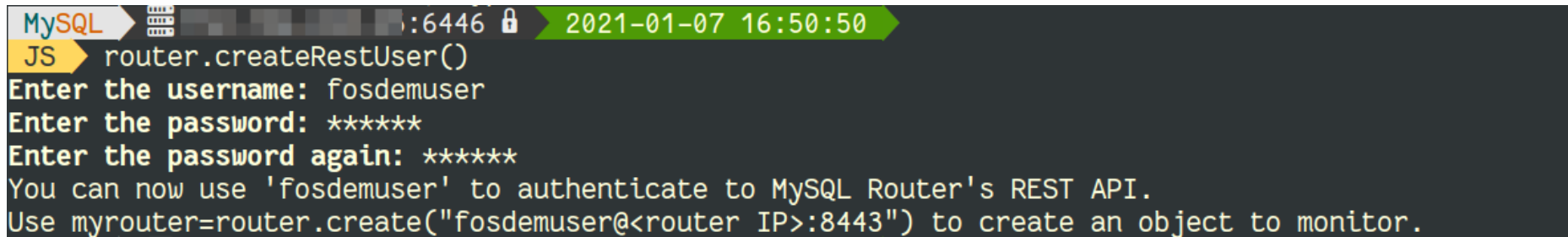


```
MySQL JS router.createRestUser()  
Enter the username: fosdemuser  
Enter the password: *****  
Enter the password again: *****  
You can now use 'fosdemuser' to authenticate to MySQL Router's REST API.  
Use myrouter=router.create("fosdemuser@<router IP>:8443") to create an object to monitor.
```

# MySQL Router & MySQL Shell

With **MySQL Shell**'s router plugin (<https://github.com/lefred/mysqlshell-plugins>), you can take benefit of all this.

Creating a user for the REST API:



```
MySQL 6446 2021-01-07 16:50:50
JS router.createRestUser()
Enter the username: fosdemuser
Enter the password: *****
Enter the password again: *****
You can now use 'fosdemuser' to authenticate to MySQL Router's REST API.
Use myrouter=router.create("fosdemuser@<router IP>:8443") to create an object to monitor.
```

Much easier ;) Convenient++

# MySQL Router & MySQL Shell

Creating the Router object:

```
MySQL JS myrouter=router.create("fosdemuser@:8443")
Password: *****
{
  "api": "20190715",
  "blockedHosts": <Function:<lambda>>,
  "connections": <Function:<lambda>>,
  "status": <Function:<lambda>>
}
```



# MySQL Router & MySQL Shell

```
JS ➤ myrouter.status()
+-----+
| Cluster name: mycluster |
+-----+
    Refresh Succeeded: 301
    Refresh Failed: 0
Last Refresh Hostname: single-mysql:3306
+-----+
| routes |
+-----+
* mycluster_ro (alive) :
  Routing Strategy: round-robin-with-fallback    Protocol: classic
  Total Connections: 6    Active Connections: 1    Blocked Hosts: 0
  ---> mysql-2 : 3306
  ---> mysql-3 : 3306
* mycluster_rw (alive) :
  Routing Strategy: first-available              Protocol: classic
  Total Connections: 533    Active Connections: 4    Blocked Hosts: 1
  ---> single-mysql : 3306
* mycluster_x_ro (alive) :
  Routing Strategy: round-robin-with-fallback    Protocol: x
  Total Connections: 0    Active Connections: 0    Blocked Hosts: 0
  ---> mysql-2 : 33060
  ---> mysql-3 : 33060
* mycluster_x_rw (alive) :
  Routing Strategy: first-available              Protocol: x
  Total Connections: 0    Active Connections: 0    Blocked Hosts: 0
  ---> single-mysql : 33060
```

# MySQL Router & MySQL Shell

We have an host blocked by MySQL Router, which one is it ?





# MySQL Router & MySQL Shell

We have an host blocked by MySQL Router, which one is it ?

```
JS myrouter.blockedHosts()
+-----+-----+
| Route      | Blocked Host(s) |
+-----+-----+
| mycluster_ro |                  |
+-----+-----+
| mycluster_rw | 10.0.1.2         |
+-----+-----+
| mycluster_x_ro |                  |
+-----+-----+
| mycluster_x_rw |                  |
+-----+-----+
```

# MySQL Router & MySQL Shell

And finally the routing statistics:

```
JS myrouter.connections()
```

Route	Source	Destination	From Server	To Server	Connection Started
mycluster_ro	127.0.0.1:36464	10.0.1.3:3306	109 kb	12 kb	2021-01-07T15:29:30.109637Z
mycluster_rw	10.0.1.2:53108	10.0.0.2:3306	21 kb	771 bytes	2021-01-07T15:35:12.786968Z
	109.128.188.66:44432	10.0.0.2:3306	2 kb	865 bytes	2021-01-07T17:18:45.192253Z
	127.0.0.1:38154	10.0.0.2:3306	3 kb	1018 bytes	2021-01-07T15:30:17.023359Z
	109.128.188.66:44962	10.0.0.2:3306	2 kb	865 bytes	2021-01-07T17:46:11.911202Z
	127.0.0.1:38130	10.0.0.2:3306	25 kb	3 kb	2021-01-07T15:29:03.941722Z
mycluster_x_ro					
mycluster_x_rw					

# Questions ?

