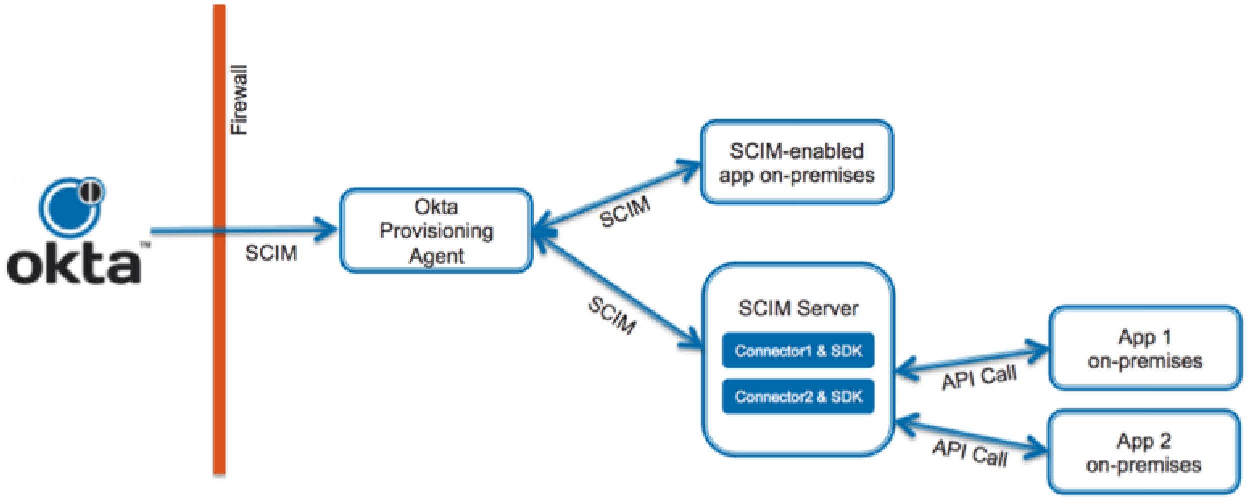
Generic SCIM Connector to synchronize between MySQL and Universal Directory

Use Case

In order to help customers import profile data, including passwords in MySQL, we recommend On-Prem Provisioning agent (OPP agent) to achieve such data import. OPP agent requires a SCIM connector which implements the CRUD operations required by the OPP agent, and often the SCIM connector is written to just satisfy a custom set of requirements from MySQL. This design and implementation strategy is unsustainable and requires the developers (both the customer and the solution provider) to update and redeploy almost all components when a change occurs. In order to achieve higher productivity and sustainability to handle changes from MySQL, a generic SCIM connector is created.

On-Prem Provisioning Architecture



A detailed description of the architecture can be found here: On-Prem Provisioning Architecture

Solution Overview

A generic scim connector will eliminate the burden of the developer from having to update and redeploy the connector to the scim server when new attribute(s) are added or existing one(s) removed in either MySQL or Universal Directory(UD). Thus the developer is not required to possess any Java development knowledge; only basic MySQL query skills and attribute mappings in Okta are required. Also, it is not limited to a specific table in MySQL, any number of tables required to produce a profile can be used.

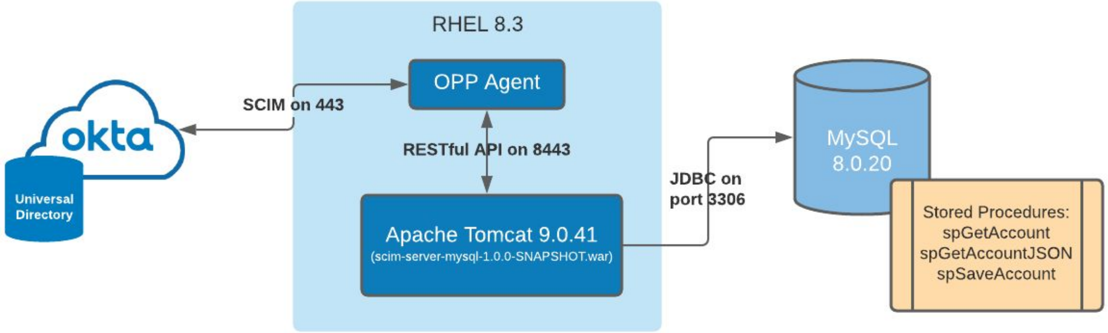
This document provides the instructions to achieve the data synchronization by leveraging the OPP agent, SCIM server and connector. Most of the install details are executed by a script to further reduce the time to prepare the environment for synchronization.

Please note that this instruction assumes the running operating system is Linux (Centos or Redhat). And this primarily targets the pre-sales personnel, so some steps may not follow the best practice, e.g. moving the Apache Tomcat to a generic folder and service enable it.

How it really works

Instead of relying on the scim connector to provide all the logic to render a User profile, which can face resource limitations like size of the scim server, why not leverage the power of the database to do all the heavy lifting and simply let the scim connector act as a bridge between the two repositories.

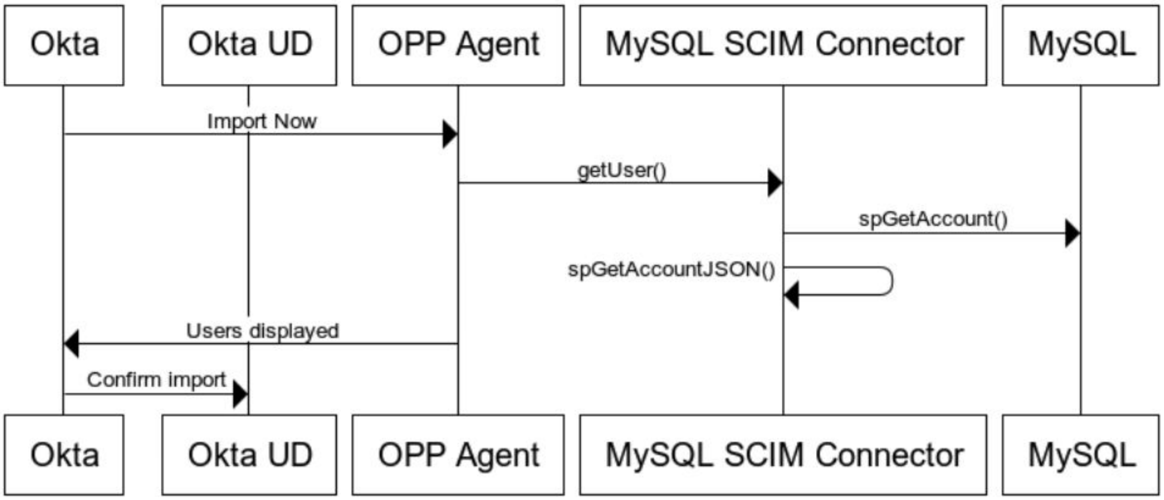
The following describes the deployment architecture of the solution



The following describes the details on each direction of synchronization:

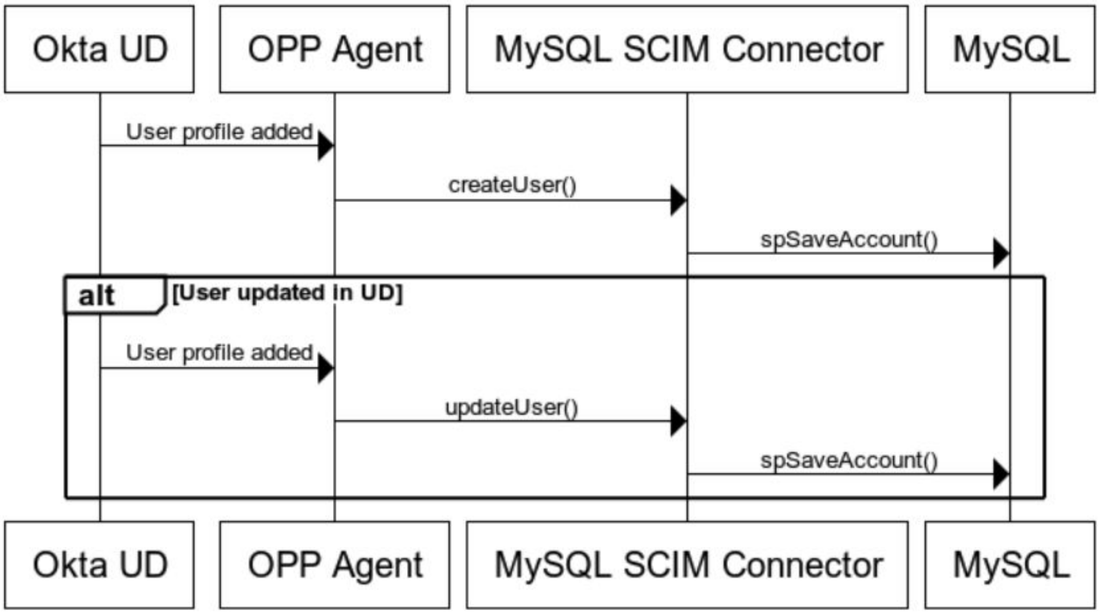
1. From MySQL to UD: MySQL procedure(s) to first retrieve all the user id (or some unique identifier for the users), then the scim connector loops through each id and make another call to another MySQL stored procedure to compose the JSON output of the user profile, the scim connector parses the JSON into User object. This is a very scalable approach to handle a large number of users, but this scim connector is not performant.

UD Synchronization from MySQL Flow



2. From UD to MySQL: the changes are basically instantaneous, the user JSON object is parsed into the scim connector and for each user, it calls a MySQL stored procedure to extract the needed attributes and save into the respective table(s).

UD Synchronization into MySQL Flow

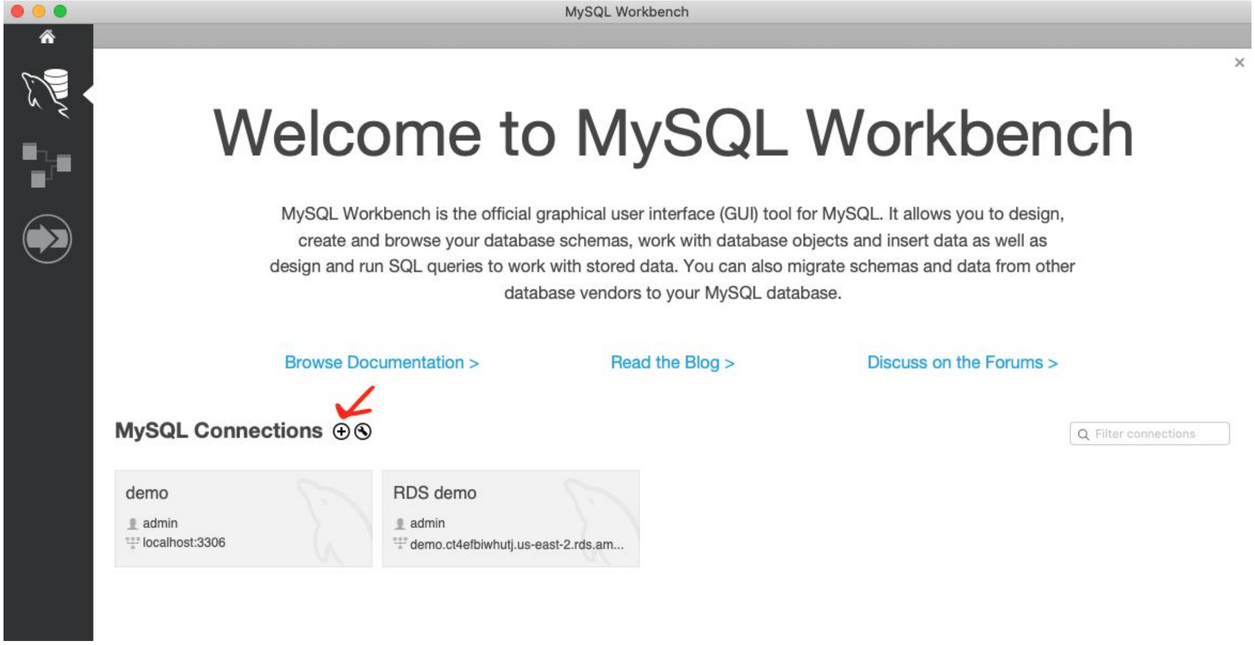


Prerequisite:

1. Obtain the latest version of the OPP agent by going to your desired org, then go to the Admin Console and select Settings -> Downloads, scroll down to find the Okta Provisioning Agent (x64 RPM).
2. Download the generic MySQL scim connector here: https://github.com/snaky912/okta/blob/master/scim-server-mysql/target/scim-server-mysql-1.0.0-SNAPSHOT.war
3. Download the java11.sh here: Script to install SCIM server for OPP
4. Obtain an Linux server, preferably a VM
5. SCP items from 1, 2 and 3 above to the VM, any folder is fine. In this document, since the VM is obtained via AWS EC2, /home/ec2-user is the default folder being referenced.
6. NOTE: You do not have to have your own MySQL, there is a default/common one available, please notify Karmen Lei if you do not want your own MySQL and skip the MySQL Setup below. Download and install a MySQL database, or obtain a MySQL RDS from AWS
7. Download MySQLWorkbench at https://dev.mysql.com/downloads/workbench/
8. Download the MySQL table and stored procedure scripts here: Sample Tables and Stored Procedures
9. Download the sample account data: Sample account data

Setting up MySQL using AWS MySQL RDS and MySQL Workbench

1. Open MySQLWorkbench, click “+” icon next the MySQL Connections to create a new connection to your database



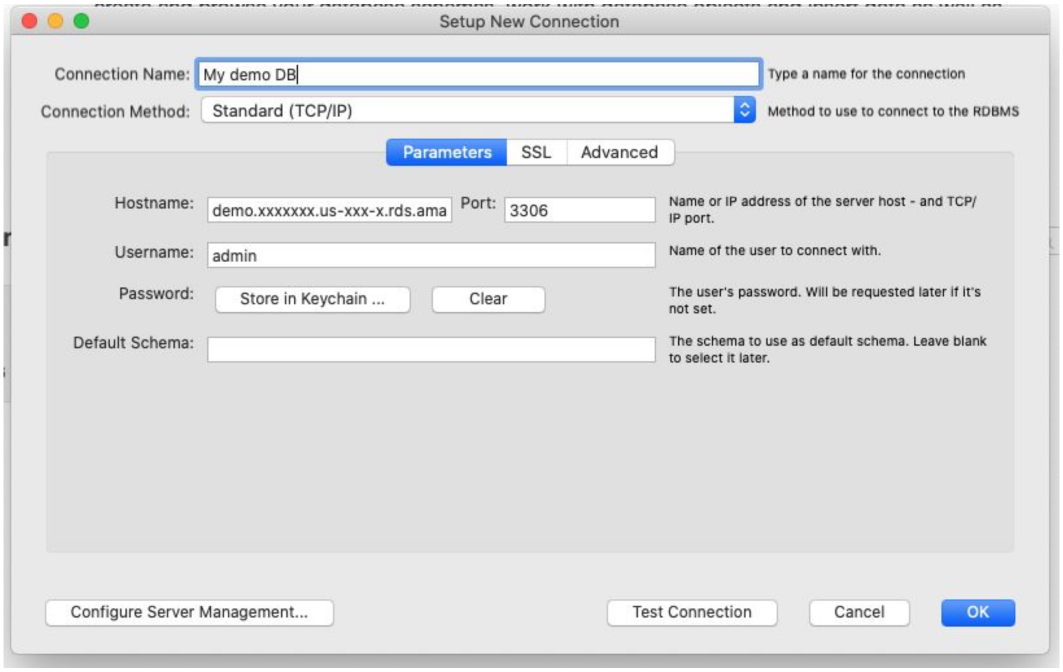
Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

demo
admin
localhost:3306

RDS demo
admin
demo.ct4efbiwhutj.us-east-2.rds.am...

2. Provide the connection and credential details, the username will be the “Master username” and password is the “Master password” when configuring the MySQL RDS instance



Setup New Connection

Connection Name: My demo DB — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

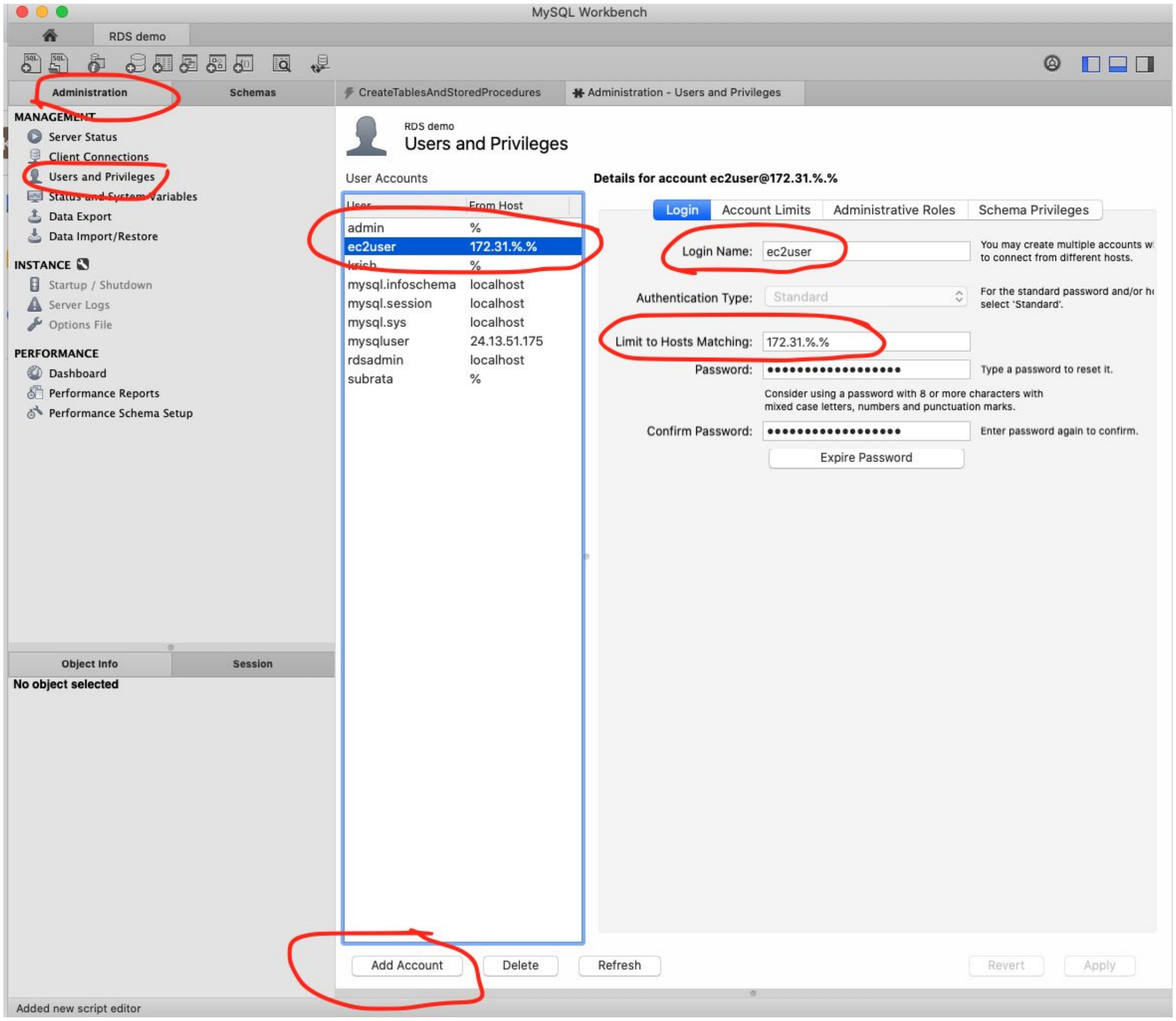
Hostname: demo.xxxxxxx.us-xxx-x.rds.ama Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: admin — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

3. Once connected, Click Administration tab and select Users and Privileges under Management, create a new user called “ec2user” with a password of “Passw0rd!!” and Limits to Hosts Matching to the IP of the Linux VM created earlier (“%” is wildcard)



User	From Host
admin	%
ec2user	172.31.%.%
krish	%
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
mysqluser	24.13.51.175
rdsadmin	localhost
subrata	%

Details for account ec2user@172.31.%.%

Login Name: ec2user
You may create multiple accounts w
to connect from different hosts.

Authentication Type: Standard
For the standard password and/or h
select 'Standard'.

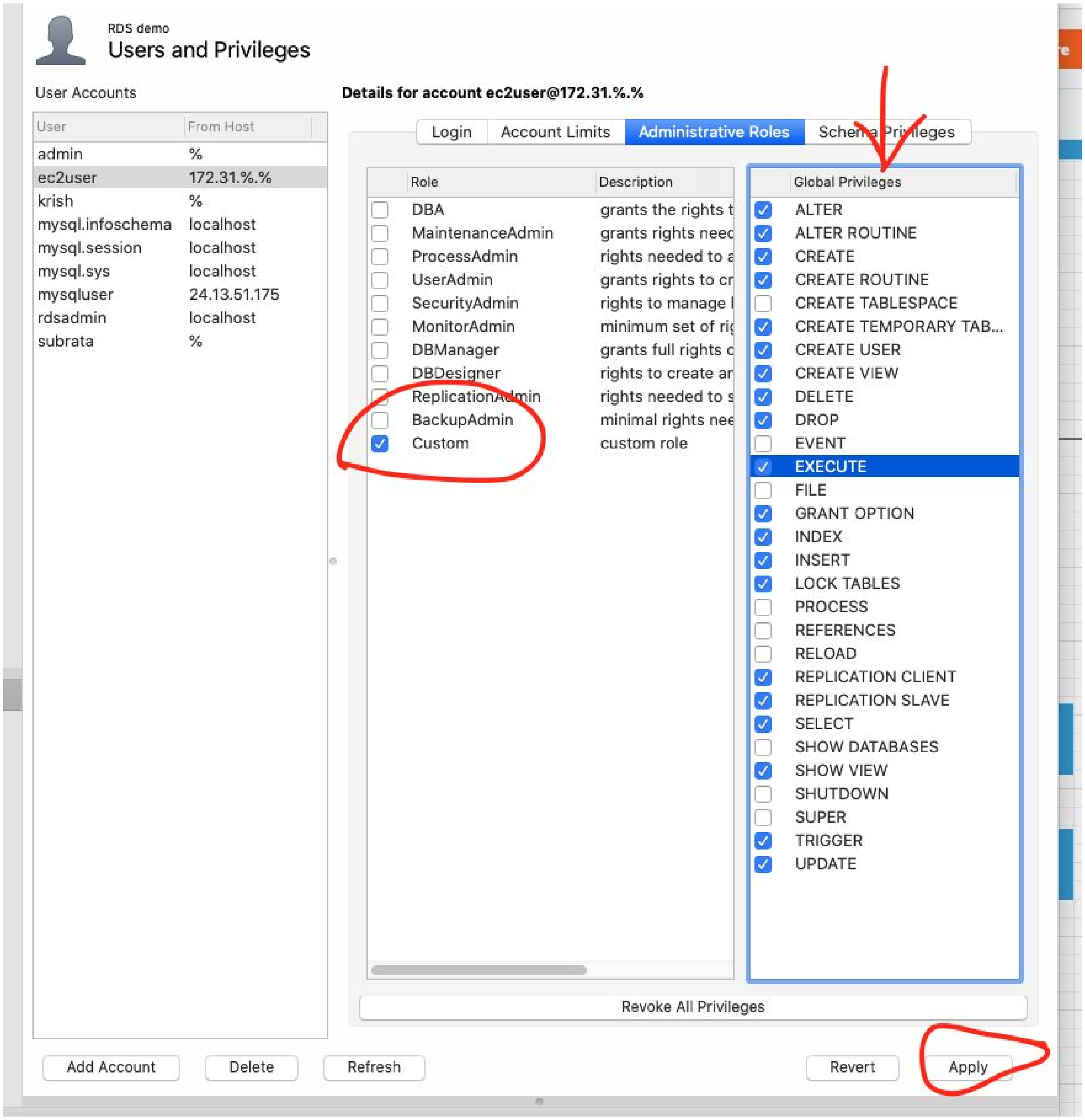
Limit to Hosts Matching: 172.31.%.%

Password: ••••••••••••••••••
Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: ••••••••••••••••••
Enter password again to confirm.

4. Click Administrative Roles and select Custom, enable the following privileges



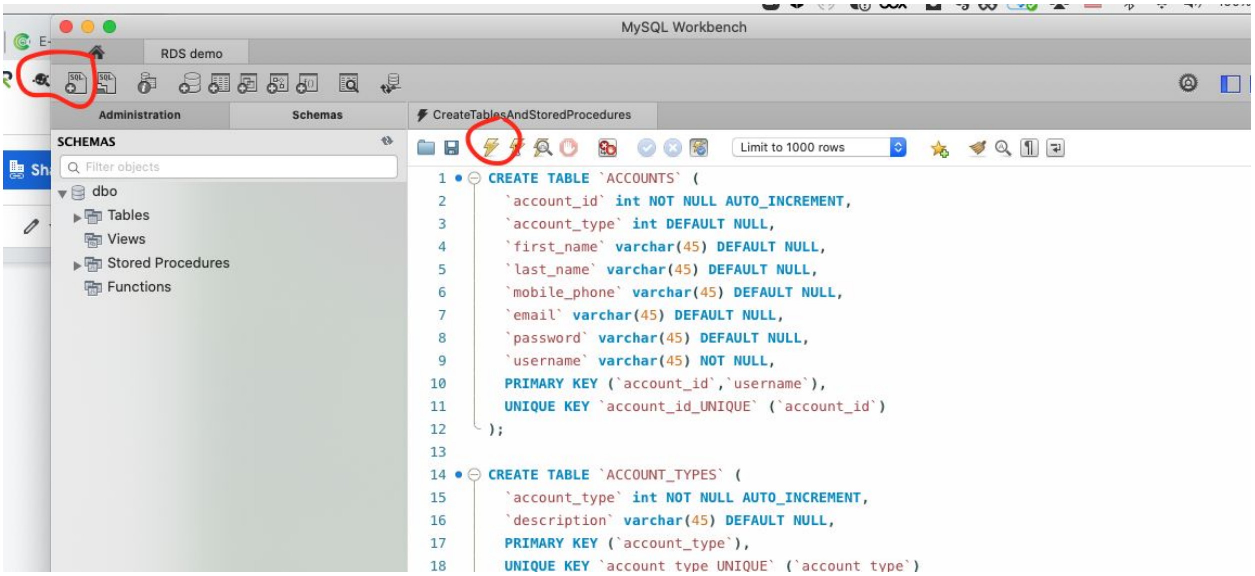
User	From Host
admin	%
ec2user	172.31.%.%
krish	%
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
mysqluser	24.13.51.175
rdsadmin	localhost
subrata	%

Details for account ec2user@172.31.%.%

	Role	Description
□	DBA	grants the rights t
□	MaintenanceAdmin	grants rights neec
□	ProcessAdmin	rights needed to a
□	UserAdmin	grants rights to cr
□	SecurityAdmin	rights to manage l
□	MonitorAdmin	minimum set of rig
□	DBManager	grants full rights c
□	DBDesigner	rights to create ar
□	ReplicationAdmin	rights needed to s
□	BackupAdmin	minimal rights nee
☑	Custom	custom role

	Global Privileges
☑	ALTER
☑	ALTER ROUTINE
☑	CREATE
☑	CREATE ROUTINE
□	CREATE TABLESPACE
☑	CREATE TEMPORARY TAB...
☑	CREATE USER
☑	CREATE VIEW
☑	DELETE
☑	DROP
□	EVENT
☑	EXECUTE
□	FILE
☑	GRANT OPTION
☑	INDEX
☑	INSERT
☑	LOCK TABLES
□	PROCESS
□	REFERENCES
□	RELOAD
☑	REPLICATION CLIENT
☑	REPLICATION SLAVE
☑	SELECT
□	SHOW DATABASES
☑	SHOW VIEW
□	SHUTDOWN
□	SUPER
☑	TRIGGER
☑	UPDATE

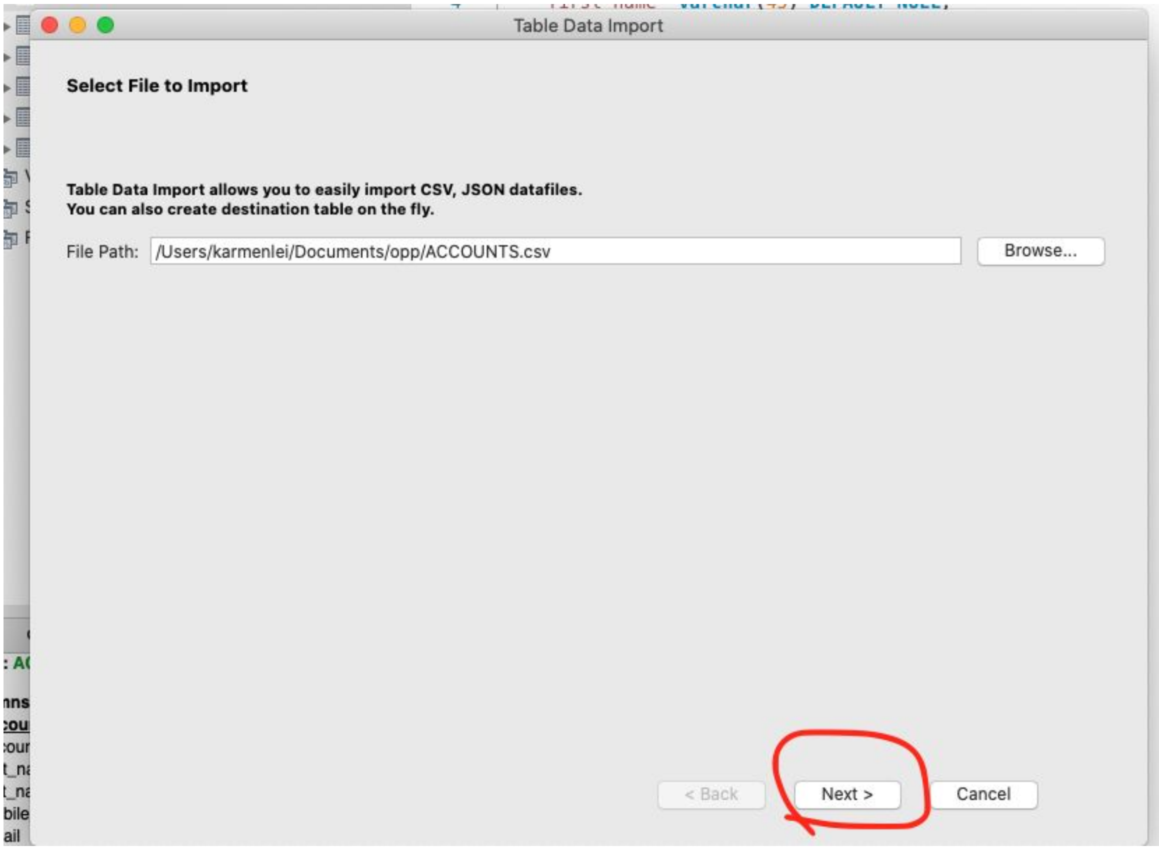
5. Click Schema tab, open the CreatesTablesAndStoredProcedures.sql, then execute it



```
CREATE TABLE `ACCOUNTS` (
  `account_id` int NOT NULL AUTO_INCREMENT,
  `account_type` int DEFAULT NULL,
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `mobile_phone` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  `username` varchar(45) NOT NULL,
  PRIMARY KEY (`account_id`,`username`),
  UNIQUE KEY `account_id_UNIQUE` (`account_id`)
);

CREATE TABLE `ACCOUNT_TYPES` (
  `account_type` int NOT NULL AUTO_INCREMENT,
  `description` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`account_type`),
  UNIQUE KEY `account_type_UNIQUE` (`account_type`)
```

6. Expand Tables or right click on Tables and select Refresh All, the tables should have been created. Right click on ACCOUNTS table and select Table Data Import Wizard, browse for the ACCOUNTS.csv downloaded from github, click Next several times till import is complete

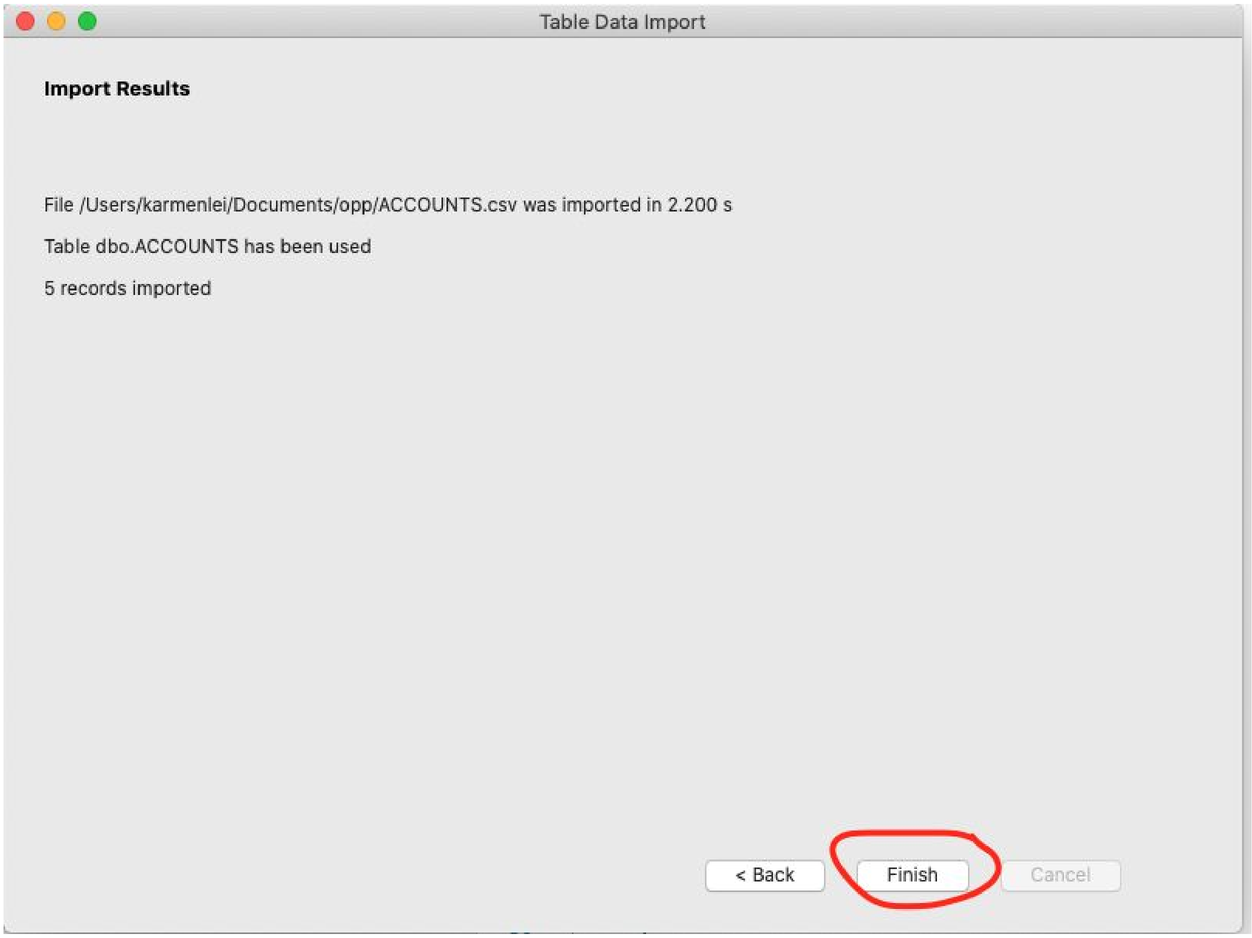


Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.

File Path: /Users/karmenlei/Documents/opp/ACCOUNTS.csv

7. Select Finish when import is complete



Import Results

File /Users/karmenlei/Documents/opp/ACCOUNTS.csv was imported in 2.200 s

Table dbo.ACCOUNTS has been used

5 records imported

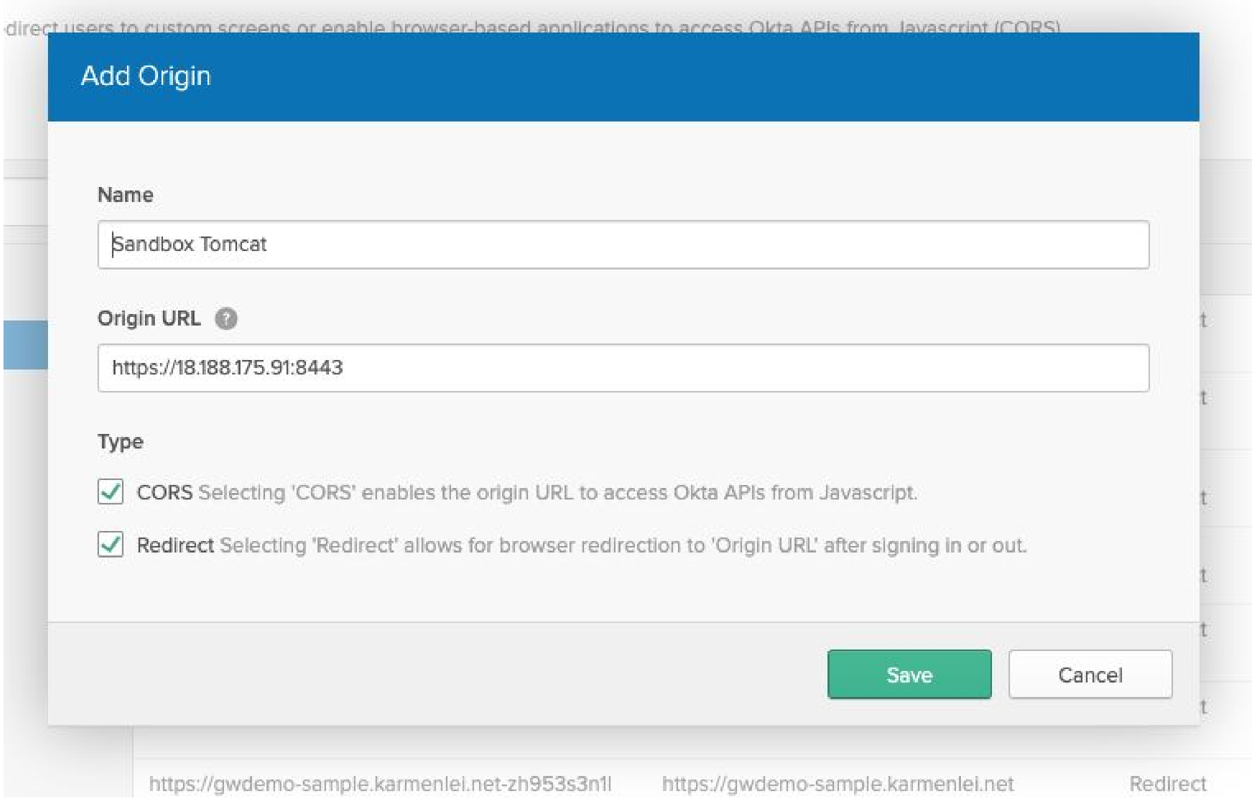
8. Your MySQL is ready

Installing OPP agent and Generic MySQL SCIM Connector

1. Go to the folder where all files are landed
2. Run “sudo yum -y localinstall <Okta provisioning agent rpm file name>”
3. Run “sudo /opt/OktaProvisioningAgent/configure_agent.sh” and follow the instructions to integrate the OPP agent with your desired org.
4. Make sure java11.sh is executable, if not, execute “chmod +x java11.sh”
5. Run “./java11.sh -u 11”
6. If you are using the default MySQL database, you can skip this step and jump to step 8. Run “sudo vi apache-tomcat-9.0.41/webapps/scim-server-mysql-1.0.0-SNAPSHOT/WEB-INF/dispatcher-servlet.xml”, look for “mySQLHost” and replace the value there with your own MySQL hostname.
7. Run “sudo apache-tomcat-9.0.41/bin/shutdown.sh” then “sudo apache-tomcat-9.0.41/bin/startup.sh”
8. You have just installed and configured your SCIM Server, the generic MySQL scim connector and OPP agent.

Configuring On-Prem Provisioning in Okta

1. Create a CORS policy for your SCIM server



direct users to custom screens or enable browser-based applications to access Okta APIs from Javascript (CORS)

Add Origin

Name

Sandbox Tomcat

Origin URL

https://18.188.175.91:8443

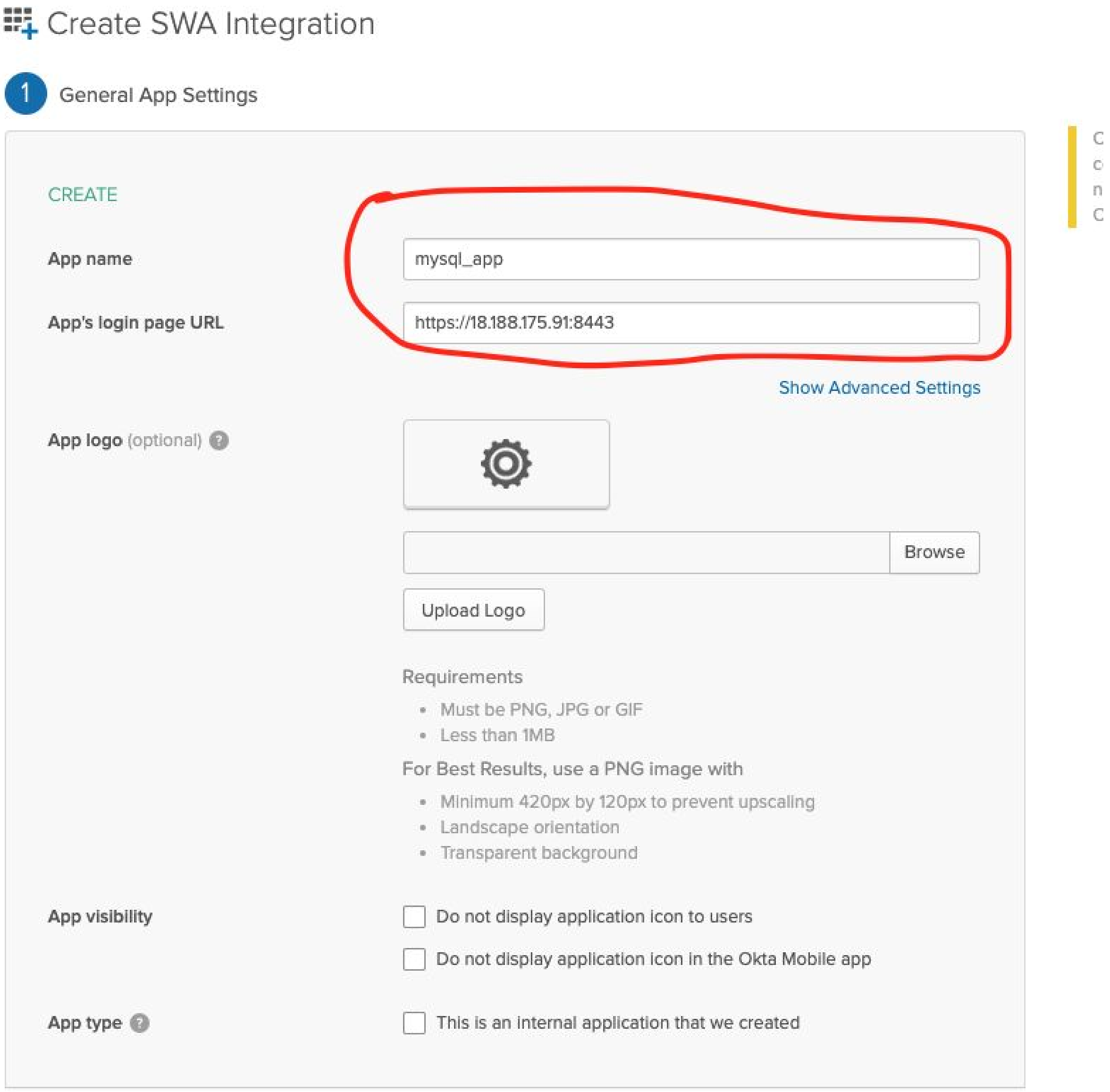
Type

☑ CORS Selecting 'CORS' enables the origin URL to access Okta APIs from Javascript.

☑ Redirect Selecting 'Redirect' allows for browser redirection to 'Origin URL' after signing in or out.

https://gwdemo-sample.karmenlei.net-zh953s3n1l	https://gwdemo-sample.karmenlei.net	Redirect

2. Create a new SWA application (Add Application -> Create New App -> Web -> SWA), then provide an App name and App’s login page URL, click Finish



Create SWA Integration

1 General App Settings

CREATE

App name: mysql_app

App's login page URL: https://18.188.175.91:8443

App logo (optional)

Requirements

- Must be PNG, JPG or GIF
- Less than 1MB

For Best Results, use a PNG image with

- Minimum 420px by 120px to prevent upscaling
- Landscape orientation
- Transparent background

App visibility

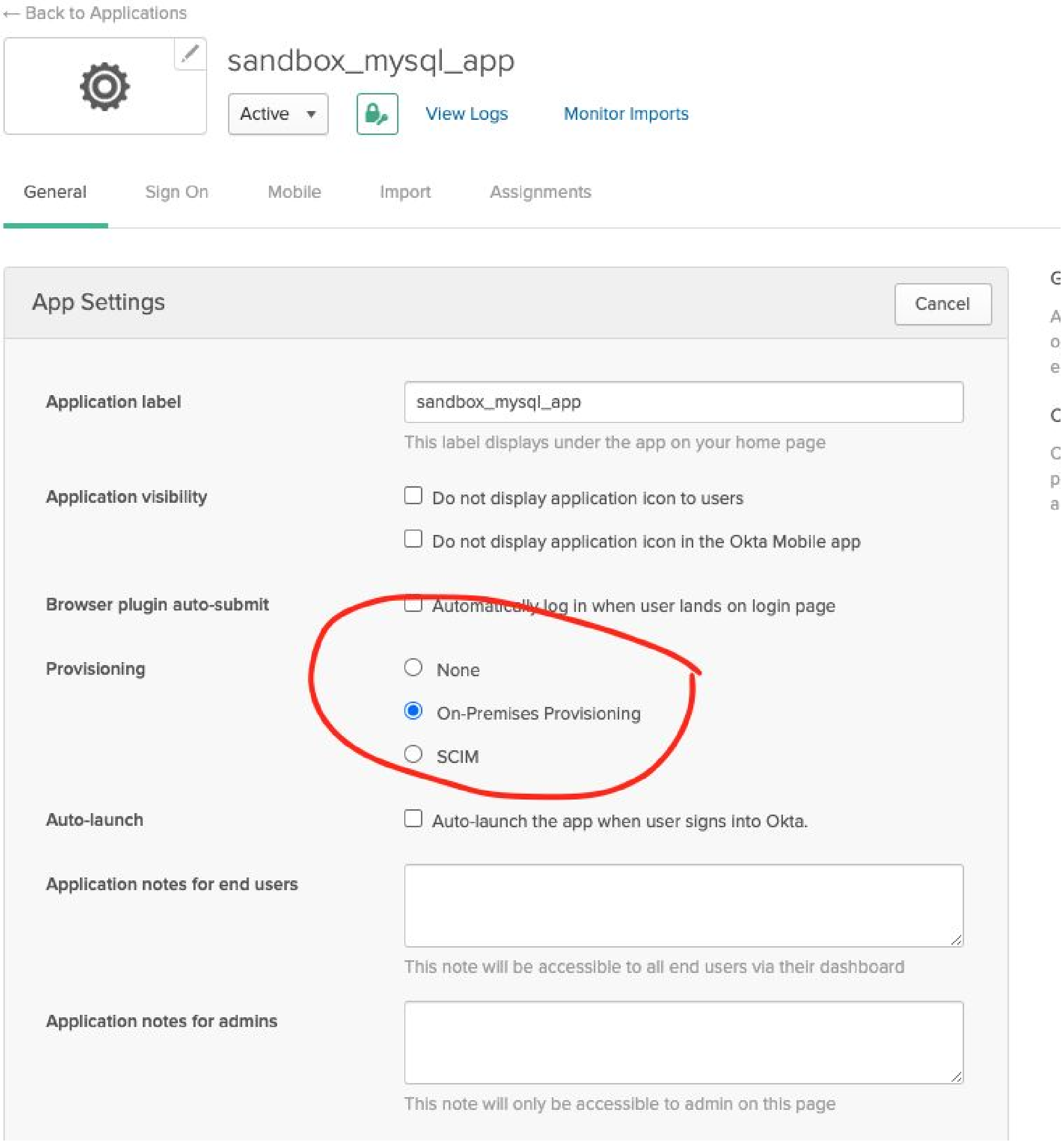
□ Do not display application icon to users

□ Do not display application icon in the Okta Mobile app

App type

□ This is an internal application that we created

3. Once the app is created, go to General tab and select Edit on App Settings, select "On-Premises Provisioning", then Save



App Settings

Application label: sandbox_mysql_app

This label displays under the app on your home page

Application visibility:
□ Do not display application icon to users
□ Do not display application icon in the Okta Mobile app

Browser plugin auto-submit: □ Automatically log in when user lands on login page

Provisioning:
○ None
● On-Premises Provisioning
○ SCIM

Auto-launch: □ Auto-launch the app when user signs into Okta.

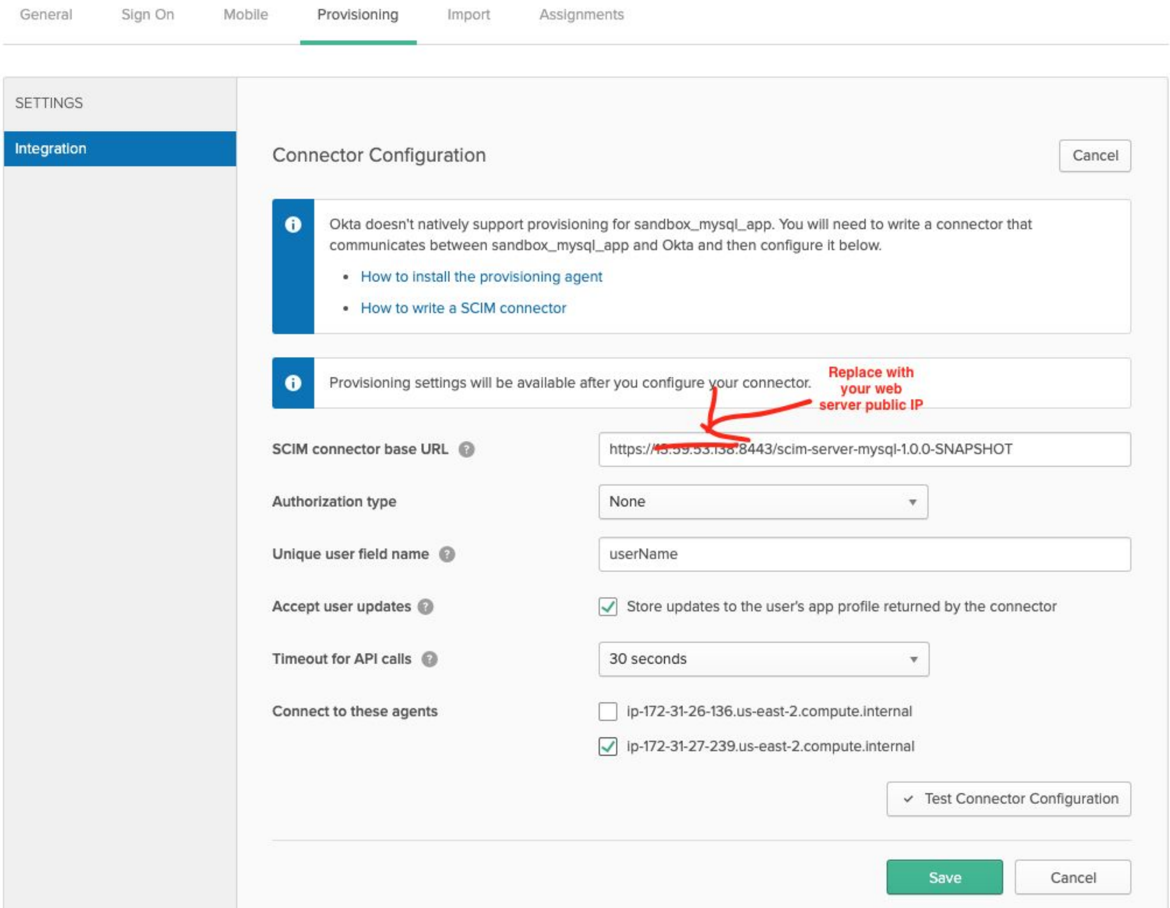
Application notes for end users:

This note will be accessible to all end users via their dashboard

Application notes for admins:

This note will only be accessible to admin on this page

4. Click Provisioning tab and click Configure SCIM Connector, provide the details like below, then click Test Connector Configuration



Connector Configuration

Okta doesn't natively support provisioning for sandbox_mysql_app. You will need to write a connector that communicates between sandbox_mysql_app and Okta and then configure it below.

- How to install the provisioning agent
- How to write a SCIM connector

Provisioning settings will be available after you configure your connector.

Replace with your web server public IP

SCIM connector base URL: https://~~13.59.53.138~~:8443/scim-server-mysql-1.0.0-SNAPSHOT

Authorization type: None

Unique user field name: userName

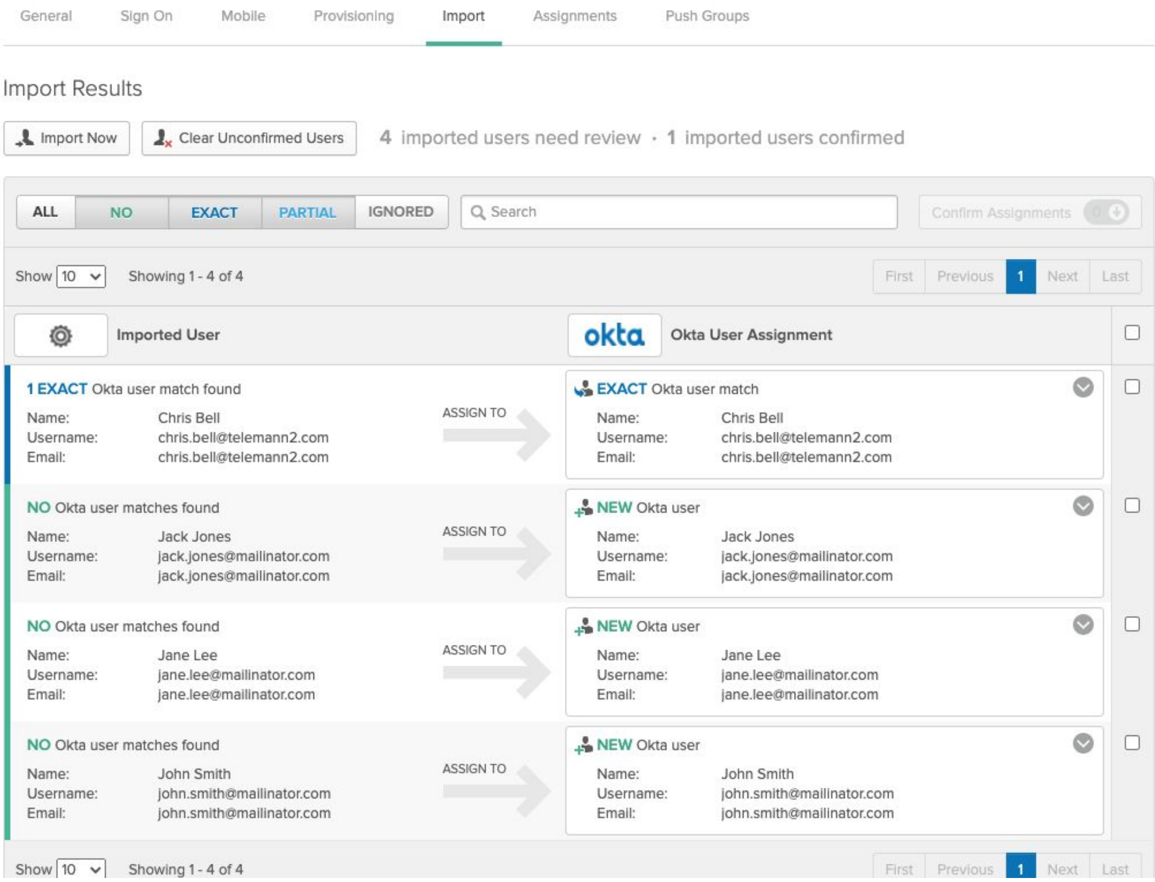
Accept user updates: ☑ Store updates to the user's app profile returned by the connector

Timeout for API calls: 30 seconds

Connect to these agents:
☐ ip-172-31-26-136.us-east-2.compute.internal
☑ ip-172-31-27-239.us-east-2.compute.internal

5. If there are no errors, click Save

6. Click Import tab and select Import Now



Import Results

4 imported users need review · 1 imported users confirmed

Show 10 Showing 1 - 4 of 4

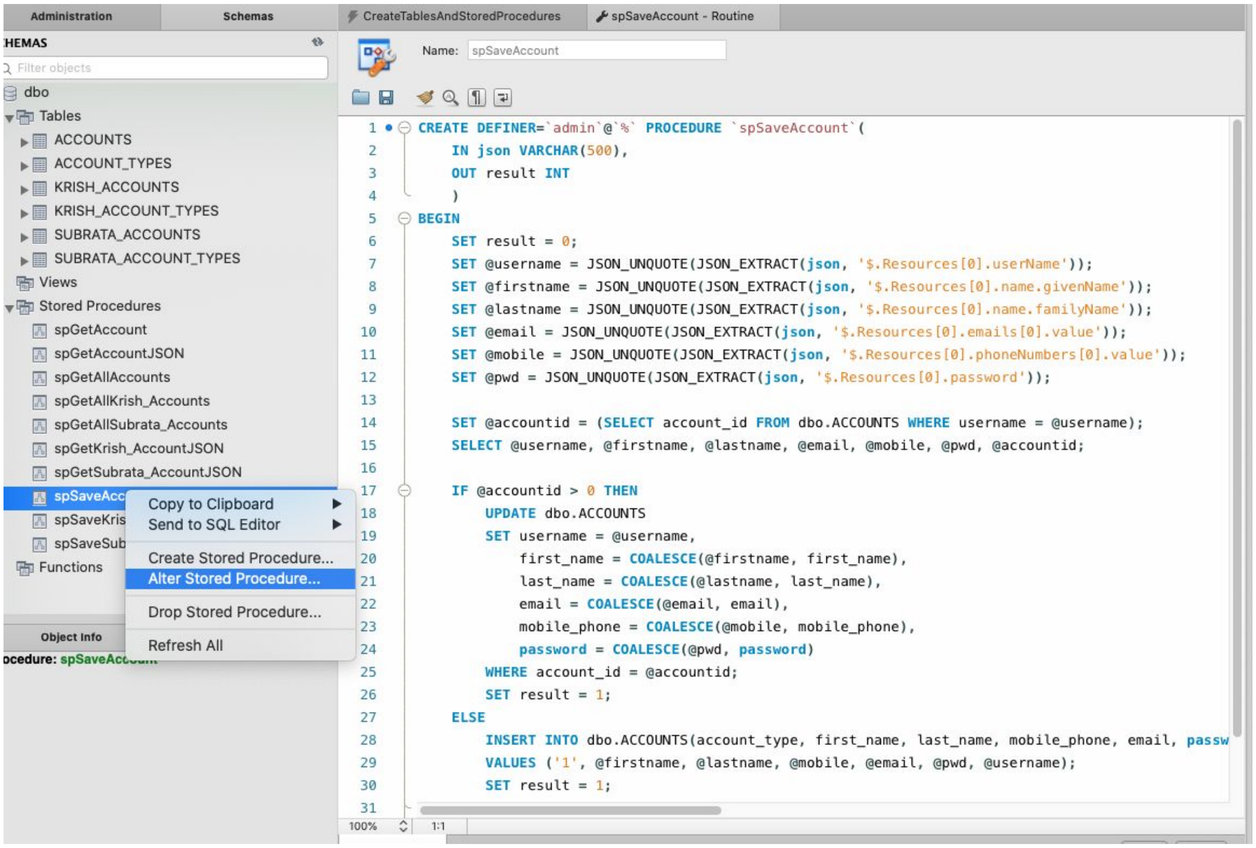
Imported User		Okta User Assignment
1 EXACT Okta user match found Name: Chris Bell Username: chris.bell@telemann2.com Email: chris.bell@telemann2.com	ASSIGN TO	**EXACT** Okta user match Name: Chris Bell Username: chris.bell@telemann2.com Email: chris.bell@telemann2.com
NO Okta user matches found Name: Jack Jones Username: jack.jones@mailinator.com Email: jack.jones@mailinator.com	ASSIGN TO	**NEW** Okta user Name: Jack Jones Username: jack.jones@mailinator.com Email: jack.jones@mailinator.com
NO Okta user matches found Name: Jane Lee Username: jane.lee@mailinator.com Email: jane.lee@mailinator.com	ASSIGN TO	**NEW** Okta user Name: Jane Lee Username: jane.lee@mailinator.com Email: jane.lee@mailinator.com
NO Okta user matches found Name: John Smith Username: john.smith@mailinator.com Email: john.smith@mailinator.com	ASSIGN TO	**NEW** Okta user Name: John Smith Username: john.smith@mailinator.com Email: john.smith@mailinator.com

Show 10 Showing 1 - 4 of 4

7. You can then import the users you want, and your on-prem provisioning configuration is considered complete.

OPTIONAL Syncing Custom Attributes from UD to MySQL

1. Open MySQL Workbench
2. In Schemas and under Stored Procedures, right click on “spSaveAccount”, select Alter Stored Procedure…



ocedure: spSaveAccount

Name: spSaveAccount

```
CREATE DEFINER=`admin`@`%` PROCEDURE `spSaveAccount`(
    IN json VARCHAR(500),
    OUT result INT
    )
BEGIN
    SET result = 0;
    SET @username = JSON_UNQUOTE(JSON_EXTRACT(json, '$.Resources[0].userName'));
    SET @firstname = JSON_UNQUOTE(JSON_EXTRACT(json, '$.Resources[0].name.givenName'));
    SET @lastname = JSON_UNQUOTE(JSON_EXTRACT(json, '$.Resources[0].name.familyName'));
    SET @email = JSON_UNQUOTE(JSON_EXTRACT(json, '$.Resources[0].emails[0].value'));
    SET @mobile = JSON_UNQUOTE(JSON_EXTRACT(json, '$.Resources[0].phoneNumbers[0].value'));
    SET @pwd = JSON_UNQUOTE(JSON_EXTRACT(json, '$.Resources[0].password'));

    SET @accountid = (SELECT account_id FROM dbo.ACCOUNTS WHERE username = @username);
    SELECT @username, @firstname, @lastname, @email, @mobile, @pwd, @accountid;

    IF @accountid > 0 THEN
        UPDATE dbo.ACCOUNTS
        SET username = @username,
            first_name = COALESCE(@firstname, first_name),
            last_name = COALESCE(@lastname, last_name),
            email = COALESCE(@email, email),
            mobile_phone = COALESCE(@mobile, mobile_phone),
            password = COALESCE(@pwd, password)
        WHERE account_id = @accountid;
        SET result = 1;
    ELSE
        INSERT INTO dbo.ACCOUNTS(account_type, first_name, last_name, mobile_phone, email, passw
        VALUES ('1', @firstname, @lastname, @mobile, @email, @pwd, @username);
        SET result = 1;
```

3. Modify the SQL inside to incorporate your custom attributes, then click Apply. The key here is the SET statements, those are the attributes extracted from the user JSON object coming from UD. Add the custom attributes with the SET statements, then add those attributes into both UPDATE and INSERT statements.

OPTIONAL Syncing Custom Attributes from MySQL to UD

1. Open MySQL Workbench
2. In Schemas and under Stored Procedures, right click on “spGetAccountJSON”, select Alter Stored Procedure…
3. Inside the SELECT statement, add in the table column desired after the “‘password’, password” line, this will ensure that the new attribute will be in the right hierarchy in the output JSON.

Reference:

1. User.json: This serves as a reference to identify the hierarchy of the user object in the OPP/SCIM implementation

```
{
    "schemas": [
    "urn:scim:schemas:core:1.0"
  ],
  "Resources": [
    {
      "schemas": [
        "urn:scim:schemas:core:1.0",
        "urn:scim:schemas:extension:enterprise:1.0",
        "urn:okta:onprem_app:1.0:user:custom"
      ],
      "id": "102",
      "userName": "admin",
      "password": "god",
      "active": false,
      "name": {
        "formatted": "Barbara Jensen",
        "givenName": "Barbara",
        "familyName": "Jensen"
      },
      "emails": [
        {
          "value": "bjensen@example.com",
          "primary": true,
          "type": "work"
        }
      ],
      "groups": [
        {
          "value": "1002",
          "display": "secondGroup"
        }
      ],
      "urn:okta:onprem_app:1.0:user:custom": {
        "isAdmin": true,
        "isOkta": false,
        "departmentName": "Administration"
      }
    }
  ]
}
```