Lucidum AWS Community License Setup Guide

To setup Lucidum AWS community product for the first use, please follow the steps below. Basic AWS knowledge is required during the installation process, including AWS IAM roles and EC2 settings. You should be able to complete the installation and start to inject AWS data into the Lucidum platform in 30 minutes. Note that the Lucidum Community product does not support AWS MFA (*Multi-factor authentication*) currently, please turn the MFA off if this option is enabled in your AWS accounts.

1. **Prerequisite:** Get Lucidum License and Terraform Scripts for Community product installation (Estimated Time: 30 minutes): Please make sure you download and install the latest Terraform and AWS CLI versions
    a. Apply for a Community license on Lucidum's website: Valid license is required for using Lucidum Community product
        i. Go to Lucidum website: www.lucidum.io
        ii. Go to the community product page, fill in your contact information, and submit the license application
        iii. You will receive the license file in your email soon
    b. Download Terraform to your computer that has access to your AWS environment: https://www.terraform.io/downloads.html
    c. Install Terraform on your computer and verify the installation by running `terraform version` under the command line (you should see the Terraform version printed out): https://learn.hashicorp.com/tutorials/terraform/install-cli?in=terraform/aws-get-started

    d. Download and install AWS CLI version 2 on your computer that has access to your AWS environment: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
    e. Verify AWS CLI installation by running `aws --version` under the command line (you should see the AWS CLI version printed out)

    f. Configure AWS credentials on your computer (https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html): Run `aws configure` under the command line to quickly set and view your AWS named profile, credentials, region, and output format (if you have multiple AWS accounts, make sure to run `aws configure --profile` repeatedly to create named profile for **each additional account** Lucidum will connect to). Also, make sure your AWS profile has **full (read and write) access** to EC2 and IAM services:

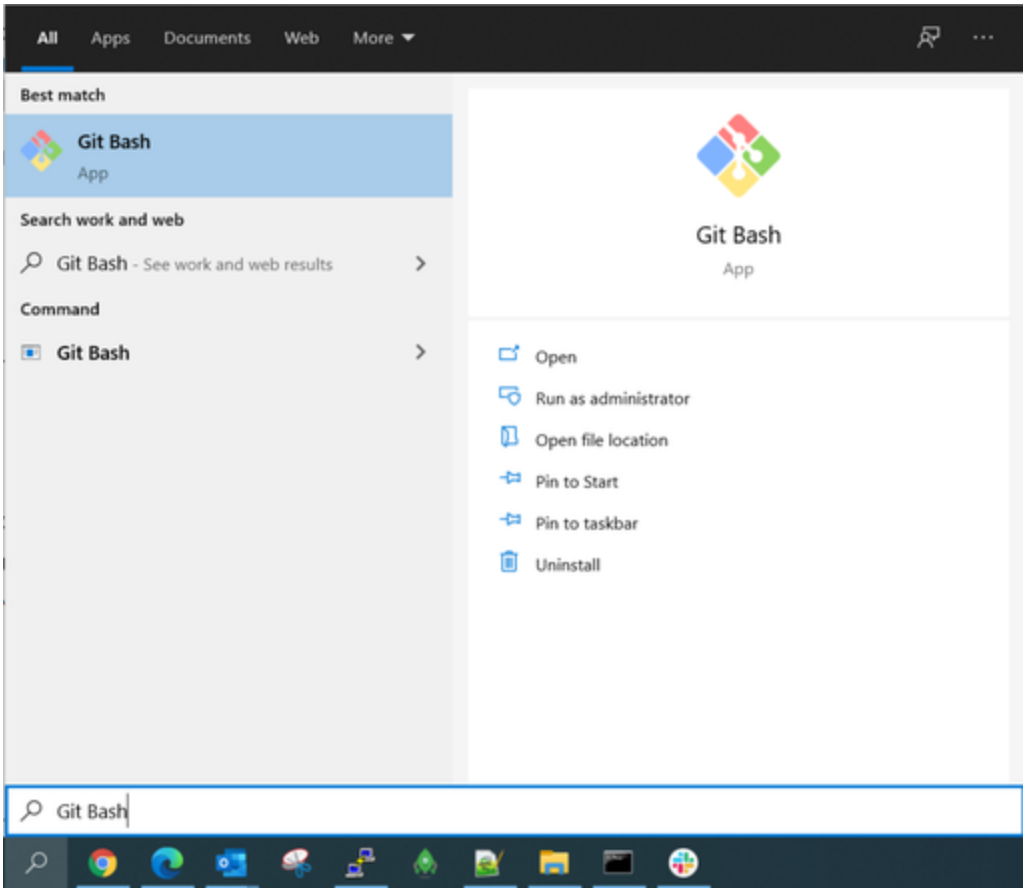
```
# Configure the profile for your MAIN AWS account
aws configure --profile profile_main_account
AWS Access Key ID [None]: (input your AWS access key here)
AWS Secret Access Key [None]: (input your AWS secret key here)
Default region name [None]: (input your AWS region here)
Default output format [None]: json

# Configure the profile for your ADDITIONAL AWS accounts
aws configure --profile profile_additional_account_1
AWS Access Key ID [None]: (input your AWS access key here)
AWS Secret Access Key [None]: (input your AWS secret key here)
Default region name [None]: (input your AWS region here)
Default output format [None]: json

......
```

    g. **Record all AWS profile names** created from the previous step, as the profile names will be used in Terraform scripts later to launch EC2 instance and create roles. The AWS profiles are also listed in the `{Your_Home_Folder}/.aws/config` file. For Windows OS, your home folder is generally under `C:\Users\{Your_Name}`, while for Linux OS, your home folder is under `/home/{Your_Name}`. Below is an example of the AWS config file with four profile names (i.e., default, profile_main_account, profile_additional_account_1, and profile_additional_account_2):

```
[default]
region = us-west-1

[profile profile_main_account]
region = us-west-1
output = json

[profile profile_additional_account_1]
region = us-west-1
output = json

[profile profile_additional_account_2]
region = us-west-1
output = json
```

h. By default, Lucidum Community product will collect data from the AWS services below (please check if these services are enabled in your AWS accounts to take full advantage of the community product)
   - *Asset information: EC2, ELB*
   - *DNS information: Route53*
   - *Micro-service information: EKS, ECS, Lambda*
   - *Metrics and Logs: CloudTrail, CloudWatch*
   - *Database information: DynamoDB*
   - *File information: S3*
   - *User information: IAM*
   - *Other contextual information: Pricing, Tags*

i. Download and install Git on your computer: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git (for Windows OS, make sure “Git Bash” component is installed by searching for the “Git Bash” application after installing Git as shown below)

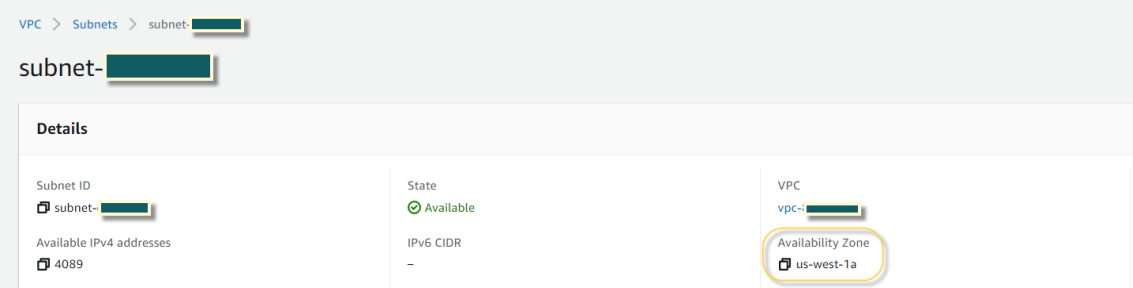


j. Create a new sub-folder, `lucidum-community`, under your home folder, e.g., "`C:\Users\{Your_Name}\lucidum-community`". This folder will be used to save and run Lucidum Terraform scripts

k. Under this `lucidum-community` folder, clone Lucidum Terraform setup scripts from https://github.com/LucidumInc/lucidum-deployment-seed by running Git Clone under command line (after running the Git Clone command, you should see a new sub-folder, `lucidum-deployment-seed`, created under the `lucidum-community` folder):

```
# Run this git clone command under the lucidum-community
folder, e.g.,
cd C:\Users\{Your_Name}\lucidum-community

git clone https://github.com/LucidumInc/lucidum-deployment-
seed.git
```

2. Launch Lucidum EC2 Instance with the Community product: We highly recommend using Lucidum Terraform scripts for easier and smoother installation (Estimated Time: 15 minutes)
   a. Go to the Lucidum Terraform script folder (e.g., `C:\Users\{Your_Name}\lucidum-community\lucidum-deployment-seed`) with your operating systems' file browser
   b. Under `lucidum-deployment-seed/amazon_aws/` folder, open `terraform.tfvars` file with a text editor and modify the Terraform variables in `terraform.tfvars` as needed. The variables are listed below with the important ones highlighted in red:
      i. `environment`: Your AWS environment
      ii. `availability_zone`: **Your AWS availability zone (required, you can get your AWS subnet ID and its availability zone from the VPC console as shown below)**



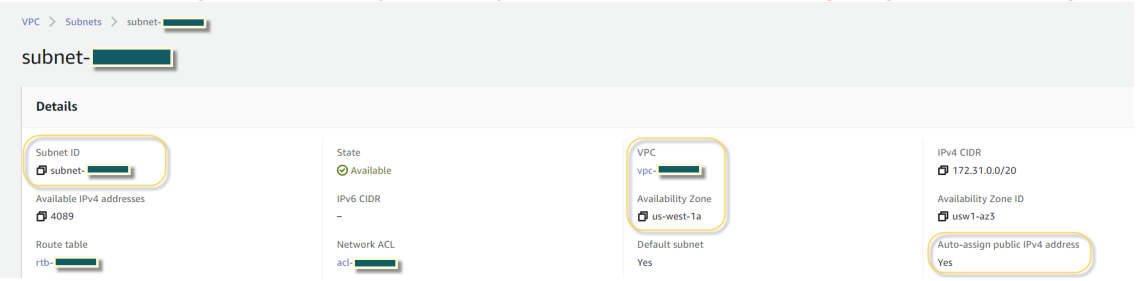
iii. `source_ami_account_number`: Lucidum account hosting the community AMI (don't change the value)

iv. `playbook_edition/playbook_version`: Lucidum AWS community version (don't change the value)

v. `instance_size`: **Your EC2 instance type (required, t3.2xlarge is recommended)**

vi. `associate_public_ip_address`: Whether to associate a public IP address with your EC2 instance. If this setting is true and the subnet ID allows public IP addresses, then a public IP address will be associated with the EC2 instance (note that this public IP address may change after the EC2 instance is restarted unless an elastic IP address is assigned to the EC2 instance, hence you may need to visit Lucidum web UI at a different URL in case the public IP address is changed)

vii. `subnet_id`: **Your AWS subnet ID for the EC2 instance (required, you can get your AWS subnet ID from the VPC console as shown below, make sure the subnet ID you select is within your AWS availability zone defined above by checking your AWS VPC Console; Also, if `associate_public_ip_address` is set as true above, this subnet needs to be public, otherwise you can only access the EC2 instance through its private IP address)**



viii. `vpc_id`: **Your AWS VPC ID for the EC2 instance (required, you can get your AWS VPC/subnet ID from the VPC console)**

ix. `key_name`: Your private key name to be associated with the EC2 instance

x. `trusted_cidrs`: **Your trusted CIDR IP address ranges for accessing the EC2 instance (required, make sure the CIDR range includes the computer's IP address from which you will initiate the connection to the Lucidum web UI. For example, if your computer's public IP address is 73.63.10.1, then this IP address needs to be added to the trusted_cidrs list; Or you can simply set the trusted_cidrs as 0.0.0.0/0 to allow all inbound accesses to the EC2 instance, but this may have some potential security issues)**

xi. `aws_region`: **Your AWS region for the EC2 instance (required)**

xii. `aws_profile`: **Your main AWS account's named profile from the AWS configuration (required, the EC2 instance will be created under this profile)**

xiii. Leave the other optional settings as default, here is an example of the `terraform.tfvars` file

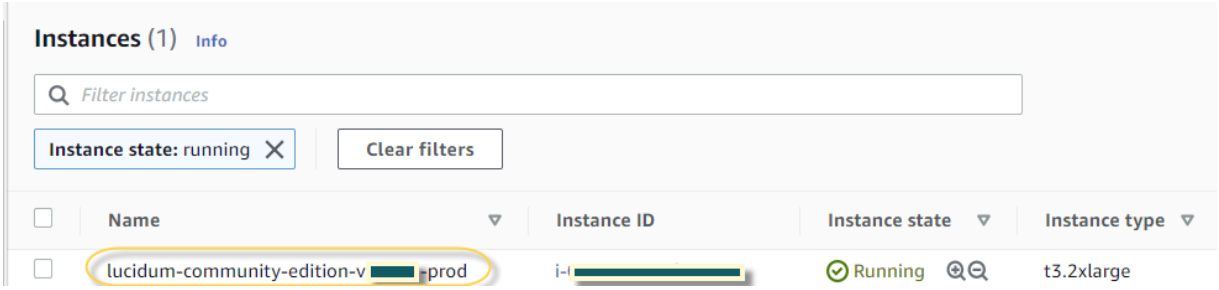
```
# Example terraform.tfvars file
environment = "prod"
availability_zone = "us-west-1a"            # Fill in
your availability zone
source_ami_account_number = "308025194586"  # Lucidum
account
playbook_version = "v0.1.15"
playbook_edition = "community"
instance_size = "t3.2xlarge"
associate_public_ip_address = true
subnet_id = "subnet-******"                 # Fill in
your subnet ID
vpc_id = "vpc-******"                       # Fill in
your VPC ID
trusted_cidrs = [ "10.0.0.0/8", "192.168.0.0/16",
"172.16.0.0/12", "73.63.10.1" ]
aws_region = "us-west-1"                    # Fill in
your region name
aws_profile = "profile_main_account"        # Fill in
your main account's profile name
key_name = ""
```

c. Save the `terraform.tfvars` file after the modifications

d. Under `lucidum-deployment-seed/amazon_aws/` folder, run `terraform init` under the command line to initiate the Terraform setup process

e. Under `lucidum-deployment-seed/amazon_aws/` folder, run `terraform plan` under the command line to preview and validate the actions Terraform is going to take

f. Under `lucidum-deployment-seed/amazon_aws/` folder, run `terraform apply` under the command line to launch the Lucidum EC2 instance in your AWS account (when prompted, type "yes" to let Terraform perform the actions). After running `terraform apply`, you should see a new EC2 instance is being created in your AWS EC2 console with the name `lucidum-community-edition-v******-prod`; Please wait until the instance state becomes "Running":



g. If the EC2 instance is no longer needed and you want to relinquish the resource: run `terraform destroy` to terminate the Lucidum EC2 instance in your AWS account (note that `terraform apply/destroy` only applies to the resources created by this Terraform script, and will not influence the resources created/modified in other ways such as through AWS console manually)

3. Run Lucidum Terraform scripts to setup cross-account role assuming if you have additional AWS accounts to access (**optional**): To make this role-assuming step easier, Lucidum provides a shell script to run over additional AWS accounts as illustrated below (Estimated Time: 15 minutes).

a. Go to the downloaded Lucidum Terraform script folder (i.e., `lucidum-deployment-seed`)

b. Edit `terraform.tfvars` under `lucidum-deployment-seed/x_account_assume_role/` sub-folder to set `trust_account` (your AWS main account ID where the Lucidum EC2 instance is running). The role-assuming process is to let the additional accounts allow this main account to access their resources, so the Lucidum EC2 instance under the main account will have permission to collect the data from additional AWS accounts. Here is an example of the `terraform.tfvars` file

```
# Example terraform.tfvars file under x_account_assume_role
trust_account = "*******"  # your main account ID where EC2
instance is running
```

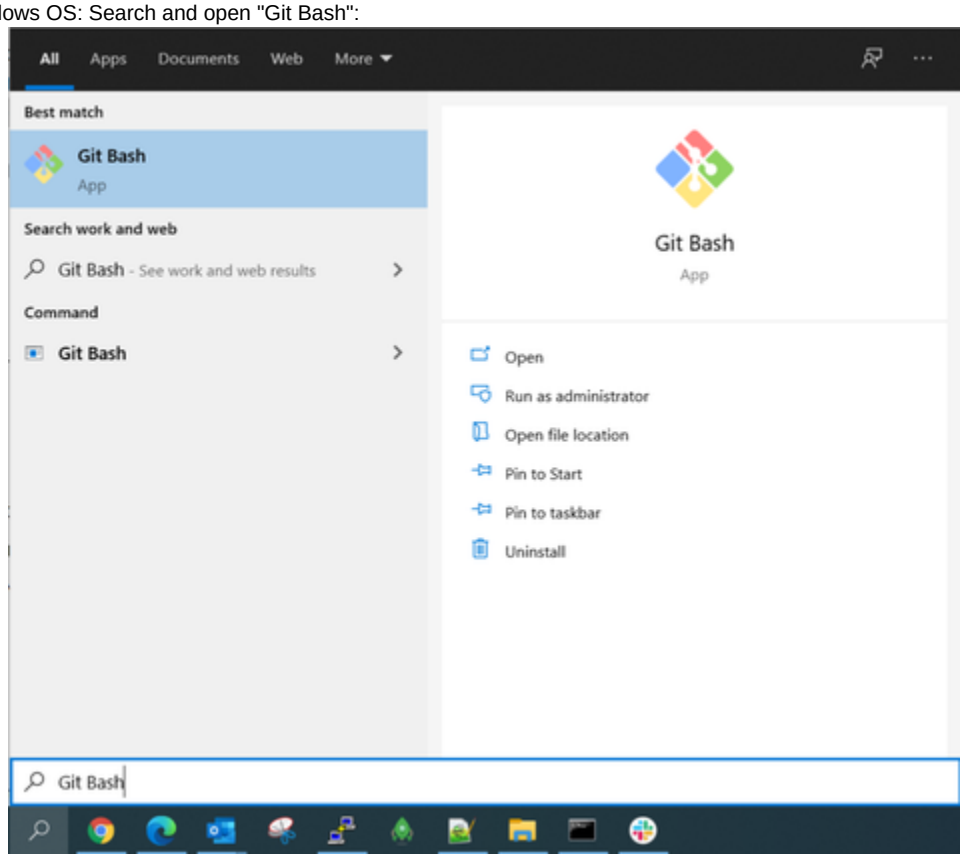
c. Edit `x_account_assume_role.sh` with a text editor under the `lucidum-deployment-seed` folder

d. Modify the first line, "`AWS_PROFILES=`", in `x_account_assume_role.sh` to add the named profiles for ALL additional AWS accounts, one on each line. For example,

```
#!/bin/bash -e

# Set aws profiles to loop thru for creating assume role in
additional accounts
AWS_PROFILES="
  profile_additional_account_1
  profile_additional_account_2
"
...
```

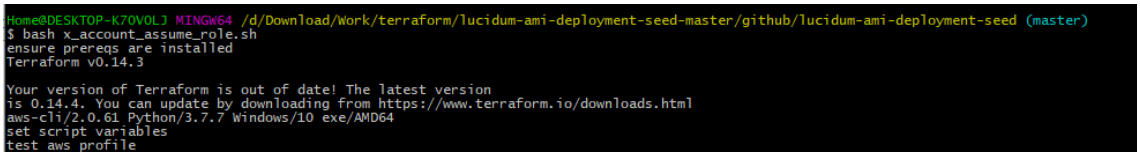
e. Save `x_account_assume_role.sh` file and run it

- For Mac/Linux OS: Run `bash x_account_assume_role.sh` under the command line directly to start the role-assuming process (when prompted, type “yes” to let Terraform perform the actions)
- For Windows OS: Search and open "Git Bash":



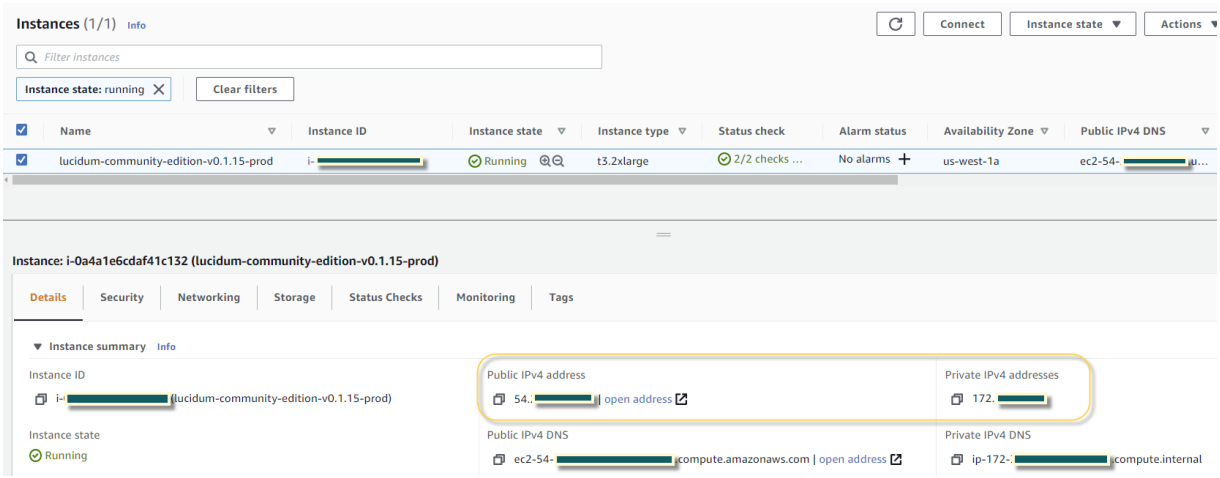
Then under `Git Bash` command line, run:

```
# Go to the Lucidum Terraform script folder, e.g.,
cd C:/Users/{Your_Name}/lucidum-community/lucidum-deployment-seed/

# Run shell scriptto start the role-assuming process
# when prompted, type “yes” to let Terraform perform the actions
bash x_account_assume_role.sh
```

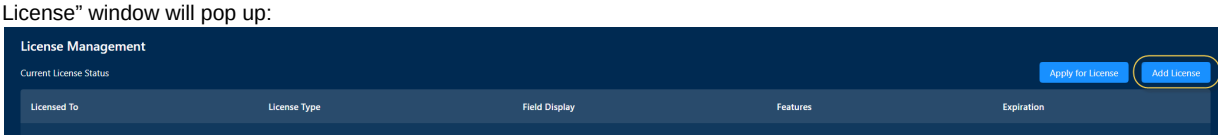


f. This shell script will create a role in each additional AWS account. The default role name will be `lucidum_assume_role` and the default role external ID will be `lucidum-access`. **All Role ARNs from different AWS accounts will be saved in** `x_account_assume_role_arns.txt` **file under the same** `lucidum-deployment-seed` **folder**. The role ARNs in this text file will be pasted into Lucidum web UI later

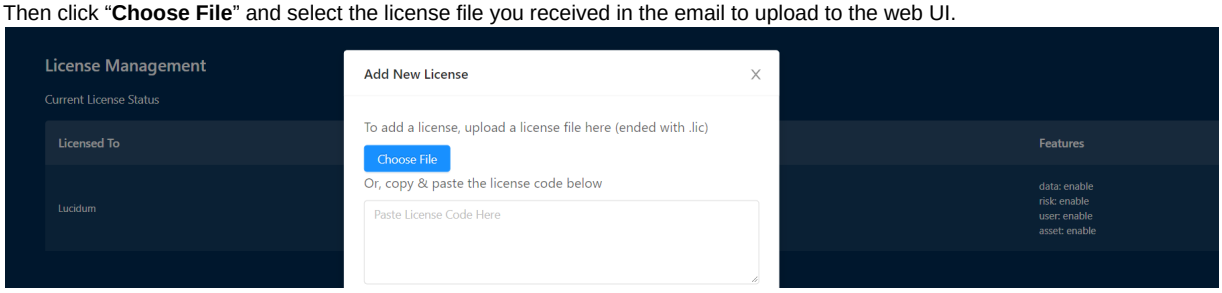
4. Open Lucidum web UI and input the license information (Estimated Time: 5 minutes):
   a. Open Lucidum web UI in an internet browser (Chrome, Firefox, or Edge) by visiting: `HTTPS://{Lucidum-EC2-IP-Address}/CMDB`. Your EC2 instance’s IP address can be found in the AWS EC2 console



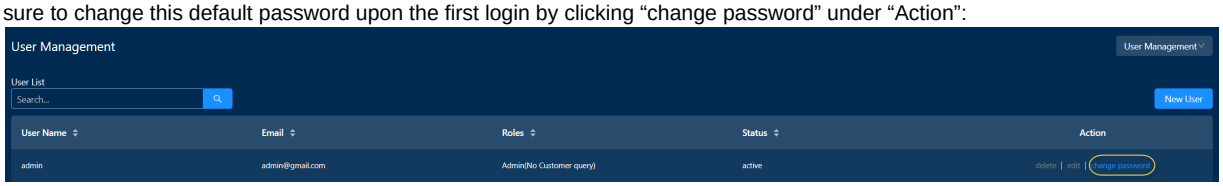
   b. Ignore the browser’s HTTPS security certificate warning as Lucidum web UI uses a self-signed HTTPS certificate and click “Continue” to go to the Lucidum UI login page
   c. Log in to the web UI with the default **username:** `admin` **and password:** `12345678`
   d. Upon the first visit, the UI will redirect you to the license management page: Click the “**Add License**” button, and a “Add New License” window will pop up:

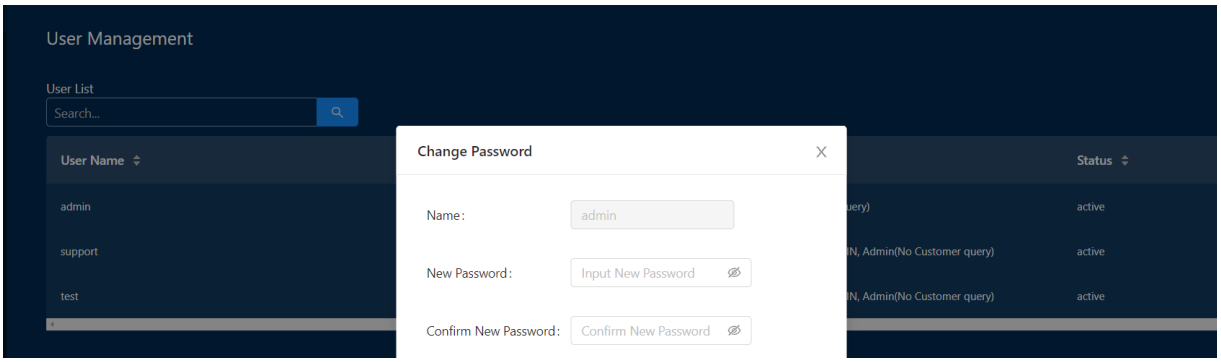


Then click “**Choose File**” and select the license file you received in the email to upload to the web UI.

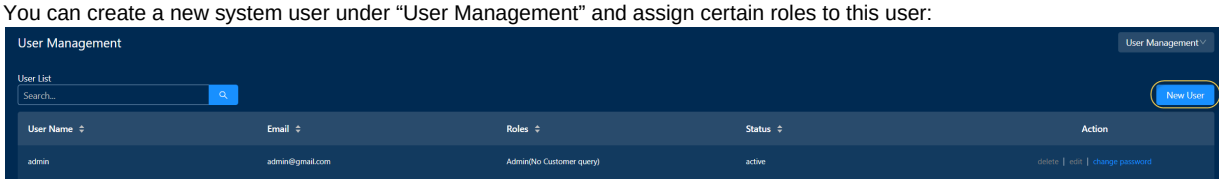


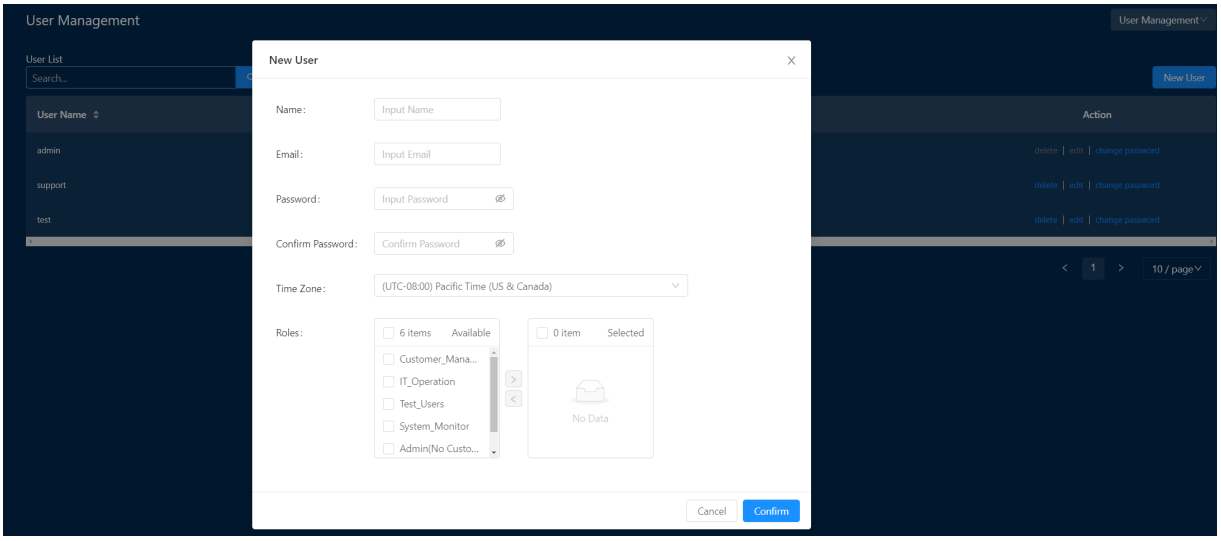
5. Setup Lucidum system users and roles if needed (**optional**): Lucidum web UI has the default “admin” user enabled after installation, you can create more system users as needed (Estimated Time: 5 minutes)
    a. Go to **Setting User Management** under Lucidum web UI: The default password for system “admin” user is 12345678, make sure to change this default password upon the first login by clicking “change password” under “Action”:



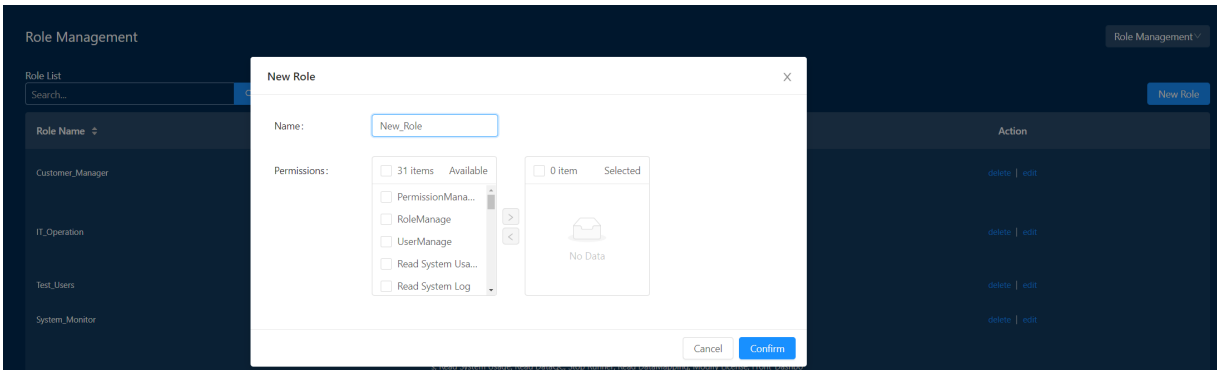


    b. You can create a new system user under “User Management” and assign certain roles to this user:

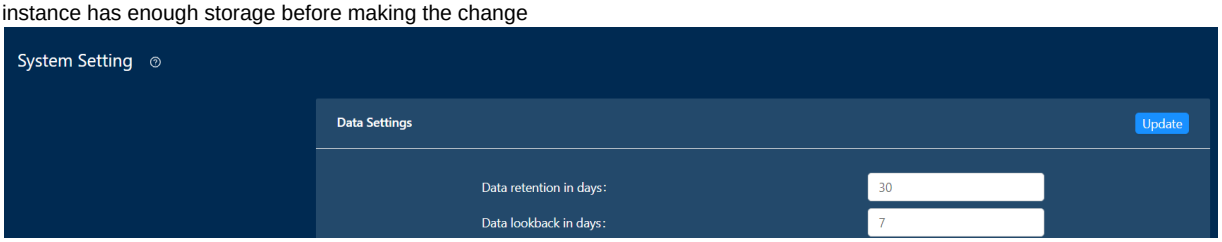




    c. You can also create a new role under “Role Management” and assign certain permissions to this role: Please refer to Lucidum’s user manual for more details on different permissions

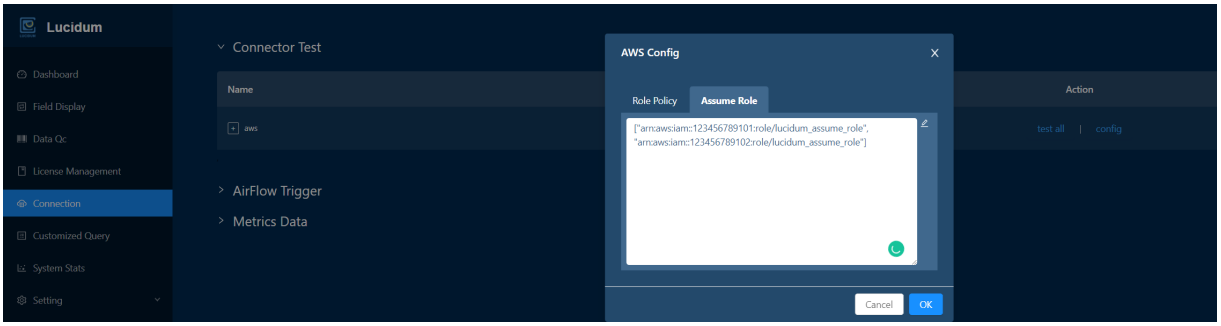


6. Adjust Lucidum system settings if needed (**optional**): Please refer to Lucidum's user manual for more details on different system settings. **Caution:** Inappropriate system settings could have negative impacts on Lucidum product (Estimated Time: 5 minutes)
    a. Go to **Setting System Setting** under Lucidum web UI
    b. You can adjust the "Data Settings", including the number of lookback days for data injection and the number of retention days for data storage. **Caution:** Longer data lookback/retention days will consume more storage space, please make sure your EC2 instance has enough storage before making the change



    c. You can provide sender email settings for query-based email alerting: Please refer to Lucidum's user manual for more details on the query-based alerting feature

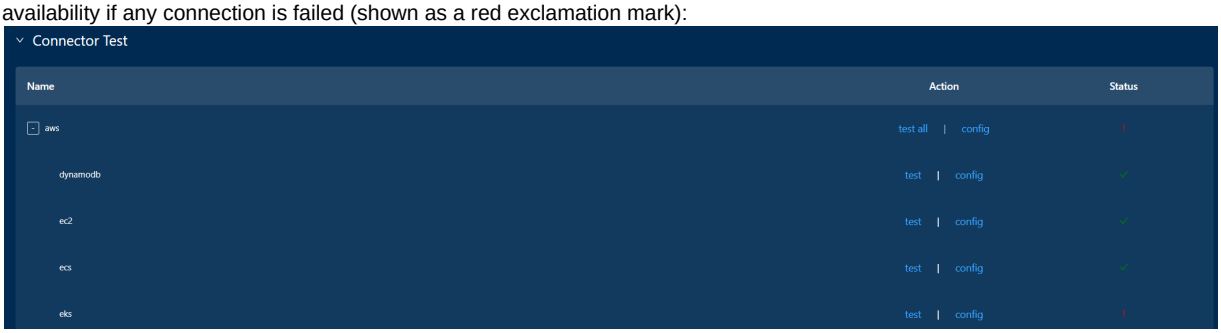
7. Fill in cross-account role ARNs under connector configuration if needed (**optional**): This will need the `x_account_assume_role_arns.txt` file from Step 3-f (Estimated Time: 5 minutes)
    a. Go to **Connection Connector Test** under Lucidum web UI
    b. Click **"config"** under **"Action",** fill in the role ARNs with *double quotes* as a *comma-separated* list from all additional AWS accounts (copied from the text file in Step 3-f) in the **"Assume Role"** box, and click "OK". Lucidum web UI will test if the role assuming is working for the additional AWS accounts. If you see any error message on the UI, please double-check the role assuming settings in your additional AWS accounts.



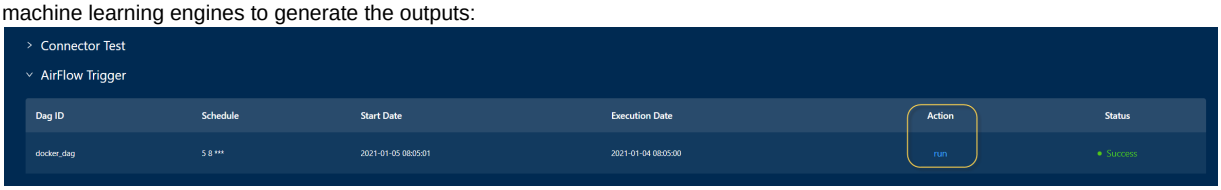
    c. At this stage, Lucidum is ready to connect to different AWS accounts with role assuming.

8. Test connections to different AWS services (Estimated Time: 5 minutes)
    a. Go to **Connection Connector Test** under Lucidum web UI
    b. Click **“test all”** under **“Action”** to run connection tests:

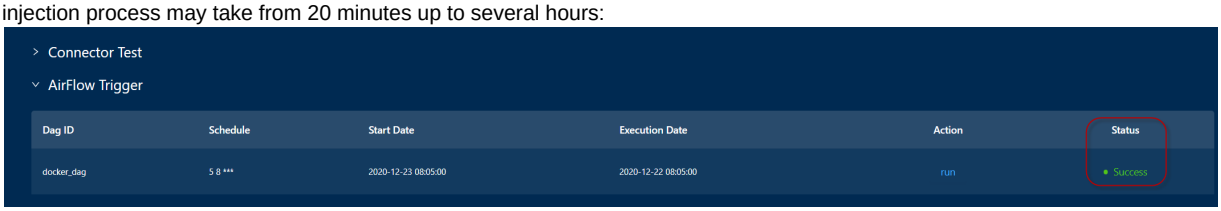
    c. Expand the “aws” connector and check the connection status. Double-check your AWS role permissions and/or resource availability if any connection is failed (shown as a red exclamation mark):



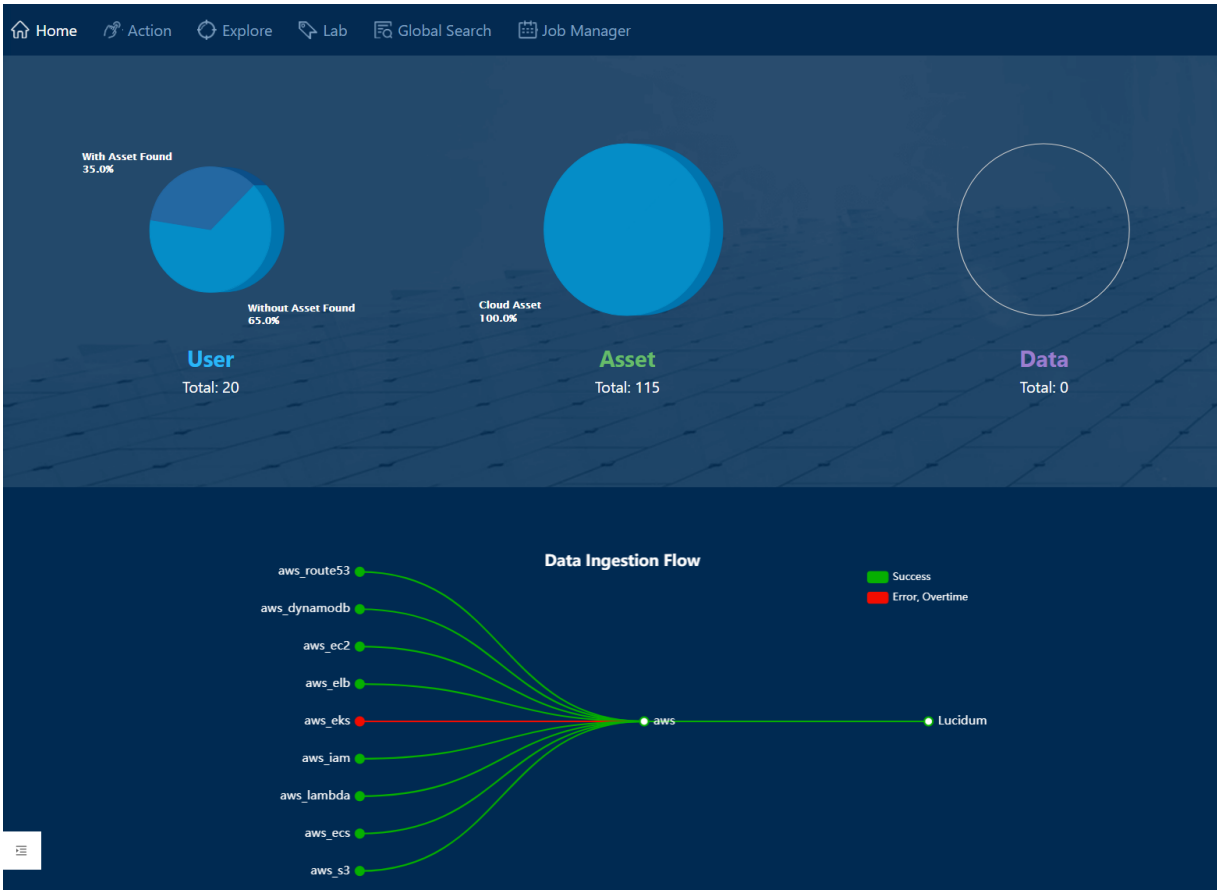
9. Start Lucidum Airflow scheduled job for AWS data injection (Estimated Time: 20 minutes or more)
    a. Go to **Connection Airflow Trigger** under Lucidum web UI
    b. Click **“run”** under **“Action”** to trigger the daily scheduled “docker_dag” job. This will run Lucidum AWS data injection and machine learning engines to generate the outputs:



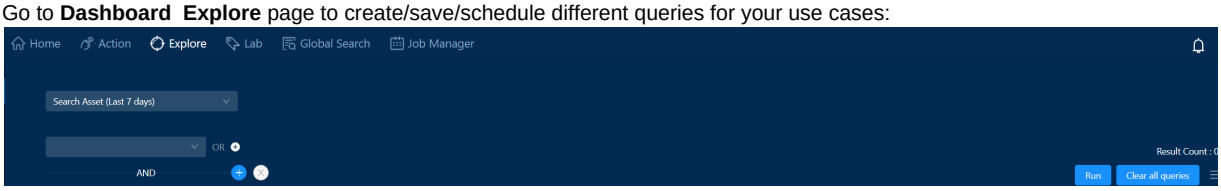
    c. Wait until the docker_dag’s status becomes **“Success”**. Depending on the data volume from your AWS resources, the data injection process may take from 20 minutes up to several hours:



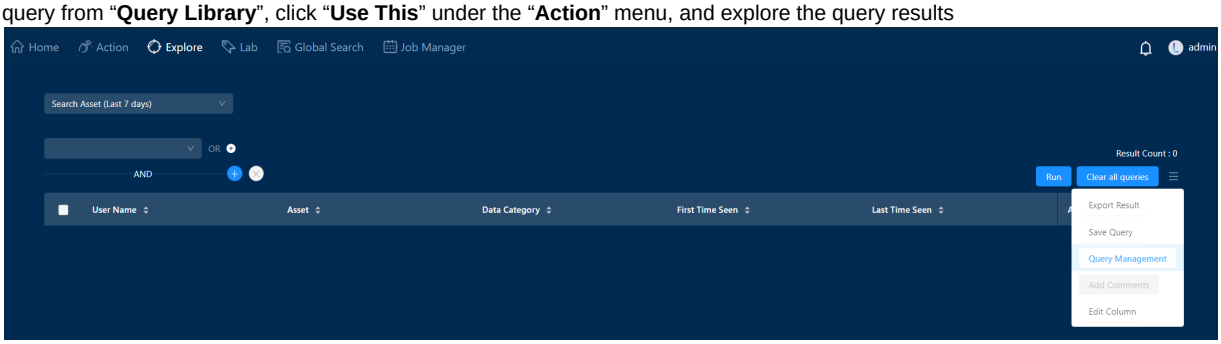
10. Explore the outputs in Lucidum web UI after the Airflow job is completed
    a. Go to **Dashboard** under Lucidum web UI to look at different dashboards and reports

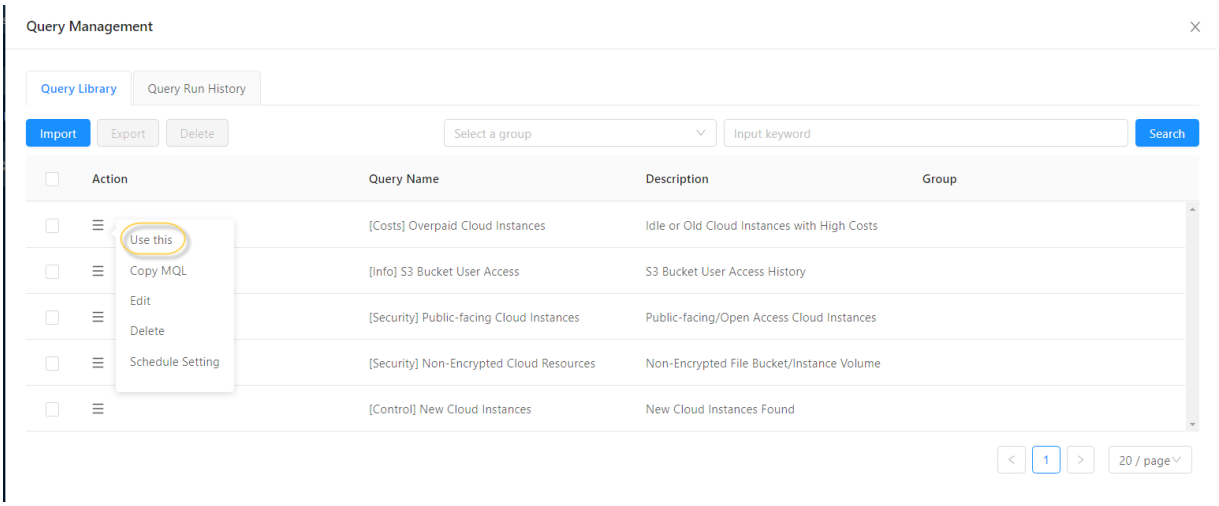


b. Go to **Dashboard Explore** page to create/save/schedule different queries for your use cases:



c. Lucidum provides some example queries under **Explore Query Management** as a quick start: You can select an example query from "**Query Library**", click "**Use This**" under the "**Action**" menu, and explore the query results





d. Please refer to Lucidum's product manual for more details on web UI usage and enjoy your journey in Lucidum!

Extra settings for AWS EKS Access (Optional)

When an Amazon EKS cluster is created, the IAM entity (user or role) that creates the cluster is added to the Kubernetes RBAC authorization table as the administrator. Initially, only that IAM user can make calls to the Kubernetes API server using kubectl. Thus, you need to add the `lucidum_assume_role` role created in the additional account to the kubectl configurations to get read-only permissions.

- First, you need to get the ARN of the role we just created from the above section
- You need to use **kubectl**, Kubernetes' admin command-line interface to map the role to it. **For each cluster** you want Lucidum to connect to, do the following steps as a **logged-in Kubernetes admin**:
  - Open the kubectl editor to view and edit the configMap configurations:

```
kubectl describe configmap -n kube-system aws-auth
kubectl edit -n kube-system configmap/aws-auth
```

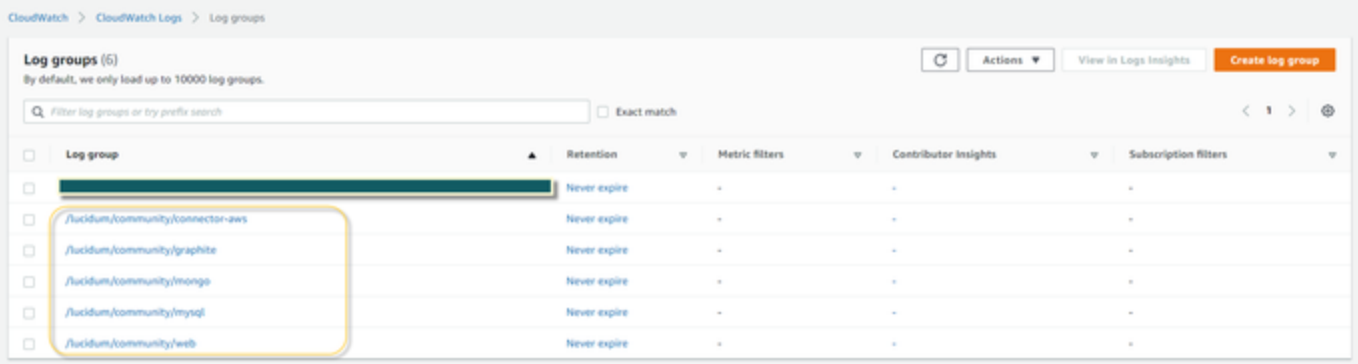
  - Append the new role mapping, while replacing the '*rolerarn*' field with the role ARN (make sure *rolearn*, *username,* and *groups* are at the SAME level). In this example, the role is assigned to *system:masters* group, but it can also be assigned to any other group with read permission to Kubernetes API:

```
        mapRoles: |
         - rolearn: arn:aws:iam::*****:role
/lucidum_assume_role
           username: lucidum_read
           groups:
           - system:masters
```

  - Save the file, and you should see a message indicating your edit was successful
  - The role can now authenticate against the Kubernetes cluster. If there are any errors, check if the configMap has the correct format and if the groups have the correct permissions.

Lucidum system logging

Lucidum system detailed logs will be saved into different CloudWatch Log groups under `us-west-1` region in your main AWS account (where the EC2 instance is running):



- `/lucidum/community/connector-aws`: Logs for AWS data injection
- `/lucidum/community/graphite`: Logs for Lucidum Graphite system monitoring service
- `/lucidum/community/mongo`: Logs for Lucidum Mongo Database
- `/lucidum/community/mysql`: Logs for Lucidum MySQL Database
- `/lucidum/community/web`: Logs for Lucidum Web UI application

For any application errors and malfunctions, please copy the related logs and submit a support request with the logs on Lucidum's user forum: https://lucidum.io/forums/. Lucidum technical support will get back to you as soon as possible.