Login to AWS SSO user through CLI

Contents

[Scope 2](#_Toc114751814)

[Problem Statement 2](#_Toc114751815)

[Introduction 2](#_Toc114751816)

[Install AWS CLI 2](#_Toc114751817)

[Enable AWS SSO 3](#_Toc114751818)

[AWS SSO. 3](#_Toc114751819)

[Create AWS organization 3](#_Toc114751820)

[Create an AWS SSO user 3](#_Toc114751821)

[Create a permission set and assign it to the AWS SSO user 4](#_Toc114751822)

[Set up AWS Profile for AWS CLI with AWS SSO configuration 4](#_Toc114751823)

[Login on AWS CLI via AWS SSO to run commands 5](#_Toc114751824)

[AWS SSO sign-in page 6](#_Toc114751825)

[List AWS SSO sessions 7](#_Toc114751826)

[Sign out of the AWS SSO session 7](#_Toc114751827)

Login to AWS SSO user through CLI

Document Name: Login to AWS SSO user through CLI

Owner: Bhagyashree Patle

# Scope

AWS SSO makes it easy to centrally manage SSO Access to multiple AWS accounts, moves the authentication to the IdP (Identity Provider) and removes the need for managing static, long-lived credentials.

# Problem Statement

Create a CLI setup to use SSO  seamlessly in your developer workflow so that sso will perform without going to the developer’s portal every time and paste short-lived credentials to the console.

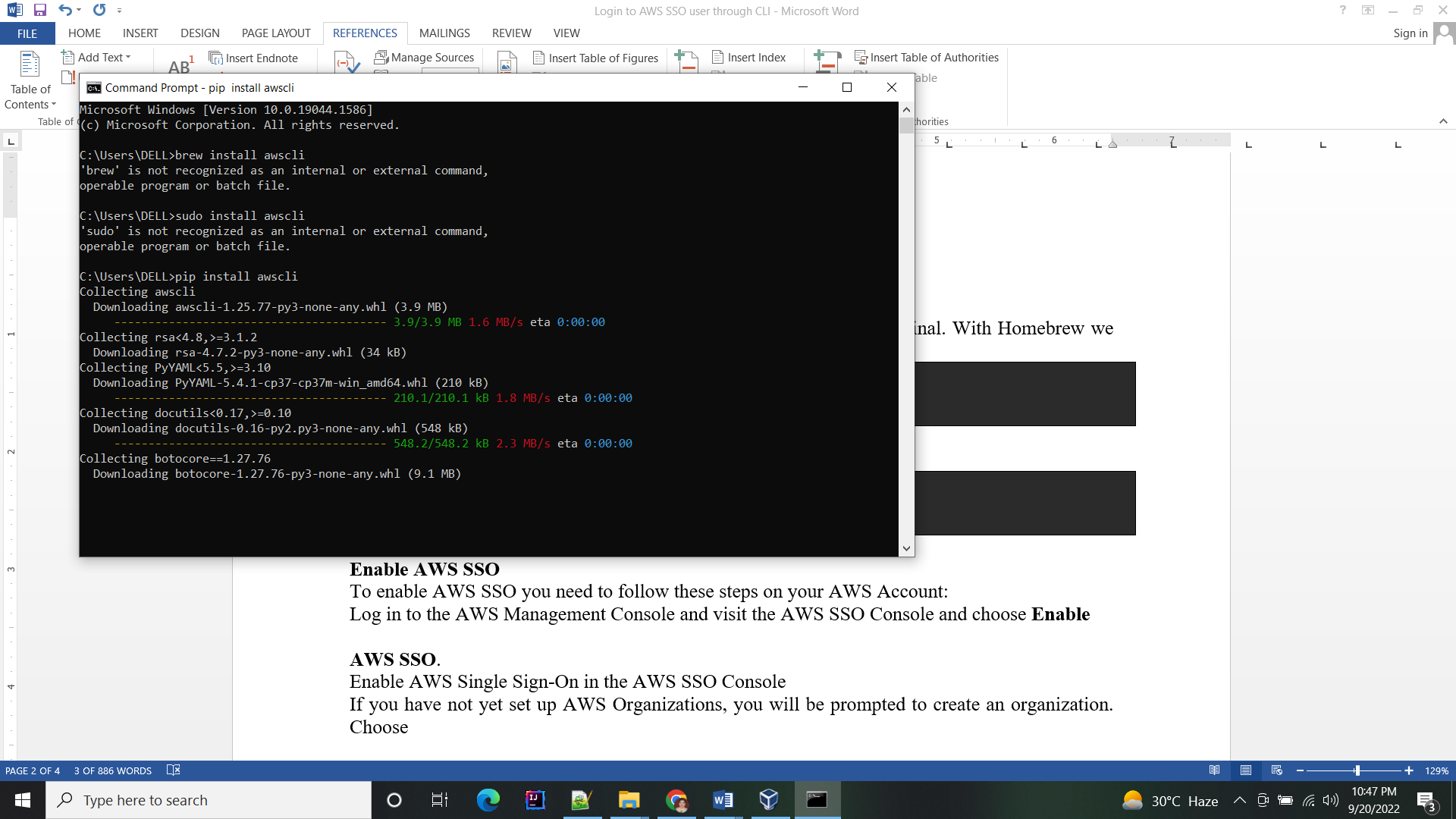
# Introduction

To get access to your AWS Account with the AWS CLI and AWS SSO, you need to install AWS CLI and enable AWS SSO in the AWS Console. After enabling AWS SSO, you create an SSO user with a permission set.

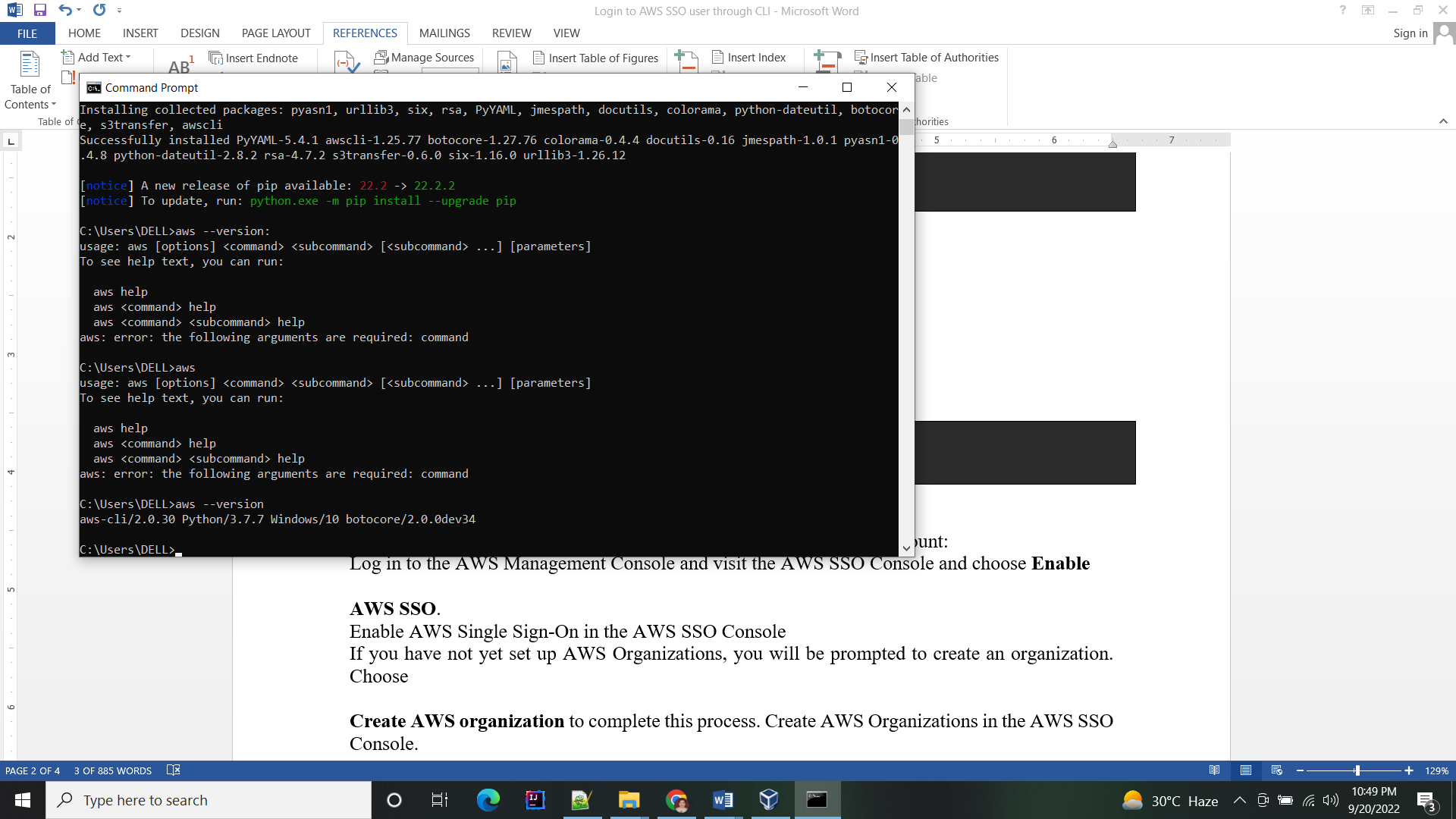
# Install AWS CLI

The AWS CLI allows you to interact with AWS services in your terminal. With pip we install AWS CLI with the following command:

pip install awscli



You can validate the version by running aws --version :



## Enable AWS SSO

To enable AWS SSO you need to follow these steps on your AWS Account:

To enable IAM Identity Center :

1. Sign in to the AWS Management Console with your AWS Organizations management account credentials.

2. Open the IAM Identity Center console.

3. Under Enable IAM Identity Center, choose Enable.

4. If you have not yet set up AWS Organizations, you will be prompted to create an organization.

## AWS SSO.

Enable AWS Single Sign-On in the AWS SSO Console

If you have [not yet set up AWS Organizations](https://towardsthecloud.com/aws-organizations), you will be prompted to create an organization. Choose

## Create AWS organization

To complete this process. Create AWS Organizations in the AWS SSO Console.

Once you’ve successfully enabled AWS SSO, you’ll see the user portal URL at the bottom of the page, copy yours and save it, because you’ll need it when you’re setting up the AWS profile in the next step. AWS SSO enabled page with user portal URL

## Create an AWS SSO user

Now you need to create an AWS SSO user, you’ll need that to authenticate against the AWS SSO user portal URL that you copied when you enabled AWS SSO in the previous step.

You create a new AWS SSO user by clicking the “new user” button on the AWS SSO user page in the AWS Console. Follow the steps in the wizard to complete the creation and you’ll then see the new user pop up (as shown below).AWS SSO Console with newly created user

## Create a permission set and assign it to the AWS SSO user

The newly created user needs access to your AWS account, therefore you need to assign a permission set to it.

For this example, I created a permission set based on an AWS-managed policy called “PowerUserAccess”. This policy has a very broad range of access, I would advise limiting the permission if you were to implement the SSO user on a production account.Give AWS SSO user access to your AWS account through permission sets

## Set up AWS Profile for AWS CLI with AWS SSO configuration

To Set up an AWS Profile for your AWS CLI we’re going to update the ~/.aws/config file (if it doesn’t exist yet, create the file) with the following configuration of our AWS SSO setup:

#######################

#### AWS SSO ####

#######################

[profile aws-sso-demo]

sso\_start\_url = https://d-936708b7d6.awsapps.com/start

sso\_region = eu-west-1

sso\_account\_id = 012345678910

sso\_role\_name = PowerUserAccess

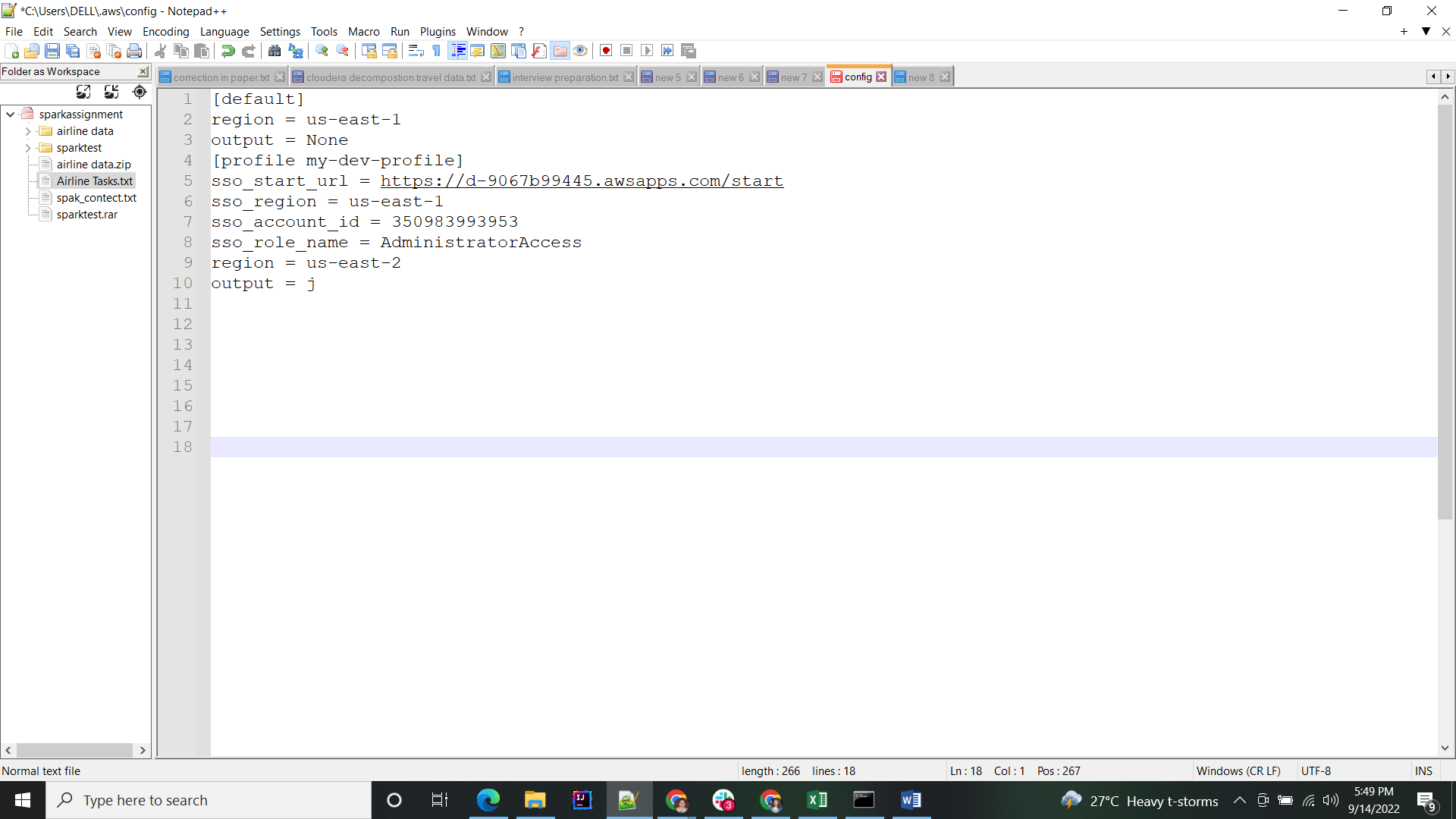
region = eu-west-1

output = json

[profile aws-sso-demo-cli]

region = eu-west-1

credential\_process = /usr/local/bin/aws-vault exec aws-sso-demo --json



Make sure to change the sso\_start\_url with the URL you obtained when you enable AWS SSO. Replace sso\_account\_id with your own AWS account id and update sso\_role\_name with the permission set that you created.

As you can see, [AWS Vault](https://github.com/99designs/aws-vault) is used to handle the credential process. Currently, a lot of SDKs and tools don’t support AWS SSO by default in AWS CLI v2. Therefore we use AWS Vault to bridge the gap and retrieve temporary credentials that are supplied by AWS SSO. To install AWS Vault on your, run the following command:

brew install --cask aws-vault

## Login on AWS CLI via AWS SSO to run commands

Now that everything is configured, we can actually login to the AWS CLI via AWS SSO with AWS Vault.

First export the SSO profile in your terminal that we created in the ~/.aws/config file:

export AWS\_PROFILE=aws-sso-demo-cli

When we run AWS commands in the terminal it will automatically use the profile that we exported in the shell.

To verify we can successfully run AWS commands, type the following in your terminal:

aws sts get-caller-identity

If you’re running it the first time after exporting the profile it will automatically open the browser and will ask you to sign in with the SSO user that you created:

## AWS SSO sign-in page

and after that, it validates your sign in and you can return to your terminal. You should see that AWS returned the identity of your AWS SSO user.

~ on aws-sso-demo-cli (eu-west-1)

➜ aws sts get-caller-identity

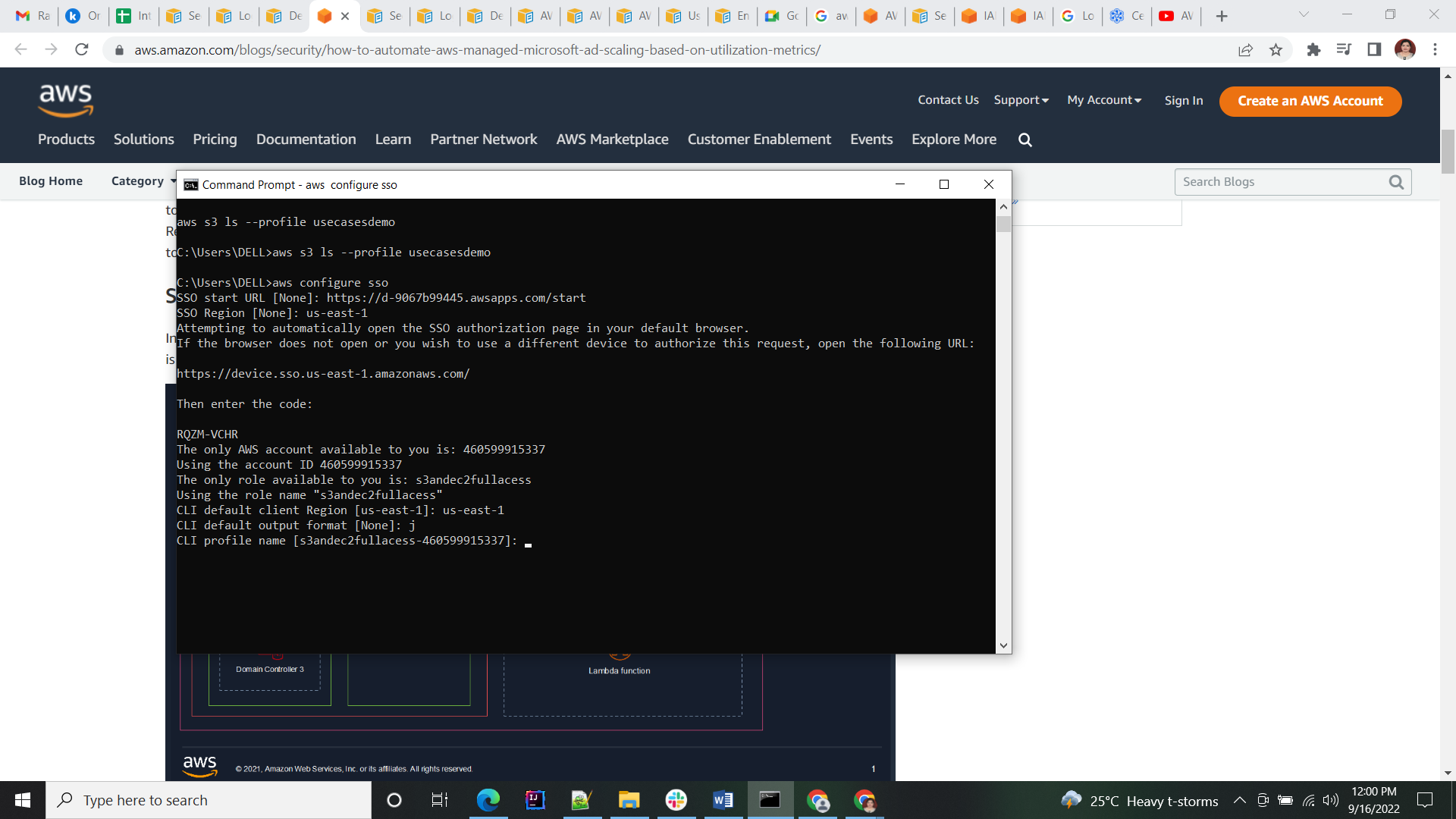
{

"UserId": "AROAWNK2CITGHESKW3YIE:danny",

"Account": "012345678910",

"Arn": "arn:aws:sts::012345678910:assumed-role/AWSReservedSSO\_PowerUserAccess\_8c1e59bfe711192a/danny"

}



## List AWS SSO sessions

If you’re using multiple AWS SSO sessions in your CLI, you can list which sessions are active with the following command:

➜ aws-vault list

Profile Credentials Sessions

======= =========== ========

aws-sso-demo - -

aws-sso-demo-cli - sso.GetRoleCredentials:3h59m47s

As you can see, the session aws-sso-demo-cli expires in 3h59m47s. This is defined in the permission set that we created for the AWS SSO user. The default AWS SSO session duration is 1 hour and can be set up to 12 hours.

## Sign out of the AWS SSO session

When you are done using your AWS SSO profile, you can choose to do nothing and let the AWS temporary credentials and your AWS SSO credentials expire.

However, you can also choose to run the following command to immediately delete all cached credentials from the secure key store:

aws-vault clear