**😊 AWS Security Token Service (STS)**

AWS STS stands for AWS Security Token Service. It's a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users federated via an external identity provider. These temporary credentials have a defined lifespan and can be used to access AWS services programmatically or through the AWS Management Console. STS helps in scenarios where you want to delegate access to AWS resources securely without sharing long-term credentials.

**😄 Use cases of STS:**

AWS STS has several use cases, including:

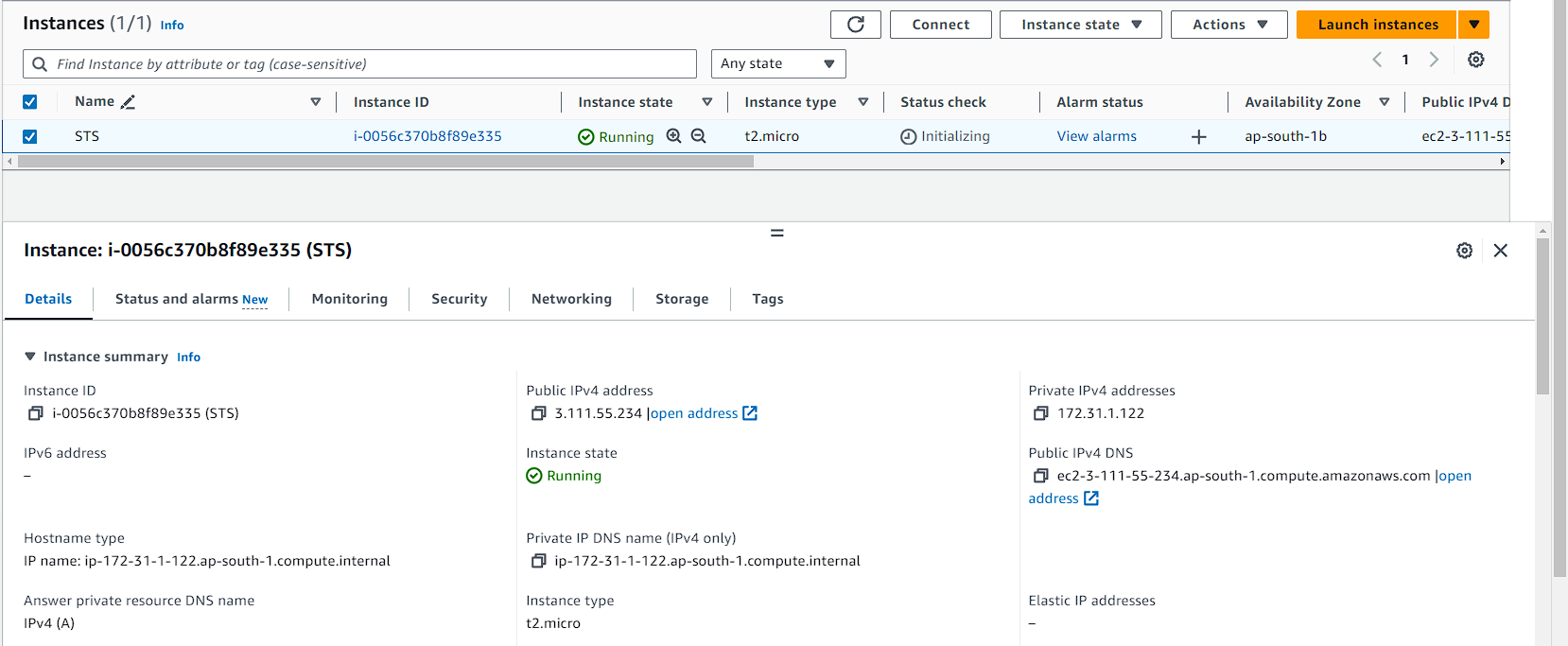
1. **Identity Federation:** You can use AWS STS to enable federated single sign-on (SSO) to AWS using identities from your corporate directory or an external identity provider such as Active Directory, LDAP, or social identity providers like Google or Facebook. This allows your users to access AWS resources without having to create AWS IAM users for each of them.
2. **Temporary Access to AWS Resources:** STS allows you to grant temporary access to AWS resources to users or applications without having to create permanent IAM credentials. This is particularly useful for scenarios such as cross-account access, where you want to grant access to resources in another AWS account temporarily.
3. **AWS CLI and SDK Operations:** You can use STS to generate temporary credentials that can be used with the AWS Command Line Interface (CLI) or AWS SDKs to make API calls to AWS services. This is especially useful in automated scripts or applications running on EC2 instances or other environments where securely storing long-term credentials is challenging.
4. **Assume Role for Delegated Access:** With STS, IAM users or applications can assume a role to obtain temporary security credentials with a specified set of permissions. This is useful when you want to delegate access to specific AWS resources or actions within your AWS account.
5. **Cross-Account Access:** STS allows IAM users in one AWS account to assume a role in another AWS account, enabling cross-account access. This is beneficial in scenarios where you need to access resources or perform actions in another AWS account while maintaining security and isolation between accounts.
6. **Session-based Access Control:** By using STS to generate short-lived session tokens, you can enforce fine-grained access control policies based on time constraints. This ensures that access to resources is only granted for the duration required, reducing the risk of unauthorized access.

Overall, AWS STS provides a flexible and secure way to manage temporary access to AWS resources, facilitate identity federation, and enforce access control policies in your AWS environment.

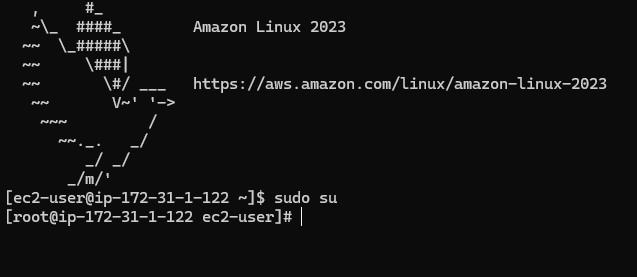
**In this guide, we're setting up and demonstrating the use of AWS Security Token Service (STS) to grant temporary access to AWS resources, particularly focusing on EC2 instances accessing S3 buckets. The end goal is to showcase how STS can provide temporary, limited-privilege credentials to EC2 instances, enabling secure access to AWS services without the need for long-term credentials. This setup enhances security by reducing the exposure of permanent credentials and allows for fine-grained access control and session-based access policies.**

**😄 To begin with the Lab:**

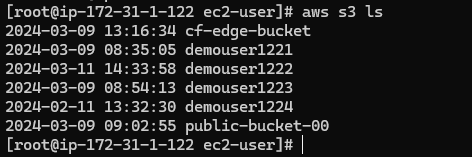
1. Log in to AWS console and navigate to IAM. Here you are going to create a role for EC2 which has S3 read-only access attached to it.
2. Once the role is created now you have to navigate to EC2 and create an instance then attach your role with the instance.



1. Now connect to your instance via SSH. Once you are logged in switch to root user.



1. Then if you do a listing for S3 buckets you will see that everything is working perfectly depending on the policies that have been attached.



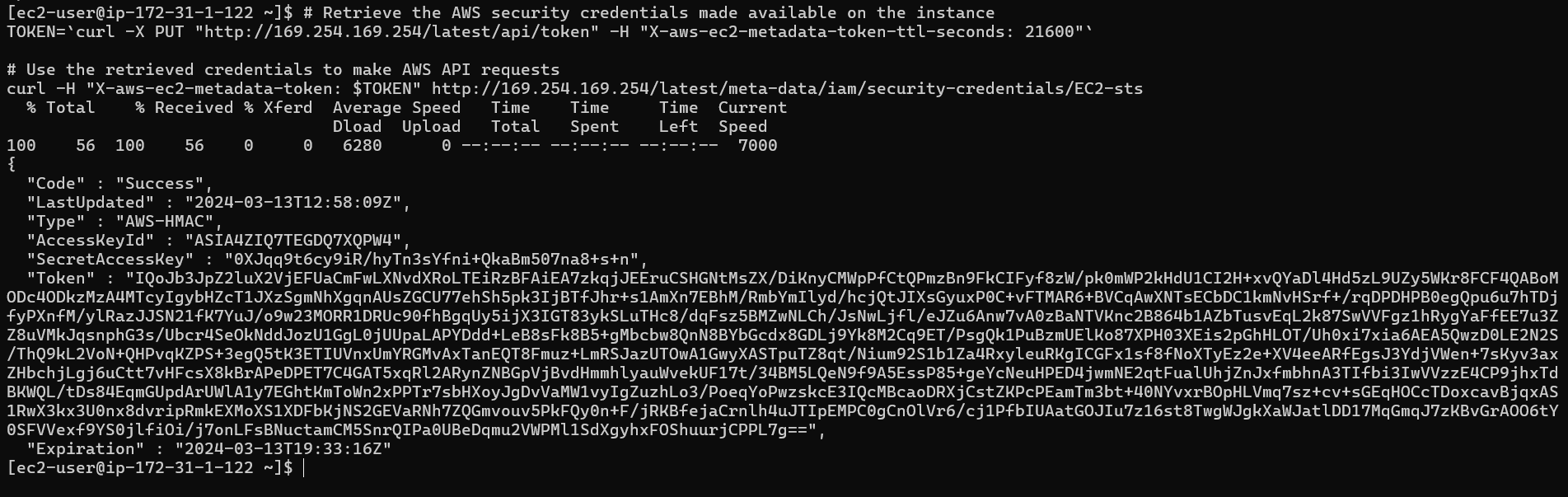
1. Now you might be thinking about how your EC2 instance is performing this type of activity because to do so you need to do the authentication for this operation to work.
2. So, as we know the IAM role helps here. However, in the backend IAM role will provide metadata that contains the access key secret key and the token.
3. Now if you will run the below command and in the last add the IAM role name that you have attached to your EC2.
4. Then you can see the access key, secret access key for this particular instance that STS (security token service) has generated. These keys are temporary in nature and they will expire as soon as your instance is terminated.
5. With that you can also see the expiration time which it has given to you from its side.

**# Retrieve the AWS security credentials made available on the instance**

**TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`**

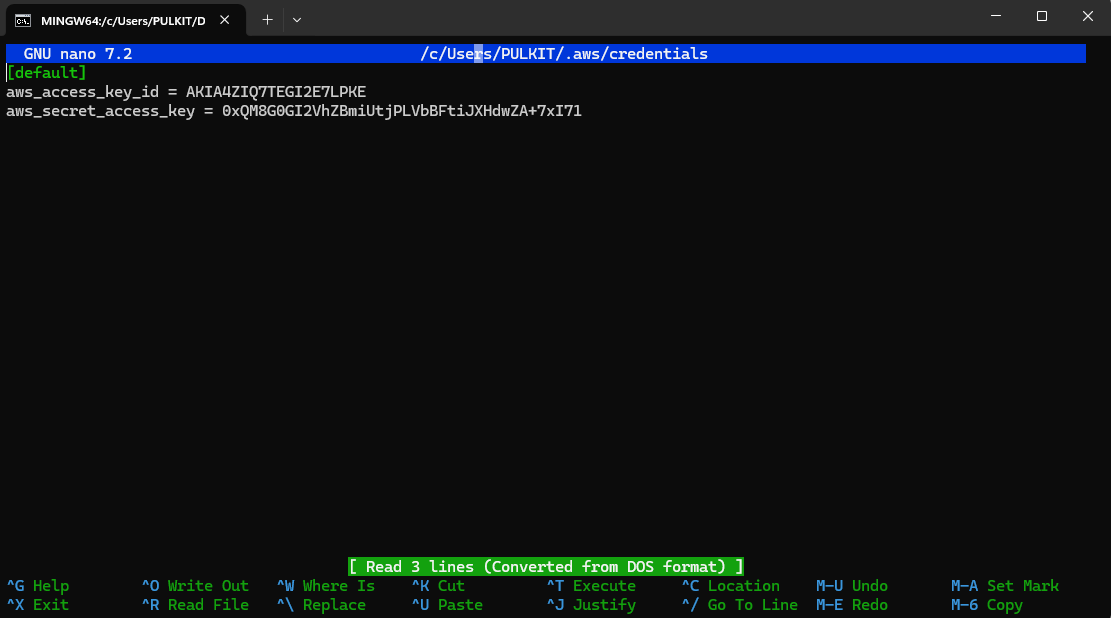
**# Use the retrieved credentials to make AWS API requests**

**curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/iam/security-credentials/role\_name**

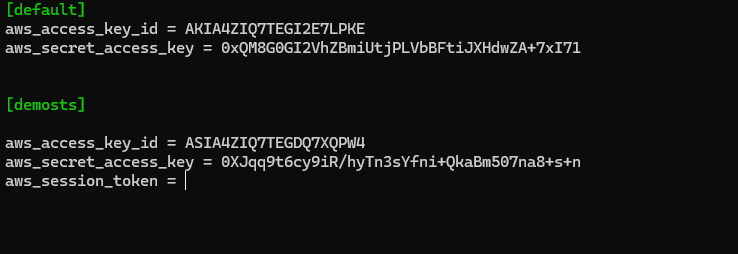


1. So even when we say that the IAM role is connected to the EC2 instance in the backend, you still have the access key secret, key, and the token. However, remember that these things are different from the long-term credentials that are typically generated for the IAM user.
2. The responsibility of generating those temporary credentials which were the access key secret, key, and token is the responsibility of the secure token service. So, the STS is responsible for generating those credentials.
3. Now if you want to use these STS-created access key and token you can also do that. First copy the access key, secret access key and token in your notepad.
4. Once you are done then you have to open GitBash and run this command. You can use this command only in gitbash. Once you have write this command and hit enter then you will be able to see your long term Access key and secret access key if you have configured them before.

**nano ~/.aws/credentials**



1. Now you are going to create a new profile here for your temporary credentials.
2. Here below your long-term credentials, you are going to create a new profile where you are going to define the access key, secret access key and session token. Then you are going to save it.



1. Once you have saved the temporary credentials now you will do a listing of the bucket from your local machine along with the profile of demosts or whatever name you have given.
2. You will be able to list your buckets.

