**POC : Automating SAS Token Generation and File Download from Azure Storage Using Jenkins Pipeline**

**Purpose of SAS Token**

A **Shared Access Signature (SAS) token** is a secure way to grant limited access to Azure Storage resources without exposing account keys. It allows users to:

* Define granular permissions (read, write, delete, list, etc.).
* Set expiration times for temporary access.
* Restrict access by IP address or protocol.
* Improve security by avoiding direct access to account keys.

**Best Practices for SAS Token Usage**

* **Minimize Token Lifespan:** Set the expiration date as short as possible to reduce exposure.
* **Use Least Privilege:** Grant only necessary permissions.
* **Restrict IP Access:** Limit usage to specific IP ranges if possible.
* **Use HTTPS:** Ensure tokens enforce HTTPS for secure access.
* **Rotate Regularly:** If used in long-term scenarios, implement rotation mechanisms.

**Jenkins Pipeline Script for SAS Token Generation and File Download**

This Jenkins pipeline automates the following tasks:

1. Authenticate using a **Service Principal Name (SPN)**.
2. Generate a **SAS token**.
3. Download files using **AzCopy**.

**Prerequisites**

* Jenkins installed with necessary plugins.
* Azure CLI installed on the Jenkins agent.
* Service Principal with appropriate role assignments.
* azcopy tool installed.

**Jenkins Declarative Pipeline**

pipeline {

agent any

environment {

AZURE\_SUBSCRIPTION\_ID = '<your-subscription-id>'

AZURE\_TENANT\_ID = '<your-tenant-id>'

AZURE\_CLIENT\_ID = '<your-client-id>'

AZURE\_CLIENT\_SECRET = '<your-client-secret>'

STORAGE\_ACCOUNT\_NAME = '<your-storage-account>'

CONTAINER\_NAME = '<your-container>'

BLOB\_NAME = '<your-blob-name>'

LOCAL\_DOWNLOAD\_PATH = '/tmp/downloaded-file'

}

stages {

stage('Authenticate to Azure') {

steps {

script {

sh '''

az login --service-principal \

-u $AZURE\_CLIENT\_ID \

-p $AZURE\_CLIENT\_SECRET \

--tenant $AZURE\_TENANT\_ID

'''

}

}

}

stage('Generate SAS Token') {

steps {

script {

def sasToken = sh(script: '''

az storage blob generate-sas \

--account-name $STORAGE\_ACCOUNT\_NAME \

--container-name $CONTAINER\_NAME \

--name $BLOB\_NAME \

--permissions r \

--expiry $(date -u -d "+1 hour" +"%Y-%m-%dT%H:%MZ") \

--https-only \

--output tsv

''', returnStdout: true).trim()

env.SAS\_TOKEN = sasToken

}

}

}

stage('Download File Using AzCopy') {

steps {

script {

sh '''

azcopy copy \

"https://$STORAGE\_ACCOUNT\_NAME.blob.core.windows.net/$CONTAINER\_NAME/$BLOB\_NAME?$SAS\_TOKEN" \ **// Here SAS token attached to azcopy request**

"$LOCAL\_DOWNLOAD\_PATH"

'''

}

}

}

}

post {

always {

echo "Pipeline execution completed."

}

}

}

**Explanation of the Pipeline Stages**

1. **Authenticate to Azure:** Uses az login with SPN credentials to authenticate the pipeline to Azure.
2. **Generate SAS Token:** Uses az storage blob generate-sas to create a temporary SAS token valid for 1 hour with read-only access.
3. **Download File Using AzCopy:** Uses azcopy copy to download the file using the SAS token.
4. **Post Actions:** Ensures a message is logged when the pipeline completes.

**Expected real time problems:**

Sometimes If the SAS token expiration date is small, download files might take longer 1 hr. that time Jenkins job is failed with Authentication Error. So be carefull when choosing the expiration date.

**Conclusion**

This Jenkins pipeline efficiently automates secure file downloads from Azure Storage by leveraging **Service Principal Authentication, SAS token generation, and AzCopy**. By following best practices, this setup enhances security and ensures efficient resource management in Azure environments.