Okta AWS SAML Authentication

Technical Specification

Abstract

Low Level Technical Solution & design details enlightening on the application workflow, framework components, technologies involved, application infrastructure and other related aspects

Author

Rishi Raj Bansal

Document Control

Version History

Date	Version	Amendments	Updated By
15 February 2020	1.0	Initial Draft	Rishi Raj Bansal

Table of Contents

Introduction **3**
Overview 3
Initial Project Setup/Installation **3**
Pulumi Setup 3
Pulumi Project Setup 4
Deployment Workflow **5**
Pulumi - Infrastructure Setup **6**
Setup Instructions 6
Operations performed by Pulumi Stack 8
Application Configuration **8**
Pulumi Project Configuration 8
Environment Files 8
Appendix **9**
Technology Stack 9

Introduction

Overview

Enabling SAML based Authentication SSO system for AWS accounts using Okta as IdP.

The objective of the project is to allow users to securely login into the AWS account (Console) from one-click login without using a password. This will make easier for users to access the application faster and avoiding the hassle to providing the login credentials everytime.

Certain IAM Role can be associated with the account access with the required privileges. These privileges can be controlled from IAM Policies. Multiple IAM Policies can be associated with single IAM Role, hence providing an isolated layers of access which can be turned on/off. Each policy can be centralized around one AWS service or multiple.

Initial Project Setup/Installation

Pulumi Setup

Prerequisites: Pulumi v1.9.1 already installed.

To setup/configure Pulumi IaC template tool, follow these steps.

1. AWS IAM User Creation
2. Setup AWS profile
3. Account creation on Pulumi app backend
4. Create Pulumi Backend Access Token

AWS IAM User Creation

This step is required only, If IAM user is not already created.

Create IAM user in AWS and save its ***Access Key ID*** and ***Secret Access Key.***

Setup AWS profile

To successful authentication with AWS, Pulumi needs AWS credentials to connect with AWS.

To authenticate with AWS, ***~/.aws/credentials*** file based mechanism is used. Credentials stored in this file are used by Pulumi to connect AWS.

Will create separate profile for ‘nuage’ that will be used by Pulumi. However, this is optional and ‘default’ profile can also be used.

To add non-default profile in ***~/.aws/credentials*** file, add profile section as follows:

```
[default]
aws_access_key_id = AAAAZS4UNA26ZNMAAABB
aws_secret_access_key = AAAAds3S2wr1kItugty9RsikamRa7qS3TpNAAAAA

[nuage]
aws_access_key_id = AAAAAS4UNA26SYIAAAAA
aws_secret_access_key = AAAAAn0jnWHbYyGhAAAAA
```

This authentication is based on 'nuage' IAM user.

Account creation on Pulumi app backend

Pulumi requires a data store/backend to save the state and stack details of the infrastructure.

Pulumi provides a free service backend to store the state details on cloud service. It needs no additional configuration and just need to create and account and ready to use.

Create an account on Pulumi backend using following URL:
https://app.pulumi.com/

Create Pulumi Backend Access Token

The Pulumi Service backend login process involves using access tokens.

To create access token, login to https://app.pulumi.com/. Go to Setting -> Access Tokens page and create new token.

Save that token as it will need to login into Pulumi while creating/deploying stack.

Pulumi Project Setup

Prerequisites: Python **v 3.6.x** or later already installed.

Following steps are needed to install the Pulumi application successfully:

1. Create project directory where the project artifacts of 'Pulumi application' will be copied

2. Clone the code from GitHub repository to newly created project directory.

3. CD to application root folder:

```
$ cd awsAccountSAMLAuthentication\src
```

4. Install Dependencies :

```
$ pip install –r requirements.txt
```

5. Update environment specific properties in configuration files. If default settings looks fine, then they can be left intact.

Deployment Workflow

After setting up projects and setting up Pulumi as mentioned in the above section, this section is used to follow the steps to make application work in union with its all distinct components integrated and configured.

Following steps need to be executed to complete Deployment Workflow:

1. Create Infrastructure Stack by Pulumi
2. Rerun the Pulumi template to update Okta AWS App with ARNs

1. **Create Infrastructure Stack by Pulumi** ☜

   This will create Okta OIN AWS app with base properties and its other components. At this moment, Pulumi need Okta's app metadata details to associate them with IAM Identity Provider and Role creations.

   Copy down the ***AWS IAM SAML Provider ARN*** and ***AWS IAM Okta Role ARN*** generated as the output from the Pulumi template as this is required in next step.

2. **Rerun the Pulumi template**

   Pulumi template need to execute again to update the Okta AWS App with the ARNs of IAM Identity Provider and Role generated in above step.

   In the Pulumi project, open the environment configuration file and update property **AWS_IAM_SAML_PROVIDER_ARN** and **AWS_IAM_SAML_ROLE_OKTA_ARN** with the ARNs values copied in above step.

```
AWS_IAM_SAML_PROVIDER_ARN = arn:aws:iam::659051841213:saml-provider/NuageOkta001
AWS_IAM_SAML_ROLE_OKTA_ARN = arn:aws:iam::659051841213:role/OktaRoleForSAMLAccess
```

   Run the pulumi template from the same terminal used in Step 1:

```
$ pulumi up -y
```

   This will update the Okta AWS app's Sign On settings with AWS IAM associations.

This completes Application Deployment Workflow.

Pulumi - Infrastructure Setup

Setup Instructions

To create and deploy the infrastructure on AWS and Okta using Pulumi, follow these instructions:

1. Setup Environment variables
2. Login to Pulumi App Backend
3. Create Stack (One-time operation)
4. Configure Stack (One-time operation)
5. Deploy Stack

Setup Environment variables

Open the terminal window and execute following commands.

1. Set Pulumi path

For Ubuntu:
```
export PATH="$PATH":/usr/local/pulimi/bin
```

For Windows:
```
SET PATH=%PATH%;C:\Development\Pulumi\v1.9.1\bin\
```

To avoid setting it everytime, this path can be setup in user profile for Ubuntu and in System environment variables for Windows.

2. Set *PULUMI_ACCESS_TOKEN* environment variable. Use the token created in Pulumi app backend account.

For Ubuntu:
```
export PULUMI_ACCESS_TOKEN=pul-eec761854odcb3c2ba1a6cbba2ec3c7911adbf30
```

For Windows:
```
SET PULUMI_ACCESS_TOKEN=pul-eec761854odcb3c2ba1a6cbba2ec3c7911adbf30
```

3. Set project configuration environment file in path

For Ubuntu:
```
export ENV_FILE=.env.development
```

For Windows:
```
SET ENV_FILE=.env.development
```

Login to Pulumi App Backend

```
$ pulumi login
Logging in using access token from PULUMI_ACCESS_TOKEN
Logged into pulumi.com as rishirajbansal (https://app.pulumi.com/rishirajbansal)
```

Create Stack (One-time operation)

This command only needs to execute first time when needs to create stack. Once stack is created, this stack can be used for all further operations.

```
$ pulumi stack init dev
```

Configure Stack (One-time operation)

While creating stack, also need to provide: region and aws profile. If default aws profile is need to use, then replace the name with default in aws:profile. Along with these details, also need to provide Okta configuration details.

```
$ pulumi config set aws:region us-east-1
$ pulumi config set aws:profile nuage

$ pulumi config set okta:orgName dev-433222
$ pulumi config set okta:baseUrl okta.com
$ pulumi config set okta:apiToken 000zjuI4lLJAAAAAzgce8x3B-eVjfZLde_GI5sIj
```

OR

To avoid executing these commands manually, direct changes can be done in '***Pulumi.dev.yaml'*** file. This file contains all configuration details related to the Pulumi stack.

Deploy Stack

This will create and deploy the infrastructure on AWS and Okta based on the above configuration.

☞ *If this is the first time, Okta app is being created, then the values for ARNs in environment configuration file should be set to 'None'.*

```
AWS_IAM_SAML_PROVIDER_ARN = None
AWS_IAM_SAML_ROLE_OKTA_ARN = None
```

Change directory to the path where the '__main__.py' file exists. This is the main template file of Pulumi.

```
$ cd awsAccountSAMLAuthentication\src\iac
$ pulumi up

It will preview stack and show the details the of all operations that will be perform on AWS asking user consent y/n.
```

To avoid interruption and to automatically approve and perform the update, use following command instead:

```
$ pulumi up -y
```

Operations performed by Pulumi Stack

On deploying the stack configuration from Pulumi, it will generate following components on AWS Cognito:

- ✓ Okta OIN based Amazon web services creation
- ✓ IAM Identity Provider creation of SAML type
- ✓ IAM Role creation for SAML Provider
- ✓ (Optional) Administrator Access Policy Attachment to IAM Role created

Output generated from Pulumi Stack

After generating the stack in AWS, Pulumi will show following information from the newly created stack on console:

AWS IAM Okta Role ARN
AWS IAM SAML Provider ARN

These details will be used in updating the Okta OIN AWS application for Sign On Settings.

Application Configuration

This section clarifies on the application configuration used in codebase, how to customize application execution properties based on the configuration setup in properties files.

Pulumi Project Configuration

Environment Files

- Environment files are used to set environment specific variables which are accessible from anywhere in the application.
- Purpose of these files are to maintain server/environment based values which can be different for dev, local or Prod.

Following files are used:

.env.development	For development server
.env.production	For production server

Environment files properties

AWS Details

AWS_IAM_SAML_PROVIDER_ARN	ARN Value generated for IAM SAML Identity Provider
AWS_IAM_SAML_ROLE_OKTA_ARN	ARN Value generated for IAM Role for Okta
OKTA_AWS_OIN_APP_RES_NAME	Logical name of Okta AWS app, this is used by Pulumi for internal references.
AWS_IAM_SAML_PROVIDER_RES_NAME	Logical name of IAM SAML Identity Provider, this is used by Pulumi for internal references.
AWS_IAM_SAML_ROLE_OKTA_RES_NAME	Logical name of IAM Role, this is used by Pulumi for internal references.
AWS_IAM_ADMIN_POLICY_RES_NAME	Logical name of Admin Policy assign to Role, this is used by Pulumi for internal references.
OKTA_AWS_OIN_APP_NAME	Physical name of Okta AWS app, this is used by Pulumi to create the resource with this name on Okta.
AWS_IAM_SAML_PROVIDER_NAME	Physical name of IAM SAML Identity Provider, this is used by Pulumi to create the resource with this name on AWS.
AWS_IAM_SAML_ROLE_OKTA_NAME	Physical name of IAM Role, this is used by Pulumi to create the resource with this name on AWS.
ENABLE_ADMIN_ACCESS_ON_ROLE	Option to Enable 'Administrative' Access to the IAM Role being created. Possible Values: Yes/No

Appendix

Technology Stack

Name	Type	Purpose
Development Language	Python *v 3.7*	Base Development Language for application development
Libraries (Major)	pulumi *v 1.10.1*	Pulumi SDK library
	pulumi-aws *v 1.21.0*	Pulumi AWS Provider library
Tools	Pulumi v *1.9.1*	Python based IaC Template creation tool
	JetBrains PyCharm IDE	IDE used for Python Development
	venv	Module provides support for creating lightweight "virtual environments" with their own site directories, optionally isolated from system site directories.