|  |  |
| --- | --- |
| **DOCUMENT RULES:** | |
| **Task Number / Name:** | **DevOPS-Scenarios** |
| **Task name & column name should be written:** | **Bold (CTRL+B)** |
| **Commands should be written in the after # sign:** | *Italic (CTRL+I) #hostname* |
| **Output photo should be cropped or compressed:**  **Photo could be more than one:**  **If you need extra lines, add the line next after it:** | ***Description photo should be with title bar (CTRL + I + B)*** |
| **All other text should be written:** | Standard |
| **Font name and text size:** | Calibri and 9 |
| **Group name:** | Dev\_ops\_ |
| **Student name and surname:** | Murad Abbaszade |
| **E-mail:** | [muradabbaszade6@gmail.com](mailto:muradabbaszade6@gmail.com) |
| **WhatsApp number:** | **+994703664205** |

|  |  |
| --- | --- |
| Keeper |  |
| **CommanderCLI** |  |
| EU: keeper shell --server keepersecurity.eu |  |
| **KSM Integration:Ansible** |  |
| login  Whoami  ls -l |  |
| python3 -m pip install -U keeper-secrets-manager-ansible |  |
| In CLI navigate to the folder where configuration and playbooks will be stored: cd my-playbooks Initialize config file for plugin ot use keeper\_ansible --keeper\_token [ONE TIME TOKEN] Run Playbook ansible-playbook ksm-sample-playbook.yml View password that was inserted into a file cat /tmp/my\_password |  |
| KSM Integration Docker |  |
| Install Dependencies  pip3 install keeper-secrets-manager-cli && \ apt install mysql-client-core-8.0 |  |
| nano Dockerfile  FROM mysql:debian  RUN apt-get update && \  apt-get install -y python3 python3-pip python3-venv && \  apt-get clean  RUN python3 -m pip install --upgrade pip && \  python3 -m venv /venv # Avoid system installed modules that might interfere.  ENV PATH="$VIRTUAL\_ENV/bin:$PATH"  RUN pip3 install --upgrade pip # Upgrade pip since the distro's Python might be old enough that it doesn't like to install newer modules.  RUN pip3 install keeper-secrets-manager-cli # Install Keeper Secrets Manager CLI  EXPOSE 3306  # Import our configuration from the injected file, place it to the environment variable called KSM\_CONFIG  # and then use the built in entrypoint.sh file to start it using KSM wrapper command `exec` that will  # replace environment variables that have values starting w/ `keeper://`  # NOTE: We could have just imported KSM\_CONFIG value as an environment variable into the container  # but that might lead to leaking secrets, hence we have to mount a volume that will have a  # out configuration string in the file.  # Some resources:  # https://devops.stackexchange.com/a/3904  # https://diogomonica.com/2017/03/27/why-you-shouldnt-use-env-variables-for-secret-data/  # https://pythonspeed.com/articles/docker-build-secrets/  ENTRYPOINT ["/bin/sh", "-c", "KSM\_CONFIG=$(cat /data/ksmconfig.b64) ksm exec -- /entrypoint.sh mysqld"] |  |
| docker build -t ksmexample . |  |
| docker run \ -v "$(pwd)/vol:/data" \ -e MYSQL\_ROOT\_PASSWORD='keeper://[RECORD UID]/custom\_field/rootpwd' \ -e MYSQL\_USER='keeper://[RECORD UID]/field/login' \ -e MYSQL\_PASSWORD='keeper://[RECORD UID]/field/password' \ -e MYSQL\_DATABASE='mydb' \ -p 3306:3306 \ --rm \ -it ksmexample |  |
| mysql -h 127.0.0.1 -P 3306 --protocol=tcp -u root -p |  |
| **KSM Integration:Github** |  |
| keeper shell |  |
| Generate new client for your application **Wait for Commander to finish installation**  sm client add --app [APP NAME] --unlock-ip --config-init json | Successfully generated Client Device  ====================================  Initialized Config: {"hostname": "keepersecurity.com","clien[...]ZcUefxPWewf03LLGeoei0="}  IP Lock: Disabled  Token Expires On: 2021-10-20 15:07:02  App Access Expires on: Never |
| Create new Secret | img_1.png |
|  | img_2.png |
|  | img_3.png |
| In the new or existing GitHub Action add following action: | - name: Retrieve secrets from KSM  id: ksmsecrets  uses: Keeper-Security/ksm-action@master  with:  keeper-secret-config: ${{ secrets.KSM\_CONFIG\_JSON }}  secrets: |  zOVOneDczofWFlfizjC5Qw/file/private-key.asc > file:/tmp/signing\_secret\_key\_ring\_file.asc  zOVOneDczofWFlfizjC5Qw/custom\_field/signing.keyId > env:SIGNING\_KEY\_ID  zOVOneDczofWFlfizjC5Qw/custom\_field/signing.password > env:SIGNING\_PASSWORD  zOVOneDczofWFlfizjC5Qw/custom\_field/ossrhUsername > env:OSSRH\_USERNAME  zOVOneDczofWFlfizjC5Qw/custom\_field/ossrhPassword > env:OSSRH\_PASSWORD |
| **KSM Integration GitLab** |  |
| Navigate to GitLab Settings -> CI/CD -> Variables  Create a new variable by setting Key as any name for the variable and Value should be JSON or Base64 Configuration string |  |
| In order to use Keeper Secrets Manager CLI tool, first we need to install it from PyPi registry. This can be achieved by adding following line to the before\_script: area: | before\_script:  - python3 -V # Python3 and pip module are required in the image  - python3 -m pip install keeper-secrets-manager-cli |
| Inject Field Secret value into environment variable using Keeper Notation: | - export MY\_PWD=$(ksm secret notation keeper://6ya\_fdc6XTsZ7i7x9Jcodg/field/password) |
| Inject Custom Field value: | - export MY\_ISBNCODE=$(ksm secret notation keeper://6ya\_fdc6XTsZ7i7x9Jcodg/custom\_field/isbncode) |
| Download file: | - ksm secret download -u 6ya\_fdc6XTsZ7i7x9Jcodg --name "mykey.pub" --file-output "/tmp/mykey.pub" |
| Complete example of the pipeline: | image: python:latest  before\_script:  - python3 -V # Print out python version for debugging  - python3 -m pip install keeper-secrets-manager-cli  job1:  stage: build  script:  - export MY\_PWD=$(ksm secret notation keeper://6ya\_fdc6XTsZ7i7x9Jcodg/field/password)  - export MY\_ISBNCODE=$(ksm secret notation keeper://6ya\_fdc6XTsZ7i7x9Jcodg/custom\_field/isbncode)  - ksm secret download -u 6ya\_fdc6XTsZ7i7x9Jcodg --name "mykey.pub" --file-output "/tmp/mykey.pub"  - file /tmp/mykey.pub |
| KSM Integration:Hashicorp Vault |  |
| Create plugin directory: mkdir plugins |  |
| wget -q https://github.com/Keeper-Security/secrets-manager/releases/download/vault-plugin-secrets-ksm%2Fv1.0.0/vault-plugin-secrets-ksm\_1.0.0\_linux\_amd64.zip |  |
| unzip \*.zip -d plugins |  |
| Start Hashicorp Vault server in dev mode | vault server -dev -dev-root-token-id=root -dev-plugin-dir=/root/hashicorp-vault/plugins |
| In the new Terminal Navigate to the working directory | cd hashicorp-vault |
| Set vault server url in the env vars | export VAULT\_ADDR=http://127.0.0.1:8200 |
| Enable KSM Vault plugin | vault secrets enable -path=myksm vault-plugin-secrets-ksm |
| Check that KSM plugin was registered | vault secrets list |
| Configure KSM | vault write myksm/config ksm\_config=[KSM CONFIG IN BASE 64] |
| List secrets | vault list myksm/records |
| Read record by UID | vault read -format=json myksm/record uid=[RECORD UID] |
| Get TOTP values from record | vault read -format=json myksm/record/totp uid=[RECORD UID] |
| Deregister the plugin Make sure that the server is running | vault secrets disable myksm  vault plugin deregister secret vault-plugin-secrets-ksm  vault plugin list secret |
| **KSM Integration:PowerShell** |  |
| pwsh |  |
| **Install Microsoft PowerShell Secret Management Module:** | Install-Module -Name Microsoft.PowerShell.SecretManagement -Force |
| Install Keeper Secrets Manager for PowerShell | Install-Module -Name SecretManagement.Keeper -Force |
| Install Secrets Management extension | Install-Module -Name Microsoft.Powershell.SecretStore -Force |
| Register a Vault to use for Configuration Storage | Register-SecretVault -Name MyLocalStore -ModuleName Microsoft.Powershell.SecretStore |
| Register the Keeper Vault | Register-KeeperVault -Name MyKeeperVault -LocalVaultName MyLocalStore -OneTimeToken [ONE-TIME TOKEN] |
|  |  |
| Listing Secrets | Get-SecretInfo -Vault MyKeeperVault |
| Getting a Single Secret | Get-Secret "[SECRET NAME]" -AsPlainText |
| Get an Individual Secret Value From a Secret | Get-Secret "[SECRET NAME].[FIELD NAME]" -AsPlainText |
| Set a Value to a Secret | Set-Secret "[SECRET NAME].[FIELD NAME]" -Secret [VALUE TO SET] -Vault MyKeeperVault |
| Download a File | Get-Secret "[SECRET NAME]".files["FILE NAME"] | Set-Content -Path <FILE PATH> -AsByteStream |
| Unregistering Vaults | Unregister-SecretVault \* |
| Delete Module | Uninstall-Module -Name SecretManagement.Keeper |
| Reset Secret Store | Reset-SecretStore -PassThru |
| KSM Integration:Terraform |  |
| Navigate to the example folder | cd terraform-example |
| Initialize Terraform | terraform init |
| Run terraform plan to check for the current state of the Terraform state | terraform plan |
| Apply new changes using | terraform apply -auto-approve |
| Initialize and apply Terraform Configuration | terraform init && terraform apply |
| In order to connect to MySQL, we need to install MySQL Client | apt install -y mysql-client-core-5.7 |
| Connect to MySQL: | mysql -h 127.0.0.1 -P 3306 --protocol=tcp -p |
| **KSM SDK .Net** |  |
| Generate the standard console application | dotnet new console |
| Install KSM Package: | dotnet add package Keeper.SecretsManager |
| Modify a program file dotnet run | using System;  using System.Threading.Tasks;  using SecretsManager;  using System.Text.Json;  namespace ConsoleApp1  {  public static class Program  {  const string ONE\_TIME\_TOKEN = "[ONE TIME TOKEN]";    private static void Main()  {  GetSecrets().Wait();  }  private static async Task GetSecrets()  {  var storage = new LocalConfigStorage("ksm-config-demo.json");  SecretsManagerClient.InitializeStorage(storage, ONE\_TIME\_TOKEN);  var options = new SecretsManagerOptions(storage);  // Get all secrets  var secrets = await SecretsManagerClient.GetSecrets(options);  // Get first record  var firstRecord = secrets.Records[0];  // Turn record to json string  string jsonString = JsonSerializer.Serialize(firstRecord, new JsonSerializerOptions { WriteIndented = true });  Console.WriteLine(jsonString);  }  }  } |
| Overwrite existing application code | using System;  using System.Threading.Tasks;  using SecretsManager;  using System.Text.Json;  namespace ConsoleApp1  {  public static class Program  {  const string FOLDER\_UID = "[FOLDER UID]";  private static void Main()  {  CreateLoginRecord().Wait();  }  private static async Task CreateLoginRecord()  {    // Note: Reusing configuration that was generated in the first step  var storage = new LocalConfigStorage("ksm-config-demo.json");  var options = new SecretsManagerOptions(storage);  var newRecord = new KeeperRecordData();  newRecord.type = "login";  newRecord.title = "Sample Katacoda KSM Record";  newRecord.fields = new[]  {  new KeeperRecordField { type = "login", value = new[] { "username@email.com" } },  new KeeperRecordField { type = "password", value = new[] { "Pa$$word123" } },  };  newRecord.notes = "\tThis record was created\n\tvia KSM Katacoda .NET C# Example";  var recordUid = await SecretsManagerClient.CreateSecret(options, FOLDER\_UID, newRecord);    Console.WriteLine("Saved record UID: [" + recordUid + "]");  }  }  } |
| Overwrite existing application code | using SecretsManager;  using System;  using System.Collections.Generic;  using System.Threading.Tasks;  namespace ConsoleApp1  {  public static class Program  {  private static void Main()  {  GetSecrets().Wait();  }  const string FOLDER\_UID = "[FOLDER UID]";  private static async Task GetSecrets()  {  // Note: Reusing configuration that was generated in the first step  var storage = new LocalConfigStorage("ksm-config-demo3.json");  SecretsManagerClient.InitializeStorage(storage, ONE\_TIME\_TOKEN);  var options = new SecretsManagerOptions(storage);  var newRecord = new KeeperRecordData();  newRecord.type = "MyCustomType";  newRecord.title = "Sample Katacoda KSM Record (Custom record type)";  newRecord.fields = new[]  {  new KeeperRecordField { type = "login", value = new[] { "username@email.com" }, required = true, label = "My Login"},  new KeeperRecordField { type = "password", value = new[] { "Pa$$word123" }, required = true, label = "My Password"},  new KeeperRecordField { type = "securityQuestion", value = new[]  {  new Dictionary<string, string>  {  {"question", "What is the question?"},  { "answer", "This is the answer!" }  }  }, label = "My Security Question & Answer", privacyScreen = false},  new KeeperRecordField { type = "phone", value = new[]  {  new Dictionary<string,string>  {  { "region", "US" },  { "number", "510-444-3333" },  { "ext", "2345" },  { "type", "Mobile" }  }  }, label = "My Private Phone", privacyScreen = true  },  new KeeperRecordField { type = "date", value = new[] {(object) 1641934793000 }, label = "My Birthdate" },  new KeeperRecordField { type = "name", value = new[]  {  new Dictionary<string, string>  {  {"first", "John"},  {"middle", "Patrick"},  {"last", "Smith"}  }  }  },  new KeeperRecordField {  type = "otp",  value = new[] { "otpauth://totp/Example:alice@google.com?secret=JBSWY3DPEHPK3PXP&issuer=Example"},  label = "MyTOTP"  }  };  newRecord.notes = "\tThis custom type record was created\n\tvia KSM Katacoda .NET C# Example";    var recordUid = (await SecretsManagerClient.CreateSecret(options, FOLDER\_UID, newRecord));    Console.WriteLine("Saved record UID: [" + recordUid + "]");  // Retrieve saved record from the server  var savedSecret = await SecretsManagerClient.GetSecrets(options, new []{recordUid});    var oneTimeCodeUrl = Notation.GetValue(savedSecret, "keeper://" + recordUid + "/field/MyTOTP");    var oneTimeCode = CryptoUtils.GetTotpCode(oneTimeCodeUrl);  Console.Write("One time code: " + oneTimeCode.Code + " (TTL: " + oneTimeCode.TimeLeft + " sec.)");  }  }  } |
| **KSM SDK GoLang** |  |
| Install Secrets Manager GoLang Package | go get github.com/keeper-security/secrets-manager-go/core |
| package main  // Import Secrets Manager  import ksm "github.com/keeper-security/secrets-manager-go/core"  func main() {  sm := ksm.NewSecretsManagerFromSettings("[ONE TIME TOKEN]", "", true)  allRecords, \_ := sm.GetSecrets([]string{}) // Retrieve all records from Secrets Manager    password := allRecords[0].Password() // Get password from first record  print("My password from Keeper: ", password, "\n") // WARNING: Avoid logging sensitive data  }  go run ksm-example-ott.go |  |
| package main  // Import Secrets Manager  import ksm "github.com/keeper-security/secrets-manager-go/core"  func main() {  // JUST FOR DEMO PURPOSE, WE ARE PLACING THIS HARDCODED STRING IN THE CODE  // NEVER PASTE YOUR CONFIGURATION OR ONE TIME TOKE.  // ALWAYS USE SOME OTHER SECURE STORAGE TO KEEP THIS VALUES SECURE  configStr := `{  "appKey": "8Kxn5SvtkRSsaYIur7mHKtLq3NFNB7AZRa9cqi2PSQE=",  "clientId": "AQmmLePs6I9Vcr23MclXzvNJ0ZV3x6deAPyuHD9zrDmXv03JnO6Aadalikg17Px7u+DLbeY29C/OVoe4AcyfIA==",  "privateKey": "MIGHAgEAMBMGByqGSM49AgEGCCaGSM49AwEHBG0wawIBAQQgWR9b5lm9NCGRysQuLsx6lSBJd63x1TjwCSZnSmBRpn6hbANCAAR4iL27W2Qbzqv/2e4i37boapTQc4aBDuf8XlMy5w84X0vz/yLvluXwLGzKZnYg/gYhCnbCXdftIGICOA9deZcP",  "serverPublicKeyId": "7"  }`  config := ksm.NewMemoryKeyValueStorage(configStr) // Credentials in Base64 or Json  sm := ksm.NewSecretsManagerFromConfig(config)    allRecords, \_ := sm.GetSecrets([]string{}) // Retrieve all records from Secrets Manager  password := allRecords[0].Password() // Get password from first record  print("My password from Keeper: ", password, "\n") // WARNING: Avoid logging sensitive data  } |  |
| **KSM SDK:Java** |  |
| Create Java Project | gradle init \  --project-name ksmsample \  --package com.keepersecurity.ksmsample \  --type java-application \  --dsl groovy \  --test-framework junit |
| plugins {  id 'java'  id 'application'  }  group 'com.keepersecurity.secrets-manager'  version '1.0-SNAPSHOT'  repositories {  mavenCentral()  }  dependencies {  implementation 'com.keepersecurity.secrets-manager:core:16.3.+'  }  application {  // Define the main class for the application.  mainClassName = project.hasProperty("mainClass") ? project.getProperty("mainClass") : "NULL"  } |  |
| Create file | touch src/main/java/com/keepersecurity/ksmsample/KSMListRecords.java |
|  |  |
| gradle -PmainClass=com.keepersecurity.ksmsample.KSMListRecords run |  |
|  |  |
| gradle -PmainClass=com.keepersecurity.ksmsample.KSMAddRecord run |  |
| **KSM SDK:Javascript** |  |
| Initialize NPM package.json file: |  |
| npm install |  |
|  |  |
|  |  |
| **KSM SDK Python** |  |
| touch main.py |  |
| python3 main.py |  |
| touch main-cache.py |  |
| python3 main-cache.py |  |
| touch main-create-record.py |  |
| python3 main-create-record.py |  |
| touch main-create-record-upload-file.py |  |
| python3 main-create-record-upload-file.py |  |
| **Secrets Management in Commander** |  |
| Install Python  apt install python3-pip -y |  |
| pip3 install pyOpenSSL --upgrade |  |
| pip3 install keeper-secrets-manager-core keepercommander |  |
| keeper shell |  |
| View all available commands related to Secrets Manager | secrets-manager |
| List all available applications | secrets-manager app list |
| Get application by name | secrets-manager app get [APP NAME or UID] |
| Create application | secrets-manager app create SampleApplication |
| Remove application | secrets-manager app remove SampleApplication --purge |
| Add one editable record to an Application | secrets-manager share add --app SampleApplication --secret [RECORD UID] --editable |
| Add one editable shared folder to an Application | secrets-manager share add --app SampleApplication --secret [SHARED FOLDER UID] --editable |
| Add client and retrieve One-Time Access Token | sm client add --app SampleApplication --unlock-ip --name SampleClientName |
| Add client and retrieve initialized config JSON string | sm client add --app SampleApplication --unlock-ip --name SampleClientName --config-init json |
| Add client and retrieve initialized config Base64 string | sm client add --app SampleApplication --unlock-ip --name SampleClientName --config-init b64 |
| **Secrets Manager CLI** |  |
| pip3 install keeper-secrets-manager-cli |  |
| ksm profile init --token XX:XXXX |  |
| List All Secrets | ksm secret list |
| Get Individual Secret | ksm secret get [UID] |
| Update a password | ksm secret update --uid [UID] --field password=MyNewPassword |
| Set config to environment variable with the initialized config | export KSM\_CONFIG=[KSM CONFIG in JSON OR BASE 64] |
| Get password and store it in env variable | export mypwd=$(ksm secret notation keeper://[RECORD UID]/field/password) |
| Download a file: | ksm secret download -u [RECORD UID] --name "mykey.pub" --file-output "/tmp/mykey.pub" |