Aws secret manager

// get secret Manager keys:

[HttpGet]

[Route("SercretKey")]

public async Task<IActionResult> GetSecrets()

{

//string SecretName = "example/dev/key";

//string region = "ap-south-1";

//IAmazonSecretsManager client = new AmazonSecretsManagerClient(RegionEndpoint.GetBySystemName(region));

//GetSecretValueRequest request = new GetSecretValueRequest

//{

// SecretId = SecretName,

// // VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT if unspecified.

//};

//// for key 3

//// for key 2

//string SecretProdName = "example/prod/key";

//string region1 = "ap-south-1";

//IAmazonSecretsManager client2 = new AmazonSecretsManagerClient(RegionEndpoint.GetBySystemName(region1));

//GetSecretValueRequest request2 = new GetSecretValueRequest

//{

// SecretId = SecretProdName,

// // VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT if unspecified.

//};

// string SecretName3= "prod/test/key";

// string SecretName3 = "test/Api/keys";

// string SecretName3= "sai/test/key";

//string region3 = "ap-south-1";

//IAmazonSecretsManager client3 = new AmazonSecretsManagerClient(RegionEndpoint.GetBySystemName(region3));

//GetSecretValueRequest request3 = new GetSecretValueRequest

//{

// SecretId = SecretName3,

// // VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT if unspecified.

//};

// GetSecretValueResponse response;

//GetSecretValueResponse response2;

//GetSecretValueResponse response3;

// for sai test api keys:

string secretName4 = "sai/test/key";

string region4 = "ap-south-1";

IAmazonSecretsManager client = new AmazonSecretsManagerClient(RegionEndpoint.GetBySystemName(region4));

GetSecretValueRequest request4 = new GetSecretValueRequest

{

SecretId = secretName4,

VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT if unspecified.

};

GetSecretValueResponse response4;

try

{

//response = await client.GetSecretValueAsync(request);

//response2 = await client2.GetSecretValueAsync(request2);

//response3 = await client3.GetSecretValueAsync(request3);

response4 = await client.GetSecretValueAsync(request4);

}

catch (Exception ex)

{

throw ex;

}

// // for secret 1 Encrypted data

// var secret = response.SecretString.ToString();

// var userSecretProvider = \_dataProtectionProvider.CreateProtector(SecretName);

// byte[] secretEncoding = userSecretProvider.Protect(System.Text.Encoding.UTF8.GetBytes(secret));

//var encryptedData1 = Convert.ToBase64String(secretEncoding);

// // for encrypt dat for secret2

// var secret2 = response2.SecretString.ToString();

// var userSecretProtector2 = \_dataProtectionProvider.CreateProtector(SecretProdName);

// byte[] secret2ByteInfo = userSecretProtector2.Protect(System.Text.Encoding.UTF8.GetBytes((secret2)));

// var encryptedData2 = Convert.ToBase64String(secret2ByteInfo);

// for encrypt data for secret3:

var secret4 = response4.SecretString.ToString();

var userSecretProtector4 = \_dataProtectionProvider.CreateProtector(secretName4);

byte[] secret4ByteInfo = userSecretProtector4.Protect(System.Text.Encoding.UTF8.GetBytes((secret4)));

var encryptedData3 = Convert.ToBase64String(secret4ByteInfo);

// decrypt:

byte[] encryptDataByte = Convert.FromBase64String(encryptedData3);

var decrept1Base64= userSecretProtector4.Unprotect(encryptDataByte);

string decrept4 = System.Text.Encoding.UTF8.GetString(decrept1Base64);

SecretModel member = JsonSerializer.Deserialize<SecretModel>(decrept4.ToString());

// SecretModel model = new SecretModel();

// string json = JsonSerializer.Serialize(decrept3);

SecretModel model = JsonConvert.DeserializeObject<SecretModel>(decrept4);

Console.WriteLine(model);

//var obj = new {

// secret,

// secret2,

// secret3

//};

// return Ok( encryptedData3);

return Ok(new {model,encryptedData3});

// return Ok(secret3);

// return Ok(new { secret, secret2, secret3 });

}

//[HttpGet]

//[Route("EncryptData")]

//public IActionResult EncrptyData()

//{

// string plaintext = "arn:aws:secretsmanager:ap-south-1:242909465937:secret:test/Api/keys-2oZZ1J";

// // // encrypt

// // var protection = \_dataProtectionProvider.CreateProtector("user-data-encryption-key");

// // byte[] Encryptdata = protection.Protect(System.Text.Encoding.UTF8.GetBytes(userData));

// // var data = Convert.ToBase64String(Encryptdata);

// // Console.WriteLine(data);

// // // decrypt:

// // byte[] userInfo = Convert.FromBase64String(data);

// // var decrypted = protection.Unprotect(userInfo);

// //string decrypteddata = System.Text.Encoding.UTF8.GetString(decrypted);

// // Console.WriteLine(decrypteddata);

// // return Ok(new {data,decrypteddata});

// // Symmetric encryption :

// string base64key = string.Empty;

// using (Aes aes = Aes.Create())

// {

// aes.KeySize = 256;

// aes.GenerateKey();

// base64key = Convert.ToBase64String(aes.Key);

// //

// aes.Padding = PaddingMode.Zeros;

// aes.Key = Convert.FromBase64String(base64key);

// aes.GenerateIV();

// var IVkey = Convert.ToBase64String(aes.IV);

// ICryptoTransform cryptoTransform = aes.CreateEncryptor();

// byte[] encryptedData;

// using (MemoryStream ms = new MemoryStream())

// {

// using (CryptoStream cs = new CryptoStream(ms, cryptoTransform, CryptoStreamMode.Write))

// {

// using (StreamWriter sw = new StreamWriter(cs))

// {

// sw.Write(plaintext);

// }

// encryptedData = ms.ToArray();

// };

// }

// string EncryptedData = Convert.ToBase64String(encryptedData);

// string PlainText = "";

// if (EncryptedData == plaintext)

// {

// ICryptoTransform decrypt = aes.CreateDecryptor();

// byte[] chiper = Convert.FromBase64String(EncryptedData);

// using (MemoryStream ms = new MemoryStream(chiper))

// {

// using (CryptoStream cs = new CryptoStream(ms, cryptoTransform, CryptoStreamMode.Write))

// {

// using (StreamReader sw = new StreamReader(cs))

// {

// PlainText = sw.ReadToEnd();

// }

// };

// }

// }

// return Ok( new { EncryptedData, plaintext });

// // return Ok( new { EncryptedData } );

// }

// // SYMMETRIC DECRYPTION:

//}

//// test encryption:

//[HttpGet]

//[Route("GetGenerateKey")]

//public string GenerateKey()

//{

// string base64key = string.Empty;

// using (Aes aes = Aes.Create())

// {

// aes.KeySize = 256;

// aes.GenerateKey();

// base64key = Convert.ToBase64String(aes.Key);

// }

// return base64key;

//}

//[HttpPost]

//[Route("Encrypt")]

//public string Encrypt(string key, string plaintext, out string IVkey)

//{

// using (Aes aes = Aes.Create())

// {

// aes.Padding = PaddingMode.Zeros;

// aes.Key = Convert.FromBase64String(key);

// aes.GenerateIV();

// IVkey = Convert.ToBase64String(aes.IV);

// ICryptoTransform cryptoTransform = aes.CreateEncryptor();

// byte[] encryptedData;

// using (MemoryStream ms = new MemoryStream())

// {

// using (CryptoStream cs = new CryptoStream(ms, cryptoTransform, CryptoStreamMode.Write))

// {

// using (StreamWriter sw = new StreamWriter(cs))

// {

// sw.Write(plaintext);

// }

// encryptedData = ms.ToArray();

// };

// }

// return Convert.ToBase64String(encryptedData);

// }

//}

//// decrypter:

//public static string Decrypt(string CipherText, string key, string IVkey)

//{

// using (Aes aes = Aes.Create())

// {

// aes.Padding = PaddingMode.Zeros;

// aes.Key = Convert.FromBase64String(key);

// aes.IV = Convert.FromBase64String(IVkey);

// IVkey = Convert.ToBase64String(aes.IV);

// ICryptoTransform cryptoTransform = aes.CreateDecryptor();

// string PlainText = "";

// byte[] chiper = Convert.FromBase64String(CipherText);

// using (MemoryStream ms = new MemoryStream(chiper))

// {

// using (CryptoStream cs = new CryptoStream(ms, cryptoTransform, CryptoStreamMode.Write))

// {

// using (StreamReader dr = new StreamReader(cs))

// {

// PlainText = dr.ReadToEnd();

// }

// };

// }

// return PlainText;

// }

//}

[HttpGet]

[Route("Secret")]

public async Task<IActionResult> GetSecret()

{

string secretName = "sai/test/key";

string region = "ap-south-1";

IAmazonSecretsManager client = new AmazonSecretsManagerClient(RegionEndpoint.GetBySystemName(region));

GetSecretValueRequest request = new GetSecretValueRequest

{

SecretId = secretName,

VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT if unspecified.

};

GetSecretValueResponse response;

try

{

response = await client.GetSecretValueAsync(request);

}

catch (Exception e)

{

// For a list of the exceptions thrown, see

// https://docs.aws.amazon.com/secretsmanager/latest/apireference/API\_GetSecretValue.html

throw e;

}

string secret = response.SecretString.ToString();

// SecretModel member = JsonSerializer.Deserialize<SecretModel>(secret.ToString());

// SecretModel model = new SecretModel();

// string json = JsonSerializer.Serialize(decrept3);

// SecretModel model = JsonConvert.DeserializeObject<SecretModel>(secret.ToString());

var userSecretProtector = \_dataProtectionProvider.CreateProtector(secretName);

byte[] secret4ByteInfo = userSecretProtector.Protect(System.Text.Encoding.UTF8.GetBytes((secret)));

var encryptedData = Convert.ToBase64String(secret4ByteInfo);

// decrypt:

byte[] encryptDataByte = Convert.FromBase64String(encryptedData);

var decrept1Base64 = userSecretProtector.Unprotect(encryptDataByte);

string decrept = System.Text.Encoding.UTF8.GetString(decrept1Base64);

// SecretModel member = JsonSerializer.Deserialize<SecretModel>(decrept);

// SecretModel model = new SecretModel();

// string json = JsonSerializer.Serialize(decrept3);

SecretModel model = JsonConvert.DeserializeObject<SecretModel>(decrept);

Console.WriteLine(model);

return Ok(new {model,encryptedData});

// Your code goes here

}