Mazure

bblommers

Jan 24, 2024

CONTENTS

CHAPTER

ONE

WELCOME TO THE MAZURE DOCUMENTATION!

Mazure is a proxy that mocks the Azure SDK.

Using a mocked Azure will:

- Save you time, as you no longer have to wait for long-running HTTP-requests to Azure
- Save you time, as you no longer have to write your own mocks
- Save you money, as you no longer have to create resources in Azure (and no longer have to remember to delete them after every test!)
- Gain you confidence that you're using the Azure SDK correctly

CHAPTER

TWO

USAGE

1. Run the Mazure Docker image:

` docker run -p 5005:5005 mazureproxy/proxy:0.0.2 `

2. Configure the SDK of your choice to use the proxy:

SDK Specific Configuration.

CHAPTER

THREE

SUPPORTED SERVICES

Please see our *list of supported services here*.

Want to see support for another service? Let us know!

3.1 SDK Specific Configuration

Every SDK has a slightly different way to configure a proxy. Please see the examples below to get an idea how to do this in the language of your choice.

Have you found a different way to configure a proxy for a specific language, or want to contribute an example for another language, please see the *Edit on Github*-button at the top of the page.

3.1.1 Python

The Azure SDK for Python needs the correct SSL certificate before it trusts the Mazure-proxy.

Download the certificate from our Github: https://github.com/getmazure/mazure/blob/main/mazure/mazure_proxy/ca.crt

Run your tests like you normally would, but set the following environment variables:

```
REQUESTS_CA_BUNDLE=/path/to/ca.crt HTTPS_PROXY=http://localhost:5005 pytest -sv tests/
```

3.1.2 Dotnet

The Azure SDK for DotNet needs to told that it can trust our certificate.

Configure a custom HTTP-client like this:

```
var handler = new HttpClientHandler();
handler = new HttpClientHandler
    {
        ClientCertificateOptions = ClientCertificateOption.Manual,
        ServerCertificateCustomValidationCallback =
            (httpRequestMessage, cert, cetChain, policyErrors) => true
    };
var blobClientOptions = new BlobClientOptions();
blobClientOptions.Transport = new HttpClientTransport(handler);

var blobServiceClient = new BlobServiceClient(
    new Uri("https://storage_account.blob.core.windows.net"),
    blobClientOptions
);
```

Warning: Clients for other services may have a different way to configure the ClientOptions - see the Azure documentation for more details.

Setup the dotnet tests like you would normally:

```
dotnet restore Application/
```

But configure the actual test execution to use the proxy:

```
ALL_PROXY="http://localhost:5005" dotnet test Application/
```

3.1.3 CLI

The Azure CLI needs the Mazure SSL certificate to trust the proxy.

Download the certificate from our Github: https://github.com/getmazure/mazure/blob/main/mazure/mazure_proxy/ca.crt

Run your tests like you normally would, but set the following environment variables:

```
REQUESTS_CA_BUNDLE=/path/to/ca.crt HTTPS_PROXY=http://localhost:5005 az group list
```

3.1.4 Java

The Azure SDK for Java needs two things to connect to the Mazure-proxy:

- The HTTP-Client needs to be configured to use our proxy, and
- The Mazure SSL certificate needs to be added to the Java keytool

Download our SSL certificate from Github: https://github.com/getmazure/mazure/blob/main/mazure/mazure_proxy/ca.crt

Add it to the keytool using this command:

```
sudo keytool -import -noprompt -alias mazure -keystore $JAVA_HOME/lib/security/cacerts  -
↪file path/to/ca.crt -storepass "changeit"
```

Configure your Azure tests to use a custom HTTP-client:

```
Configuration configuration = new Configuration()
    .put("java.net.useSystemProxies", "true")
    .put("http.proxyHost", "localhost")
    .put("http.proxyPort", "5005");

HttpClient nettyHttpClient = new NettyAsyncHttpClientBuilder()
    .configuration(configuration)
    .build();

AzureProfile profile = new AzureProfile(AzureEnvironment.AZURE);
BasicAuthenticationCredential credential = new FakeBasicAuthenticationCredential("test",
↪"pass");
AzureResourceManager azure = AzureResourceManager
    .configure()
    .withHttpClient(nettyHttpClient)
    .authenticate(credential, profile)
    .withDefaultSubscription();
```

3.2 Supported Services

The following services/operations are supported. Please let us know if you'd like to support for an operation that is not listed here.

- Resource Management:
  - Resource Manager:
    - Resource Groups:
      - Check Existence
      - Create
      - Delete
      - Get
      - List
    - Subscriptions:

        - List
        - ListLocations
- Storage Resource Provider:
    - Check Storage Account Name Availability
    - Create Storage Account
    - List Storage Accounts
    - Get Storage Account Keys
- Storage:
    - Blob Service:
        - List Containers
        - Create Container
        - Delete Container
        - List Blobs
        - Put Blob
        - Get Blob