Lowkey Vault

Azure Key Vault on a dev machine

Prepared for Free Software Conference 2022

Presented by Istvan Zoltan Nagy

2022

Contents

Context

What was the issue? Why is it important?

What is Azure Key Vault and why should you use it?

- Azure Key Vault is a managed service available for Azure users.
- Secure storage for keys, secrets, certificates.
- Supports hardware protection (Hardware Security Modules)
- Microsoft started to engage open-source software. They have even announced a program giving service credits for open-source projects in 2021.

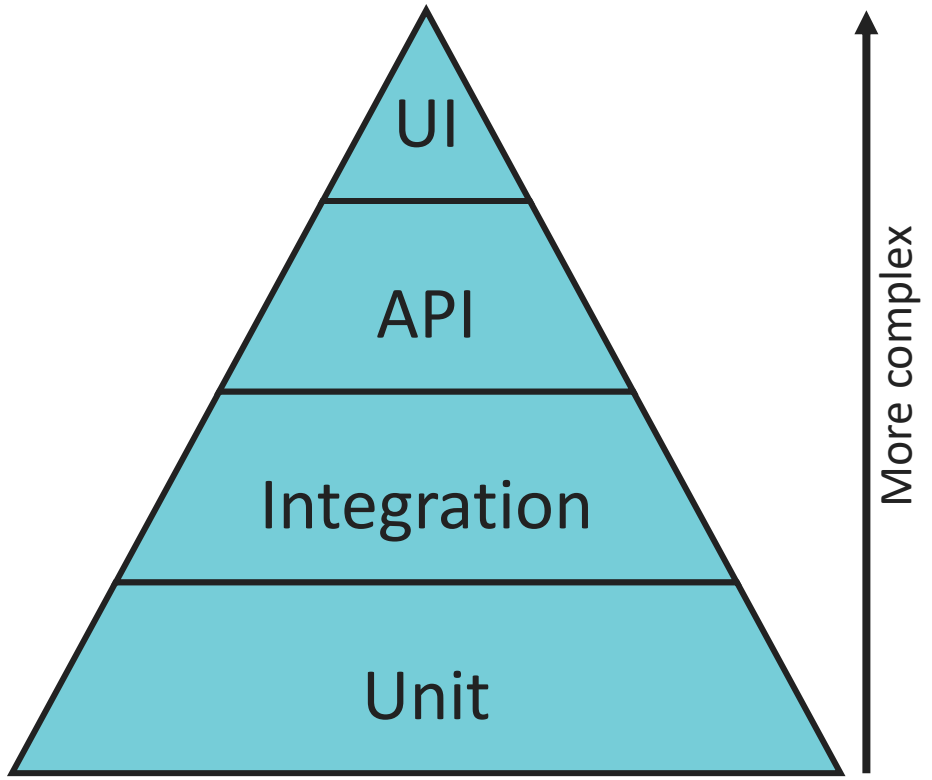
Testing pyramid

We are testing different things on different levels:

- Internal logic
- Integration with external services/dependencies
- API response for given requests
- End-to-end behaviour

Higher-level tests are slower, more complex, using real services instead of test doubles.

We must push our tests to the lower levels as much as possible.



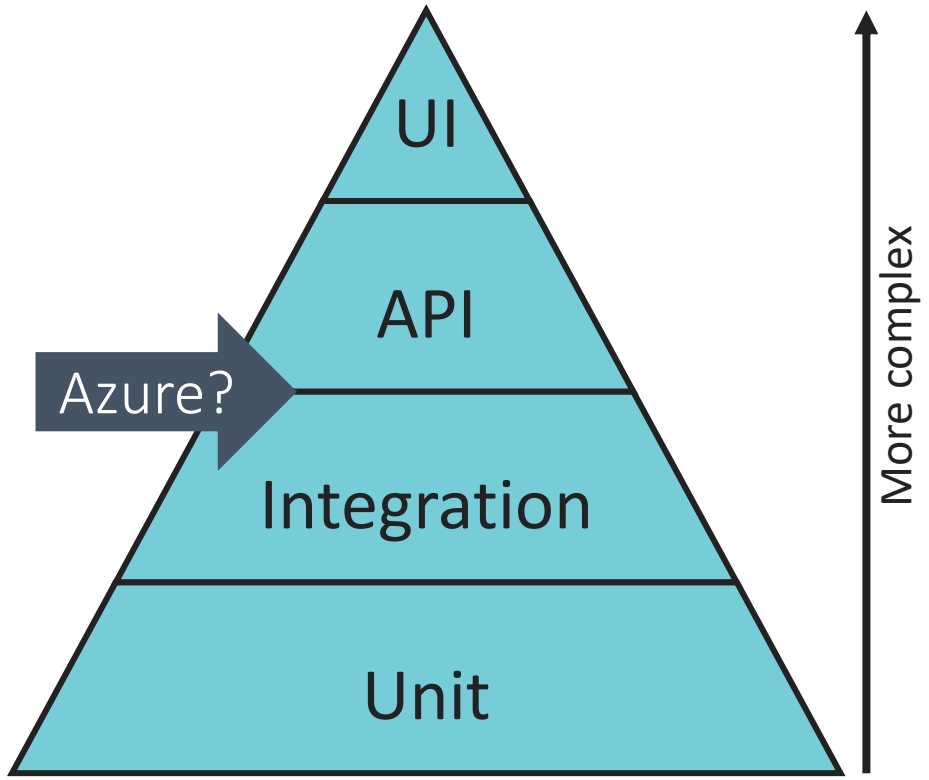
Where can we test Azure integration?

External dependencies can be tested in Integration/API layer (not just Azure).

It is challenging to:

- Set up these dependencies from tests
- Clean up after test execution
- Solve isolation between test cases
- Solve isolation between test runs

If we exclude this part, how will we know whether it works or not?

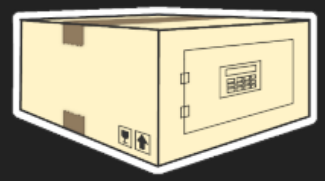


Lowkey Vault

What is it? How does it solve the issue?

What is Lowkey Vault?

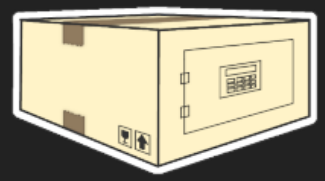
- An Azure Key Vault test double (Fake, even the colour is Celtic blue)
- MIT License
- Covers API v7.2 for keys and secrets
- Executable Jar or Docker container
- Can use multiple vaults in one container
- Client is compatible with official Azure clients using HTTP client provider
- Testcontainers integration using third-party module
- Provides a management API to allow dynamic vault create/delete/change



LOWKEY VAULT

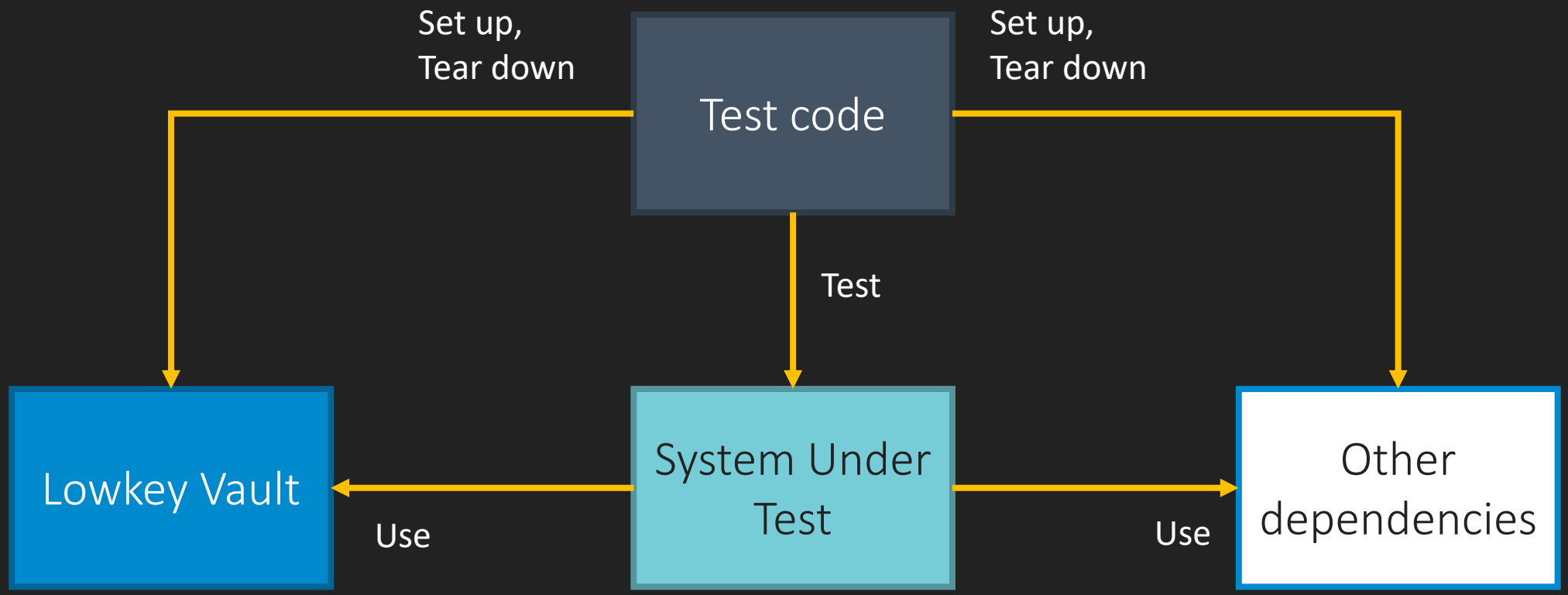
What does the Lowkey Vault Client give us?

- Allows convenient use of Lowkey Vault management API
- Solves routing issues
  - In case you need to use multiple vaults at the same time
  - When the vault content is restored from file, allows using random port for the container
- Ignores certificate issues caused by self-signed vault certificate
- Automatically
  - Adds authentication headers
  - Sets API version



LOWKEY VAULT

How does it solve our issue?



Demo

Let’s test the theory!

Starting Lowkey Vault: java -jar lowkey-vault-app-1.2.2.jar

```
                           Lowkey Vault version:  (v1.2.2)
                                   Boot version:  (v2.6.7)

2022-05-04 13:19:57.896  INFO 14992 --- [           main] c.g.nagyesta.lowkeyvault.LowkeyVaultApp  : Starting LowkeyVaultApp using Java 11.0.12 on EPHUDEBW0016 with PID 14992 (C:\Users\Istvan_Nagy\.gradle\caches\modules-2\files-2.1\com.github.nagyesta.lowkey-vault\lowkey-vault-app\1.2.2\3e43ddc5afcbfdd6a9782fdeb829cbe919c3b0e5\lowkey-vault-app-1.2.2.jar started by Istvan_Nagy in C:\Projects\Bench\Github\lowkey-vault-example)
2022-05-04 13:19:57.900  INFO 14992 --- [           main] c.g.nagyesta.lowkeyvault.LowkeyVaultApp  : No active profile set, falling back to 1 default profile: "default"
2022-05-04 13:19:58.833  INFO 14992 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2022-05-04 13:19:58.844  INFO 14992 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-05-04 13:19:58.844  INFO 14992 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.62]
2022-05-04 13:19:58.981  INFO 14992 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-05-04 13:19:58.981  INFO 14992 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1039 ms
2022-05-04 13:19:59.045  INFO 14992 --- [           main] c.g.n.lowkeyvault.AppConfiguration       : Starting up vault with port: 8443 , auto-registering vaults: 'primary,secondary'
2022-05-04 13:19:59.060  INFO 14992 --- [           main] c.g.n.lowkeyvault.AppConfiguration       : Vaults registered!
2022-05-04 13:19:59.956  INFO 14992 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2022-05-04 13:19:59.969  INFO 14992 --- [           main] c.g.nagyesta.lowkeyvault.LowkeyVaultApp  : Started LowkeyVaultApp in 2.451 seconds (JVM running for 2.82)
```

<epam>

The example app: Entry point

```java
@SpringBootApplication
public class LowkeyVaultExampleApplication {

    /**
     * Entry point of the application.
     *
     * @param args The command line arguments.
     */
    public static void main(final String[] args) {
        SpringApplication.run(LowkeyVaultExampleApplication.class, args);
    }

}
```

The example app: Azure Key Vault specific repository interfaces

```java
/**
 * Repository interface encrypting/decrypting data using Azure Key Vault.
 */
public interface AzureKeyRepository {

    /**
     * Decrypts the cipher using Azure Key Vault.
     *
     * @param cipher encrypted data.
     * @return decrypted data
     */
    1 usage   1 implementation
    String decrypt(byte[] cipher);

    /**
     * Encrypts the clear text using Azure Key Vault.
     *
     * @param clearText Plain text that should be encrypted.
     * @return encrypted data
     */
    1 usage   1 implementation
    byte[] encrypt(String clearText);

}
```

```java
/**
 * Repository interface fetching DB connection credentials from Azure Key Vault.
 */
public interface AzureSecretRepository {

    /**
     * Loads DB connection URL from Azure Key Vault.
     *
     * @return connection URL
     */
    1 usage   1 implementation
    String getDatabaseConnectionUrl();

    /**
     * Loads DB username from Azure Key Vault.
     *
     * @return username
     */
    1 usage   1 implementation
    String getDatabaseUserName();

    /**
     * Loads DB password from Azure Key Vault.
     *
     * @return password
     */
    1 usage   1 implementation
    String getDatabasePassword();
}
```

DEMO

The example app: Key repository implementation

```java
@Component
public class AzureKeyRepositoryImpl implements AzureKeyRepository {

    3 usages
    private final KeyClient keyClient;
    3 usages
    private final Function<JsonWebKey, CryptographyClient> cryptographyClientProvider;
    2 usages
    @Value("${key.name}")
    private String vaultKeyName;

    @Autowired
    public AzureKeyRepositoryImpl(final KeyClient keyClient,
                                  final Function<JsonWebKey, CryptographyClient> cryptographyClientProvider) {...}

    1 usage
    @Override
    public String decrypt(final byte[] ciphertext) {
        final KeyVaultKey key = keyClient.getKey(vaultKeyName);
        final byte[] plainText = cryptographyClientProvider.apply(key.getKey()) CryptographyClient
                .decrypt(EncryptionAlgorithm.RSA_OAEP_256, ciphertext) DecryptResult
                .getPlainText();
        return new String(plainText, StandardCharsets.UTF_8);
    }

    1 usage
    @Override
    public byte[] encrypt(final String plainText) {
        final KeyVaultKey key = keyClient.getKey(vaultKeyName);
        return cryptographyClientProvider.apply(key.getKey()) CryptographyClient
                .encrypt(EncryptionAlgorithm.RSA_OAEP_256, plainText.getBytes(StandardCharsets.UTF_8)) EncryptResult
                .getCipherText();
    }
}
```

DEMO

The example app: Secret repository implementation

```java
@Component
public class AzureSecretRepositoryImpl implements AzureSecretRepository {

    4 usages
    private final SecretClient secretClient;
    1 usage
    @Value("${secret.name.user}")
    private String vaultSecretNameForDbUserName;
    1 usage
    @Value("${secret.name.pass}")
    private String vaultSecretNameForDbPassword;
    1 usage
    @Value("${secret.name.url}")
    private String vaultSecretNameForDbConnectionUrl;

    @Autowired
    public AzureSecretRepositoryImpl(final SecretClient secretClient) { this.secretClient = secretClient; }

    1 usage
    @Override
    public String getDatabaseConnectionUrl() { return secretClient.getSecret(vaultSecretNameForDbConnectionUrl).getValue(); }

    1 usage
    @Override
    public String getDatabaseUserName() { return secretClient.getSecret(vaultSecretNameForDbUserName).getValue(); }

    1 usage
    @Override
    public String getDatabasePassword() { return secretClient.getSecret(vaultSecretNameForDbPassword).getValue(); }
}
```

DEMO

The example app: Vault client config

```java
@Configuration
public class AzureConfiguration {

    3 usages
    private final TokenCredential tokenCredential;
    3 usages
    private final HttpClient httpClient;
    2 usages
    @Value("${vault.url}")
    private String vaultUrl;

    @Autowired
    public AzureConfiguration(final TokenCredential tokenCredential, final HttpClient httpClient) {...}

    @Bean
    public KeyClient keyClient() {
        return new KeyClientBuilder()
                .vaultUrl(vaultUrl)
                .httpClient(httpClient)
                .serviceVersion(KeyServiceVersion.V7_2)
                .credential(tokenCredential)
                .buildClient();
    }

    @Bean
    public SecretClient secretClient() {
        return new SecretClientBuilder()
                .vaultUrl(vaultUrl)
                .httpClient(httpClient)
                .serviceVersion(SecretServiceVersion.V7_2)
                .credential(tokenCredential)
                .buildClient();
    }
}
```

DEMO

The example app: Crypto client config

```java
/**
 * Function that can provide a {@link CryptographyClient} based on a {@link JsonWebKey}.
 */
@Component
public class CryptographyClientProvider implements Function<JsonWebKey, CryptographyClient> {

    2 usages
    private final HttpClient httpClient;

    2 usages
    private final TokenCredential tokenCredential;

    @Autowired
    public CryptographyClientProvider(final HttpClient httpClient, final TokenCredential tokenCredential) {...}

    @Override
    public CryptographyClient apply(final JsonWebKey jsonWebKey) {
        return new CryptographyClientBuilder()
                .serviceVersion(CryptographyServiceVersion.V7_2)
                .keyIdentifier(jsonWebKey.getId())
                .httpClient(httpClient)
                .credential(tokenCredential)
                .buildClient();
    }
}
```

DEMO

The example app: Azure credentials and HTTP client config

```
14  @Configuration
15  public class AzureAccessConfiguration {
16
        1 usage
17      @Value("${vault.user}")
18      private String keyVaultUserName;
        1 usage
19      @Value("${vault.pass}")
20      private String keyVaultPassword;
21
22      @Bean
23      @ConditionalOnMissingBean(type = "com.azure.core.credential.TokenCredential")
24      public TokenCredential tokenCredential() {
25          return new BasicAuthenticationCredential(keyVaultUserName, keyVaultPassword);
26      }
27
28      @Bean
29      @ConditionalOnMissingBean(type = "com.azure.core.http.HttpClient")
30      public HttpClient httpClient() {
31          return HttpClient.createDefault();
32      }
33
34  }
```

The example app: Test setup

```
@SpringBootTest(classes = {
        AzureAccessTestProcessConfiguration.class,
        AzureAccessTestDockerConfiguration.class,
        AzureAccessTestExternalStartConfiguration.class,
        LowkeyVaultExampleApplication.class},
        properties = {"vault.url=https://localhost:8443", "logging.level.root=WARN"})
class LowkeyVaultExampleApplicationTests {

    1 usage
    private static final int RSA_KEY_SIZE = 2048;
    2 usages
    @Autowired
    private AzureKeyRepository keyRepository;
    3 usages
    @Autowired
    private AzureSecretRepository secretRepository;
    1 usage
    @Autowired
    private KeyClient keyClient;
    3 usages
    @Autowired
    private SecretClient secretClient;
    1 usage
    @Value("${key.name}")
    private String keyName;
    1 usage
    @Value("${secret.name.user}")
    private String userName;
    1 usage
    @Value("${secret.name.pass}")
    private String password;
    1 usage
    @Value("${secret.name.url}")
    private String connectionUrl;
```

The example app: Testing secrets

```
41      @Test
42      void testSecretRepositoryShouldFetchDBCredentialsWhenCalled() {
43          //given
44          final String admin = "admin";
45          final String pass = "s3cret";
46          final String url = "jdbc:h2:mem:test_mem";
47          secretClient.setSecret(userName, admin);
48          secretClient.setSecret(password, pass);
49          secretClient.setSecret(connectionUrl, url);
50
51          //when
52          final String user = secretRepository.getDatabaseUserName();
53          final String password = secretRepository.getDatabasePassword();
54          final String connectionUrl = secretRepository.getDatabaseConnectionUrl();
55
56          //then
57          Assertions.assertEquals(admin, user);
58          Assertions.assertEquals(pass, password);
59          Assertions.assertEquals(url, connectionUrl);
60      }
```

The example app: Testing keys

```
63      @Test
64      void testKeyRepositoryEncryptThenDecryptShouldResultInOriginalTextWhenCalled() {
65          //given
66          final String secret = "a secret message";
67          keyClient.createRsaKey(new CreateRsaKeyOptions(keyName)
68                  .setKeyOperations(KeyOperation.ENCRYPT, KeyOperation.DECRYPT, KeyOperation.WRAP_KEY, KeyOperation.UNWRAP_KEY)
69                  .setKeySize(RSA_KEY_SIZE));
70
71          //when
72          final byte[] encrypted = keyRepository.encrypt(secret);
73          final String decrypted = keyRepository.decrypt(encrypted);
74
75          //then
76          Assertions.assertEquals(secret, decrypted);
77          Assertions.assertNotEquals(encrypted.length, decrypted.getBytes(StandardCharsets.UTF_8).length);
78      }
```

The example app: Test output

```
Starting Gradle Daemon...
Gradle Daemon started in 1 s 204 ms
> Task :clean
> Task :compileJava
> Task :processResources
> Task :classes
> Task :bootJarMainClassName
> Task :bootJar
> Task :jar
> Task :assemble
> Task :checkstyleMain
> Task :compileTestJava
> Task :processTestResources
> Task :testClasses
> Task :checkstyleTest

> Task :test

LowkeyVaultExampleApplicationTests STANDARD_OUT

     INFO c.g.n.l.e.LowkeyVaultExampleApplicationTests       : Starting LowkeyVaultExampleApplicationTests using Java 11.0.12 on EPHUDEBW0016 with PID 14104 (started
      by Istvan_Nagy in C:\Projects\Bench\Github\lowkey-vault-example)
     INFO c.g.n.l.e.LowkeyVaultExampleApplicationTests       : No active profile set, falling back to 1 default profile: "default"
     INFO c.g.n.l.e.LowkeyVaultExampleApplicationTests       : Started LowkeyVaultExampleApplicationTests in 1.825 seconds (JVM running for 3.006)

> Task :check
> Task :build

BUILD SUCCESSFUL in 19s
11 actionable tasks: 11 executed
```

The example app: Lowkey Vault logs

```
: Initializing Spring DispatcherServlet 'dispatcherServlet'
: Initializing Servlet 'dispatcherServlet'
: Completed initialization in 2 ms
: Sending token to client without processing payload: /secrets/user
: Received request to https://localhost:8443 create secret: user using API version: 7.2
: Received request to https://localhost:8443 create secret: pass using API version: 7.2
: Received request to https://localhost:8443 create secret: url using API version: 7.2
: Received request to https://localhost:8443 get secret: user with version: -LATEST- using API version: 7.2
: Received request to https://localhost:8443 get secret: pass with version: -LATEST- using API version: 7.2
: Received request to https://localhost:8443 get secret: url with version: -LATEST- using API version: 7.2
: Sending token to client without processing payload: /keys/rsa-key/create
: Received request to https://localhost:8443 create key: rsa-key using API version: 7.2
: Received request to https://localhost:8443 get key: rsa-key with version: -LATEST- using API version: 7.2
: Received request to https://localhost:8443 get key: rsa-key with version: 38e4b674dd83433cb9ba6c02f9020fd2 using API version: 7.2
: Received request to https://localhost:8443 encrypt using key: rsa-key with version: 38e4b674dd83433cb9ba6c02f9020fd2 using API versi

: Received request to https://localhost:8443 get key: rsa-key with version: -LATEST- using API version: 7.2
: Received request to https://localhost:8443 get key: rsa-key with version: 38e4b674dd83433cb9ba6c02f9020fd2 using API version: 7.2
: Received request to https://localhost:8443 decrypt using key: rsa-key with version: 38e4b674dd83433cb9ba6c02f9020fd2 using API versi
```

References

Lowkey Vault – Project home

https://github.com/nagyesta/lowkey-vault

Lowkey Vault – Example project

https://github.com/nagyesta/lowkey-vault-example

Test Pyramid

https://martinfowler.com/articles/practical-test-pyramid.html

Azure Key Vault

https://azure.microsoft.com/en-us/services/key-vault/

Azure credits for open source projects

https://opensource.microsoft.com/azure-credits

Testcontainers

https://www.testcontainers.org/

Thank you!

For more information, contact:

Istvan Zoltan Nagy

Software Architect

Istvan_Nagy@epam.com