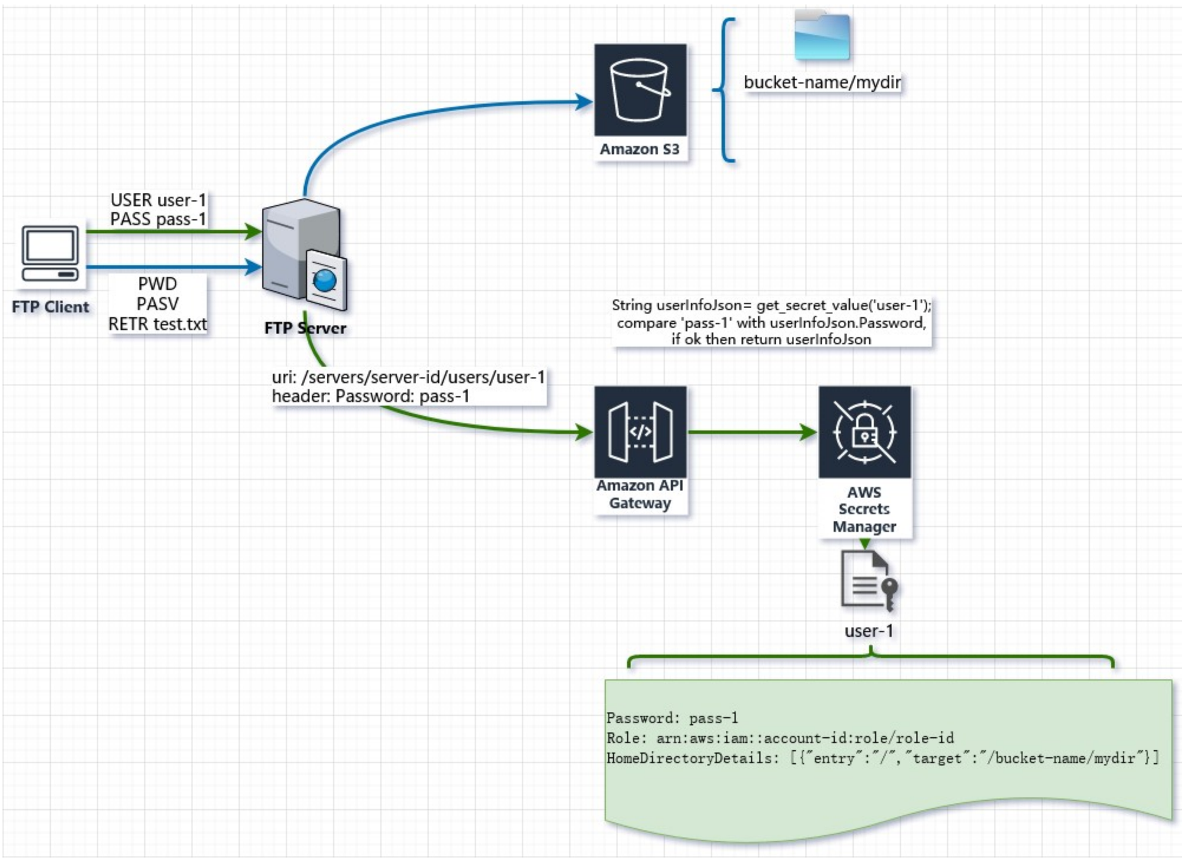
AWS FTP

总体结构


bucket-name/mydir
Amazon S3
USER user-1
PASS pass-1
PWD
PASV
RETR test.txt
FTP Client
FTP Server
String userInfoJson= get_secret_value('user-1');
compare 'pass-1' with userInfoJson.Password,
if ok then return userInfoJson
uri: /servers/server-id/users/user-1
header: Password: pass-1
Amazon API Gateway
AWS Secrets Manager
user-1
Password: pass-1
Role: arn:aws:iam::account-id:role/role-id
HomeDirectoryDetails: [{"entry":"/","target":"/bucket-name/mydir"}]

绿线部分

1. FTP Client 连接 FTP Server，并发送 `USER user-1\r\n PASS pass-1\r\n` 指令，表示使用 user-1/pass-1 用户名/密码登录
2. FTP Server 接收到登录指令后，向 api-gateway 发起用户认证 REST 请求，`uri=/servers/${server-id}/users/user-1 ,header=Password: pass-1`
3. api-gateway 处理用户认证 REST 请求，调用一个 lambda 函数，访问 SecretsManager 使用 user-1 查找对应的密码文件并读取文件内容，然后根据文件内容（在当前应用中是一个 ftp 用户信息的 json），通过比对 pass-1 和 json 中的 Password 字段是否一致，一致则将表示身份验证通过，返回整个用户信息 json
4. FTP Server 根据返回的用户信息 json 来判断是否登录成功，当前用户 role, home 目录等信息

蓝线部分

1. 用户登录成功后，希望下载 home 目录下某个文件，FTP Client 会发送 `PWD\r\nPASV\r\nRETR xxx.txt\r\n` 指令给 FTP Server
2. FTP Server 收到指令后会从对应的 S3 bucket 中下载对应的文件返回给 FTP Client

Api Gateway

> 提供 ftp 用户认证功能

部署详见 https://docs.aws.amazon.com/transfer/latest/userguide/authenticating-users.html

网关模板 https://s3.amazonaws.com/aws-transfer-resources/custom-idp-templates/aws-transfer-custom-idp-secrets-manager-apig.template.yml

模板中的认证逻辑代码：

```python
import os
import json
import boto3
import base64
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    resp_data = {}

    if 'username' not in event or 'serverId' not in event:
        print("Incoming username or serverId missing  - Unexpected")
        return response_data

    input_username = event['username']
    print("Username: {}, ServerId: {}".format(input_username, event['serverId']));

    if 'password' in event:
        input_password = event['password']
    else:
        print("No password, checking for SSH public key")
        input_password = ''

    # 调用 secretsmanager sdk 查询用户信息
    resp = get_secret("SFTP/" + input_username)

    if resp != None:
        resp_dict = json.loads(resp)
    else:
        print("Secrets Manager exception thrown")
        return {}

    if input_password != '':
        if 'Password' in resp_dict:
            # 校验密码是否正确
            resp_password = resp_dict['Password']
        else:
            print("Unable to authenticate user - No field match in Secret for password")
            return {}

        if resp_password != input_password:
            print("Unable to authenticate user - Incoming password does not match stored")
            return {}
    else:
        # 因为不使用 publicKey 认证，所以略过
        if 'PublicKey' in resp_dict:
            resp_data['PublicKeys'] = [resp_dict['PublicKey']]
        else:
            print("Unable to authenticate user - No public keys found")
            return {}
    # 获取用户 role
    if 'Role' in resp_dict:
        resp_data['Role'] = resp_dict['Role']
    else:
        print("No field match for role - Set empty string in response")
        resp_data['Role'] = ''

    # 用不到，可略过
    if 'Policy' in resp_dict:
        resp_data['Policy'] = resp_dict['Policy']
    # 获取用户目录映射
    if 'HomeDirectoryDetails' in resp_dict:
        print("HomeDirectoryDetails found - Applying setting for virtual folders")
        resp_data['HomeDirectoryDetails'] = resp_dict['HomeDirectoryDetails']
        resp_data['HomeDirectoryType'] = "LOGICAL"
    elif 'HomeDirectory' in resp_dict:
        print("HomeDirectory found - Cannot be used with HomeDirectoryDetails")
        resp_data['HomeDirectory'] = resp_dict['HomeDirectory']
    else:
        print("HomeDirectory not found - Defaulting to /")

        print("Completed Response Data: "+json.dumps(resp_data))
        return resp_data

# 访问 secretsmanager
def get_secret(id):
    region = os.environ['SecretsManagerRegion']
    print("Secrets Manager Region: "+region)

    client = boto3.session.Session().client(service_name='secretsmanager', region_name=region)

    try:
        resp = client.get_secret_value(SecretId=id)
        # Decrypts secret using the associated KMS CMK.
        # Depending on whether the secret is a string or binary, one of these fields will be populated.
        if 'SecretString' in resp:
            print("Found Secret String")
            return resp['SecretString']
        else:
            print("Found Binary Secret")
            return base64.b64decode(resp['SecretBinary'])
    except ClientError as err:
        print('Error Talking to SecretsManager: ' + err.response['Error']['Code'] + ', Message: ' + str(err))
        return None
```

FTP Server

> 提供 ftp 协议解析功能

控制台操作

详见 https://aws.amazon.com/cn/blogs/china/new-aws-transfer-for-ftp-and-ftps-in-addition-to-existing-sftp/

代码操作

> 初始化 sdk client

```java
private static TransferClient cli;
private static final String accessKeyId = ""; //需要向运维申请
private static final String secretKeyId = ""; //需要向运维申请

@BeforeClass
public static void init() {
    cli = TransferClient.builder().region(Region.AP_SOUTHEAST_1)
        .credentialsProvider(() -> new AwsCredentials() {
            @Override
            public String accessKeyId() {
                return accessKeyId;
            }

            @Override
            public String secretAccessKey() {
                return secretKeyId;
            }
        }).build();
}
```

> 创建 ftp server

```java
@Test
public void testCreateFtpServer() {

    String apiGateway = ""; //用于身份认证的网关 api url，即上述中部署好的网关 api url
    String invocationRole = ""; // ftp server 调用 apigateway 时的角色 (aws 服务之间调用需要角色)
    Consumer<IdentityProviderDetails.Builder> identityProviderDetails = builder -> builder.url(apiGateway).invocationRole(invocationRole);
    CreateServerResponse resp = cli.createServer(builder -> {
        builder.identityProviderDetails(identityProviderDetails)
            .identityProviderType(IdentityProviderType.API_GATEWAY) //使用 apigateway 作为身份验证
            .protocols(Protocol.FTP) //协议 ftp
            .endpointType(EndpointType.PUBLIC);
    });

    System.out.println(resp); //{   "ServerId": "s-7317991c322a440db"}

}
```

> 启动 ftp server

```
@Test
public void testStartServer() {
    String serverId = ""; //创建成功后返回的 server-id
    StartServerResponse resp = cli.startServer(builder ->
builder.serverId(serverId));

    System.out.println(resp); //{   "ServerId": "s-7317991c322a440db"}
}
```

> 停止 ftp server

```
@Test
public void testStopServer() {
    String serverId = ""; //创建成功后返回的 server-id
    StopServerResponse resp = cli.stopServer(builder ->
builder.serverId(serverId));

    System.out.println(resp); //{   "ServerId": "s-7317991c322a440db"}
}
```

S3

> 提供 ftp 文件存储功能

S3 常用 api

> 初始化 sdk client

```

private static S3Client cli;
private static final String accessKeyId = ""; //需要向运维申请
private static final String secretKeyId = ""; //需要向运维申请

@BeforeClass
public static void init() {
    cli = S3Client.builder().region(Region.AP_SOUTHEAST_1)
        .credentialsProvider(() -> new AwsCredentials() {
            @Override
            public String accessKeyId() {
                return accessKeyId;
            }

            @Override
            public String secretAccessKey() {
                return secretKeyId;
            }
        })
        // 使用 bucket 传输加速功能
        .serviceConfiguration(builder ->
builder.accelerateModeEnabled(true))
        // 超时配置
        .overrideConfiguration(builder -> {
 builder.apiCallTimeout(s3.getTimeout()).apiCallAttemptTimeout(s3.getTimeou
t());
                })
        build();

}
```

对指定 bucket 开启传输加速

```
PutBucketAccelerateConfigurationResponse resp2 =
cli.putBucketAccelerateConfiguration(builder -> {
                builder.bucket("bucket 名
称").accelerateConfiguration(builder1 ->
builder1.status(BucketAccelerateStatus.ENABLED));
            });
```

定义模型

```
@Accessors(chain = true)
@Data
static class S3File {

    private String bucket;

    private String dir;

    private String name;

    private String uploadPath;

    private String downloadPath;

    public String getKey() {
        return dir + "/" + name;
    }
}
```

列出所有 bucket

```java
@Test
public void testListBuckets() {
    ListBucketsResponse resp = cli.listBuckets();
    resp.buckets().forEach(System.out::println);
}
```

> 添加一个 bucket

```java
@Test
public void testAddBucket() {
    CreateBucketResponse resp = cli.createBucket(builder -> builder.bucket("bucket 名称"));

    System.out.println(resp);
    //        CreateBucketResponse(Location=http://ftp-test-java.s3.amazonaws.com/)
}
```

> 往 bucket 添加一个 文件

```java
@Test
public void testPutObject() {
    S3File f = new S3File()
        .setBucket("bucket 名称")
        .setDir("s3 上文件目录")
        .setName("s3 上文件名")
        .setUploadPath("本地文件路径");

    long start = System.currentTimeMillis();

    File file = new File(f.getUploadPath());
    try (FileInputStream in = new FileInputStream(file)) {

        PutObjectResponse resp = cli.putObject(builder -> builder.bucket(f.getBucket()).key(f.getKey()), RequestBody.fromInputStream(in, file.length()));

        System.out.println("time: " + (System.currentTimeMillis() - start) / 1000);
        System.out.println(resp);
    } catch (Exception e) {
        e.printStackTrace();
    }
    //        PutObjectResponse(ETag="de450e053077183149e3f2e7ad998e0e")

}
```

> 从 bucket 下载一个文件

```java
@Test
```

```java
public void testGetObject() {
    S3File f = new S3File()
        .setBucket("bucket 名称")
        .setDir("s3 上文件目录")
        .setName("s3 上文件名")
        .setDownloadPath("下载后的本地保存位置");
    try (FileOutputStream out = new FileOutputStream(f.getDownloadPath()))
{
        GetObjectResponse resp = cli.getObject(
            builder -> builder.bucket(f.getBucket()).key(f.getKey()),
ResponseTransformer.toOutputStream(out));

        System.out.println(resp);
    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

SecretsManager

> 提供 ftp 用户管理和存储功能

代码操作

> 初始化 cli

```java
private static SecretsManagerClient cli;
private static final String accessKeyId = ""; //需要向运维申请
private static final String secretKeyId = ""; //需要向运维申请

private static ObjectMapper om;

@BeforeClass
public static void init() {
    om = new ObjectMapper();
    cli = SecretsManagerClient.builder()
        .credentialsProvider(() -> new AwsCredentials() {
            @Override
            public String accessKeyId() {
                return accessKeyId;
            }

            @Override
            public String secretAccessKey() {
                return secretKeyId;
            }
        })
        .region(Region.AP_SOUTHEAST_1)
        .build();
}
```

> 定义模型

```java
@ToString
@Data
@Accessors(chain = true)
static class FtpUserInfo {

    @JsonIgnore
    private String username;

    @JsonProperty("Password")
    private String password;

    @JsonProperty("Role")
    private String role; //允许访问 s3 对应 bucket 的一个角色

    @JsonProperty("HomeDirectoryDetails")
    private String homeDirectoryDetails; //DirMapping json

}

@ToString
@Data
@AllArgsConstructor
@NoArgsConstructor
static class DirMapping {

    @JsonProperty("Entry")
    private String entry; //ftp 目录

    @JsonProperty("Target")
    private String target; //s3 目录

}
```

> 添加一个 secret (即 ftp 的一个用户)

```java
@Test
public void testAddSecret() throws JsonProcessingException {

    //允许访问 s3 对应 bucket 的一个角色
    String role = ""; //向运维申请

    // ftp 用户 homedir 与 s3 路径的映射
    ArrayList<DirMapping> dirMapping = new ArrayList<>(1);
    dirMapping.add(new DirMapping("/", "/ftp-test-s3333/test-dir"));

    FtpUserInfo user = new FtpUserInfo()
        .setUsername("test-java")
        .setPassword("123456")
        .setRole(role)
        .setHomeDirectoryDetails(om.writeValueAsString(dirMapping));
    String secretJson = om.writeValueAsString(user);

    CreateSecretResponse resp = cli.createSecret(builder ->builder.name("SFTP/" + user.getUsername()).secretString(secretJson));

    System.out.println(resp);
    //        CreateSecretResponse(ARN=arn:aws:secretsmanager:ap-southeast-1:066742168474:secret:SFTP/test-java-oEE2I9, Name=SFTP/test-java, VersionId=e391e945-7e0f-4f8d-9626-3ea50fea1914)
}
```

> 删除一个 secret (即 ftp 一个用户)

```java
@Test
public void testDeleteSecret() {
    String username = "test-java";
    DeleteSecretResponse resp = Optional.ofNullable(cli.deleteSecret((builder -> builder.secretId("SFTP/" + username).forceDeleteWithoutRecovery(true))))
        .orElseThrow(() -> new RuntimeException("delete secret error, result is null."));

    System.out.println(resp);
    //        DeleteSecretResponse(ARN=arn:aws:secretsmanager:ap-southeast-1:066742168474:secret:test-name-PwW7Qs, Name=test-name, DeletionDate=2020-05-14T05:18:47.634Z)
}
```