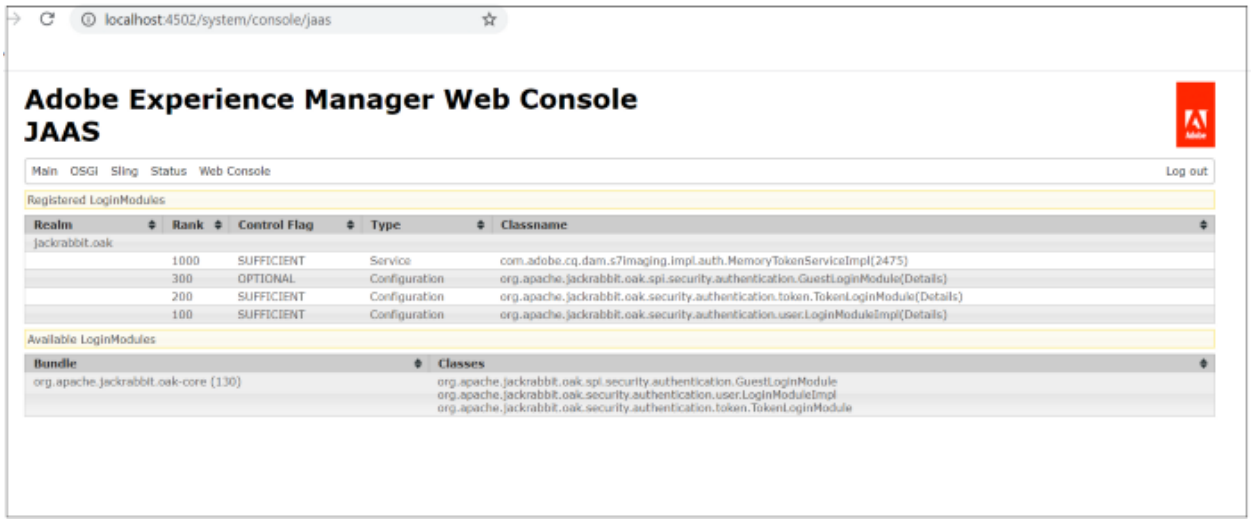
OAuth 2.0 Setup using the grant type JWT Bearer Token

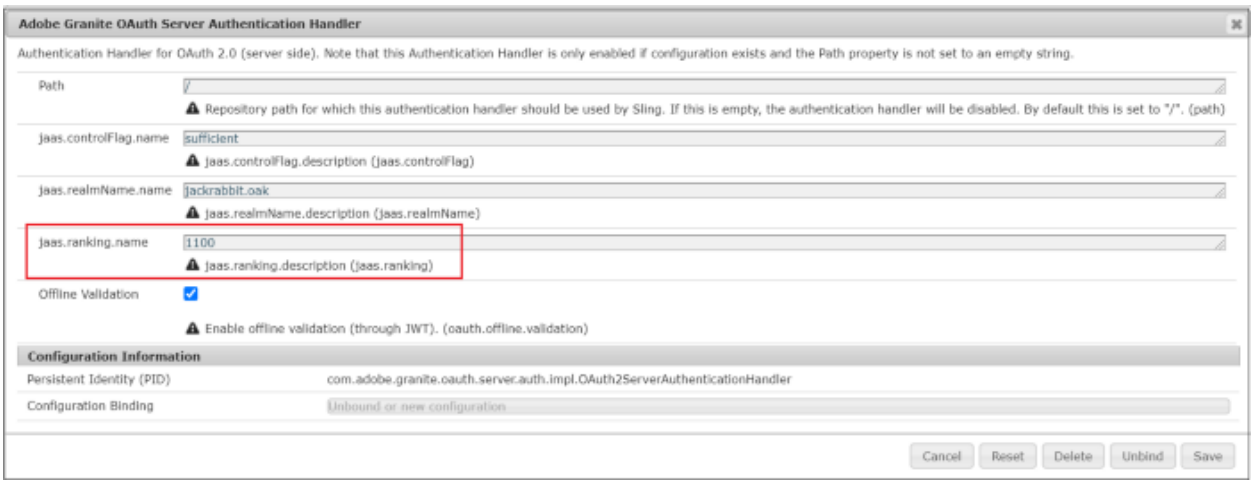
1) Enable OAuth Server Authentication Handler

As a first step enable the OAuth Server Authentication Handler, by default AEM won't enable the OAuth Server Authentication handler.

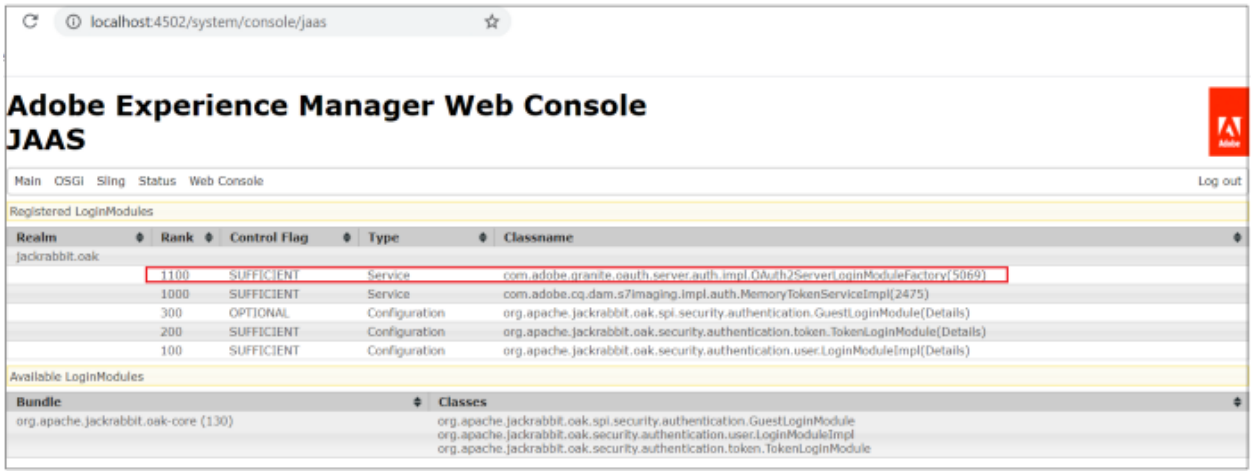
System Console →Main →JASS



To enable the OAuth Server Authentication Handler, change the "jaas.ranking.name" value to 1100 in "Adobe Granite OAuth Server Authentication Handler" and save the configuration — http://localhost:4502/system/console/configMgr/com.adobe.granite.oauth.server.auth.impl.OAuth2ServerAuthenticationHandler



OAuth Server Authentication Handler is enabled now.



Path: /apps/system/config/com.adobe.granite.oauth.server.auth.impl.OAuth2ServerAuthenticationHandler.config

2) Install custom scope code on the environment as shown below

```
import javax.servlet.http.HttpServletRequest;

import org.apache.jackrabbit.api.security.user.User;
import org.osgi.service.component.annotations.Component;

import com.adobe.granite.oauth.server.Scope;
import com.adobe.granite.oauth.server.ScopeWithPrivileges;

@Component (service = Scope.class)
public class ContentReadScope implements ScopeWithPrivileges {

        private static final String CONTENT_RESOURCE_URI = "/content";
        private static final String CONTENT_RESOURCE_READ_SCOPE_NAME = "content_read";

        public ContentReadScope() {

        }

        @Override
        public String getDescription(HttpServletRequest arg0) {
                return "Read Content";
        }

        @Override
        public String getEndpoint() {
                return null;
        }
```

```
        @Override
        public String getName() {
                return CONTENT_RESOURCE_READ_SCOPE_NAME;
        }

        @Override
        public String getResourcePath(User user) {
                return CONTENT_RESOURCE_URI;
        }

        @Override
        public String[] getPrivileges() {
                return new String[] { "jcr:read" };
        }
}
```

3) Create Oauth Client

Client Name

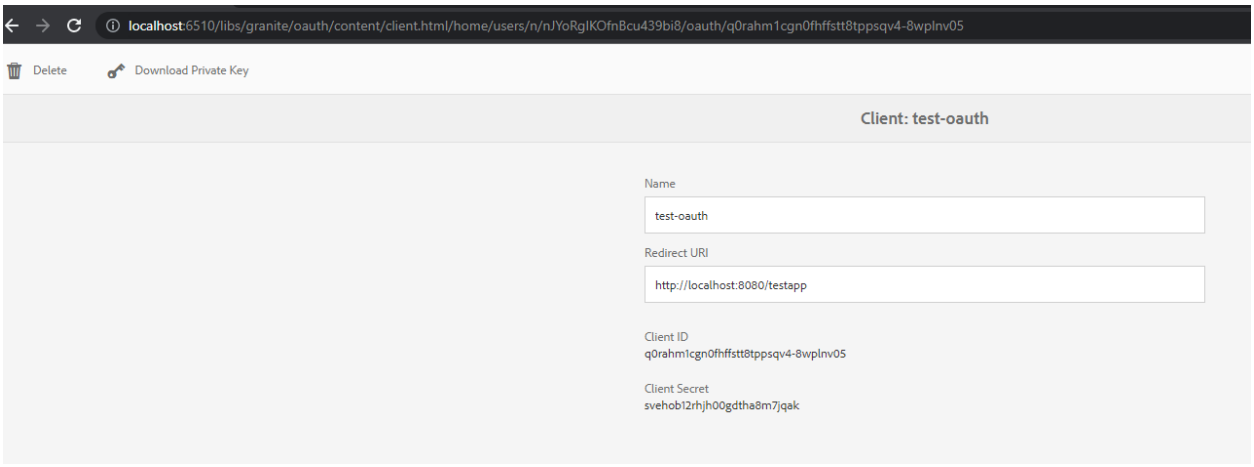
test-oauth

Client ID

q0rahm1cgn0fhffstt8tppsqv4-8wplnv05

Client Secret

svehob12rhjh00gdtha8m7jqak

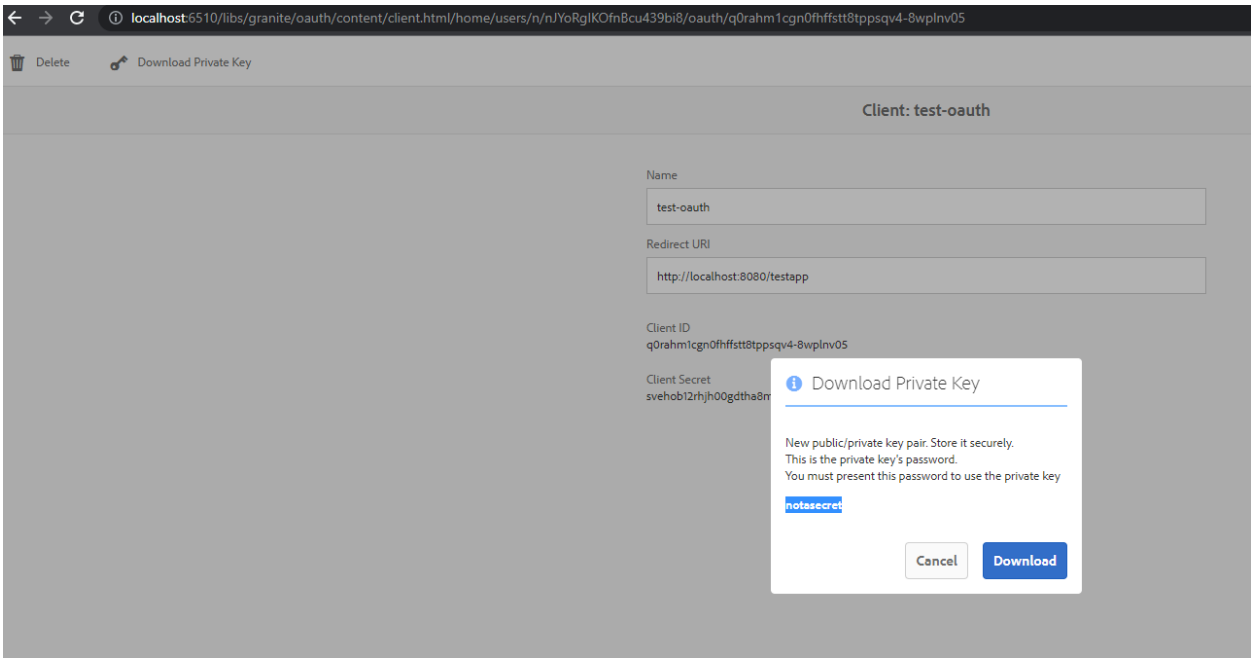


Path:

a)/home/users/n/nJYoRgIKOfnBcu439bi8/oauth

b)/home/users/oauth

4) Download the AEM OAuth client private key to sign the JWT bearer token also note the private key password (keep the private key and password safe)



5) Generate the public certificate from the downloaded private key, execute the below command — Enter the private key password.

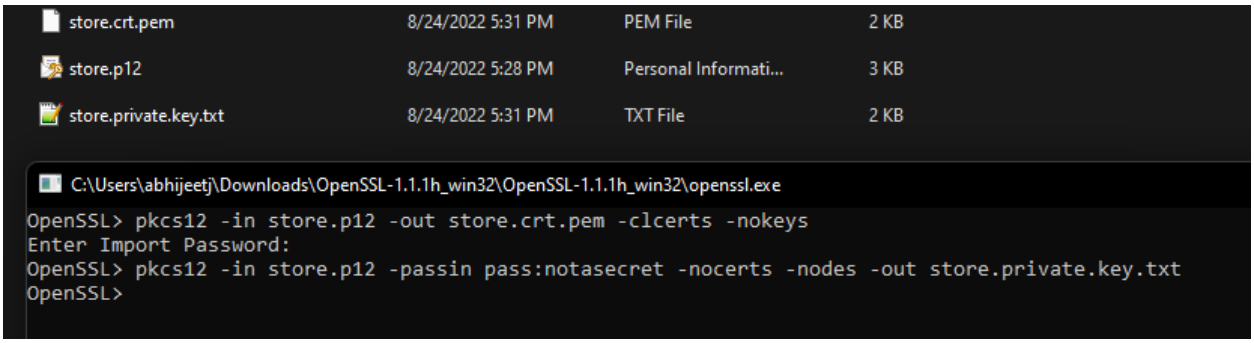
Command: openssl pkcs12 -in store.p12 -out store.crt.pem -clcerts -nokeys

NOTE: This command will generate store.crt.pem file.

6) Extract the private key

Command: openssl pkcs12 -in store.p12 -passin pass:notasecret -nocerts -nodes -out store.private.key.txt

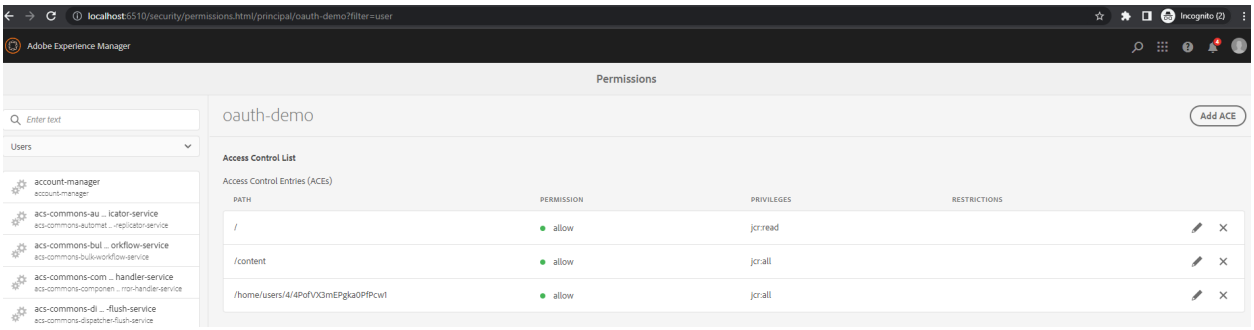
NOTE: This command will generate the store.private.key.txt file.



7) Create a demo user say oauth-demo and assign the following permissions.

a) Read permission at root path ie. / path

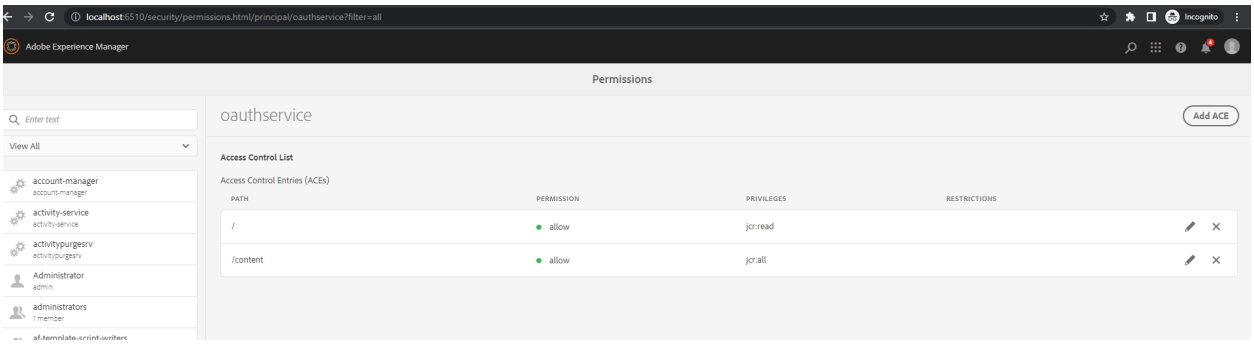
b) All permissions at /content path



8) Assign oauthservice user following permissions.

a) Read permission at root path ie. / path

b) All permissions at /content path



9) Let us now create a JWT token signed with the private key generated in the previous step.

You can use any JWT libraries to generate the JWT token for real scenario's but I am using https://jwt.io/ for demo.

Header

```
{
"alg": "RS256",
"typ": "JWT"
}
```

Payload

```
{
"aud": "<Token Endpoint>",
"iss": "<Client Id>",
"sub": "<user name>",
"exp": <Current time in Milliseconds+expiry>,
"iat": <Current time in Milliseconds>,
"scope": "<scope>",
"cty": "code"
}
```

Eg:

{

"aud": "http://localhost:6510/oauth/token",

"iss": "q0rahm1cgn0fhffstt8tppsqv4-8wplnv05",

"sub": "oauth-demo",

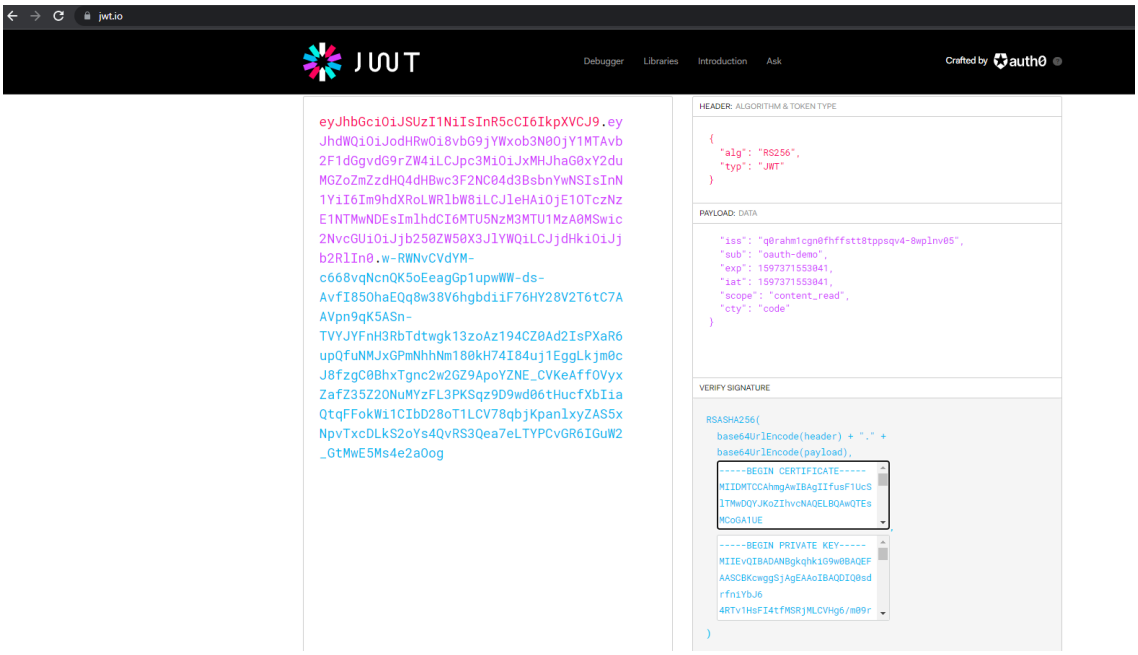
"exp": 1597371553041,

"iat": 1597371553041,

"scope": "content_read",

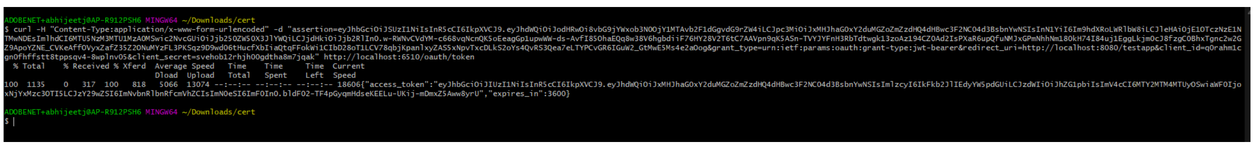
"cty": "code"

}



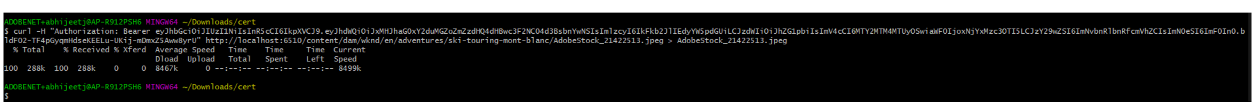
10) Now the JWT token is ready, the token can be used to retrieve the access token from AEM.

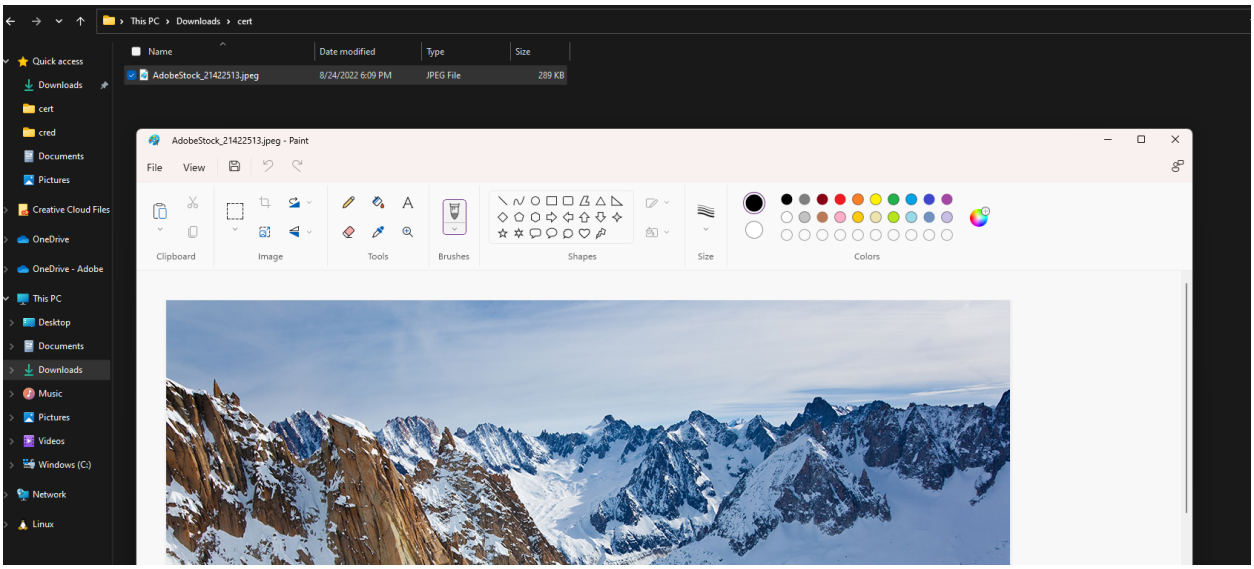
Command: curl -H "Content-Type:application/x-www-form-urlencoded" -d "assertion=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwOi8vbG9jYWxob3N0OjY1MTAvb2F1dGgvdG9rZW4iLCJpc3MiOiJxMHJhaG0xY2duMGZoZmZzdHQ4dHBwc3F2NC04d3BsbnYwNSIsInN1YiI6Im9hdXRoLWRlbW8iLCJleHAiOjE1OTczNzE1NTMwNDEsImlhdCI6MTU5NzM3MTU1MzA0MSwic2NvcGUiOiJjb250ZW50X3JlYWQiLCJjdHkiOiJjb2RlIn0.w-RWNvCVdYM-c668vqNcnQK5oEeagGp1upwWW-ds-AvfI85OhaEQq8w38V6hgbdiiF76HY28V2T6tC7AAVpn9qK5ASn-TVYJYFnH3RbTdtwgk13zoAz194CZ0Ad2IsPXaR6upQfuNMJxGPmNhhNm180kH74I84uj1EggLkjm0cJ8fzgC0BhxTgnc2w2GZ9ApoYZNE_CVKeAffOVyxZafZ35Z2ONuMYzFL3PKSqz9D9wd06tHucfXbIiaQtqFFokWi1CIbD28oT1LCV78qbjKpanlxyZAS5xNpvTxcDLkS2oYs4QvRS3Qea7eLTYPCvGR6IGuW2_GtMwE5Ms4e2aOog&grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&redirect_uri=http://localhost:8080/testapp&client_id=q0rahm1cgn0fhffstt8tppsqv4-8wplnv05&client_secret=svehob12rhjh00gdtha8m7jqak" http://localhost:6510/oauth/token



11) Now you should be able to download the assets from /content/dam through the access token (based on the scope used while generating the JWT token)

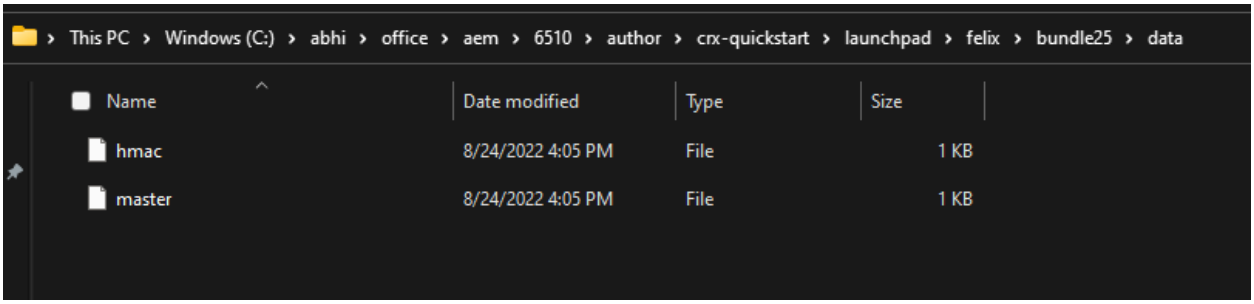
curl -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJxMHJhaG0xY2duMGZoZmZzdHQ4dHBwc3F2NC04d3BsbnYwNSIsImlzcyI6IkFkb2JlIEdyYW5pdGUiLCJzdWIiOiJhZG1pbiIsImV4cCI6MTY2MTM4MTUyOSwiaWF0IjoxNjYxMzc3OTI5LCJzY29wZSI6ImNvbnRlbnRfcmVhZCIsImN0eSI6ImF0In0.bldF02-TF4pGyqmHdseKEELu-UKij-mDmxZ5Aww8yrU" http://localhost:6510/content/dam/wknd/en/adventures/ski-touring-mont-blanc/AdobeStock_21422513.jpeg > AdobeStock_21422513.jpeg



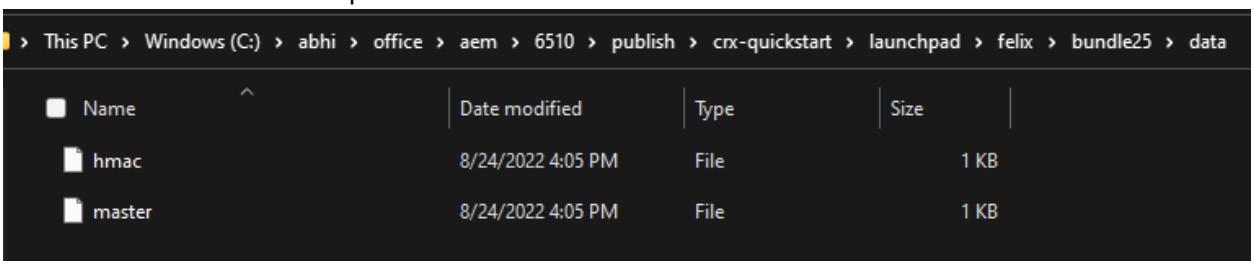


To perform the same setup on publish server you need to perform following

12) Sync the hmac keys across environments as suggested here
https://www.nextrow.com/blog/adobe-experience-manager/crypto-support-in-aem-part-2
Eg: Look for com.adobe.granite.crypto.file bundle on local server as shown in below screenshot
Master and HMAC files from author



Master and HMAC files from publish



Restart the publish server after syncing the master and hmac keys.

13) Install the custom scope code on publish server as mentioned in above Step 2.
14) Replicate the configuration and OAuth client path created in Step 1 and Step 3.
15) Replicate the OAuth user created in Step 7.
16) Assign the permission for the OAuth user and oauthservice user as mentioned in Step 7 and Step8.
17) Let us now create a JWT token signed with the private key generated in Step 9 but with publish environment details.

Header

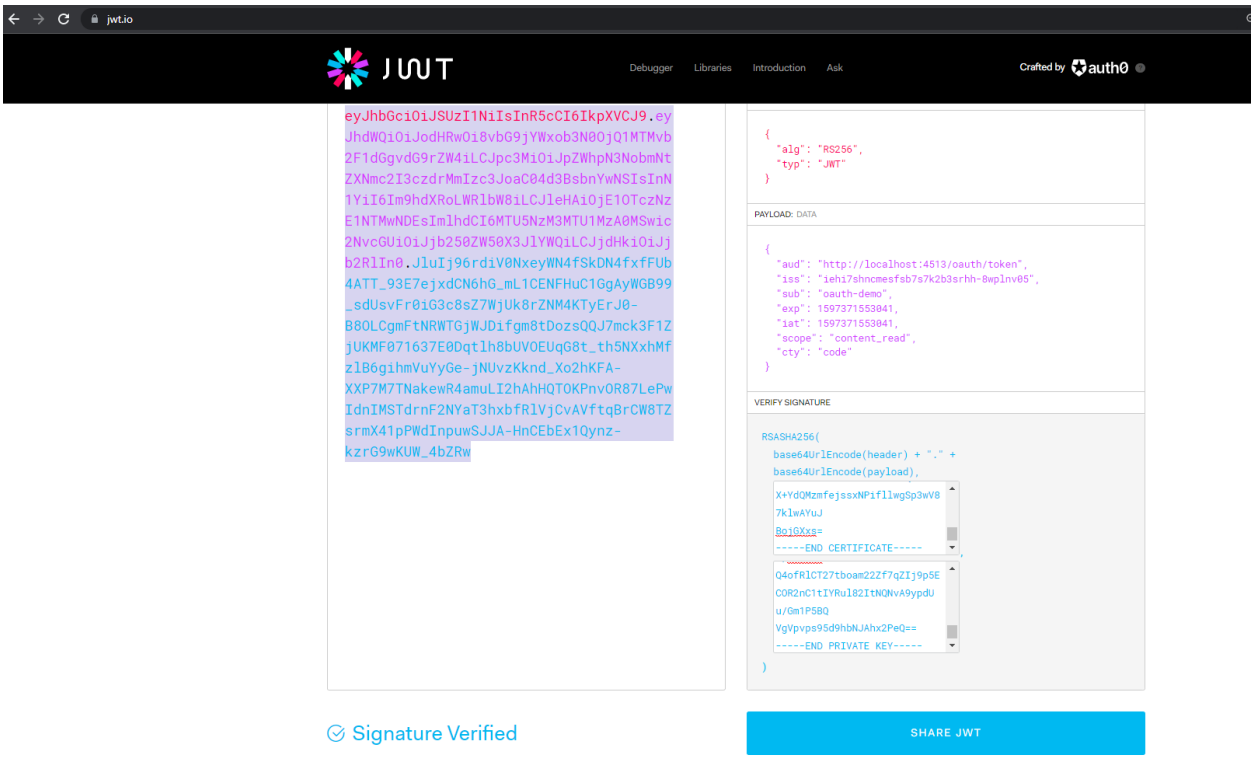
```
{
"alg": "RS256",
"typ": "JWT"
}
```

Payload

```
{
"aud": "<Token Endpoint>",
"iss": "<Client Id>",
"sub": "<user name>",
"exp": <Current time in Milliseconds+expiry>,
"iat": <Current time in Milliseconds>,
"scope": "<scope>",
"cty": "code"
}
```

Eg:

```
{
  "aud": "http://localhost:4513/oauth/token",
  "iss": "q0rahm1cgn0fhffstt8tppsqv4-8wplnv05",
  "sub": "oauth-demo",
  "exp": 1597371553041,
  "iat": 1597371553041,
  "scope": "content_read",
  "cty": "code"
}
```



18) Now the JWT token is ready, the token can be used to retrieve the access token from AEM.

curl -H "Content-Type:application/x-www-form-urlencoded" -d "assertion=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwOi8vbG9jYWxob3N0OjQ1MTMvb2F1dGggvdG9rZW4iLCJpc3MiOiJpZWhpN3NobmNtZXNmc2I3czdrMmIzc3JoaC04d3BsbnYwNSIsInN1YiI6Im9hdXRoLWRlbW8iLCJleHAiOjE1OTczNzE1NTMwNDEsImlhdCI6MTU5NzM3MTU1MzA0MSwic2NvcGUiOiJjb250ZW50X3JlYWQiLCJjdHkiOiJjb2RlIn0.JluIj96rdiV0NxeyWN4fSkDN4fxfFUb4ATT_93E7ejxdCN6hG_mL1CENFHuC1GgAyWGB99_sdUsvFr0iG3c8sZ7WjUk8rZNM4KTyErJ0-B8OLCgmFtNRWTGjWJDifgm8tDozsQQJ7mck3F1ZjUKMF071637E0Dqtlh8bUVOEUqG8t_th5NXxhMfzlB6gihmVuYyGe-jNUvzKknd_Xo2hKFA-XXP7M7TNakewR4amuLI2hAhHQTOKPnvOR87LePwIdnIMSTdrnF2NYaT3hxbfRlVjCvAVftqBrCW8TZsrmX41pPWdInpuwSJJA-HnCEbEx1Qynz-kzrG9wKUW_4bZRw&grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&redirect_uri=http://localhost:8080/testapp&client_id=iehi7shncmesfsb7s7k2b3srhh-8wplnv05&client_secret=h1n2v0f4mcde4ijk5cpppn4g6r" http://localhost:4513/oauth/token

19) Now you should be able to download the assets from /content/dam through the access token (based on the scope used while generating the JWT token)

curl -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJpZWhpN3NobmNtZXNmc2I3czdrMmIzc3JoaC04d3BsbnYwNSIsImlzcyI6IkFkb2JlIEdyYW5pdGUiLCJzdWIiOiJhZG1pbiIsImV4cCI6MTY2MTUyNTI2NywiaWF0IjoxNjYxNTIxNjY3LCJzY29wZSI6ImNvbnRlbnRfcmVhZCIsImN0eSI6ImF0In0.LJwGtHM-FloFI4ps7sEi1m-GPtD2CMyvm3odHdOxSfA" http://localhost:4513/content/dam/wknd/en/adventures/ski-touring-mont-blanc/AdobeStock_21422513.jpeg > AdobeStock_publish_21422513.jpeg