Ganpat University | Institute of Computer Technology
|| विद्यया समाजोत्कर्षः ||

Name: Tushar Panchal

En.No: 21162101014

Sub: CS(Cloud Security)

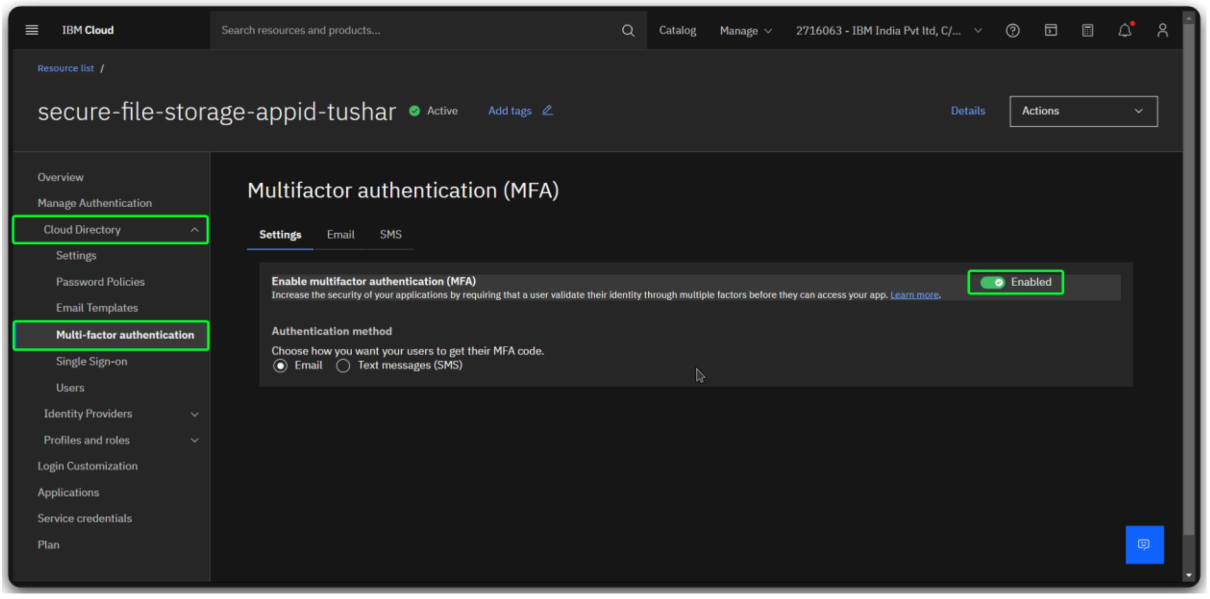
Branch: CBA

Batch:71

---------------------------PRACTICAL 12----------------------------

You are tasked with setting up a secure authentication system for an enterprise application hosted on IBM Cloud. The application should enforce MFA for added security and integrate with a custom identity provider to accommodate users from a legacy system.

1. Enable MFA from Cloud directory settings of App ID instance



secure-file-storage-appid-tushar Active Add tags

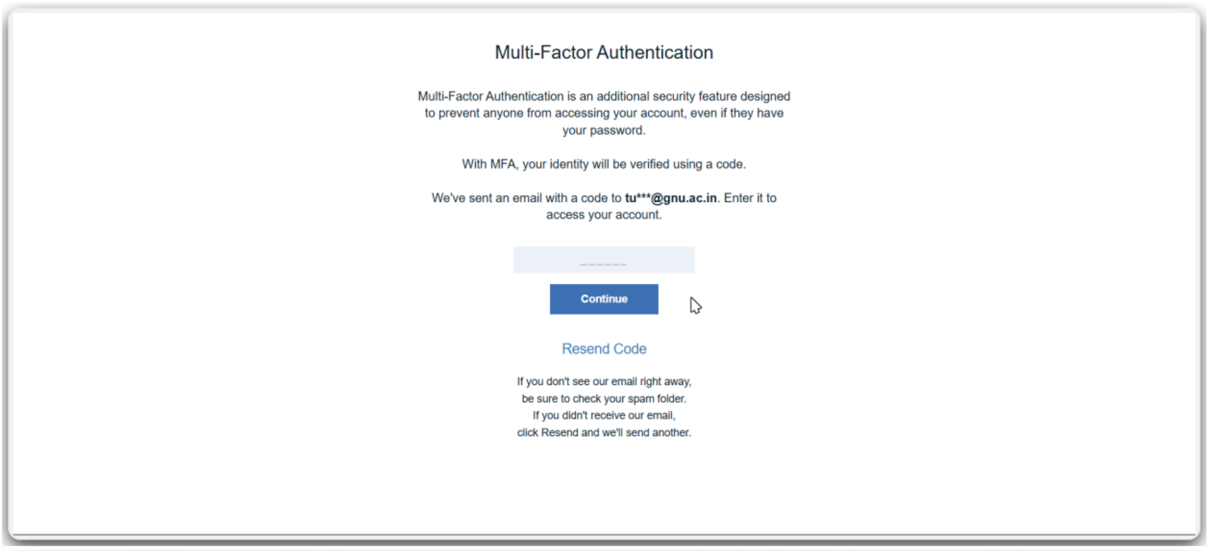
Multifactor authentication (MFA)

Enable multifactor authentication (MFA) — Enabled
Increase the security of your applications by requiring that a user validate their identity through multiple factors before they can access your app. Learn more.

Authentication method
Choose how you want your users to get their MFA code.
[✓ Email] Text messages (SMS)

2. Check if MFA works

Now while trying to login you get the one time password on the mail



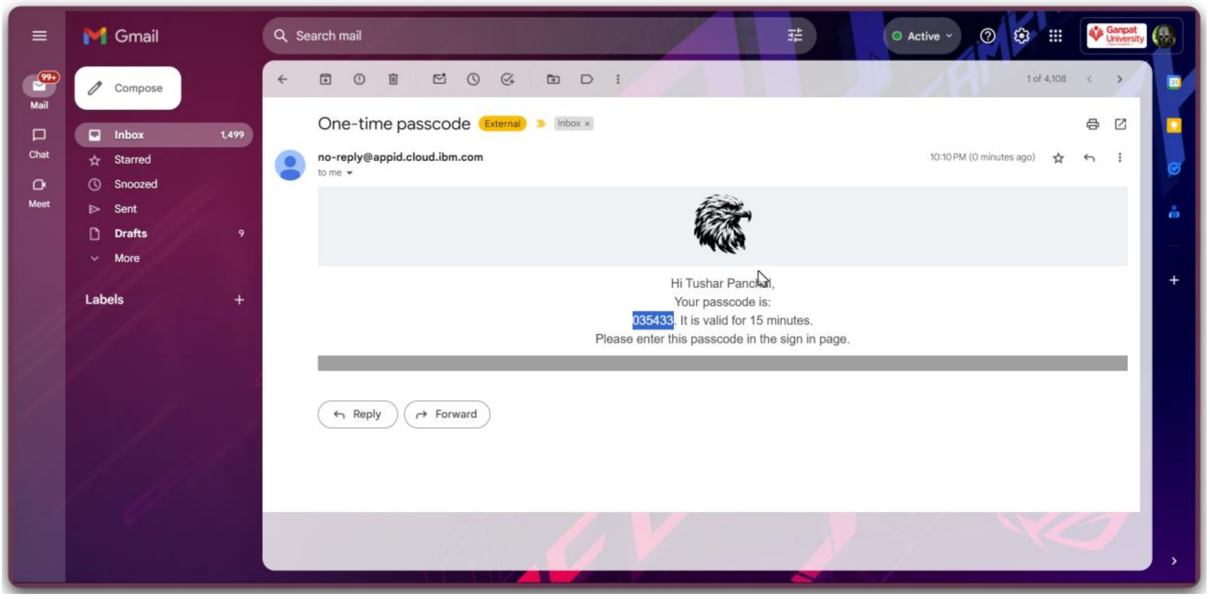
Multi-Factor Authentication

Multi-Factor Authentication is an additional security feature designed to prevent anyone from accessing your account, even if they have your password.

With MFA, your identity will be verified using a code.

We've sent an email with a code to **tu***@gnu.ac.in**. Enter it to access your account.

If you don't see our email right away, be sure to check your spam folder. If you didn't receive our email, click Resend and we'll send another.



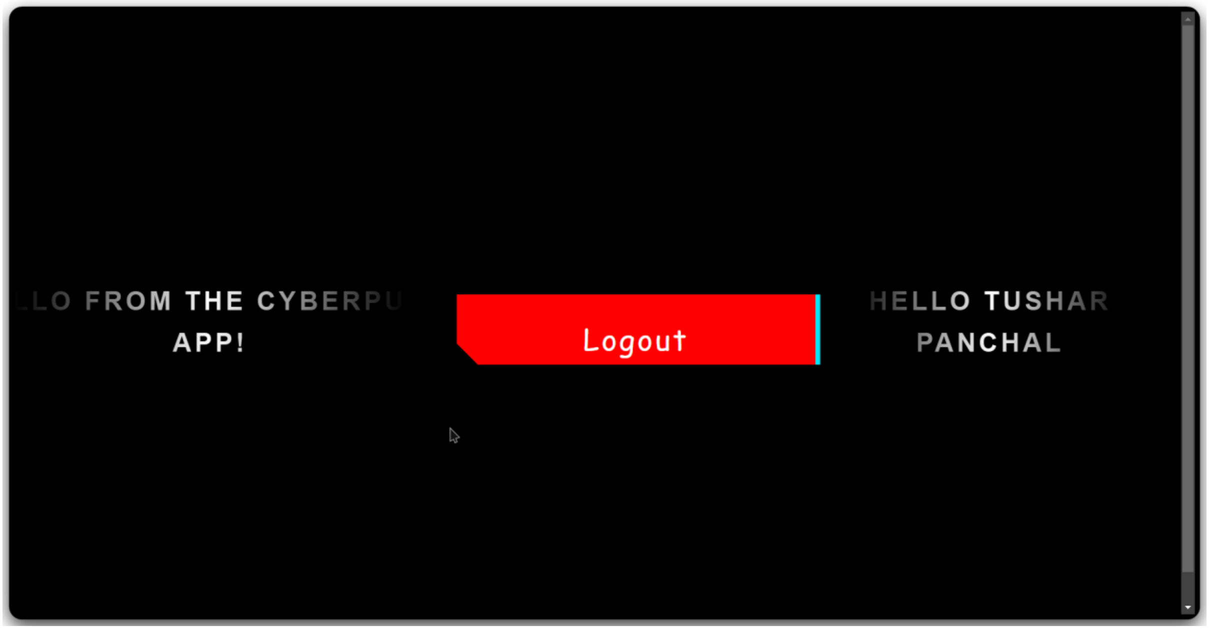
One-time passcode

no-reply@appid.cloud.ibm.com
to me

10:10 PM (0 minutes ago)

Hi Tushar Panchal,
Your passcode is:
035433. It is valid for 15 minutes.
Please enter this passcode in the sign in page.

As you can see above we get our passcode for MFA so we can log in with that PASSCODE from mail to our appid MFA

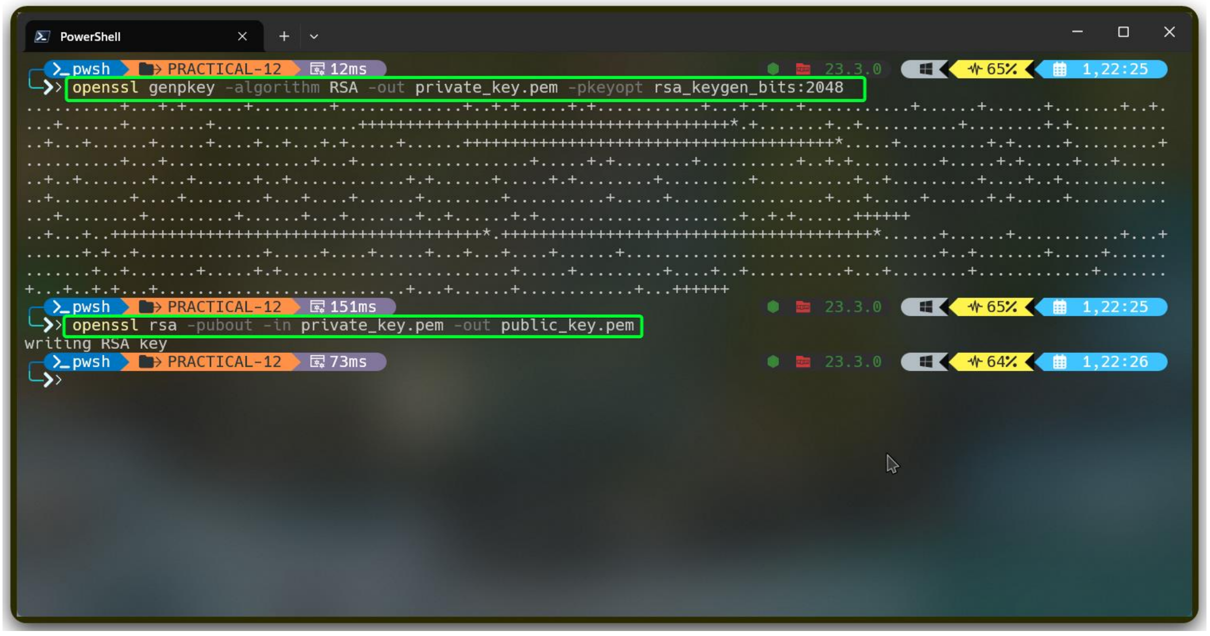


As you can see above i logged in successfully.

Custom Identity:

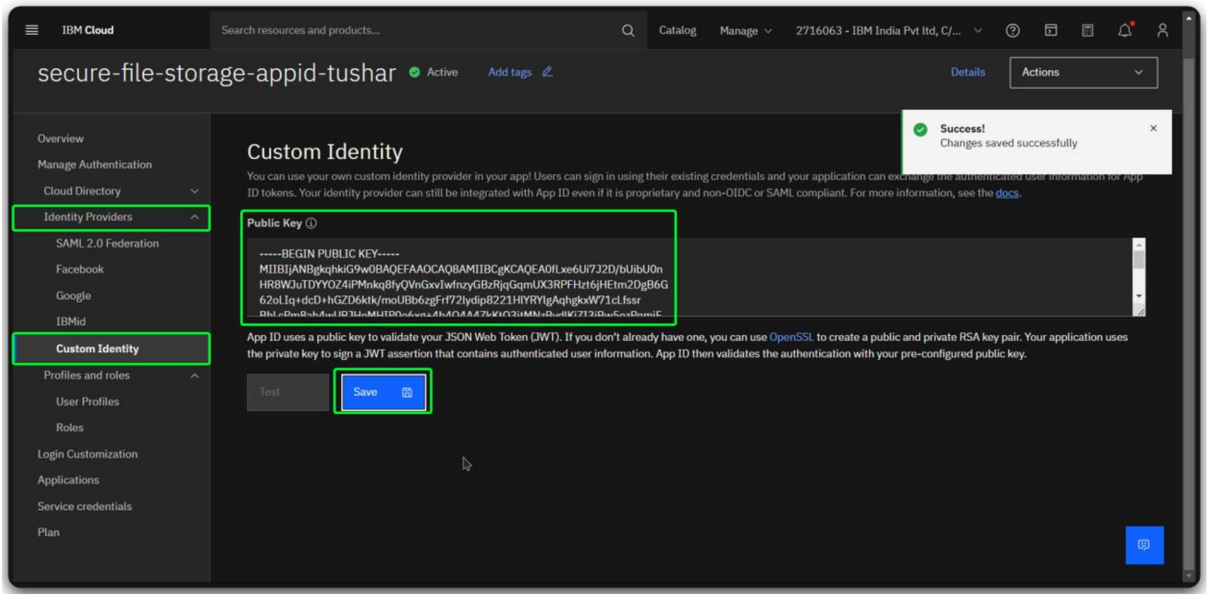
Run this below commands to generate private and public keys

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
openssl rsa -pubout -in private_key.pem -out public_key.pem
```



```
>_pwsh  PRACTICAL-12  12ms                                   23.3.0   65%   1,22:25
>> openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
.........+...+.+......+........+.............+..+.+.....+.+..........+...+.+...+..........+.....+.....+.......+..+.
...+.....+.......+.............++++++++++++++++++++++++++++++++++++++++*.+.......+..+.........+.......+.+.........
..+...+.....+....+....+...+..+.+.....+......+++++++++++++++++++++++++++++++++++++++++*.....+.........+.+.....+........+
.........+..+.................+...+...................+.....+.+........+..........+..+.+.........+.....+.+.....+...+....
..+..+.......+...+......+...+.........+.+......+.....+.+.........+.........+.........+..+.........+....+..+..........
..+.......+....+........+...+...+.....+........+..........+.....+..............+...+....+......+.+.....+.........
...+........+.........+......+...+......+..+.....+.+...................+..+.+......++++++
..+...+..+++++++++++++++++++++++++++++++++++++++*.+++++++++++++++++++++++++++++++++++++++*.....+......+..........+...+
......+.+..+...........+.....+....+.....+...+...+....+......+.....................+..+.......+......+.......
......+..+......+.....+.+....................+.....+........+....+..+.........+...+.......+.........+......
+...+..+.+...+..........................+...+......+............+...++++++
>_pwsh  PRACTICAL-12  151ms                                  23.3.0   65%   1,22:25
>> openssl rsa -pubout -in private_key.pem -out public_key.pem
writing RSA key
>_pwsh  PRACTICAL-12  73ms                                   23.3.0   64%   1,22:26
>>
```

Now go to custom identity section of your AppID service and paste the public id that we have created above



secure-file-storage-appid-tushar Active Add tags

Success!
Changes saved successfully

Custom Identity

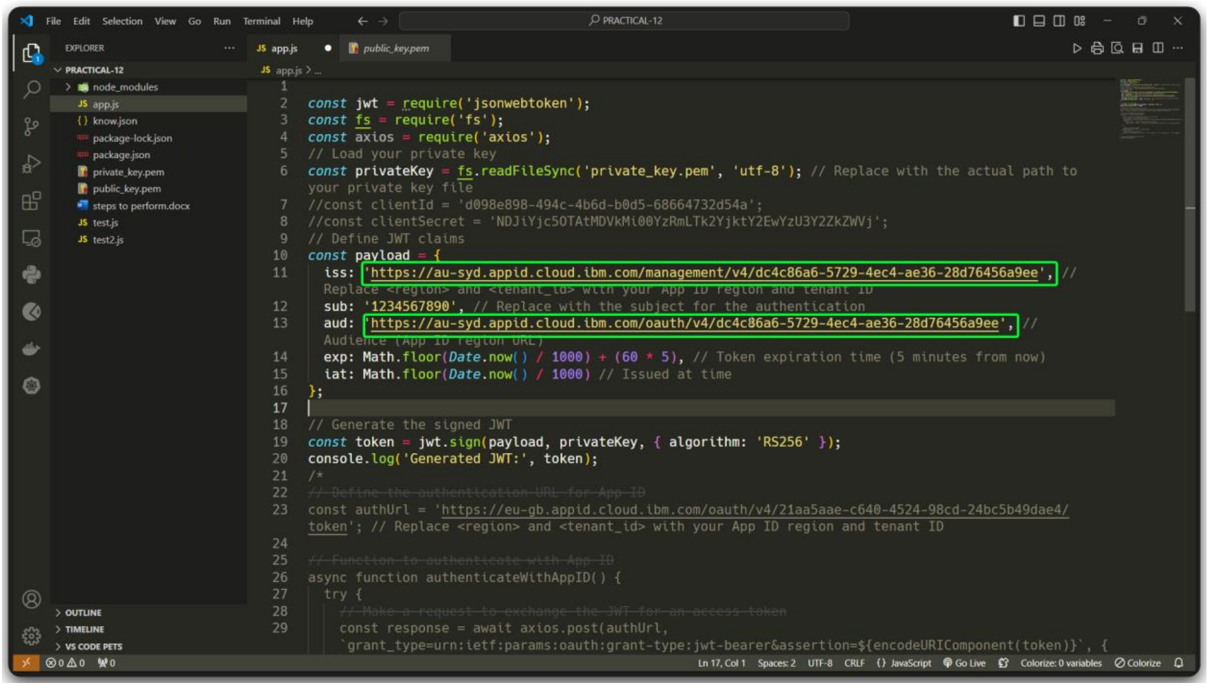
You can use your own custom identity provider in your app! Users can sign in using their existing credentials and your application can exchange the authenticated user information for App ID tokens. Your identity provider can still be integrated with App ID even if it is proprietary and non-OIDC or SAML compliant. For more information, see the docs.

Public Key

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0fLxe6Ui7J2D/bUibU0n
HR8WJuTDYYOZ4iPMnkq8fyQVnGxvIwfnzyGBzRjqGqmUX3RPFHzt6jHEtm2DgB6G
62oLIq+dcD+hGZD6ktk/moUBb6zgFrf72lydip8221HIYRYlgAqhgkxW71cLfssr
```

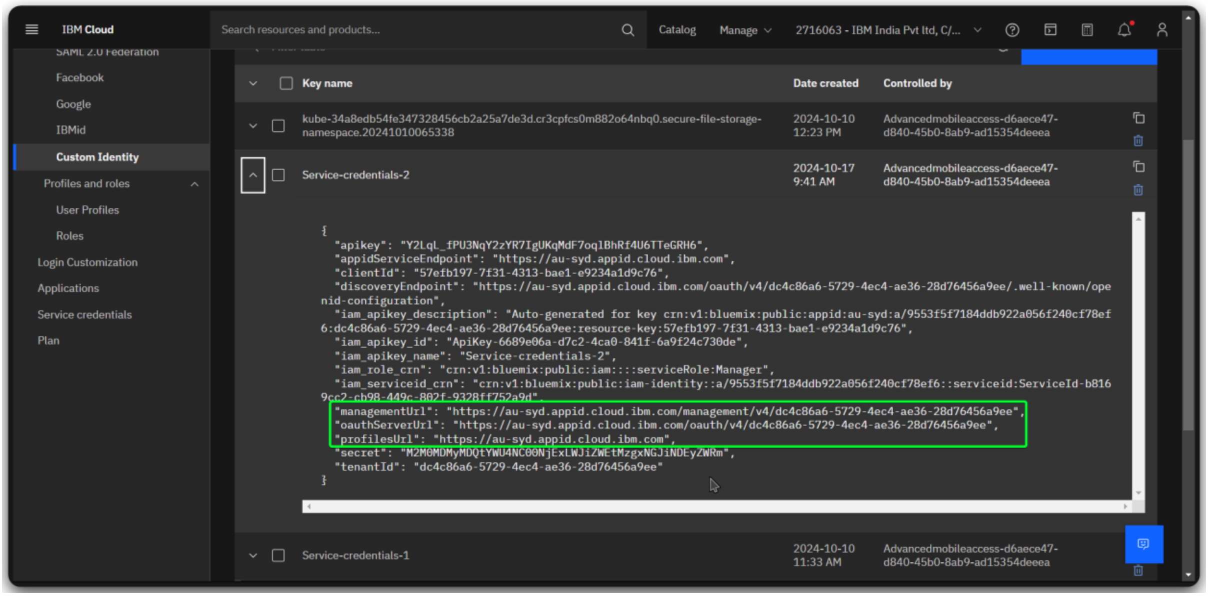
App ID uses a public key to validate your JSON Web Token (JWT). If you don't already have one, you can use OpenSSL to create a public and private RSA key pair. Your application uses the private key to sign a JWT assertion that contains authenticated user information. App ID then validates the authentication with your pre-configured public key.

Make changes in app.js code as your service credentials



```javascript

const jwt = require('jsonwebtoken');
const fs = require('fs');
const axios = require('axios');
// Load your private key
const privateKey = fs.readFileSync('private_key.pem', 'utf-8'); // Replace with the actual path to your private key file
//const clientId = 'd098e898-494c-4b6d-b0d5-68664732d54a';
//const clientSecret = 'NDJiYjc5OTAtMDVkMi00YzRmLTk2YjktY2EwYzU3Y2ZkZWVj';
// Define JWT claims
const payload = {
  iss: 'https://au-syd.appid.cloud.ibm.com/management/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee', // Replace <region> and <tenant_id> with your App ID region and tenant ID
  sub: '1234567890', // Replace with the subject for the authentication
  aud: 'https://au-syd.appid.cloud.ibm.com/oauth/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee', // Audience (App ID region URL)
  exp: Math.floor(Date.now() / 1000) + (60 * 5), // Token expiration time (5 minutes from now)
  iat: Math.floor(Date.now() / 1000) // Issued at time
};

// Generate the signed JWT
const token = jwt.sign(payload, privateKey, { algorithm: 'RS256' });
console.log('Generated JWT:', token);
/*
~~// Define the authentication URL for App ID~~
const authUrl = 'https://eu-gb.appid.cloud.ibm.com/oauth/v4/21aa5aae-c640-4524-98cd-24bc5b49dae4/token'; // Replace <region> and <tenant_id> with your App ID region and tenant ID

~~// Function to authenticate with App ID~~
async function authenticateWithAppID() {
  try {
    ~~// Make a request to exchange the JWT for an access token~~
    const response = await axios.post(authUrl, `grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion=${encodeURIComponent(token)}`, {
```

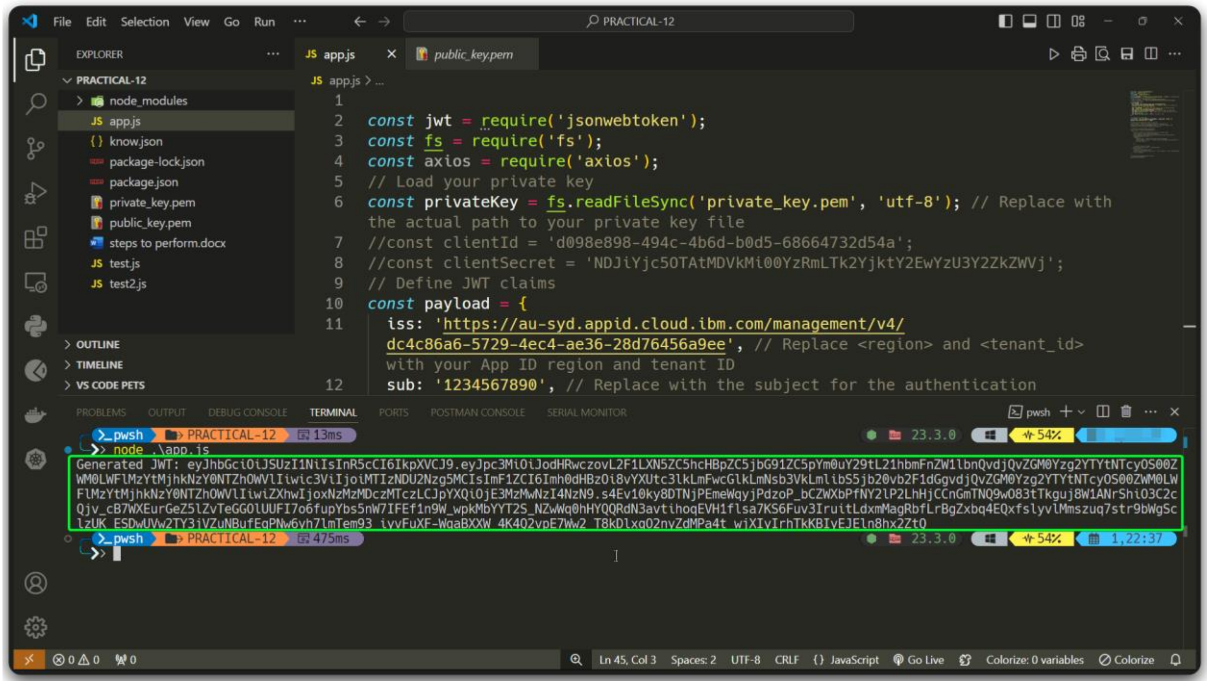
To above changes from your AppID service credentials like below you can see



Key name	Date created	Controlled by
kube-34a8edb54fe347328456cb2a25a7de3d.cr3cpfcs0m882o64nbq0.secure-file-storage-namespace.20241010065338	2024-10-10 12:23 PM	Advancedmobileaccess-d6aece47-d840-45b0-8ab9-ad15354deeea
Service-credentials-2	2024-10-17 9:41 AM	Advancedmobileaccess-d6aece47-d840-45b0-8ab9-ad15354deeea
Service-credentials-1	2024-10-10 11:33 AM	Advancedmobileaccess-d6aece47-d840-45b0-8ab9-ad15354deeea

```
{
  "apikey": "Y2LqL_fPU3NqY2zYR7IgUKqMdF7oqlBhRf4U6TTeGRH6",
  "appidServiceEndpoint": "https://au-syd.appid.cloud.ibm.com",
  "clientId": "57efb197-7f31-4313-bae1-e9234a1d9c76",
  "discoveryEndpoint": "https://au-syd.appid.cloud.ibm.com/oauth/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee/.well-known/ope
nid-configuration",
  "iam_apikey_description": "Auto-generated for key crn:v1:bluemix:public:appid:au-syd:a/9553f5f7184ddb922a056f240cf78ef
6:dc4c86a6-5729-4ec4-ae36-28d76456a9ee:resource-key:57efb197-7f31-4313-bae1-e9234a1d9c76",
  "iam_apikey_id": "ApiKey-6689e06a-d7c2-4ca0-841f-6a9f24c730de",
  "iam_apikey_name": "Service-credentials-2",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/9553f5f7184ddb922a056f240cf78ef6::serviceid:ServiceId-b816
9cc2-cb98-449c-802f-9328ff752a9d",
  "managementUrl": "https://au-syd.appid.cloud.ibm.com/management/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee",
  "oauthServerUrl": "https://au-syd.appid.cloud.ibm.com/oauth/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee",
  "profilesUrl": "https://au-syd.appid.cloud.ibm.com",
  "secret": "M2M0MDMyMDQtYWU4NC00NjExLWJiZWEtMzgxNGJiNDEyZWRm",
  "tenantId": "dc4c86a6-5729-4ec4-ae36-28d76456a9ee"
}
```

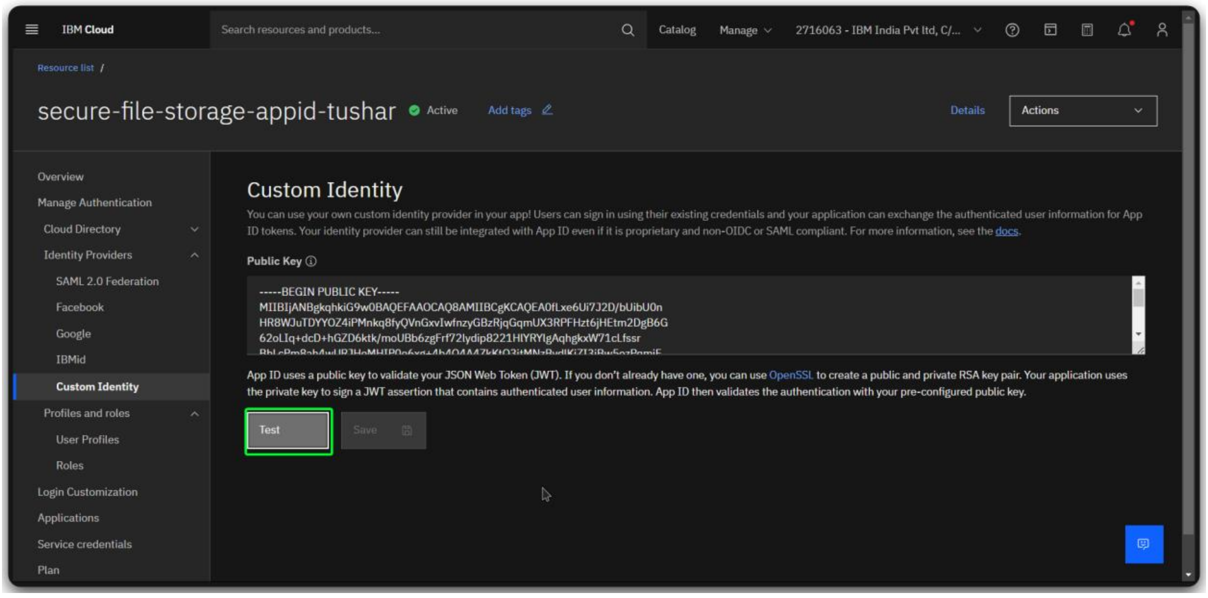
Now run the app.js to get JWT



```
const jwt = require('jsonwebtoken');
const fs = require('fs');
const axios = require('axios');
// Load your private key
const privateKey = fs.readFileSync('private_key.pem', 'utf-8'); // Replace with the actual path to your private key file
//const clientId = 'd098e898-494c-4b6d-b0d5-68664732d54a';
//const clientSecret = 'NDJiYjc5OTAtMDVkMi00YzRmLTk2YjktY2EwYzU3Y2ZkZWVj';
// Define JWT claims
const payload = {
  iss: 'https://au-syd.appid.cloud.ibm.com/management/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee', // Replace <region> and <tenant_id> with your App ID region and tenant ID
  sub: '1234567890', // Replace with the subject for the authentication
```

```
>_pwsh  PRACTICAL-12  13ms
>> node .\app.js
Generated JWT: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2F1LXN5ZC5hcHBpZC5jbG91ZC5pYm0uY29tL21hbmFnZW1lbnQvdjQvZGM0Yzg2YTYtNTcyOS00Z
WM0LWFlMzYtMjhkNzY0NTZhOWVlIiwic3ViIjoiMTIzNDU2Nzg5MCIsImF1ZCI6Imh0dHBzOi8vYXUtc3lkLmFwcGlkLmNsb3VkLmlibS5jb20vb2F1dGgvdjQvZGM0Yzg2YTYtNTcyOS00ZWM0LW
FlMzYtMjhkNzY0NTZhOWVlIiwiZXhwIjoxNzMzMDczMTczLCJpYXQiOjE3MzMwNzI4NzN9.s4Ev10ky8DTNjPEmeWqyjPdzoP_bCZWXbPfNY2lP2LhHjCCnGmTNQ9w083tTkguj8W1ANrShi03C2c
Qjv_cB7WXEurGeZ5lZvTeGGOlUUFI7o6fupYbs5nW7IFEf1n9W_wpkMbYYT2S_NZwWq0hHYQRdN3avtihoqEVH1flsa7KS6Fuv3IruitLdxmMagRbfLrBgZxbq4EQxfslyvlMmszuq7str9bWgSc
lzUK_ESDwUVw2TY3iVZuNBufEqPNw6yh7lmTem93_iyvFuXF-WqaBXXW_4K4Q2vpE7Ww2_T8kDlxqO2nyZdMPa4t_wiXIyIrhTkKBIyEJEln8hx2ZtQ
>_pwsh  PRACTICAL-12  475ms
>>
```

To test this JWT hit test button at custom identity



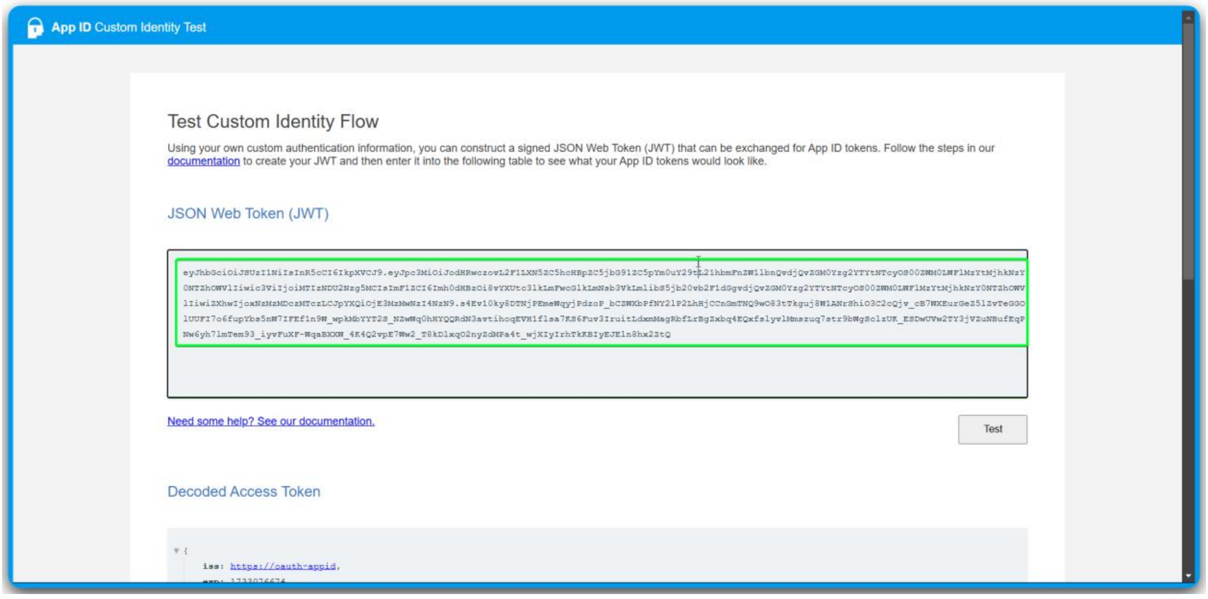
Custom Identity

You can use your own custom identity provider in your app! Users can sign in using their existing credentials and your application can exchange the authenticated user information for App ID tokens. Your identity provider can still be integrated with App ID even if it is proprietary and non-OIDC or SAML compliant. For more information, see the docs.

Public Key

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0fLxe6Ui7J2D/bUibU0n
HR8WJuTDYYOZ4iPMnkq8fyQVnGxvIwfnzyGBzRjqGqmUX3RPFHzt6jHEtm2DgB6G
62oLIq+dcD+hGZD6ktk/moUBb6zgFrf72lydip8221HIYRYlgAqhgkxW71cLfssr
```

App ID uses a public key to validate your JSON Web Token (JWT). If you don't already have one, you can use OpenSSL to create a public and private RSA key pair. Your application uses the private key to sign a JWT assertion that contains authenticated user information. App ID then validates the authentication with your pre-configured public key.



Test Custom Identity Flow

Using your own custom authentication information, you can construct a signed JSON Web Token (JWT) that can be exchanged for App ID tokens. Follow the steps in our documentation to create your JWT and then enter it into the following table to see what your App ID tokens would look like.

JSON Web Token (JWT)

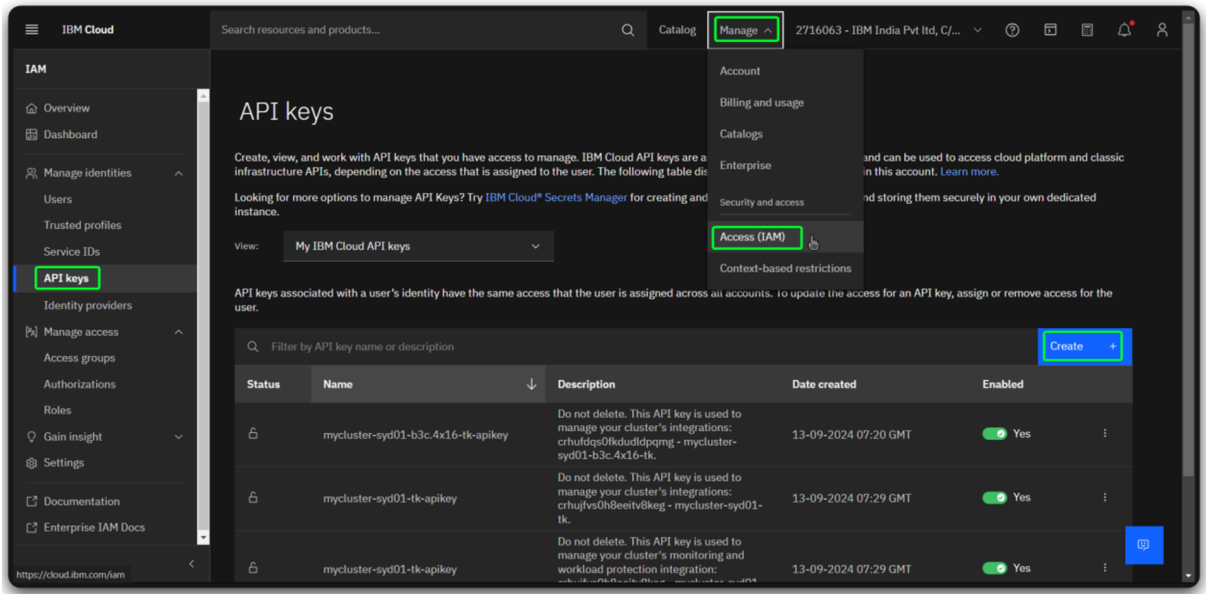
```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2F1LXN5ZC5hcHBpZC5jbG91ZC5pYm0uY29tL21hbmFnZW1lbnQvdjQvZGM0Yzg2YTYtNTcyOS00ZWM0LWFlMzYtMjhkNzY
0NTZhOWVlIiwic3ViIjoiMTIzNDU2Nzg5MCIsImF1ZCI6Imh0dHBzOi8vYXUtc3lkLmFwcGlkLmNsb3VkLmlibS5jb20vb2F1dGgvdjQvZGM0Yzg2YTYtNTcyOS00ZWM0LWFlMzYtMjhkNzY0NTZhOWV
lIiwiZXhwIjoxNzMzMDczMTczLCJpYXQiOjE3MzMwNzI4NzN9.s4Ev10ky8DTNjPEmeWqyjPdzoP_bCZWXbPfNY21P2LhHjCCnGmTNQ9wO83tTkguj8W1ANrShiO3C2cQjv_cB7WXEurGeZ5lZvTeGGO
1UUFI7o6fupYbs5nW7IFEf1n9W_wpkMbYYT2S_NZwWq0hHYQQRdN3avtihoqEVH1flsa7KS6Fuv3IruitLdxmMagRbfLrBgZxbq4EQxfslyvlMmszuq7str9bNgSclzUK_ESDwUVw2TY3jVZuNBufEqP
Nw6yh7lmTem93_iyvFuXF-WqaBXXW_4K4Q2vpE7Ww2_T8kDlxqO2nyZdMPa4t_wjXIyIrhTkKBIyEJEln8hx2ZtQ
```

Need some help? See our documentation.

Test

Decoded Access Token

```
▾ {
    iss: https://oauth-appid,
```



API keys

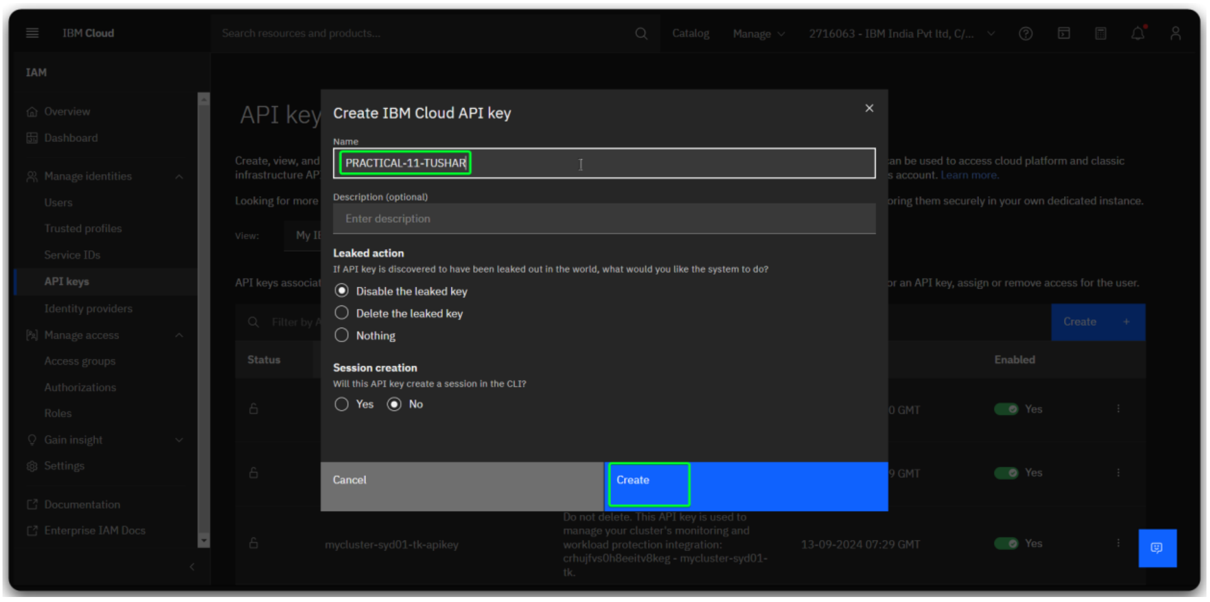
Create, view, and work with API keys that you have access to manage. IBM Cloud API keys are a and can be used to access cloud platform and classic infrastructure APIs, depending on the access that is assigned to the user. The following table dis in this account. Learn more.

Looking for more options to manage API Keys? Try IBM Cloud® Secrets Manager for creating and nd storing them securely in your own dedicated instance.

API keys associated with a user's identity have the same access that the user is assigned across all accounts. To update the access for an API key, assign or remove access for the user.

Status	Name	Description	Date created	Enabled
🔓	mycluster-syd01-b3c.4x16-tk-apikey	Do not delete. This API key is used to manage your cluster's integrations: crhufdqs0fkdudldpqmg - mycluster-syd01-b3c.4x16-tk.	13-09-2024 07:20 GMT	Yes
🔓	mycluster-syd01-tk-apikey	Do not delete. This API key is used to manage your cluster's integrations: crhujfvs0h8eeitv8keg - mycluster-syd01-tk.	13-09-2024 07:29 GMT	Yes
🔓	mycluster-syd01-tk-apikey	Do not delete. This API key is used to manage your cluster's monitoring and workload protection integration:	13-09-2024 07:29 GMT	Yes

Go to above section and hit create button



Create IBM Cloud API key

Name: PRACTICAL-11-TUSHAR

Description (optional): Enter description

Leaked action

If API key is discovered to have been leaked out in the world, what would you like the system to do?

[✓ Disable the leaked key] Delete the leaked key Nothing

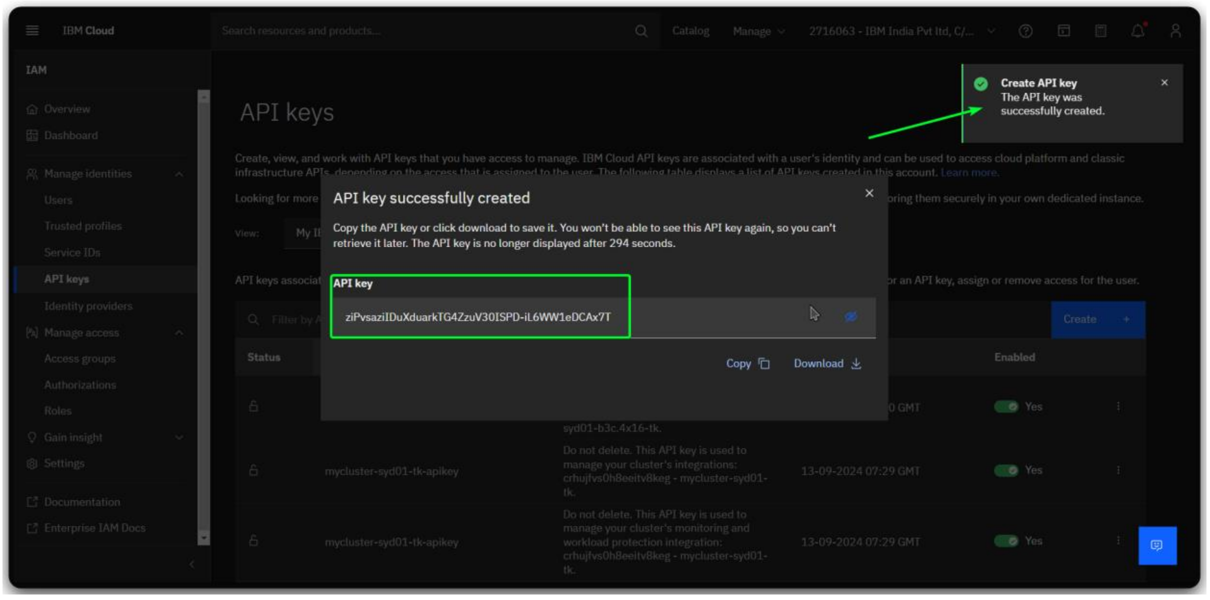
Session creation

Will this API key create a session in the CLI?

Yes [✓ No]

Cancel | Create

As you can see above our api key has been created successfully



Create API key
The API key was successfully created.

Create, view, and work with API keys that you have access to manage. IBM Cloud API keys are associated with a user's identity and can be used to access cloud platform and classic infrastructure APIs, depending on the access that is assigned to the user. The following table displays a list of API keys created in this account. Learn more.

API key successfully created

Copy the API key or click download to save it. You won't be able to see this API key again, so you can't retrieve it later. The API key is no longer displayed after 294 seconds.

API key

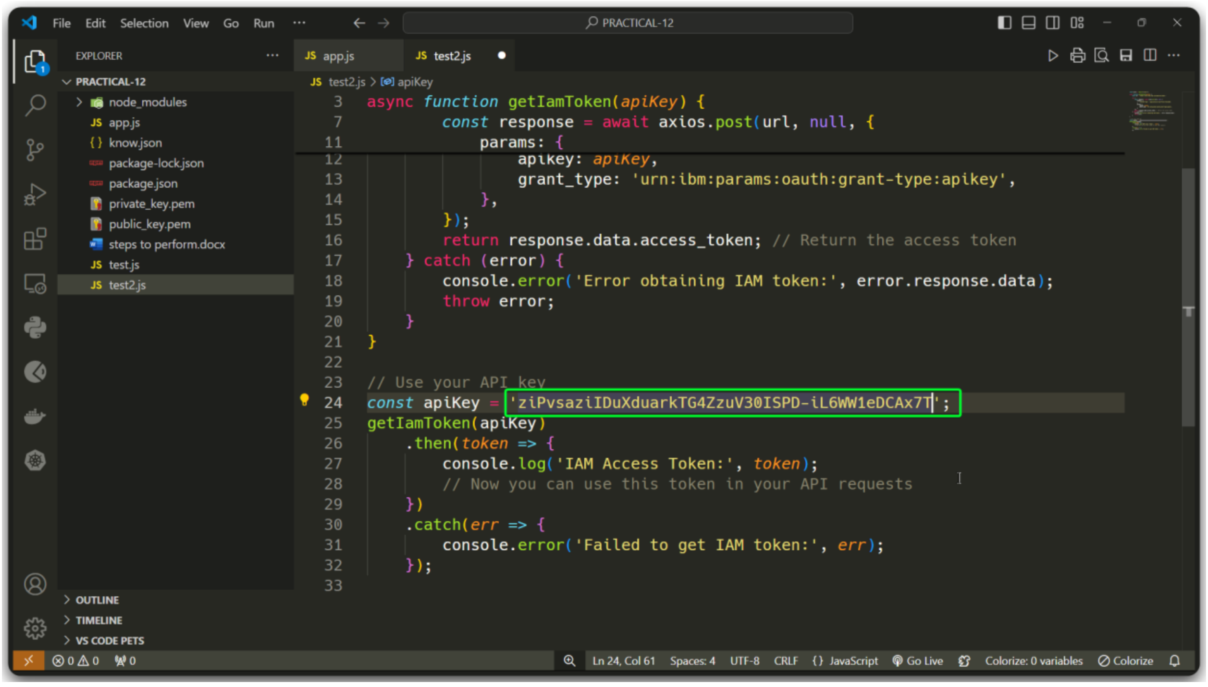
ziPvsaziIDuXduarkTG4ZzuV30ISPD-iL6WW1eDCAx7T

Copy | Download

Do not delete. This API key is used to manage your cluster's integrations: crhujfvs0h8eeitv8keg - mycluster-syd01-tk. — mycluster-syd01-tk-apikey — 13-09-2024 07:29 GMT — Yes

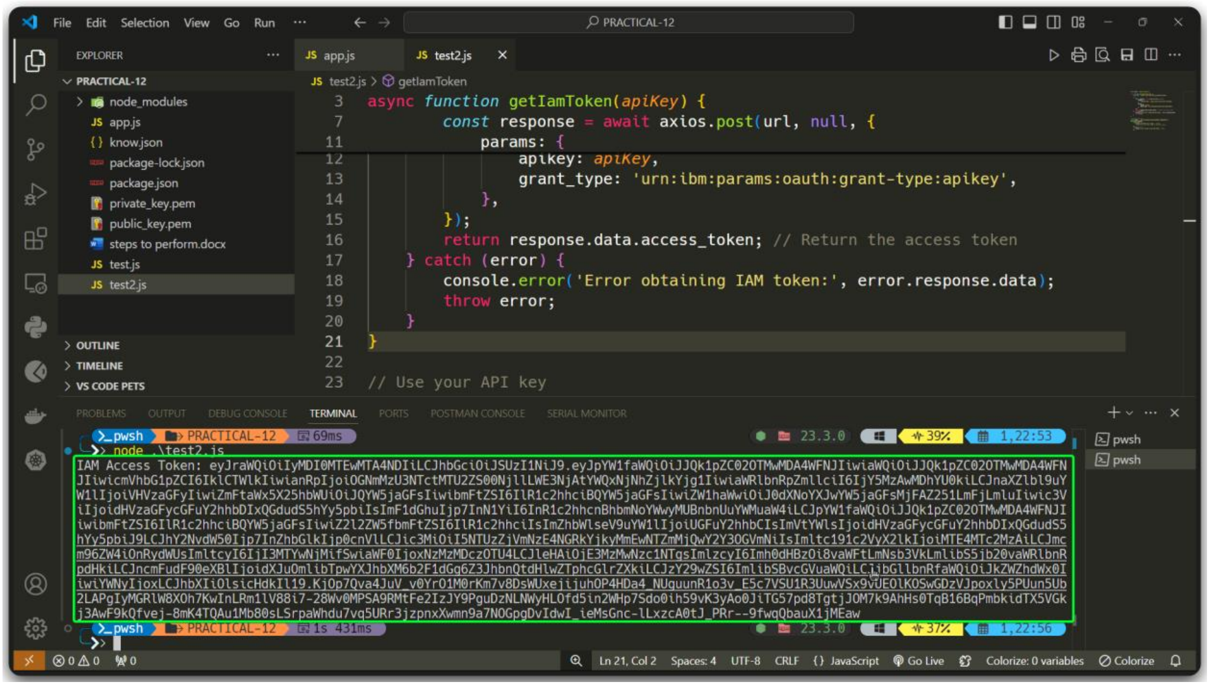
Do not delete. This API key is used to manage your cluster's monitoring and workload protection integration: crhujfvs0h8eeitv8keg - mycluster-syd01-tk. — mycluster-syd01-tk-apikey — 13-09-2024 07:29 GMT — Yes

Paste that API key into test2.js



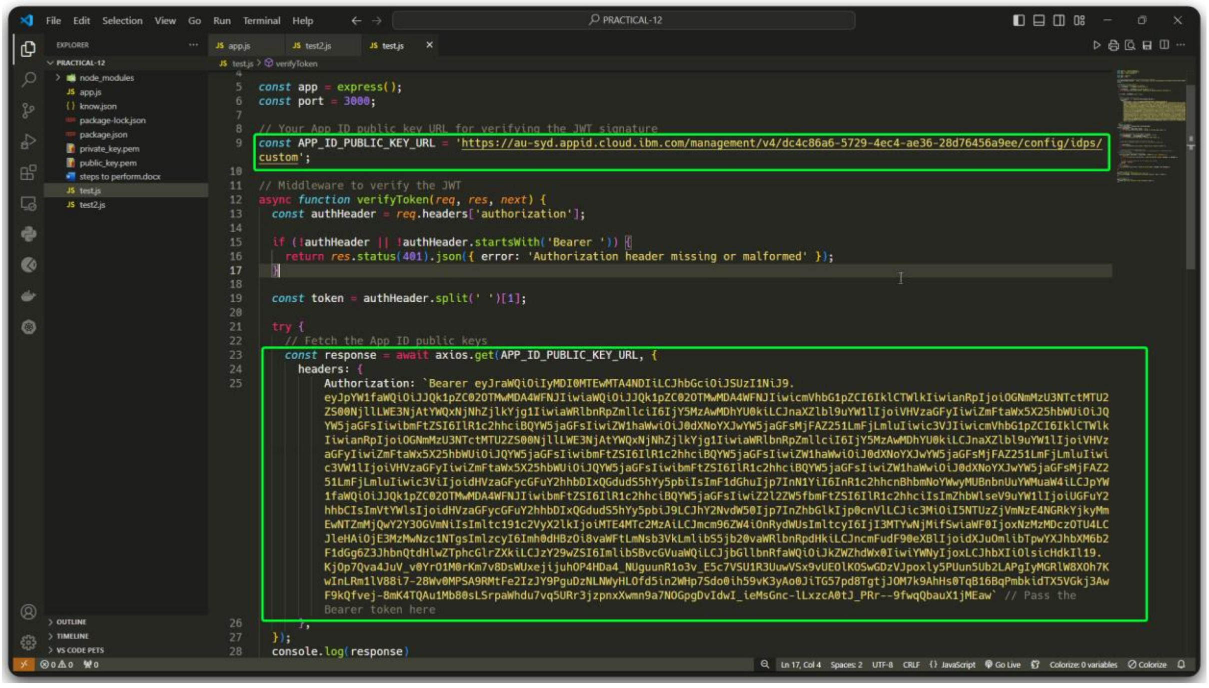
```javascript
async function getIamToken(apiKey) {
        const response = await axios.post(url, null, {
            params: {
                apiKey: apiKey,
                grant_type: 'urn:ibm:params:oauth:grant-type:apikey',
            },
        });
        return response.data.access_token; // Return the access token
    } catch (error) {
        console.error('Error obtaining IAM token:', error.response.data);
        throw error;
    }
}

// Use your API key
const apiKey = 'ziPvsaziIDuXduarkTG4ZzuV30ISPD-iL6WW1eDCAx7T';
getIamToken(apiKey)
    .then(token => {
        console.log('IAM Access Token:', token);
        // Now you can use this token in your API requests
    })
    .catch(err => {
        console.error('Failed to get IAM token:', err);
    });
```

Now run the test2.js and generate access token

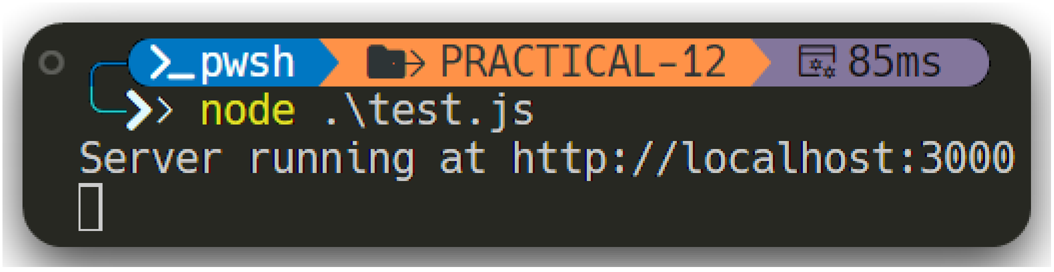


```
>> node .\test2.js
IAM Access Token: eyJraWQiOiIyMDI0MTEwMTA4NDIiLCJhbGciOiJSUzI1NiJ9.eyJpYW1faWQiOiJJQk1pZC02OTMwMDA4WFNJIiwiaWQiOiJJQk1pZC02OTMwMDA4WFN
JIiwicmVhbG1pZCI6IklCTWlkIiwianRpIjoiOGNmMzU3NTctMTU2ZS00NjllLWE3NjAtYWQxNjNhZjlkYjg1IiwiaWRlbnRpZmllciI6IjY5MzAwMDhYU0kiLCJnaXZlbl9uY
W1lIjoiVHVzaGFyIiwiZmFtaWx5X25hbWUiOiJQYW5jaGFsIiwibmFtZSI6IlR1c2hhciBQYW5jaGFsIiwiZW1haWwiOiJ0dXNoYXJwYW5jaGFsMjFAZ251LmFjLmluIiwic3V
iIjoidHVzaGFycGFuY2hhbDIxQGdudS5hYy5pbiIsImF1dGhuIjp7InN1YiI6InR1c2hhcnBhbmNoYWwyMUBnbnUuYWMuaW4iLCJpYW1faWQiOiJJQk1pZC02OTMwMDA4WFNJI
iwibmFtZSI6IlR1c2hhciBQYW5jaGFsIiwiZ2l2ZW5fbmFtZSI6IlR1c2hhciIsImZhbWlseV9uYW1lIjoiUGFuY2hhbCIsImVtYWlsIjoidHVzaGFycGFuY2hhbDIxQGdudS5
hYy5pbiJ9LCJhY2NvdW50Ijp7InZhbGlkIjp0cnVlLCJic3MiOiI5NTUzZjVmNzE4NGRkYjkyMmEwNTZmMjQwY2Y3OGVmNiIsImltc191c2VyX2lkIjoiMTE4MTc2MzAiLCJmc
m96ZW4iOnRydWUsImltcyI6IjI3MTYwNjMifSwiaWF0IjoxNzMzMDczOTU4LCJleHAiOjE3MzMwNzc1NTgsImlzcyI6Imh0dHBzOi8vaWFtLmNsb3VkLmlibS5jb20vaWRlbnR
pdHkiLCJncmFudF90eXBlIjoidXJuOmlibTpwYXJhbXM6b2F1dGg6Z3JhbnQtdHlwZTphcGlrZXkiLCJzY29wZSI6ImlibSBvcGVuaWQiLCJjbGllbnRfaWQiOiJkZWZhdWx0I
iwiYWNyIjoxLCJhbXIiOlsicHdkIl19.KjOp7Qva4JuV_v0YrO1M0rKm7v8DsWUxejijuhOP4HDa4_NUguunR1o3v_E5c7VSU1R3UuwVSx9vUEOlKOSwGDzVJpoxly5PUun5Ub
2LAPgIyMGRlW8XOh7KwInLRm1lV88i7-28Wv0MPSA9RMtFe2IzJY9PguDzNLNWyHLOfd5in2WHp7Sdo0ih59vK3yAo0JiTG57pd8TgtjJOM7k9AhHs0TqB16BqPmbkidTX5VGk
j3AwF9kQfvej-8mK4TQAu1Mb80sLSrpaWhdu7vq5URr3jzpnxXwmn9a7NOGpgDvIdwI_ieMsGnc-lLxzcA0tJ_PRr--9fwqQbauX1jMEaw
>>
```

Now paste this access token and change your public url from your service credentials into test.js and run it

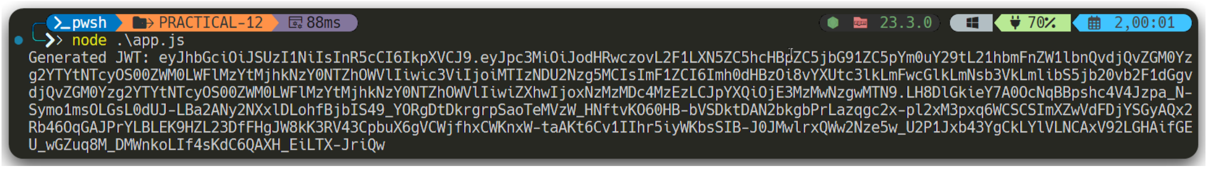


```javascript
const app = express();
const port = 3000;

// Your App ID public key URL for verifying the JWT signature
const APP_ID_PUBLIC_KEY_URL = 'https://au-syd.appid.cloud.ibm.com/management/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee/config/idps/custom';

// Middleware to verify the JWT
async function verifyToken(req, res, next) {
  const authHeader = req.headers['authorization'];

  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return res.status(401).json({ error: 'Authorization header missing or malformed' });
  }

  const token = authHeader.split(' ')[1];

  try {
    // Fetch the App ID public keys
    const response = await axios.get(APP_ID_PUBLIC_KEY_URL, {
      headers: {
        Authorization: `Bearer eyJraWQiOiIyMDI0MTEwMTA4NDIiLCJhbGciOiJSUzI1NiJ9.
        eyJpYW1faWQiOiJJQk1pZC02OTMwMDA4WFNJIiwiaWQiOiJJQk1pZC02OTMwMDA4WFNJIiwicmVhbG1pZCI6IklCTWlkIiwianRpIjoiOGNmMzU3NTctMTU2
        ZS00NjllLWE3NjAtYWQxNjNhZjlkYjg1IiwiaWRlbnRpZmllciI6IjY5MzAwMDhYU0kiLCJnaXZlbl9uYW1lIjoiVHVzaGFyIiwiZmFtaWx5X25hbWUiOiJQ
        YW5jaGFsIiwibmFtZSI6IlR1c2hhciBQYW5jaGFsIiwiZW1haWwiOiJ0dXNoYXJwYW5jaGFsMjFAZ251LmFjLmluIiwic3VJIiwicmVhbG1pZCI6IklCTWlk
        IiwianRpIjoiOGNmMzU3NTctMTU2ZS00NjllLWE3NjAtYWQxNjNhZjlkYjg1IiwiaWRlbnRpZmllciI6IjY5MzAwMDhYU0kiLCJnaXZlbl9uYW1lIjoiVHVz
        aGFyIiwiZmFtaWx5X25hbWUiOiJQYW5jaGFsIiwibmFtZSI6IlR1c2hhciBQYW5jaGFsIiwiZW1haWwiOiJ0dXNoYXJwYW5jaGFsMjFAZ251LmFjLmluIiwi
        c3VW1lIjoiVHVzaGFyIiwiZmFtaWx5X25hbWUiOiJQYW5jaGFsIiwibmFtZSI6IlR1c2hhciBQYW5jaGFsIiwiZW1haWwiOiJ0dXNoYXJwYW5jaGFsMjFAZ2
        51LmFjLmluIiwic3ViIjoidHVzaGFycGFuY2hhbDIxQGdudS5hYy5pbiIsImF1dGhuIjp7InN1YiI6InR1c2hhcnBhbmNoYWwyMUBnbnUuYWMuaW4iLCJpYW
        1faWQiOiJJQk1pZC02OTMwMDA4WFNJIiwibmFtZSI6IlR1c2hhciBQYW5jaGFsIiwiZ2l2ZW5fbmFtZSI6IlR1c2hhciIsImZhbWlseV9uYW1lIjoiUGFuY2
        hhbCIsImVtYWlsIjoidHVzaGFycGFuY2hhbDIxQGdudS5hYy5pbiJ9LCJhY2NvdW50Ijp7InZhbGlkIjp0cnVlLCJic3MiOiI5NTUzZjVmNzE4NGRkYjkyMm
        EwNTZmMjQwY2Y3OGVmNiIsImltc191c2VyX2lkIjoiMTE4MTc2MzAiLCJmcm96ZW4iOnRydWUsImltcyI6IjI3MTYwNjMifSwiaWF0IjoxNzMzMDczOTU4LC
        JleHAiOjE3MzMwNzc1NTgsImlzcyI6Imh0dHBzOi8vaWFtLmNsb3VkLmlibS5jb20vaWRlbnRpdHkiLCJncmFudF90eXBlIjoidXJuOmlibTpwYXJhbXM6b2
        F1dGg6Z3JhbnQtdHlwZTphcGlrZXkiLCJzY29wZSI6ImlibSBvcGVuaWQiLCJjbGllbnRfaWQiOiJkZWZhdWx0IiwiYWNyIjoxLCJhbXIiOlsicHdkIl19.
        KjOp7Qva4JuV_v0YrO1M0rKm7v8DsWUxejijuhOP4HDa4_NUguunR1o3v_E5c7VSU1R3UuwVSx9vUEOlKOSwGDzVJpoxly5PUun5Ub2LAPgIyMGRlW8XOh7K
        wInLRm1lV88i7-28Wv0MPSA9RMtFe2IzJY9PguDzNLNWyHLOfd5in2WHp7Sdo0ih59vK3yAo0JiTG57pd8TgtjJOM7k9AhHs0TqB16BqPmbkidTX5VGkj3Aw
        F9kQfvej-8mK4TQAu1Mb80sLSrpaWhdu7vq5URr3jzpnxXwmn9a7NOGpgDvIdwI_ieMsGnc-lLxzcA0tJ_PRr--9fwqQbauX1jMEaw` // Pass the
        Bearer token here
      },
    });
  console.log(response)
```



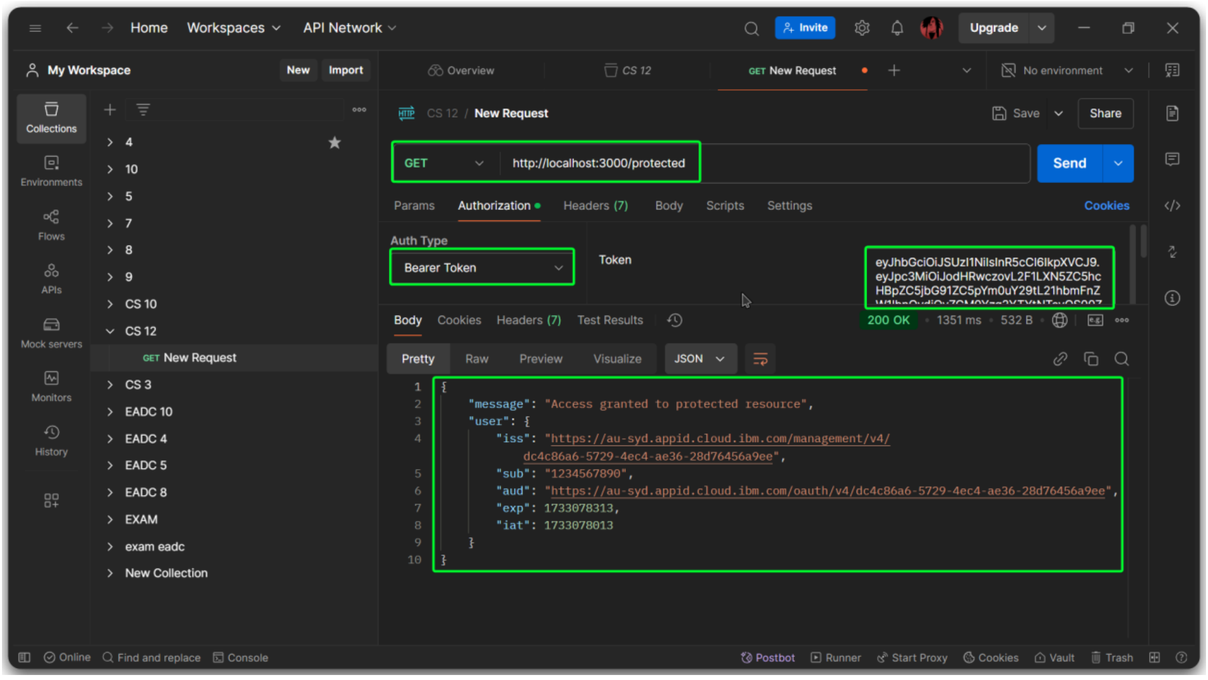
```
>_pwsh  PRACTICAL-12  85ms
>> node .\test.js
Server running at http://localhost:3000
```

Now run app.js into second terminal window



```
>_pwsh  PRACTICAL-12  88ms                                                        23.3.0    70%   2,00:01
>> node .\app.js
Generated JWT: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2F1LXN5ZC5hcHBpZC5jbG91ZC5pYm0uY29tL21hbmFnZW1lbnQvdjQvZGM0Yz
g2YTYtNTcyOS00ZWM0LWFlMzYtMjhkNzY0NTZhOWVlIiwic3ViIjoiMTIzNDU2Nzg5MCIsImF1ZCI6Imh0dHBzOi8vYXUtc3lkLmFwcGlkLmNsb3VkLmlibS5jb20vb2F1dGgv
djQvZGM0Yzg2YTYtNTcyOS00ZWM0LWFlMzYtMjhkNzY0NTZhOWVlIiwiZXhwIjoxNzMzMDc4MzEzLCJpYXQiOjE3MzMwNzgwMTN9.LH8DlGkieY7A0OcNqBBpshc4V4Jzpa_N-
Symo1msOLGsL0dUJ-LBa2ANy2NXxlDLohfBjbIS49_YORgDtDkrgrpSaoTeMVzW_HNftvKO60HB-bVSDktDAN2bkgbPrLazqgc2x-pl2xM3pxq6WCSCSImXZwVdFDjYSGyAQx2
Rb46OqGAJPrYLBLEK9HZL23DfFHgJW8kK3RV43CpbuX6gVCWjfhxCWKnxW-taAKt6Cv1IIhr5iyWKbsSIB-J0JMwlrxQWw2Nze5w_U2P1Jxb43YgCkLYlVLNCAxV92LGHAifGE
U_wGZuq8M_DMWnkoLIf4sKdC6QAXH_EiLTX-JriQw
```

Go to the postman and paste our before generated JWT token from app.js into authorization of postman select bearer token and in token paste that



GET http://localhost:3000/protected

Auth Type: Bearer Token

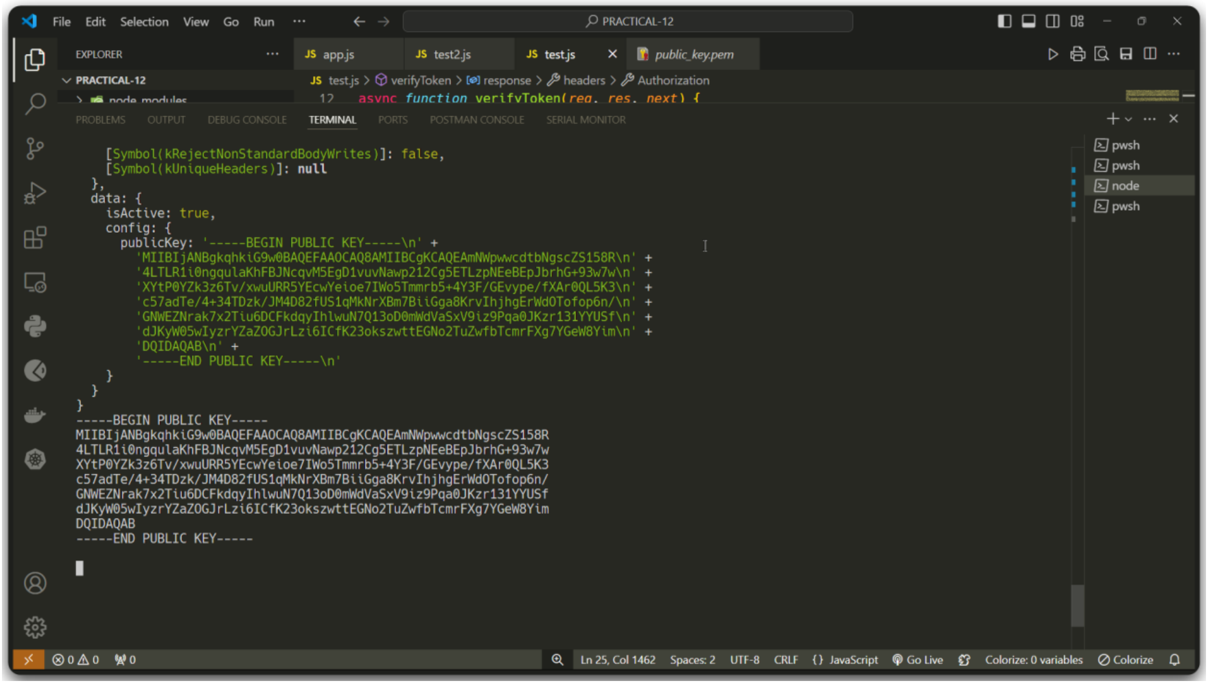
Token

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2F1LXN5ZC5hc
HBpZC5jbG91ZC5pYm0uY29tL21hbmFnZ
```

200 OK · 1351 ms · 532 B

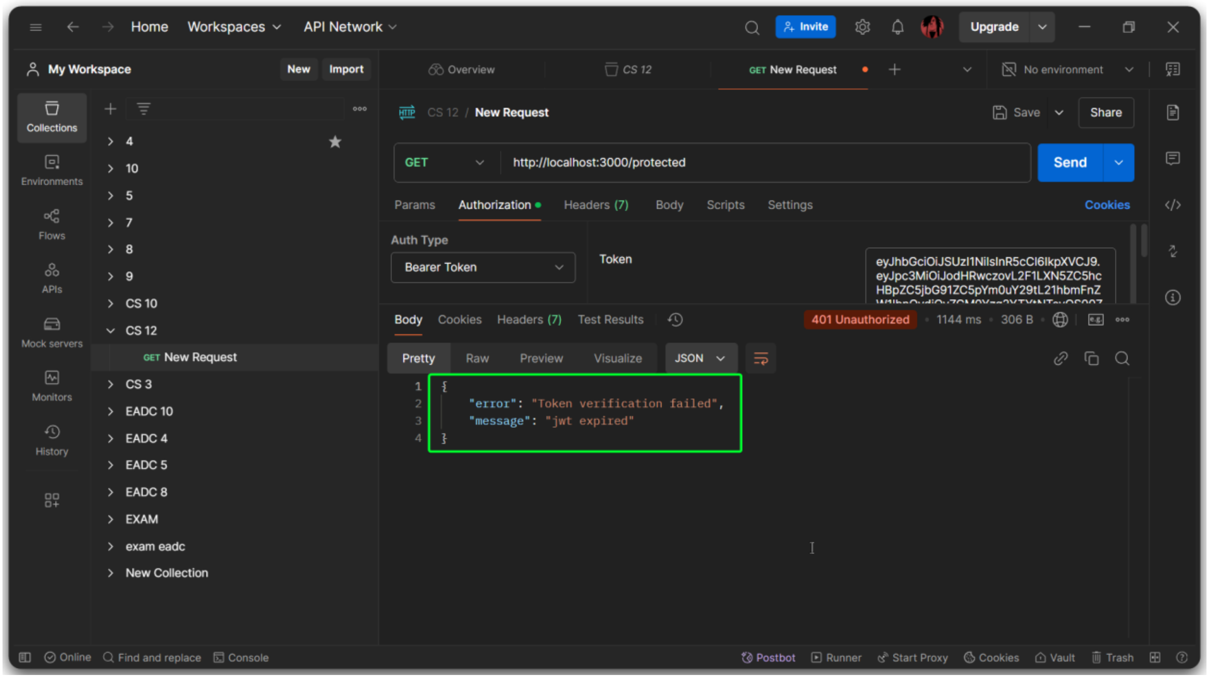
```json
{
    "message": "Access granted to protected resource",
    "user": {
        "iss": "https://au-syd.appid.cloud.ibm.com/management/v4/
          dc4c86a6-5729-4ec4-ae36-28d76456a9ee",
        "sub": "1234567890",
        "aud": "https://au-syd.appid.cloud.ibm.com/oauth/v4/dc4c86a6-5729-4ec4-ae36-28d76456a9ee",
        "exp": 1733078313,
        "iat": 1733078013
    }
}
```

You can see that we got our public key in terminal as well



```
      [Symbol(kRejectNonStandardBodyWrites)]: false,
      [Symbol(kUniqueHeaders)]: null
    },
    data: {
      isActive: true,
      config: {
        publicKey: '-----BEGIN PUBLIC KEY-----\n' +
          'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmNWpwwcdtbNgscZS158R\n' +
          '4LTLR1i0ngqulaKhFBJNcqvM5EgD1vuvNawp212Cg5ETLzpNEeBEpJbrhG+93w7w\n' +
          'XYtP0YZk3z6Tv/xwuURR5YEcwYeioe7IWo5Tmmrb5+4Y3F/GEvype/fXAr0QL5K3\n' +
          'c57adTe/4+34TDzk/JM4D82fUS1qMkNrXBm7BiiGga8KrvIhjhgErWdOTofop6n/\n' +
          'GNWEZNrak7x2Tiu6DCFkdqyIhlwuN7Q13oD0mWdVaSxV9iz9Pqa0JKzr131YYUSf\n' +
          'dJKyW05wIyzrYZaZOGJrLzi6ICfK23okszwttEGNo2TuZwfbTcmrFXg7YGeW8Yim\n' +
          'DQIDAQAB\n' +
          '-----END PUBLIC KEY-----\n'
      }
    }
  }
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmNWpwwcdtbNgscZS158R
4LTLR1i0ngqulaKhFBJNcqvM5EgD1vuvNawp212Cg5ETLzpNEeBEpJbrhG+93w7w
XYtP0YZk3z6Tv/xwuURR5YEcwYeioe7IWo5Tmmrb5+4Y3F/GEvype/fXAr0QL5K3
c57adTe/4+34TDzk/JM4D82fUS1qMkNrXBm7BiiGga8KrvIhjhgErWdOTofop6n/
GNWEZNrak7x2Tiu6DCFkdqyIhlwuN7Q13oD0mWdVaSxV9iz9Pqa0JKzr131YYUSf
dJKyW05wIyzrYZaZOGJrLzi6ICfK23okszwttEGNo2TuZwfbTcmrFXg7YGeW8Yim
DQIDAQAB
-----END PUBLIC KEY-----
```

If we run that request second time it will show us error that is our token is expired.



GET http://localhost:3000/protected

Auth Type: Bearer Token

Token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2F1LXN5ZC5hcHBpZC5jbG91ZC5pYm0uY29tL21hbmFnZ

401 Unauthorized · 1144 ms · 306 B

```
{
    "error": "Token verification failed",
    "message": "jwt expired"
}
```