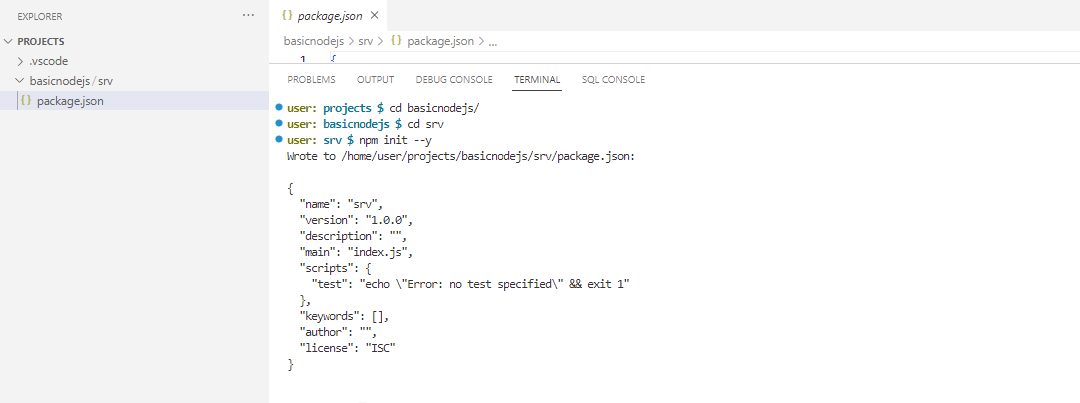
Lets create a folder basicnodejs

Then inside create another folder srv, which will be for service provide.

Now from terminal change the directory and go inside srv folder

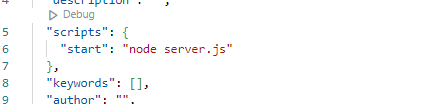
Run command: npm init –y



Now add/modify under scripts.

"start": "node server.js"

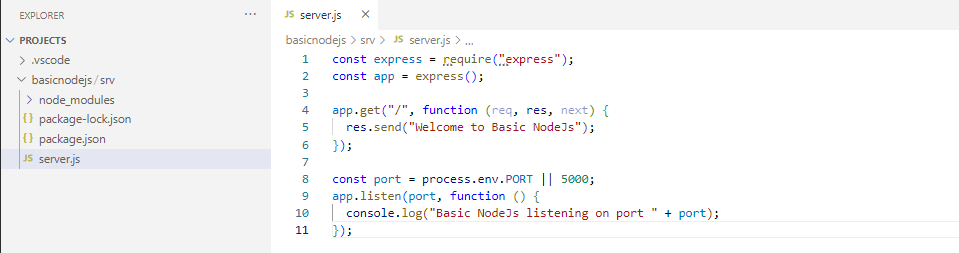
Whenever we run npm run start, it will execute the file server.js



We have to install express package- npm install express



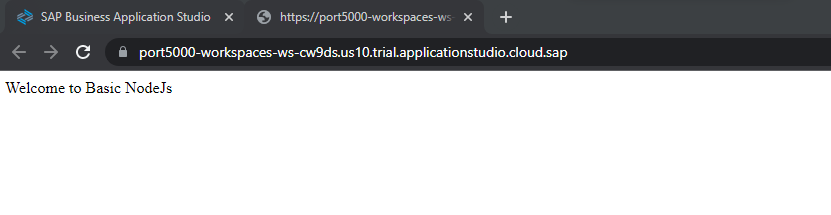
Create a server.js file inside srv folder, and add the below code



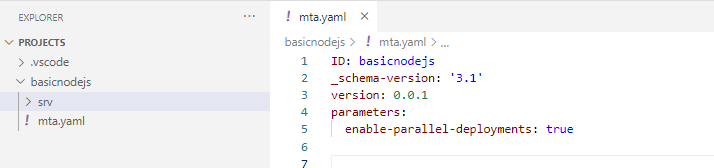
Execute the command – npm run start



Our node js application is running the static port 5000 that we have mentioned.



Now lets create a simple mta.yaml file in the root directory and deploy our application to BTP.



Providing the details to mta for deployment details –

ID: basicnodejs

\_schema-version: '3.1'

version: 0.0.1

parameters:

  enable-parallel-deployments: true

modules:

  - name: basicnodejs-service

    type: nodejs

    path: srv

    build-parameters:

      ignore:

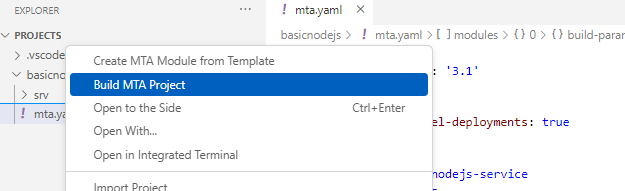
        - 'default-\*.json'

        - .env

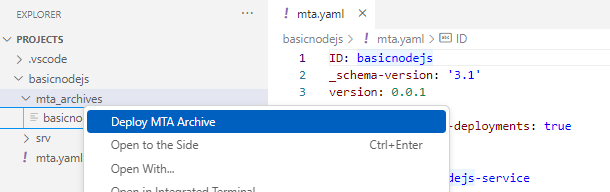
        - '\*node\_modules\*'

        - package-lock.json

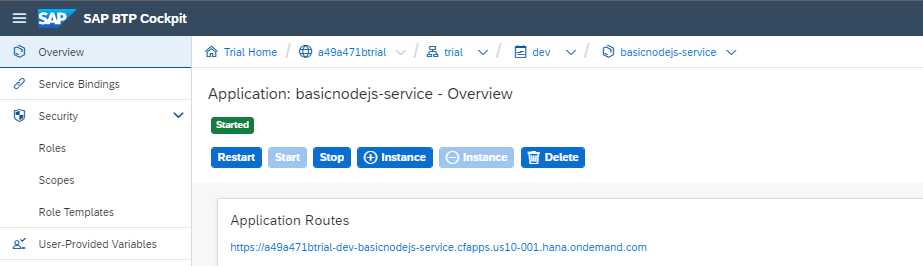
Build the MTA project

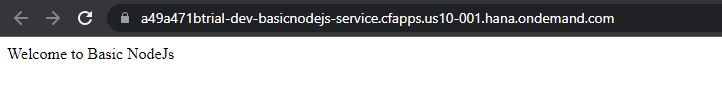


Deploy it



App is deployed and ready to run





We can access without any restriction from anywhere without having BTP account.

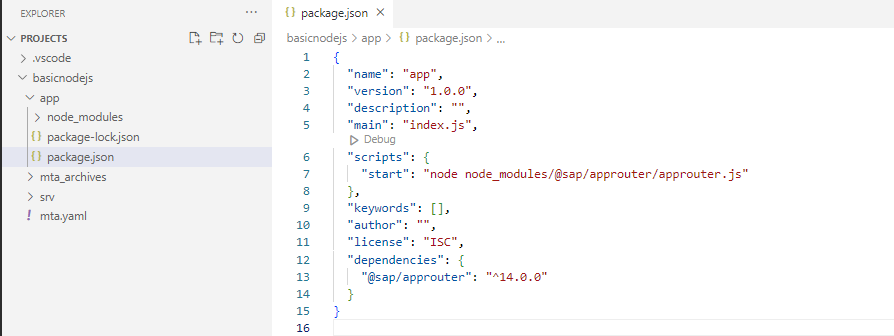
So to make it restricted we will use approuter and authentication service in BTP

For approuter, or to handle the routing, create app folder and package.json inside it

Install @sap/approuter inside app folder as a npm package.

Now add the script

"start": "node [node\_modules/@sap/approuter/approuter.js](mailto:node_modules/@sap/approuter/approuter.js)"



Now create file xs-app.json which will handle the reroutes.



“srv-api” is just a name that holds some information of a destination url

Lets create the approuter module.

  - name: basicnodejs-approuter

    type: approuter.nodejs

    path: app

    build-parameters:

      ignore:

        - 'default-\*.json'

        - .env

        - '\*node\_modules\*'

        - package-lock.json

    parameters:

      memory: 256M

      disk-quota: 512M

      keep-existing-routes: true

Now if we deployed the mta with this details only. It will crash.

Approuter will require some parameter which will provided by the Service module

ID: basicnodejs

\_schema-version: '3.1'

version: 0.0.1

parameters:

  enable-parallel-deployments: true

modules:

  - name: basicnodejs-service

    type: nodejs

    path: srv

    build-parameters:

      ignore:

        - 'default-\*.json'

        - .env

        - '\*node\_modules\*'

        - package-lock.json

    provides:

      - name: srv-api

        properties:

          srv-url: ${default-url}

  - name: basicnodejs-approuter

    type: approuter.nodejs

    path: app

    build-parameters:

      ignore:

        - 'default-\*.json'

        - .env

        - '\*node\_modules\*'

        - package-lock.json

    parameters:

      memory: 256M

      disk-quota: 512M

      keep-existing-routes: true

    requires:

      - name: srv-api

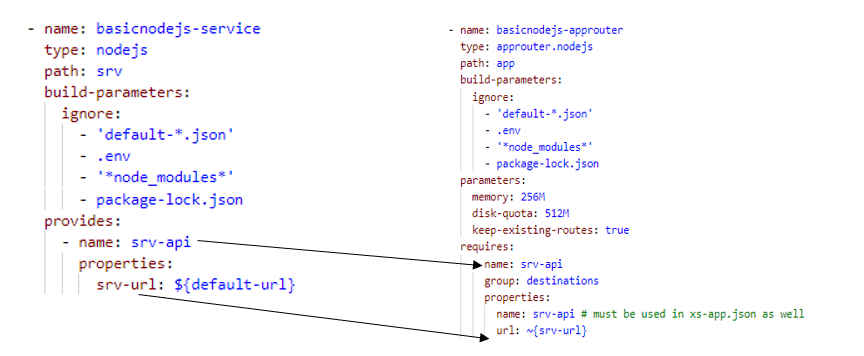
        group: destinations

        properties:

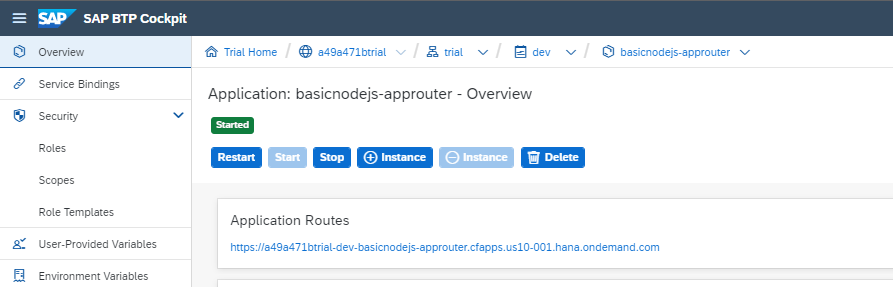
          name: srv-api # must be used in xs-app.json as well

          url: ~{srv-url}

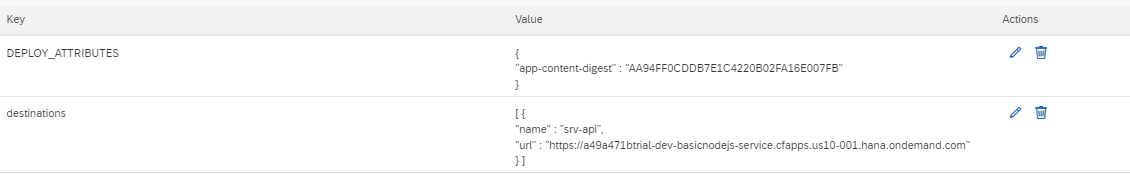
basicnodejs-service provides configuration variables which will be required in basicnodejs-approuter



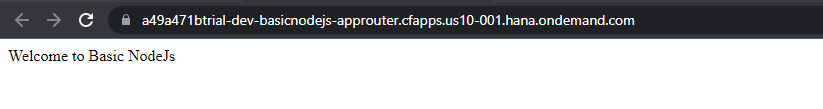
Now again deploy the mta.yaml



If we check the User-Provided Variables we can find the variables that we have set in the approuter requires section



Now open the approuter url. We can access it without any credentials.



Now the restrictions and authentication part

To do that we will need BTP authorization and Trust management service (XSUAA)

Which we will create using MTA.yaml file.

We will require xs-security.json file, create it in the root directory ,which hold the information of user authorization.

Lets add xsuaa service

resources:

 - name: basicnodejs-xsuaa

   type: org.cloudfoundry.managed-service

   parameters:

     path: ./xs-security.json

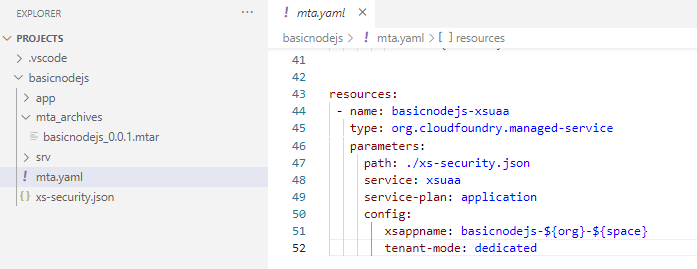
     service: xsuaa

     service-plan: application

     config:

        xsappname: basicnodejs-${org}-${space}

        tenant-mode: dedicated



Build and deploy mta.yaml



But who will use this service.

We have to specify that two modules are going to access this service.

ID: basicnodejs

\_schema-version: '3.1'

version: 0.0.1

parameters:

  enable-parallel-deployments: true

modules:

  - name: basicnodejs-service

    type: nodejs

    path: srv

    build-parameters:

      ignore:

        - 'default-\*.json'

        - .env

        - '\*node\_modules\*'

        - package-lock.json

    provides:

      - name: srv-api

        properties:

          srv-url: ${default-url}

    requires:

      - name: basicnodejs-xsuaa

  - name: basicnodejs-approuter

    type: approuter.nodejs

    path: app

    build-parameters:

      ignore:

        - 'default-\*.json'

        - .env

        - '\*node\_modules\*'

        - package-lock.json

    parameters:

      memory: 256M

      disk-quota: 512M

      keep-existing-routes: true

    requires:

      - name: srv-api

        group: destinations

        properties:

          name: srv-api # must be used in xs-app.json as well

          url: ~{srv-url}

      forwardAuthToken: true

      - name: basicnodejs-xsuaa

resources:

 - name: basicnodejs-xsuaa

   type: org.cloudfoundry.managed-service

   parameters:

     service: xsuaa

     service-plan: application

     config:

        xsappname: basicnodejs-${org}-${space}

        tenant-mode: dedicated

It will not use the authentication mechanism of BTP until we mention route.

In our xs-app.json file modify authenticationMethod none to route

{

    "authenticationMethod": "route",

    "routes": [{

        "source": "^/(.\*)$",

        "target": "$1",

        "destination": "srv-api"

    }]

}

This will redirect us to the BTP login page if you are not logged in.

Even if we provide the BTP credentials we can not access our desired application.

One more parameter in the XSUAA service which tells where to redirect after the authentication

resources:

 - name: basicnodejs-xsuaa

   type: org.cloudfoundry.managed-service

   parameters:

     service: xsuaa

     service-plan: application

     config:

        xsappname: basicnodejs-${org}-${space}

        tenant-mode: dedicated

        oauth2-configuration:

          redirect-uris:

          - "https://\*.hana.ondemand.com/\*\*"

Deploy and execute the approuter. It will ask you to login and then landed you on the Service.

Wait….. Even we have done the authentication mechanism, we are able to access the direct basicnodejs-service url in BTP.

Because we have not provided any condition if the user is authenticated by the XSUAA. In CAPM it is handled by the framework.

But our case we have to do it manually.

Lets add some npm package and modify server. File

npm install @sap/xsenv @sap/xssec passport

{

  "name": "srv",

  "version": "1.0.0",

  "description": "",

  "scripts": {

    "start": "node server.js"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "@sap/xsenv": "^3.4.0",

    "@sap/xssec": "^3.2.17",

    "express": "^4.18.2",

    "passport": "^0.6.0"

  }

}

Server.js

const express = require("express");

const passport = require("passport");

const xsenv = require("@sap/xsenv");

const JWTStrategy = require("@sap/xssec").JWTStrategy;

const services = xsenv.getServices({ uaa:"basicnodejs-xsuaa" });  // XSUAA service

const app = express();

passport.use(new JWTStrategy(services.uaa));

app.use(passport.initialize());

app.use(passport.authenticate("JWT", { session: false }));

app.get("/", function (req, res, next) {

  res.send("Welcome User: " + req.user.name.givenName);

});

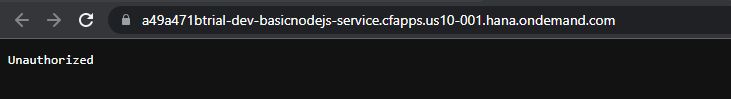
const port = process.env.PORT || 5000;

app.listen(port, function () {

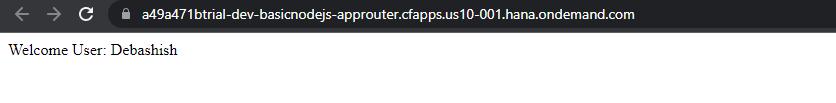
  console.log("Basic NodeJs listening on port " + port);

});

Now try to execute the service url from BTP



Execute Approuter



Lets Run our app locally using XSUAA service

We are changing the port number from 5000 to 4000.

We have to create default destinations which will be called for our local application.

Create default-env.json file under app folder. Provide the below details –

"destinations": [

        {

            "name": "srv-api",

            "url": "http://localhost:4000/",

            "forwardAuthToken": true,

            "strictSSL": false

        }

    ]

This is refers to service url. We have to provide the redirect uris for our local application

resources:

  - name: basicnodejs-xsuaa

    type: org.cloudfoundry.managed-service

    parameters:

      service: xsuaa

      service-plan: application

      config:

        xsappname: basicnodejs-${org}-${space}

        tenant-mode: dedicated

        oauth2-configuration:

          redirect-uris:

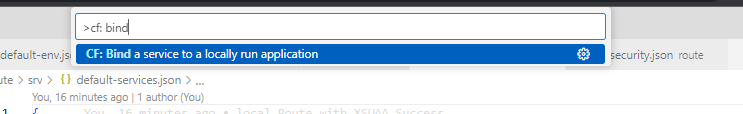
            - https://\*.hana.ondemand.com/\*\*

            - https://\*.trial.applicationstudio.cloud.sap/\*\*

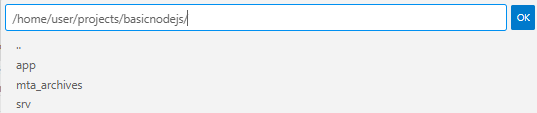
Now we have to bind our local app with xsuaa service.

Now build and deploy the mta.yaml, as it will update xsuaa service.

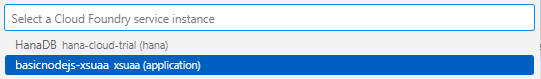
To do that in easy way, open the command palette -> CF: bind a service locally run application



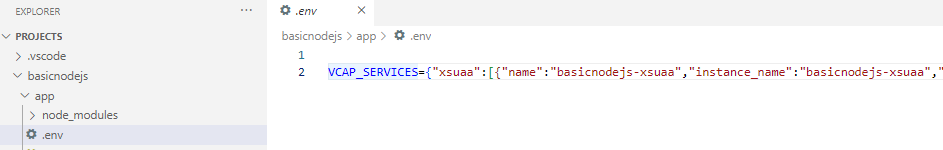
Select the project -> app folder (in our case)



Choose the required xsuaa service



This will create the .env file, which holds the binding information of the approuter apps with xsuaa service



Copy and paste it in default-env.json file. And format it to json



Now add defult-services.json file. Make sure file name should be same like default-services.json

Add the following –

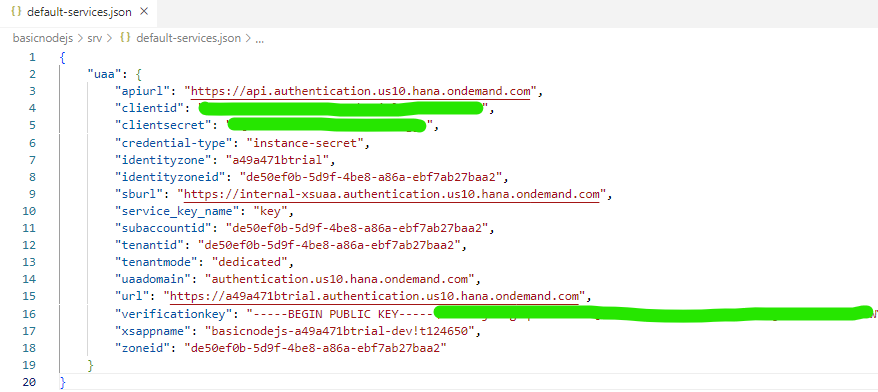
{

    "uaa": {

    }

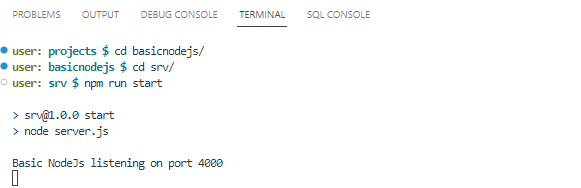
}

Now paste the credential part of our xsuaa service from app/default-env.json file

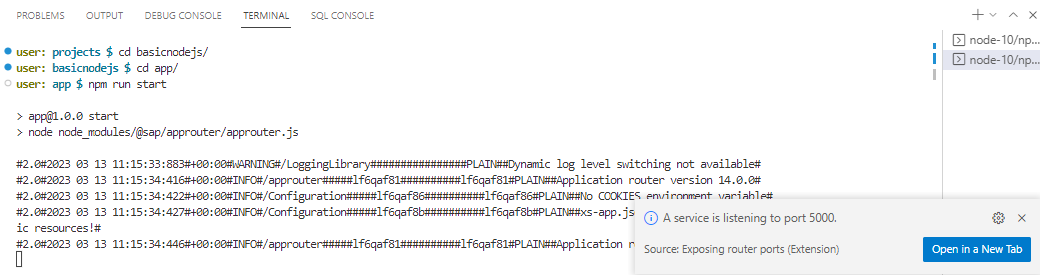


Now start the service first. Navigate to the srv folder.

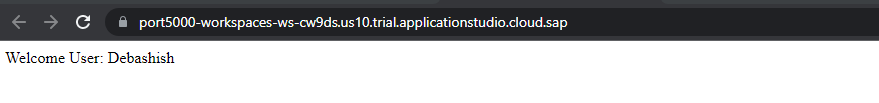
npm run start



Open another terminal and navigate to app folder.



Open in a New Tab



Our App is running locally

After making any changes server.js file, Restart the srv service.

