**Containerize NodeJS application and deploy to azure container instance using arm template**

**Step 1: -** 1.First install npm and node on your machine.

2. create NodeJS application**.**

**Step 2: -** Now create **dockerfile** for containerization.

FROM node:12-alpine3.14

WORKDIR /app

COPY package.json /app

RUN npm install && npm cache clean --force

COPY . /app

CMD node index.js

EXPOSE 80

**Step 3: -** Then create azure container registry arm template in your vs code.

File name = acrtemplate.json

{

    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",

    "contentVersion": "1.0.0.0",

    "parameters": {},

    "functions": [],

    "variables": {},

    "resources": [

        {

            "name": "nodejscontainerregistry",

            "type": "Microsoft.ContainerRegistry/registries",

            "apiVersion": "2019-05-01",

            "location": "[resourceGroup().location]",

            "sku": {

                "name": "Standard"

            },

            "properties": {

                "adminUserEnabled": true

            }

        }

    ],

    "outputs": {}

}

**Step 4**: - Create one more azure container instance template in vs code.

File name = conatianerinstance.json

{

  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

  "contentVersion": "1.0.0.0",

  "parameters": {

    "noCpu": {

      "defaultValue": "1",

      "type": "string"

    },

    "memoryInGB": {

      "defaultValue": "4",

      "type": "string"

    },

    "registryServerName": {

      "type": "string"

    },

    "registryUserName": {

      "type": "string"

    },

    "registryPassword": {

      "type": "string"

    },

    "image": {

      "type": "string",

      "defaultValue":"nodejscontainerregistry.azurecr.io/nodeapplication:latest"

    },

    "env\_ENVIRONMENT\_NAME": {

      "type": "string",

      "defaultValue":"Dev"

    }

  },

  "variables": {

    "containerInstanceName": "nodewitharmtemplateanddocker",

    "location": "[resourceGroup().location]",

    "instanceApiVersion": "2018-10-01"

  },

  "resources": [

    {

      "type": "Microsoft.ContainerInstance/containerGroups",

      "apiVersion": "[variables('instanceApiVersion')]",

      "name": "[variables('containerInstanceName')]",

      "location": "[variables('location')]",

      "properties": {

        "containers": [

          {

            "name": "[variables('containerInstanceName')]",

            "properties": {

              "image": "[parameters('image')]",

              "ports": [

                {

                  "port": 80

                }

              ],

              "environmentVariables": [

                {

                  "name": "ENVIRONMENT\_NAME",

                  "value": "[parameters('env\_ENVIRONMENT\_NAME')]"

                }

              ],

              "resources": {

                "requests": {

                  "memoryInGB": "[parameters('memoryInGB')]",

                  "cpu": "[parameters('noCpu')]"

                }

              }

            }

          }

        ],

        "imageRegistryCredentials": [

          {

            "server": "[parameters('registryServerName')]",

            "username": "[parameters('registryUserName')]",

            "password": "[parameters('registryPassword')]"

          }

        ],

        "ipAddress": {

          "ports": [

            {

              "protocol": "TCP",

              "port": 80

            }

          ],

          "type": "Public",

          "dnsNameLabel": "mynodeapp"

        },

        "osType": "Linux"

      }

    }

  ],

  "outputs": {

    "instance\_name": {

      "value": "[variables('containerInstanceName')]",

      "type": "string"

    }

  }

}

**Step 5**:- Now create parameter file

File name = param.ContainerInstance.json

{

    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",

    "contentVersion": "1.0.0.0",

    "parameters": {

        "noCpu": {

            "value": "1"

        },

        "memoryInGB": {

            "value": "4"

        },

        "registryServerName": {

            "value": "string"

        },

        "registryUserName": {

            "value": "string"

        },

        "registryPassword": {

            "value": "string"

        },

        "image": {

            "value": "string"

        }

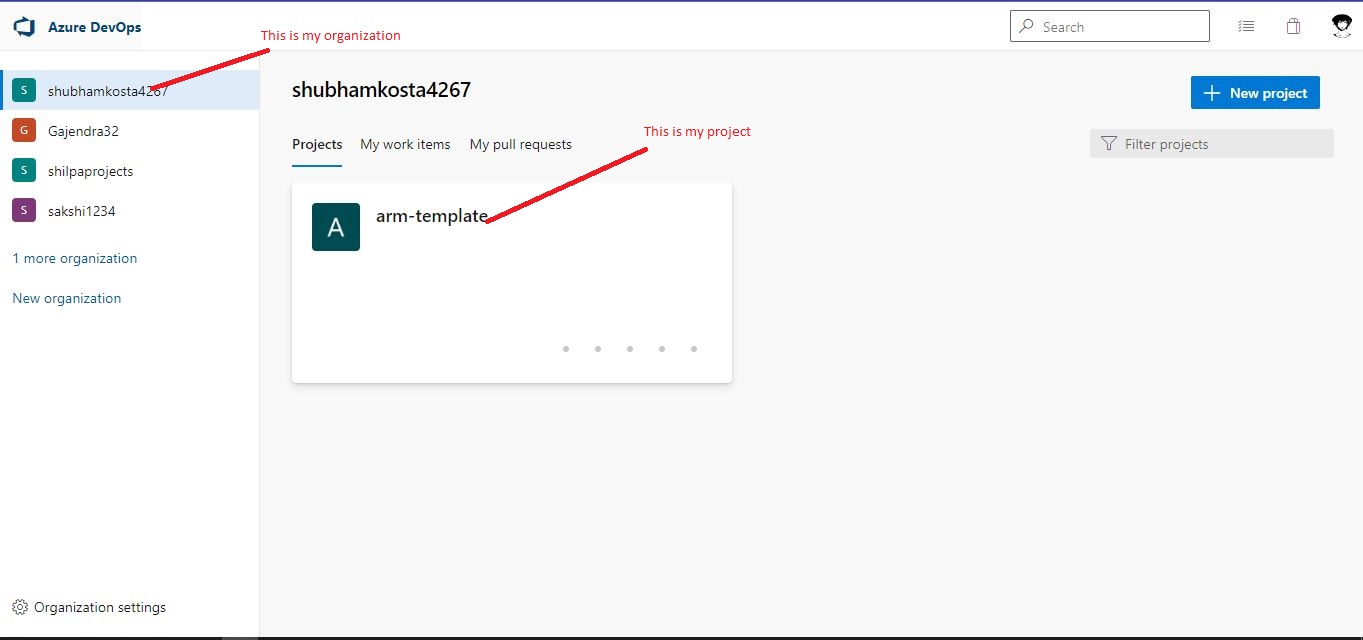
    }

}

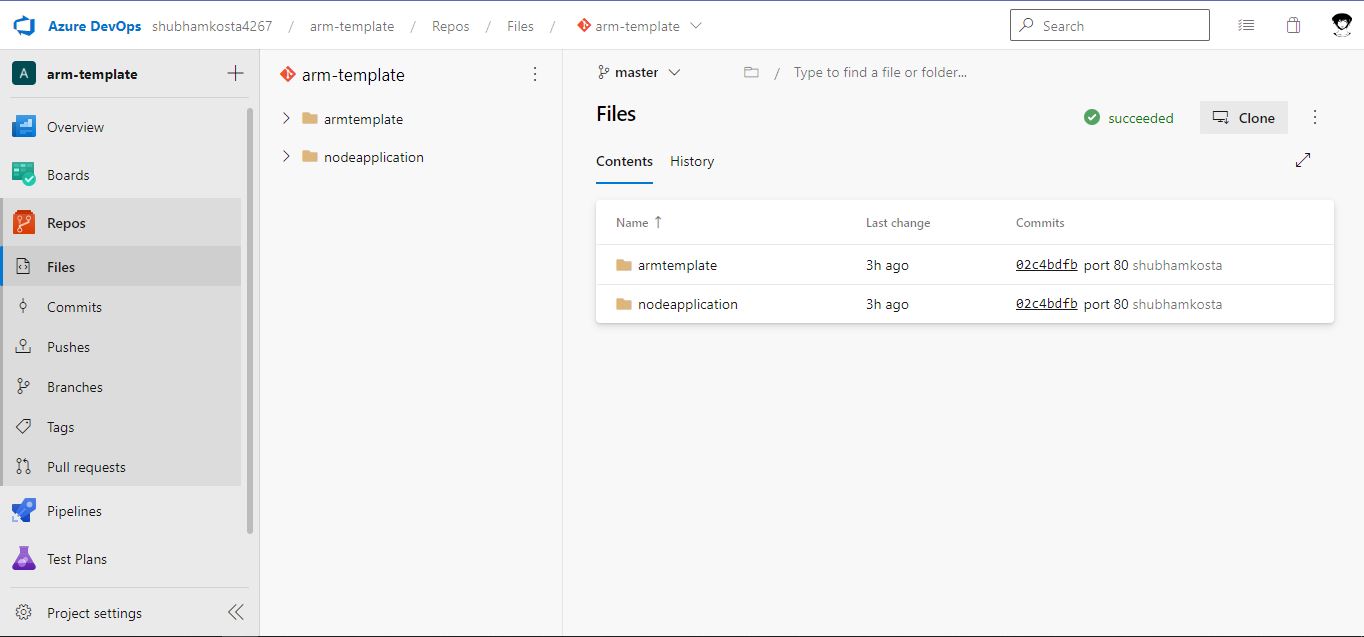
**Step 6: - 1. Now login into the azure devops account and go to your organization.**

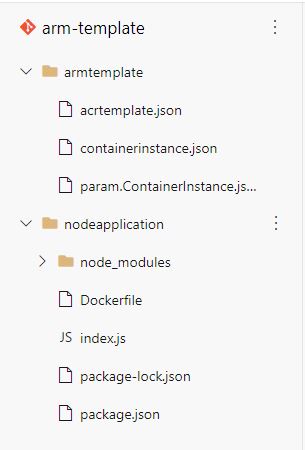
**2. create one project inside the organization.**

**3. Upload your all code and template here.**

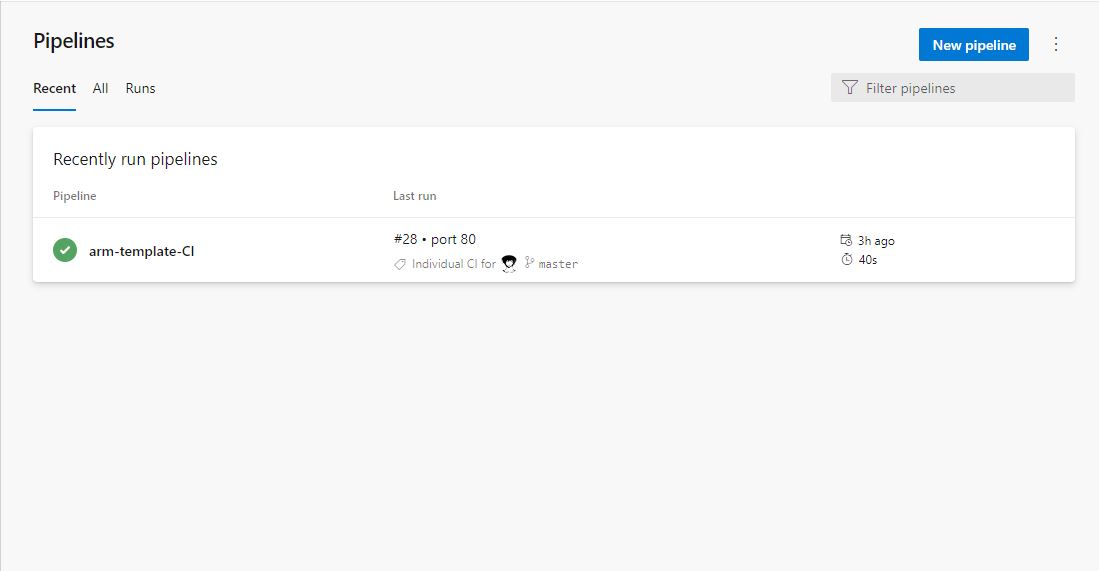
****

**Step 7:-** Push your code into azure repo

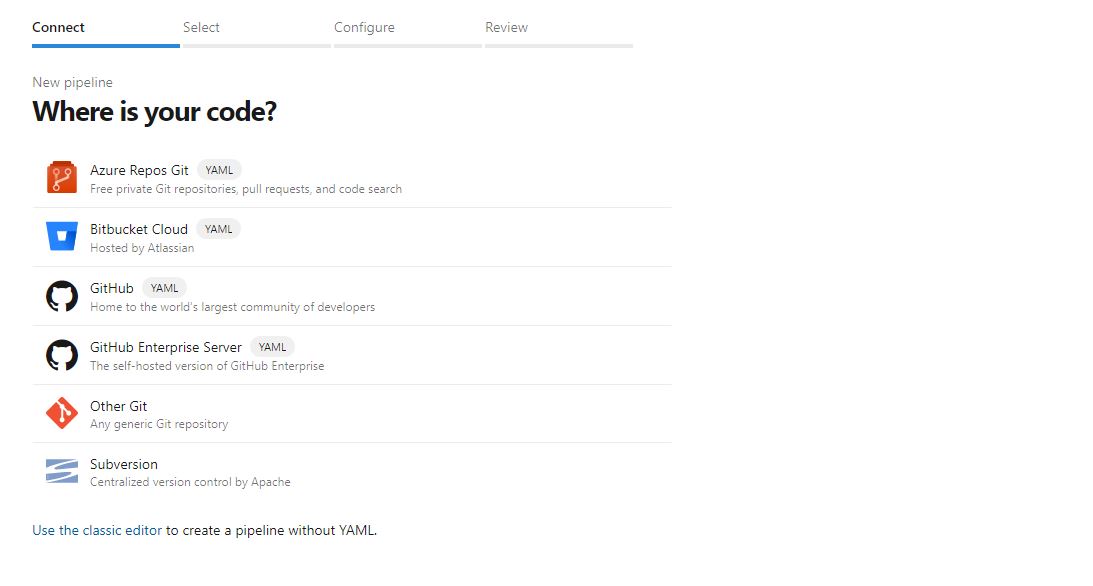


****

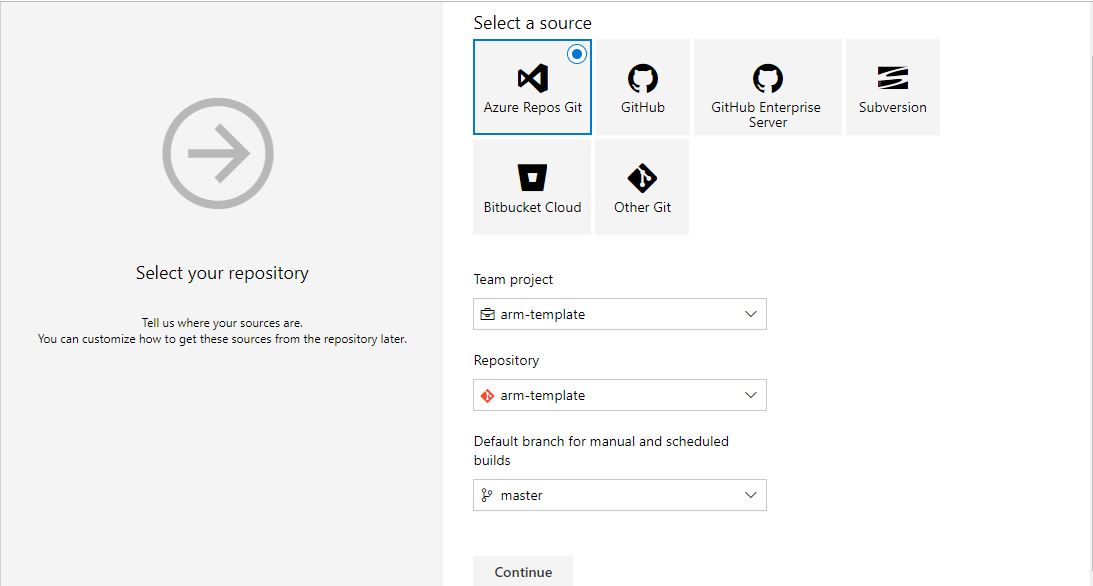
**Step 8: -** Now go to azure pipeline and create new pipeline

****

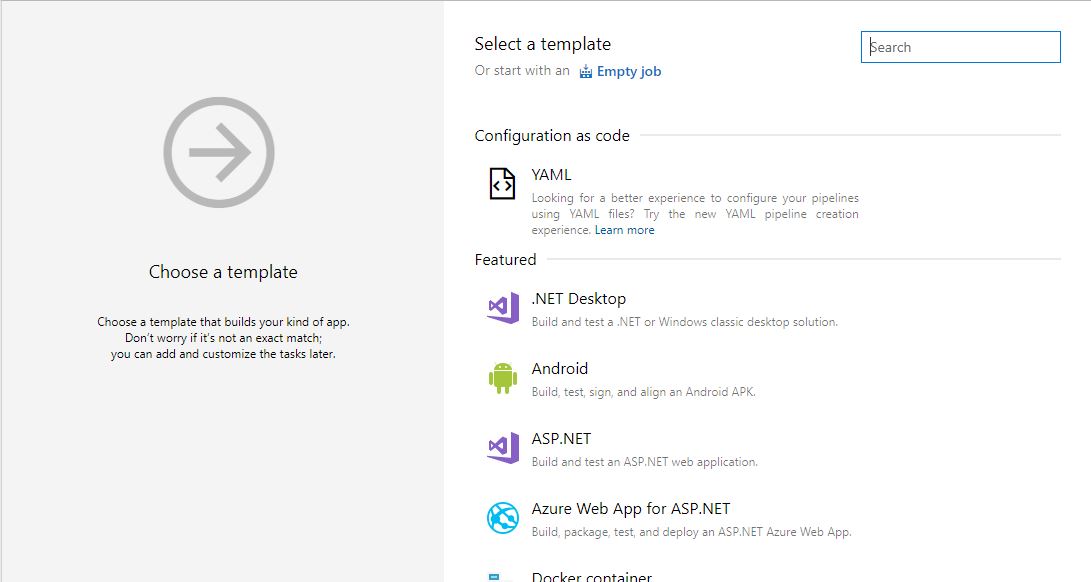
**Step 9: -** Select classic editor

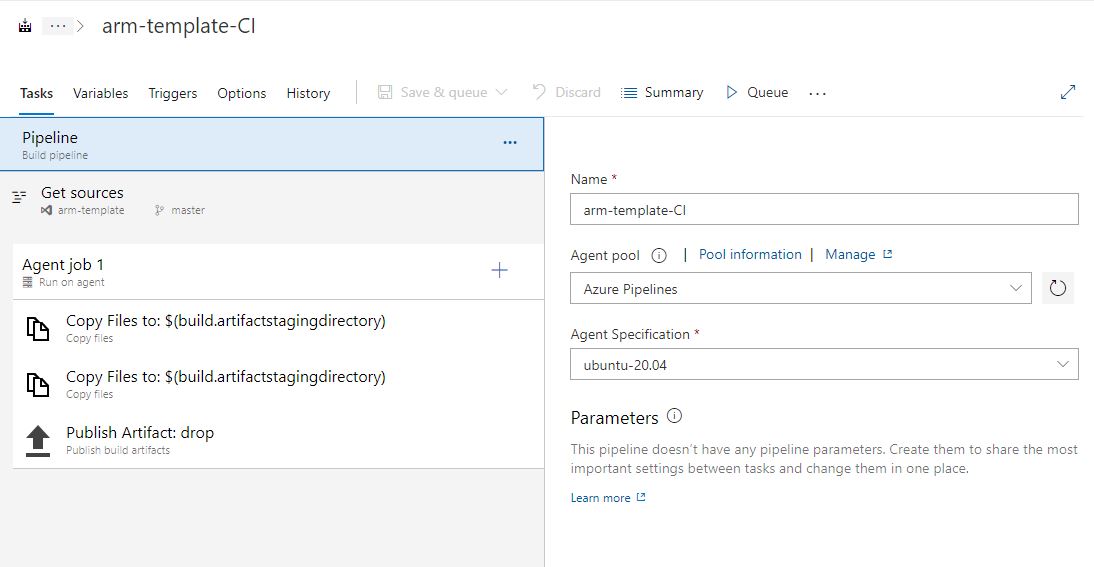
****

**Step 10:-** Select your azure repo and continue.

****

**Step 11:- Now select empty job**

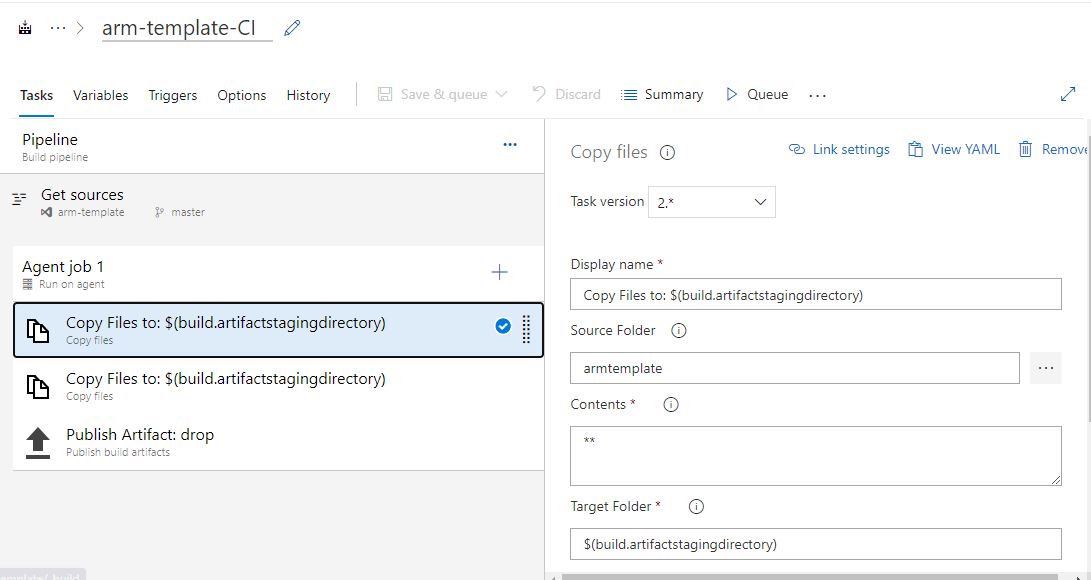
****

**Step 12:- Select pipeline then select your ubuntu machine  
**

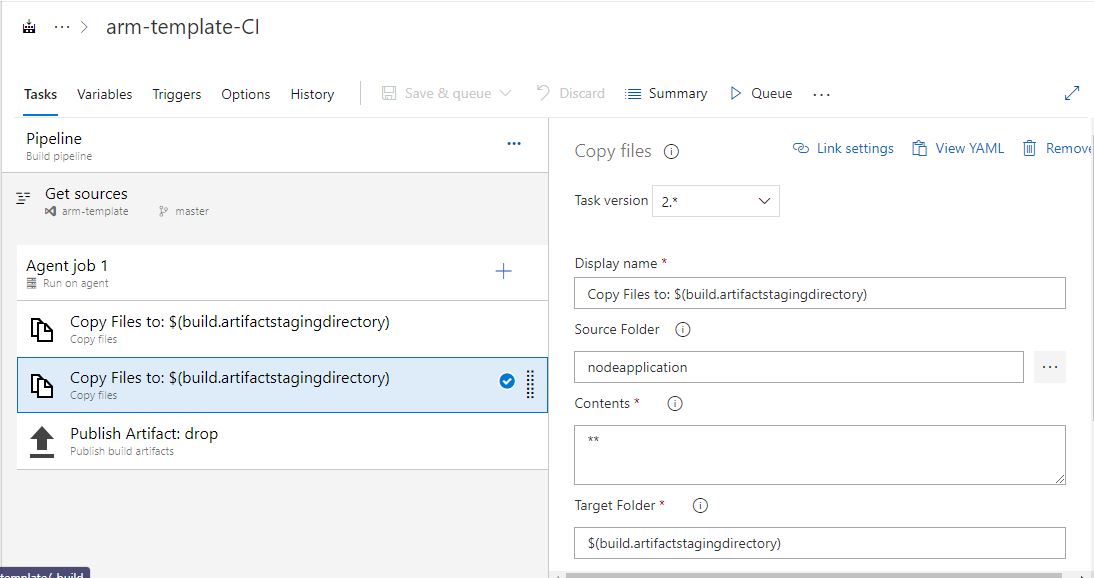
**Step 13**: - Now select agent and click on (+) mark

**Step 14**: - Search “copy files” and add 2 times

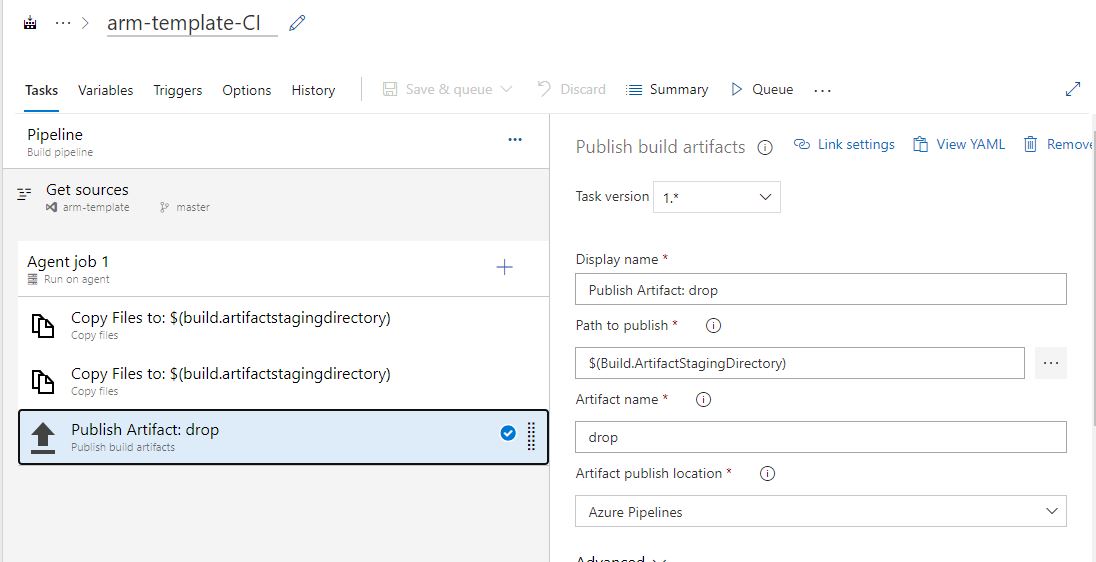
**File 1**



File 2

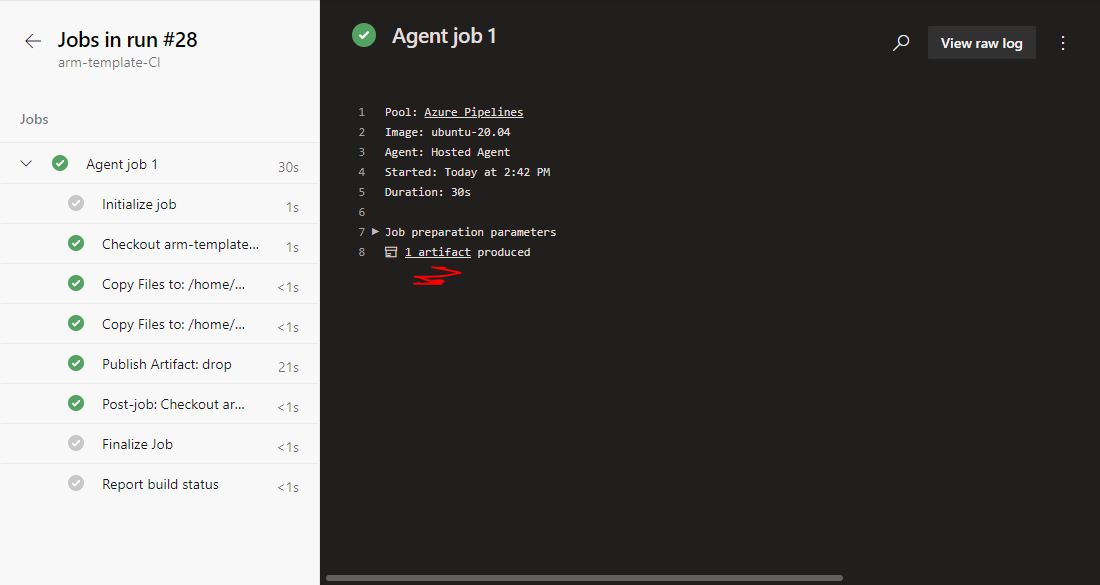


Step 15:- Now search publish build artifacts

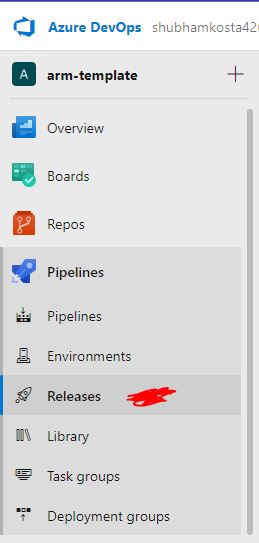


Now save all setting and run your pipeline……..

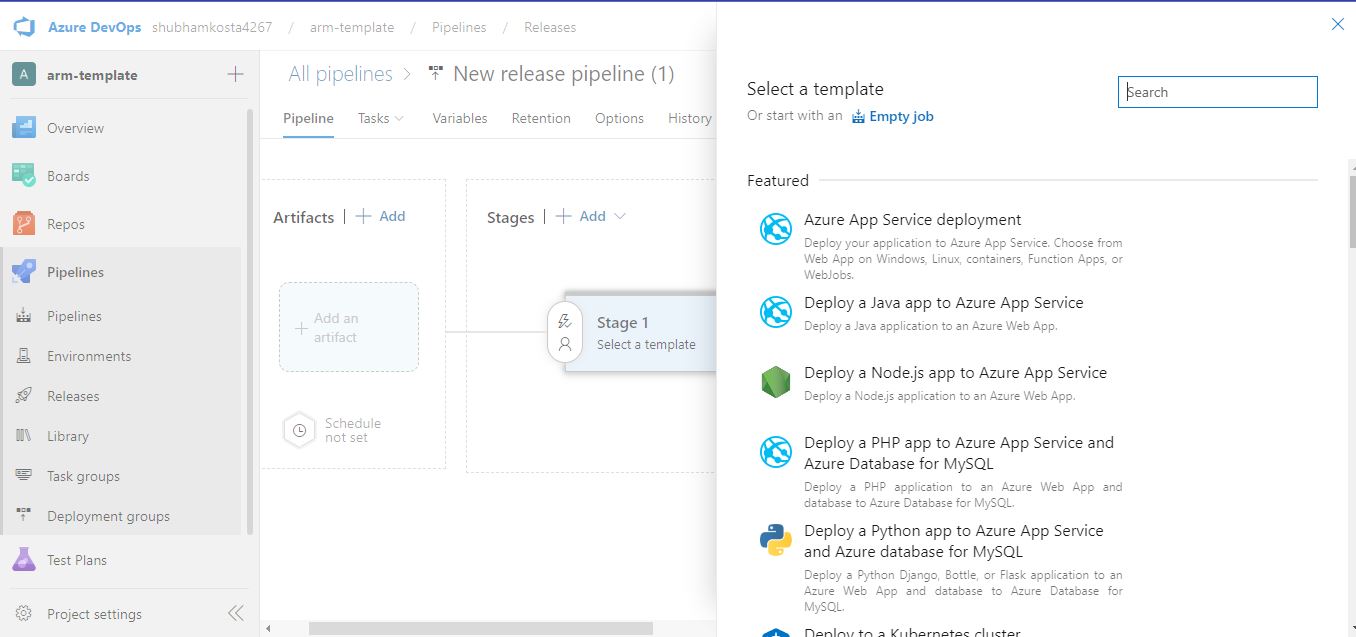
**Step 16**:- Now your project artifact created



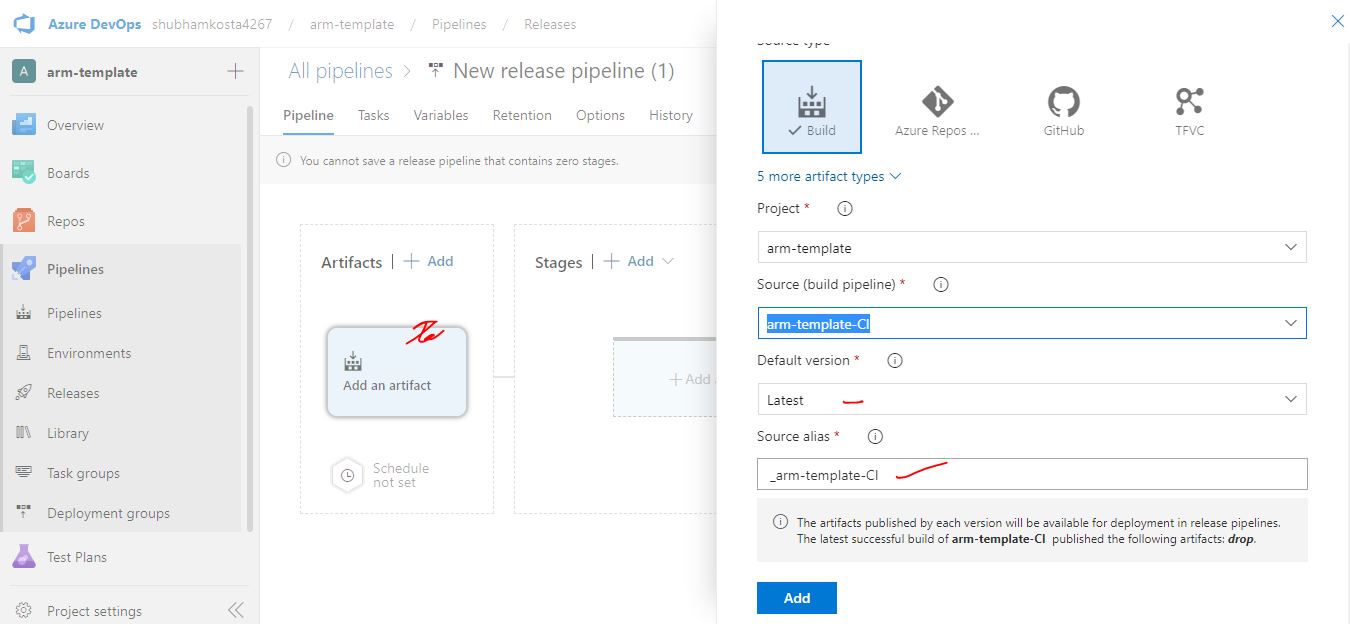
**Step 17**:- Click on release pipeline



**Step 18**:- Click on empty job



**Step 19**:- Now click on add artifact

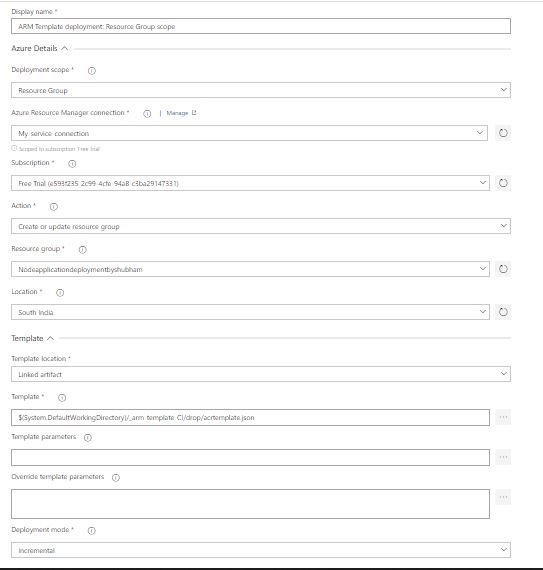


Select your artifacts repository and click on add ……..

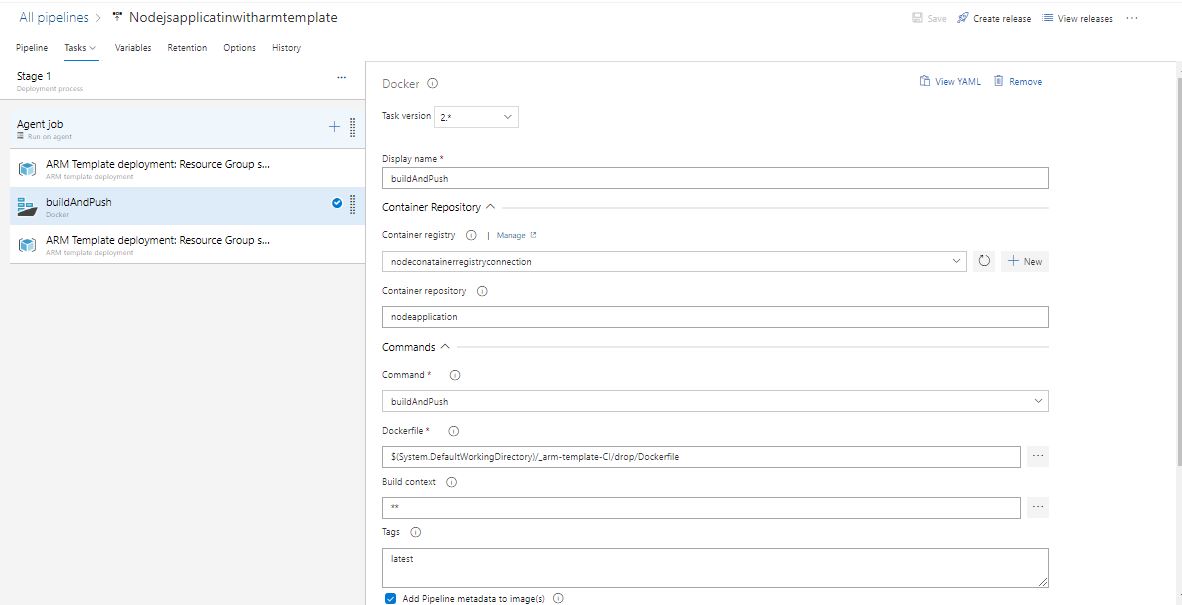
**Step 20**:- Now click on stages..

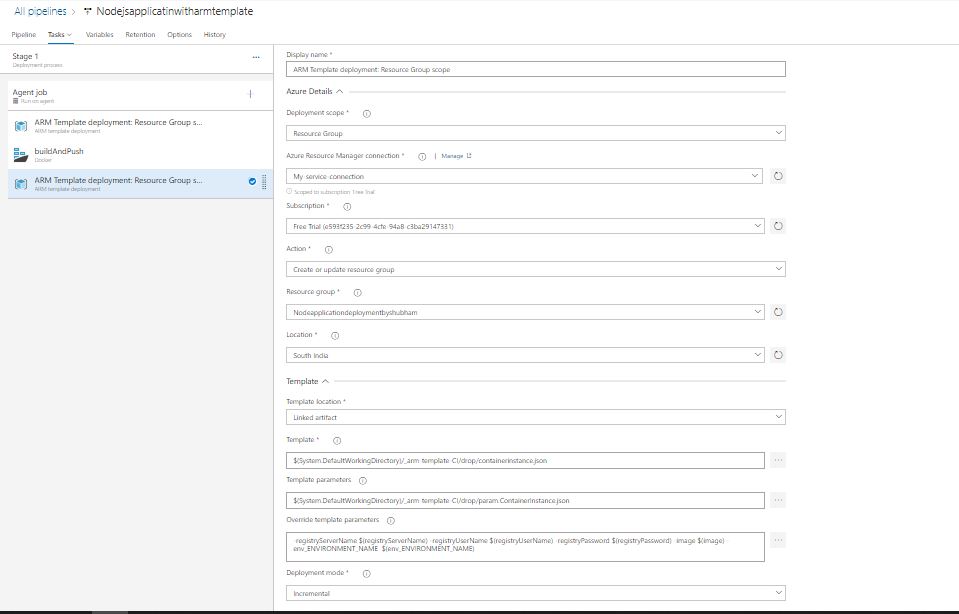
Then create pipeline stages.

**Step 21**:- Select your first arm template which created azure container registry..



**Step 22**:- Now search docker and click on it and insert your dockerfile here.

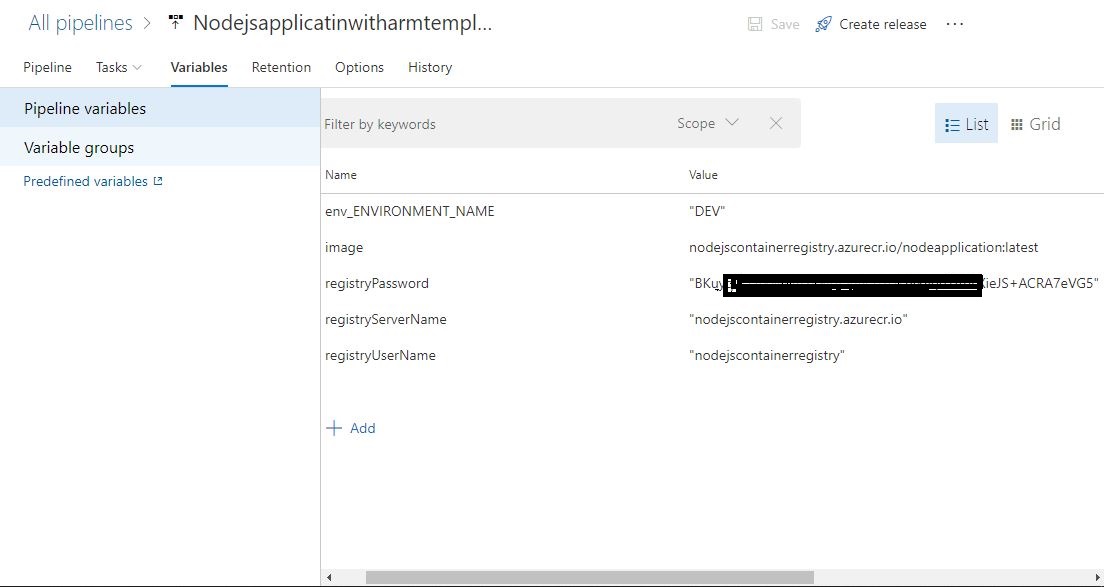


Step 23: - Now select your azure container instance template. 

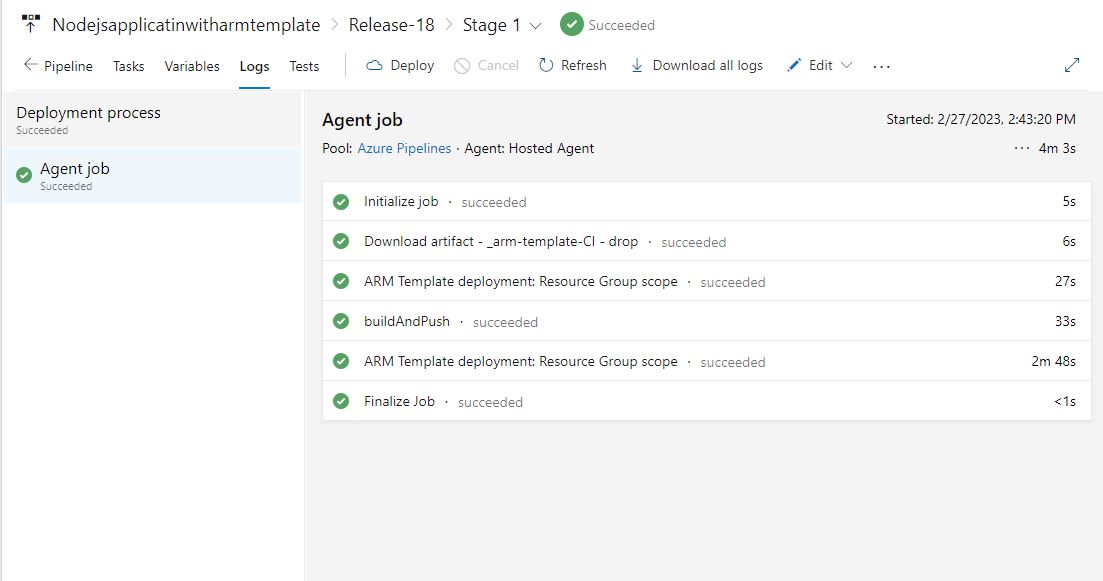
**Step 24**:- Add Override template parameters…

-registryServerName $(registryServerName) -registryUserName $(registryUserName) -registryPassword $(registryPassword) -image $(image) -env\_ENVIRONMENT\_NAME $(env\_ENVIRONMENT\_NAME)

**Step 25**:- Add variable to the azure release pipeline

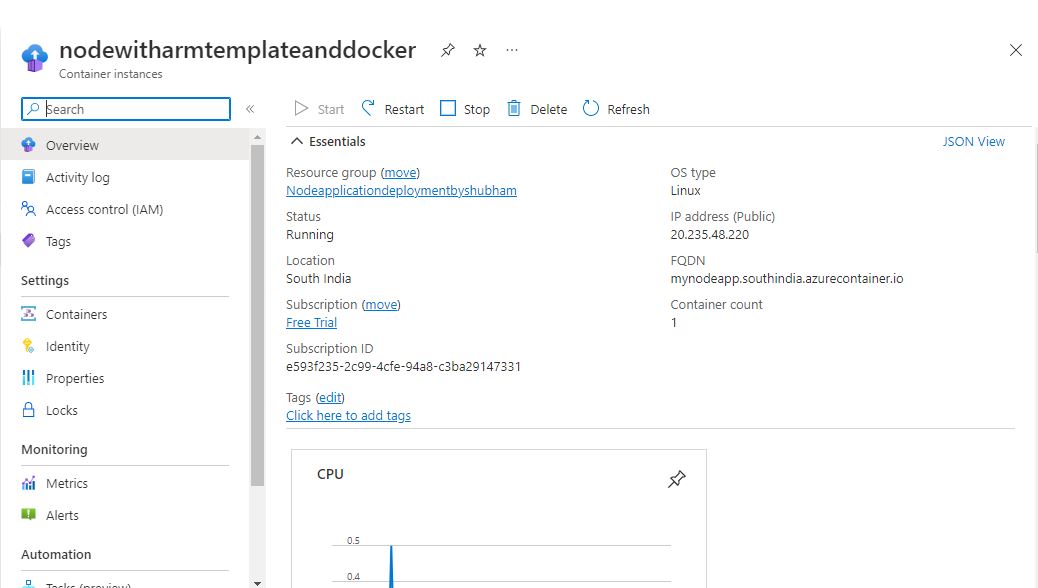


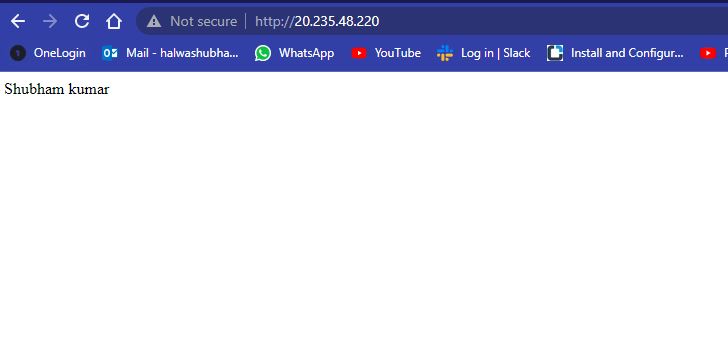
**Step 26**:- Now pipeline configure successfully, click on create release…



**Step 27**:- Now check your azure portal

And copy Ip address and paste into the browser…





**Thank you…**