Azure Devops task list YAML Schema reference for azure

ex:1

Pipeline is collection of jobs

Manual Steps:

- Get code from git
    - `git clone https://github.com/Aravindh-29/DbTestApp.git`
- build the docker image from docker file present in the folder
    - `cd DbTestApp/DbConnectionTester/`
    - `docker image build -t dbapp:latest .`
- Creating a pipeline for above task

```yaml
stages:
  - stage: ci
    displayName: building code
    jobs:
      - job: docker
        displayName: build image on Docker
        pool:
          vmImage: ubuntu-22.04
        steps:
          - task: Docker@2
            inputs:
              command: build
              Dockerfile: '**/Dockerfile'
```

Pipeline is collection of jobs:

```yaml
jobs: ci
  - job: build
    displayName: docker build
    pool:
      vmImage: ubuntu-22.04
    steps:
      - task: Docker@2
        inputs:
```

```
            command: build
            Dockerfile: '**/Dockerfile'
```

ex:2

Lets build a project [https://github.com/Aravindh-29/DbTestApp.git]

- Jobs:
  - Dotnet build
    - `dotnet build DbConnectionTester.sln`
  - Docker build
    - `docker compose build`
- Dotnet build Task:

- task: DotNetCoreCLI@2 inputs: command: 'build' projects: DbConnectionTester.sln
- task: DockerCompose@1 inputs: containerregistrytype: 'Azure Container Registry' dockerComposeFile: '**/docker-compose.yml' action: Build services

pipeline

```
jobs:
  - job: dotnetbuild
    displayName: Build using dotnet
    pool:
      vmImage: ubuntu-22.04
    steps:
      - task: DotNetCoreCLI@2
        inputs:
          command: 'build'
          projects: '**/DbConnectionTester.sln'

  - job: dockercompose
    displayName: Build using docker compose
    dependsOn: dotnetbuild
    pool:
      vmImage: ubuntu-22.04
    steps:
      - task: DockerCompose@0
        inputs:
          containerregistrytype: 'Azure Container Registry'
          dockerComposeFile: '**/docker-compose.yml'
          action: 'Build services'
          projectName: 'dbconnectiontester'
```

Pipeline is collection of stages

```yaml
stages:
  - stage: build
    displayName: Build Stage
    jobs:
      - job: dotnetbuild
        displayName: Building dotnet project
        pool:
          vmImage: ubuntu-22.04
        steps:
          - task: DotNetCoreCLI@2
            inputs:
              command: 'build'
              projects: '**/*.sln'   # Correct pattern: look for .sln file

  - stage: compose
    displayName: Compose Stage
    dependsOn: build    # stage dependsOn, not job
    jobs:
      - job: dockercompose
        displayName: Docker Compose
        pool:
          vmImage: ubuntu-22.04
        steps:
          - task: DockerCompose@1
            inputs:       # you missed this 'inputs' keyword
              containerregistrytype: 'Azure Container Registry'
              dockerComposeFile: '**/docker-compose.yml'
              action: 'Build services'   # correct wording
              projectName: 'dbconnectiontester'
```

Triggers

- This controls when the pipeline will be executed.
- pipeline execution is controlled by
  - trigger
  - pr(pull request) -> Branch merge request ichinappudu automatic pipeline run avuthundhi.
  - schedule -> specific time ki (like daily at 6AM) pipeline run cheyydam. [crontab.guru]
  - Manual - manual ga trigger cheyyadam.

```yaml
trigger:
  branches:
    include:
      - main

pr:
  branches:
    include:
      - main
```

```
schedules:
  - cron: "0 6 * * *"
    displayName: Daily 6AM Run
    branches:
      include:
        - main
    always: true
```

spring Petclicnic pipeline

- Lets make changes in the pipeline to trigger this pipeline
  - whenever any changes are done in main branch
  - On saturday at 1:00 Am.

```
trigger:
  - main
schedules:
  - cron: '30 1 * * 6'
    displayName: saturday at 1:30 Am
    branches:
      include:
        - main
parameters:
  - name: jdkVersion
    displayName: java version
    type: string
    default: 1.17
pool:
  vmImage: ubuntu-22.04

stages:
  - stage: build_stage
    displayName: Build Stage
    jobs:
      - job: build
        displayName: Maven build
        steps:
          - task: Maven@4
              inputs:
                mavenPOMFile: 'pom.xml'
                publishJUnitResults: true
                testResultsFiles: '**/surefire-reports/TEST-*.xml'
                javaHomeOption: 'JDKVersion'
                jdkVersionOption: "${{parametersjdkVersion}}"
```

Artifact Locations

- Artifact locations are used to store the output of the pipeline.

Variables

```
variables:
  DEFAULT_BRANCH: master
```

```
variables:
  DEFAULT_BRANCH: master

trigger:
  - "$(DEFAULT_BRANCH)"

schedules:
  - cron: '30 1 * * 6'
    displayName: saturday at 1:30 Am
    branches:
      include:
        - "$(DEFAULT_BRANCH)"
parameters:
  - name: jdkVersion
    displayName: java version
    type: string
    default: 1.17
pool:
  vmImage: ubuntu-22.04

stages:
  - stage: build_stage
    displayName: Build Stage
    jobs:
      - job: build
        displayName: Maven build
        steps:
          - task: Maven@4
              inputs:
                mavenPOMFile: 'pom.xml'
                publishJUnitResults: true
                testResultsFiles: '**/surefire-reports/TEST-*.xml'
                javaHomeOption: 'JDKVersion'
                jdkVersionOption: "${{parametersjdkVersion}}"
```

Variable Groups

- it is recomended to use variable groups
    - when variables have sensitive information

- too many variables
- to make sensitive variables secure, varibale groups can be linked to azure key vault

Templates

- Reusable pipelines or parts of pipelines are reffered as Templates.
- full template

```yaml
pool:
  vmImage: ubuntu-22.04
variables:
  DEFAULT_BRANCH: master
trigger:
  - "$(DEFAULT_BRANCH)"
schedules:
  - cron: '30 1 * * 6'
    branches:
      include:
        - "$(DEFAULT_BRANCH)"
parameters:
  - name: jdkversion
    type: string
    default: '1.17'
    values:
      - 1.17
      - 1.18
      - 1.8
      - 1.9

stages:
  - stage: javabuild
    displayName: javabuild
    jobs:
      - job: build
        displayName: Build Simple MVN App
        steps:
          - task: Maven@4
            inputs:
              mavenPOMFile: 'pom.xml'
              goals: 'clean package'
              publishJUnitResults: true
              testResultsFiles: '**/surefire-reports/TEST-*.xml'
              javaHomeOption: 'JDKVersion'
              jdkVersionOption: ${{parameters.jdkversion}}
```

- Deviding above pipeline

1. stages template [write parameters in this code]

```
parameters:
  - name: jdkversion
    type: string
    default: '1.17'
stages:
  - stage: javabuild
    displayName: javabuild
    jobs:
      - job: build
        displayName: Build Simple MVN App
        steps:
          - task: Maven@4
            inputs:
              mavenPOMFile: 'pom.xml'
              goals: 'clean package'
              publishJUnitResults: true
              testResultsFiles: '**/surefire-reports/TEST-*.xml'
              javaHomeOption: 'JDKVersion'
              jdkVersionOption: ${{parameters.jdkversion}}
```

2. azure pipeline template

```
pool:
  vmImage: ubuntu-22.04
variables:
  DEFAULT_BRANCH: master
trigger:
  - "$(DEFAULT_BRANCH)"
schedules:
  - cron: '30 1 * * 6'
    branches:
      include:
        - "$(DEFAULT_BRANCH)"
parameters:
  - name: jdkver
    type: string
    default: '1.17'
    values:
      - 1.17
      - 1.18
      - 1.19
resources:
 repositories:
   - repository: templates
     name: AzureDevopsTemplates
```

```
      ref: main
      type: git
  stages:
    - template: Maven/build-stage.yaml@templates
      parameters:
        jdkversion: "${{parameters.jdkver}}
```

Azure Service Connections

- Docker Connection to push images
- from project settings navigate to service connections inside create new service connection, select docker registry, provide your dockerhub id and password and enable for all pipelines and save.

```
stages:
  - stage:
    jobs:
      - job: docker
        steps:
          - task: Docker@2
            inputs:
              command: 'buildAndPush'
              Dockerfile: '**/Dockerfile'
              containerRegistry: Docker_Hub
              tags: '$(Build.BuildId)'
              repository: 'aravindh146/adomay27'
```

Deployments in AzureDevops

A Complete build and publish Activity

Here we are going to take an Application we build it and deploy it in System test Environment through Terraform.

Step1:

- Import your Application Repository to AzureRepos, Ensure you have the same application in a Folder.
- Open the Application in VS code and keep it Ready.

Step2:

- Initially We will write a pipeline for only to build the code.
- We will write a build stage now, Here i am going to use .Net Application.

```yaml
stages:
  - stage: Build
    jobs:
      - job: Build
        pool:
          vmImage: ubuntu-22.04
        steps:
          - task: DotNetCoreCLI@2
            inputs:
              command: 'build'
              projects: '**/*.csproj'
          - task: DotNetCoreCLI@2
            inputs:
              command: 'publish'
              publishWebProjects: false
              arguments: '-o $(Build.ArtifactStagingDirectory)/Output'
              zipAfterPublish: true
              modifyOutputPath: true
          - task: PublishBuildArtifacts@1
            inputs:
              pathToPublish: $(Build.ArtifactStagingDirectory)
              artifactName: DbAppOutputs
```

or

- Publish will build by default to produce outputs, so no need of including build task.

```yaml
stages:
  - stage: Build
    jobs:
      - job: Build
        pool:
          vmImage: ubuntu-22.04
        steps:
          - task: DotNetCoreCLI@2
            displayName: 'dotnet publish'
            inputs:
              command: 'publish'
              publishWebProjects: false
              arguments: '-o $(Build.ArtifactStagingDirectory)/Output'
              zipAfterPublish: true
              modifyOutputPath: true
          - task: PublishBuildArtifacts@1
            inputs:
              pathToPublish: $(Build.ArtifactStagingDirectory)
              artifactName: DbAppOutputs
```

☑ Understanding the Pipeline with Real-Time Example

🎯 Scenario:

You're building a **.NET Core web application** called `MyWebApp`, and your goal is to:

1. **Build** the app (compile the code)
2. **Package the build output** (DLLs, config files, etc.)
3. **Store the output (artifact)** so it can be used later for deployment

🧱 Step-by-Step Breakdown of the Pipeline Tasks

🔨 1. `DotNetCoreCLI@2` - **Build Task**

```
- task: DotNetCoreCLI@2
  inputs:
    command: 'build'
    projects: '**/*.csproj'
```

◇ **What it does:** This compiles your code into DLLs. It checks for syntax errors, restores dependencies, and produces a build in the `/bin` directory.

◇ **Real-life analogy:** Imagine you're preparing for a presentation. This is like writing and organizing your slides (but **not yet printing or packaging them**).

📦 2. `DotNetCoreCLI@2` - **Publish Task**

```
- task: DotNetCoreCLI@2
  inputs:
    command: 'publish'
    publishWebProjects: true
    arguments: '-o $(Build.ArtifactStagingDirectory)/Output'
    zipAfterPublish: true
```

◇ **What it does:** The **publish** command **prepares the build output** for deployment. It puts all the necessary files (DLLs, config files, wwwroot, etc.) into a single output folder.

It also **zips** the result into one file (if `zipAfterPublish: true`) so it's ready to be sent to a server.

◇ **Real-life analogy:** You're done creating the slides. Now you **export the final version as a PDF** to share with your manager. That "PDF" is like your **published output**.

📤 3. `PublishBuildArtifacts@1` - **Artifact Upload Task**

```
- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)/Output'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

◇ **What it does:** This task **uploads** the published files as an **artifact** in Azure DevOps. These artifacts can be **used in later stages** (like deploying to a server).

◇ **Real-life analogy:** You've created the PDF (published). Now you **upload it to Google Drive** (artifact storage) so your team or manager can access and use it.

Full Example in Real-Time Terms

Imagine you are:

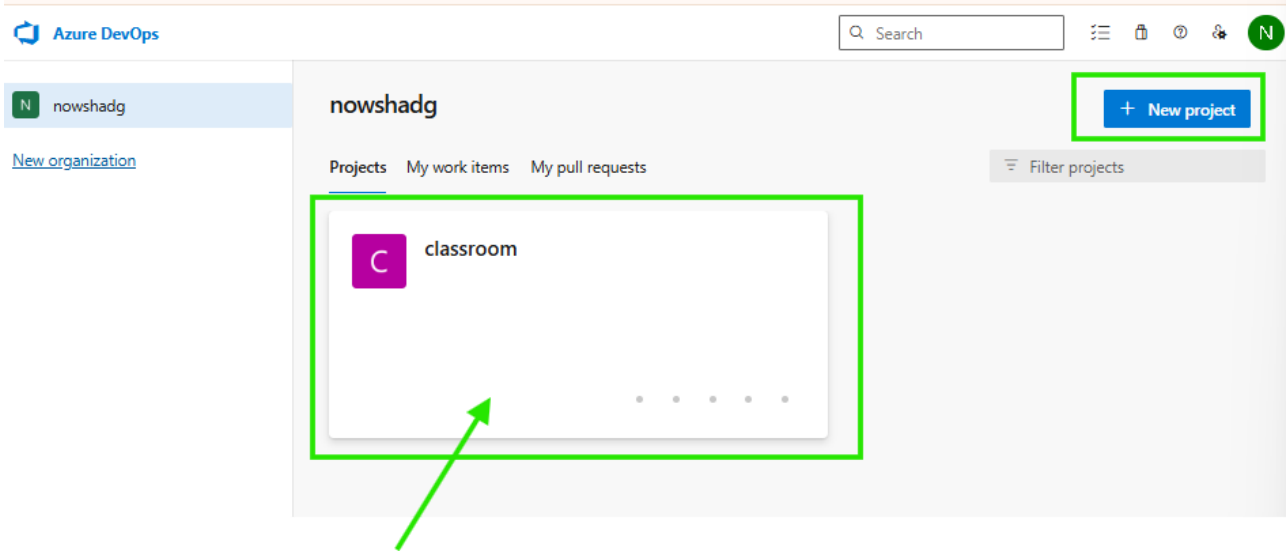
- Writing code = Development
- `build` step = Compile the code
- `publish` step = Create a deployable package
- `PublishBuildArtifacts` step = Upload it to cloud storage for your DevOps pipeline

Step3:

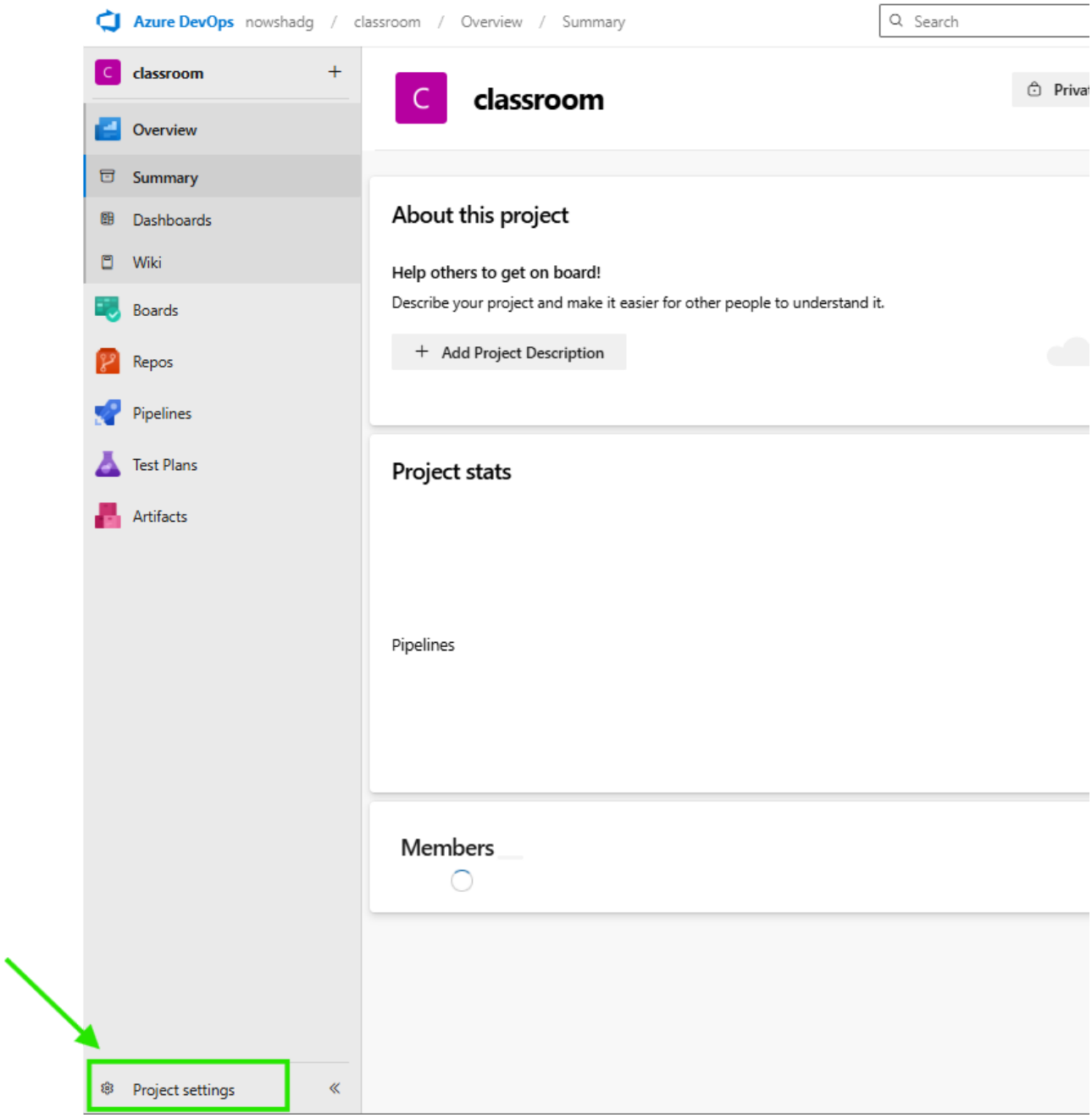
- As we are using Azure Devops free version we will get only one Microsoft hosted agent, for our requirement we need to create one self hosted agent with configuration on Azure CLI.

How to configure/Add Self Hosted Agent

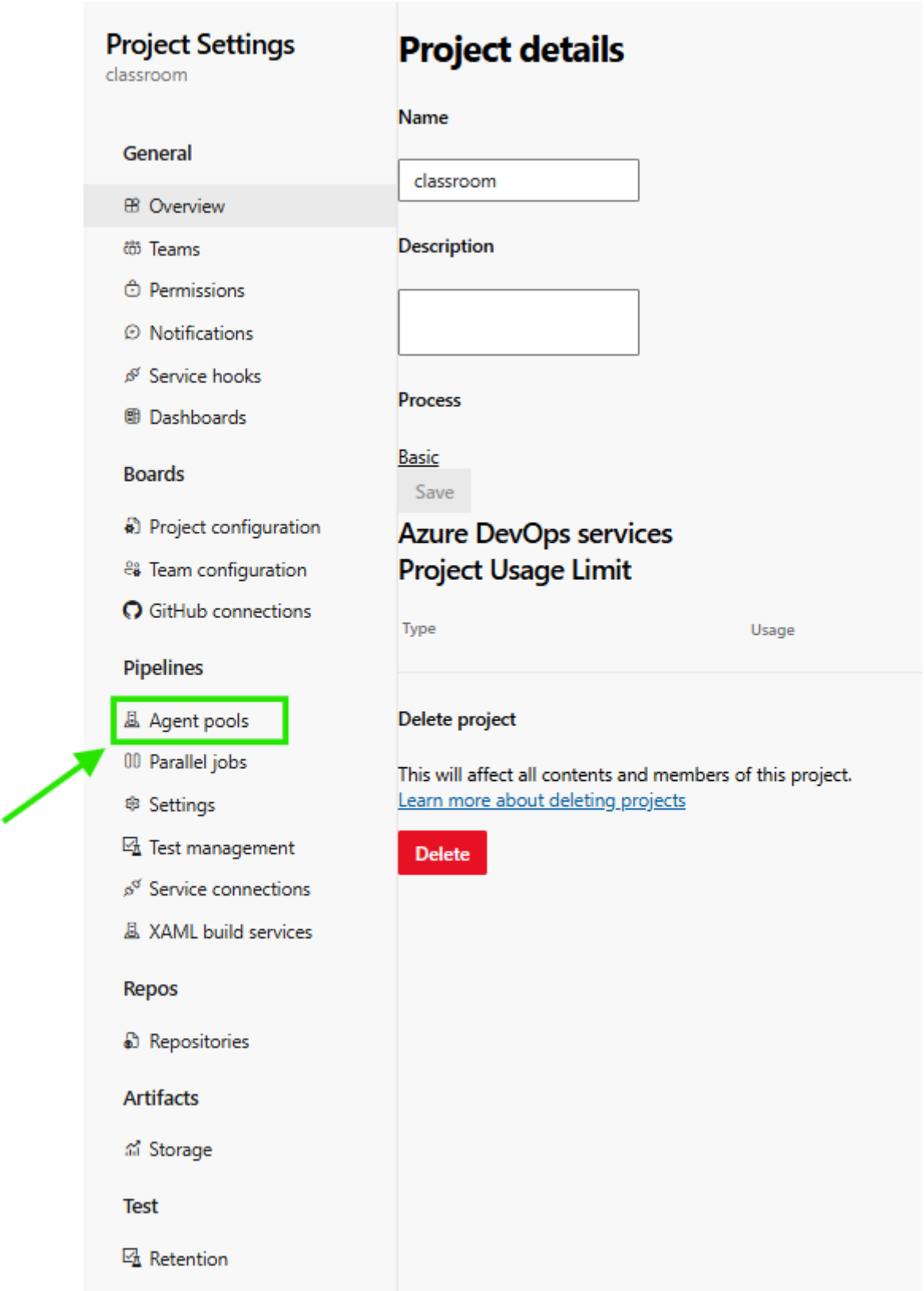
1. Naviagate to the Azure Devops open the project



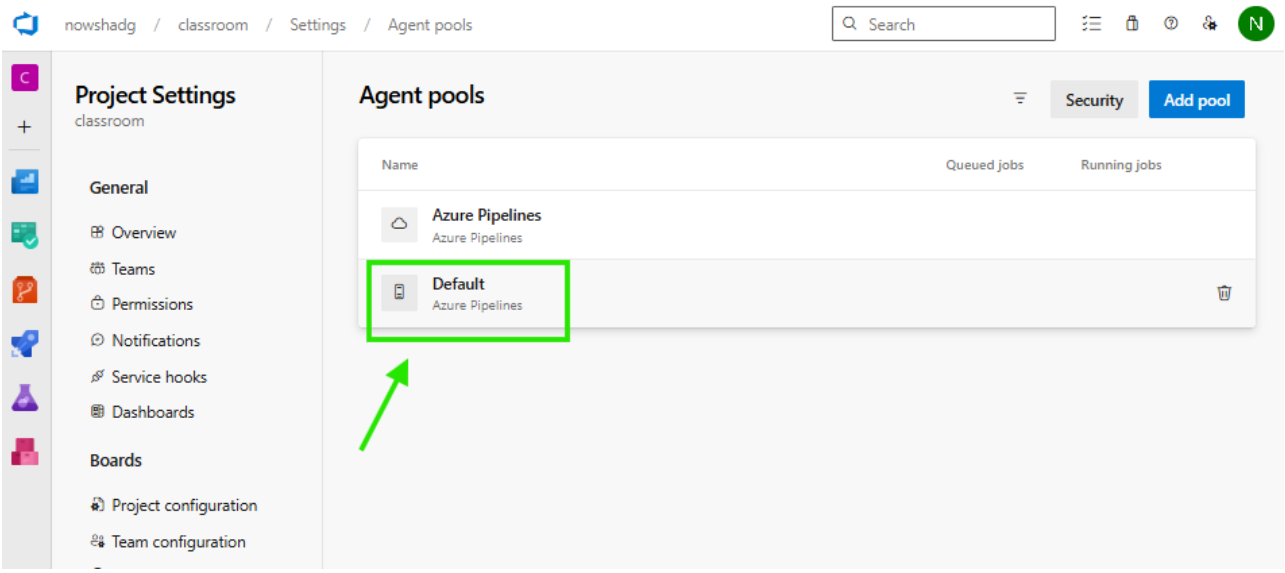
2. Now Open Project Settings



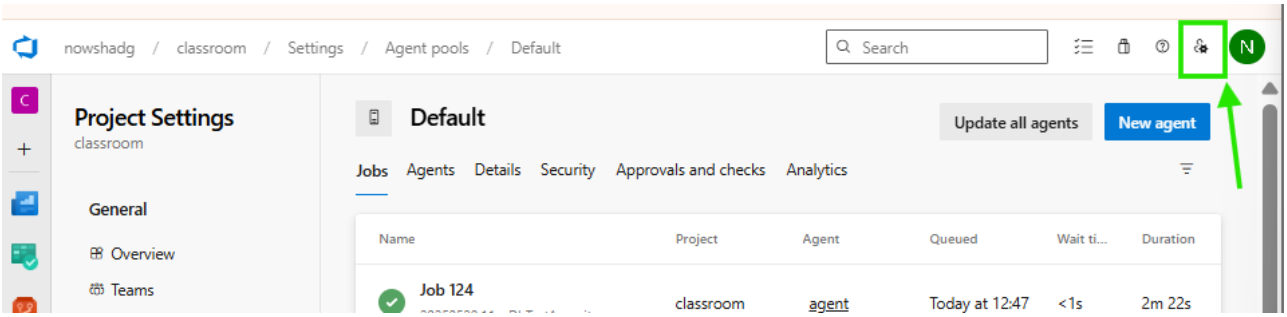
3. Select Agent pool



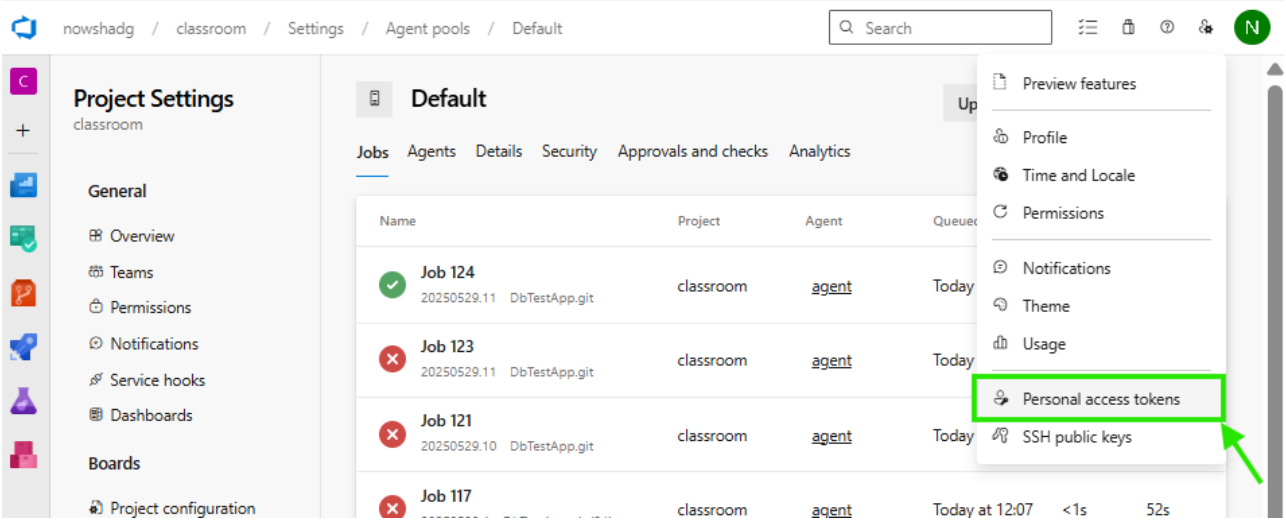
4. Select Default



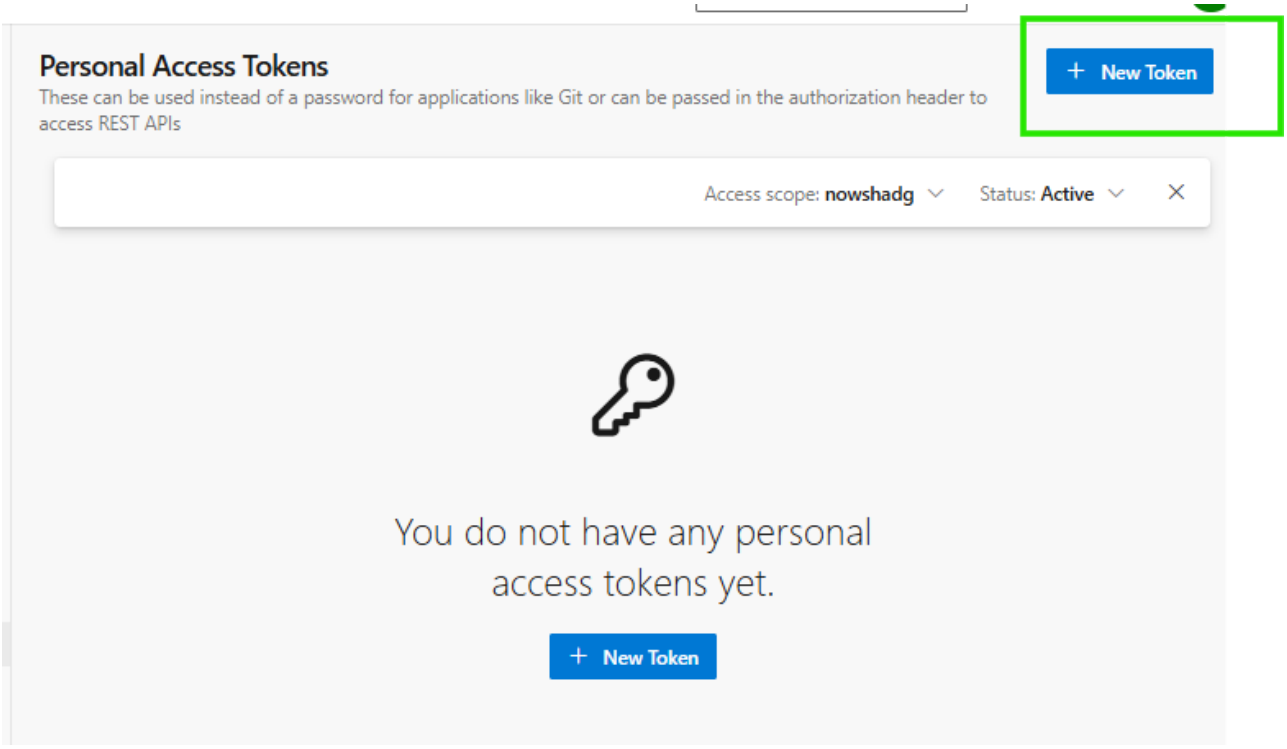
5. Select UserSettings



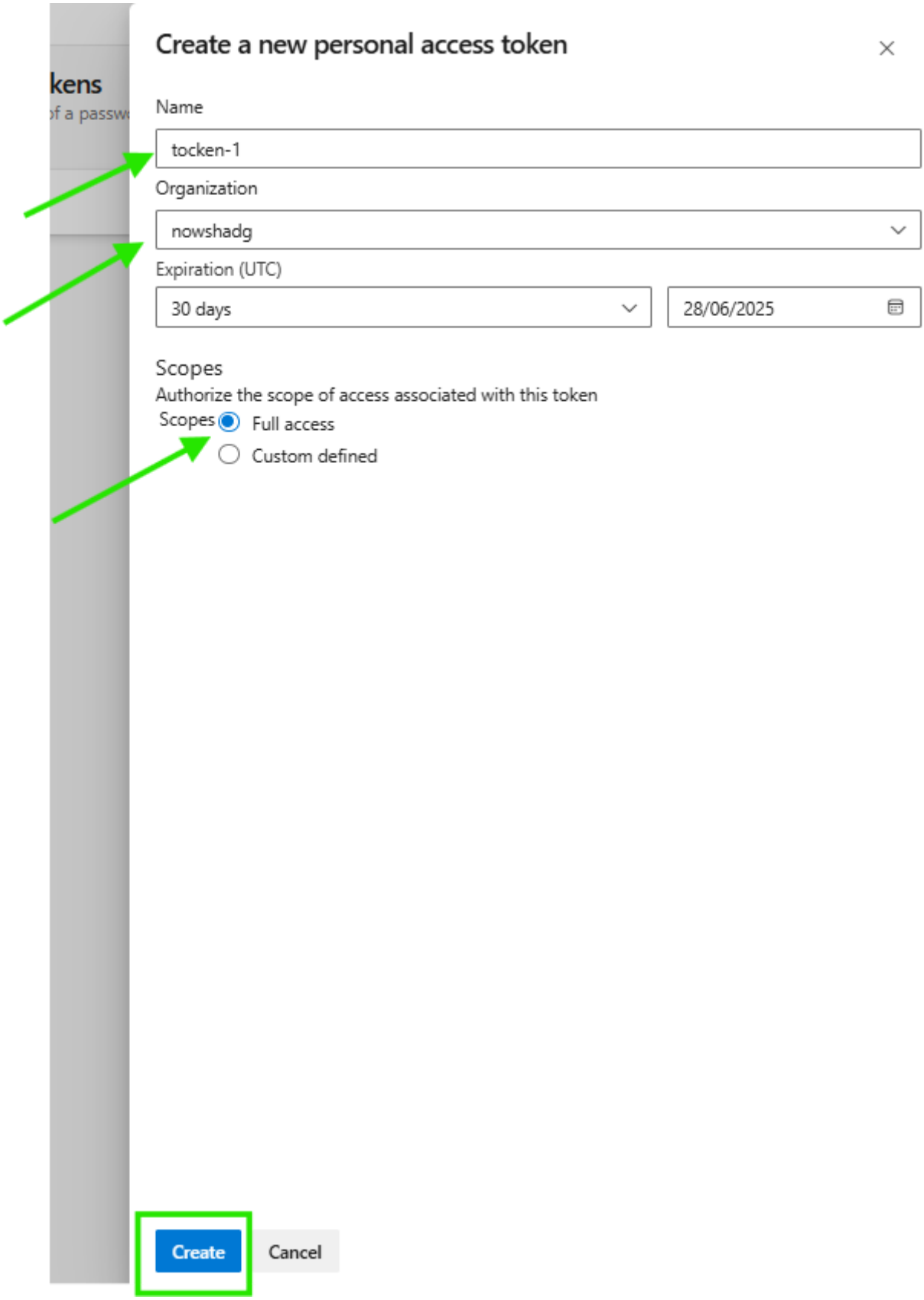
6. Select Personal Access tockens



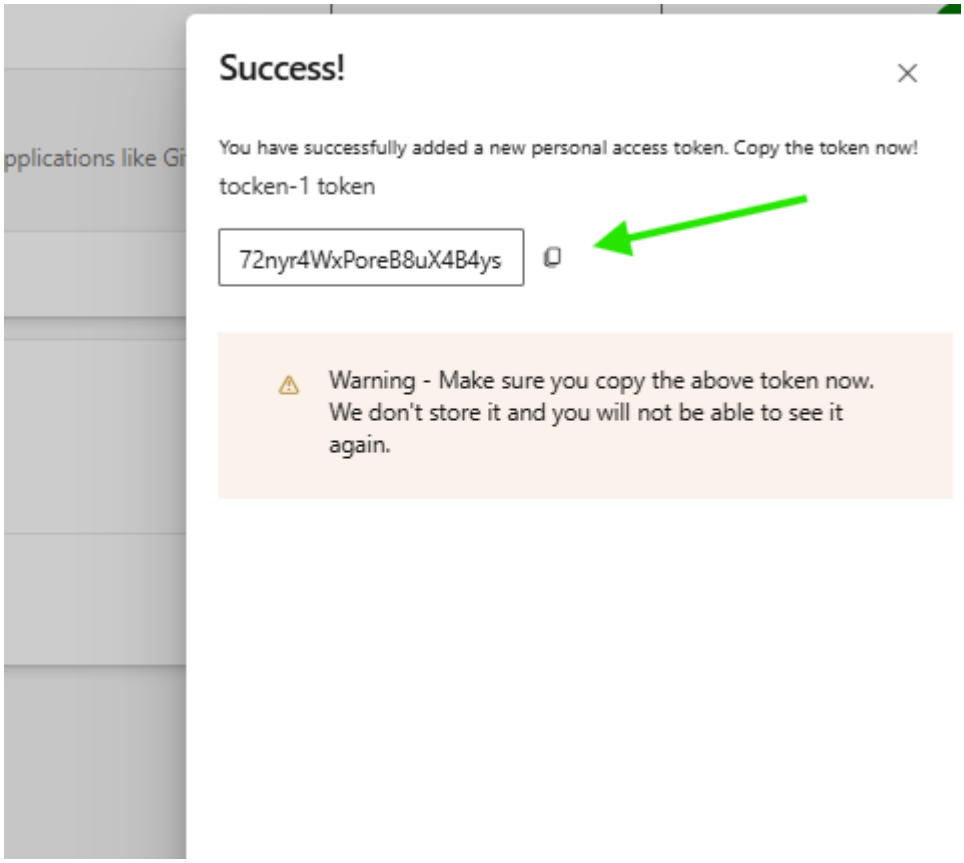
7.Create New


Personal Access Tokens
These can be used instead of a password for applications like Git or can be passed in the authorization header to access REST APIs
+ New Token
Access scope: nowshadg
Status: Active
You do not have any personal access tokens yet.
+ New Token

7. Fill the details and create



8. Copy the tocken it will show only once copy and save it in notepad.



9. Now create a Ubuntu Server having more RAM/Cpu's for fast execution, either in Azure or AWS.

10. Connect from your powershell by ssh, and install Azure CLI or AWS CLI based on where you want to create infra to deploy the Application.

Installing Azure CLI

- Open the linkclick here and follow Option 1 for easiest installation

```
Aravindh@agent:~$ curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
Hit:1 http://azure.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB
]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-backports InRelease [126
kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-security InRelease [126 k
B]
Get:5 http://azure.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [
15.0 MB]
Get:6 http://azure.archive.ubuntu.com/ubuntu noble/universe Translation-en [
5982 kB]
Get:7 http://azure.archi                                  .e/universe amd64 Components
 [3871 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Meta
data [301 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages
 [269 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu noble/multiverse Translation-e
n [118 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu noble/multiverse amd64 Compone
```

```
Aravindh@agent:~$ az version
{
  "azure-cli": "2.73.0",
  "azure-cli-core": "2.73.0",
  "azure-cli-telemetry": "1.1.0",
  "extensions": {}
}
Aravindh@agent:~$
```

```
Aravindh@agent:~$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicel
ogin and enter the code E5GV6LZJK to authenticate.
```

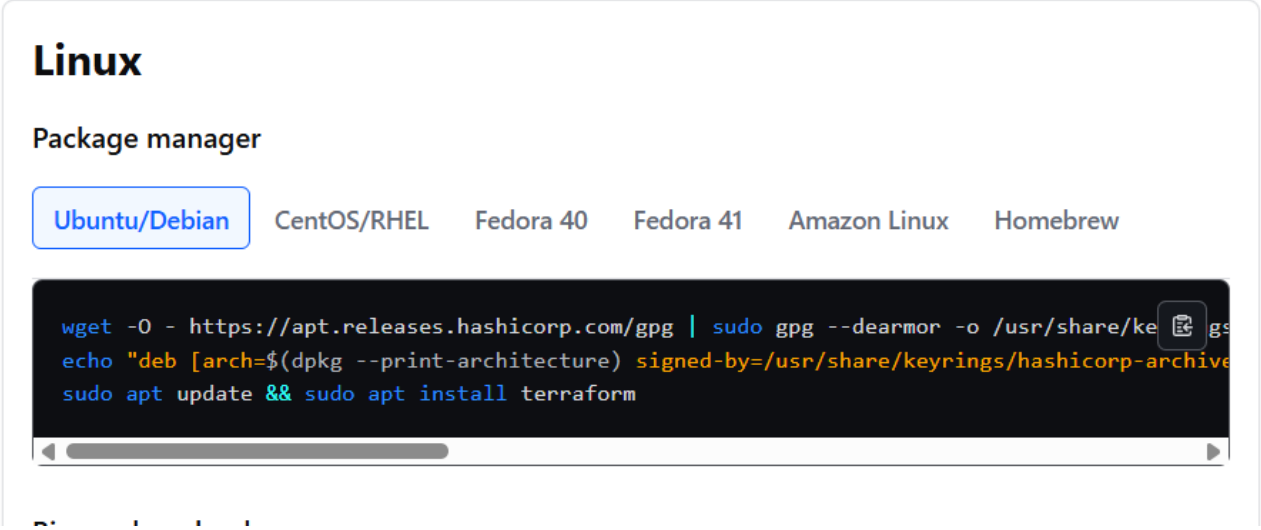
https://microsoft.com/devicelogin
Ctrl+Click to follow link

From below image shows i have loged in and get resource groups

```
Aravindh@agent:~$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicel
ogin and enter the code E5GV6LZJK to authenticate.

Retrieving tenants and subscriptions for the selection...

[Tenant and subscription selection]

No     Subscription name     Subscription ID                       Tenant
-----  --------------------  ------------------------------------  ---------
--------
[1] *  Azure subscription 1  89fac5fe-1629-40ec-8898-945fdce7bdc4  Default D
irectory

The default is marked with an *; the default tenant is 'Default Directory' a
nd subscription is 'Azure subscription 1' (89fac5fe-1629-40ec-8898-945fdce7b
dc4).

Select a subscription and tenant (Type a number or Enter for no changes): 1

Tenant: Default Directory
Subscription: Azure subscription 1 (89fac5fe-1629-40ec-8898-945fdce7bdc4)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you
 want to use more easily. Learn more about it and its configuration at https
://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibu
g

[Warning] The login output has been updated. Please be aware that it no long
er displays the full list of available subscriptions by default.

Aravindh@agent:~$ az group list -o table
Name              Location    Status
----------------  ----------  ---------
SSH_Key           eastus      Succeeded
Temp              eastus      Succeeded
NetworkWatcherRG  eastus      Succeeded
Aravindh@agent:~$
```

Installing Terraform

- Open the link click here and copy the link and paste and run. this will install Terraform in your machine.



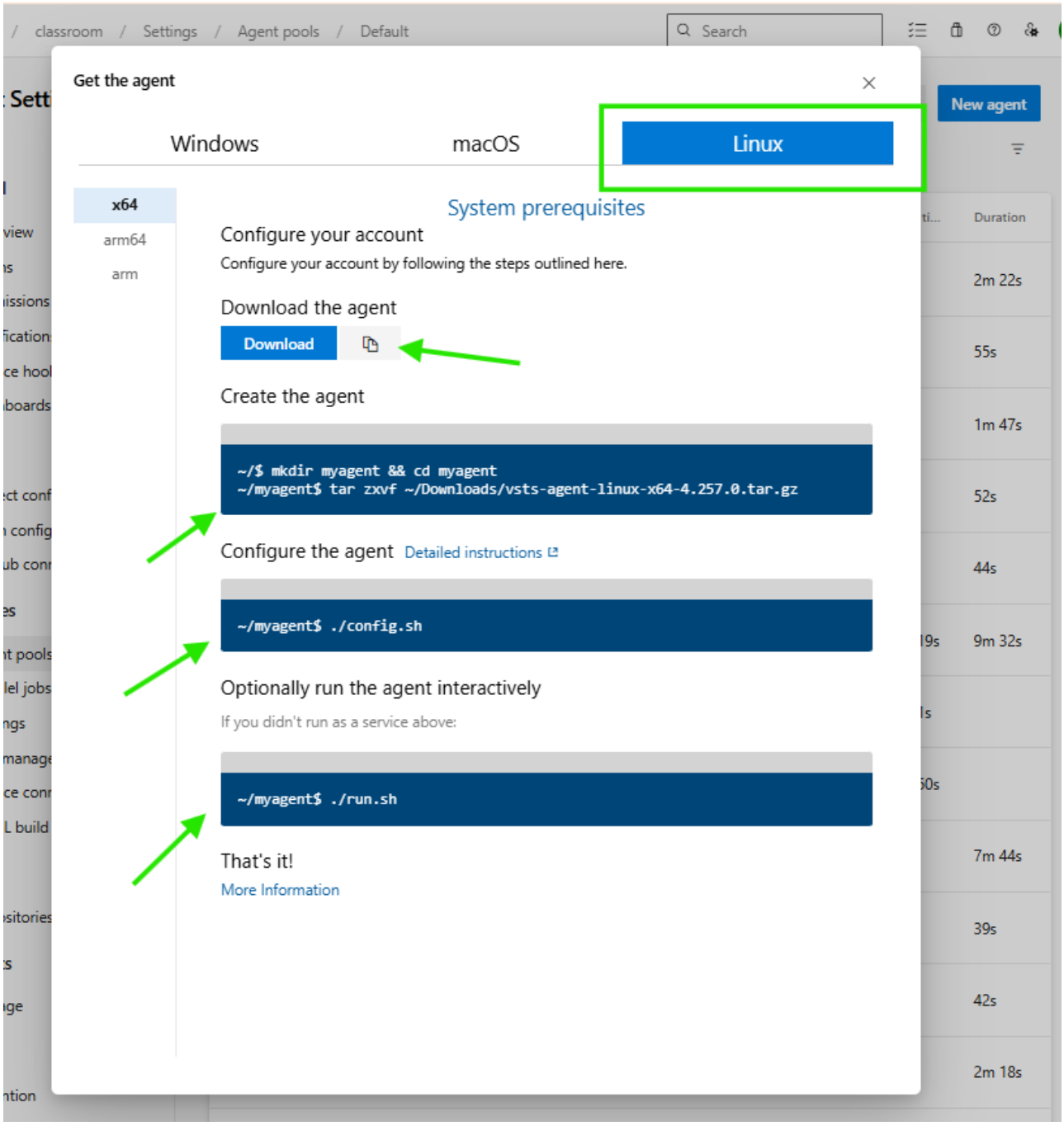

```
Aravindh@agent:~$ wget -O - https://apt.releases.hashicorp.com/gpg | sudo gp
g --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/h
ashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(grep -oP
'(?<=UBUNTU_CODENAME=).*' /etc/os-release || lsb_release -cs) main" | sudo t
ee /etc/apt/sources.list.d/hashicorp.list
sudo apt update && sudo apt install terraform
--2025-05-29 16:29:37--  https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 18.160.
10.69, 18.160.10.45, 18.160.10.71, ...
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.160
.10.69|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

-                   100%[================>]   3.89K  --.-KB/s    in 0s

2025-05-29 16:29:37 (1.23 GB/s) - written to stdout [3980/3980]

deb [arch=amd64 signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
 https://apt.releases.hashicorp.com noble main
Hit:1 http://azure.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu noble-security InRelease
```

```
Aravindh@agent:~$ terraform version
Terraform v1.12.1
on linux_amd64
Aravindh@agent:~$
```

Configuring as Self Hosted Agent

- Open AzureDevops > Project Settings > Agent pool > Default > create new.

- When you select New below image shows

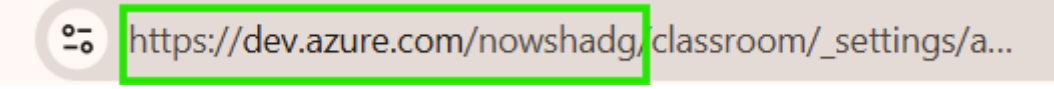


- copy the agent first and use `wget` and download it.

```
Aravindh@agent:~$ wget https://download.agent.dev.azure.com/agent/4.257.0/vs
ts-agent-linux-x64-4.257.0.tar.gz
--2025-05-29 16:37:24--  https://download.agent.dev.azure.com/agent/4.257.0/
vsts-agent-linux-x64-4.257.0.tar.gz
Resolving download.agent.dev.azure.com (download.agent.dev.azure.com)... 23.
215.0.202, 23.215.0.207, 2600:1408:c400:5::17c7:370f, ...
Connecting to download.agent.dev.azure.com (download.agent.dev.azure.com)|23
.215.0.202|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 147578415 (141M) [application/octet-stream]
Saving to: ‘vsts-agent-linux-x64-4.257.0.tar.gz’

vsts-agent-linux-x 100%[================>] 140.74M   617MB/s    in 0.2s

2025-05-29 16:37:25 (617 MB/s) - ‘vsts-agent-linux-x64-4.257.0.tar.gz’ saved
 [147578415/147578415]
```

```
Aravindh@agent:~$ ls
vsts-agent-linux-x64-4.257.0.tar.gz
Aravindh@agent:~$
Aravindh@agent:~$  mkdir myagent && cd myagent
Aravindh@agent:~/myagent$ myagent$ tar zxvf ~/Downloads/vsts-agent-linux-x64
-4.257.0.tar.gz
myagent$: command not found
Aravindh@agent:~/myagent$ ls
Aravindh@agent:~/myagent$ cd ..
Aravindh@agent:~$ ls
myagent  vsts-agent-linux-x64-4.257.0.tar.gz
Aravindh@agent:~$ sudo mv vsts-agent-linux-x64-4.257.0.tar.gz myagent/
Aravindh@agent:~$ ls
myagent
Aravindh@agent:~$ cd myagent/
Aravindh@agent:~/myagent$ ls
vsts-agent-linux-x64-4.257.0.tar.gz
Aravindh@agent:~/myagent$ tar xvzf vsts-agent-linux-x64-4.257.0.tar.gz
./
./env.sh
./license.html
./run-docker.sh
./config.sh
./externals/
./externals/node16/
./externals/node16/LICENSE
./externals/node16/include/
./externals/node16/include/node/
```

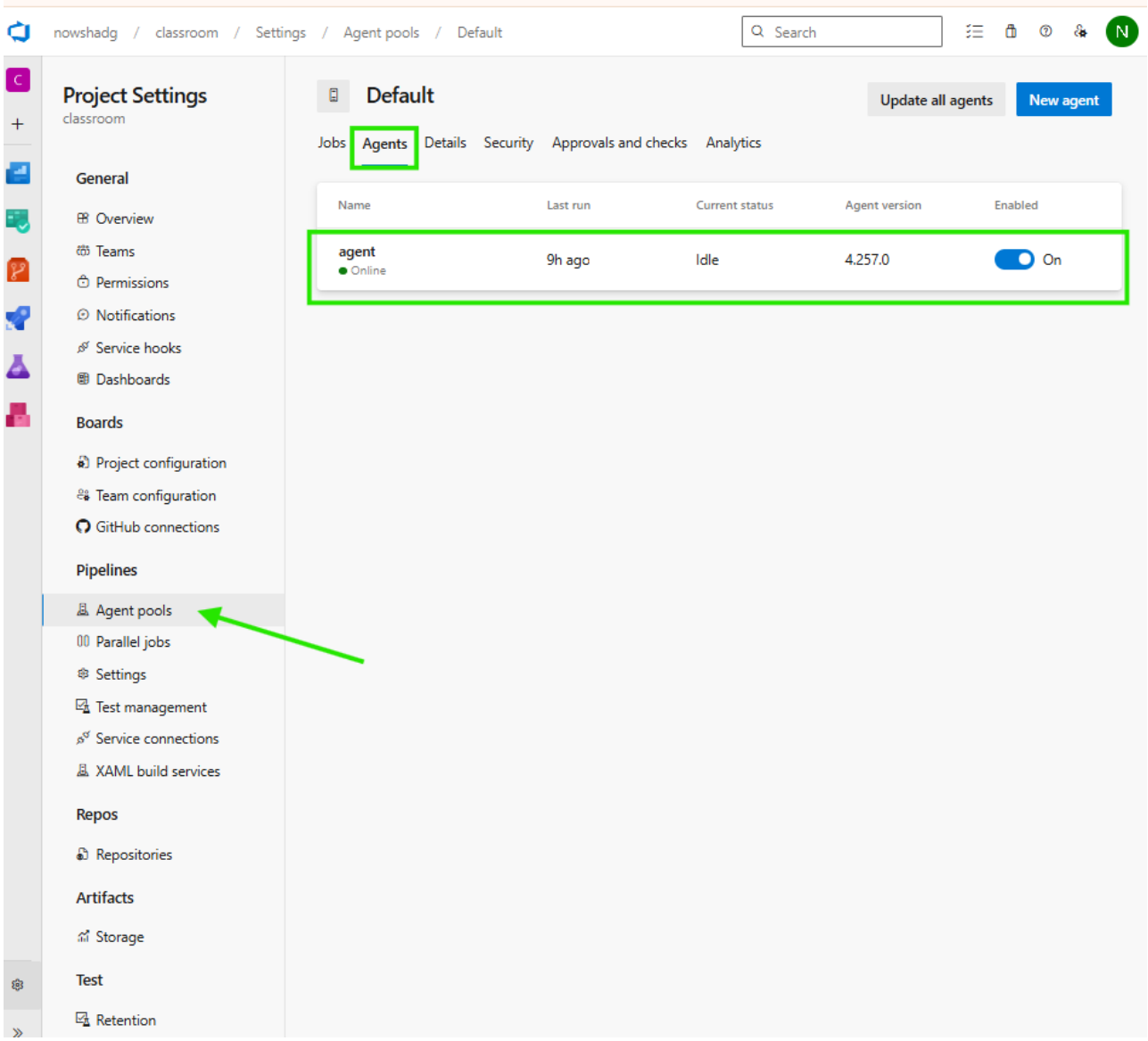
```
Aravindh@agent:~/myagent$ ./config.sh

  ___                      ______ _            _ _
 / _ \                     | ___ (_)          | (_)
/ /_\ \_____   _ _ __ ___  | |_/ /_ _ __   ___| |_ _ __   ___  ___
|  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \/ __|
| | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                   | |
        agent v4.257.0             |_|          (commit afb5fb6)


>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer
 Everywhere End User License Agreement. This step is not required for buildi
ng sources from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
  /home/Aravindh/myagent/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (pres
s enter for N) > y

>> Connect:

Enter server URL >
```



- copy and paste url `https://dev.azure.com/nowshadg`

- follow the asking promts and react and run finally `./run.sh` this brigs the server online.

```
A copy of the Team Explorer Everywhere license agreement can be found at:
  /home/Aravindh/myagent/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (pres
s enter for N) > y

>> Connect:

Enter server URL > https://dev.azure.com/nowshadg
Enter authentication type (press enter for PAT) >
Enter personal access token > ********************************************
************************************
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) >
Enter agent name (press enter for agent) >
Scanning for tool capabilities.
Connecting to the server.
Pool Default already contains an agent with name agent.
Enter replace? (Y/N) (press enter for N) >
Enter agent name (press enter for agent) >
Scanning for tool capabilities.
Connecting to the server.
Pool Default already contains an agent with name agent.
Enter replace? (Y/N) (press enter for N) > Y
Successfully replaced the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2025-05-29 16:45:10Z: Settings Saved.
Aravindh@agent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2025-05-29 16:45:30Z: Listening for Jobs
```



Step4:

1. Now we will write pipeline script for deployment the Application

```
stages:
  - stage: Build
    jobs:
      - job: Build
        pool:
          vmImage: ubuntu-22.04
        steps:
          - task: DotNetCoreCLI@2
            displayName: 'dotnet publish'
            inputs:
              command: 'publish'
              publishWebProjects: false
              arguments: '-o $(Build.ArtifactStagingDirectory)/Output'
              zipAfterPublish: true
              modifyOutputPath: true
          - task: PublishBuildArtifacts@1
            inputs:
```

```
            pathToPublish: $(Build.ArtifactStagingDirectory)
            artifactName: DbAppOutputs
  - stage: Deploy
    dependsOn: Build
      jobs:
        - job: systemtest environment
          pool: Default
          steps:
            - task: DownloadBuildArtifacts@1
              inputs:
                buildType: 'current'
                artifactName: DbAppOutputs
                downloadPath: '$(System.ArtifactsDirectory)'
            - bash: |
                terraform init
                terraform apply -var-file='dev.tfvars' -auto-approve'
              displayName: Terraform VM deployment
              workingDirectory: DbConnectionTester/deploy/Terraform
```