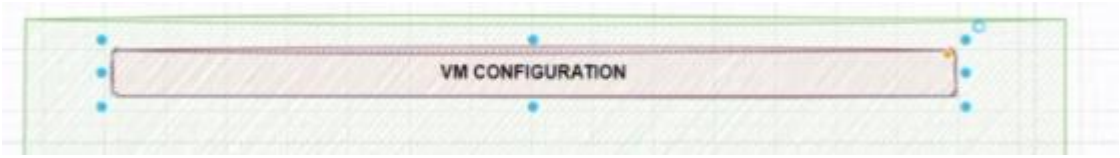
11 August 2024

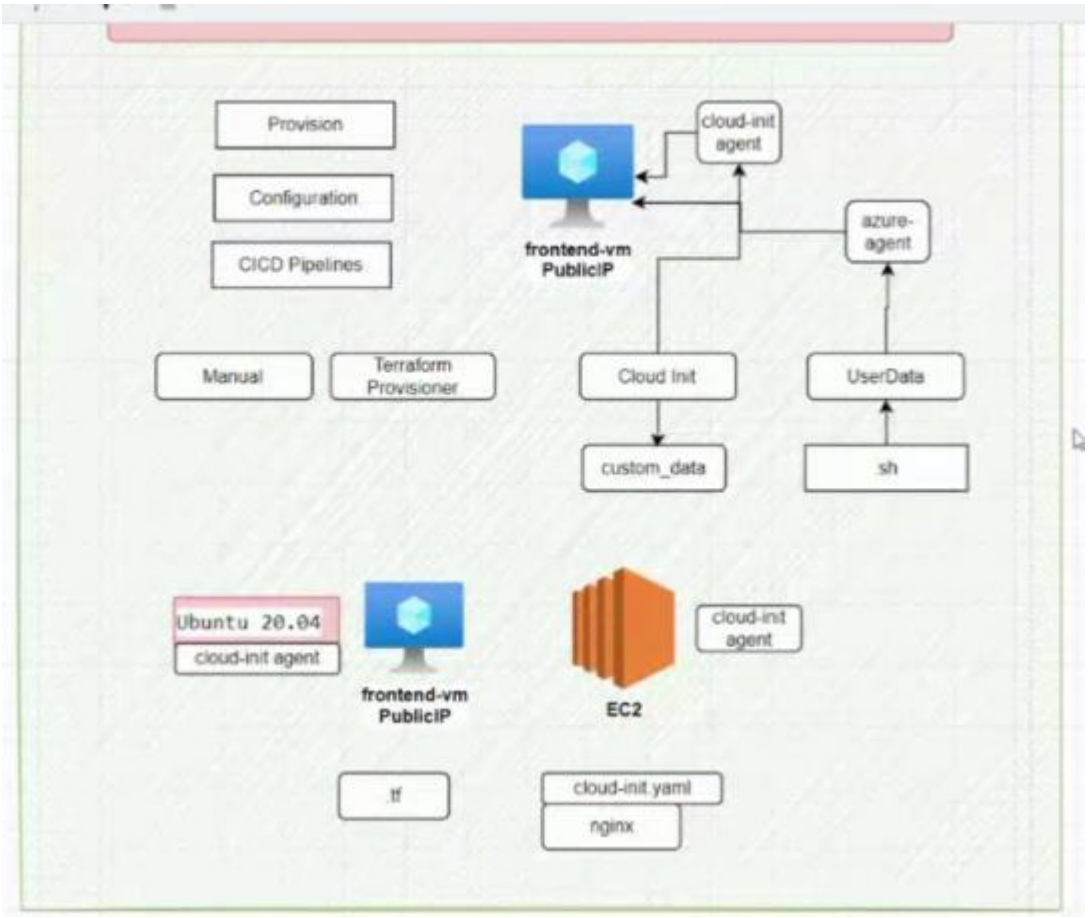
AGENDA – Writing in yaml

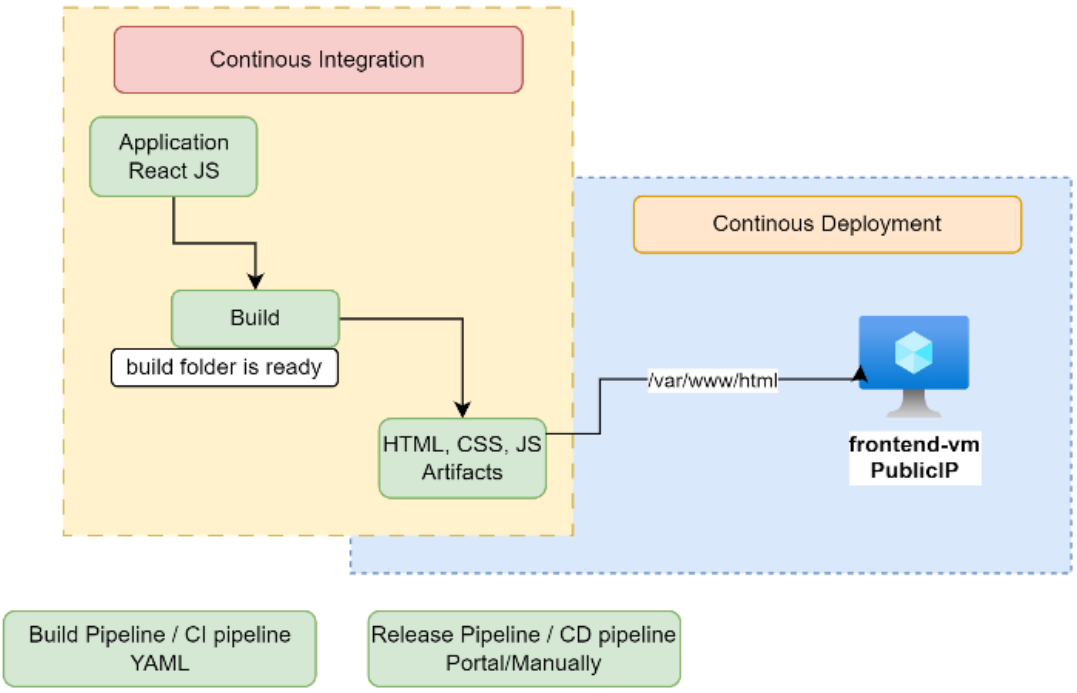
1) What is backlog, sprint?

2) 2 branching strategy

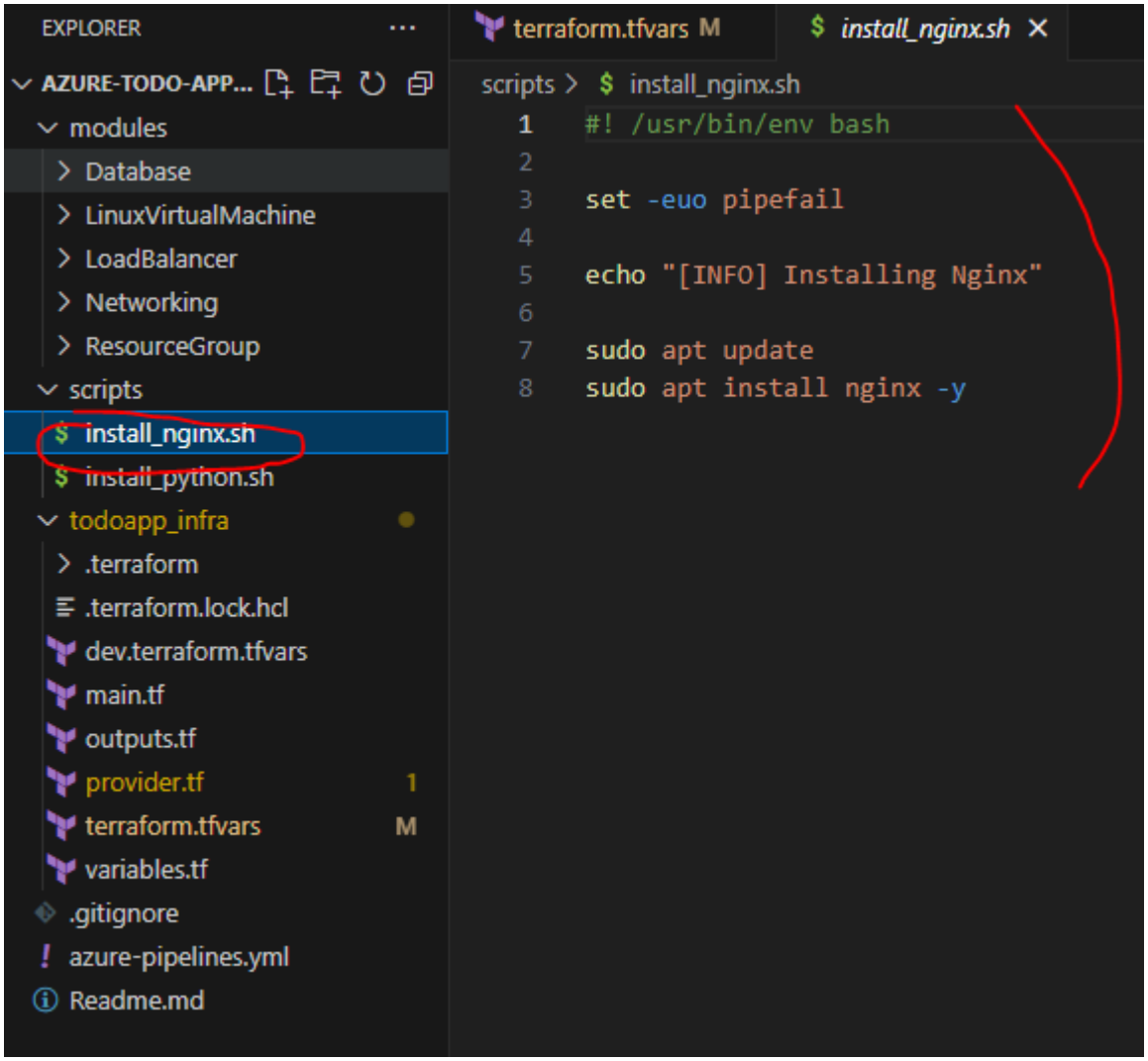
i) trunk based in terraform – in main branch we made 2 folders preprod and prod. We made 2 pipelines for them. Now using git clone we made feature branch for dev folder. Now pushed it and raised a PR then if plan was fine then PR was approved. Then it merged and automatically pipeline ran then all stages of pipeline ran along with tfsec, tuple of, linter, checkov all tools ran and then apply ran and got deployed.

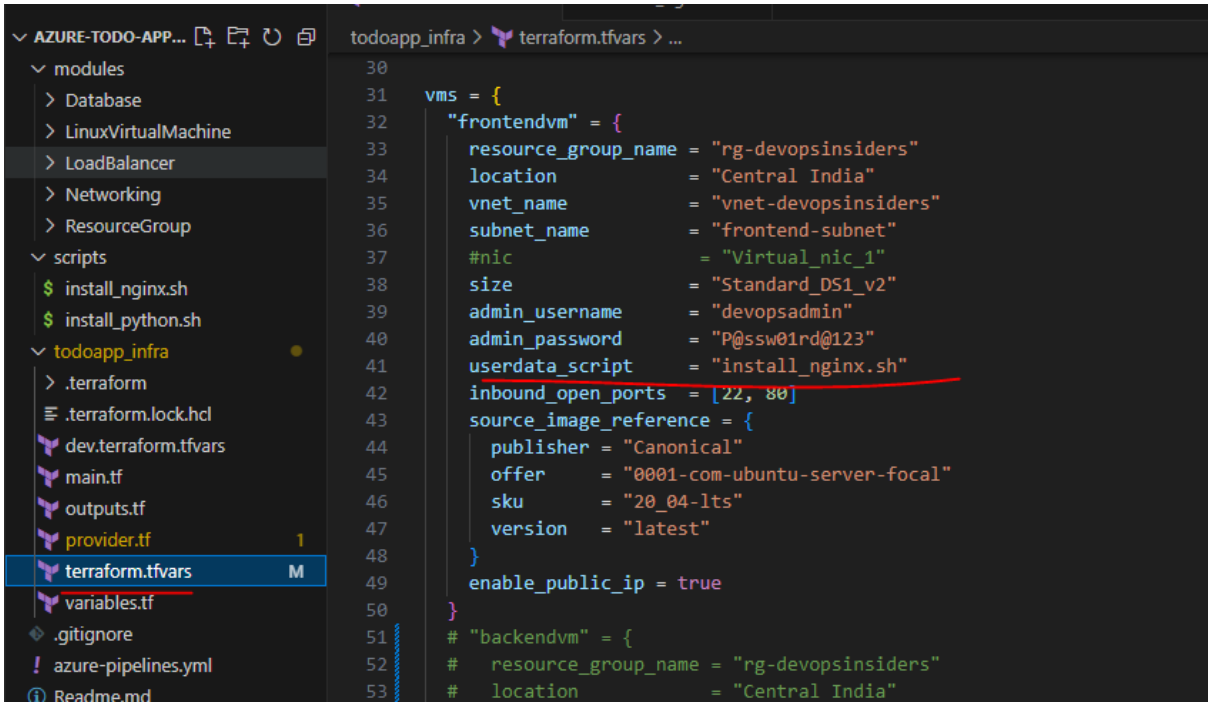


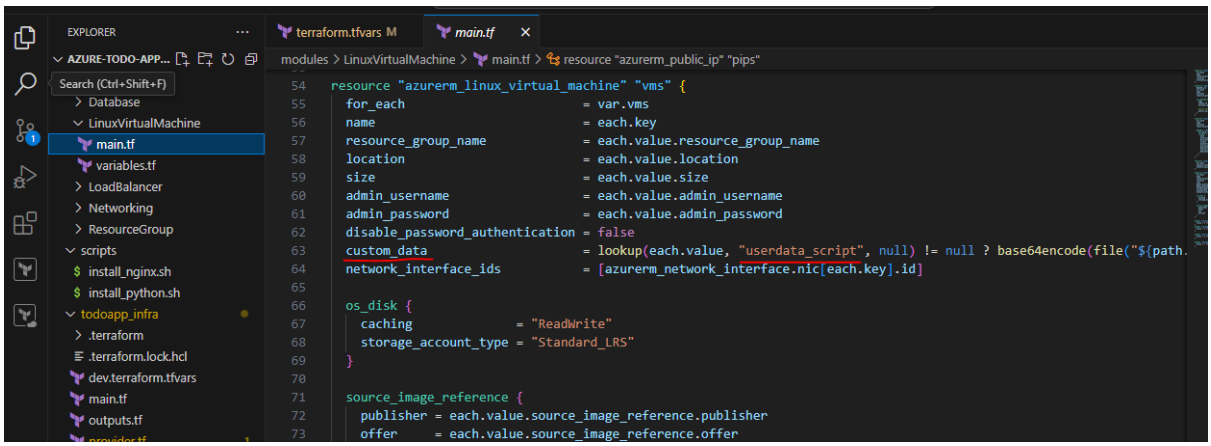




3) Now put nginx script in code





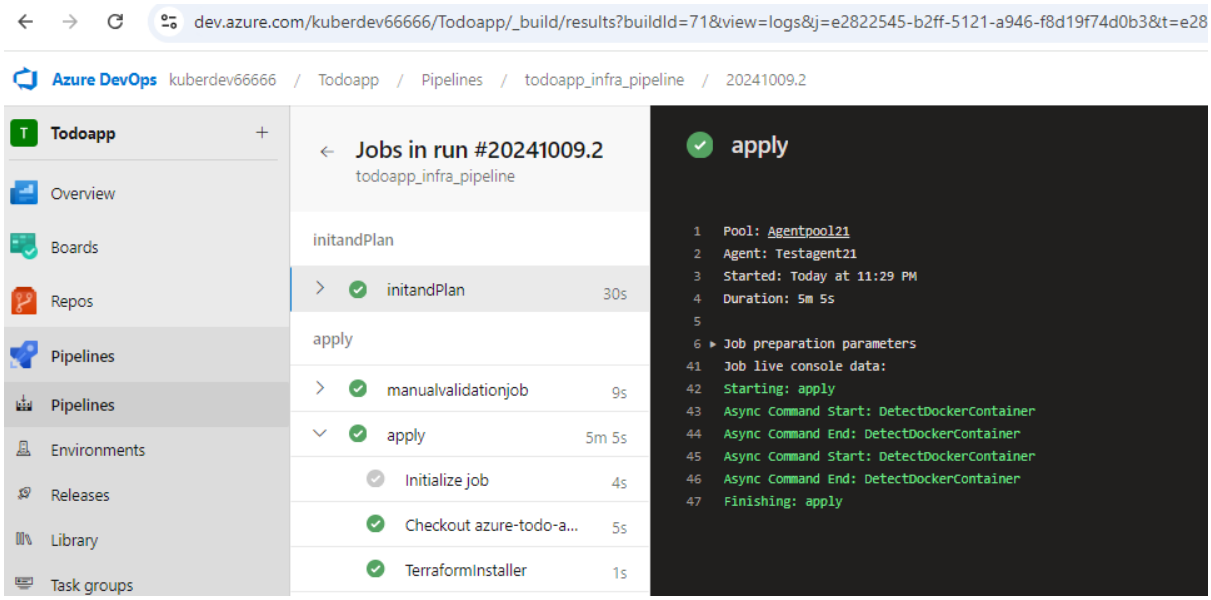


4) git add .

git commit -m “updated”

git push

5) run pipeline



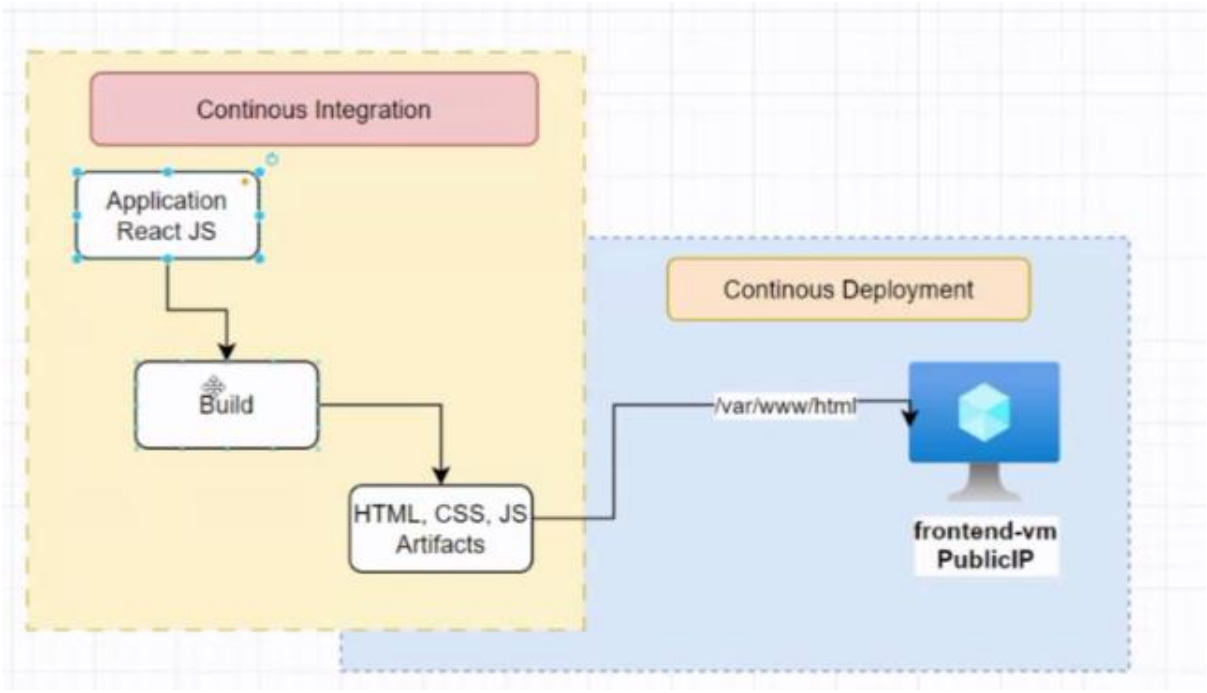
6) Now ssh into frontend vm and check whether nginx is there or not as we had already used nginx script in our code. So nginx is there.

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

devopsadmin@frontendvm:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-10-09 18:02:18 UTC; 4min 5s ago
       Docs: man:nginx(8)
   Main PID: 2337 (nginx)
      Tasks: 2 (limit: 4081)
     Memory: 4.9M
     CGroup: /system.slice/nginx.service
             ├─2337 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─2338 nginx: worker process

Oct 09 18:02:17 frontendvm systemd[1]: Starting A high performance web server and a reverse proxy server...
Oct 09 18:02:18 frontendvm systemd[1]: Started A high performance web server and a reverse proxy server.
devopsadmin@frontendvm:~$
```

7) Now we want that in single click our application can be made, then gets build, artefacts can be made (converted into html, css, javascript) and then it gets deployed on our vm as shown in below diagram.



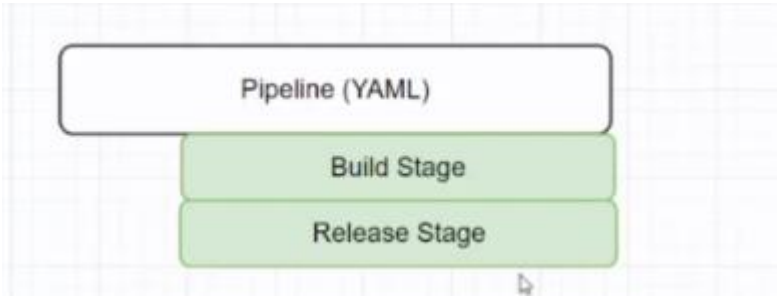
8) Continuous Integration – The process of making artefacts by building the code of application automatically

9) Continuous Deployment – The process of transferring artefacts on to vm automatically.

10) Build pipeline and release pipeline is shown below

11) Release or CD pipeline is not available in yaml

12) So in single pipeline we will create 2 stages



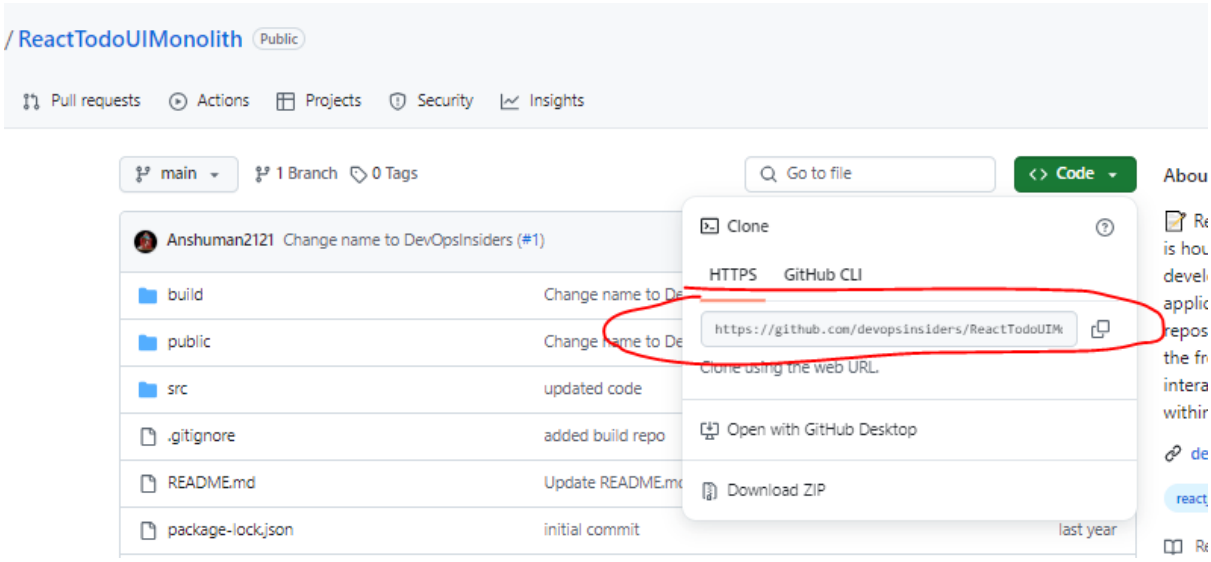
13) Now go to dev.azure.com and import below code in repo from github

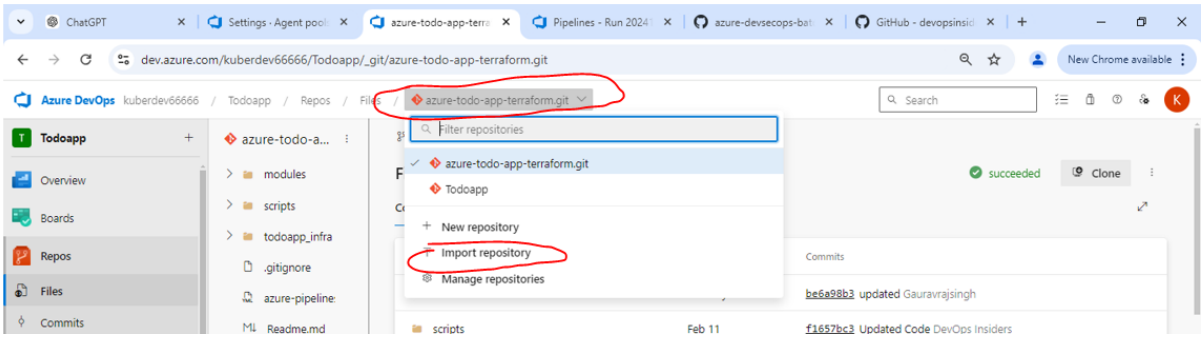
ReactTodoUIMonolith Public

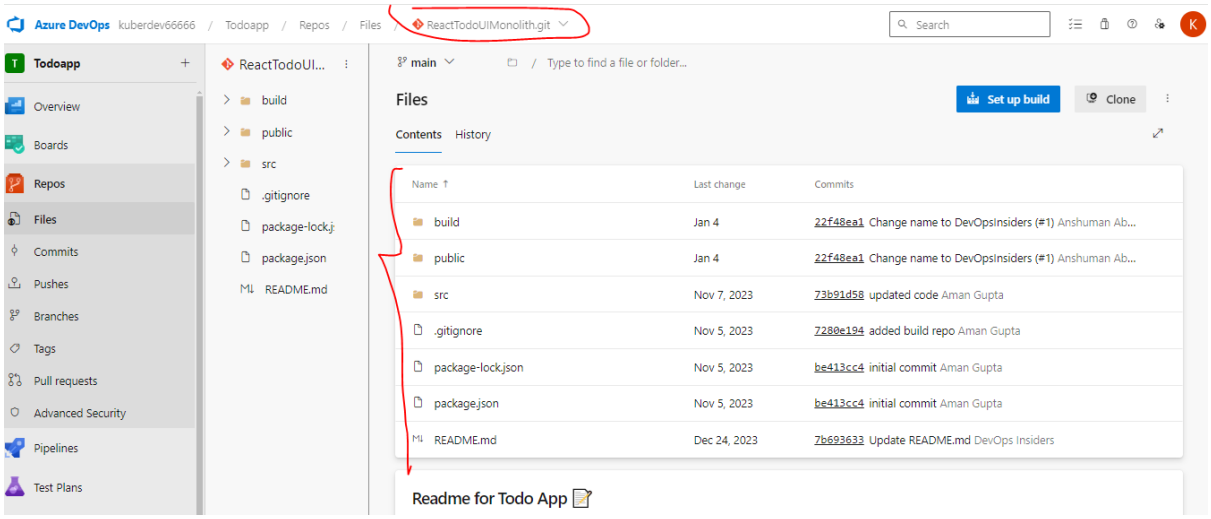
ReactTodoUIMonolith Repository is housing the user interface code developed in React for the Todo application's monolithic architecture. This repository is responsible for managing the fronten...

reactjs monolithic todoapp

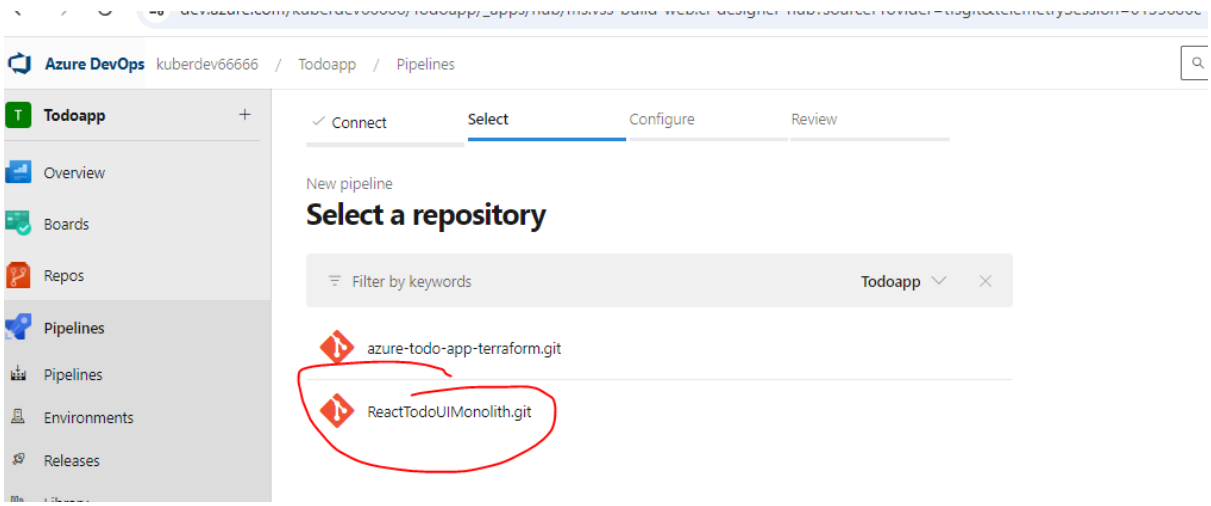
JavaScript 6 26 Updated on Jan 4



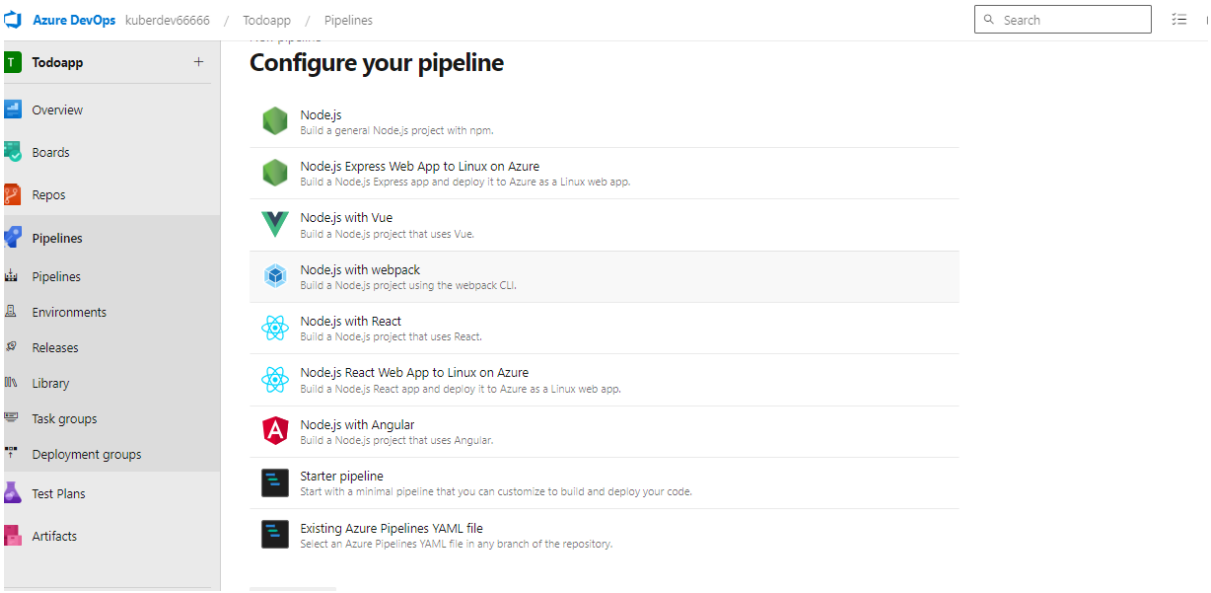




14) Now we will make our pipeline – select new pipeline – select repo below



15) Now azure devops is so smart that it identified that it's a node.js app

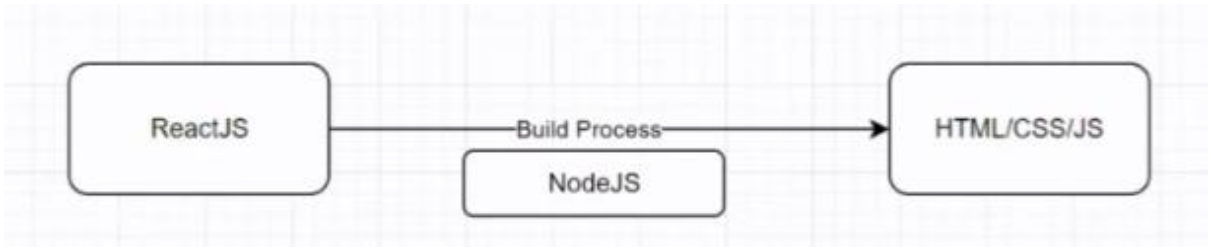


16) Select starter pipeline

17) SEARCH – azure YAML schema reference

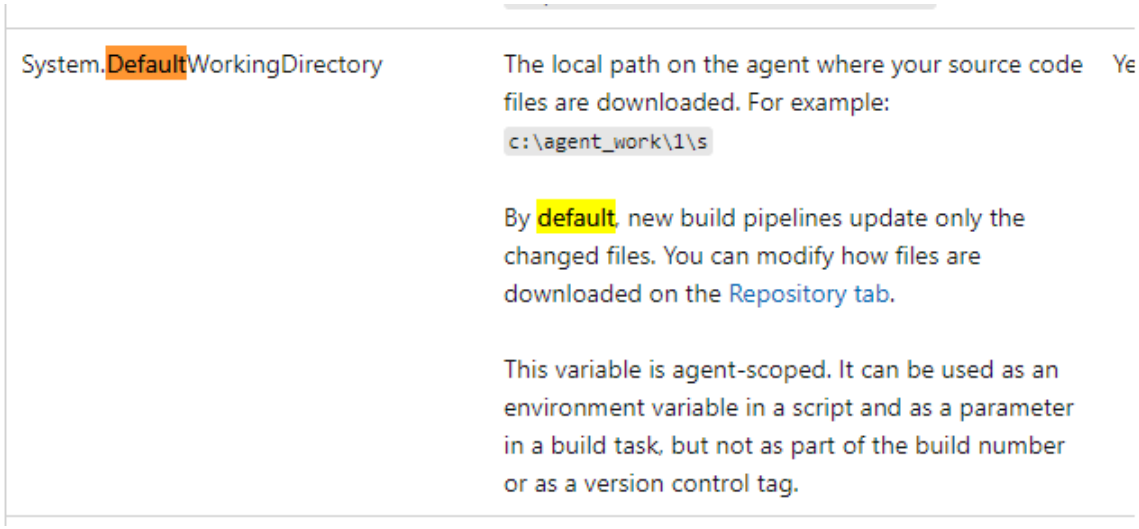
https://learn.microsoft.com/en-us/azure/devops/pipelines/yaml-schema/?view=azure-pipelines

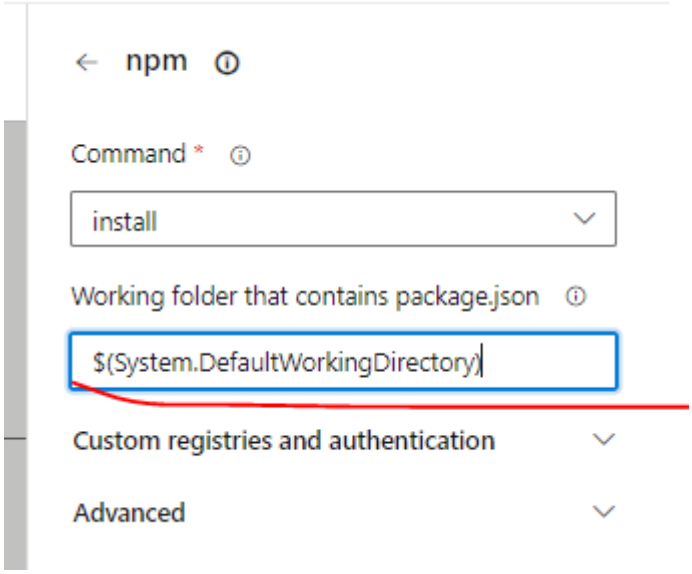
18) node.js acts as medium to convert react.js into html, css, javascript so that it can run on browser



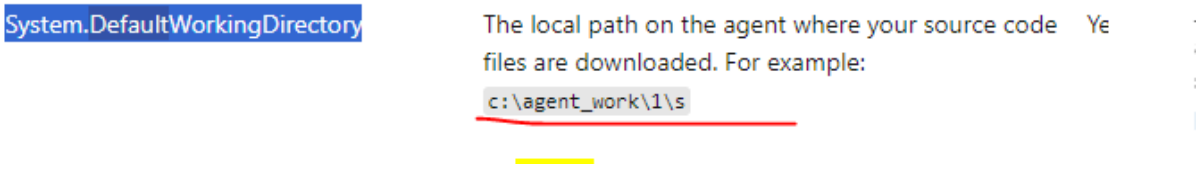
19) SEARCH - predefined variables azure devops

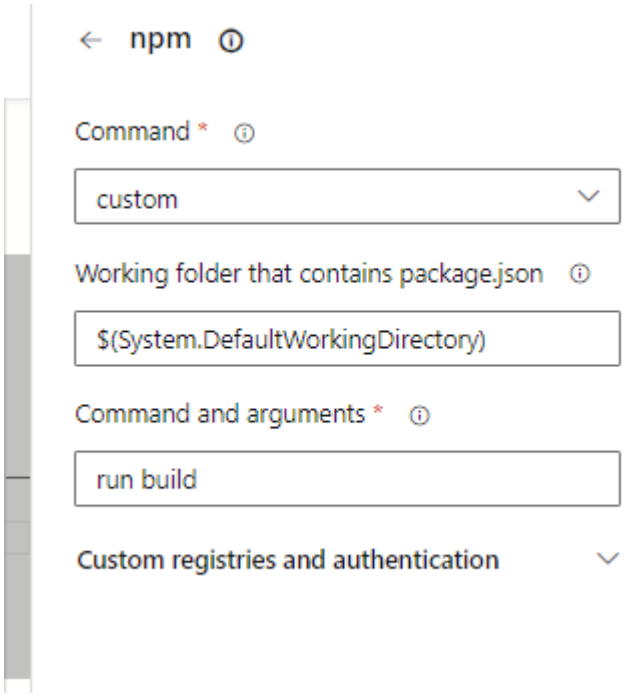
https://learn.microsoft.com/en-us/azure/devops/pipelines/build/variables?view=azure-devops&tabs=yaml





20 Jis jagah agent ke andar code clone hoga, us s wali directory ka path yaha aajayega aur usi directory ya folder ke andar package.json file hogi

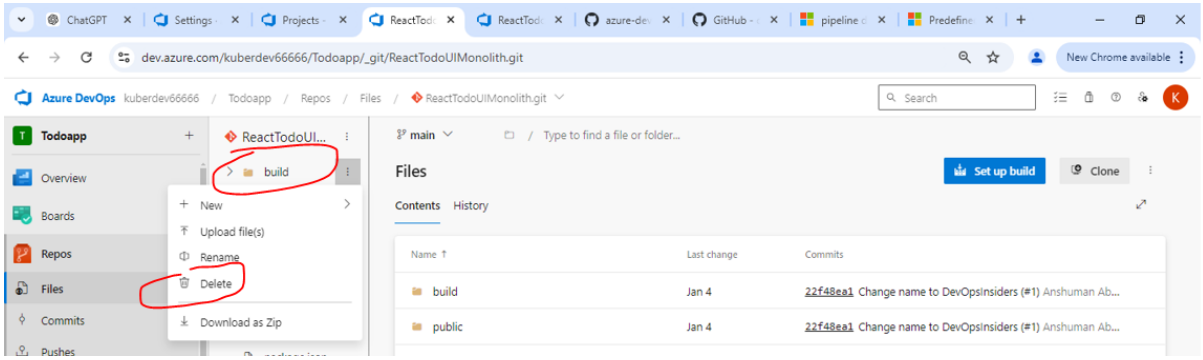




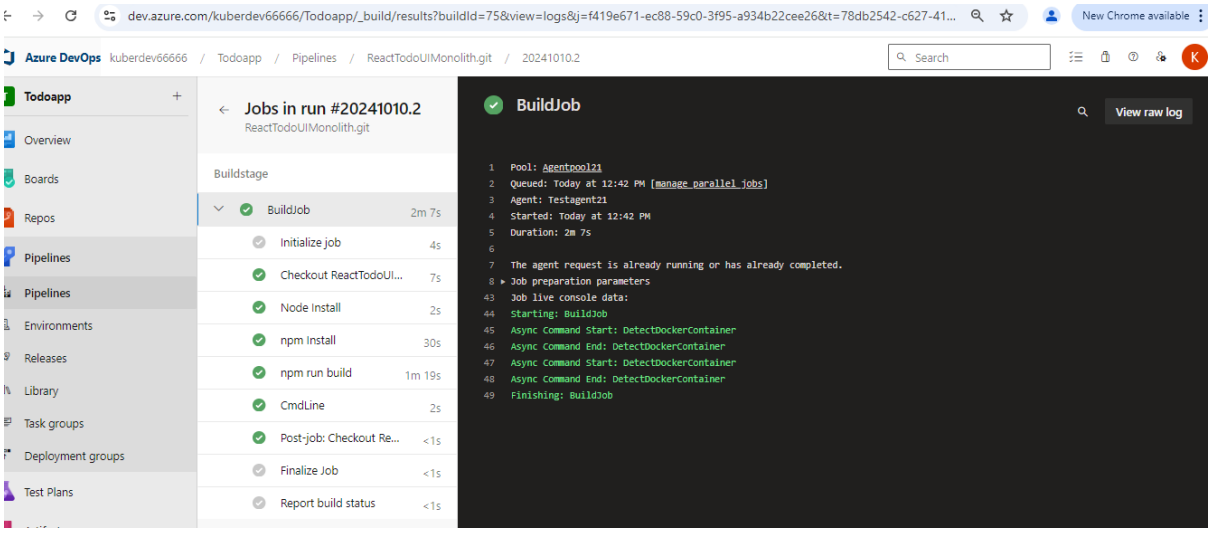
21) Now checking whether build folder is creating or not so we can use command line or power shell

```
    customCommand: 'run build'
  Settings
- task: CmdLine@2
  inputs:
    script: 'ls'
```

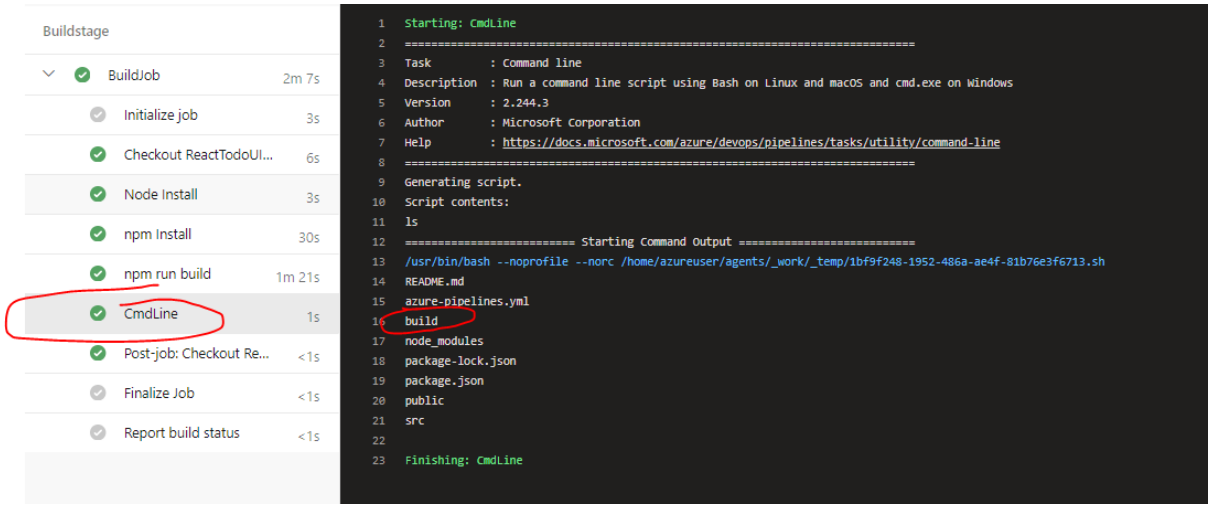
22) Now delete build folder because it has to be generated or created



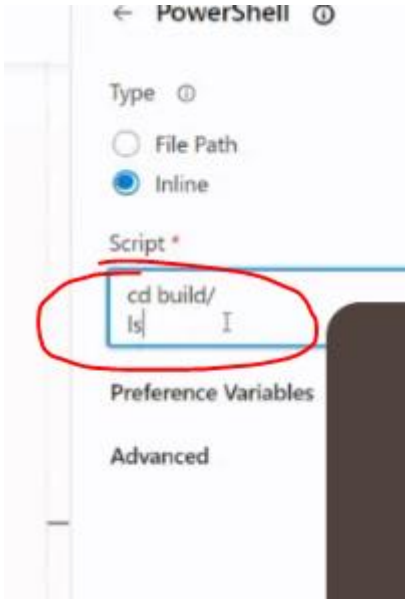
23) Now running pipeline

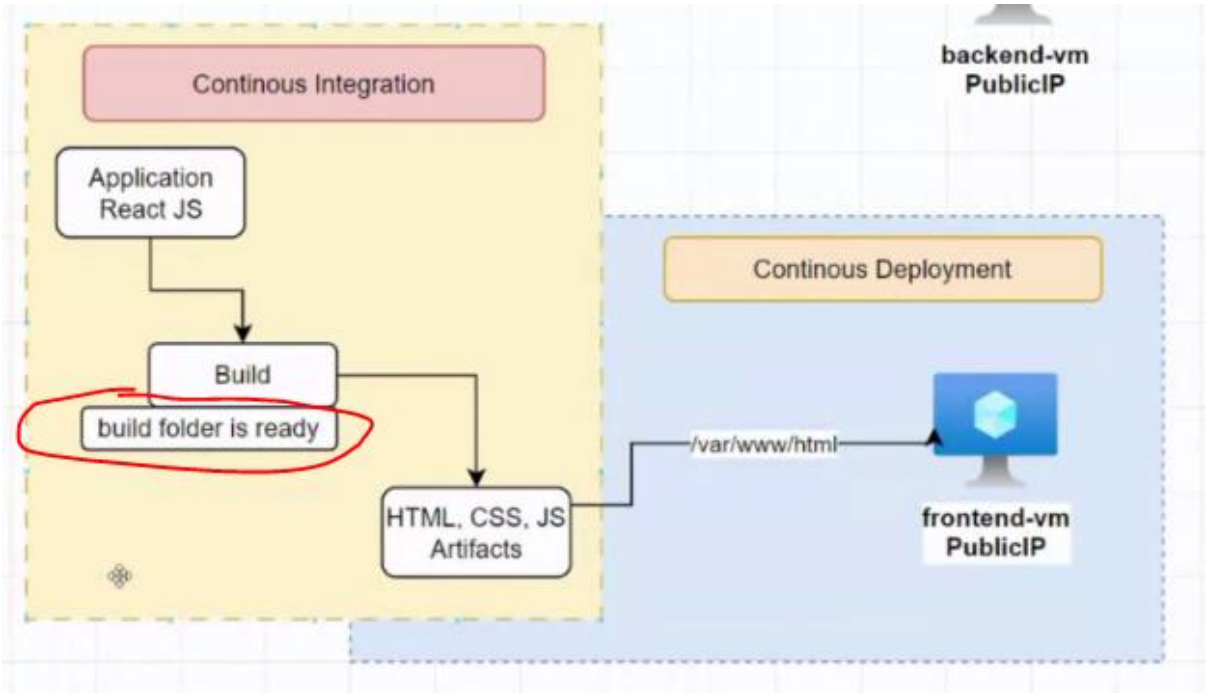


24) Now we can see in pipeline that our build folder is present, after running ls

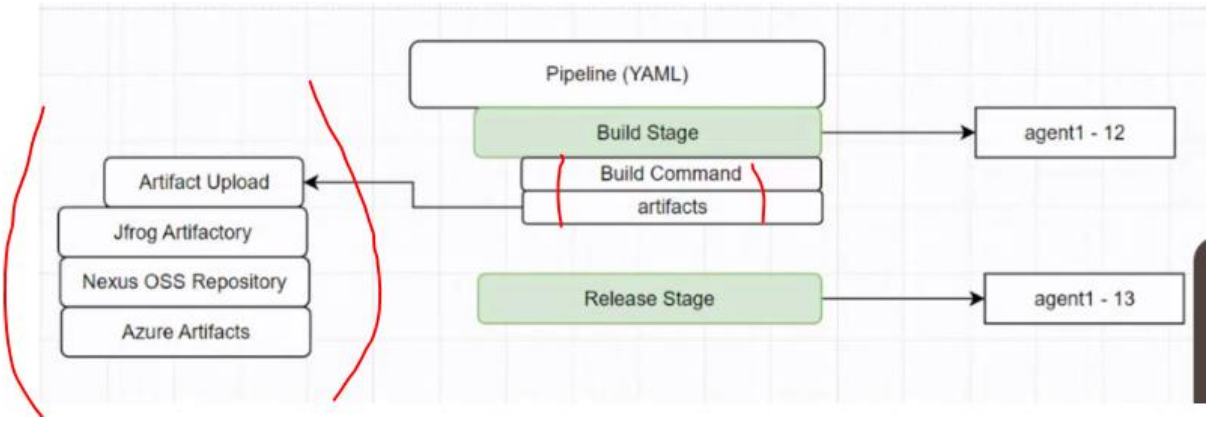


25) Now what is made inside build folder can see as below





26)



27) Use “publish pipeline artifacts” or storage account to upload and publish artefacts i.e. build folder. So basically transferring content of build folder into storage account.

28) SEARCH – azure storage blob directory

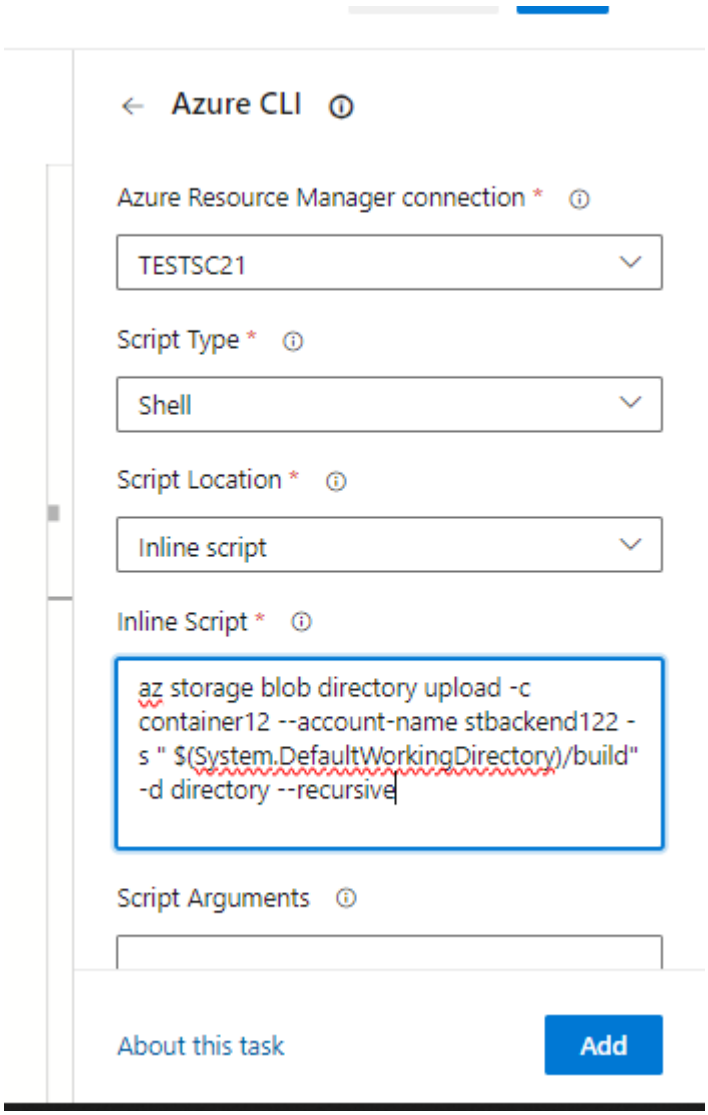
https://learn.microsoft.com/en-us/cli/azure/storage/blob/directory?view=azure-cli-latest#az-storage-blob-directory-upload

Upload a local directory to a storage blob directory.

Azure CLI | Copy | Open Cloud Shell

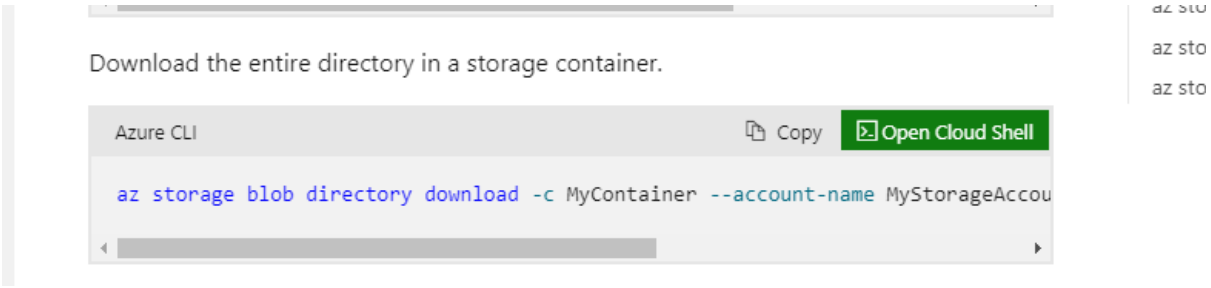
```
az storage blob directory upload -c MyContainer --account-name MyStorageAccount
```

az storage blob directory upload -c container12 --account-name stbackend122 -s "$(System.DefaultWorkingDirectory)/build" -d directory --recursive

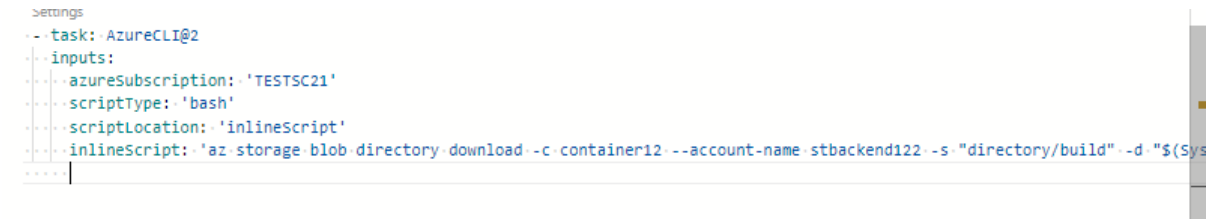


29) Install azure cli on vm

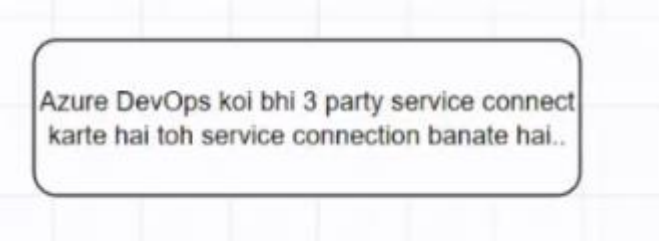
30) Now write release stage and use below one

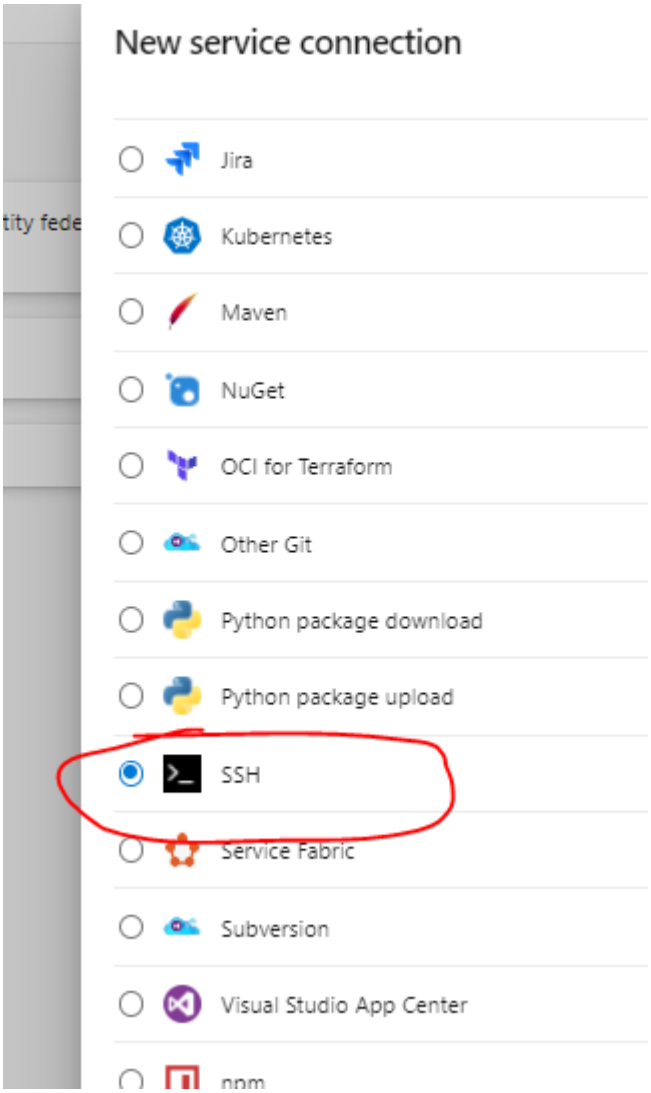


az storage blob directory download -c **container12** --account-name **stbackend122** -s SourceDirectoryPath -d "<local-path>" --recursive

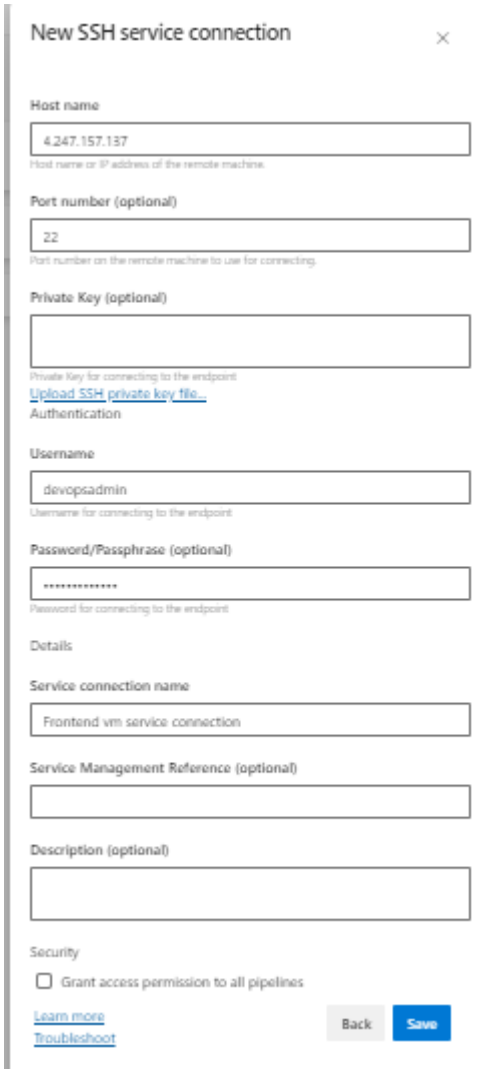


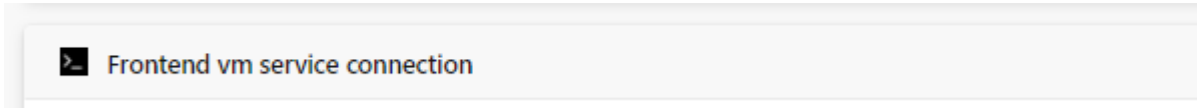
31) In ADO, for making 3rd party service to connect, we use service connection





31a) Give details of frontend vm





32) Lets transfer build artifacts folder into release folder

'az storage blob directory download -c container12 --account-name stbackend122 -s "directory/build" -d "$(System.DefaultWorkingDirectory)/release" --recursive'

```
'az storage blob directory download -c container12 --account-name stbackend122 -s "directory/build" -d "$(System.DefaultWorkingDirectory)/release" --recursive'
```

33) After this run public ip of frontend vm in browser