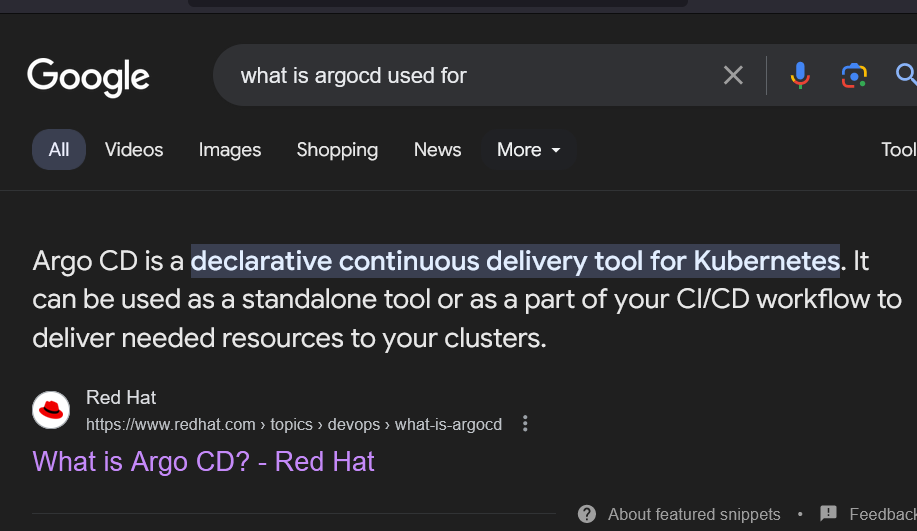
<https://www.digitalocean.com/community/tutorials/how-to-deploy-to-kubernetes-using-argo-cd-and-gitops>

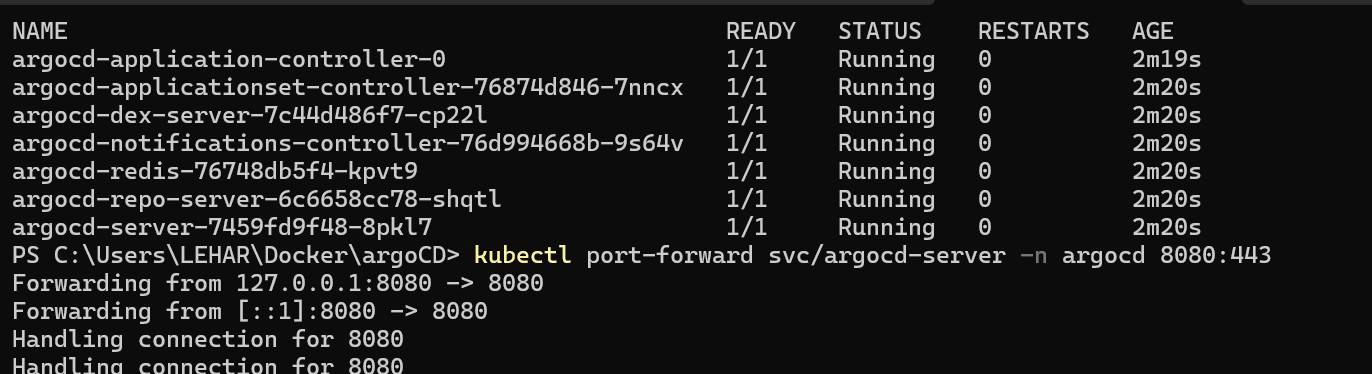
<https://argo-cd.readthedocs.io/en/stable/cli_installation/>



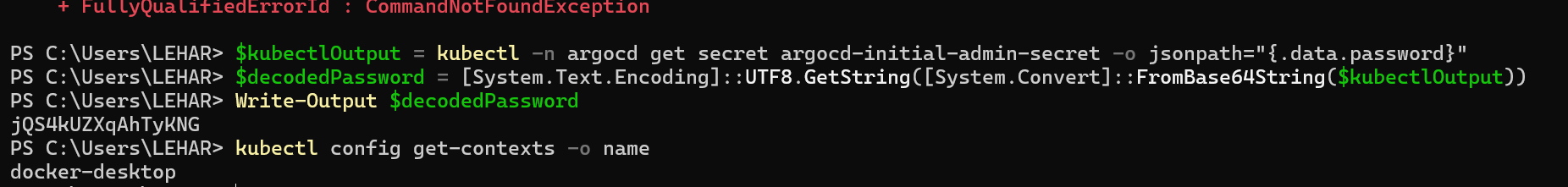
Installing argo cd

<https://www.digitalocean.com/community/tutorials/how-to-deploy-to-kubernetes-using-argo-cd-and-gitops>

with this link on windows woth digital ocean\



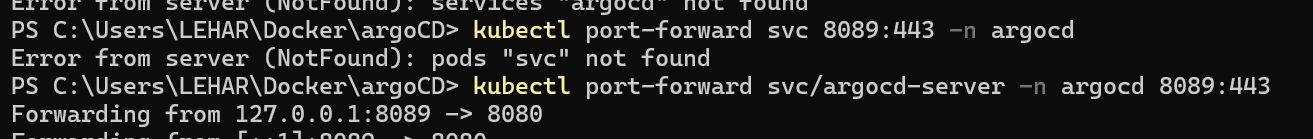
to login get password

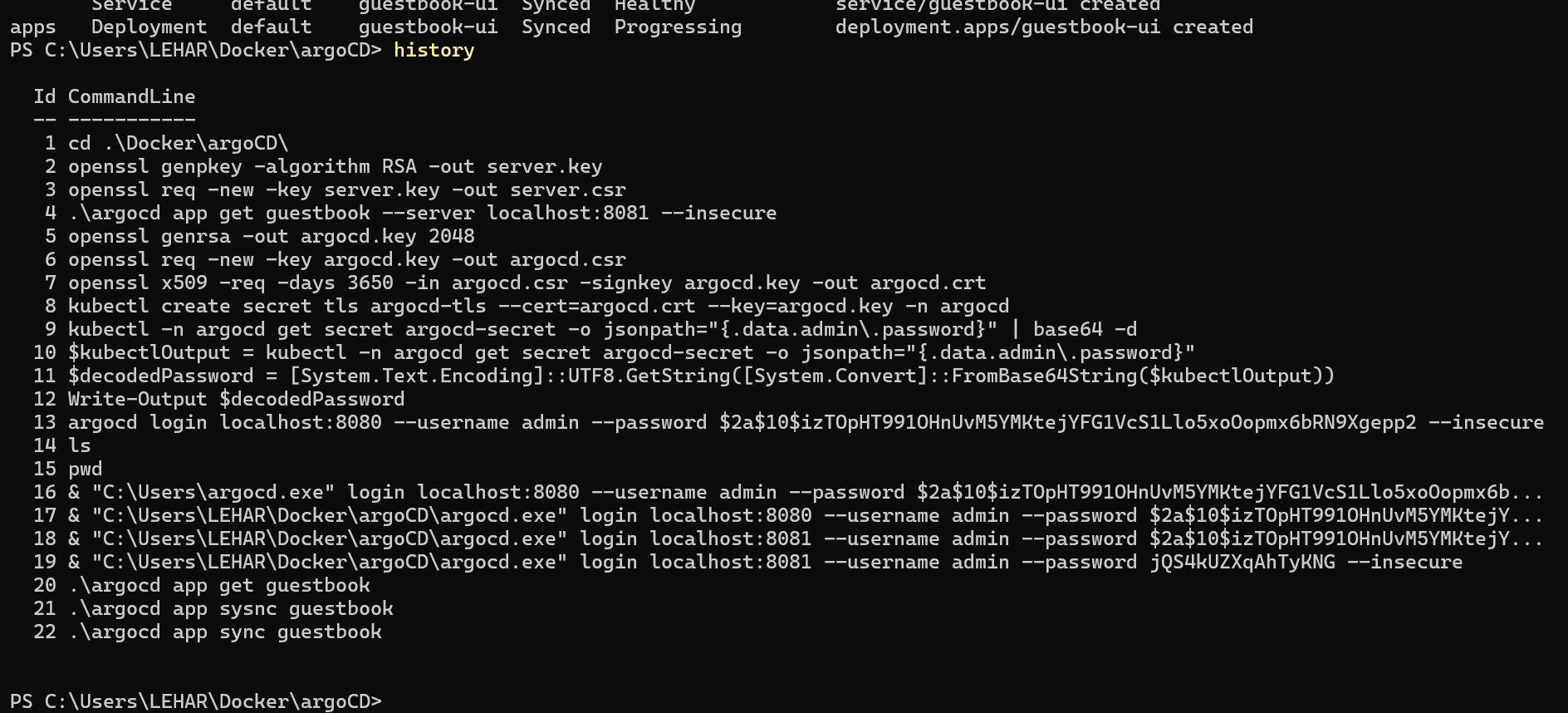


Admin

Password

jQS4kUZXqAhTyKNG





Cmds

1 cd .\Docker\argoCD\

2 openssl genpkey -algorithm RSA -out server.key

3 openssl req -new -key server.key -out server.csr

4 .\argocd app get guestbook --server localhost:8081 --insecure

5 openssl genrsa -out argocd.key 2048

6 openssl req -new -key argocd.key -out argocd.csr

7 openssl x509 -req -days 3650 -in argocd.csr -signkey argocd.key -out argocd.crt

8 kubectl create secret tls argocd-tls --cert=argocd.crt --key=argocd.key -n argocd

9 kubectl -n argocd get secret argocd-secret -o jsonpath="{.data.admin\.password}" | base64 -d

10 $kubectlOutput = kubectl -n argocd get secret argocd-secret -o jsonpath="{.data.admin\.password}"

11 $decodedPassword = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($kubectlOutput))

12 Write-Output $decodedPassword

13 argocd login localhost:8080 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6bRN9Xgepp2 --insecure

14 ls

15 pwd

16 & "C:\Users\argocd.exe" login localhost:8080 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6b...

17 & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8080 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejY...

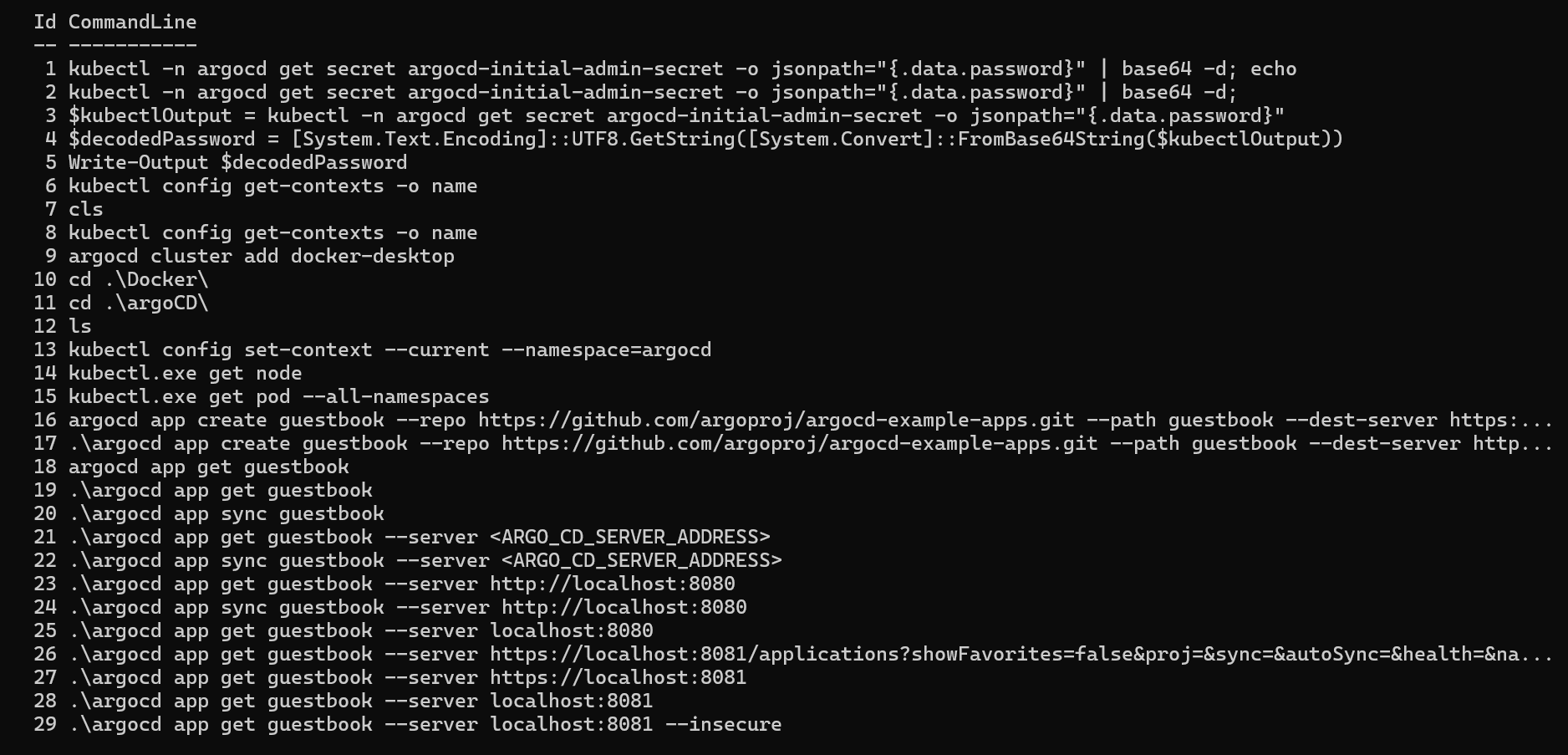
18 & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8081 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejY...

19 & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8081 --username admin --password jQS4kUZXqAhTyKNG --insecure

20 .\argocd app get guestbook

21 .\argocd app sysnc guestbook

22 .\argocd app sync guestbook



1 kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d; echo

2 kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d;

3 $kubectlOutput = kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}"

4 $decodedPassword = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($kubectlOutput))

5 Write-Output $decodedPassword

6 kubectl config get-contexts -o name

7 cls

8 kubectl config get-contexts -o name

9 argocd cluster add docker-desktop

10 cd .\Docker\

11 cd .\argoCD\

12 ls

13 kubectl config set-context --current --namespace=argocd

14 kubectl.exe get node

15 kubectl.exe get pod --all-namespaces

16 argocd app create guestbook --repo https://github.com/argoproj/argocd-example-apps.git --path guestbook --dest-server https:...

17 .\argocd app create guestbook --repo https://github.com/argoproj/argocd-example-apps.git --path guestbook --dest-server http...

18 argocd app get guestbook

19 .\argocd app get guestbook

20 .\argocd app sync guestbook

21 .\argocd app get guestbook --server <ARGO\_CD\_SERVER\_ADDRESS>

22 .\argocd app sync guestbook --server <ARGO\_CD\_SERVER\_ADDRESS>

23 .\argocd app get guestbook --server http://localhost:8080

24 .\argocd app sync guestbook --server http://localhost:8080

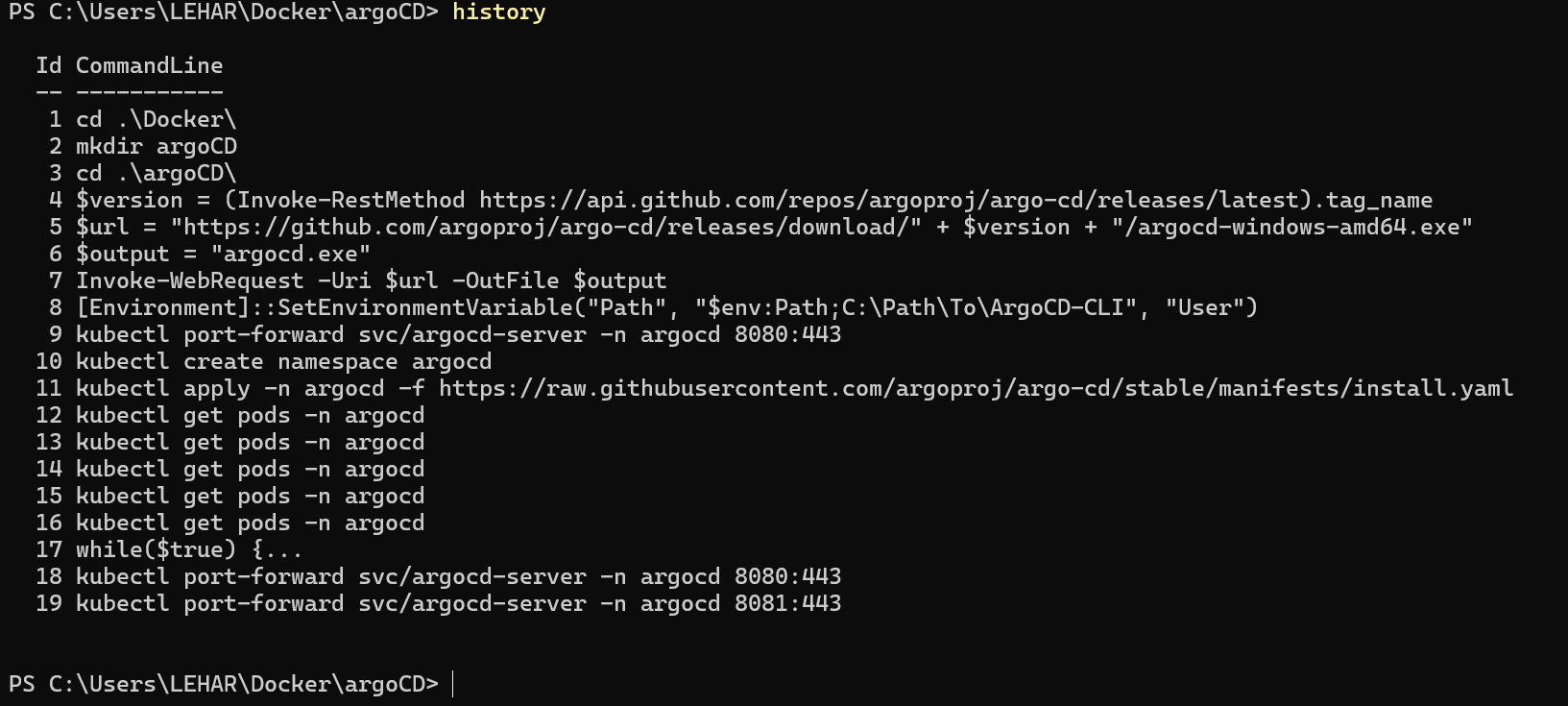
25 .\argocd app get guestbook --server localhost:8080

26 .\argocd app get guestbook --server https://localhost:8081/applications?showFavorites=false&proj=&sync=&autoSync=&health=&na...

27 .\argocd app get guestbook --server https://localhost:8081

28 .\argocd app get guestbook --server localhost:8081

29 .\argocd app get guestbook --server localhost:8081 –insecure



PS C:\Users\LEHAR\Docker\argoCD> history

Id CommandLine

-- -----------

1 cd .\Docker\

2 mkdir argoCD

3 cd .\argoCD\

4 $version = (Invoke-RestMethod https://api.github.com/repos/argoproj/argo-cd/releases/latest).tag\_name

5 $url = "https://github.com/argoproj/argo-cd/releases/download/" + $version + "/argocd-windows-amd64.exe"

6 $output = "argocd.exe"

7 Invoke-WebRequest -Uri $url -OutFile $output

8 [Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Path\To\ArgoCD-CLI", "User")

9 kubectl port-forward svc/argocd-server -n argocd 8080:443

10 kubectl create namespace argocd

11 kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

12 kubectl get pods -n argocd

13 kubectl get pods -n argocd

14 kubectl get pods -n argocd

15 kubectl get pods -n argocd

16 kubectl get pods -n argocd

17 while($true) {...

18 kubectl port-forward svc/argocd-server -n argocd 8080:443

19 kubectl port-forward svc/argocd-server -n argocd 8081:443

Code with generated key and all

Local host:8081

PS C:\Users\LEHAR> cd .\Docker\argoCD\

PS C:\Users\LEHAR\Docker\argoCD> openssl genpkey -algorithm RSA -out server.key

.....................+..+.+++++++++++++++++++++++++++++++++++++++\*..............+...........+.+..+...............+......+.+......+.....+.........+....+..+...+....+...+..+.+...........+....+.........+.....+......+.+.....+.+..+............+......+...+....+..+...+++++++++++++++++++++++++++++++++++++++\*.........+...+............+.....+...+...+.......+.....+....+.....+....+.....+......+....+.....+............+.......+...+............+......+......+........+...+....+...+...+...+..........................+..

...........+...+....+.....+.............+......+..+.......+........+...+.+...+.....+.+...........+.........+.......+.....+....+........+...+...+...............+....+............+........+..........+...+..+................+.....+.+.....+....+.....+......+.+.........+...+...+...+..+.......+...+..+............+..........+...+........+...+.........+.............+...+......+......+..+...+....+...+...++++++

PS C:\Users\LEHAR\Docker\argoCD> openssl req -new -key server.key -out server.csr

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:AU

State or Province Name (full name) [Some-State]:rajasthan

Locality Name (eg, city) []:jaipur

Organization Name (eg, company) [Internet Widgits Pty Ltd]:8bit

Organizational Unit Name (eg, section) []:8bit

Common Name (e.g. server FQDN or YOUR name) []:argocd

Email Address []:agrawallehar66@gmail.com

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:argocd

An optional company name []:8bit

PS C:\Users\LEHAR\Docker\argoCD> .\argocd app get guestbook --server localhost:8081 --insecure

time="2024-05-10T15:25:33+05:30" level=fatal msg="rpc error: code = Unauthenticated desc = no session information"

PS C:\Users\LEHAR\Docker\argoCD> openssl genrsa -out argocd.key 2048

PS C:\Users\LEHAR\Docker\argoCD> openssl req -new -key argocd.key -out argocd.csr

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:AU

State or Province Name (full name) [Some-State]:rajasthan

Locality Name (eg, city) []:jaipur

Organization Name (eg, company) [Internet Widgits Pty Ltd]:8bit

Organizational Unit Name (eg, section) []:8bit

Common Name (e.g. server FQDN or YOUR name) []:argocd

Email Address []:agrawallehar66@gmail.com

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:argocd

An optional company name []:8bit

PS C:\Users\LEHAR\Docker\argoCD> openssl x509 -req -days 3650 -in argocd.csr -signkey argocd.key -out argocd.crt

Certificate request self-signature ok

subject=C=AU, ST=rajasthan, L=jaipur, O=8bit, OU=8bit, CN=argocd, emailAddress=agrawallehar66@gmail.com

PS C:\Users\LEHAR\Docker\argoCD> kubectl create secret tls argocd-tls --cert=argocd.crt --key=argocd.key -n argocd

secret/argocd-tls created

PS C:\Users\LEHAR\Docker\argoCD> kubectl -n argocd get secret argocd-secret -o jsonpath="{.data.admin\.password}" | base64 -d

base64 : The term 'base64' is not recognized as the name of a cmdlet, function, script file, or operable program.

Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

At line:1 char:84

+ ... ecret argocd-secret -o jsonpath="{.data.admin\.password}" | base64 -d

+ ~~~~~~

+ CategoryInfo : ObjectNotFound: (base64:String) [], CommandNotFoundException

+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\LEHAR\Docker\argoCD> $kubectlOutput = kubectl -n argocd get secret argocd-secret -o jsonpath="{.data.admin\.password}"

PS C:\Users\LEHAR\Docker\argoCD> $decodedPassword = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($kubectlOutput))

PS C:\Users\LEHAR\Docker\argoCD> Write-Output $decodedPassword

$2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6bRN9Xgepp2

PS C:\Users\LEHAR\Docker\argoCD> argocd login localhost:8080 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6bRN9Xgepp2 --insecure

argocd : The term 'argocd' is not recognized as the name of a cmdlet, function, script file, or operable program.

Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

At line:1 char:1

+ argocd login localhost:8080 --username admin --password $2a$10$izTOpH ...

+ ~~~~~~

+ CategoryInfo : ObjectNotFound: (argocd:String) [], CommandNotFoundException

+ FullyQualifiedErrorId : CommandNotFoundException

Suggestion [3,General]: The command argocd was not found, but does exist in the current location. Windows PowerShell does not load commands from the current location by default. If you trust this command, instead type: ".\argocd". See "get-help about\_Command\_Precedence" for more details.

PS C:\Users\LEHAR\Docker\argoCD> pwd

Path----

C:\Users\LEHAR\Docker\argoCD

PS C:\Users\LEHAR\Docker\argoCD> & "C:\Users\argocd.exe" login localhost:8080 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6bRN9Xgepp2 --insecure

+ & "C:\Users\argocd.exe" login localhost:8080 --username admin --passw

+ CategoryInfo : ObjectNotFound: (C:\Users\argocd.exe:String) [

PS C:\Users\LEHAR\Docker\argoCD> & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8080 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6bRN9Xgepp2 --insecure

time="2024-05-10T15:31:54+05:30" level=fatal msg="dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it."

PS C:\Users\LEHAR\Docker\argoCD> & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8081 --username admin --password $2a$10$izTOpHT991OHnUvM5YMKtejYFG1VcS1Llo5xoOopmx6bRN9Xgepp2 --insecure

WARNING: server certificate had error: tls: failed to verify certific

PS C:\Users\LEHAR\Docker\argoCD> & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8081 --username admin --password jQS4kUZXqAhTyKNG --insecure

'admin:login' logged in successfully

Context 'localhost:8081' updated

PS C:\Users\LEHAR\Docker\argoCD> .\argocd app get guestbook

PS C:\Users\LEHAR\Docker\argoCD> .\argocd app sync guestbook

Manage applications

Usage:

argocd app [flags]

argocd app [command]

Examples:

# List all the applications.

argocd app list

# Get the details of a application

argocd app get my-app

# Set an override parameter

argocd app set my-app -p image.tag=v1.0.1

Available Commands:

actions Manage Resource actions

add-source Adds a source to the list of sources in the application

create Create an application

delete Delete an application

delete-resource Delete resource in an application

diff Perform a diff against the target and live state.

edit Edit application

get Get application details

history Show application deployment history

list List applications

logs Get logs of application pods

manifests Print manifests of an application

patch Patch application

patch-resource Patch resource in an application

remove-source Remove a source from multiple sources application. Counting starts with 1. Default value is -1.

resources List resource of application

rollback Rollback application to a previous deployed version by History ID, omitted will Rollback to the previous version

set Set application parameters

sync Sync an application to its target state

terminate-op Terminate running operation of an application

unset Unset application parameters

wait Wait for an application to reach a synced and healthy state

Flags:

--as string Username to impersonate for the operation

--as-group stringArray Group to impersonate for the operation, this flag can be repeated to specify multiple groups.

--as-uid string UID to impersonate for the operation

--certificate-authority string Path to a cert file for the certificate authority

--client-certificate string Path to a client certificate file for TLS

--client-key string Path to a client key file for TLS

--cluster string The name of the kubeconfig cluster to use

--context string The name of the kubeconfig context to use

--disable-compression If true, opt-out of response compression for all requests to the server

-h, --help help for app

--insecure-skip-tls-verify If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure

--kubeconfig string Path to a kube config. Only required if out-of-cluster

-n, --namespace string If present, the namespace scope for this CLI request

--password string Password for basic authentication to the API server

--proxy-url string If provided, this URL will be used to connect via proxy

--request-timeout string The length of time to wait before giving up on a single server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests. (default "0")

--tls-server-name string If provided, this name will be used to validate server certificate. If this is not provided, hostname used to contact the server is used.

--token string Bearer token for authentication to the API server

--user string The name of the kubeconfig user to use

--username string Username for basic authentication to the API server

Global Flags:

--auth-token string Authentication token

--client-crt string Client certificate file

--client-crt-key string Client certificate key file

--config string Path to Argo CD config (default "C:\\Users\\LEHAR/.config/argocd/config")

--controller-name string Name of the Argo CD Application controller; set this or the ARGOCD\_APPLICATION\_CONTROLLER\_NAME environment variable when the controller's name label differs from the default, for example when installing via the Helm chart (default "argocd-application-controller")

--core If set to true then CLI talks directly to Kubernetes instead of talking to Argo CD API server

--grpc-web Enables gRPC-web protocol. Useful if Argo CD server is behind proxy which does not support HTTP2.

--grpc-web-root-path string Enables gRPC-web protocol. Useful if Argo CD server is behind proxy which does not support HTTP2. Set web root.

-H, --header strings Sets additional header to all requests made by Argo CD CLI. (Can be repeated multiple times to add multiple headers, also supports comma separated headers)

--http-retry-max int Maximum number of retries to establish http connection to Argo CD server

--insecure Skip server certificate and domain verification

--kube-context string Directs the command to the given kube-context

--logformat string Set the logging format. One of: text|json (default "text")

--loglevel string Set the logging level. One of: debug|info|warn|error (default "info")

--plaintext Disable TLS

--port-forward Connect to a random argocd-server port using port forwarding

--port-forward-namespace string Namespace name which should be used for port forwarding

--redis-haproxy-name string Name of the Redis HA Proxy; set this or the ARGOCD\_REDIS\_HAPROXY\_NAME environment variable when the HA Proxy's name label differs from the default, for example when installing via the Helm chart (default "argocd-redis-ha-haproxy")

--redis-name string Name of the Redis deployment; set this or the ARGOCD\_REDIS\_NAME environment variable when the Redis's name label differs from the default, for example when installing via the Helm chart (default "argocd-redis")

--repo-server-name string Name of the Argo CD Repo server; set this or the ARGOCD\_REPO\_SERVER\_NAME environment variable when the server's name label differs from the default, for example when installing via the Helm chart (default "argocd-repo-server")

--server string Argo CD server address

--server-crt string Server certificate file

--server-name string Name of the Argo CD API server; set this or the ARGOCD\_SERVER\_NAME environment variable when the server's name label differs from the default, for example when installing via the Helm chart (default "argocd-server")

Use "argocd app [command] --help" for more information about a command.

PS C:\Users\LEHAR\Docker\argoCD> .\argocd app sync guestbook

TIMESTAMP GROUP KIND NAMESPACE

NAME STATUS HEALTH HOOK MESSAGE

2024-05-10T15:37:03+05:30 Service default

guestbook-ui OutOfSync Missing

2024-05-10T15:37:03+05:30 apps Deployment default

guestbook-ui OutOfSync Missing

2024-05-10T15:37:05+05:30 Service default

guestbook-ui Synced Healthy

2024-05-10T15:37:06+05:30 apps Deployment default

guestbook-ui OutOfSync Missing deployment.apps/guestbook-ui created

2024-05-10T15:37:06+05:30 Service default

guestbook-ui Synced Healthy service/guestbook-ui created

2024-05-10T15:37:06+05:30 apps Deployment default

guestbook-ui Synced Progressing deployment.apps/guestbook-ui created

Name: argocd/guestbook

Project: default

Server: https://kubernetes.default.svc

Namespace: default

URL: https://localhost:8081/applications/guestbook

Source:

- Repo: https://github.com/argoproj/argocd-example-apps.git

Target: HEAD

Path: guestbook

SyncWindow: Sync Allowed

Sync Policy: Manual

Sync Status: Synced to HEAD (d7927a2)

Health Status: Progressing

Operation: Sync

Sync Revision: d7927a27b4533926b7d86b5f249cd9ebe7625e90

Phase: Succeeded

Start: 2024-05-10 15:37:03 +0530 IST

Finished: 2024-05-10 15:37:05 +0530 IST

Duration: 2s

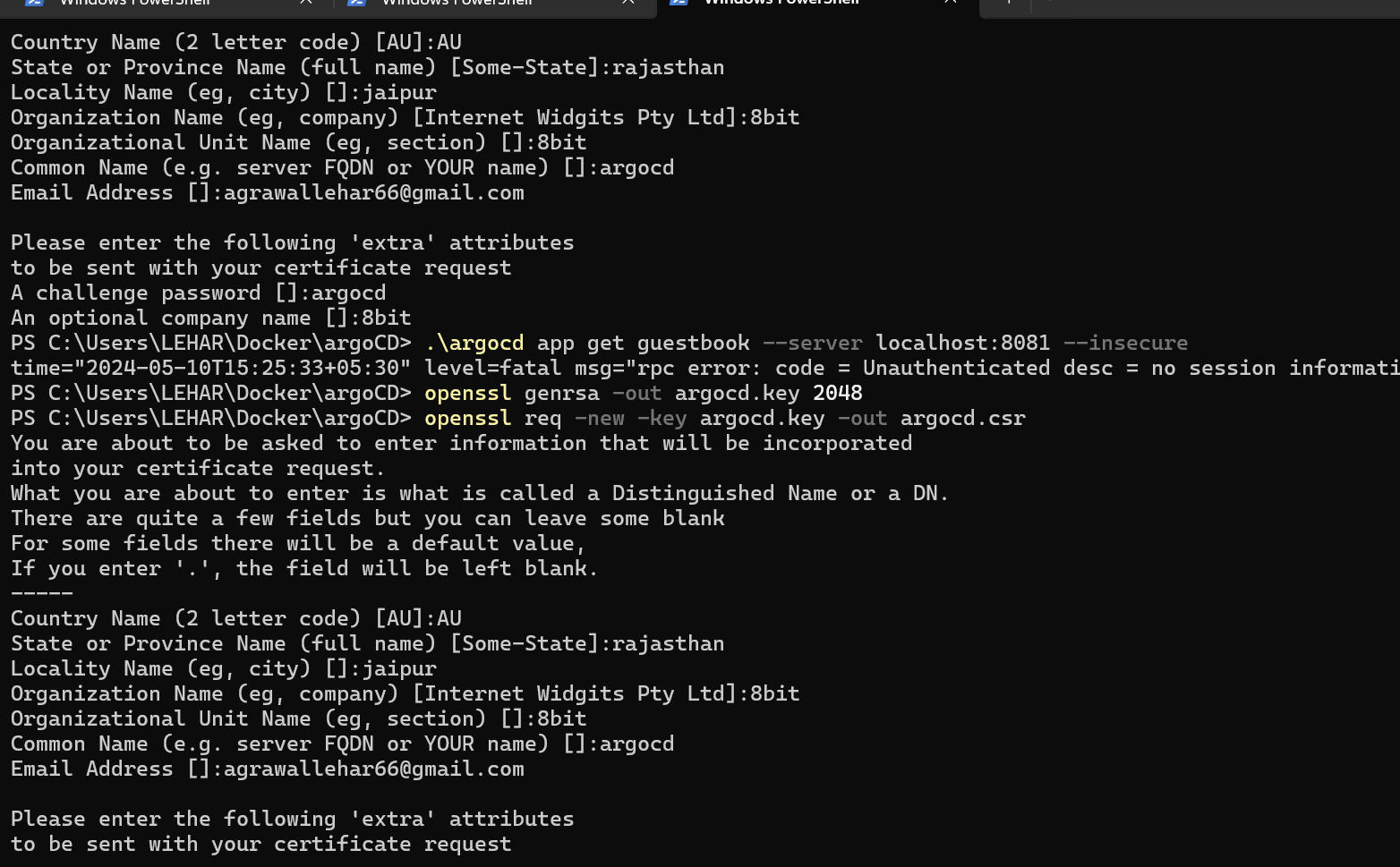
Message: successfully synced (all tasks run)

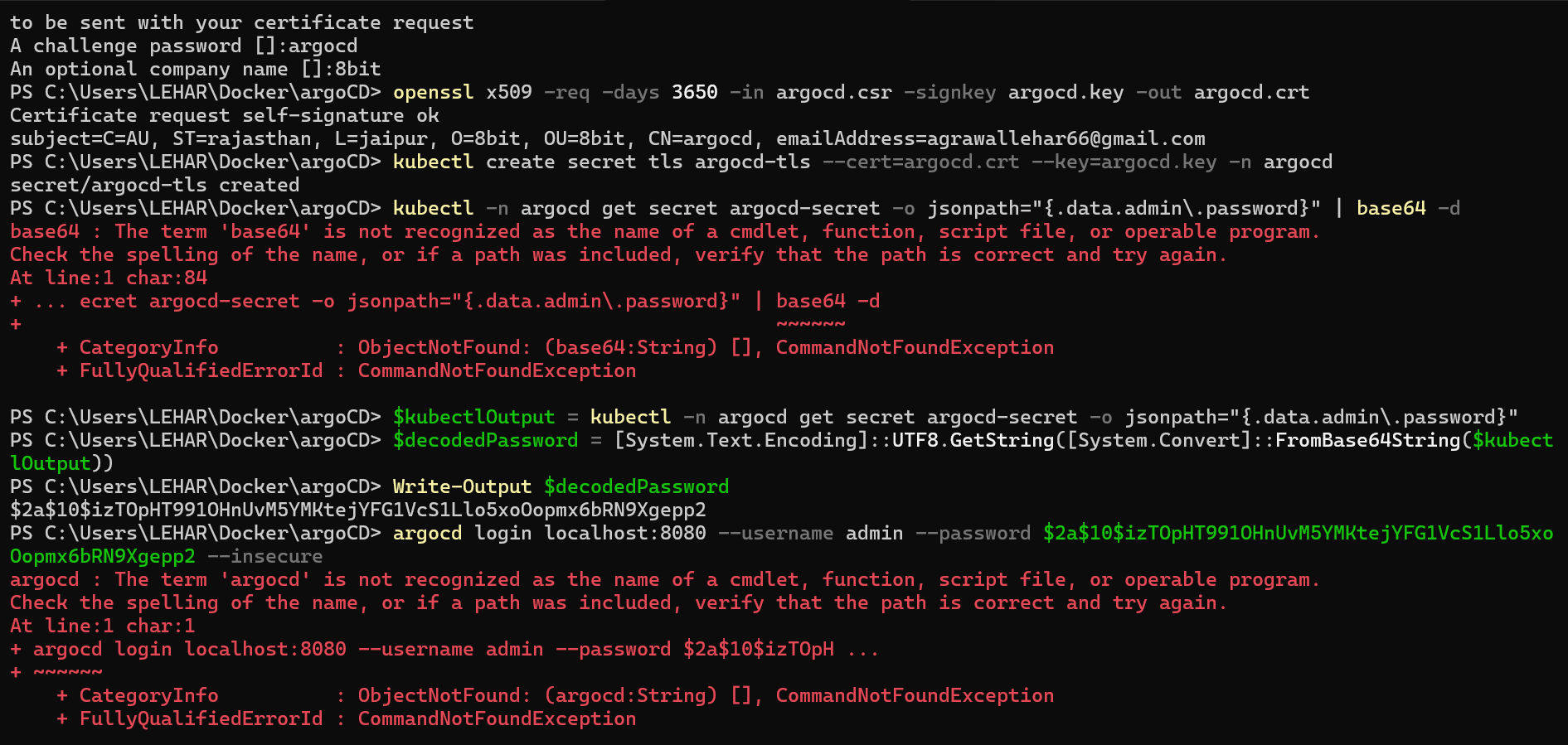
GROUP KIND NAMESPACE NAME STATUS HEALTH HOOK MESSAGE

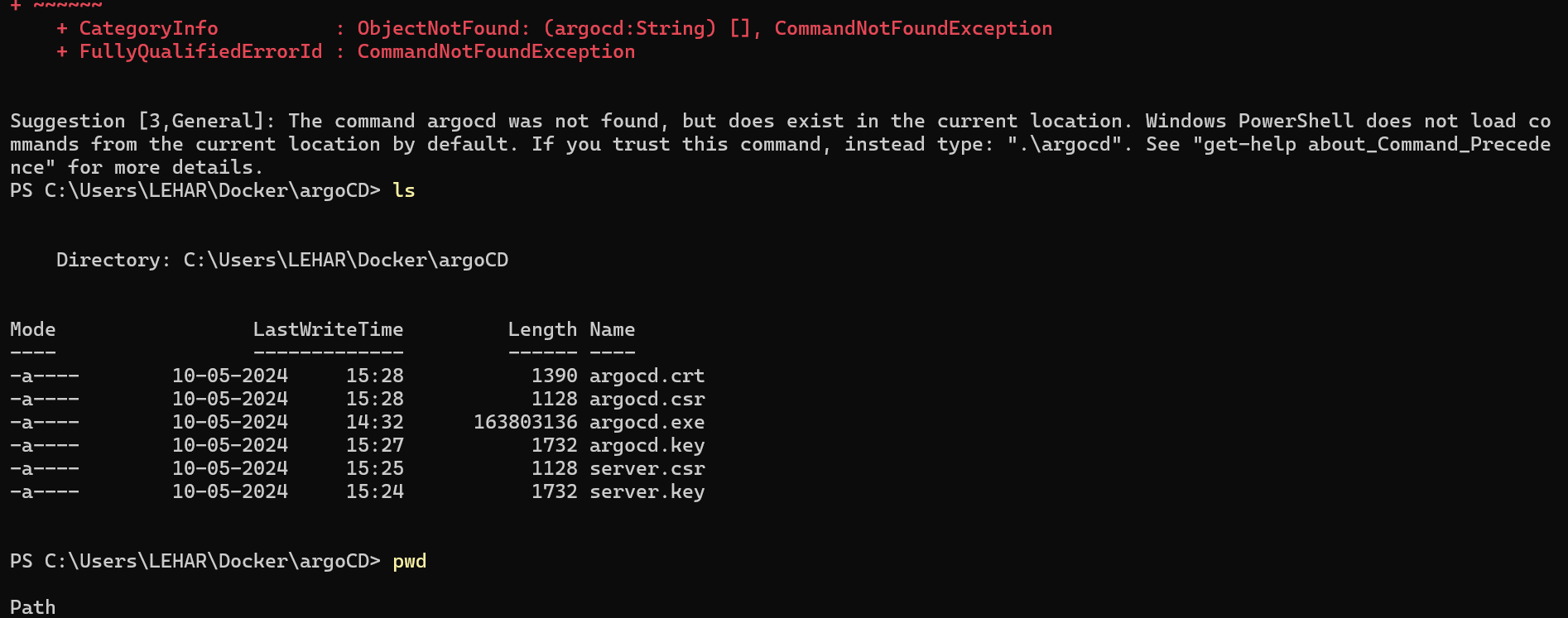
Service default guestbook-ui Synced Healthy service/guestbook-ui created

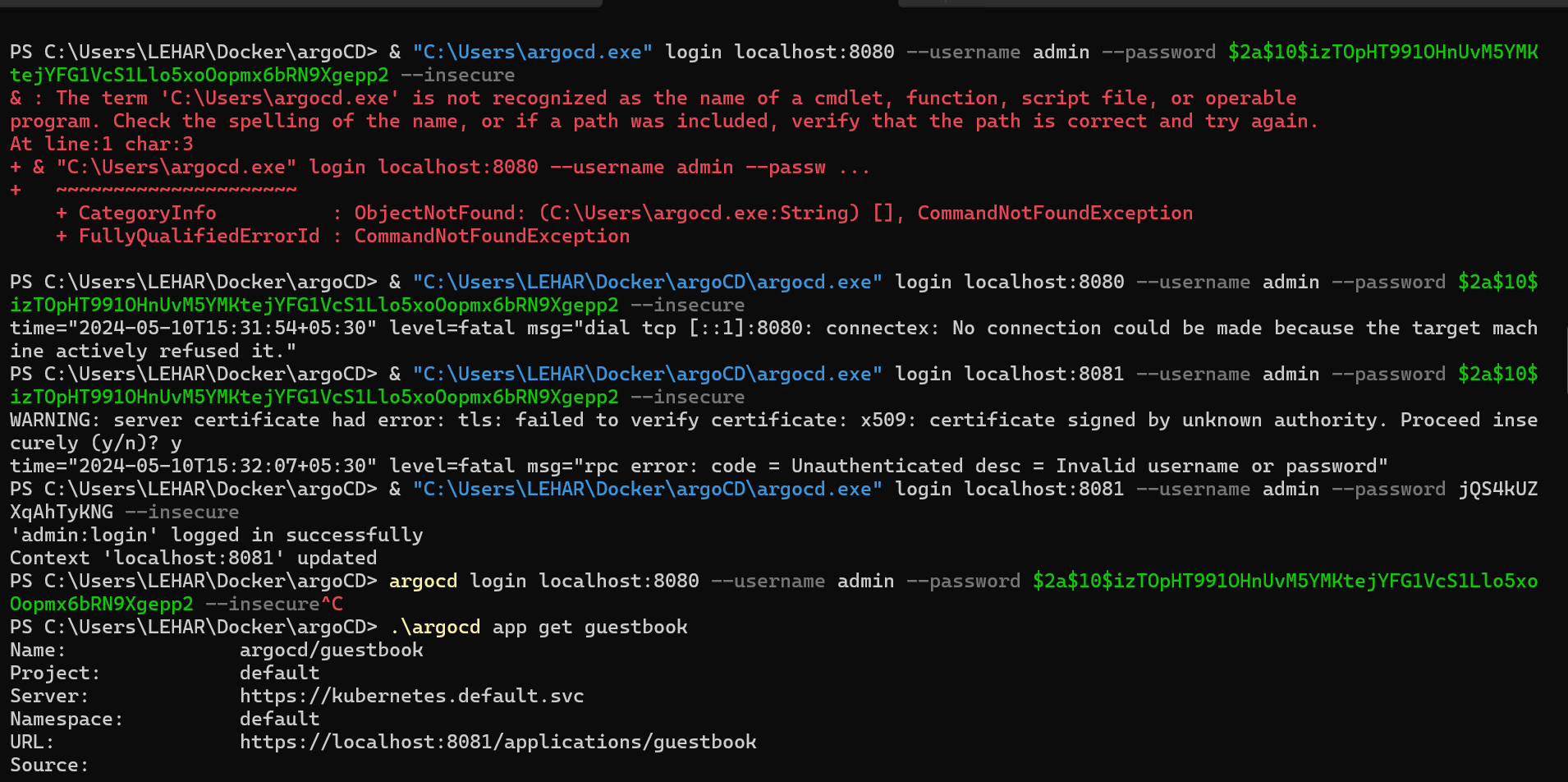
apps Deployment default guestbook-ui Synced Progressing deployment.apps/guestbook-ui created

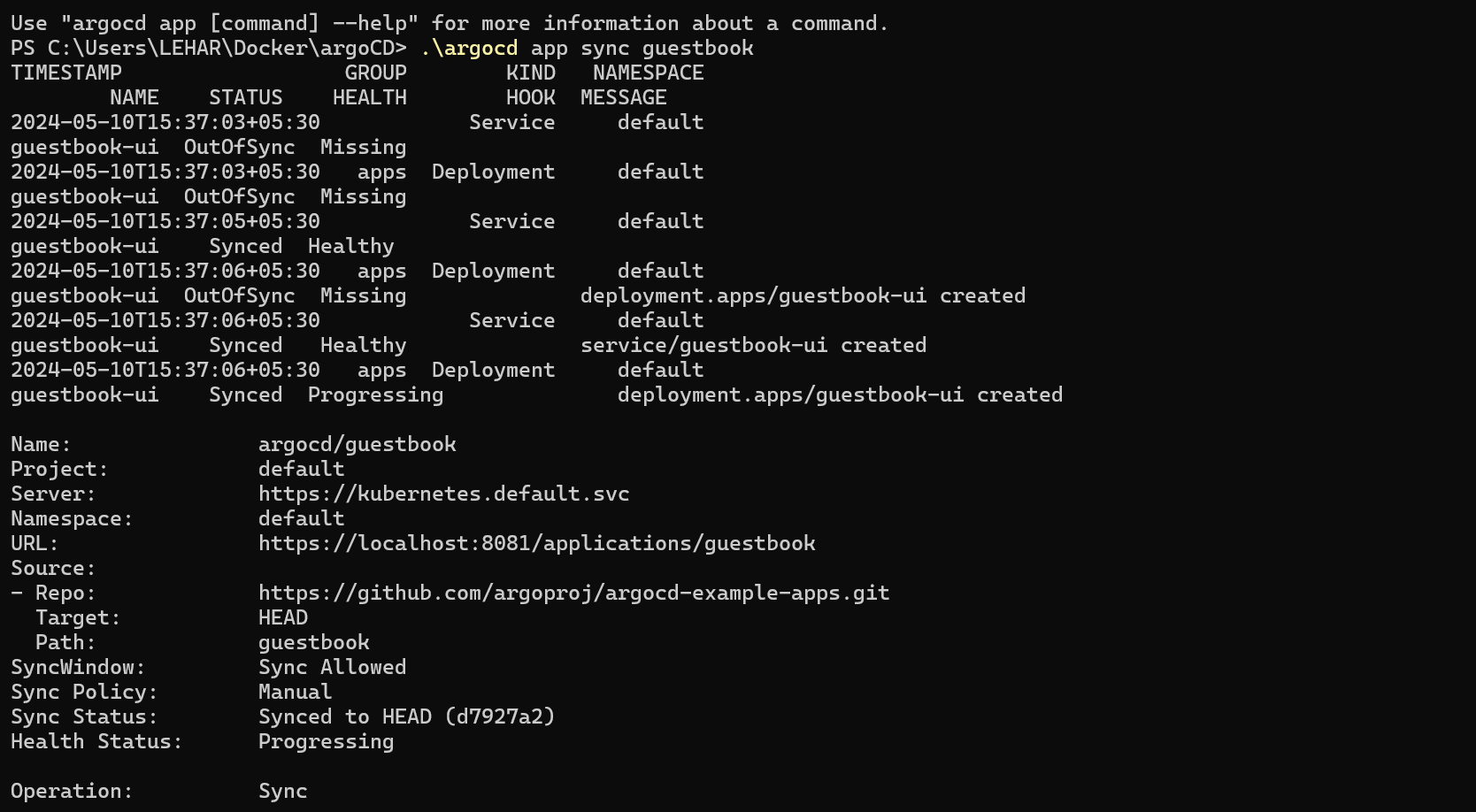




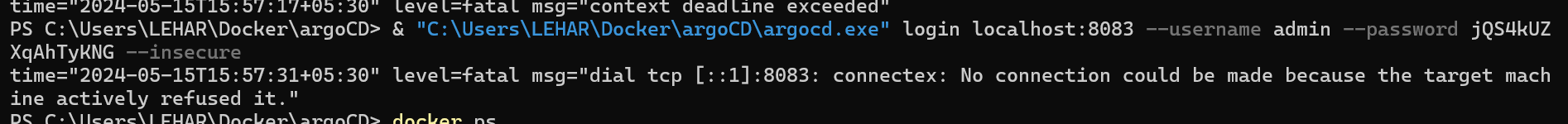




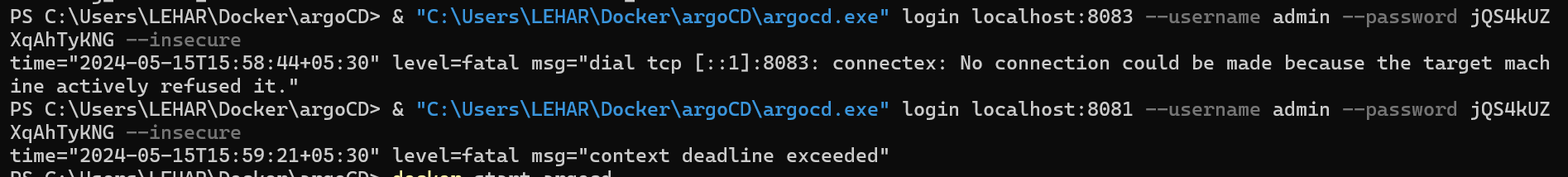




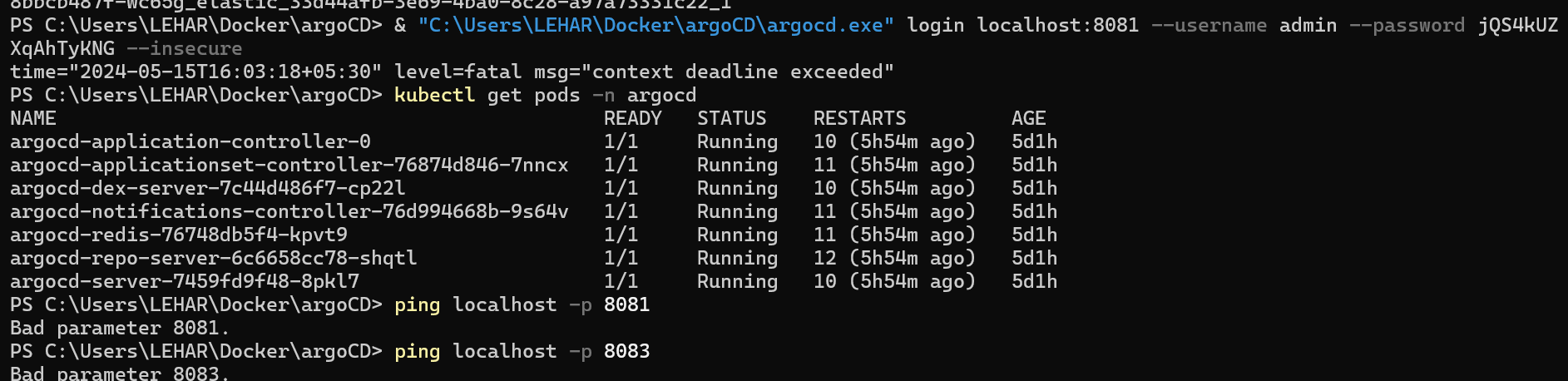
Again when you want to run argoCd



Connection refused that means your port is not acive or sued forward to another port

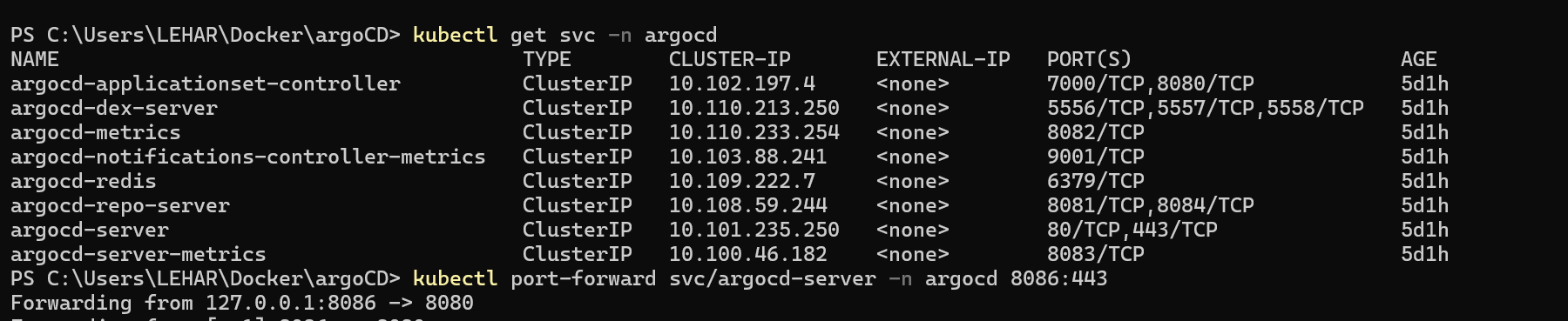


Check pods in argo cd



Tried ping means connection is not established

Thes eprot are used



Forward to another port



PS C:\Users\LEHAR\Docker\argoCD> kubectl port-forward svc/argocd-server -n argocd 8089:443

ArgoCd password

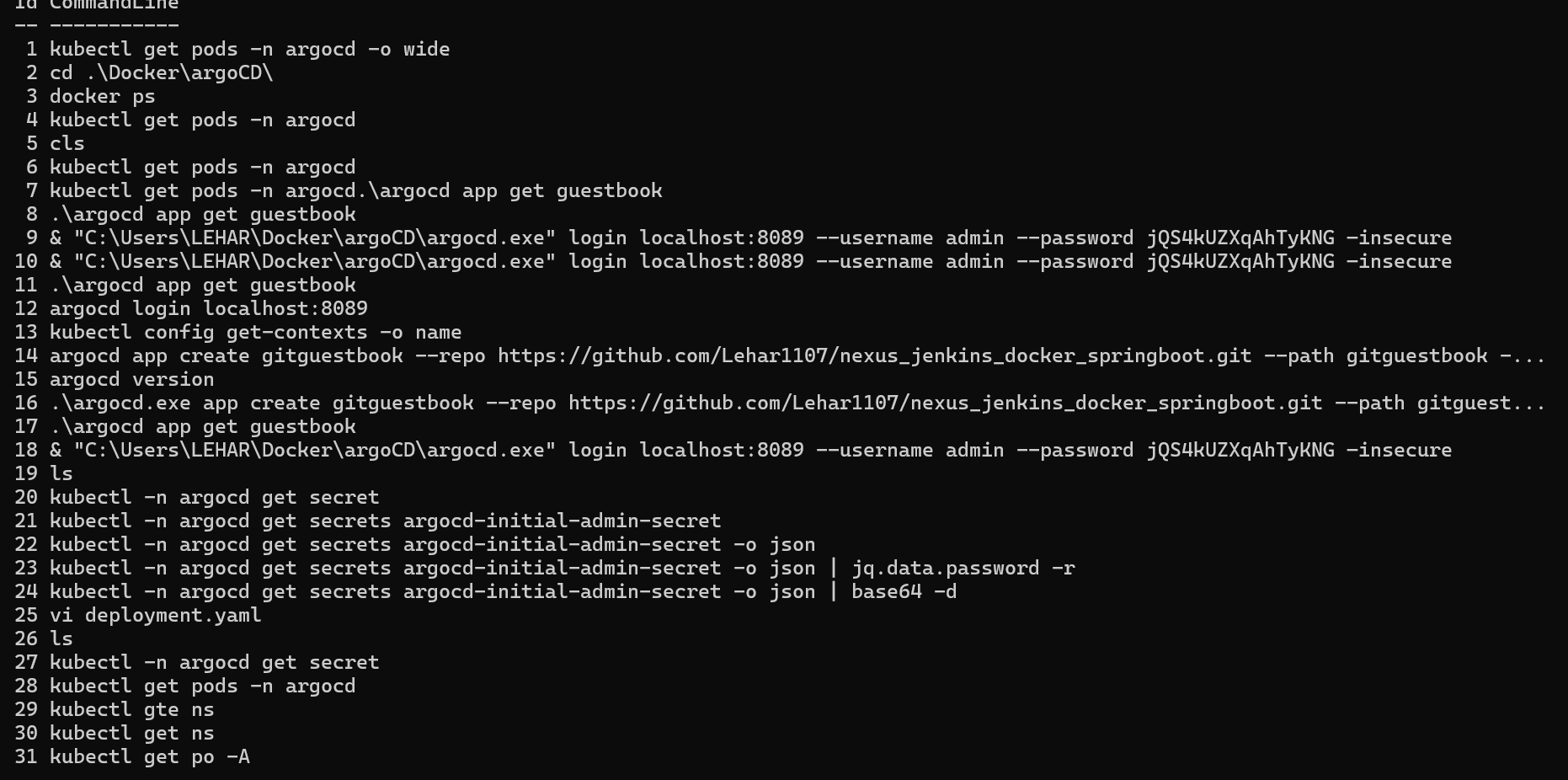
PS C:\Users\LEHAR\Docker\argoCD> & "C:\Users\LEHAR\Docker\argoCD\argocd.exe" login localhost:8089 --username admin --password jQS4kUZXqAhTyKNG –insecure

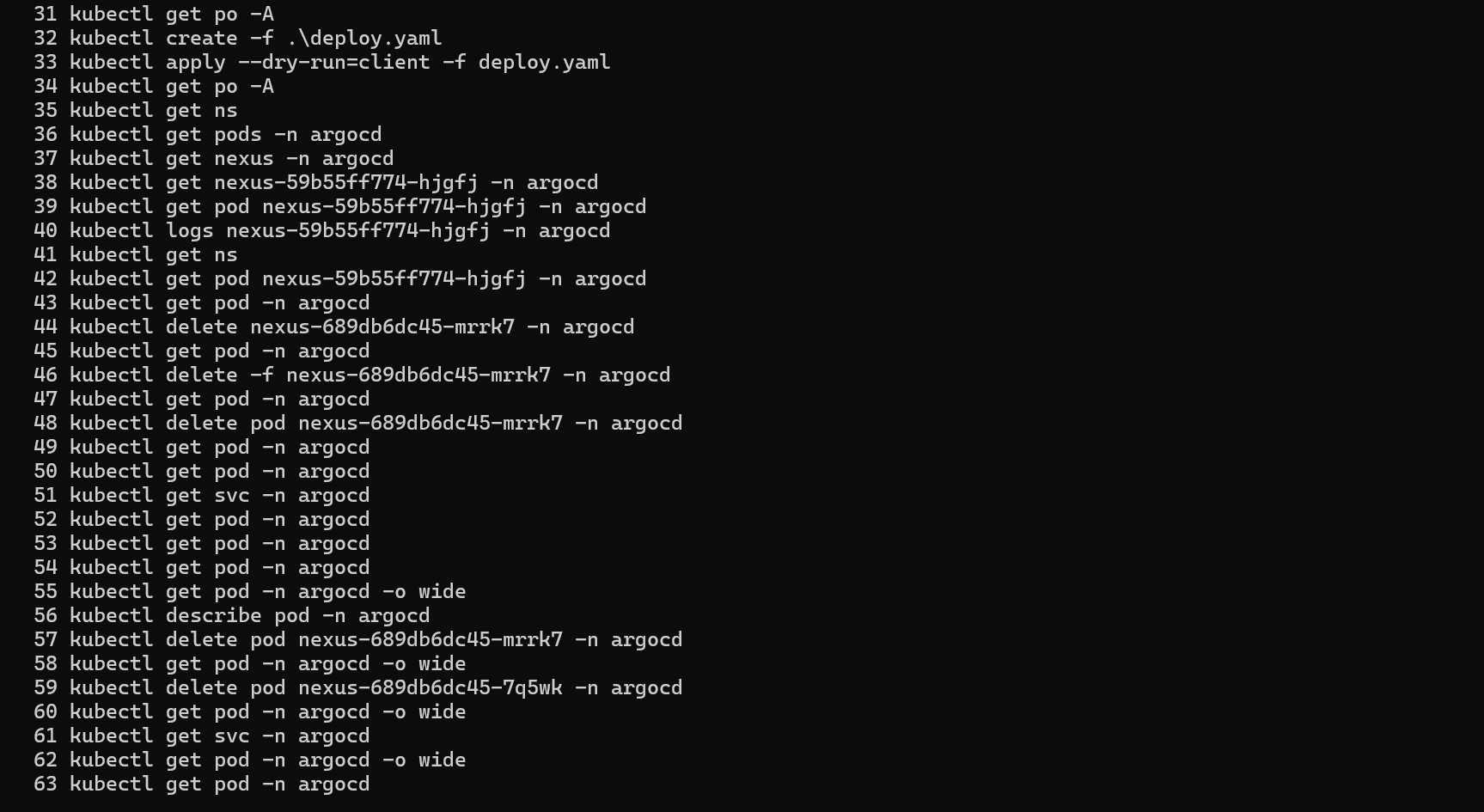
13 argocd login localhost:8080 --username admin --password jQS4kUZXqAhTyKNG --insecure

PS C:\Users\LEHAR\Docker\argoCD> .\argocd app get guestbook

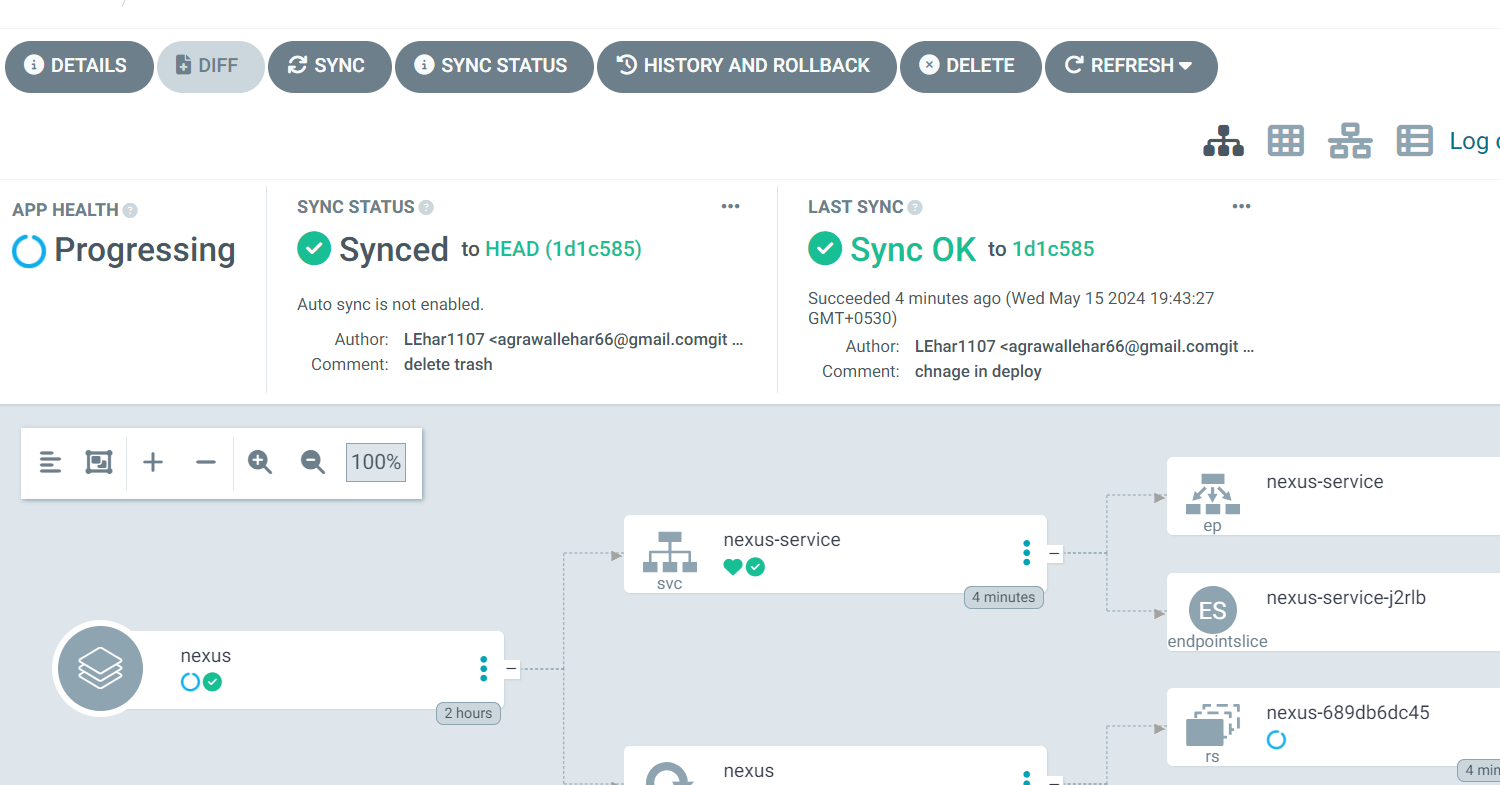
PS C:\Users\LEHAR\Docker\argoCD> .\argocd app sysnc guestbook

Syncd doen









C:\Users\LEHAR\AppData\Local\Temp\kubectl.exe-edit-2450778613.yaml

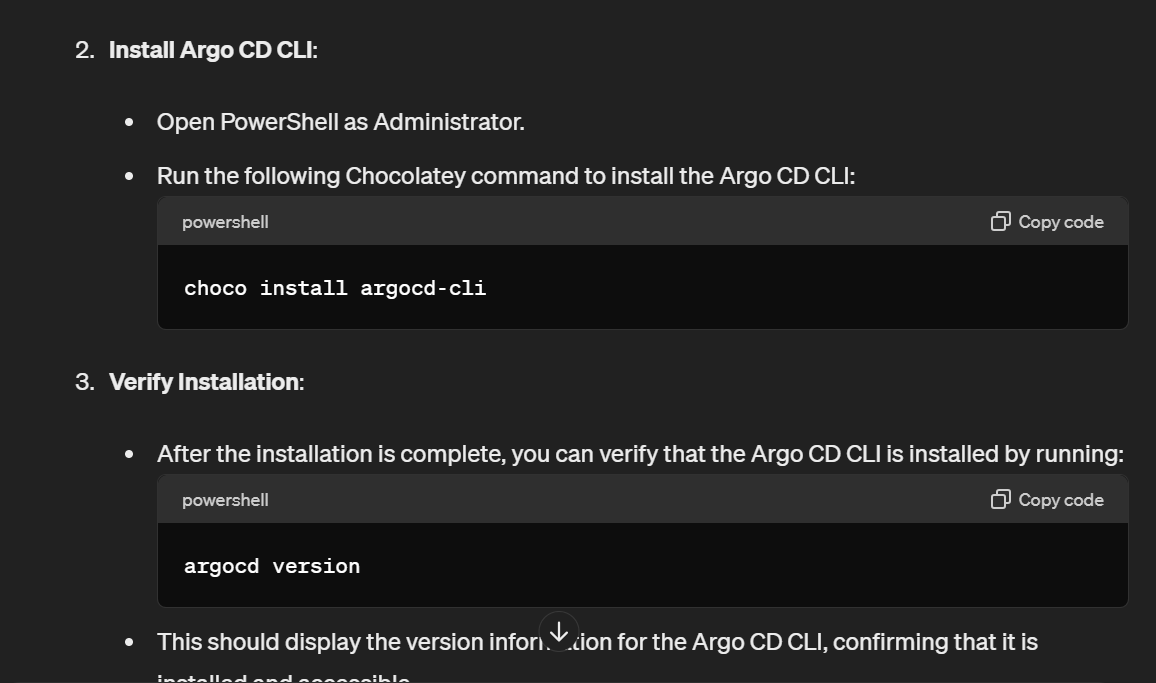
argocd app create my-app --repo https://github.com/Lehar1107/kubernetes.git --dest-server https://kubernetes.default.svc --dest-namespace default

HERE ARGO CD INSTALL ON DOCKER NOW I NEED TO RUN THE ARGOCD COMMANDS ON THE POWERSHELL SO I INTALL CHOCLTEY IF YOU WANT TO ISNTALL IT GO TO SITE AND SEARCH THEN INSTALL ARGOCD COMMANDS ON RUN AS ADMINISTRATOR.

INSTALL ARGOCD COMMANDS ON SYSTEM ADMINISTRATOR POWERSHELL

choco install argocd-cli

argocd version



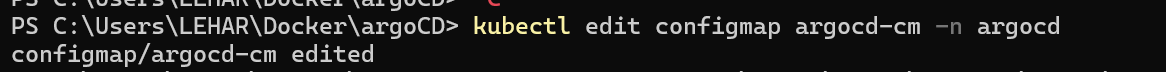
ADD task in argo cd

Add rbac security in argo cd so we need to create user first

Step-1 Edit the config in which user should be added

PS C:\Users\LEHAR\Docker\argoCD> kubectl edit configmap argocd-cm -n argocd

configmap/argocd-cm edited



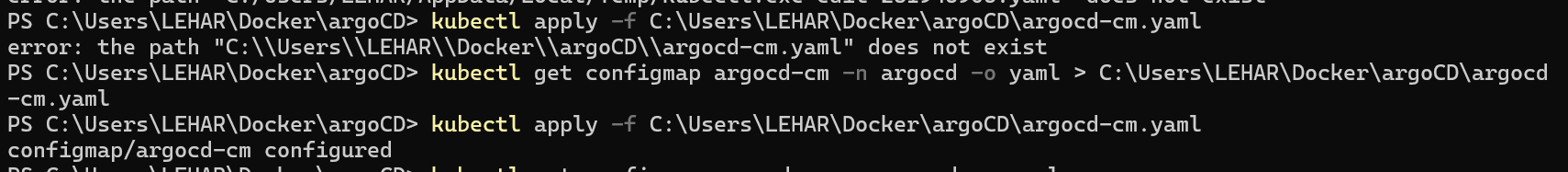
Step-2:

First get then apply else error will occur

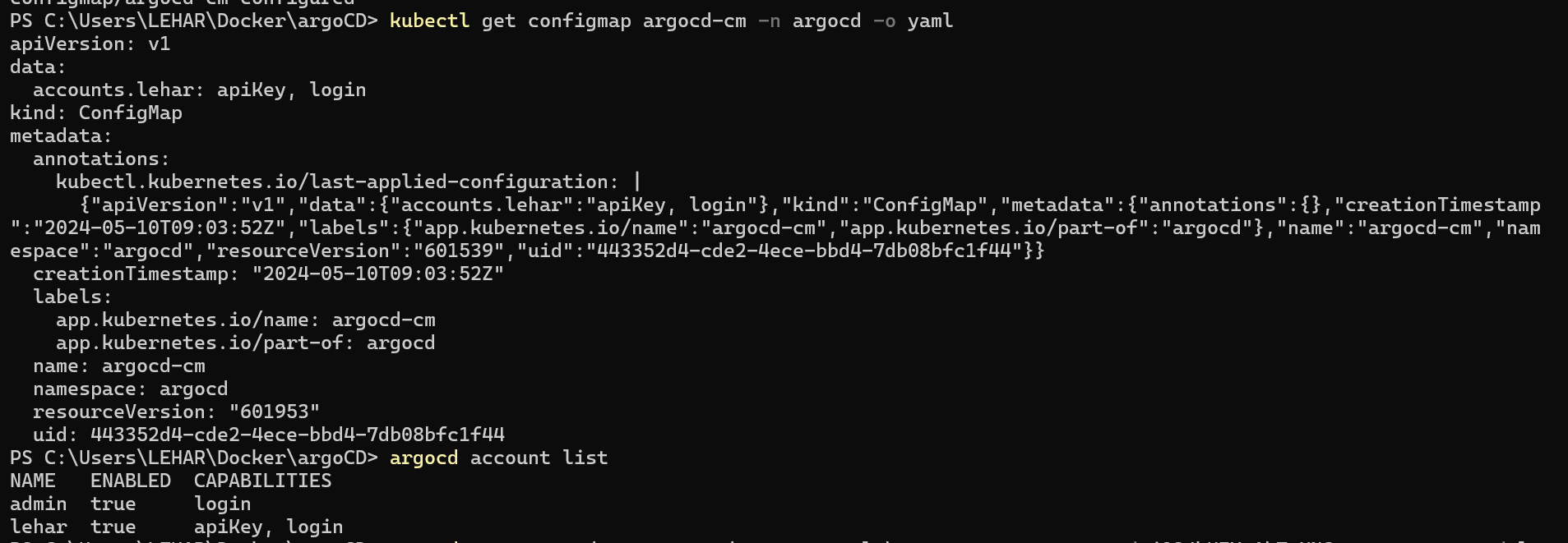
PS C:\Users\LEHAR\Docker\argoCD> kubectl get configmap argocd-cm -n argocd -o yaml > C:\Users\LEHAR\Docker\argoCD\argocd-cm.yaml

PS C:\Users\LEHAR\Docker\argoCD> kubectl apply -f C:\Users\LEHAR\Docker\argoCD\argocd-cm.yaml

configmap/argocd-cm configured



Added the account lehar



Step -4 :

To update your new password for lehar user

argocd account update-password --account lehar --current-password jQS4kUZXqAhTyKNG --new-password lehar1234

<https://www.opsmx.com/blog/how-to-configure-rbac-in-argo-cd/>

TO CREATE USER

<https://foxutech.medium.com/how-to-manage-argocd-rbac-configuration-5d532d72624c>

ADD USER-SERVICE WHAT AN USER CAN DO (LEHAR)

After running the edit command you don’t get the configuration but it will chnge you can add this configuration what I have pasterd below with running commands

PS C:\Users\LEHAR\Docker\argoCD> kubectl edit configmap argocd-rbac-cm -n argocd -o yaml > argocd-rbac-cm.yml

PS C:\Users\LEHAR\Docker\argoCD> kubectl get configmap argocd-rbac-cm -n argocd -o yaml > argocd-rbac-cm.yaml

PS C:\Users\LEHAR\Docker\argoCD> kubectl apply -f argocd-rbac-cm.yaml

configmap/argocd-rbac-cm configured

PS C:\Users\LEHAR\Docker\argoCD> kubectl get configmap argocd-rbac-cm -n argocd -o yaml

apiVersion: v1

data:

policy.csv: |

p, role:lehar, applications, create, \*/\*, allow

p, role:lehar, applications, get, \*/\*, allow

p, role:lehar, applications, sync, \*/\*, allow

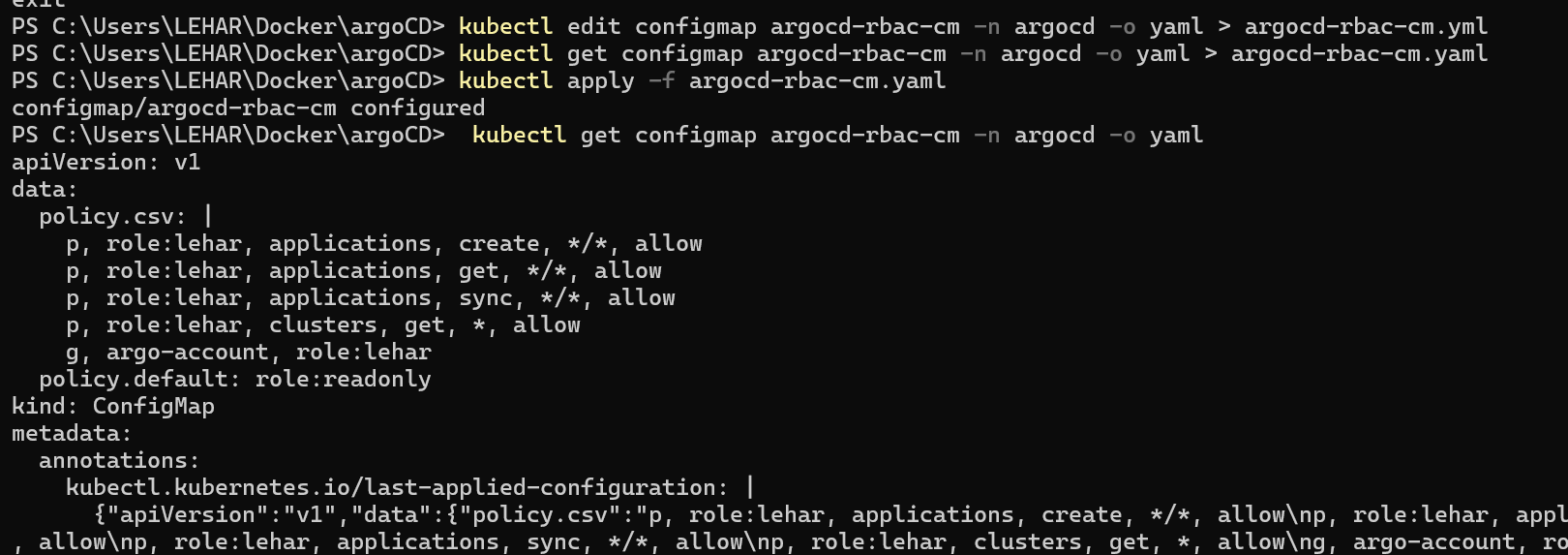
p, role:lehar, clusters, get, \*, allow

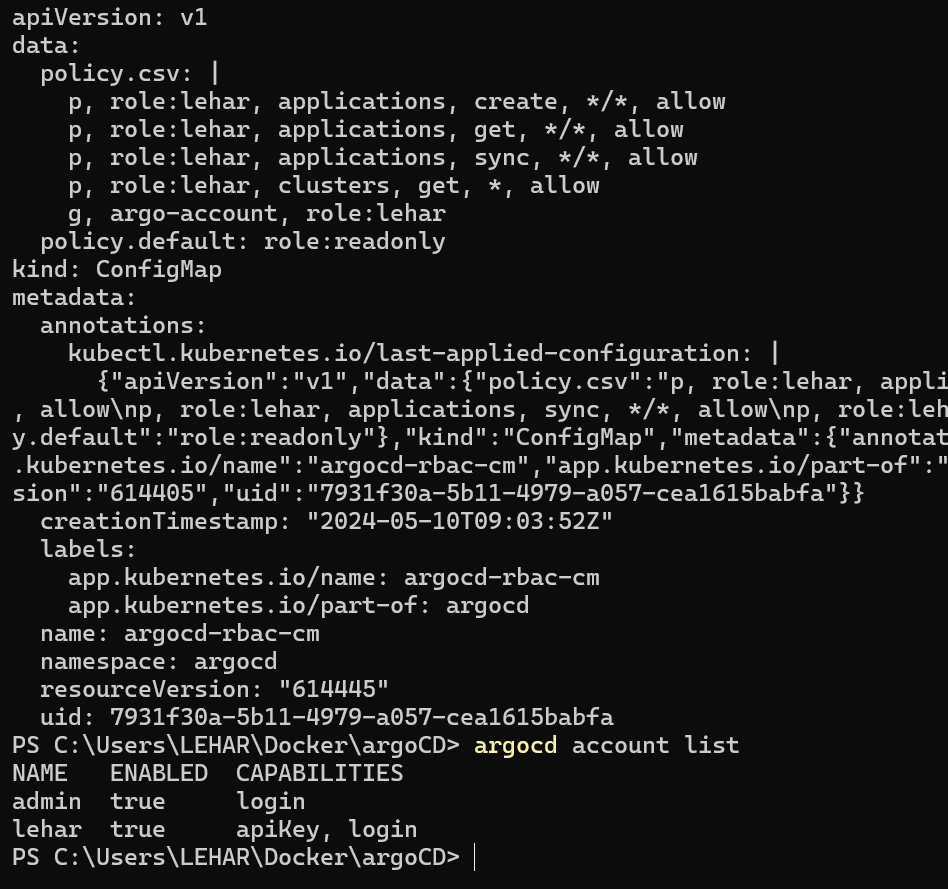
g, argo-account, role:lehar

policy.default: role:readonly

kind: ConfigMap

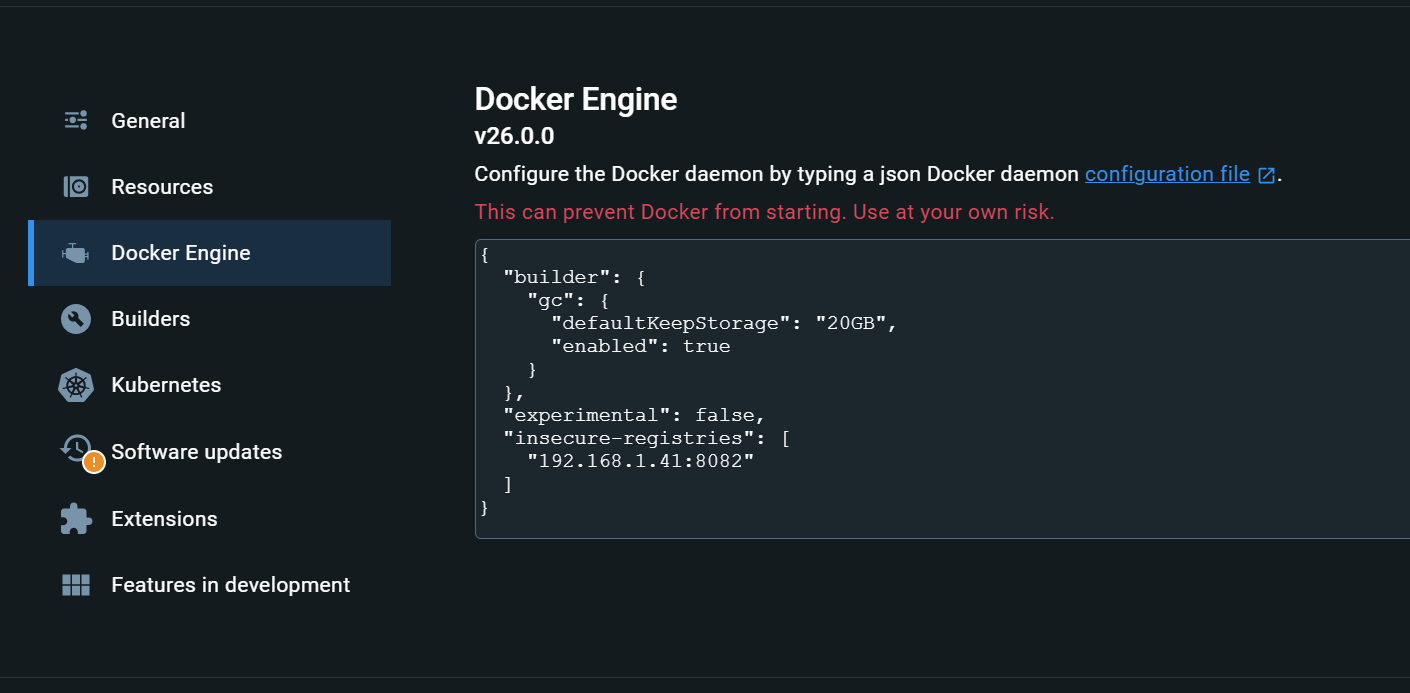
metadata:

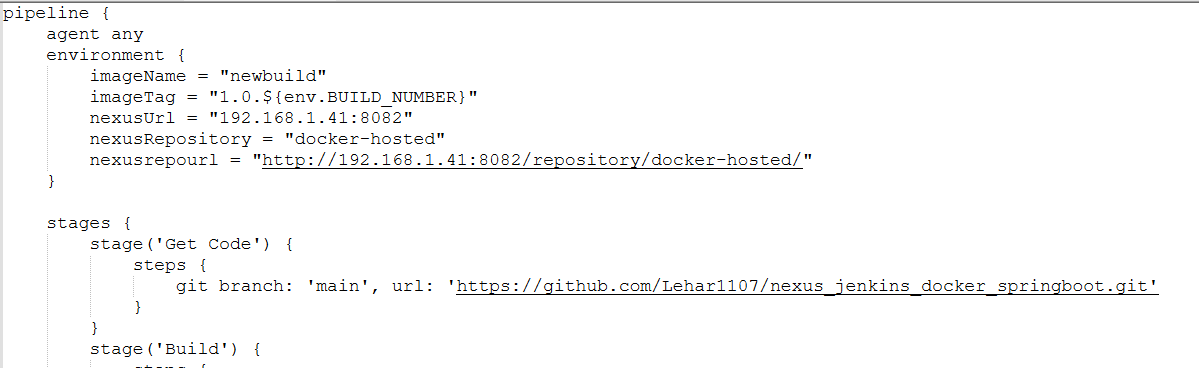




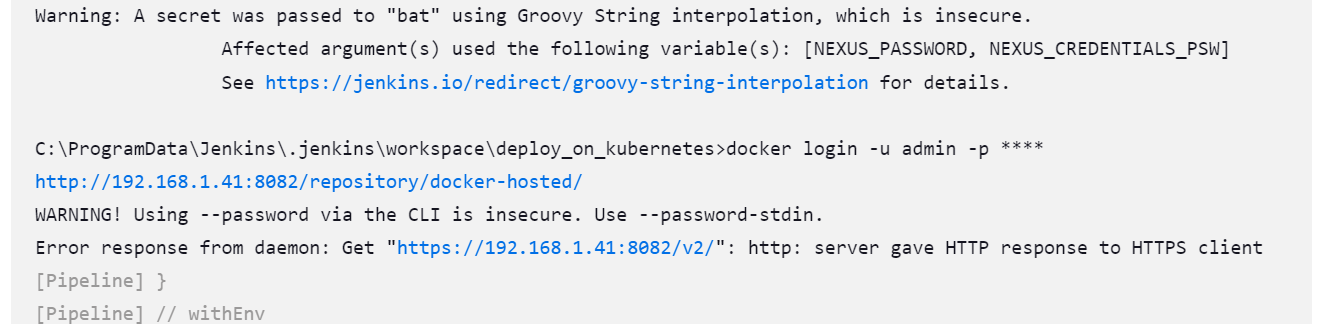
After getting the permission logging in lehr can also see the account list

<https://argo-cd.readthedocs.io/en/stable/operator-manual/rbac/>

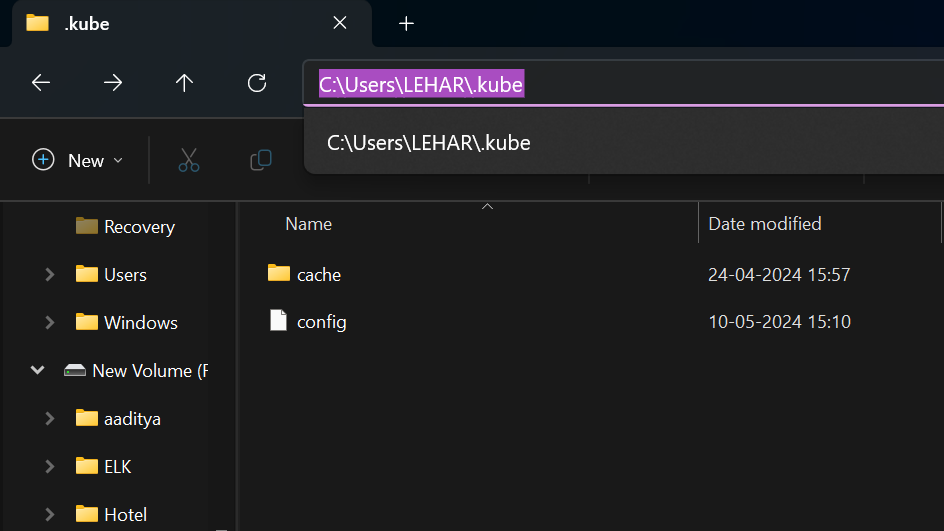
If you connect to another wifi then check the ip address of your laptop  


Also change the adxdres in docker desktop  


Here also

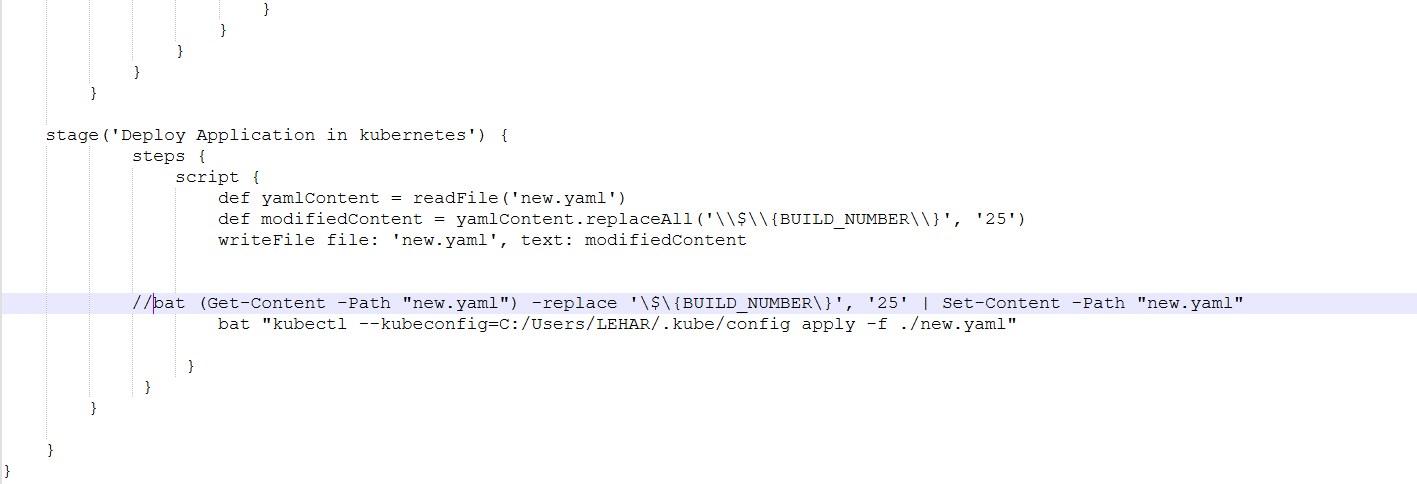
Else this errror will occur 

After giving the path of conf kubernetes yaml wil run make sure that you have its backup

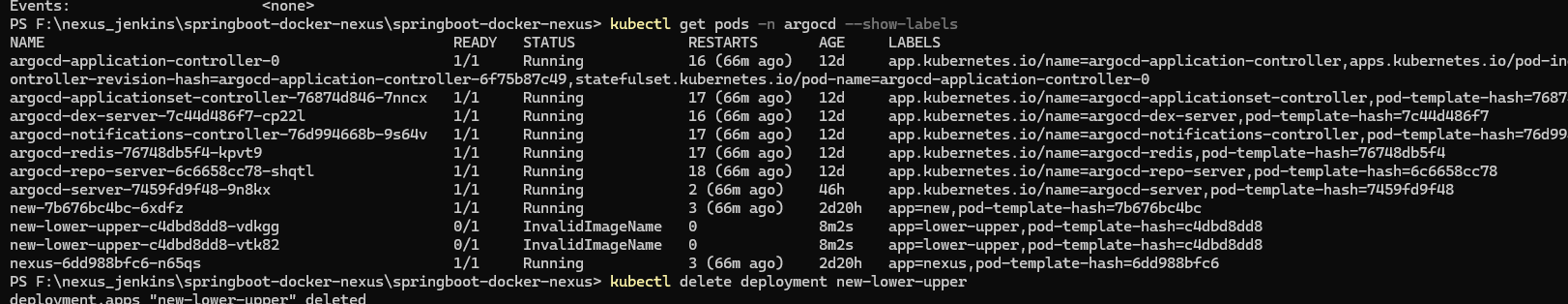


This is the path of your local

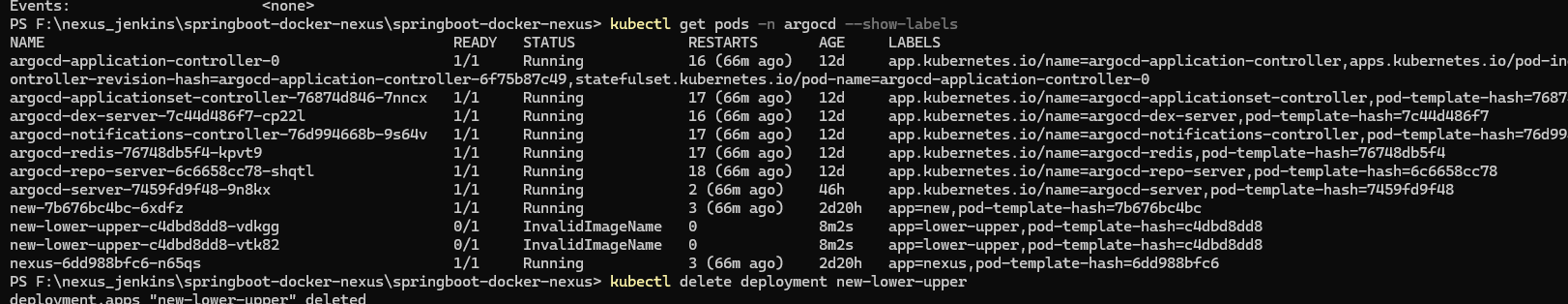
New stage is added jenkins



Pod is deployed in kubernetes cluster



But image is not ruuning so you need to add that above buildno path nd all you have to wirte to overcome this error



If you add the code

def yamlContent = readFile('new.yaml')

def modifiedContent = yamlContent.replaceAll('\\$\\{BUILD\_NUMBER\\}', '25')

writeFile file: 'new.yaml', text: modifiedContent

then your pod is running this is all for windows poweshell in linux you need to change

now thus change I can’t see in argocd let’s see what to do

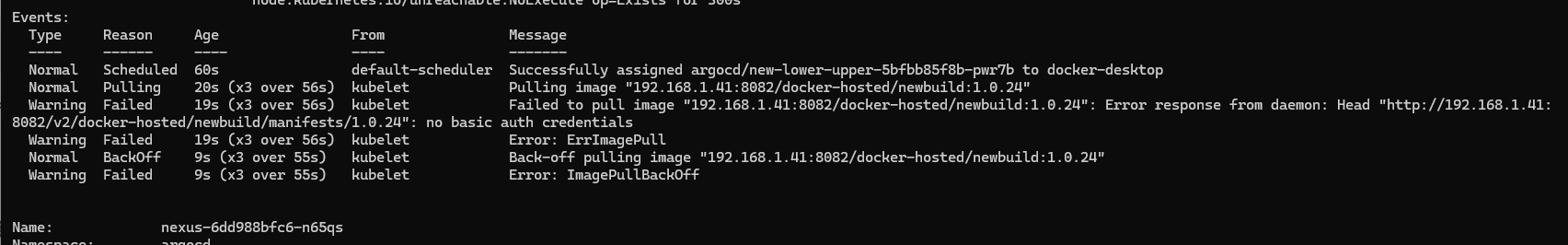


Image was created but Kubernetes can’t pull it from nextus

Need to create secret

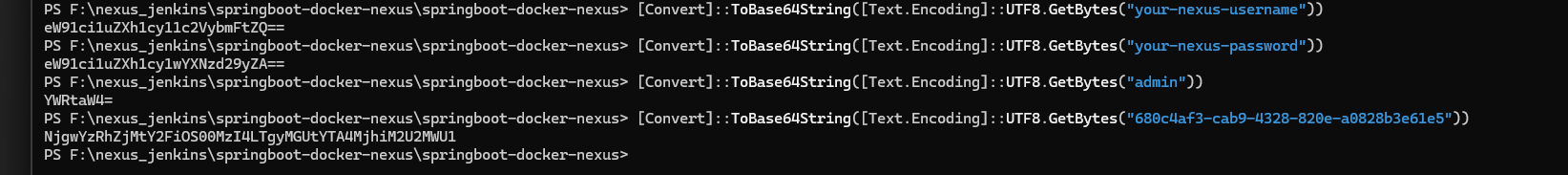
PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes("admin"))

YWRtaW4=

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes("680c4af3-cab9-4328-820e-a0828b3e61e5"))

NjgwYzRhZjMtY2FiOS00MzI4LTgyMGUtYTA4MjhiM2U2MWU1

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus>



Need to create secret in windows powershell

$Username = 'admin'

$EncodedUsername = [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes($Username))

Write-Output $EncodedUsername

YWRtaW4=

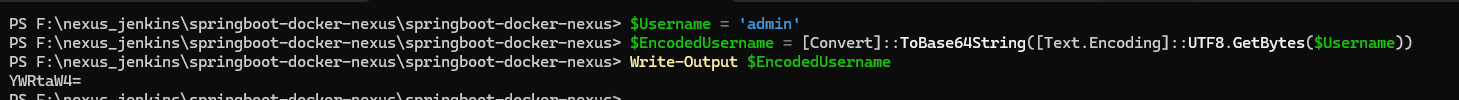
PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes('680c4af3-cab9-4328-820e-a0828b3e61e5'))

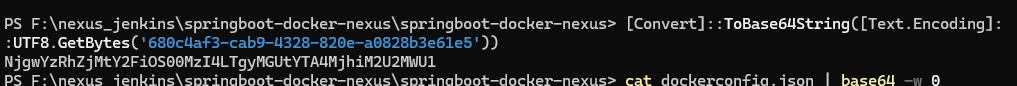
NjgwYzRhZjMtY2FiOS00MzI4LTgyMGUtYTA4MjhiM2U2MWU1

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> $Base64String = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes((Get-Content -Path "dockerconfig.json" -Raw)))

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> Write-Output $Base64String

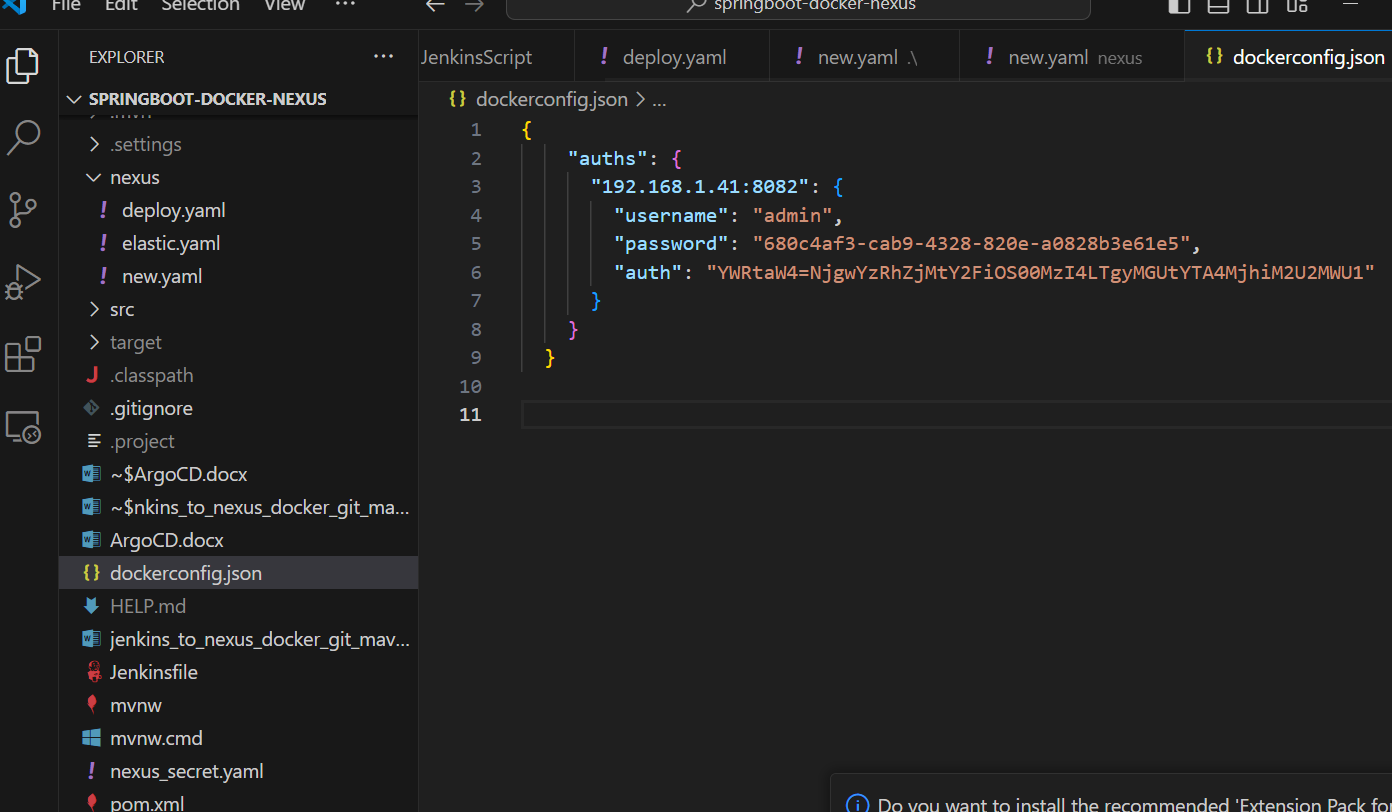
ew0KICAgICJhdXRocyI6IHsNCiAgICAgICIxOTIuMTY4LjEuNDE6ODA4MiI6IHsNCiAgICAgICAgInVzZXJuYW1lIjogImFkbWluIiwNCiAgICAgICAgInBhc3N3b3JkIjogIjY4MGM0YWYzLWNhYjktNDMyOC04MjBlLWEwODI4YjNlNjFlNSIsDQogICAgICAgICJhdXRoIjogIllXUnRhVzQ9Tmpnd1l6Umhaak10WTJGaU9TMDBNekk0TFRneU1HVXRZVEE0TWpoaU0yVTJNV1UxIg0KICAgICAgfQ0KICAgIH0NCiAgfQ0KDQogIA==





After turing into base64

Crate dockerconfig.json file



Write this password

{

    "auths": {

      "192.168.1.41:8082": {

        "username": "admin",

        "password": "680c4af3-cab9-4328-820e-a0828b3e61e5",

        "auth": "YWRtaW4=NjgwYzRhZjMtY2FiOS00MzI4LTgyMGUtYTA4MjhiM2U2MWU1"

      }

    }

  }

Firstly run these commands

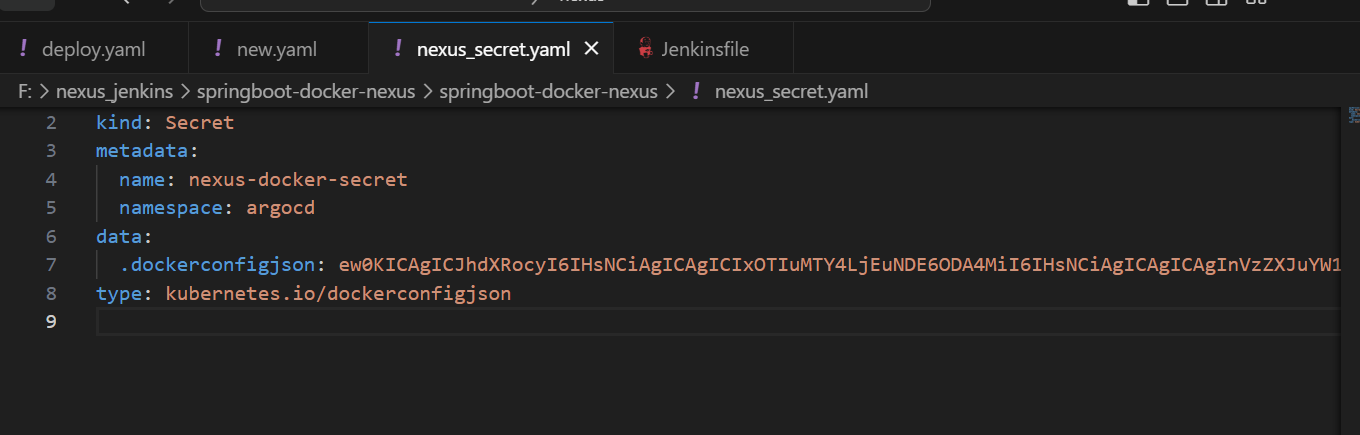
PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> $Base64String = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes((Get-Content -Path "dockerconfig.json" -Raw)))

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> Write-Output $Base64String

ew0KICAgICJhdXRocyI6IHsNCiAgICAgICIxOTIuMTY4LjEuNDE6ODA4MiI6IHsNCiAgICAgICAgInVzZXJuYW1lIjogImFkbWluIiwNCiAgICAgICAgInBhc3N3b3JkIjogIjY4MGM0YWYzLWNhYjktNDMyOC04MjBlLWEwODI4YjNlNjFlNSIsDQogICAgICAgICJhdXRoIjogIllXUnRhVzQ9Tmpnd1l6Umhaak10WTJGaU9TMDBNekk0TFRneU1HVXRZVEE0TWpoaU0yVTJNV1UxIg0KICAgICAgfQ0KICAgIH0NCiAgfQ0KDQogIA==

the key is generated and write it in secret of kubenetetes

Create a secrete file of kuberneytes



apiVersion: v1

kind: Secret

metadata:

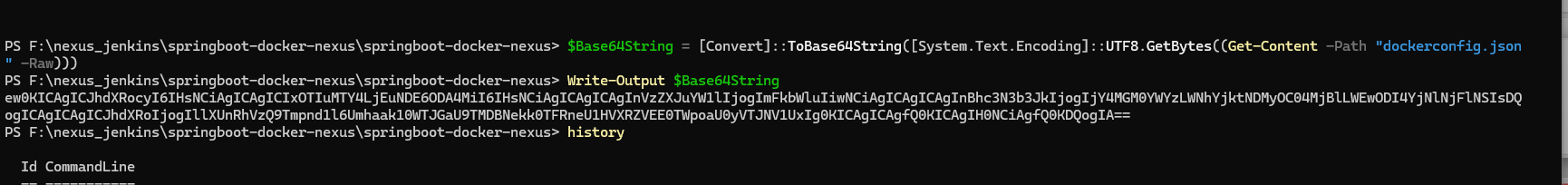
  name: nexus-docker-secret

  namespace: argocd

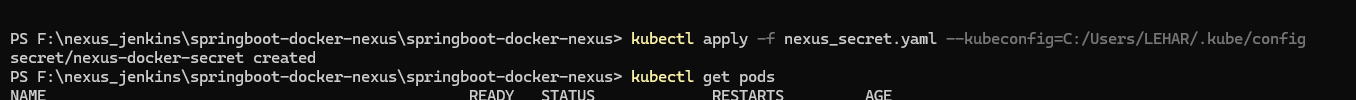
data:

  .dockerconfigjson: ew0KICAgICJhdXRocyI6IHsNCiAgICAgICIxOTIuMTY4LjEuNDE6ODA4MiI6IHsNCiAgICAgICAgInVzZXJuYW1lIjogImFkbWluIiwNCiAgICAgICAgInBhc3N3b3JkIjogIjY4MGM0YWYzLWNhYjktNDMyOC04MjBlLWEwODI4YjNlNjFlNSIsDQogICAgICAgICJhdXRoIjogIllXUnRhVzQ9Tmpnd1l6Umhaak10WTJGaU9TMDBNekk0TFRneU1HVXRZVEE0TWpoaU0yVTJNV1UxIg0KICAgICAgfQ0KICAgIH0NCiAgfQ0KDQogIA==

type: kubernetes.io/dockerconfigjson

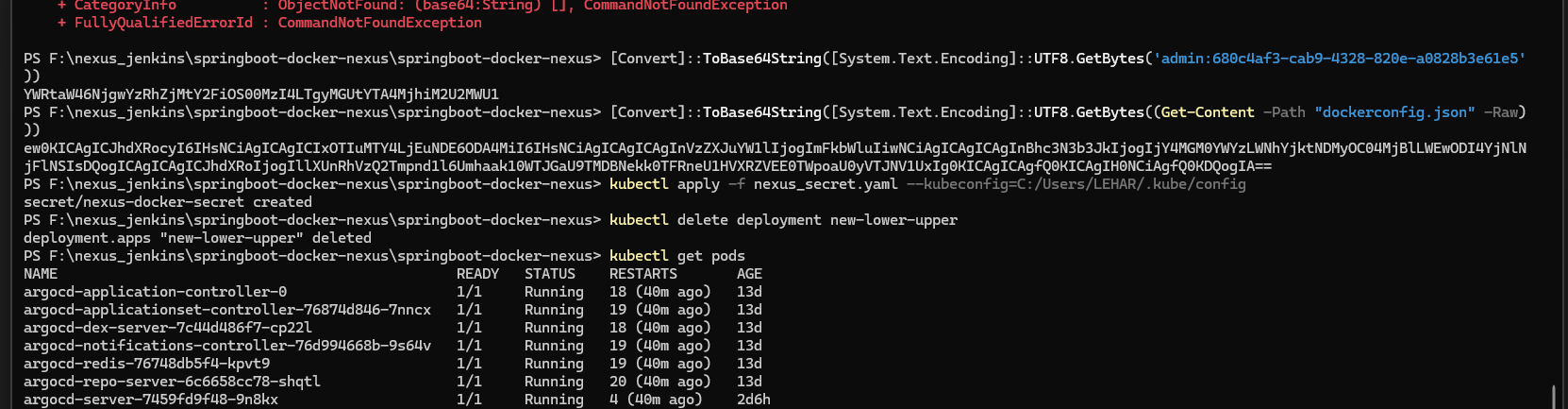


Thenk apply it



$Base64String = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes((Get-Content -Path "dockerconfig.json" -Raw)))

Write-Output $Base64String



PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes('admin:680c4af3-cab9-4328-820e-a0828b3e61e5'))

YWRtaW46NjgwYzRhZjMtY2FiOS00MzI4LTgyMGUtYTA4MjhiM2U2MWU1

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes((Get-Content -Path "dockerconfig.json" -Raw)))

ew0KICAgICJhdXRocyI6IHsNCiAgICAgICIxOTIuMTY4LjEuNDE6ODA4MiI6IHsNCiAgICAgICAgInVzZXJuYW1lIjogImFkbWluIiwNCiAgICAgICAgInBhc3N3b3JkIjogIjY4MGM0YWYzLWNhYjktNDMyOC04MjBlLWEwODI4YjNlNjFlNSIsDQogICAgICAgICJhdXRoIjogIllXUnRhVzQ2Tmpnd1l6Umhaak10WTJGaU9TMDBNekk0TFRneU1HVXRZVEE0TWpoaU0yVTJNV1UxIg0KICAgICAgfQ0KICAgIH0NCiAgfQ0KDQogIA==

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> kubectl apply -f nexus\_secret.yaml --kubeconfig=C:/Users/LEHAR/.kube/config

secret/nexus-docker-secret created

PS F:\nexus\_jenkins\springboot-docker-nexus\springboot-docker-nexus> kubectl delete deployment new-lower-upper

deployment.apps "new-lower-upper" deleted