Practical 1:

Linux Commands

a) Linux Directory Commands

Command: pwd

Purpose: Shows the current working directory.

syntax

pwd

Example Output: /home/user

Command: mkdir

Purpose: Creates a new directory.

syntax

mkdir new_directory

Check: Use ls to see the directory in the current folder.

Command: rm -rf

Purpose: Deletes a directory and its contents recursively.

syntax

rm -rf new_directory

mkdir /home/runner/prac1/folder1 /home/runner/prac1/folder2

Caution: Be careful while using this command as it permanently deletes the directory.

Command: ls

Purpose: Lists files and directories in the current directory.

syntax

ls

Command: cd

Purpose: Changes the current directory.

syntax

cd new_directory

cd /home/runner/prac1/folder1

pwd

cd /home/runner/prac1/folder2

pwd

cd -

Command: cd -

Purpose: Switches back to the previous directory.

syntax

cd -

b) Linux File Commands

Command: touch

Purpose: Creates an empty file.

syntax

touch file.txt

Command: cat

Purpose: Displays the content of a file.

syntax

cat file.txt

Command: rm

Purpose: Deletes a file.

syntax

rm file.txt

Command: cp

Purpose: Copies a file from one location to another.

syntax

cp file.txt /path/to/destination/

cp fol1/file1.txt /folder1

Command: mv

Purpose: Moves or renames a file.

syntax(Move):

mv file.txt /path/to/destination/

mv folder1/file1.txt /folder2/

syntax (Rename):

mv old_name.txt new_name.txt

Command: rename

Purpose: Renames files based on a pattern.

syntax

rename 's/old/new/' *.txt

This will rename all .txt files where "old" is replaced by "new."

c) Linux Permission Commands

Command: su

Purpose: Switches to the superuser account.

syntax

su

Command: id

Purpose: Displays the user ID and group ID.

syntax

id

Command: useradd

Purpose: Adds a new user.

syntax

sudo useradd new_user

Command: passwd

Purpose: Changes or sets a user's password.

syntax

sudo passwd new_user

Command: groupadd

Purpose: Adds a new group.

syntax

sudo groupadd new_group

Command: chmod

Purpose: Changes file permissions.

7=The owner of the file can read, write, and execute it.

5=The group members associated with the file can read and execute it (but cannot write).

5=All other users can read and execute it (but cannot write).

syntax

chmod 755 file.txt

change mode=chmod

This sets read, write, and execute permissions for the owner and read/execute for others.

Command: chown

Purpose: Changes the owner of a file.

syntax

sudo chown user file.txt

By running sudo chown user file.txt, you're making user the owner of file.txt. This means that user will have full control over the file's permissions (like reading, writing, and executing it), assuming the file permissions allow it.

Why Use It?

d) Linux File Content & Filter Commands

Command: head

Purpose: Displays the first 10 lines of a file.

syntax

head file.txt

Command: tail

Purpose: Displays the last 10 lines of a file.

syntax

tail file.txt

Command: grep full form=Global Regular Expression Print

Purpose: Searches for a specific pattern in a file.

syntax

grep 'pattern' file.txt

e) Linux Utility Commands

Command: find

Purpose: Searches for files in a directory hierarchy.

syntax

find /path/to/search -name "filename"

Command: locate

Purpose: Locates files by name.

It relies on a database that indexes files on your system, which needs to be built and updated regularly.

syntax

locate filename

Command: df

Purpose: Displays disk space usage.

The df command stands for Disk Free.-h: This stands for "human-readable" format. It makes the output

easier to understand by displaying the disk space in a more user-friendly way, such as in GB, MB, or KB, rather than just in bytes.

syntax

df -h

f) Linux Networking Commands

Command: ip

Purpose: Displays IP address and network configuration.

syntax

ip addr show

Command: ping

Purpose: Tests network connectivity.

The ping command is used to test the connectivity between your system and a remote server or device over a network (like the internet). It sends small packets of data to the specified address and waits for a response. If it receives a response, it indicates that there is network connectivity between your system and the target.

syntax

ping google.com

g) Editing Crontab for Scheduling

Command: crontab -e

Purpose: Edit the crontab file to schedule tasks.

Example: Schedule a system-wide message at 10 AM daily:

wall "Hello, this is a scheduled message."

0 6 * * * /path/to/your/script.sh

Iska matlab hoga ki /path/to/your/script.sh script har din subah 6:00 baje run hoga.

h) Using the vi Editor=vi editor usig guidelines=used to edit files using command line.

Open a file in vi:

vi filename.txt

Insert text:

Press i to go into insert mode.

Type your text.

Save and quit:

Press Esc to exit insert mode.

Type :wq to save and exit.

Search for a term:

Press / followed by the term you want to search.

Press n to jump to the next occurrence.

Practical 2: Shell Scripting

a) Shell Script to Print a Message

#!/bin/bash

The line #!/bin/bash is called a shebang, which tells the system that this script should be run in the Bash shell. The echo command prints a message that you can see on the terminal.

The term "shebang" comes from combining "sharp" (#) and "bang" (!). It's a special character sequence (#!) at the beginning of a script that tells the operating system which interpreter (like Bash, Python, etc.) to use to execute the file.

Create the script:

code

nano message.sh

Write the script:

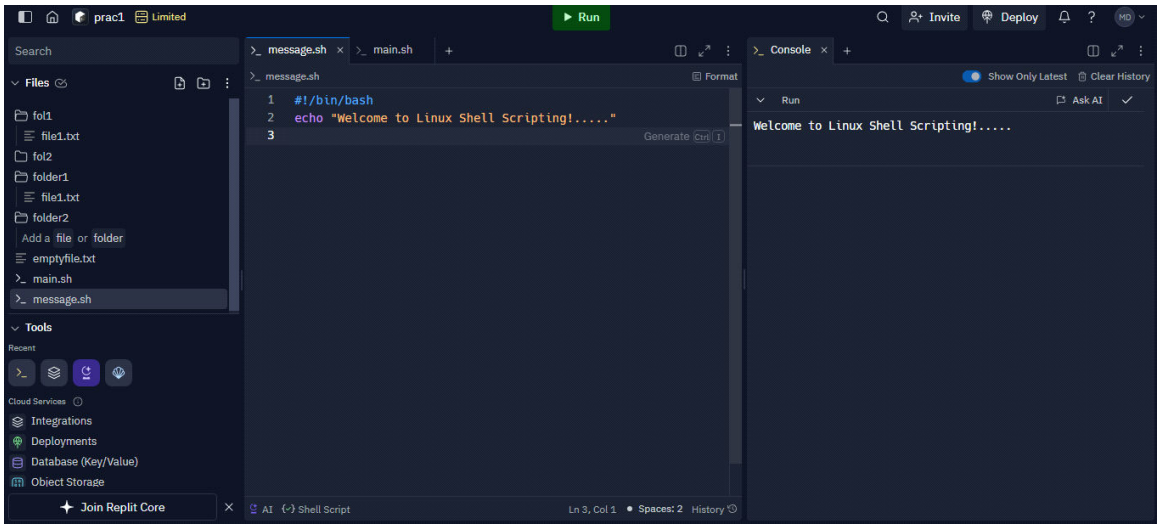
code

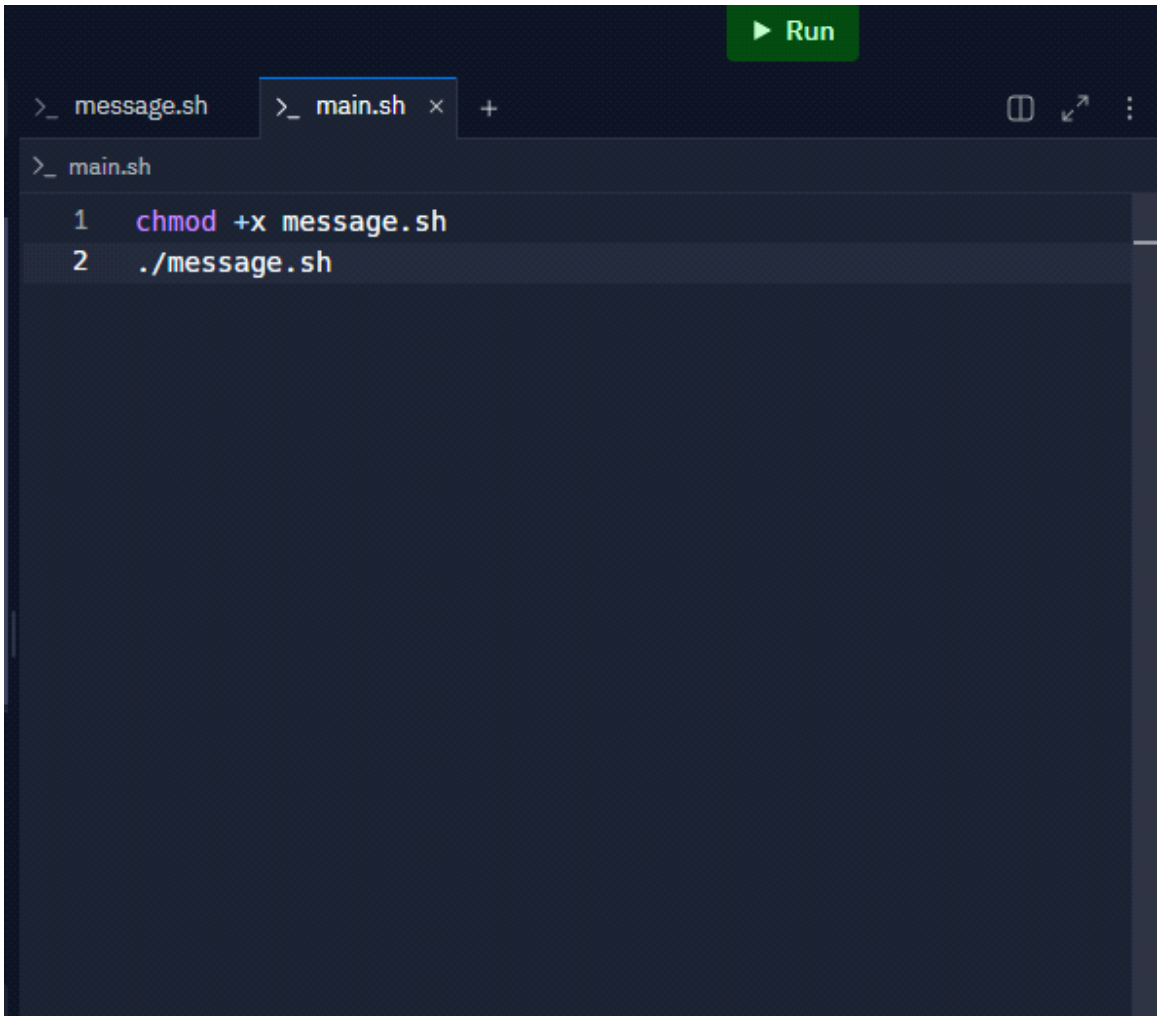
```
#!/bin/bash

echo "Welcome to Linux Shell Scripting!"
```

Save and run the script:

code

```
chmod +x message.sh

./message.sh
```





b) Shell Script to Access Command-Line Arguments

Create the script:

code

nano args.sh

Write the script:

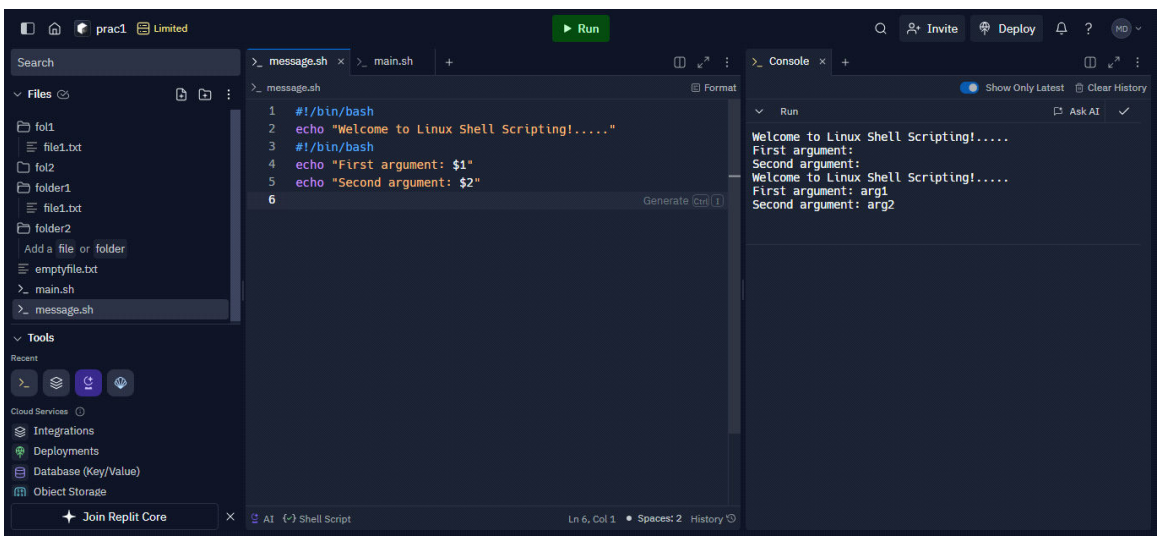
code

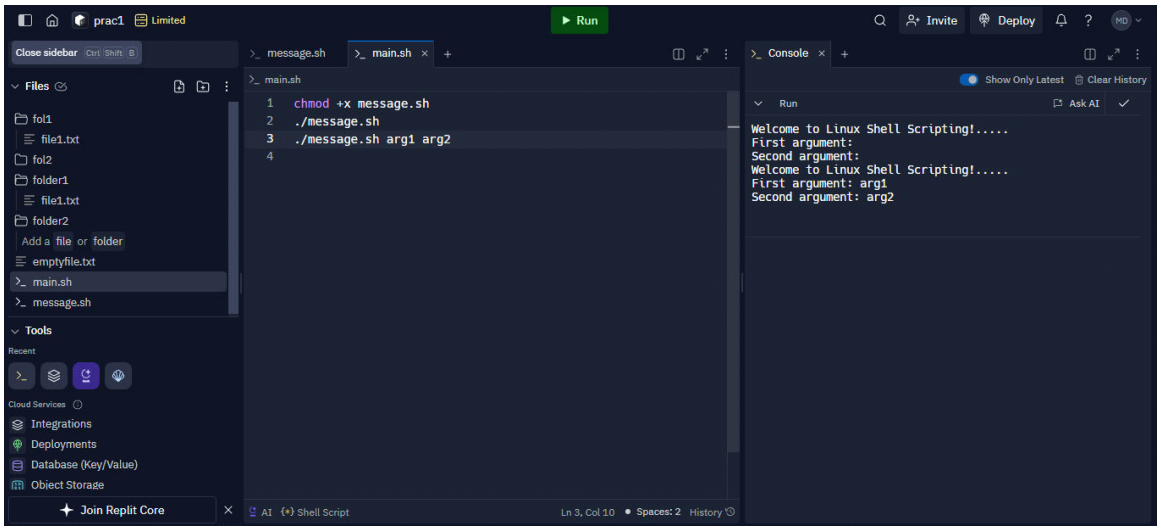
```
#!/bin/bash

echo "First argument: $1"

echo "Second argument: $2"
```

Save and run the script with arguments:

code

```
./args.sh arg1 arg2
```





c) Shell Script to Create Files with Command-Line Names

Create the script:

code

```
nano create_files.sh
```

Write the script:

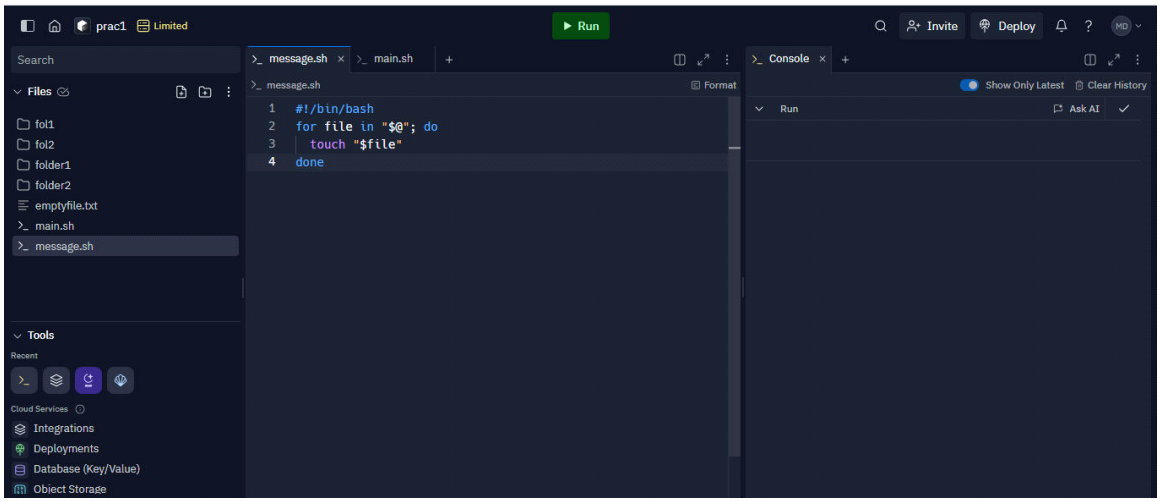
code

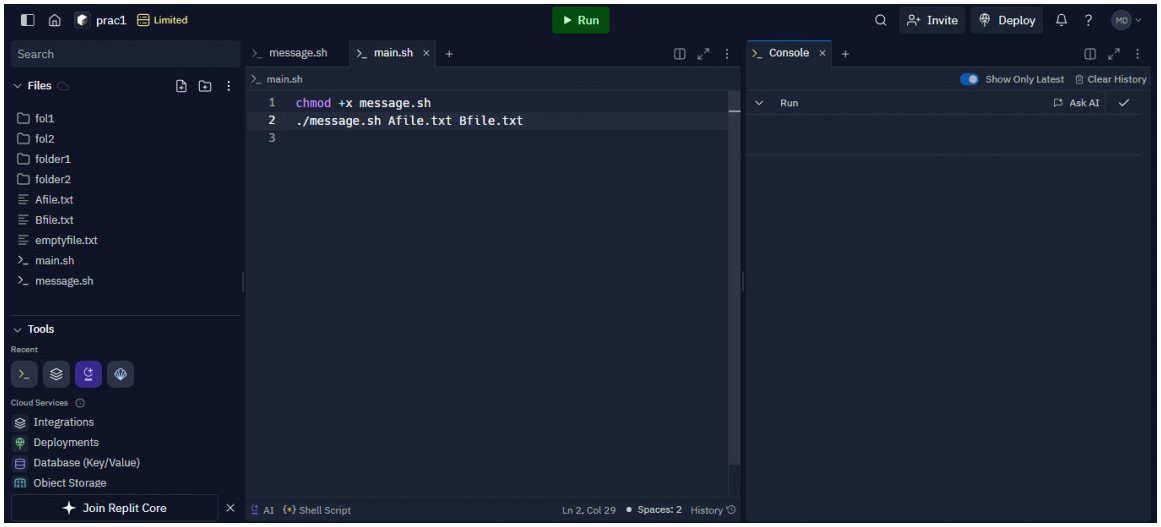
```
#!/bin/bash

for file in "$@"; do

    touch "$file"

done
```

Save and run the script:

code

```
./create_files.sh file1.txt file2.txt
```





for file in "$@"; do ... done:

The for loop iterates over each argument passed to the script.

"$@" is a special variable that represents all the arguments passed to the script as a list, preserving any spaces within each argument.

Each individual argument (file name in this case) is stored in the variable file during each iteration of the loop.

touch "$file":

The touch command is used to create an empty file with the specified name, or to update the timestamp of an existing file.

Here, "$file" is each file name from the list of arguments. So, touch creates each file with the name provided in the command line.

d) Shell Script to Find Factorial of a Number

Create the script:

code

nano factorial.sh

Write the script:

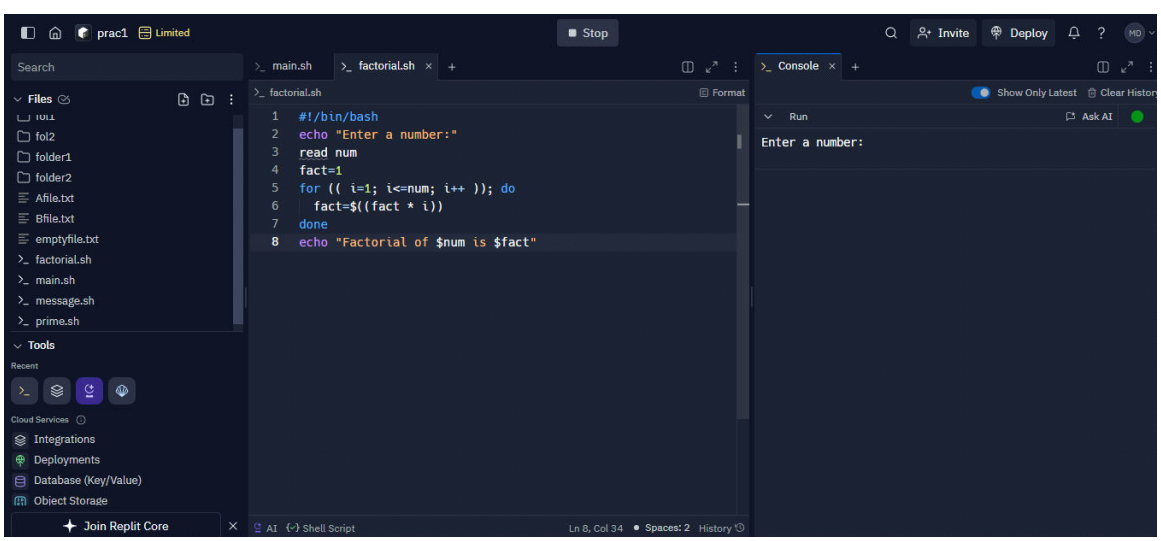
code

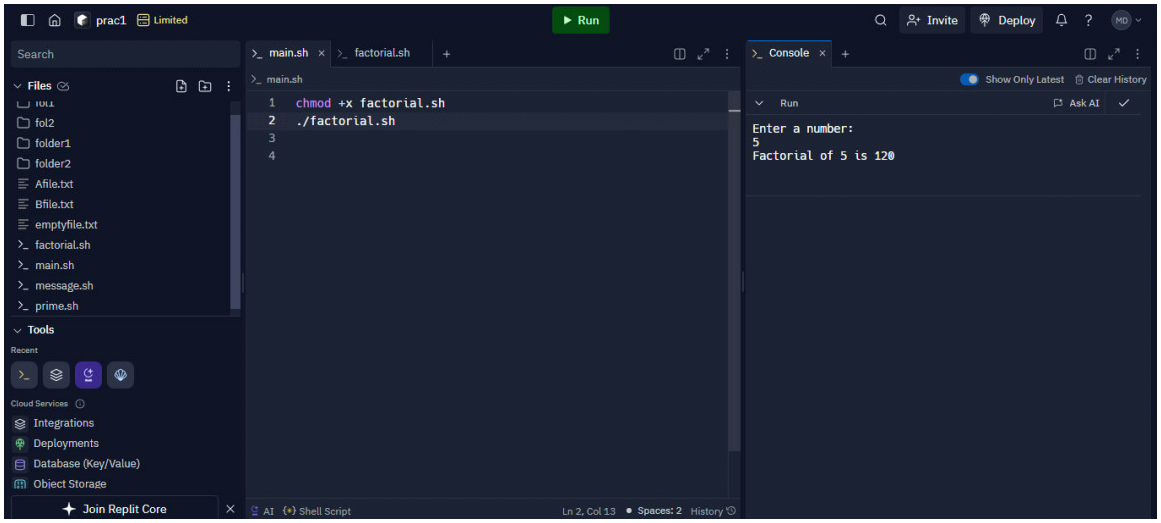
```
#!/bin/bash

echo "Enter a number:"

read num

fact=1

for (( i=1; i<=num; i++ )); do

    fact=$((fact * i))

done

echo "Factorial of $num is $fact"
```

Save and run the script:

code

```
chmod +x factorial.sh

./factorial.sh
```





e) Shell Script to Check if a Number is Prime

Create the script:

code

nano prime.sh

Write the script:

code

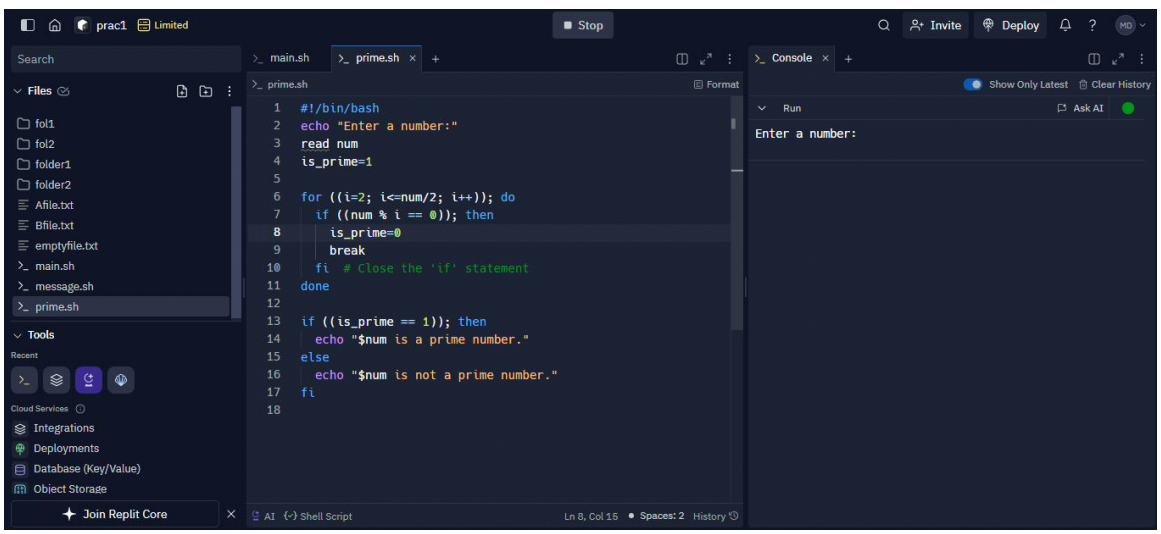
```
#!/bin/bash

echo "Enter a number:"

read num

is_prime=1


for ((i=2; i<=num/2; i++)); do

    if ((num % i == 0)); then

        is_prime=0

        break
```

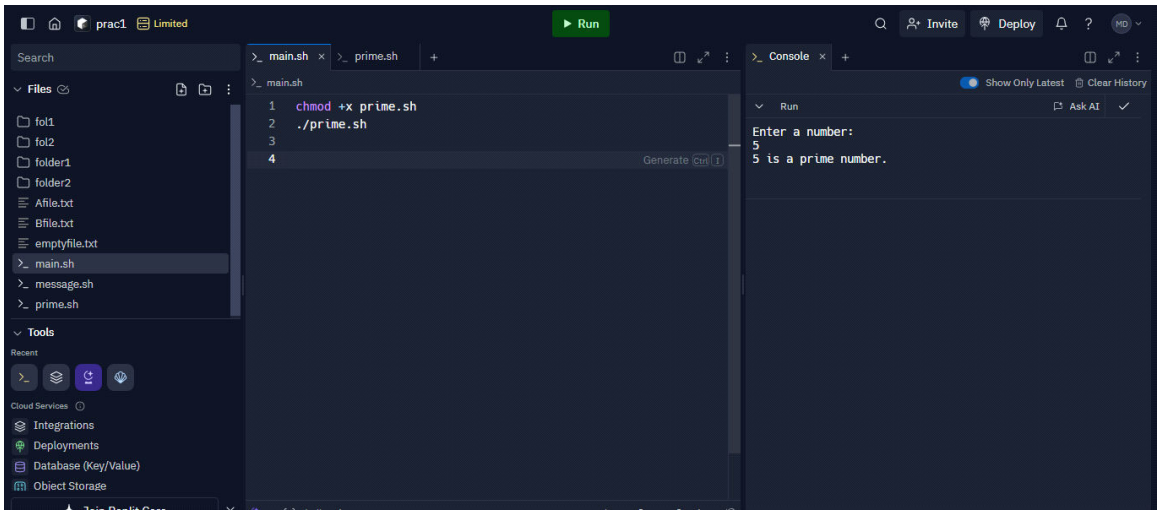
```
    fi    # Close the 'if' statement
done

if ((is_prime == 1)); then
    echo "$num is a prime number."
else
    echo "$num is not a prime number."
fi
```

Save and run the script:

code

```
chmod +x prime.sh
./prime.sh
```




prac1
Limited
Run
Invite
Deploy
Search
Files
fol1
fol2
folder1
folder2
Afile.txt
Bfile.txt
emptyfile.txt
main.sh
message.sh
prime.sh
Tools
Recent
Cloud Services
Integrations
Deployments
Database (Key/Value)
Object Storage
main.sh
prime.sh
chmod +x prime.sh
./prime.sh
Generate
Console
Show Only Latest
Clear History
Run
Ask AI
Enter a number:
5
5 is a prime number.