PRACTICAL NO. 1

Introduction to DevOps

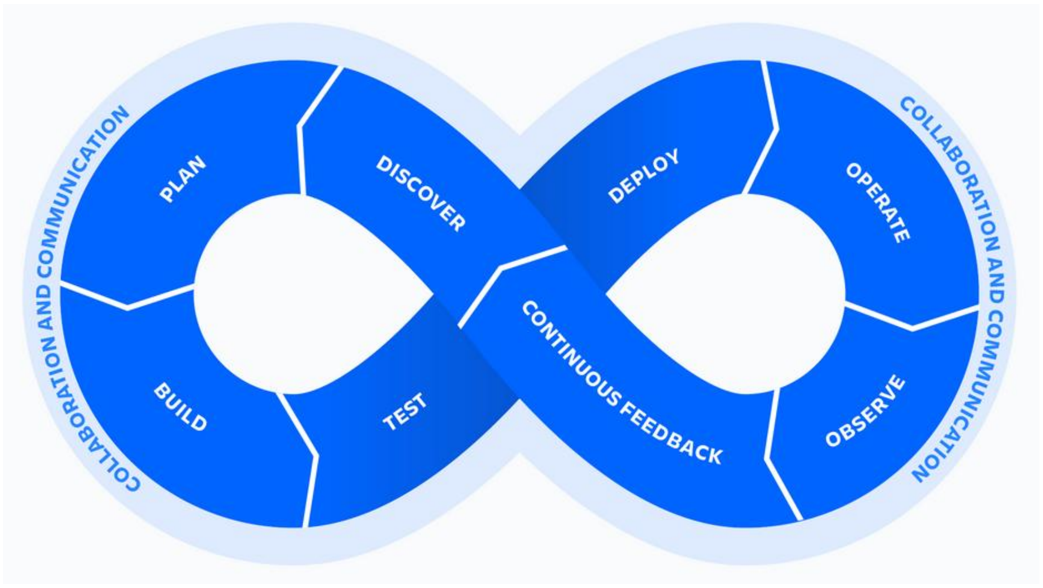
LOB1	To Learn what DevOps is, including its principles and the benefits it offers to organizations.
LO1	Recall and explain the key principles and benefits of DevOps.

1. What is DevOps?

DevOps is the combination of cultural philosophies, practices, and tools to enhance an organization's capability to deliver applications and services rapidly. By fostering faster product evolution and improvement compared to traditional software development and infrastructure management methods, DevOps empowers organizations to better meet customer needs and maintain a competitive edge in the market.

DevOps is a methodology that combines and automates software development (Dev) and information technology operations (Ops) to:

- Improve and shorten the systems development life cycle.
- Collaborate between developers and operations engineers.
- Create a culture where everyone works toward the same goal.


COLLABORATION AND COMMUNICATION
PLAN
DISCOVER
BUILD
TEST
DEPLOY
OPERATE
CONTINUOUS FEEDBACK
OBSERVE
COLLABORATION AND COMMUNICATION

2. Key Principles of DevOps

DevOps is guided by a set of principles that promote collaboration, efficiency, and quality in software development and IT operations. These principles form the foundation for DevOps practices.

a. Collaboration and Communication

i. Break down silos between development, operations, and other teams.
ii. Foster open communication and shared goals.
iii. Use collaboration tools like Slack, Microsoft Teams, or Jira to streamline workflows.

Example: Developers and operations teams jointly plan deployment strategies to avoid downtime during a product launch.

b. Continuous Integration (CI)

i. Developers frequently merge code changes into a shared repository.
ii. Automated tests run to identify integration issues early.

Example: A team uses Jenkins or GitLab CI to automatically test and integrate code after each commit.

c. Continuous Delivery (CD)

i. Ensure that code changes are automatically prepared for deployment.
ii. Focus on frequent, reliable releases to production.

Example: An e-commerce company uses AWS CodePipeline to deploy updates daily without affecting user experience.

d. Automation

i. Automate repetitive tasks like testing, building, and deploying software.
ii. Use tools for infrastructure provisioning, monitoring, and scaling.

Example: A team uses Ansible to automate server configuration and deployment, saving hours of manual work.

e. Infrastructure as Code (IaC)

i. Manage infrastructure using code, ensuring consistency and scalability.
ii. Tools like Terraform and AWS CloudFormation help define infrastructure in code.

Example: A company uses IaC to replicate its production environment for testing, reducing setup errors.

f. Monitoring and Feedback

i. Continuously monitor applications and infrastructure for performance and issues.
ii. Use feedback loops to improve processes and resolve problems quickly.

Example: Netflix uses Prometheus and Grafana to monitor server health and optimize performance.

g. Continuous Testing

i. Test software automatically at every stage of the development lifecycle.
ii. Ensure quality and security before deployment.

Example: A team integrates Selenium for automated UI testing in its CI/CD pipeline.

h. Shared Responsibility

i. Developers and operations teams share accountability for the success of applications.
ii. Teams collaborate to resolve issues and improve the product.

Example: Both developers and operations participate in post-mortem analyses after a production outage.

3. Key Practices of DevOps

1. Version Control: Use tools like Git to manage code changes and enable collaboration.
2. CI/CD Pipelines: Automate building, testing, and deploying code.
3. Microservices Architecture: Break applications into smaller, independent services.
4. Containerization: Use Docker or Kubernetes to ensure consistency across environments.
5. Agile Development: Adopt Agile methodologies to enhance flexibility and adaptability.
6. Incident Management: Implement robust systems for detecting, reporting, and resolving incidents.
7. Change Management: Manage changes effectively to minimize risks and downtime.

By adopting these principles and practices, organizations can achieve faster software delivery, higher quality, and improved collaboration across teams.

4. Benefits of Implementing DevOps

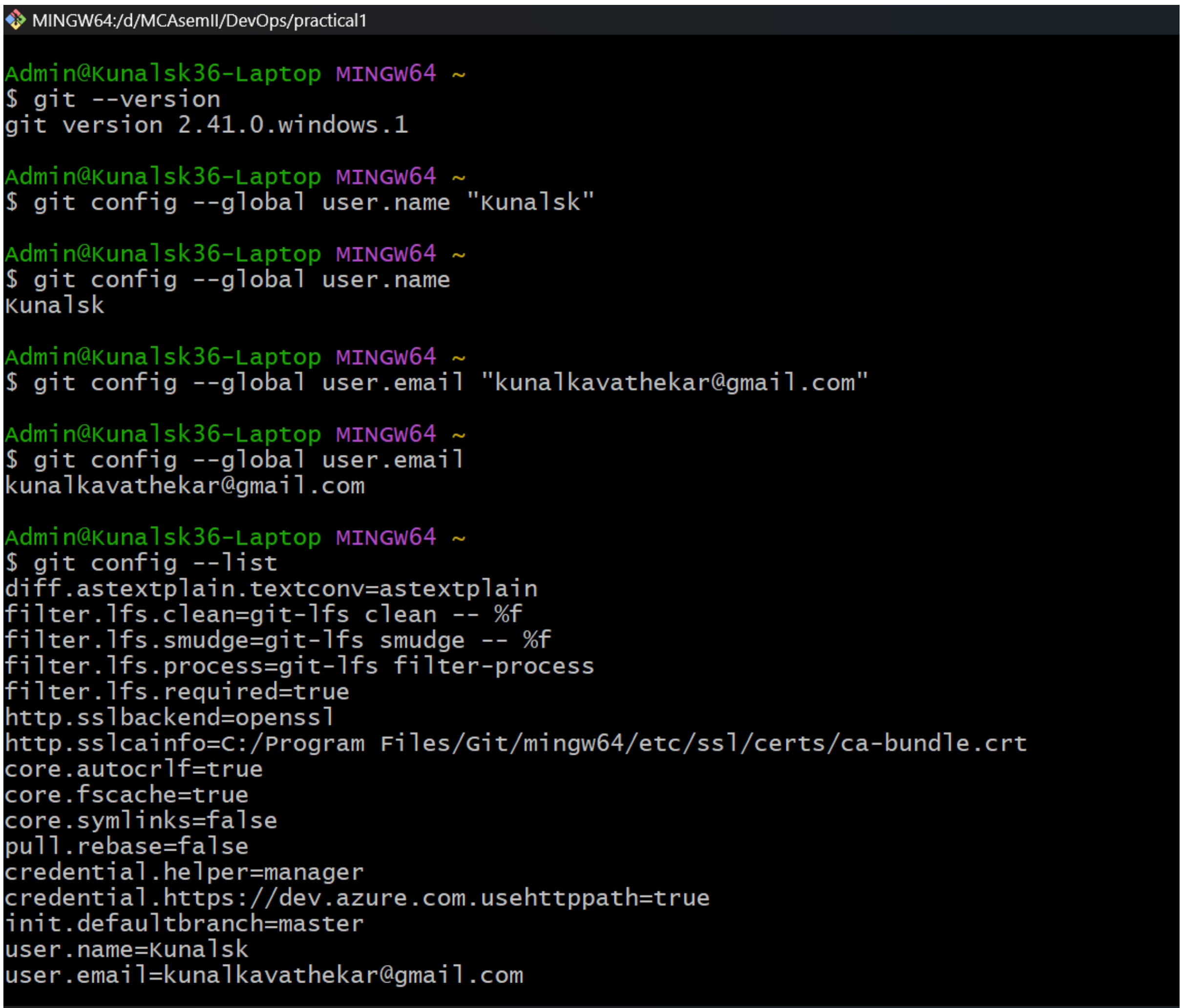
a. Faster Time to Market - Accelerates the development and deployment process, enabling organizations to deliver new features, updates, and bug fixes faster.
b. Improved Collaboration and Efficiency - Breaks down silos between development and operations teams, fostering better communication and shared responsibilities.
c. Reduced Costs - Automation and streamlined workflows minimize manual effort, reducing operational costs.
d. Continuous Improvement - Feedback loops and monitoring allow organizations to learn from successes and failures, improving processes over time.
e. Increased Deployment Frequency - DevOps enables frequent, incremental updates instead of infrequent, large releases.
f. Enhanced Customer Satisfaction - Faster delivery of new features and higher-quality applications result in better user experiences and happier customers.
g. Scalability and Flexibility - Infrastructure as Code (IaC) and containerization allow applications to scale easily with demand.
h. Better Incident Management - Continuous monitoring and quick feedback loops ensure faster identification and resolution of issues.

5. Basic Git Setup

1. Installing Git
    a. Install Git on your local machine and verify the installation using git --version.
2. Configuring Git
    a. Set up your user name and email:
    b. git config --global user.name "Your Name"
    c. git config --global user.email "your.email@example.com"
3. Initializing a Repository
    a. Create a new directory and initialize it as a Git repository using git init.
    b. Explore the .git folder to understand its structure.
4. Open an account on www.github.com for an online repository which is available at anytime and anywhere.

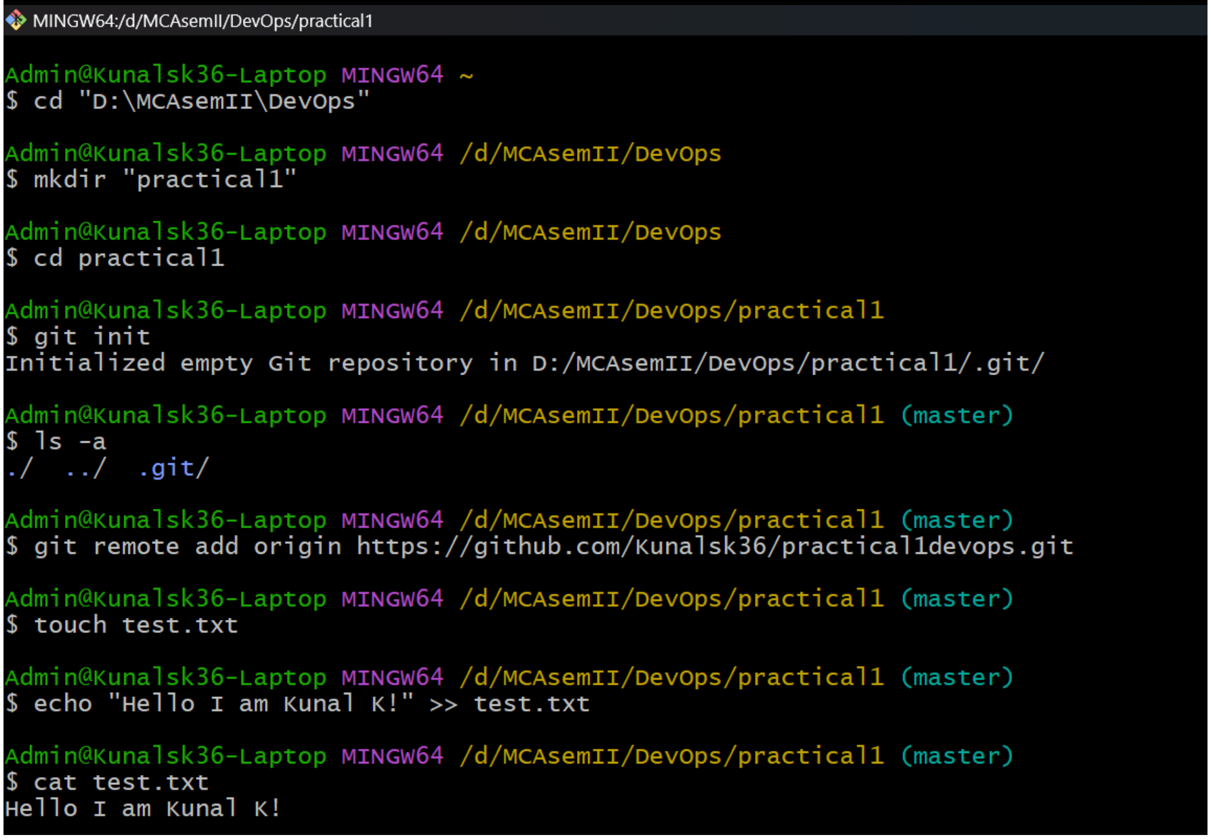
Exercise: -

1. Team Setup: Form teams and assign roles (e.g., developer, tester, operations).
2. Git Installation: Download and Install git on your machine.
3. Configuration: Configure Git using your mail account and then verify it.
4. Initialization: Initialize or create a new Git repository and verify it.
5. Git Hub account: Sign in to www.github.com for an online repository which is available at anytime and anywhere.



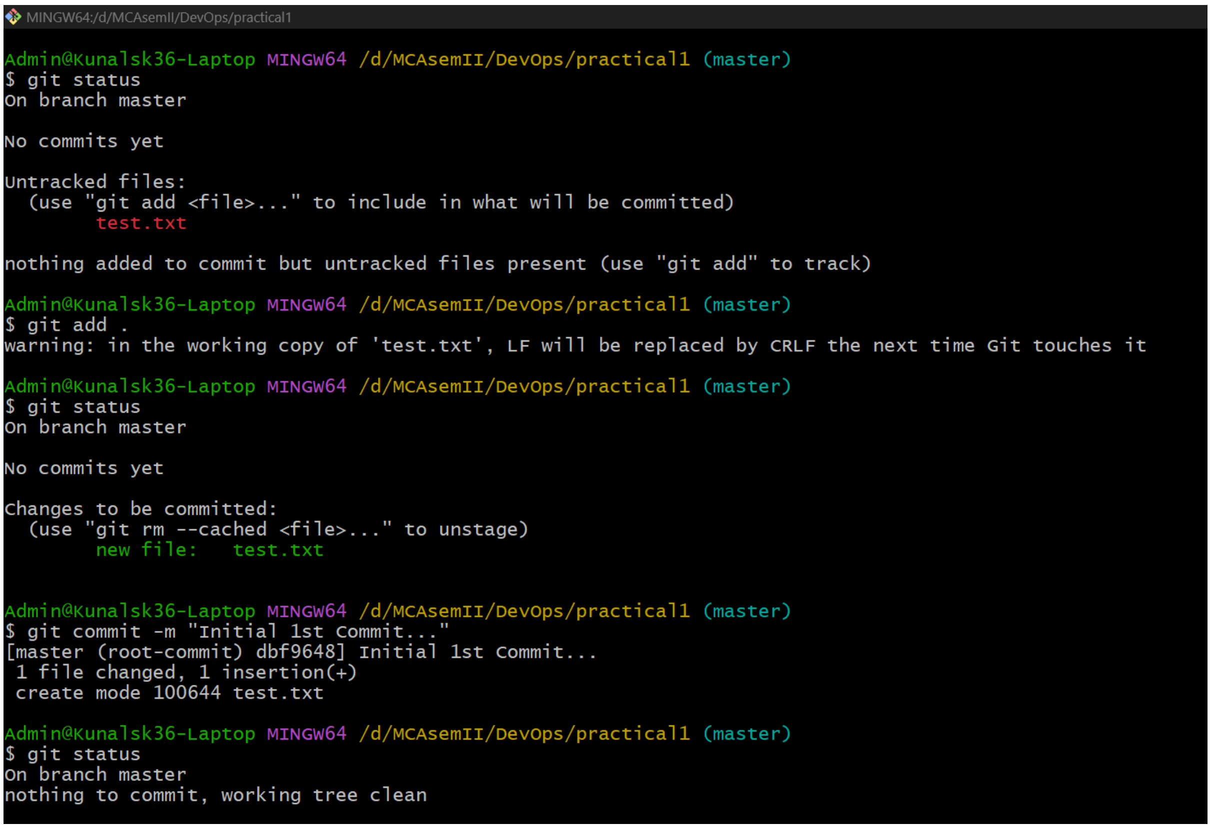
MINGW64:/d/MCAsemII/DevOps/practical1

```
Admin@Kunalsk36-Laptop MINGW64 ~
$ git --version
git version 2.41.0.windows.1

Admin@Kunalsk36-Laptop MINGW64 ~
$ git config --global user.name "Kunalsk"

Admin@Kunalsk36-Laptop MINGW64 ~
$ git config --global user.name
Kunalsk

Admin@Kunalsk36-Laptop MINGW64 ~
$ git config --global user.email "kunalkavathekar@gmail.com"

Admin@Kunalsk36-Laptop MINGW64 ~
$ git config --global user.email
kunalkavathekar@gmail.com

Admin@Kunalsk36-Laptop MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Kunalsk
user.email=kunalkavathekar@gmail.com
```

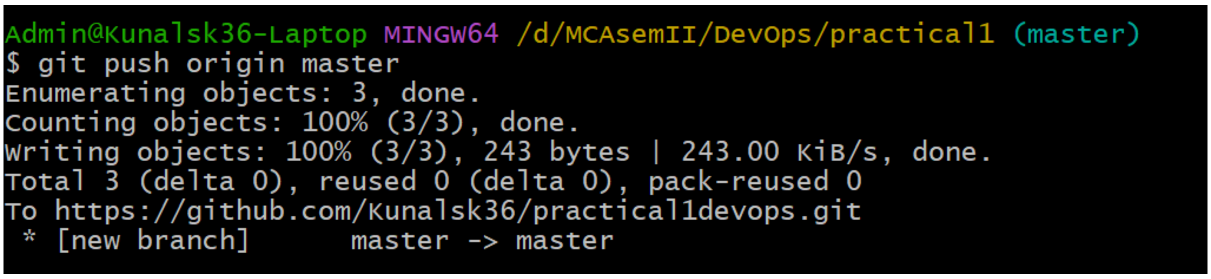


MINGW64:/d/MCAsemII/DevOps/practical1

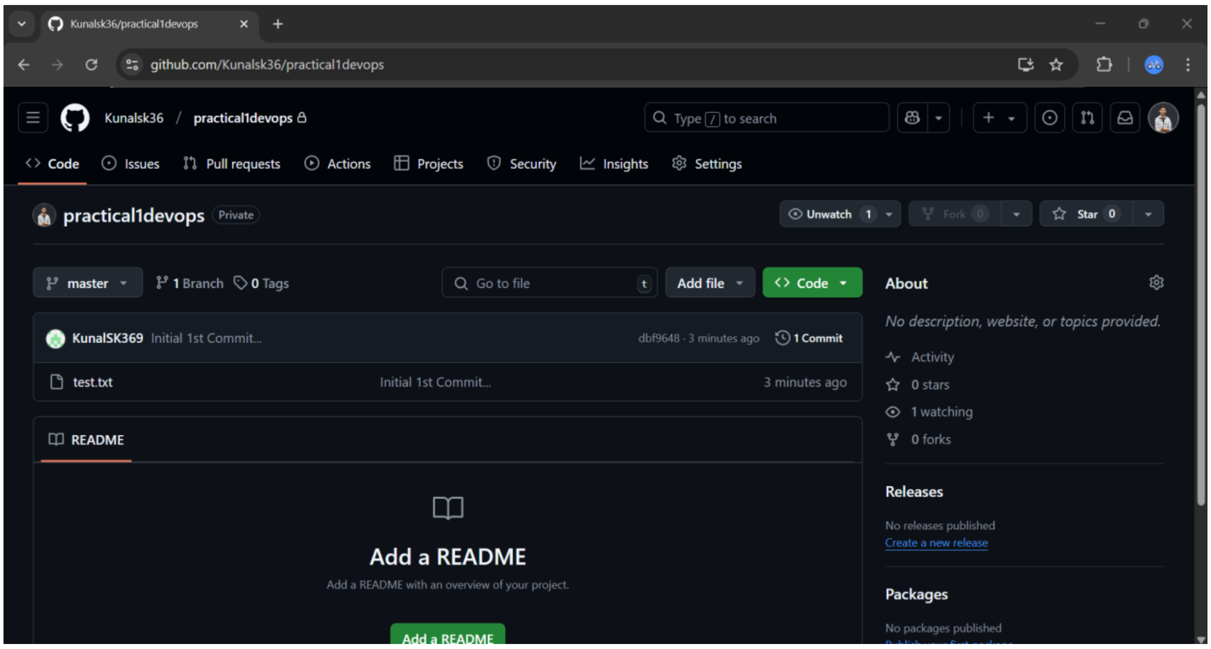
```
Admin@Kunalsk36-Laptop MINGW64 ~
$ cd "D:\MCAsemII\DevOps"

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps
$ mkdir "practical1"

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps
$ cd practical1

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1
$ git init
Initialized empty Git repository in D:/MCAsemII/DevOps/practical1/.git/

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ ls -a
./  ../  .git/

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git remote add origin https://github.com/Kunalsk36/practical1devops.git

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ touch test.txt

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ echo "Hello I am Kunal K!" >> test.txt

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ cat test.txt
Hello I am Kunal K!
```



```
MINGW64:/d/MCAsemII/DevOps/practical1

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.txt

nothing added to commit but untracked files present (use "git add" to track)

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git add .
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test.txt


Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git commit -m "Initial 1st Commit..."
[master (root-commit) dbf9648] Initial 1st Commit...
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git status
On branch master
nothing to commit, working tree clean
```



```
Admin@Kunalsk36-Laptop MINGW64 /d/MCAsemII/DevOps/practical1 (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 243 bytes | 243.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Kunalsk36/practical1devops.git
 * [new branch]      master -> master
```



github.com/Kunalsk36/practical1devops

Kunalsk36 / practical1devops

practical1devops Private

master · 1 Branch · 0 Tags

KunalSK369 Initial 1st Commit... dbf9648 · 3 minutes ago · 1 Commit

test.txt Initial 1st Commit... 3 minutes ago

About

No description, website, or topics provided.

Activity · 0 stars · 1 watching · 0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Add a README

Add a README with an overview of your project.