**Lab 1: Environment Provisioning, Git Bootstrap, Azure & Databricks Navigation (Windows)**

**Objective:**

By the end of this lab, you will be able to:

* Set up Git with signed commit support
* Use CLI tools (PowerShell and Git Bash) for basic operations
* Push signed commits to GitHub
* Provision Azure Resource Group (RG)
* Access and launch a Databricks cluster and run a test notebook

**Pre-requisites:**

* Git for Windows installed
* GPG4Win installed and available via Git Bash or Kleopatra
* Azure CLI installed and authenticated
* Access to GitHub and Azure with Databricks workspace

**Step 1: Git Installation & Signed Commit Configuration**

1.1 Install Git and GPG4Win  
Download and install:

* Git: <https://git-scm.com/download/win>
* GPG: <https://gpg4win.org/>

1.2 Set Git identity:

powershell

git config --global user.name "Your Name"

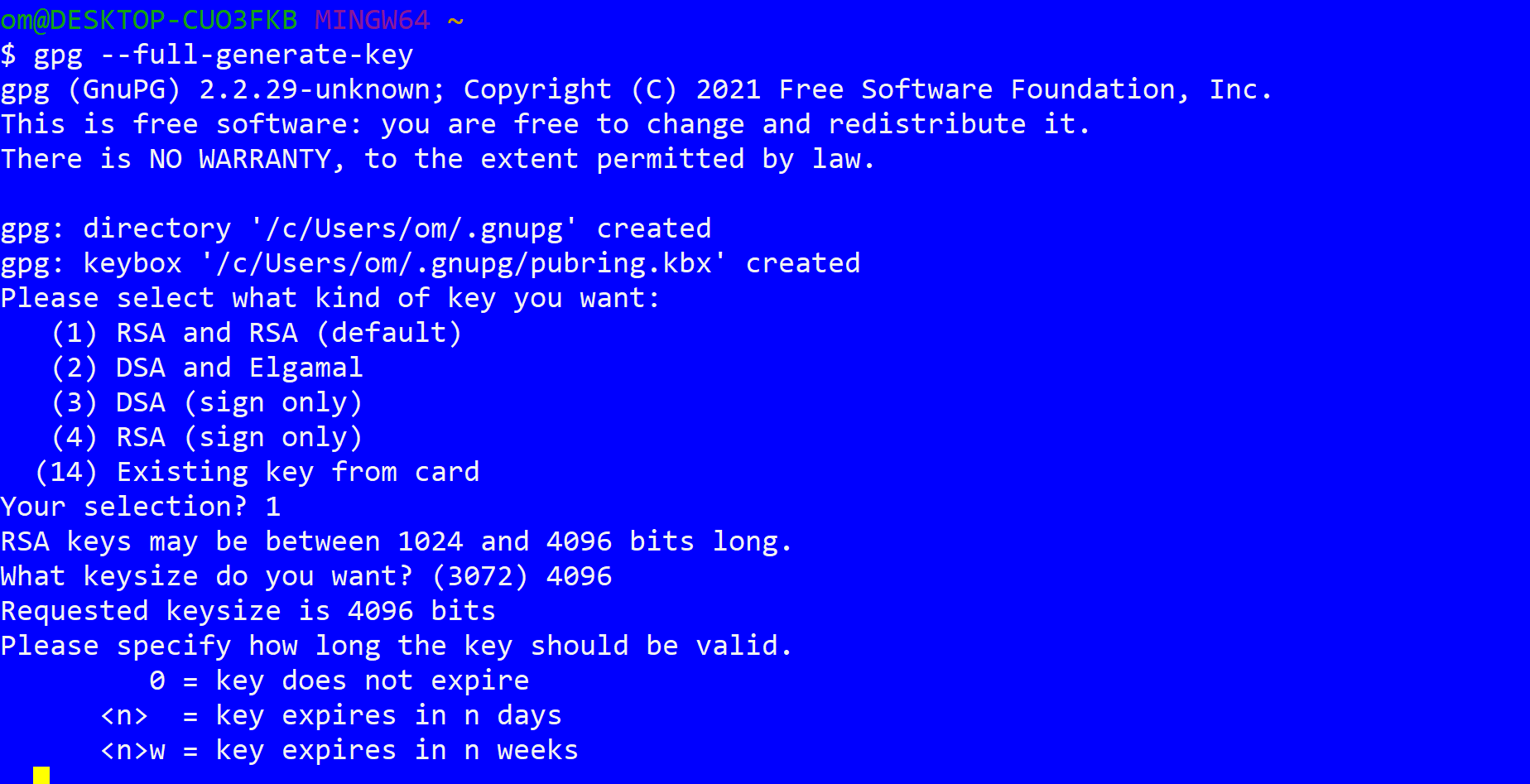
git config --global user.email "your@email.com"

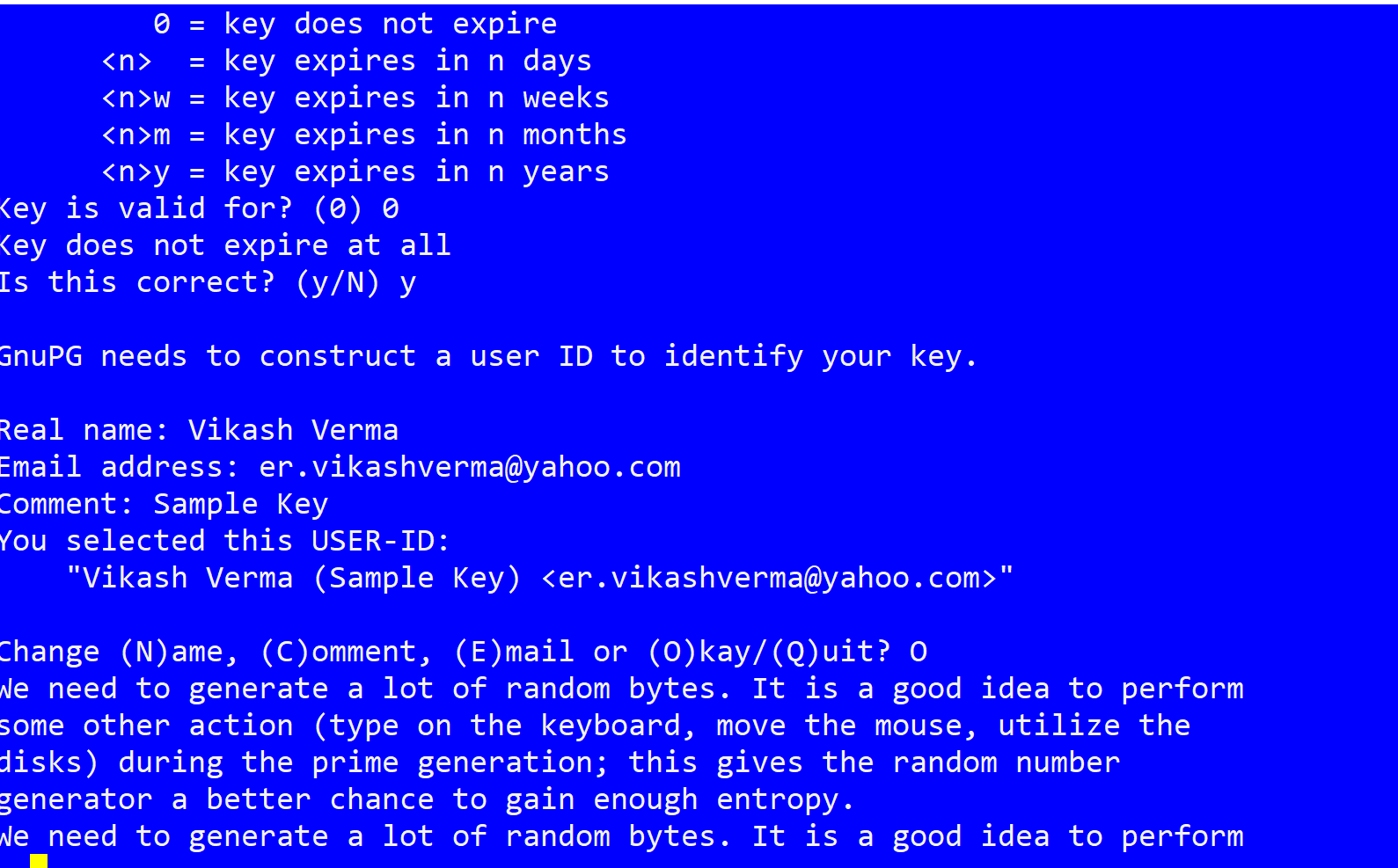
1.3 Generate GPG Key (Git Bash):

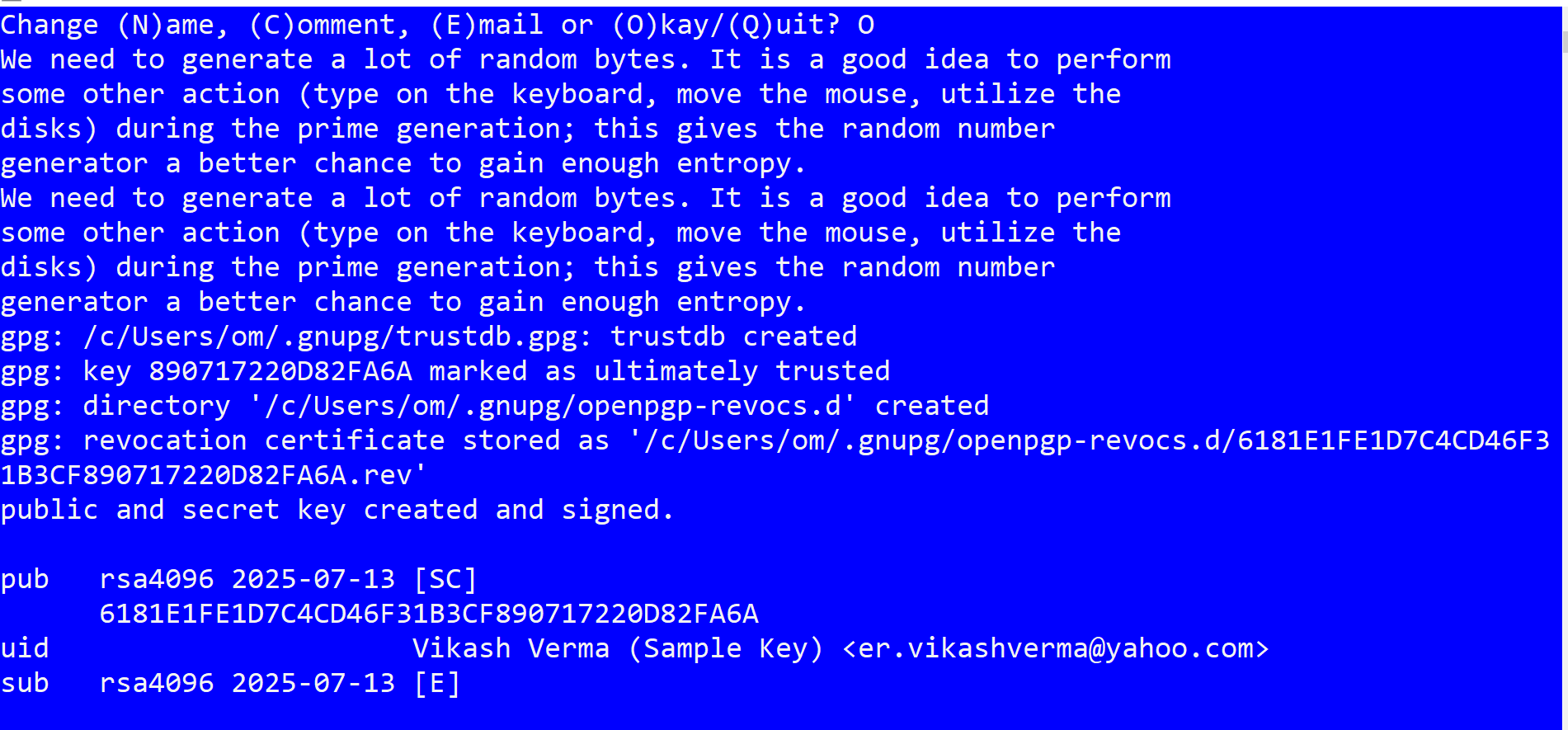
bash

gpg --full-generate-key

* Type: RSA and RSA
* Key size: 4096
* Expiry: 0 (never expires)







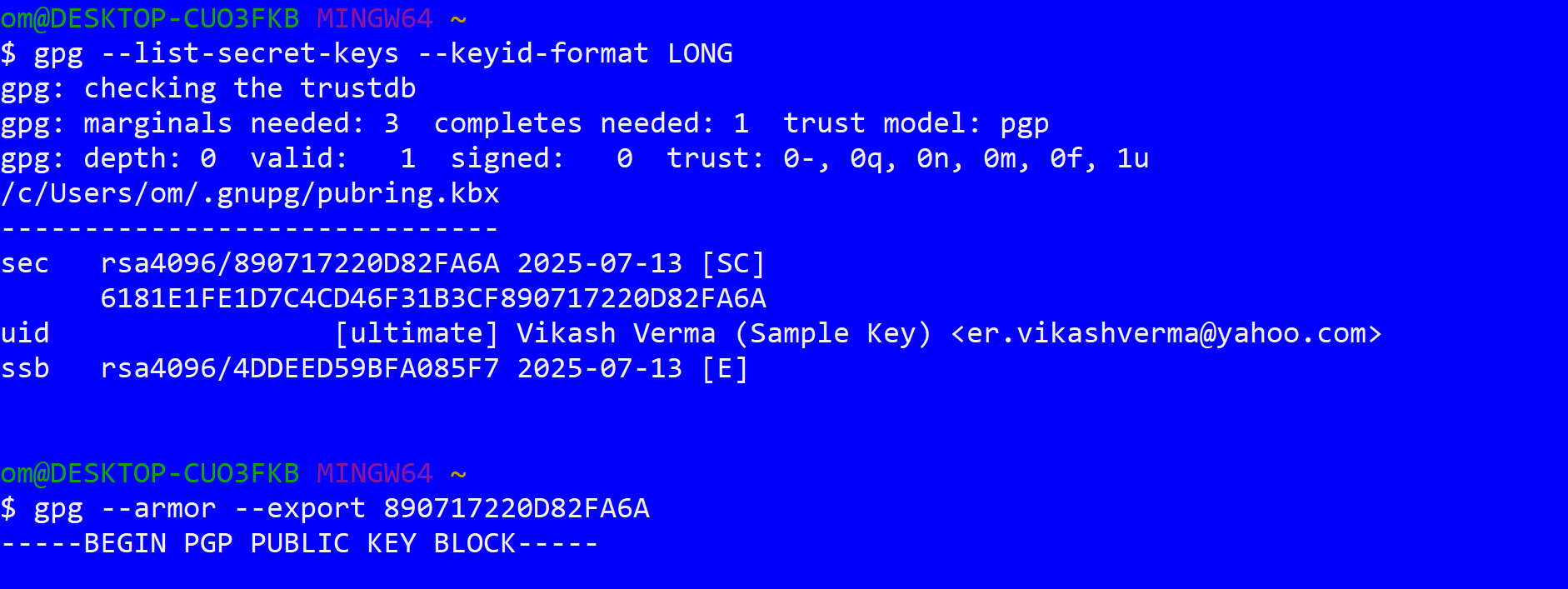
1.4 Export your GPG key:

bash

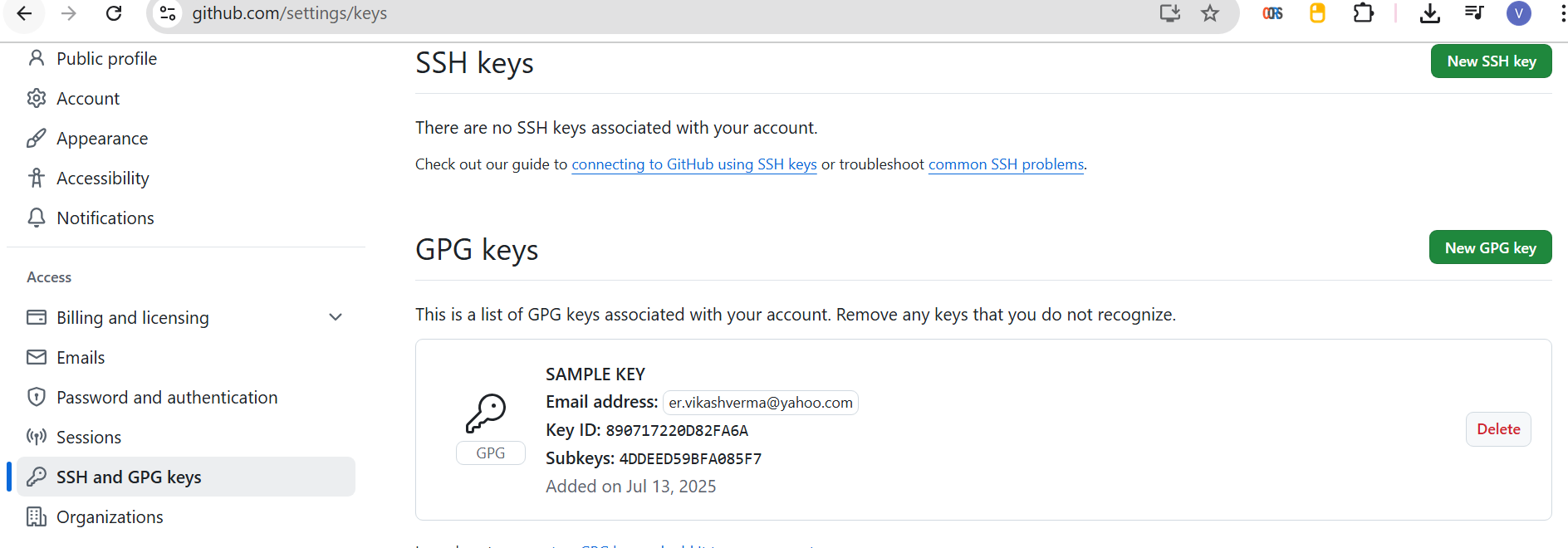
gpg --list-secret-keys --keyid-format LONG

gpg --armor --export <YOUR\_KEY\_ID>

**gpg --armor --export 890717220D82FA6A (SAMPLE)**



Paste the public key into GitHub: Settings → SSH and GPG keys → New GPG Key.



1.5 Configure Git to use the key:

powershell

git config --global user.signingkey <YOUR\_KEY\_ID>

**git config --global user.signingkey 890717220D82FA6A (SAMPLE)**

git config --global commit.gpgSign true

**Step 2: First Signed Commit & Push to GitHub**

2.1 Create and commit sample repo:

Powershell

cd "C:\Users\om\Desktop"

mkdir lab-bootstrap

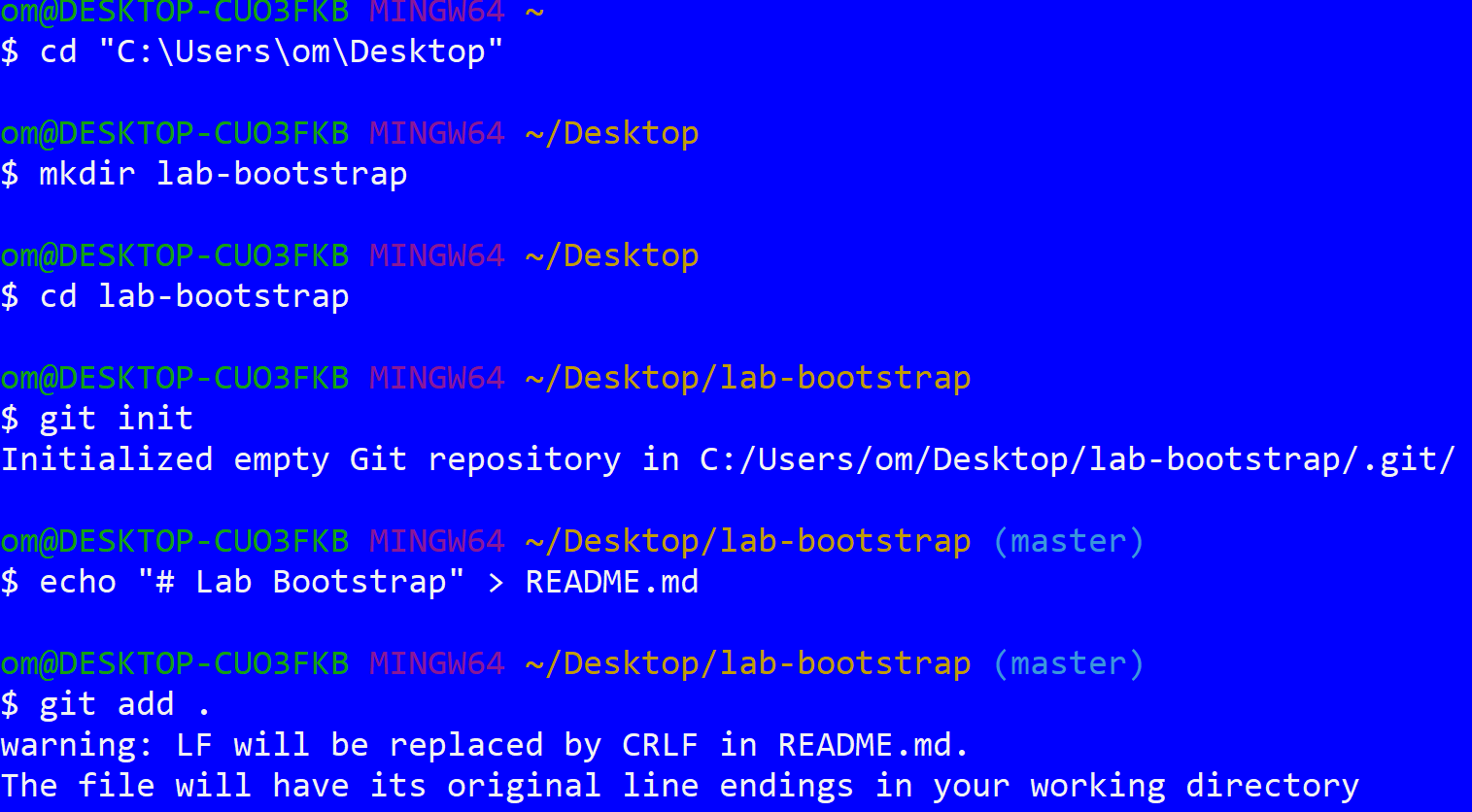
cd lab-bootstrap

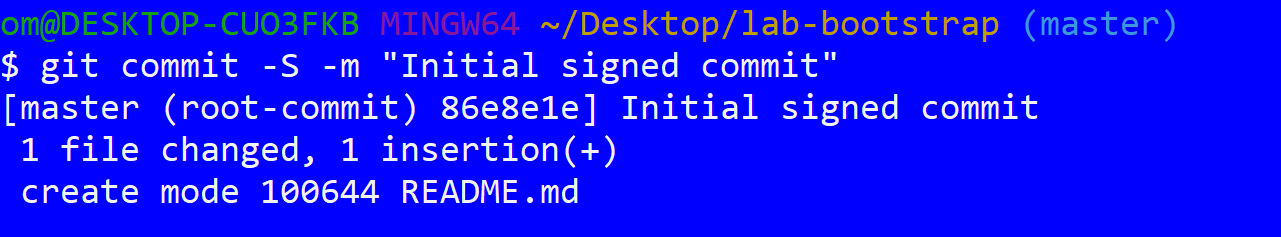
git init

echo "# Lab Bootstrap" > README.md

git add .

git commit -S -m "Initial signed commit"





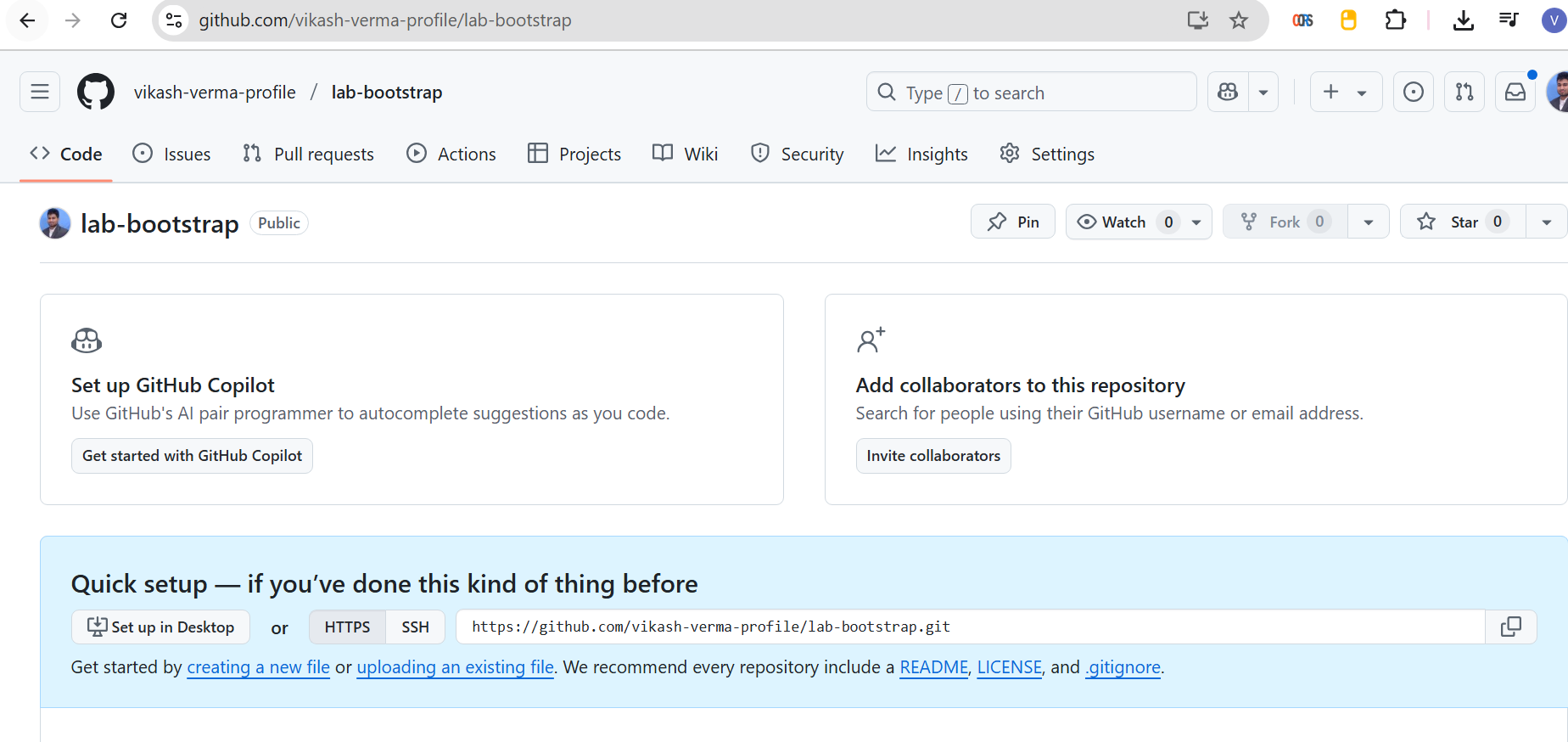
2.2 Push to GitHub:

powershell

git remote add origin https://github.com/your-user/lab-bootstrap.git

git branch -M main

git push -u origin main



**Step 3: CLI Task Using PowerShell**

powershell

cd ~

mkdir cli-lab

cd cli-lab

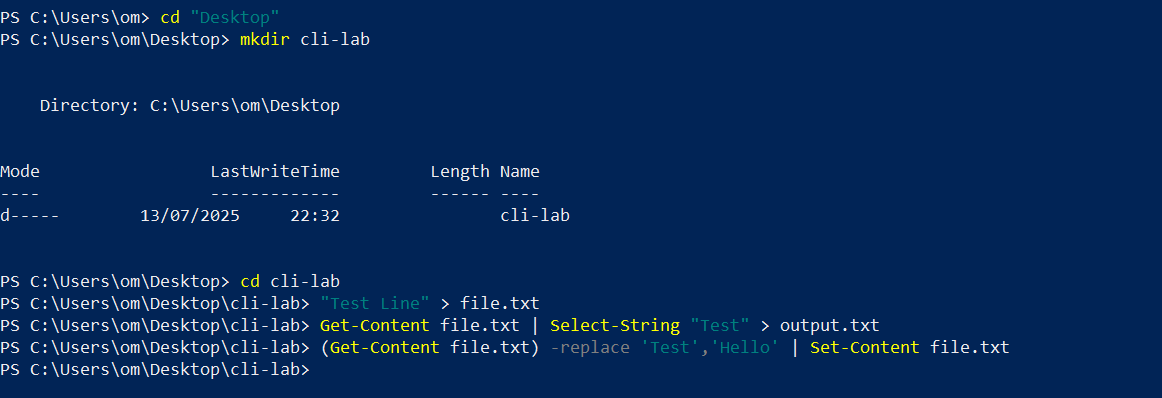
"Test Line" > file.txt

Get-Content file.txt | Select-String "Test" > output.txt

(Get-Content file.txt) -replace 'Test','Hello' | Set-Content file.txt

Key tools used:

* Select-String: grep-like pattern match
* Set-Content: file overwrite
* |: piping between commands



This quick guide demonstrates core Linux shell commands: bash, pipes (|), grep, and sed — all runnable inside **Git Bash** on Windows.

**1. Basic Bash Commands**

bash

mkdir cli-lab

cd cli-lab

echo "hello world" > file.txt

cat file.txt

* mkdir: Create a new folder
* cd: Change directory
* echo: Print text
* >: Redirect output to a file
* cat: View contents of a file

**2. Piping (|) Output Between Commands**

bash

cat file.txt | grep hello

* | takes the output of cat file.txt and passes it into grep hello

Another example:

bash

echo "apple banana mango" | tr ' ' '\n'

* tr: Replaces characters (space to newline here)

**3. grep – Pattern Matching**

bash

echo -e "error\ninfo\nwarning\nerror" > logs.txt

grep error logs.txt

More examples:

bash

grep -i error logs.txt # Case-insensitive match

grep -n warning logs.txt # Show line numbers

grep -v info logs.txt # Show lines that do NOT contain "info"

**4. sed – Stream Editing (Search and Replace)**

bash

echo "hello world" > msg.txt

sed 's/world/universe/' msg.txt

* Substitutes the word "world" with "universe"

To edit the file in-place:

bash

sed -i 's/hello/hi/' msg.txt

cat msg.txt

**5. Combine cat, grep, sed with Pipes**

bash

cat logs.txt | grep error | sed 's/error/ERROR/g'

* Find lines containing "error"
* Replace "error" with uppercase "ERROR"

**Step 4: Provision Azure Resource Group**

4.1 Login:

powershell

az login

4.2 Create resource group:

powershell

az group create --name dev-rg --location eastus

**Step 5: Databricks Setup**

5.1 Access workspace:

* Azure Portal → Databricks Resource → Launch Workspace

5.2 Create cluster:

* Name: dev-cluster
* Runtime: 12.x LTS
* Instance: Standard\_DS3\_v2

5.3 Run test notebook:  
Create a new Python notebook and run:

python

df = spark.range(1, 5)

df.show()

**Lab Completion Checklist:**

* Git and GPG key setup complete
* First signed commit pushed to GitHub
* PowerShell CLI tasks executed
* Azure RG created via CLI
* Databricks cluster created and verified with notebook

**Lab 2: Git Branching, Merge Conflicts & Pre-Commit Hook Setup (Windows)**

**Objective:**

* Learn branching workflows and resolve merge conflicts
* Automate code checks using Git pre-commit hooks

1. Create a new branch and commit:

powershell

git checkout -b feature/readme-update

Add-Content README.md "`nUpdated info"

git add README.md

git commit -S -m "Updated README in feature branch"

1. Simulate a merge conflict on main:

powershell

git checkout main

Add-Content README.md "`nConflicting change"

git add README.md

git commit -S -m "Main branch conflicting change"

1. Merge and resolve:

powershell

git merge feature/readme-update

# Resolve conflicts manually in README.md

git add .

git commit -S -m "Resolved merge conflict"

1. Setup pre-commit hook:

powershell

pip install pre-commit

Create .pre-commit-config.yaml in the repo:

yaml

repos:

- repo: https://github.com/pre-commit/pre-commit-hooks

rev: v4.5.0

hooks:

- id: trailing-whitespace

- id: end-of-file-fixer

Initialize hook:

powershell

pre-commit install

All commits now auto-run checks.

**Lab 3: Azure CLI – Storage Account & Blob Upload**

**Objective:**

* Use Azure CLI to manage storage
* Upload files to Azure Blob storage

1. Create storage account:

powershell

az storage account create --name mystoragexyz --resource-group dev-rg --location eastus --sku Standard\_LRS

1. Create blob container:

powershell

az storage container create --account-name mystoragexyz --name rawdata --auth-mode login

1. Upload sample file:

powershell

"Sample log data" > sample.txt

az storage blob upload --account-name mystoragexyz --container-name rawdata --name sample.txt --file sample.txt --auth-mode login

1. Verify upload:

powershell

az storage blob list --account-name mystoragexyz --container-name rawdata --auth-mode login --output table

**Lab 4: Databricks Workspace – Configure PAT & REST API Test**

**Objective:**

* Generate a personal access token (PAT)
* Use Databricks REST API to list clusters and create jobs

1. Generate Token:

* Databricks Workspace → User Icon → User Settings → Access Tokens → Generate New Token

1. List clusters:

powershell

curl -X GET https://<your-databricks-instance>/api/2.0/clusters/list `

-H "Authorization: Bearer <YOUR\_TOKEN>"

1. Optional – Create a job:

powershell

curl -X POST https://<your-databricks-instance>/api/2.0/jobs/create `

-H "Authorization: Bearer <YOUR\_TOKEN>" `

-H "Content-Type: application/json" `

-d '{

"name": "job-from-api",

"new\_cluster": {

"spark\_version": "13.3.x-scala2.12",

"node\_type\_id": "Standard\_DS3\_v2",

"num\_workers": 2

},

"notebook\_task": {

"notebook\_path": "/Users/you@example.com/test-notebook"

}

}'