Managing Multiple GitHub Accounts on One Machine (Personal + Work)

This guide helps you seamlessly manage multiple GitHub accounts (e.g., personal and work) on a single machine using SSH keys, aliases, and PowerShell automation. It also explains how to handle third-party repos and push new local projects to GitHub with the correct account.

✅STEP 1: Generate SSH Keys

```
# Personal Account
ssh-keygen -t ed25519 -C "your_email@personal.com" -f ~/.ssh/
id_ed25519_personal

# Work Account
ssh-keygen -t ed25519 -C "your_email@work.com" -f ~/.ssh/id_ed25519_work
```

✅STEP 2: Add Public Keys to GitHub

Copy each public key:

```
cat ~/.ssh/id_ed25519_personal.pub
cat ~/.ssh/id_ed25519_work.pub
```

Then: - Go to GitHub > Settings > SSH and GPG keys > New SSH Key - Add each to the corresponding GitHub account

---

✅STEP 3: SSH Config File with Aliases

File: `~/.ssh/config` (not `.txt`)

```
Host github.com-personal
  HostName github.com
  User git
  IdentityFile /c/Users/YOUR_USERNAME/.ssh/id_ed25519_personal
  IdentitiesOnly yes

Host github.com-work
  HostName github.com
  User git
  IdentityFile /c/Users/YOUR_USERNAME/.ssh/id_ed25519_work
  IdentitiesOnly yes
```

Run:

```
chmod 600 ~/.ssh/config ~/.ssh/id_ed25519_*
```

Test:

```
ssh -T git@github.com-personal
ssh -T git@github.com-work
```

✅STEP 4: Smart Clone Script

Create `smart-clone.ps1` **in** `C:\Users\YourName\Scripts\`

```
param(
  [Parameter(Mandatory = $true)]
  [string]$repoUrl
)

$accounts = @{
  "personal" = @{
    "username" = "your_personal_username"
    "email"    = "your_email@personal.com"
    "name"     = "Your Name"
    "alias"    = "github.com-personal"
  }
  "work" = @{
    "username" = "your_work_username"
    "email"    = "your_email@work.com"
    "name"     = "Your Name"
    "alias"    = "github.com-work"
  }
}

if ($repoUrl -notmatch "^git@github\\.com[:|\\/](.+?)\\/(.+?)(\\.git)?$") {
  Write-Host "Invalid GitHub SSH URL format." -ForegroundColor Red
  exit 1
}

$username = $matches[1]
$repoName = $matches[2].Replace(".git", "")

$account = $null
```

```
foreach ($key in $accounts.Keys) {
  if ($accounts[$key].username -eq $username) {
    $account = $key
    break
  }
}

if (-not $account) {
  Write-Host "Unknown username '$username'. Choose account manually:"
  Write-Host "1. personal"
  Write-Host "2. work"
  $choice = Read-Host "Enter 1 or 2"
  if ($choice -eq "1") { $account = "personal" }
  elseif ($choice -eq "2") { $account = "work" }
  else { Write-Host "Invalid choice."; exit 1 }
}

$alias = $accounts[$account].alias
$newUrl = "git@${alias}:${username}/${repoName}.git"

Write-Host "Cloning from: $newUrl"
git clone $newUrl

if (Test-Path "./$repoName") {
  Set-Location "./$repoName"
} else {
  Write-Host "Error: Repo folder not found."; exit 1
}

git config user.name  $accounts[$account].name
git config user.email $accounts[$account].email

Write-Host "Git identity set to:"
Write-Host "Name : $($accounts[$account].name)"
Write-Host "Email: $($accounts[$account].email)"
```

Create `smart-clone.bat` in the same folder:

```
@echo off
powershell -ExecutionPolicy Bypass -File "C:\\Users\\YourName\\Scripts\
\smart-clone.ps1" %*
```

Add the folder to PATH and restart your shell.

✅STEP 5: Run Smart Clone

```
smart-clone git@github.com:your_personal_username/repo.git
```

It will automatically: - Detect the account - Rewrite the URL with correct alias - Set Git identity

---

✅STEP 6: Use in Git Bash

Create a shell script `~/bin/smart-clone`:

```
#!/bin/bash
powershell.exe -ExecutionPolicy Bypass -File "C:\\Users\\YourName\\Scripts\
\smart-clone.ps1" "$@"
```

Make it executable:

```
chmod +x ~/bin/smart-clone
echo 'export PATH="$HOME/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

✅Cloning Other Repos (Third-party or Open Source)

When cloning a repo from a user/org that is not your personal/work: - `smart-clone` will prompt you to select: - `1. personal` - `2. work`

Select which SSH key and identity you want to use for the clone and commit history.

✅Initializing New Repos (You Create Locally)

When you create a new local repo:

```
git init my-new-project
cd my-new-project
```

Run this command to set the desired account:

For personal:

```
git config user.name "Your Name"
git config user.email "your_email@personal.com"
git remote add origin git@github.com-personal:your_username/my-new-
project.git
```

For work:

```
git config user.name "Your Name"
git config user.email "your_email@work.com"
git remote add origin git@github.com-work:your_work_username/my-new-
project.git
```

Then push:

```
git add .
git commit -m "initial commit"
git push -u origin main
```

✅Summary

Task	Action
Clone own repos	Use `smart-clone` with auto-detection
Clone other people's repos	Use `smart-clone`, choose work/personal when prompted
New repo to personal	Set user.name, user.email and remote to `github.com-personal` alias
New repo to work	Set user.name, user.email and remote to `github.com-work` alias
Git Bash support	Wrapper script calls PowerShell with Windows path

---

This setup makes switching GitHub identities seamless across all workflows, including pushing, cloning, and committing with correct credentials.