GIT BASH

En kort introduktion till **GIT BASH**

Vad är GIT BASH?

Git Bash är ett program som låter dig använda kommandon för att hantera Git och köra bash-kommandon på Windows. Det är som ett "verktygslåda" för utvecklare som gör det enklare att jobba med **versionhantering** och **terminalarbete**.

Tänk dig så här:

GIT kan fungera som en organiserad anteckningsbok där alla ändringar sparas i projektet. GIT BASH är då pennan du använder för att kunna skriva och bläddra i den boken för att kunna se vilka ändringar som är gjorda.

Varför GIT BASH?

Med GIT BASH får du en användarvänlig miljö att arbeta med tillsammans med kollegor då du enkelt kan ta ner den senaste versionen som är arbetad med till din dator och du kan pusha upp den din senaste version av projektet till kollegor.

Det finns möjligthet att använda på både Windows, macOS och Linux.

Du kan ställa in uppdateringar efter behov exempelvis om du vill uppdatera din hemsida så kan du göra det klockan 03.00 på natten och programmeraren behöver inte vara vaken och uppgradera i realtid.

Användningsområden

Köra avancerade Git-kommandon

Git Bash används när du behöver använda Git-funktioner som kanske inte stöds i grafiska verktyg. Exempel är:

`git stash`, för att tillfälligt spara ändringar.

`git cherry-pick`, för att ta en specifik commit från en gren.

`git rebase`, för att omorganisera commit-historiken.

Skript och automatisering

Körning av deploy-skript.

Automatiserade tester.

Uppdatering av repositories på fjärrservrar.

Skript och automatisering

Arbete i en Unix-liknande miljö på Window

Kommandon

- **git init**
  Skapar ett nytt Git-repository i den aktuella katalogen.
- **git clone [URL]**
  Klonar ett existerande repository från en fjärrserver (t.ex. GitHub) till din dator.
- **git status**
  Visar statusen för ditt repository, t.ex. ändrade, nya eller staged filer.
- **git add [fil]**
  Lägg till filer till staging-området inför en commit.
- **git commit -m "meddelande"**
  Sparar ändringarna i en commit med ett beskrivande meddelande.
- **git push**
  Skickar lokala commits till ett fjärrrepository.
- **git pull**
  Hämtar och slår samman ändringar från ett fjärrrepository till din lokala kopia.
- **git branch**
  Visar alla grenar i projektet.
  Använd `git branch [namn]` för att skapa en ny gren.
- **git checkout [branch]**
  Byt till en annan gren.
- **git merge [branch]**
  Slå samman en annan gren till den aktuella.

Fördelar

Linux-liknande miljö på Windows: Git Bash erbjuder ett terminalgränssnitt som liknar Linux, vilket gör det lättare för utvecklare att använda bash-kommandon även på Windows.

Fullständig Git-integration: Den levereras med Git förinstallerat och är optimerad för Git-kommandon, vilket gör det enkelt att arbeta med versioneringssystemet.

Stöd för bash-kommandon: Utvecklare kan använda standard bash-kommandon som ls, cd, cat osv., vilket är användbart om man är van vid Linux eller macOS.

Flexibel och kraftfull: Git Bash tillåter körning av skript och hantering av komplexa arbetsflöden med terminalkommandon, vilket kan vara mycket effektivt.

Lättviktig: Det är en enkel, snabb och resurssnål lösning jämfört med tyngre terminaler eller integrerade utvecklingsmiljöer (IDEs).

Kompatibilitet med Git: Många avancerade funktioner och arbetsflöden för Git hanteras smidigare genom Git Bash än med GUI-baserade verktyg.

Nackdelar

Begränsad till Windows: Även om Git Bash är användbart på Windows, är det inte nödvändigt på Linux eller macOS, eftersom dessa redan har bash-terminaler.

Saknar vissa funktioner från riktiga Linux-terminaler: Git Bash är en begränsad emulering och saknar några funktioner som en fullständig Linux-terminal erbjuder.

Inte nybörjarvänligt: Om du är ny till terminaler kan Git Bash kännas svår att använda eftersom det bygger på textbaserade kommandon istället för grafiska gränssnitt.

Prestanda: Vissa komplexa kommandon eller skript kan vara långsammare än i en riktig Linux-miljö.

Saknar avancerade funktioner: Git Bash är enklare och saknar vissa avancerade funktioner som finns i andra terminaler, som Windows Terminal eller WSL (Windows Subsystem for Linux).

Integration med andra verktyg: Det kan vara svårare att integrera Git Bash med IDE:er eller andra verktyg jämfört med vissa GUI-baserade lösningar.

Källor

- 11 Git-Cheat-Sheet Mosh.pdf - Google Drive
- Git - Documentation
- Git Bash Tutorial | Git Bash Basics | Git and GitHub Tutorial | DevOps Training | Edureka - YouTube