**Základy systému Git**

**Základy systému Git**

Jak bychom tedy mohli Git charakterizovat? Odpověď na tuto otázku je velmi důležitá, protože pokud pochopíte, co je Git a na jakém principu pracuje, budete ho bezpochyby moci používat mnohem efektivněji. Při seznámení se systémem Git se pokuste zapomenout na vše, co už možná víte o jiných systémech VCS, např. Subversion nebo Perforce. Vyhnete se tak nežádoucím vlivům, které by vás mohly při používání systému Git mást. Ačkoli je uživatelské rozhraní velmi podobné, Git ukládá a zpracovává informace poněkud odlišně od ostatních systémů. Pochopení těchto rozdílů vám pomůže předejít nejasnostem, které mohou při používání systému Git vzniknout.

[**Snímky, nikoli rozdíly**](https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git#Snímky,-nikoli-rozdíly)

Hlavním rozdílem mezi systémem Git a všemi ostatními systémy VCS (včetně Subversion a jemu podobných) je způsob, jakým Git zpracovává data. Většina ostatních systémů ukládá informace jako seznamy změn jednotlivých souborů. Tyto systémy (CVS, Perforce, Bazaar atd.) chápou uložené informace jako sadu souborů a seznamů změn těchto souborů v čase — viz obrázek 1-4.



Obrázek 1-4. Ostatní systémy ukládají data jako změny v základní verzi každého souboru.

Git zpracovává data jinak. Chápe je spíše jako sadu snímků (snapshots) vlastního malého systému souborů. Pokaždé, když v systému zapíšete (uložíte) stav projektu, Git v podstatě „vyfotí“, jak vypadají všechny vaše soubory v daném okamžiku, a uloží reference na tento snímek. Pokud v souborech nebyly provedeny žádné změny, Git v zájmu zefektivnění práce neukládá znovu celý soubor, ale pouze odkaz na předchozí identický soubor, který už byl uložen. Zpracování dat v systému Git ilustruje obrázek 1-5.



Obrázek 1-5. Git ukládá data jako snímky projektu proměnlivé v čase.

Toto je důležitý rozdíl mezi systémem Git a téměř všemi ostatními systémy VCS. Git díky tomu znovu zkoumá skoro každý aspekt správy verzí, které ostatní systémy kopírovaly z předchozí generace. Git se podobá malému systému souborů (spíše než obyčejnému VCS) s řadou skutečně výkonných nástrojů, jež jsou na něm postavené. Některé přednosti, které tato metoda správy dat nabízí, si podrobně ukážeme na systému větvení v kapitole 3.

[**Téměř každá operace je lokální**](https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git#Téměř-každá-operace-je-lokální)

Většina operací v systému Git vyžaduje ke své činnosti pouze lokální soubory a zdroje a nejsou potřeba informace z jiných počítačů v síti. Pokud jste zvyklí pracovat se systémy CVCS, kde je většina operací poznamenána latencí sítě, patrně vás při práci v systému Git napadne, že mu bohové rychlosti dali do vínku nadpřirozené schopnosti. Protože máte celou historii projektu uloženou přímo na svém lokálním disku, probíhá většina operací takřka okamžitě.

Pokud chcete například procházet historii projektu, Git kvůli tomu nemusí vyhledávat informace na serveru — načte je jednoduše přímo z vaší lokální databáze. Znamená to, že se historie projektu zobrazí téměř hned. Pokud si chcete prohlédnout změny provedené mezi aktuální verzí souboru a týmž souborem před měsícem, Git vyhledá měsíc starý soubor a provede lokální výpočet rozdílů, aniž by o to musel žádat vzdálený server nebo stahovat starší verzi souboru ze vzdáleného serveru a poté provádět lokální výpočet.

To také znamená, že je jen velmi málo operací, které nemůžete provádět offline nebo bez připojení k VPN. Jste-li v letadle nebo ve vlaku a chcete pokračovat v práci, můžete beze všeho zapisovat nové revize. Ty odešlete až po opětovném připojení k síti. Jestliže přijedete domů a zjistíte, že VPN klient nefunguje, stále můžete pracovat. V mnoha jiných systémech je takový postup nemožný nebo přinejmenším obtížný. Například v systému Perforce toho lze bez připojení k serveru dělat jen velmi málo, v systémech Subversion a CVS můžete sice upravovat soubory, ale nemůžete zapisovat změny do databáze, neboť ta je offline. Možná to vypadá jako maličkost, ale divili byste se, jak velký je v tom rozdíl.

[**Git pracuje důsledně**](https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git#Git-pracuje-důsledně)

Než je v systému Git cokoli uloženo, je nejprve proveden kontrolní součet, který je potom používán k identifikaci uloženého souboru. Znamená to, že není možné změnit obsah jakéhokoli souboru nebo adresáře, aniž by o tom Git nevěděl. Tato funkce je integrována do systému Git na nejnižších úrovních a je nedílnou součástí jeho filozofie. Nemůže tak dojít ke ztrátě informací při přenosu dat nebo k poškození souboru, aniž by to byl Git schopen zjistit.

Mechanismus, který Git k tomuto kontrolnímu součtu používá, se nazývá otisk SHA-1 (SHA-1 hash). Jedná se o řetězec o 40 hexadecimálních znacích (0–9; a–f) vypočítaný na základě obsahu souboru nebo adresářové struktury systému Git. Otisk SHA-1 může vypadat například takto:

24b9da6552252987aa493b52f8696cd6d3b00373

S těmito otisky se budete setkávat ve všech úložištích systému Git, protože je používá opravdu často. Git nic neukládá podle názvu souboru. Místo toho používá databázi adresovatelnou hodnotou otisku, který odpovídá obsahu souboru.

[**Git většinou jen přidává data**](https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git#Git-většinou-jen-přidává-data)

Jednotlivé operace ve většině případů jednoduše přidávají data do Git databáze. Přimět systém, aby udělal něco, co nelze vzít zpět, nebo aby smazal jakákoli data, je velice obtížné. Stejně jako ve všech systémech VCS můžete ztratit nebo nevratně zničit změny, které ještě nebyly zapsány. Jakmile však jednou zapíšete snímek do systému Git, je téměř nemožné ho ztratit, zvlášť pokud pravidelně zálohujete databázi do jiného repozitáře.

Díky tomu vás bude práce se systémem Git bavit. Budete pracovat s vědomím, že můžete experimentovat, a neriskujete přitom nevratné zničení své práce. Podrobnější informace o tom, jak Git ukládá data a jak lze obnovit zdánlivě ztracenou práci, najdete v kapitole 9.

[**Tři stavy**](https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git#Tři-stavy)

A nyní pozor. Pokud chcete dále hladce pokračovat ve studiu Git, budou pro vás následující informace stěžejní. Git používá pro spravované soubory tři základní stavy: zapsáno (committed), změněno (modified) a připraveno k zapsání (staged). Zapsáno znamená, že jsou data bezpečně uložena ve vaší lokální databázi. Změněno znamená, že v souboru byly provedeny změny, avšak soubor ještě nebyl zapsán do databáze. Připraveno k zapsání znamená, že jste změněný soubor v jeho aktuální verzi určili k tomu, aby byl zapsán v další revizi (tzv. commit).

Z toho vyplývá, že projekt je v systému Git rozdělen do tří hlavních částí: adresář systému Git (Git directory), pracovní adresář (working directory) a oblast připravených změn (staging area).



Obrázek 1-6. Pracovní adresář, oblast připravených změn a adresář Git

Adresář Git (repozitář) je místo, kde Git uchovává metadata a databázi objektů vašeho projektu. Je to nejdůležitější část systému Git a zároveň adresář, který se zkopíruje, když klonujete repozitář z jiného počítače.

Pracovní adresář obsahuje lokální kopii jedné verze projektu. Tyto soubory jsou staženy ze zkomprimované databáze v adresáři Git a umístěny na disk, kde je můžete používat nebo upravovat.

Oblast připravených změn je jednoduchý soubor, většinou uložený v adresáři Git, který obsahuje informace o tom, co bude obsahovat příští revize. Soubor se někdy označuje také anglickým výrazem „index“, ale oblast připravených změn (staging area) je už dnes termín běžnější.

Standardní pracovní postup vypadá v systému Git následovně:

1. Změníte soubory ve svém pracovním adresáři.
2. Soubory připravíte k uložení tak, že vložíte jejich snímky do oblasti připravených změn.
3. Zapíšete revizi. Snímky souborů, uložené v oblasti připravených změn, se trvale uloží do adresáře Git.

Nachází-li se konkrétní verze souboru v adresáři Git, je považována za zapsanou. Pokud je modifikovaná verze přidána do oblasti připravených změn, je považována za připravenou k zapsání. A pokud byla od poslední operace checkout změněna, ale nebyla připravena k zapsání, je považována za změněnou. O těchto stavech, způsobech jak je co nejlépe využívat nebo i o tom, jak přeskočit proces připravení souborů, se dozvíte v kapitole 2.