Git aneb správa verzí trochu jinak

Jan ”Yenya” Kasprzak
<kas@fi.muni.cz>
http://www.fi.muni.cz/~kas/

Masarykova univerzita

XXXIV. konference EurOpen.CZ

Obsah přednášky

1. Co je verzovací systém?
2. Distribuované verzovací systémy
3. Git – úvod
4. Git – architektura
5. Git – specifické vlastnosti
6. Závěr

Co je verzovací systém?

1 Co je verzovací systém?

2 Distribuované verzovací systémy

3 Git – úvod

4 Git – architektura

5 Git – specifické vlastnosti

6 Závěr

Jak se jmenuje tato hra?

Version Control System (VCS) – systém řízení verzí.

Source Code Management (SCM) – správa zdrojových textů.

krátce a nepřesně:

Verzovací systém

Jak se jmenuje tato hra?

Version Control System (VCS) – systém řízení verzí.

Source Code Management (SCM) – správa zdrojových textů.

krátce a nepřesně:

Verzovací systém

Jak se jmenuje tato hra?

Version Control System (VCS) – systém řízení verzí.

Source Code Management (SCM) – správa zdrojových textů.

krátce a nepřesně:

Verzovací systém

Jak se jmenuje tato hra?

Version Control System (VCS) – systém řízení verzí.

Source Code Management (SCM) – správa zdrojových textů.

krátce a nepřesně:

Verzovací systém

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Co umí verzovací systém?

- Ukládat data (**revize**) do archívu (**repozitáře**) – operace **commit**.
- Obnovit data z libovolné doby (**checkout**).
- Porovnávat verze dat (**diff**).
- Pracovat s metadaty – autor změny, komentář, atd.
- Anotovat data – *kdo napsal tento kus kódu?*
- Arbitrace přístupu více vývojářů.
- Označkovat verzi symbolickým jménem (`v2.6.30`).
- Skriptovat akce – např. zaslání mailu po *commit*.
- Větvit vývoj – např. stabilní a vývojová větev.
- Slučovat větve – např. začlenění experimentální vlastnosti.

Distribuované verzovací systémy

1 Co je verzovací systém?

2 Distribuované verzovací systémy

3 Git – úvod

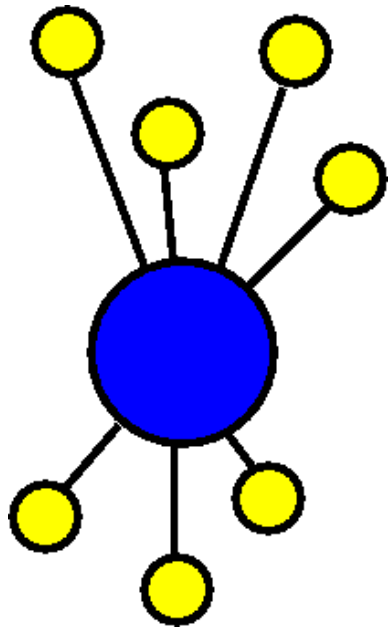
4 Git – architektura

5 Git – specifické vlastnosti

6 Závěr

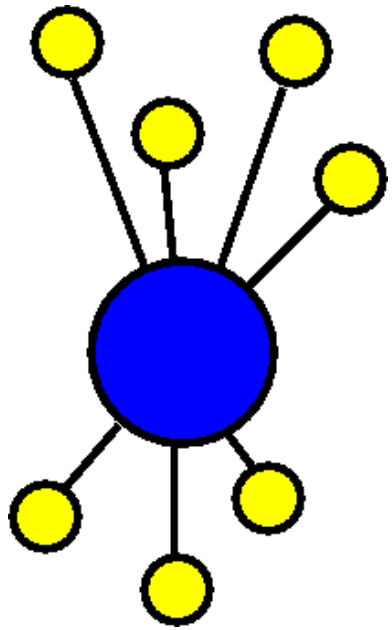
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



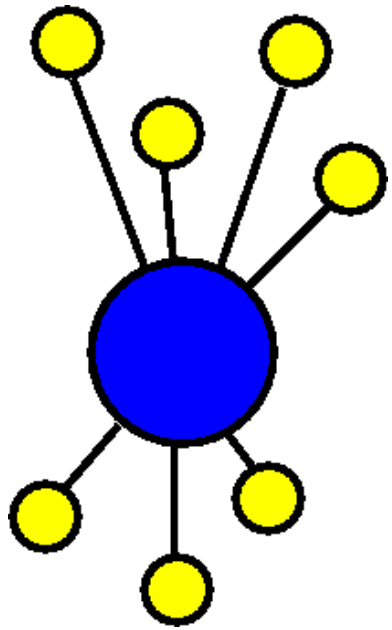
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



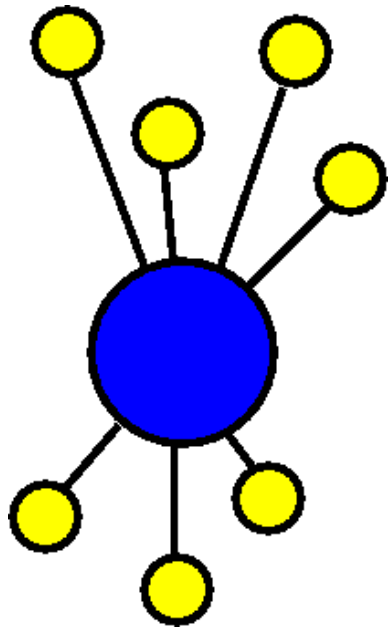
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



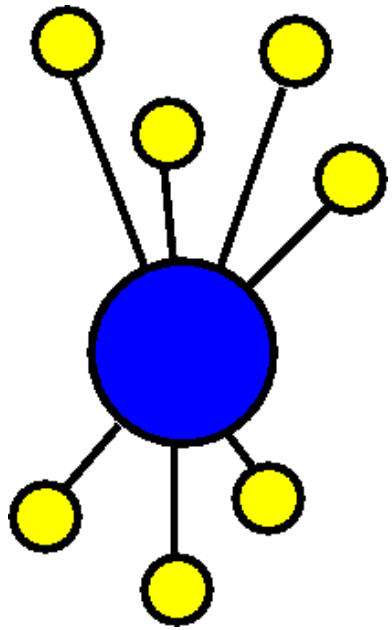
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



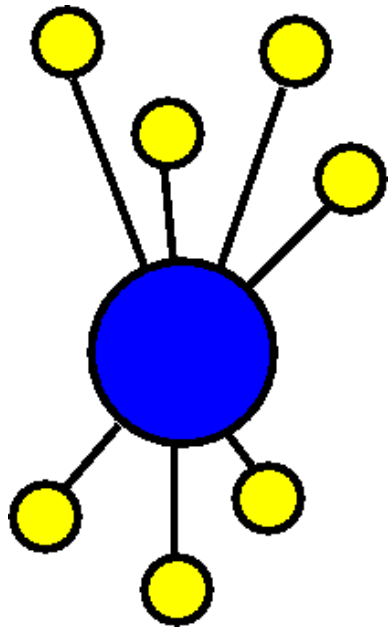
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



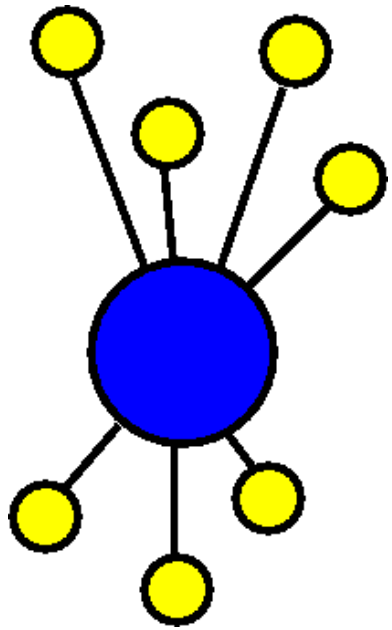
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



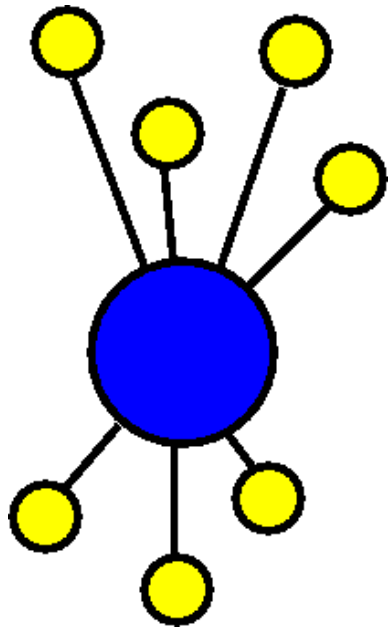
Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



Co je centralizovaný systém?

- Centralizované systémy: CVS, Subversion, RCS, ...
- Starší přístup.
- Repozitář na centrálním serveru.
- Pracovní kopie – nad nimi probíhá vývoj.
- Paralelní přístup:
  - lock/modify/commit nebo
  - modify/merge/commit.
- Náhled na data: verze je snímek dat v určitém čase.



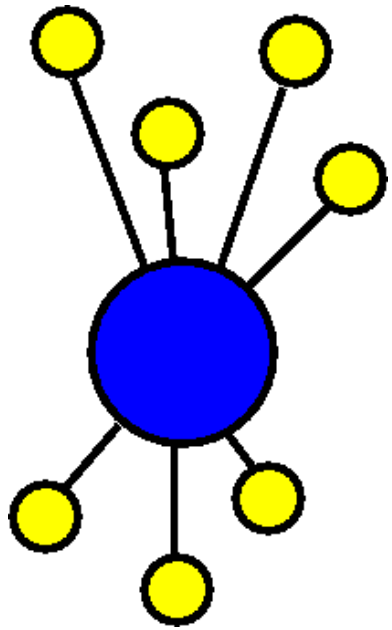
Problémy centralizovaných systémů

Those who still use a centralized SCM system should be in some mental institution.

—Linus Torvalds v Google Tech Talks

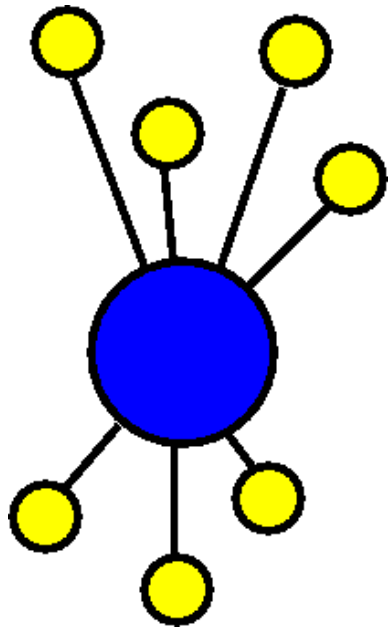
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



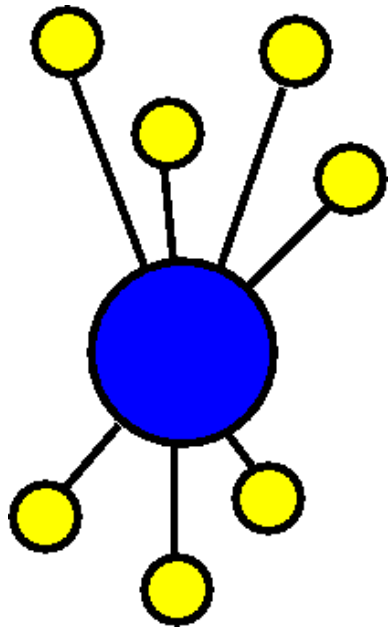
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



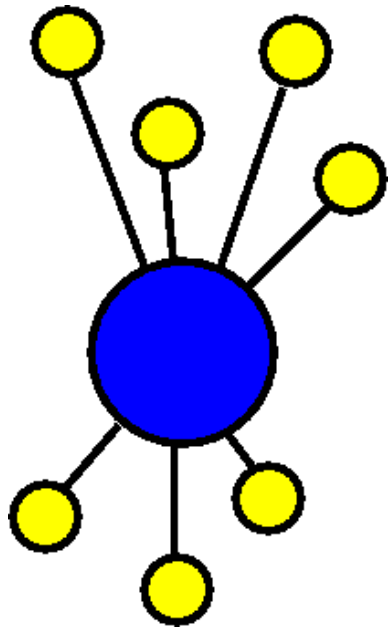
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



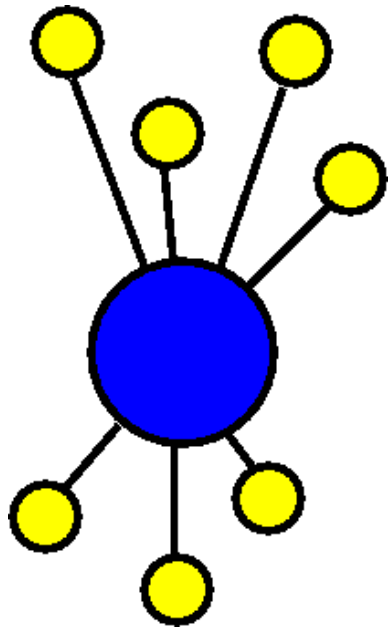
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



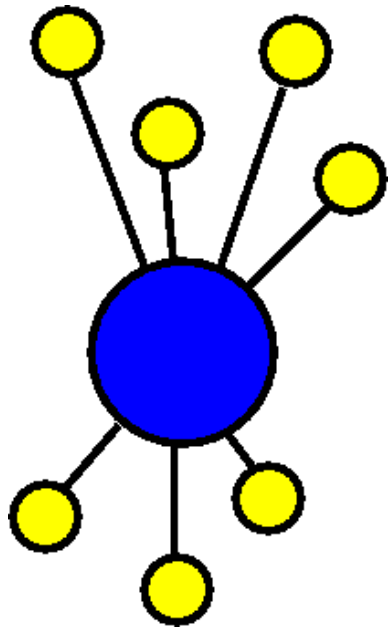
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



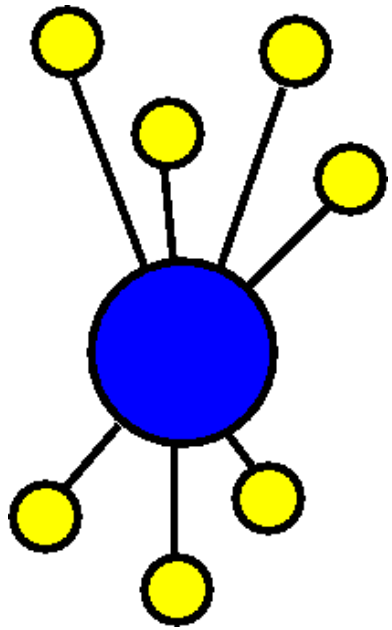
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



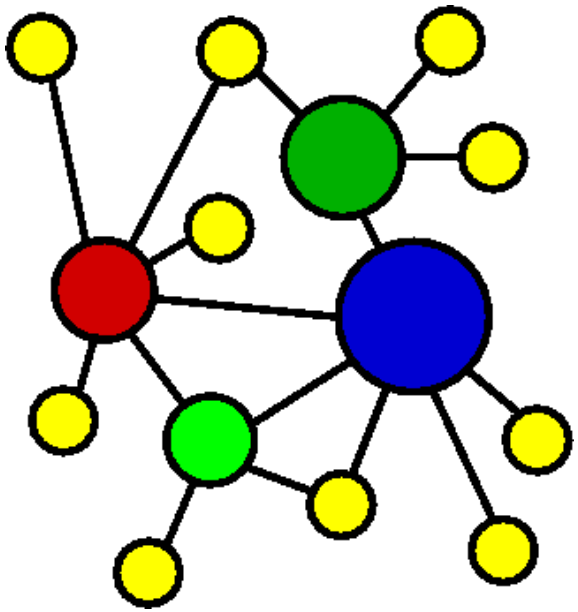
… a teď vážně:

- Nelze odpojené operace – na chatě, v letadle, …
- Těžko dělat změny do jiných větví.
- Větvit je snadné, slučovat nemožné.
- Commit 1x za den – fuj!
- Do repozitáře vidí všichni – nelze dělat experimenty.
- Těžko experimentovat ve více lidech.
- „core team“ versus ostatní svět.



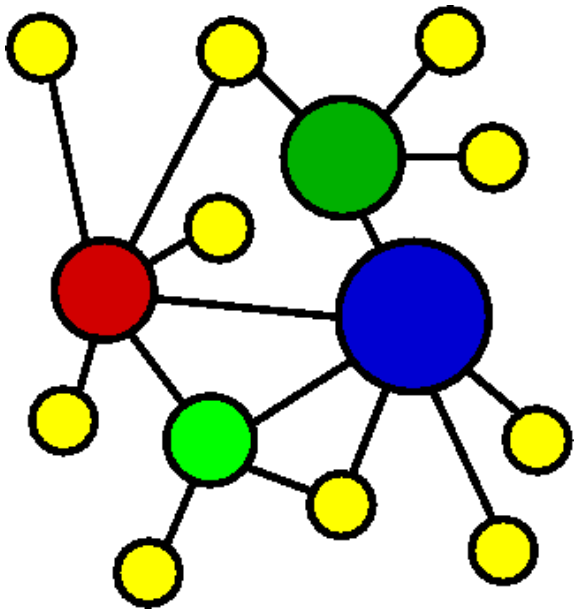
Distribuované systémy

- Není centrální repozitář.
- Každý repozitář je plnohodnotný.
- Operace **clone** místo **checkout**.
- Repozitář == pracovní kopie.
- Publikování změn – zveřejnění svého repozitáře, zaslání záplaty mailem, ...
- Cizí repozitář == vzdálená větev.



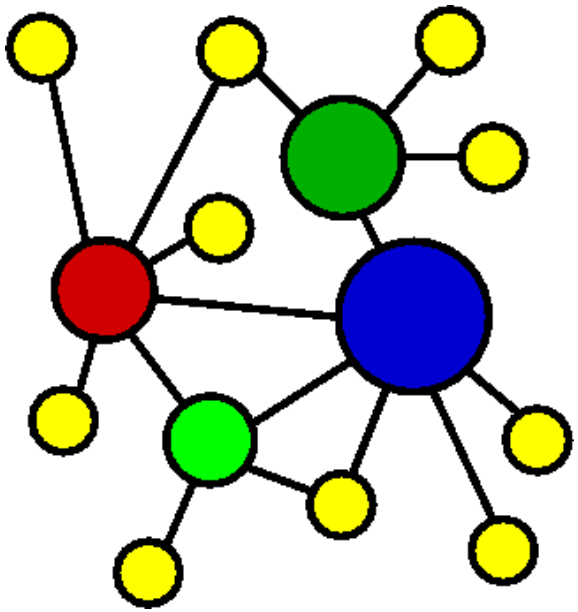
Distribuované systémy

- Není centrální repozitář.
- Každý repozitář je plnohodnotný.
- Operace **clone** místo **checkout**.
- Repozitář == pracovní kopie.
- Publikování změn – zveřejnění svého repozitáře, zaslání záplaty mailem, ...
- Cizí repozitář == vzdálená větev.



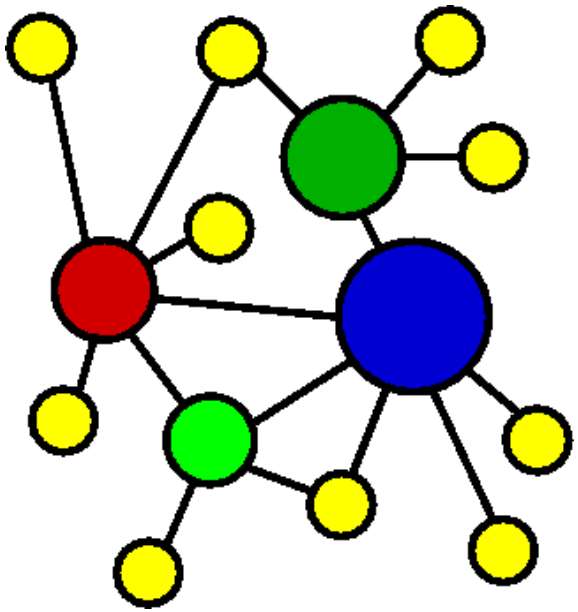
Distribuované systémy

- Není centrální repozitář.
- Každý repozitář je plnohodnotný.
- Operace **clone** místo **checkout**.
- Repozitář == pracovní kopie.
- Publikování změn – zveřejnění svého repozitáře, zaslání záplaty mailem, ...
- Cizí repozitář == vzdálená větev.



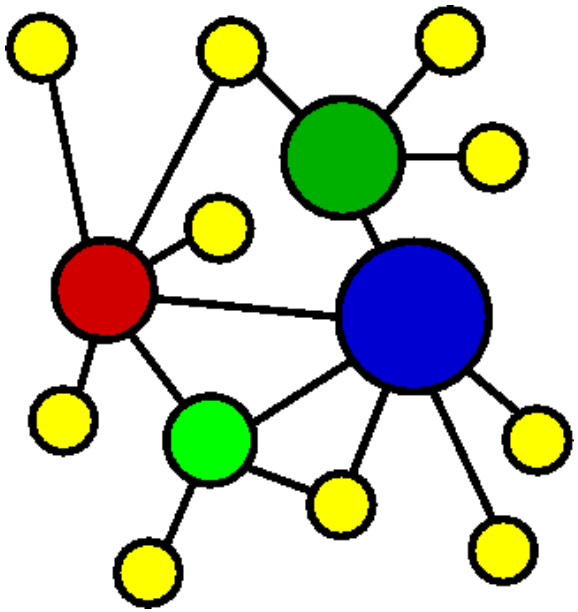
Distribuované systémy

- Není centrální repozitář.
- Každý repozitář je plnohodnotný.
- Operace **clone** místo **checkout**.
- Repozitář == pracovní kopie.
- Publikování změn – zveřejnění svého repozitáře, zaslání záplaty mailem, ...
- Cizí repozitář == vzdálená větev.



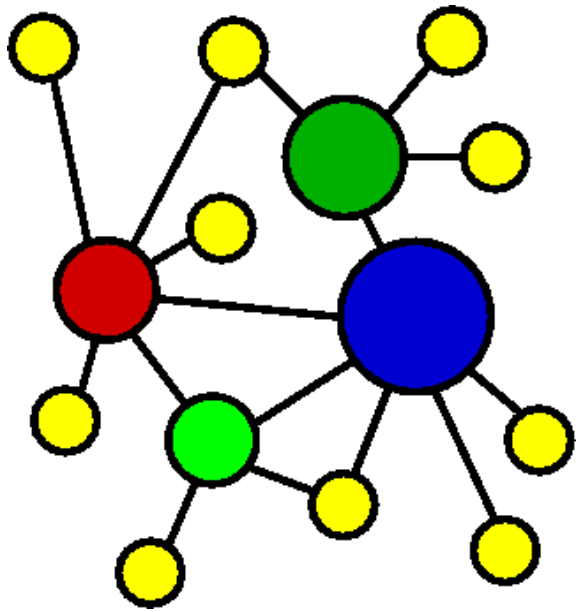
Distribuované systémy

- Není centrální repozitář.
- Každý repozitář je plnohodnotný.
- Operace **clone** místo **checkout**.
- Repozitář == pracovní kopie.
- Publikování změn – zveřejnění svého repozitáře, zaslání záplaty mailem, ...
- Cizí repozitář == vzdálená větev.



Distribuované systémy

- Není centrální repozitář.
- Každý repozitář je plnohodnotný.
- Operace **clone** místo **checkout**.
- Repozitář == pracovní kopie.
- Publikování změn – zveřejnění svého repozitáře, zaslání záplaty mailem, ...
- Cizí repozitář == vzdálená větev.



Práce s daty v distribuovaném VCS

- Distribuovanost: snímek dat – nezajímavý.
- „kudy jsme se sem dostali?“
- Sady změn (changesets).
- Odkazy na rodičovské sady změn.
- Historie vývoje: orientovaný acyklický graf
- Verze – předchozí sady změn dosažitelné z jistého místa.

Práce s daty v distribuovaném VCS

- Distribuovanost: snímek dat – nezajímavý.
- „kudy jsme se sem dostali?“
- Sady změn (changesets).
- Odkazy na rodičovské sady změn.
- Historie vývoje: orientovaný acyklický graf
- Verze – předchozí sady změn dosažitelné z jistého místa.

Práce s daty v distribuovaném VCS

- Distribuovanost: snímek dat – nezajímavý.
- „kudy jsme se sem dostali?“
- Sady změn (changesets).
- Odkazy na rodičovské sady změn.
- Historie vývoje: orientovaný acyklický graf
- Verze – předchozí sady změn dosažitelné z jistého místa.

Práce s daty v distribuovaném VCS

- Distribuovanost: snímek dat – nezajímavý.
- „kudy jsme se sem dostali?“
- Sady změn (changesets).
- Odkazy na rodičovské sady změn.
- Historie vývoje: orientovaný acyklický graf
- Verze – předchozí sady změn dosažitelné z jistého místa.

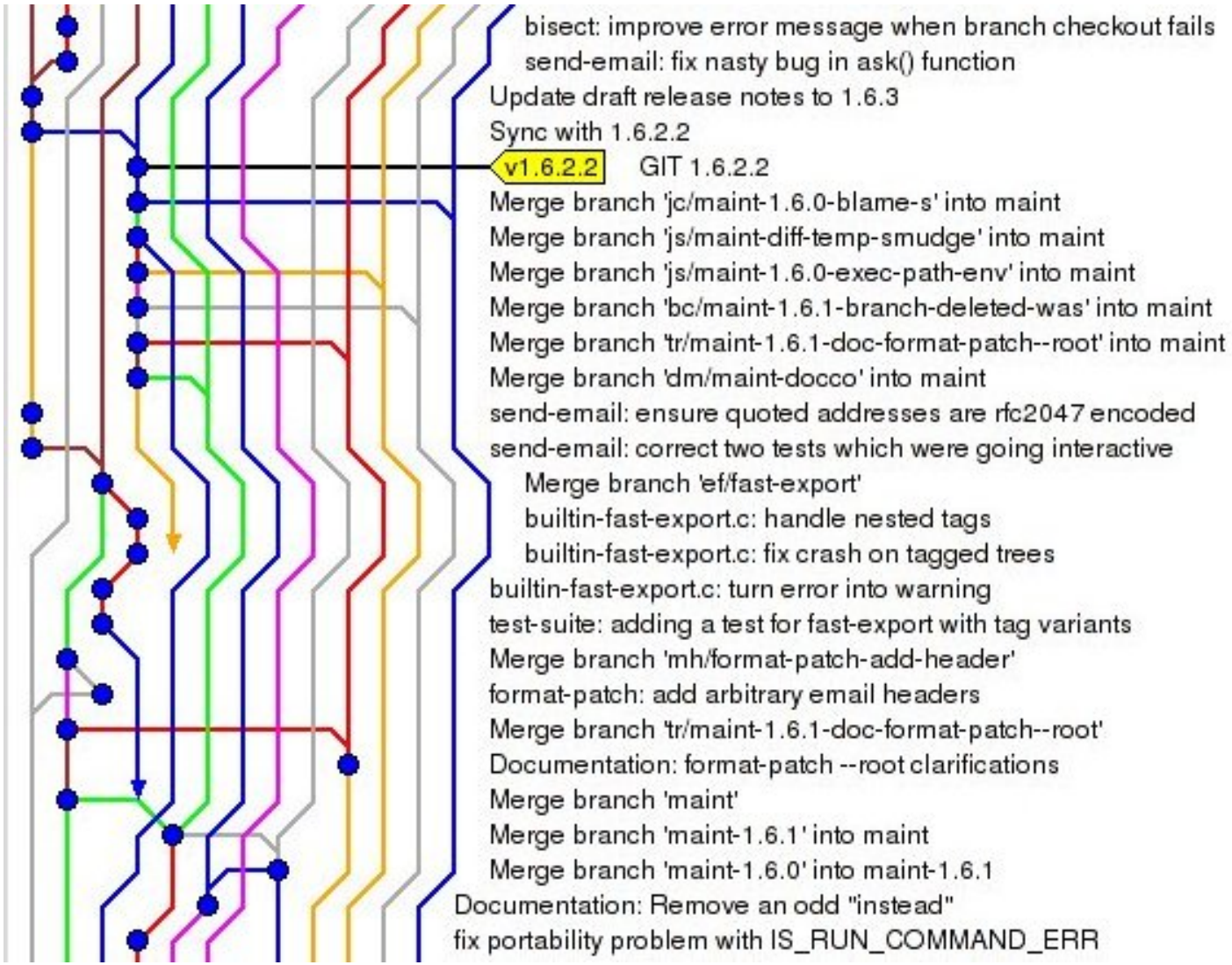
Práce s daty v distribuovaném VCS

- Distribuovanost: snímek dat – nezajímavý.
- „kudy jsme se sem dostali?“
- Sady změn (changesets).
- Odkazy na rodičovské sady změn.
- Historie vývoje: orientovaný acyklický graf
- Verze – předchozí sady změn dosažitelné z jistého místa.

Práce s daty v distribuovaném VCS

- Distribuovanost: snímek dat – nezajímavý.
- „kudy jsme se sem dostali?“
- Sady změn (changesets).
- Odkazy na rodičovské sady změn.
- Historie vývoje: orientovaný acyklický graf
- Verze – předchozí sady změn dosažitelné z jistého místa.

Příklad: sady změn v distribuovaném VCS



Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

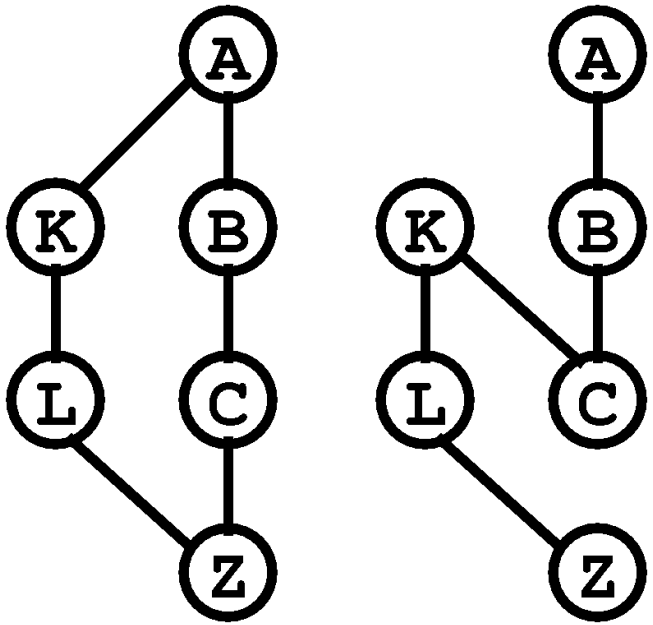
Větve a distribuovaný VCS

- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Větve a distribuovaný VCS

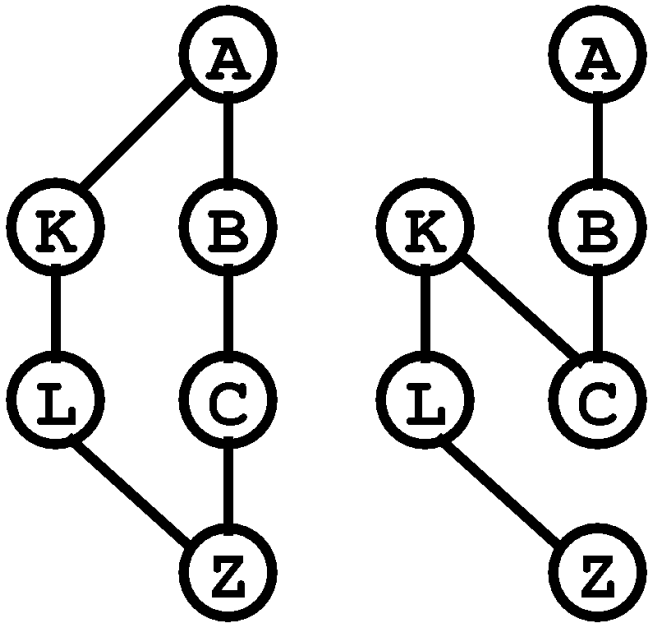
- Klon repozitáře == vzdálená větev.
- Větve se používají častěji než u centrálních VCS.
- Větvení musí být jednoduché, ale hlavně:
- slučování větví musí být jednoduché!
- Rozvětvení – sada změn s více přímými potomky.
- Slučující sada změn – více rodičů.
- Opakované slučování – vývoj obou větví pokračuje dál.
- Rychlé sloučení větví (fast-forward merge) – linearizace části grafu.

Fast-forward merge



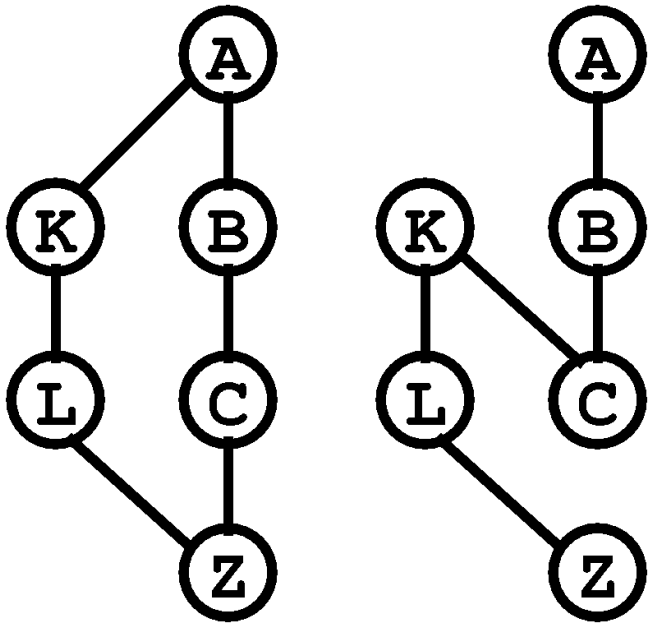
- Nelze-li fast-forward, je slučovací sada změn neprázdná.
- Cherry-pick – aplikace jen některých sad změn.
- VCS ví, které sady změn u nás už máme.

Fast-forward merge



- Nelze-li fast-forward, je slučovací sada změn neprázdná.
- Cherry-pick – aplikace jen některých sad změn.
- VCS ví, které sady změn u nás už máme.

Fast-forward merge



- Nelze-li fast-forward, je slučovací sada změn neprázdná.
- Cherry-pick – aplikace jen některých sad změn.
- VCS ví, které sady změn u nás už máme.

Výhody distribuovaných VCS

- Verzovací systém i bez přístupu k síti.
- Pískoviště – větve i pro experimentální práci.
- Repozitář – statické soubory: není třeba speciální server.
- Commit early, commit often. Malé, dobře komentované změny.
- Ad-hoc spolupráce bez účasti „centra“.
- Méně politikaření v projektech.

Výhody distribuovaných VCS

- Verzovací systém i bez přístupu k síti.
- Pískoviště – větve i pro experimentální práci.
- Repozitář – statické soubory: není třeba speciální server.
- Commit early, commit often. Malé, dobře komentované změny.
- Ad-hoc spolupráce bez účasti „centra“.
- Méně politikaření v projektech.

Výhody distribuovaných VCS

- Verzovací systém i bez přístupu k síti.
- Pískoviště – větve i pro experimentální práci.
- Repozitář – statické soubory: není třeba speciální server.
- Commit early, commit often. Malé, dobře komentované změny.
- Ad-hoc spolupráce bez účasti „centra“.
- Méně politikaření v projektech.

Výhody distribuovaných VCS

- Verzovací systém i bez přístupu k síti.
- Pískoviště – větve i pro experimentální práci.
- Repozitář – statické soubory: není třeba speciální server.
- Commit early, commit often. Malé, dobře komentované změny.
- Ad-hoc spolupráce bez účasti „centra“.
- Méně politikaření v projektech.

Výhody distribuovaných VCS

- Verzovací systém i bez přístupu k síti.
- Pískoviště – větve i pro experimentální práci.
- Repozitář – statické soubory: není třeba speciální server.
- Commit early, commit often. Malé, dobře komentované změny.
- Ad-hoc spolupráce bez účasti „centra“.
- Méně politikaření v projektech.

Výhody distribuovaných VCS

- Verzovací systém i bez přístupu k síti.
- Pískoviště – větve i pro experimentální práci.
- Repozitář – statické soubory: není třeba speciální server.
- Commit early, commit often. Malé, dobře komentované změny.
- Ad-hoc spolupráce bez účasti „centra“.
- Méně politikaření v projektech.

Git – úvod

1 Co je verzovací systém?

2 Distribuované verzovací systémy

3 Git – úvod

4 Git – architektura

5 Git – specifické vlastnosti

6 Závěr

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Historie Gitu

- Jádro Linuxu – dlouho bez verzovacího systému.
- 2002 – BitKeeper – komerční distribuovaný VCS.
- GNU Arch – open source distribuovaný VCS.
- Monotone – SHA1 jako jednoznačný identifikátor.
- 2005 – Andrew Tridgell: reverse engineering BitKeeperu.
- Ukončení licence BitKeeperu.
- Linus Torvalds: architektura systému **Git**. Důraz na rychlost.

Git – architektura

1 Co je verzovací systém?

2 Distribuované verzovací systémy

3 Git – úvod

4 Git – architektura

5 Git – specifické vlastnosti

6 Závěr

Architektura

Příkazy

- `git` *příkaz* – např. `git blame`.
- `git-`*příkaz* – např. `git-format-patch`.

Konfigurace

- Globální – `~/.gitconfig`; styl `.ini`.
- Lokální – v repozitáři.
- `git config --global user.name 'Yenya Kasprzak'`
- `git config --global user.email kas@fi.muni.cz`

Architektura

Příkazy

- `git` *příkaz* – např. `git blame`.
- `git-`*příkaz* – např. `git-format-patch`.

Konfigurace

- Globální – `~/.gitconfig`; styl `.ini`.
- Lokální – v repozitáři.
- `git config --global user.name 'Yenya Kasprzak'`
- `git config --global user.email kas@fi.muni.cz`

Architektura

Příkazy

- `git` *příkaz* – např. `git blame`.
- `git-`*příkaz* – např. `git-format-patch`.

Konfigurace

- Globální – `~/.gitconfig`; styl `.ini`.
- Lokální – v repozitáři.
- `git config --global user.name 'Yenya Kasprzak'`
- `git config --global user.email kas@fi.muni.cz`

Architektura

Příkazy

- `git` *příkaz* – např. `git blame`.
- `git-`*příkaz* – např. `git-format-patch`.

Konfigurace

- Globální – `~/.gitconfig`; styl `.ini`.
- Lokální – v repozitáři.
- `git config --global user.name 'Yenya Kasprzak'`
- `git config --global user.email kas@fi.muni.cz`

Architektura

Příkazy

- `git` *příkaz* – např. `git blame`.
- `git-`*příkaz* – např. `git-format-patch`.

Konfigurace

- Globální – `~/.gitconfig`; styl `.ini`.
- Lokální – v repozitáři.
- **`git config --global user.name 'Yenya Kasprzak'`**
- `git config --global user.email kas@fi.muni.cz`

Architektura

Příkazy

- `git` *příkaz* – např. `git blame`.
- `git-`*příkaz* – např. `git-format-patch`.

Konfigurace

- Globální – `~/.gitconfig`; styl `.ini`.
- Lokální – v repozitáři.
- **`git config --global user.name 'Yenya Kasprzak'`**
- **`git config --global user.email kas@fi.muni.cz`**

Repozitář

- Běžný adresář, podadresář `.git`.
- Obsahuje pracovní kopii dat.
- Nebo neobsahuje: bare repository.

Vytvoření repozitáře

```
$ mkdir mujprojekt; cd mujprojekt; git init [--bare]
$ git clone [--bare] http://server/projekt.git; cd projekt
```

Repozitář

- Běžný adresář, podadresář `.git`.
- Obsahuje pracovní kopii dat.
- Nebo neobsahuje: bare repository.

Vytvoření repozitáře

```
$ mkdir mujprojekt; cd mujprojekt; git init [--bare]
$ git clone [--bare] http://server/projekt.git; cd projekt
```

Repozitář

- Běžný adresář, podadresář `.git`.
- Obsahuje pracovní kopii dat.
- Nebo neobsahuje: bare repository.

Vytvoření repozitáře

```
$ mkdir mujprojekt; cd mujprojekt; git init [--bare]
$ git clone [--bare] http://server/projekt.git; cd projekt
```

Repozitář

- Běžný adresář, podadresář `.git`.
- Obsahuje pracovní kopii dat.
- Nebo neobsahuje: bare repository.

Vytvoření repozitáře

```
$ mkdir mujprojekt; cd mujprojekt; git init [--bare]
$ git clone [--bare] http://server/projekt.git; cd projekt
```

Repozitář

- Běžný adresář, podadresář `.git`.
- Obsahuje pracovní kopii dat.
- Nebo neobsahuje: bare repository.

Vytvoření repozitáře

```
$ mkdir mujprojekt; cd mujprojekt; git init [--bare]
$ git clone [--bare] http://server/projekt.git; cd projekt
```

Blob

- Git je obsahově adresovatelný filesystém.
- Ukládá bloby.
- Identifikace: SHA1 hash.
- Soubor je také blob.

Blob

- Git je obsahově adresovatelný filesystém.
- Ukládá bloby.
- Identifikace: SHA1 hash.
- Soubor je také blob.

Blob

- Git je obsahově adresovatelný filesystém.
- Ukládá bloby.
- Identifikace: SHA1 hash.
- Soubor je také blob.

Blob

- Git je obsahově adresovatelný filesystém.
- Ukládá bloby.
- Identifikace: SHA1 hash.
- Soubor je také blob.

Soubor je blob

Příklad

```
$ echo ahoj > pozdrav.txt
$ git hash-object pozdrav.txt
36a0e7cf17ffe584221529d45113d821e70764a9
$ git add pozdrav.txt
$ git cat-file -p 36a0e7cf17
ahoj
```

Strom

- Snímek projektu je strom (tree).
- Strom – jména souborů a jejich SHA1.
- Strom je blob – identifikován svým SHA1.

Strom

- Snímek projektu je strom (tree).
- Strom – jména souborů a jejich SHA1.
- Strom je blob – identifikován svým SHA1.

Strom

- Snímek projektu je strom (tree).
- Strom – jména souborů a jejich SHA1.
- Strom je blob – identifikován svým SHA1.

Strom je blob

Příklad

```
$ git write-tree
f1d1e2642787c59d61f01075e44b780e60ede900
$ git cat-file -p f1d1
100644 blob 36a0e7cf17ffe584221529d4...4a9 pozdrav.txt
```

- Strom je to, co přidal `git-add`.
- Interní příkaz, není nutné znát.

Strom je blob

Příklad

```
$ git write-tree
f1d1e2642787c59d61f01075e44b780e60ede900
$ git cat-file -p f1d1
100644 blob 36a0e7cf17ffe584221529d4...4a9 pozdrav.txt
```

- Strom je to, co přidal `git-add`.
- Interní příkaz, není nutné znát.

Strom je blob

Příklad

```
$ git write-tree
f1d1e2642787c59d61f01075e44b780e60ede900
$ git cat-file -p f1d1
100644 blob 36a0e7cf17ffe584221529d4...4a9 pozdrav.txt
```

- Strom je to, co přidal `git-add`.
- Interní příkaz, není nutné znát.

Commit

- Commit – revize (sada změn).
- Odkaz na strom – jak vypadají finální data.
- Komentář – popis změn.
- Rodičovské commity – 0 nebo více – proti čemu je změna.
- Commit je blob – identifikován svým SHA1.
- Bezpečnost – commit identifikuje celou historii!

Commit

- Commit – revize (sada změn).
- Odkaz na strom – jak vypadají finální data.
- Komentář – popis změn.
- Rodičovské commity – 0 nebo více – proti čemu je změna.
- Commit je blob – identifikován svým SHA1.
- Bezpečnost – commit identifikuje celou historii!

Commit

- Commit – revize (sada změn).
- Odkaz na strom – jak vypadají finální data.
- Komentář – popis změn.
- Rodičovské commity – 0 nebo více – proti čemu je změna.
- Commit je blob – identifikován svým SHA1.
- Bezpečnost – commit identifikuje celou historii!

Commit

- Commit – revize (sada změn).
- Odkaz na strom – jak vypadají finální data.
- Komentář – popis změn.
- Rodičovské commity – 0 nebo více – proti čemu je změna.
- Commit je blob – identifikován svým SHA1.
- Bezpečnost – commit identifikuje celou historii!

Commit

- Commit – revize (sada změn).
- Odkaz na strom – jak vypadají finální data.
- Komentář – popis změn.
- Rodičovské commity – 0 nebo více – proti čemu je změna.
- Commit je blob – identifikován svým SHA1.
- Bezpečnost – commit identifikuje celou historii!

Commit

- Commit – revize (sada změn).
- Odkaz na strom – jak vypadají finální data.
- Komentář – popis změn.
- Rodičovské commity – 0 nebo více – proti čemu je změna.
- Commit je blob – identifikován svým SHA1.
- Bezpečnost – commit identifikuje celou historii!

Jak vypadá commit?

Příklad

```
$ git commit -m 'Prvni verze'
Created initial commit f2ebc71: Prvni verze
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 pozdrav.txt
$ git cat-file -p f2eb
tree f1d1e2642787c59d61f01075e44b780e60ede900
author Yenya Kasprzak <kas@fi.muni.cz> 1240789610 +0200
committer Yenya Kasprzak <kas@fi.muni.cz> 1240789610 +0200

Prvni verze
```

Běžná práce

Příklad

```
$ echo nazdar >> pozdrav.txt
$ git add pozdrav.txt
$ git commit -m 'Pridan pozdrav "nazdar".'
Created commit b400b96: Pridan pozdrav "nazdar".
1 files changed, 1 insertions(+), 0 deletions(-)
$ git cat-file -p b400b96
tree f3424e549969ff3293ddb04970bde3128c836887
parent f2ebc71c3ca437c668c99731f6ae13efe1cb2f18
author Yenya Kasprzak <kas@fi.muni.cz> 1240790025 +0200
committer Yenya Kasprzak <kas@fi.muni.cz> 1240790025 +0200

Pridan pozdrav "nazdar".
```

Větve

- Větev – odkaz na commit.
- `.git/refs/heads/`*název* – 41 bajtů.
- Záměr: větev pro každý kus vývoje.
- Implicitní větev: master.
- Commit do větve – přesun odkazu (hlavy).
- Pojmenování větve je lokální.

Větve

- Větev – odkaz na commit.
- `.git/refs/heads/`*název* – 41 bajtů.
- Záměr: větev pro každý kus vývoje.
- Implicitní větev: master.
- Commit do větve – přesun odkazu (hlavy).
- Pojmenování větve je lokální.

Větve

- Větev – odkaz na commit.
- `.git/refs/heads/`*název* – 41 bajtů.
- Záměr: větev pro každý kus vývoje.
- Implicitní větev: master.
- Commit do větve – přesun odkazu (hlavy).
- Pojmenování větve je lokální.

Větve

- Větev – odkaz na commit.
- `.git/refs/heads/`*název* – 41 bajtů.
- Záměr: větev pro každý kus vývoje.
- Implicitní větev: master.
- Commit do větve – přesun odkazu (hlavy).
- Pojmenování větve je lokální.

Větve

- Větev – odkaz na commit.
- `.git/refs/heads/`*název* – 41 bajtů.
- Záměr: větev pro každý kus vývoje.
- Implicitní větev: master.
- Commit do větve – přesun odkazu (hlavy).
- Pojmenování větve je lokální.

Větve

- Větev – odkaz na commit.
- `.git/refs/heads/`*název* – 41 bajtů.
- Záměr: větev pro každý kus vývoje.
- Implicitní větev: master.
- Commit do větve – přesun odkazu (hlavy).
- Pojmenování větve je lokální.

Použití větví

Příklad

```
$ git checkout -b japonsky
Switched to a new branch "japonsky"
$ git branch
  master
* japonsky
$ echo konnichiwa >> pozdrav.txt
$ git add pozdrav.txt
$ git commit -m 'japonsky pozdrav'
Created commit ce1bbf5: japonsky pozdrav
1 files changed, 1 insertions(+), 0 deletions(-)
```

Slučování větví

Příklad

```
$ git checkout master
Switched to branch "master"
$ cat pozdrav.txt
ahoj
nazdar
$ git merge japonsky
Updating b400b96..ce1bbf5
Fast forward
pozdrav.txt | 1 +
1 files changed, 1 insertions(+), 0 deletions(-)
$ cat pozdrav.txt
ahoj
nazdar
konnichiwa
```

Zrušení větve

Příklad

```
$ ls .git/refs/heads/
master japonsky
$ git branch -d japonsky
```

- Commit bez odkazu.
- Garbage collection – automaticky nebo `git-gc`.

Zrušení větve

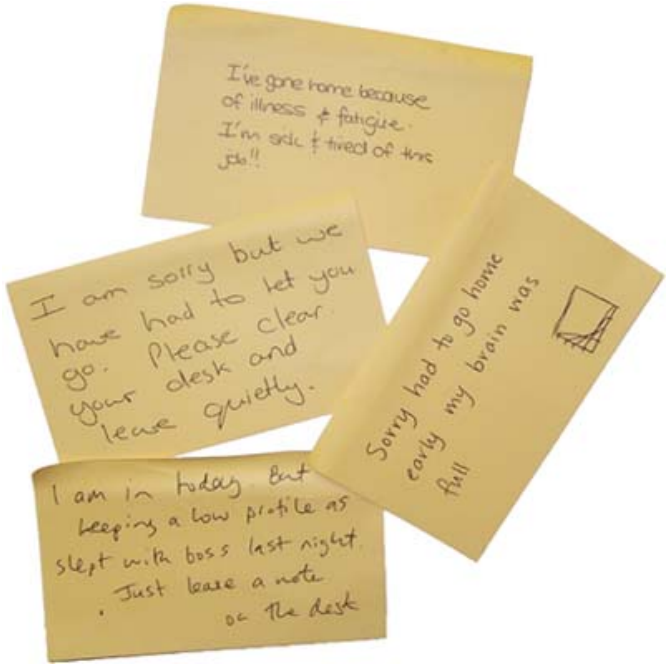
Příklad

```
$ ls .git/refs/heads/
master japonsky
$ git branch -d japonsky
```

- Commit bez odkazu.
- Garbage collection – automaticky nebo `git-gc`.

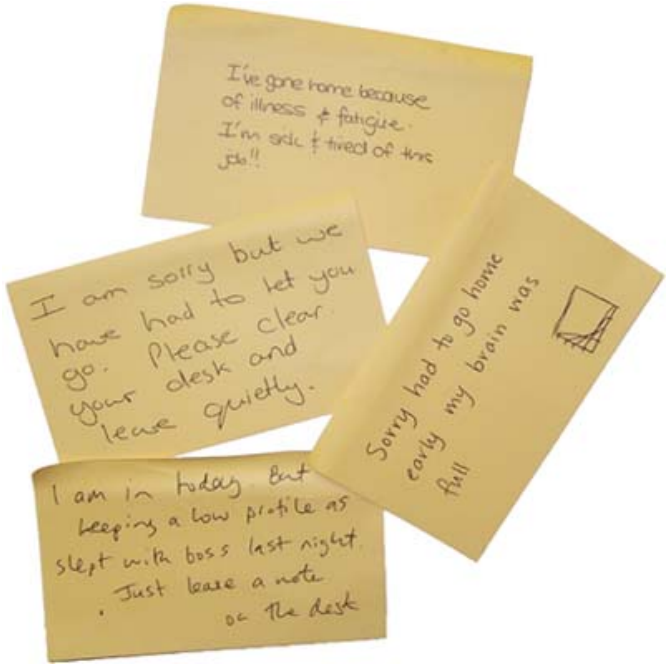
Štítky

- Štítek – odkaz na commit.
- Nemění se v čase – např. označení verze.
- `.git/refs/tags/`*název*.



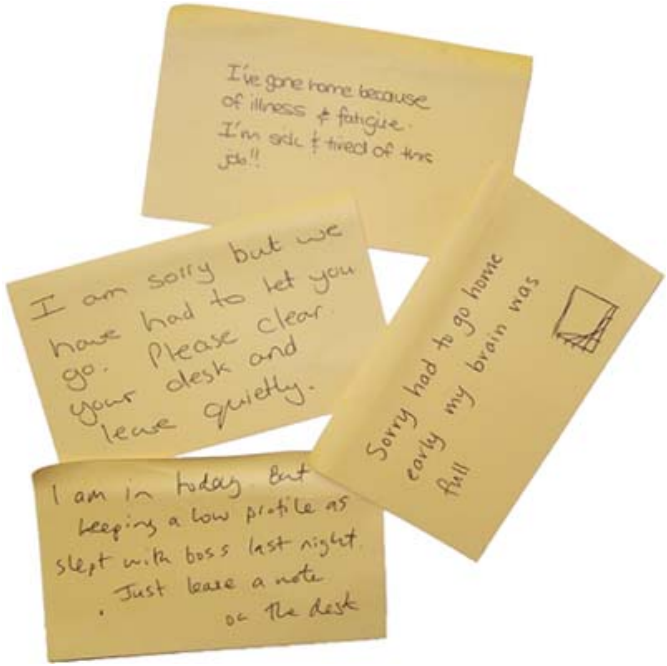
Štítky

- Štítek – odkaz na commit.
- Nemění se v čase – např. označení verze.
- `.git/refs/tags/`*název*.



Štítky

- Štítek – odkaz na commit.
- Nemění se v čase – např. označení verze.
- `.git/refs/tags/`*název*.



Git – specifické vlastnosti

1 Co je verzovací systém?

2 Distribuované verzovací systémy

3 Git – úvod

4 Git – architektura

5 Git – specifické vlastnosti

6 Závěr

Produktem je záplata

- Sada změn má být čitelná.
- Repozitář je lokální: editovatelné commity.
- `git-commit --amend`
- Needitovat již zveřejněné commity.
- Export větve jako sady záplat (`git-format-patch`, `git-am`).
- Export větve jako jedné záplaty.

Produktem je záplata

- Sada změn má být čitelná.
- Repozitář je lokální: editovatelné commity.
- `git-commit --amend`
- Needitovat již zveřejněné commity.
- Export větve jako sady záplat (`git-format-patch`, `git-am`).
- Export větve jako jedné záplaty.

Produktem je záplata

- Sada změn má být čitelná.
- Repozitář je lokální: editovatelné commity.
- `git-commit --amend`
- Needitovat již zveřejněné commity.
- Export větve jako sady záplat (`git-format-patch`, `git-am`).
- Export větve jako jedné záplaty.

Produktem je záplata

- Sada změn má být čitelná.
- Repozitář je lokální: editovatelné commity.
- `git-commit --amend`
- Needitovat již zveřejněné commity.
- Export větve jako sady záplat (`git-format-patch`, `git-am`).
- Export větve jako jedné záplaty.

Produktem je záplata

- Sada změn má být čitelná.
- Repozitář je lokální: editovatelné commity.
- `git-commit --amend`
- Needitovat již zveřejněné commity.
- Export větve jako sady záplat (`git-format-patch`, `git-am`).
- Export větve jako jedné záplaty.

Produktem je záplata

- Sada změn má být čitelná.
- Repozitář je lokální: editovatelné commity.
- `git-commit --amend`
- Needitovat již zveřejněné commity.
- Export větve jako sady záplat (`git-format-patch`, `git-am`).
- Export větve jako jedné záplaty.

Bisect

- Commit early, commit often: malé izolované změny.
- Linux – několik commitů za hodinu.
- Lokalizace problémů u uživatele.
- `git-bisect` – binární vyhledávání v grafu.

Bisect

- Commit early, commit often: malé izolované změny.
- Linux – několik commitů za hodinu.
- Lokalizace problémů u uživatele.
- `git-bisect` – binární vyhledávání v grafu.

Bisect

- Commit early, commit often: malé izolované změny.
- Linux – několik commitů za hodinu.
- Lokalizace problémů u uživatele.
- `git-bisect` – binární vyhledávání v grafu.

Bisect

- Commit early, commit often: malé izolované změny.
- Linux – několik commitů za hodinu.
- Lokalizace problémů u uživatele.
- `git-bisect` – binární vyhledávání v grafu.

Staging area

- Commit – neukládá pracovní kopii.
- Staging area (index) – data pro nový commit.
- `git-add` – přidá soubor do staging area.
- Nový commit lze postupně sestavovat.
- Nebo: `git commit -a`.

Staging area

- Commit – neukládá pracovní kopii.
- Staging area (index) – data pro nový commit.
- `git-add` – přidá soubor do staging area.
- Nový commit lze postupně sestavovat.
- Nebo: `git commit -a`.

Staging area

- Commit – neukládá pracovní kopii.
- Staging area (index) – data pro nový commit.
- `git-add` – přidá soubor do staging area.
- Nový commit lze postupně sestavovat.
- Nebo: `git commit -a`.

Staging area

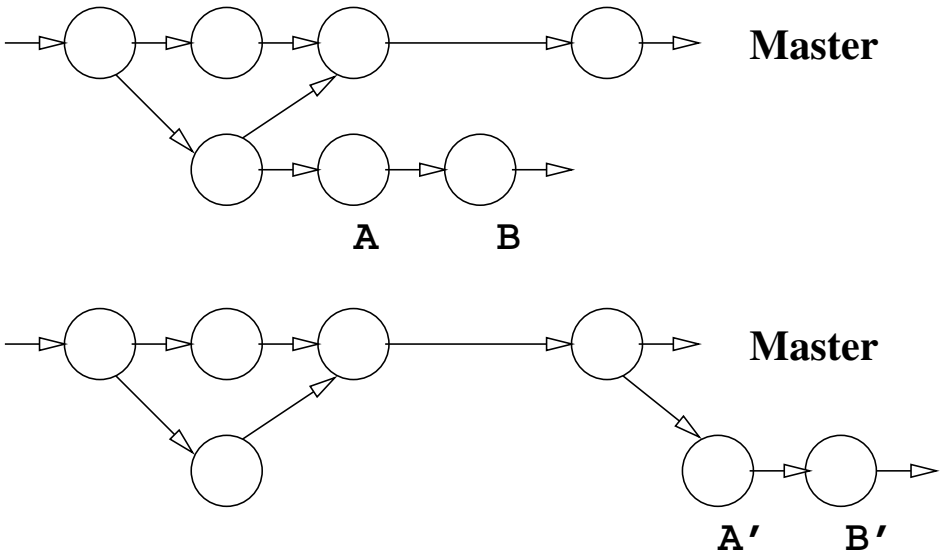
- Commit – neukládá pracovní kopii.
- Staging area (index) – data pro nový commit.
- `git-add` – přidá soubor do staging area.
- Nový commit lze postupně sestavovat.
- Nebo: `git commit -a`.

Staging area

- Commit – neukládá pracovní kopii.
- Staging area (index) – data pro nový commit.
- `git-add` – přidá soubor do staging area.
- Nový commit lze postupně sestavovat.
- Nebo: `git commit -a`.

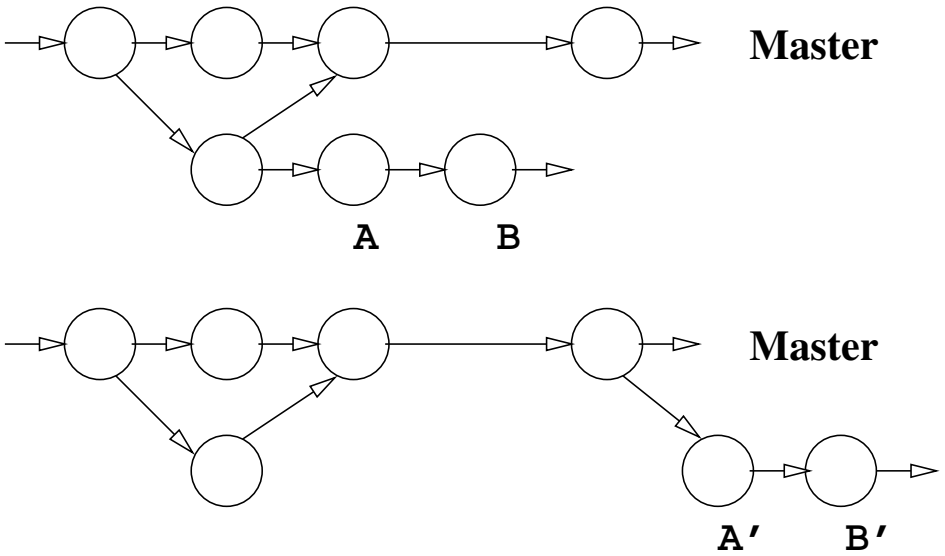
Merge versus rebase

- Delší vývoj – pravděpodobné konflikty.
- Pravidelné slučování (merge) proti upstream verzi.
- Čitelný commit – někdy lépe proti poslední verzi.
- Rebase – přesun větve nad jinou hlavu.



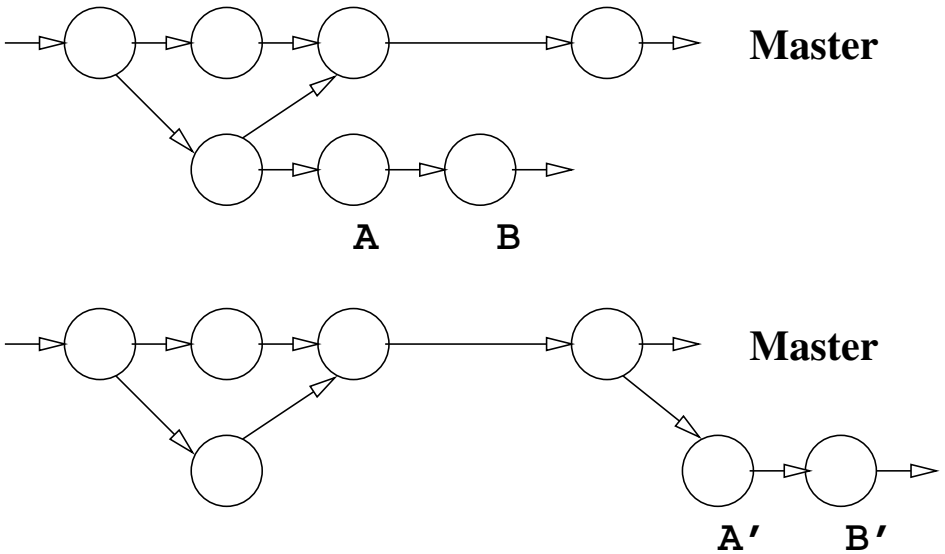
Merge versus rebase

- Delší vývoj – pravděpodobné konflikty.
- Pravidelné slučování (merge) proti upstream verzi.
- Čitelný commit – někdy lépe proti poslední verzi.
- Rebase – přesun větve nad jinou hlavu.



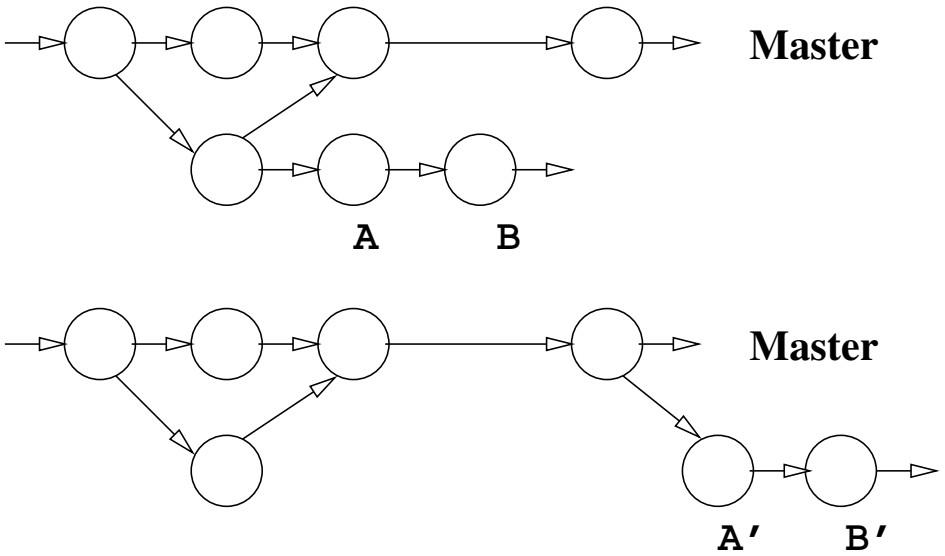
Merge versus rebase

- Delší vývoj – pravděpodobné konflikty.
- Pravidelné slučování (merge) proti upstream verzi.
- Čitelný commit – někdy lépe proti poslední verzi.
- Rebase – přesun větve nad jinou hlavu.



Merge versus rebase

- Delší vývoj – pravděpodobné konflikty.
- Pravidelné slučování (merge) proti upstream verzi.
- Čitelný commit – někdy lépe proti poslední verzi.
- Rebase – přesun větve nad jinou hlavu.



Stash

- Oprava jiné chyby během vývoje.
- Nová pracovní kopie – časově náročné.
- Commit – kazí vzhled.
- `git-stash` – dočasná větev.
- `git-stash apply` – návrat k předchozímu stavu.

Stash

- Oprava jiné chyby během vývoje.
- Nová pracovní kopie – časově náročné.
- Commit – kazí vzhled.
- `git-stash` – dočasná větev.
- `git-stash apply` – návrat k předchozímu stavu.

Stash

- Oprava jiné chyby během vývoje.
- Nová pracovní kopie – časově náročné.
- Commit – kazí vzhled.
- `git-stash` – dočasná větev.
- `git-stash apply` – návrat k předchozímu stavu.

Stash

- Oprava jiné chyby během vývoje.
- Nová pracovní kopie – časově náročné.
- Commit – kazí vzhled.
- `git-stash` – dočasná větev.
- `git-stash apply` – návrat k předchozímu stavu.

Stash

- Oprava jiné chyby během vývoje.
- Nová pracovní kopie – časově náročné.
- Commit – kazí vzhled.
- `git-stash` – dočasná větev.
- `git-stash apply` – návrat k předchozímu stavu.

Závěr

1 Co je verzovací systém?

2 Distribuované verzovací systémy

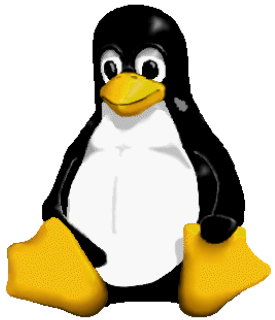
3 Git – úvod

4 Git – architektura

5 Git – specifické vlastnosti

6 Závěr

Uživatelé Gitu



Možná i vy?

`man gitcvs-migration`, Git-SVN Crash Course

Uživatelé Gitu

Možná i vy?

`man gitcvs-migration`, Git-SVN Crash Course

Shrnutí

- Distribuované SCM jsou lepší.
- Čitelná záplata, nejen zdrojový text.
- Kryptograficky ověřená historie.
- Zkuste si Git!

Shrnutí

- Distribuované SCM jsou lepší.
- Čitelná záplata, nejen zdrojový text.
- Kryptograficky ověřená historie.
- Zkuste si Git!

Shrnutí

- Distribuované SCM jsou lepší.
- Čitelná záplata, nejen zdrojový text.
- Kryptograficky ověřená historie.
- Zkuste si Git!

Shrnutí

- Distribuované SCM jsou lepší.
- Čitelná záplata, nejen zdrojový text.
- Kryptograficky ověřená historie.
- Zkuste si Git!

Dotazy?

Dotazy?

Děkuji za pozornost