# バージョン管理システム入門

～ CVS,Subversion,そしてGitまで ～

佐々木 洋平

uwabami@gfd-dennou.org

北大・理・宇宙理学/神大・理・地球惑星
地球流体電脳倶楽部

2008/10/03
神大 自然科学総合研究棟 3 号館 507

# おことわり



- 毎度毎度の無保証です.
  - 用法，用量を守って正しくお使い下さい.
- いい加減なことを言っているかもしれません.
  - 間違いなどありましたら，適宜コメントお願いします.
- 疑問，質問，合いの手，ツッコミ，大歓迎
  - 誰かがフォローしてくれることを期待しつつ...

# 今日の目標

# 今日の目標



- 「バージョン管理システム」を知らない人に
  - 知ってもらう
  - 普段の作業でも使いたくなるようにする
- 「バージョン管理システム」を知っている人に
  - CVS,Subversion
    - 比較
  - 分散バージョン管理
    - GitとCVS,Subversionの違いについて触れる

# Agenda



- バージョン管理システム入門
  - 概要
  - 集中リポジトリ型のワークフロー
    - CVS,Subversion
  - 歴史
- 分散リポジトリ型の紹介
  - arch,Bazaar,Git,Mercurial,Monotone

# Agenda



- バージョン管理システム入門
  - 概要
  - 集中リポジトリ型のワークフロー
    - CVS,Subversion
  - 歴史
- 分散リポジトリ型の紹介
  - arch,Bazaar,Git,Mercurial,Monotone

話者が使っているモノについてしか触れません

# 概要

# バージョン管理

# 一言で言えば

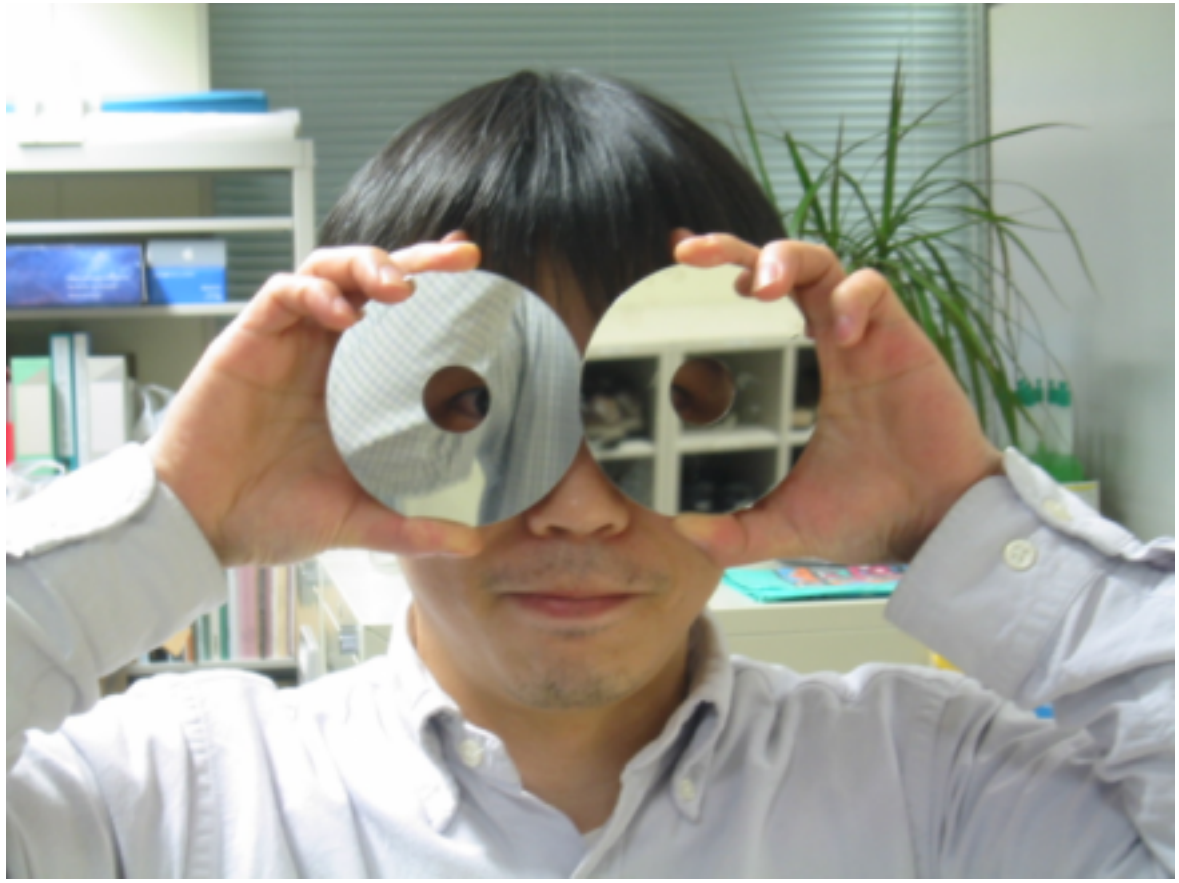


- 良い感じのバックアップのこと

# じゃあ



- 良い感じって?



# 良い感じ，じゃないと(1)

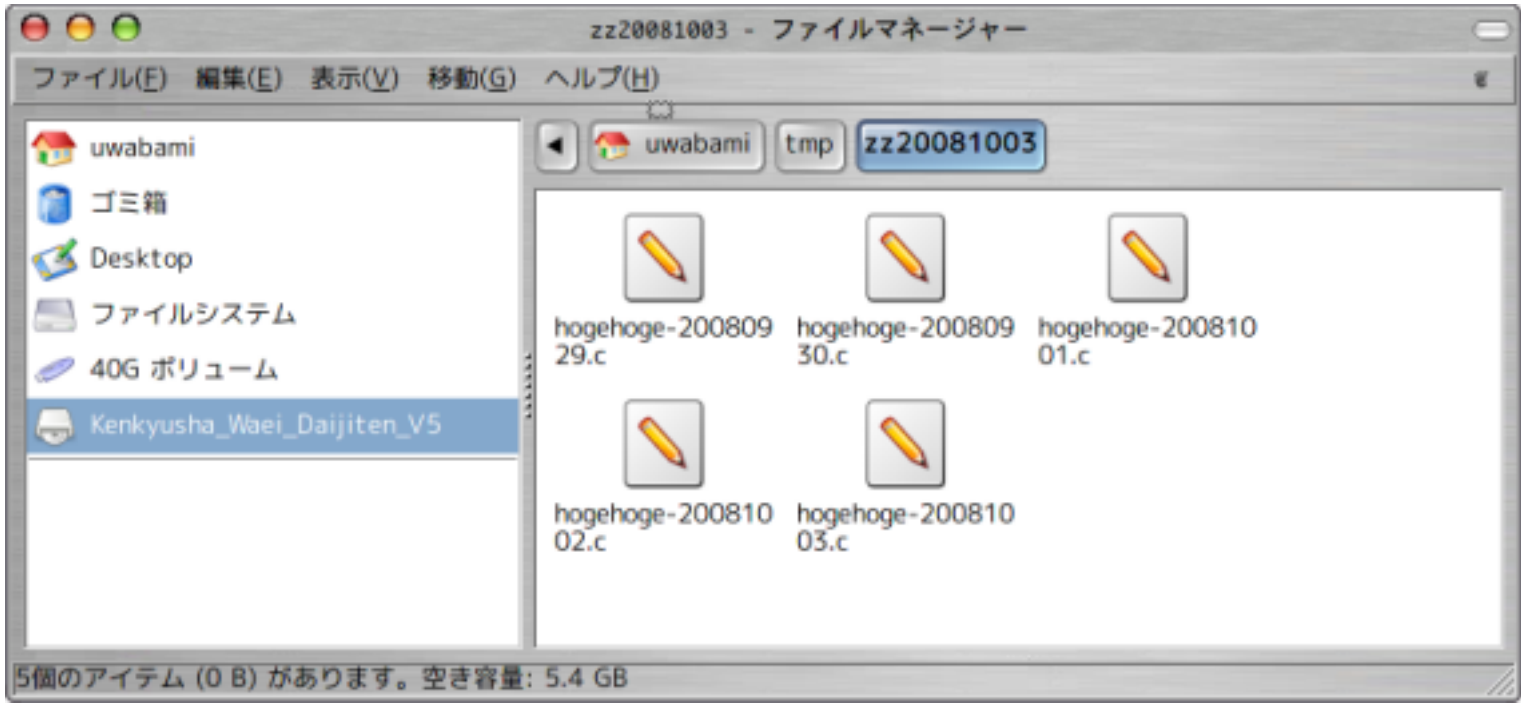


- 日付つきディレクトリ名
  - なにが変わったの?

# 良い感じ，じゃないと(2)



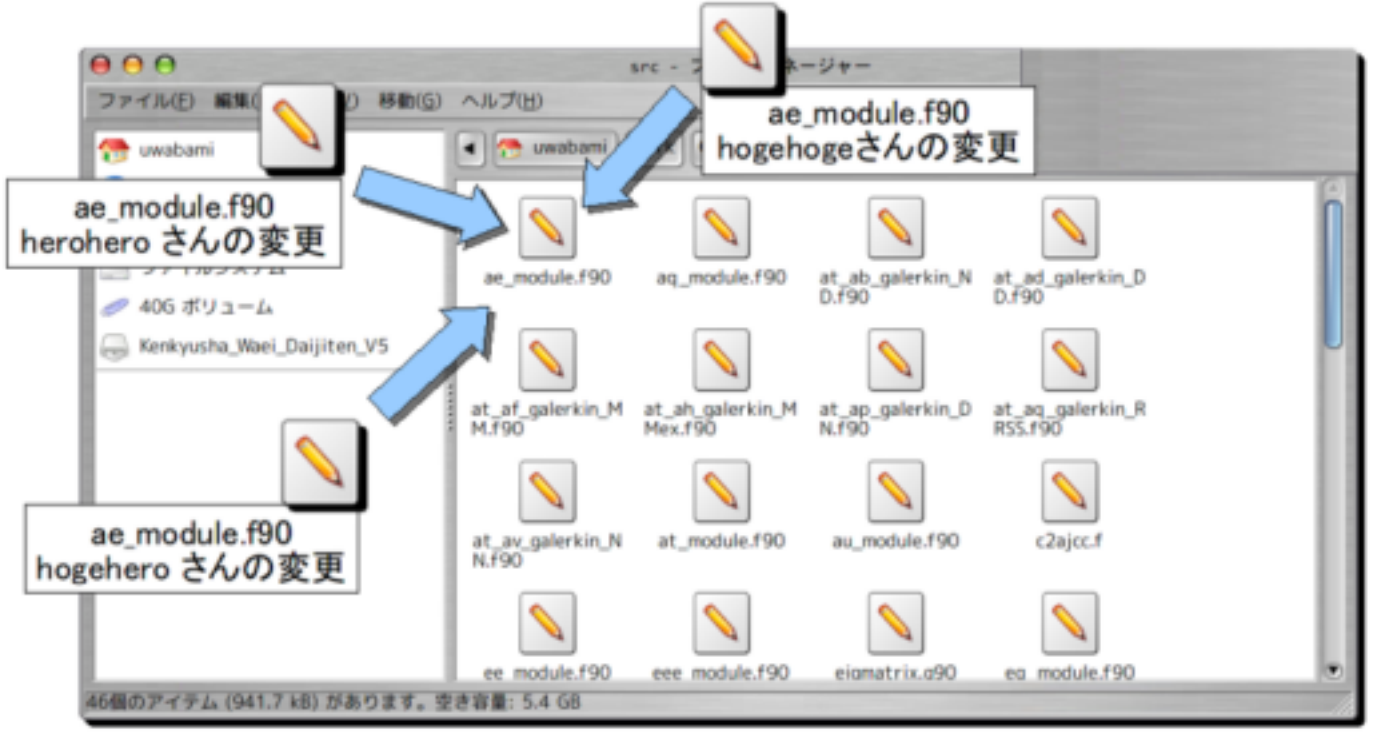
- 日付つきファイル名
  - 必要なのはどれ?



# 良い感じ，じゃないと(3)



- 複数人で作業する時に．
  - 変更をどう管理しましょう?



困った
ねぇ…
(´・ω・`)

そこで

バージョン管理

# バージョン管理をしておくと



- どのバージョンでも取得可能
  - "最新版","いつかの版"
- 変更履歴を辿れる
  - 誰が,いつ,どんな変更をしたのか
- 誰かの変更を上書きしても ok
  - いつでも元に戻せる

良い感じで日々の作業に打ち込めます.

# Agenda

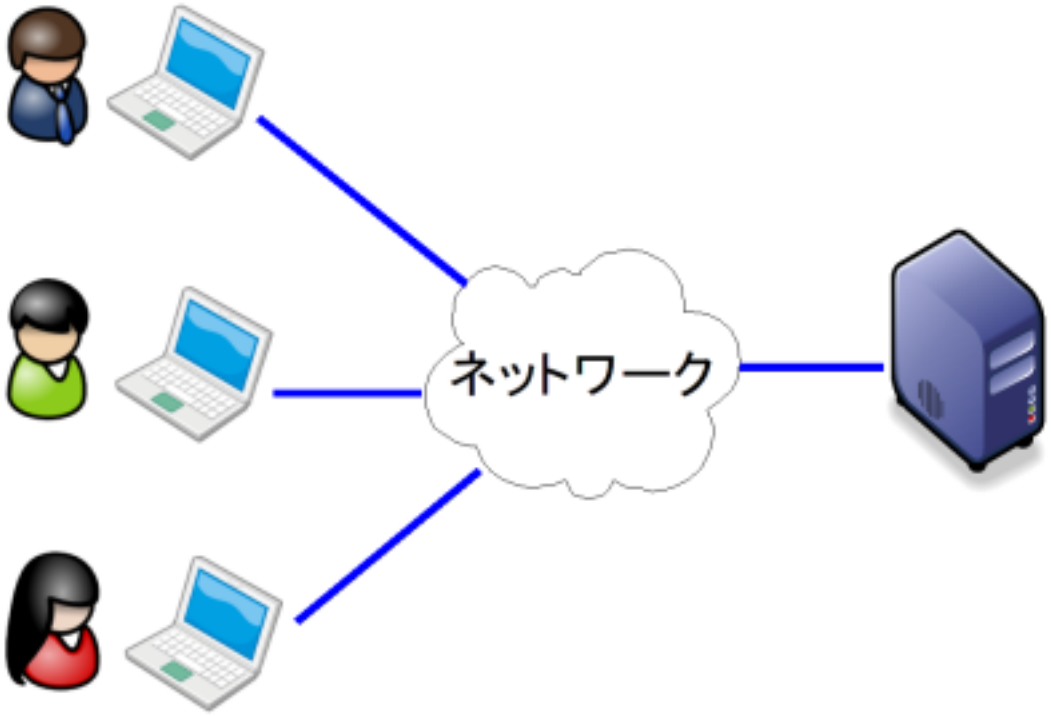


- バージョン管理システム入門
  - 概要
  - 集中リポジトリ型のワークフロー
    - CVS,Subversion
  - 歴史
- 分散リポジトリ型の紹介
  - arch,Bazaar,Git,Mercurial,Monotone

# CVS,Subversion の構成



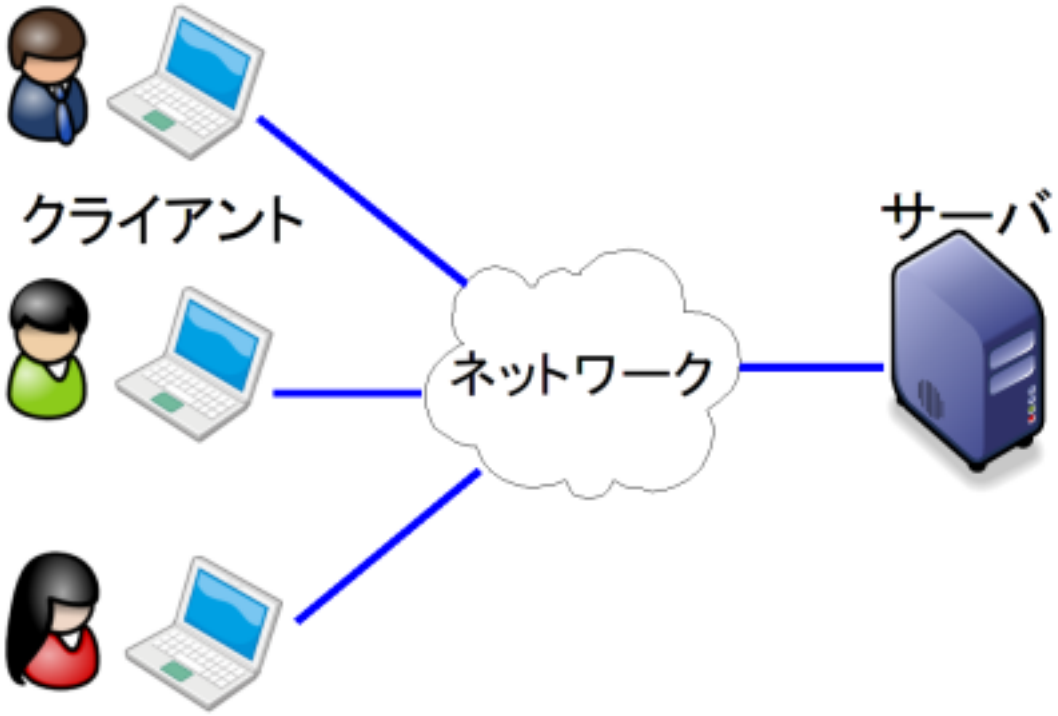
- 基本構成



# CVS,Subversion の構成



- サーバ & クライアント方式



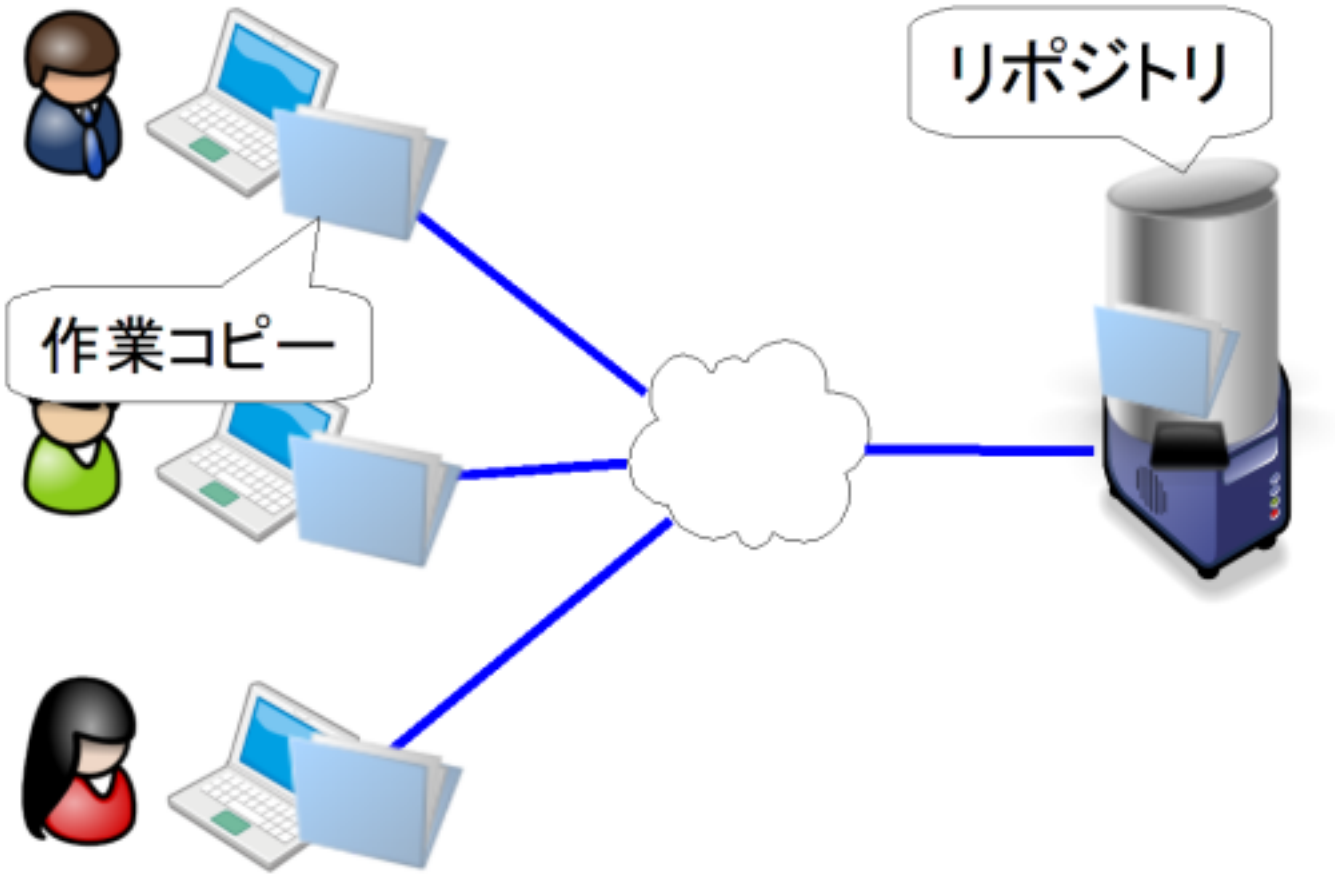
# サーバ & クライアント



- 近傍のサーバ
  - 電脳サーバ,epa サーバは cvs,svn 両方使えます
  - ep サーバ(www) は cvs が使えます
- クライアント
  - 話者はコマンドライン一辺倒なんで,良く知りません.
    - qct(GUI commit tool) なんかが良い感じらしいです.
  - Windows では WinCVS,TortoiseSVN など

# リポジトリと作業コピー





# リポジトリと作業コピー



- リポジトリ
  - 管理用の「容器,貯蔵庫」
  - 変更の内容が記録されていく所
- 作業コピー
  - リポジトリから取得して編集を行なう所

# ワークフロー



- 初期設定
  - リポジトリ作成
  - インポート
  - チェックアウト
- 日々の作業
  - 編集
  - コミット
  - アップデート

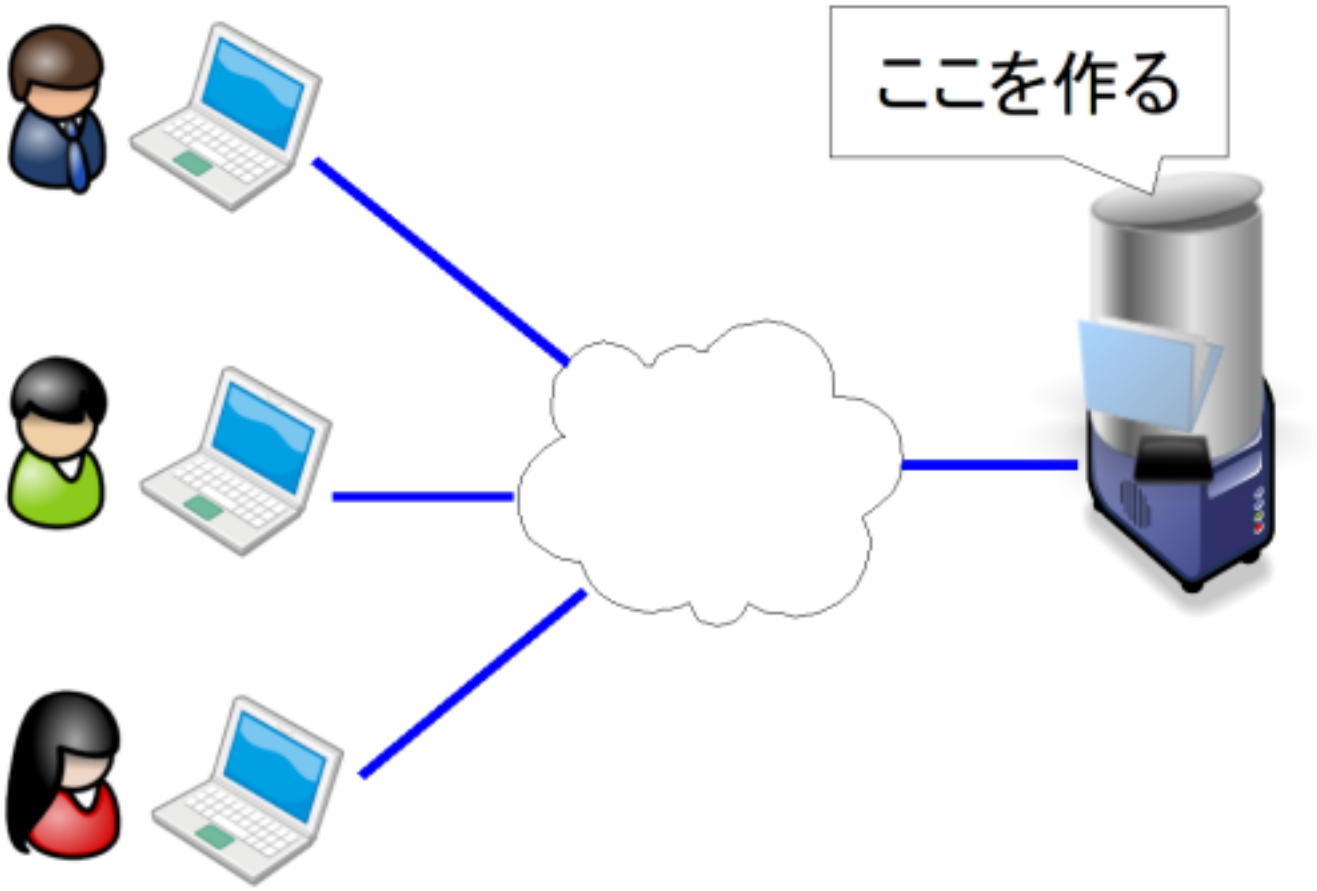
# ワークフロー



- 初期設定
  - リポジトリ作成 ←ココ
  - インポート
  - チェックアウト
- 日々の作業
  - 編集
  - コミット
  - アップデート

# 初期設定: リポジトリ作成





# ワークフロー

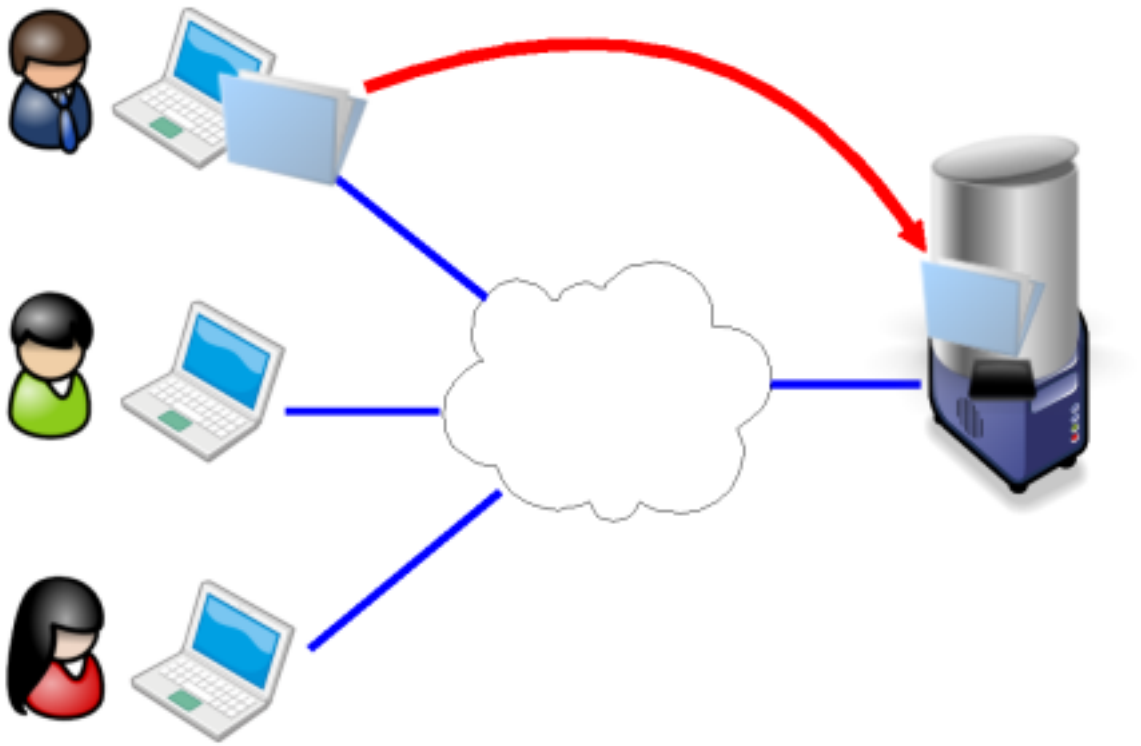


- 初期設定
  - リポジトリ作成
  - インポート ←ココ
  - チェックアウト
- 日々の作業
  - 編集
  - コミット
  - アップデート

# 初期設定: インポート



既にファイルが存在する場合に,それを登録.



# 初期設定：インポート



- インポート
  - 既にファイルが存在する場合に，それを登録すること．

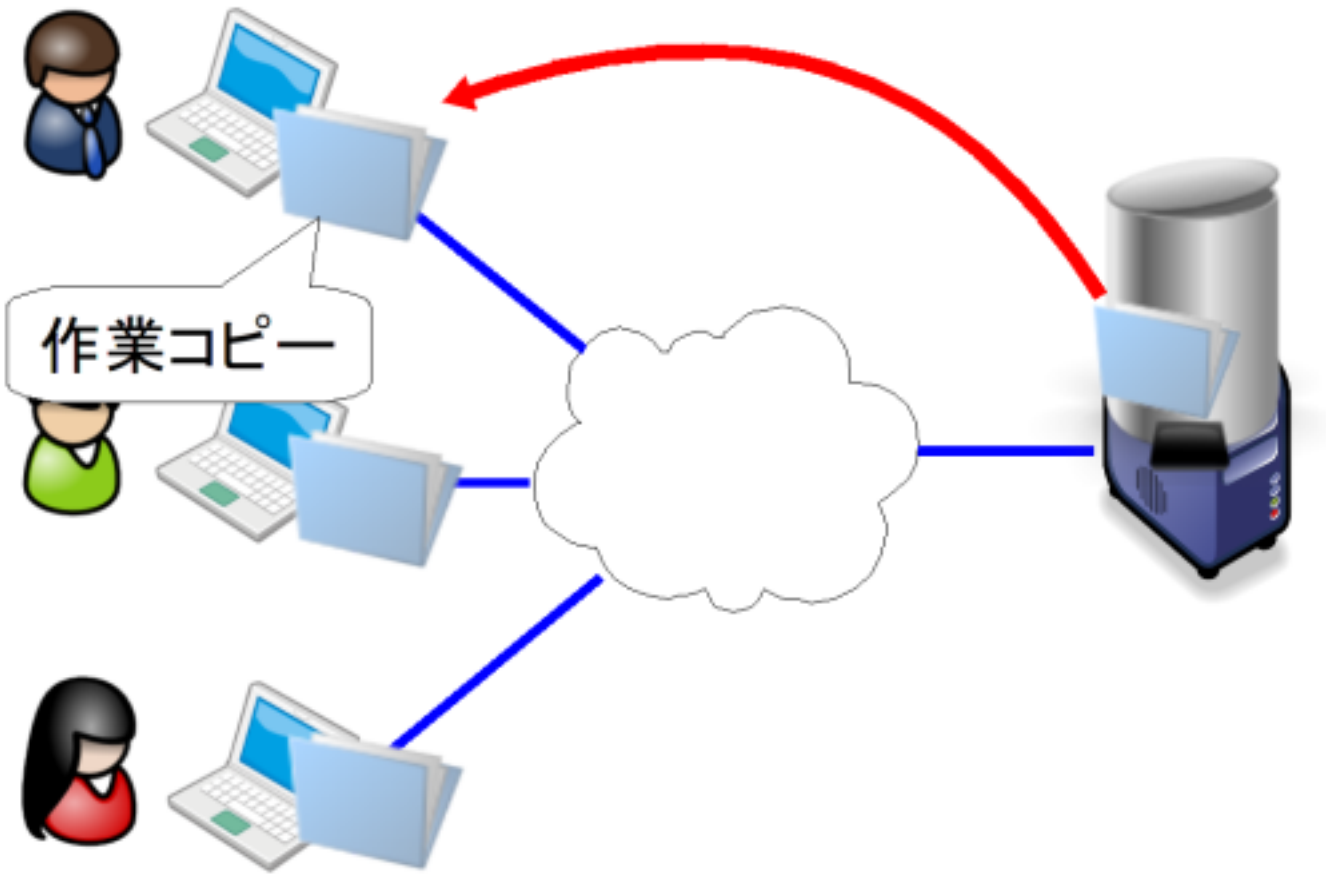
# ワークフロー



- 初期設定
  - リポジトリ作成
  - インポート
  - チェックアウト ←ココ
- 日々の作業
  - 編集
  - コミット
  - アップデート

# 初期設定: チェックアウト





# 初期設定: チェックアウト



- チェックアウト
  - リポジトリから作業コピーを作成すること
  - 作業する人，場所ごとに行います．

# ワークフロー



- 初期設定
  - リポジトリ作成
  - インポート
  - チェックアウト
- 日々の作業
  - 編集 ←ココ
  - コミット
  - アップデート

お好きに
どうぞ

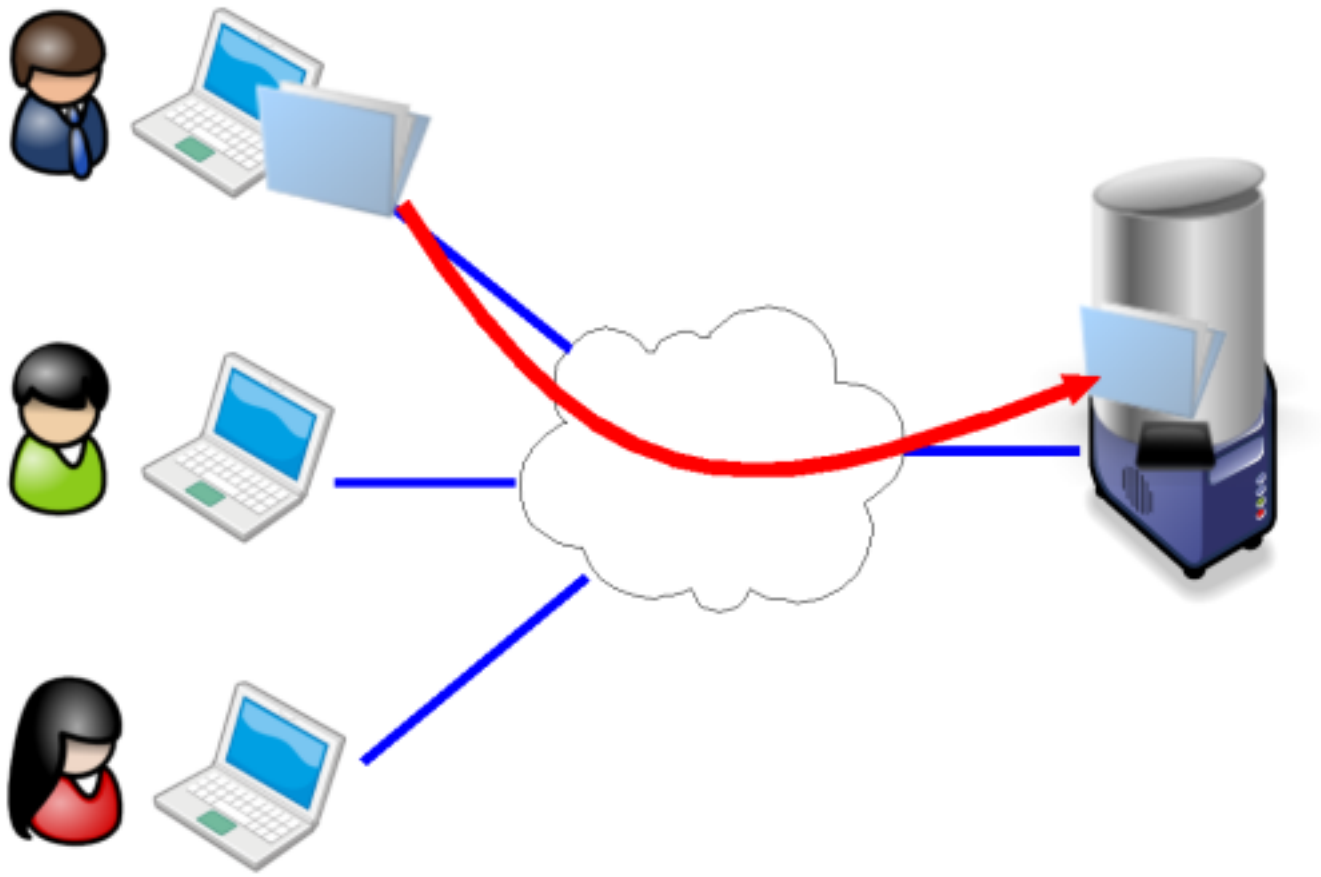
# ワークフロー



- 初期設定
  - リポジトリ作成
  - インポート
  - チェックアウト
- 日々の作業
  - 編集
  - コミット ←ココ
  - アップデート

# 日々の作業: コミット





# 日々の作業: コミット

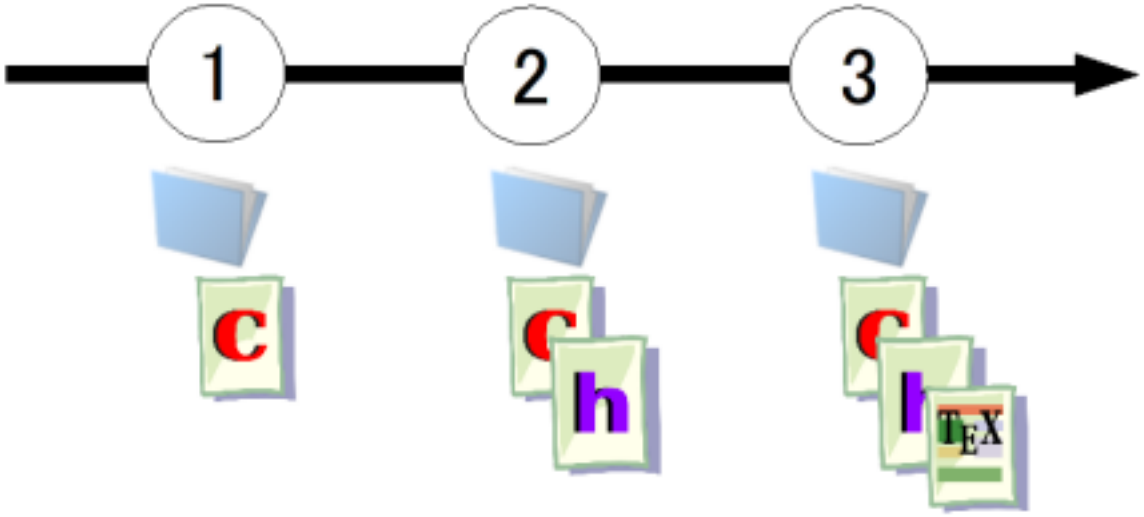


- コミット
  - 作業コピーの変更内容をリポジトリへ反映すること
    - 「チェックイン」とも言います.

# 日々の作業: コミット



- コミット毎に番号がつきます
  - 「リビジョン番号」と言います.



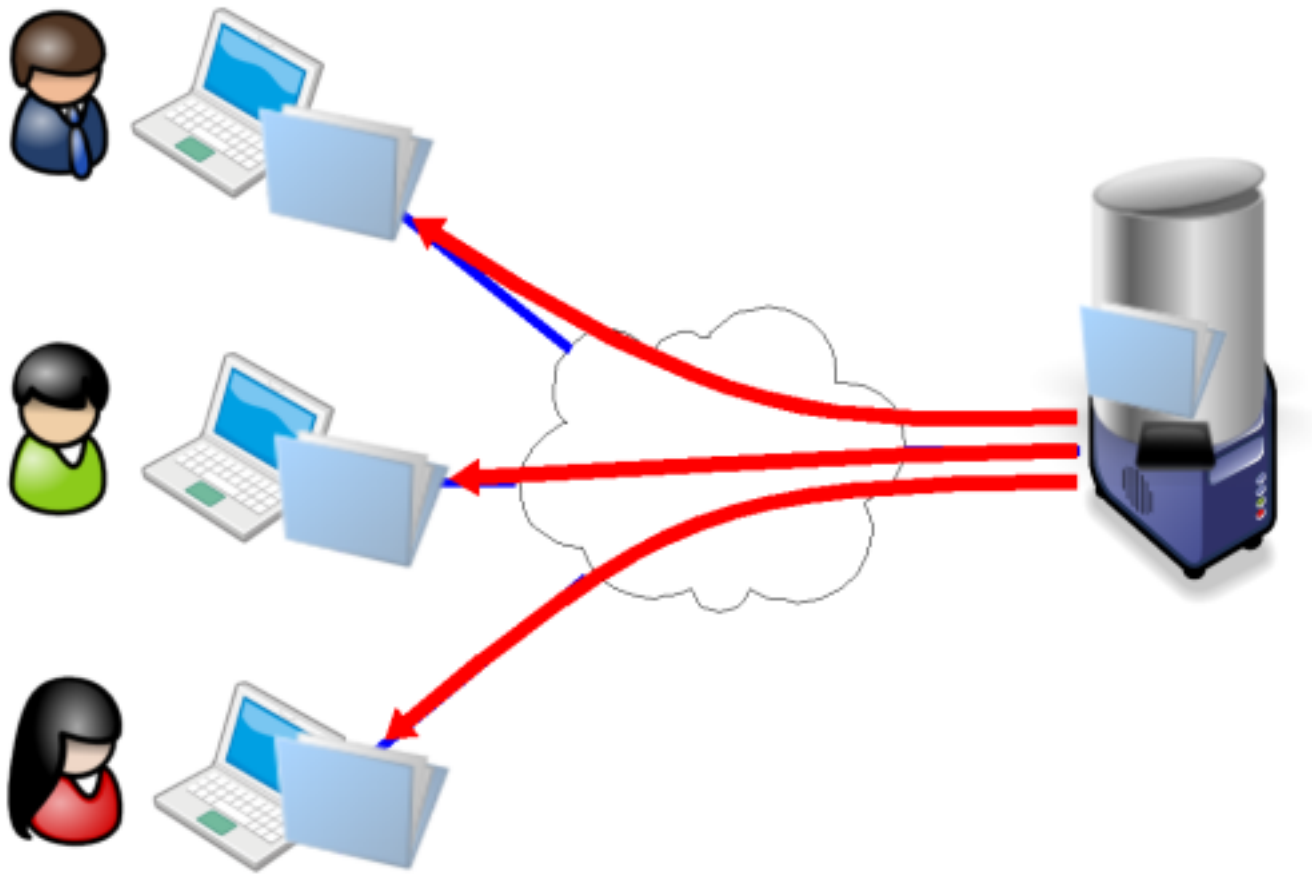
# ワークフロー



- 初期設定
  - リポジトリ作成
  - インポート
  - チェックアウト
- 日々の作業
  - 編集
  - コミット
  - アップデート ←ココ

# 日々の作業: アップデート





# 日々の作業: アップデート



- アップデート
  - リポジトリの変更内容を作業コピーに反映
  - 他人の変更も取得できる.

# 日々の作業



- 一度チェックアウトした後は
  - 編集
  - コミット
  - アップデート　....　のくりかえし

# バージョン管理をしておくと(再)



- どのリビジョンでも取得可能
  - "最新版","いつかの版"
- 変更履歴を辿れる
  - 誰が,いつ,どんな変更をしたのか
- 誰かの変更を上書きしても ok
  - いつでも元に戻せる

良い感じで日々の作業に打ち込めます.

良い感じ
に見えん

# 「良い感じ」を実感する機能

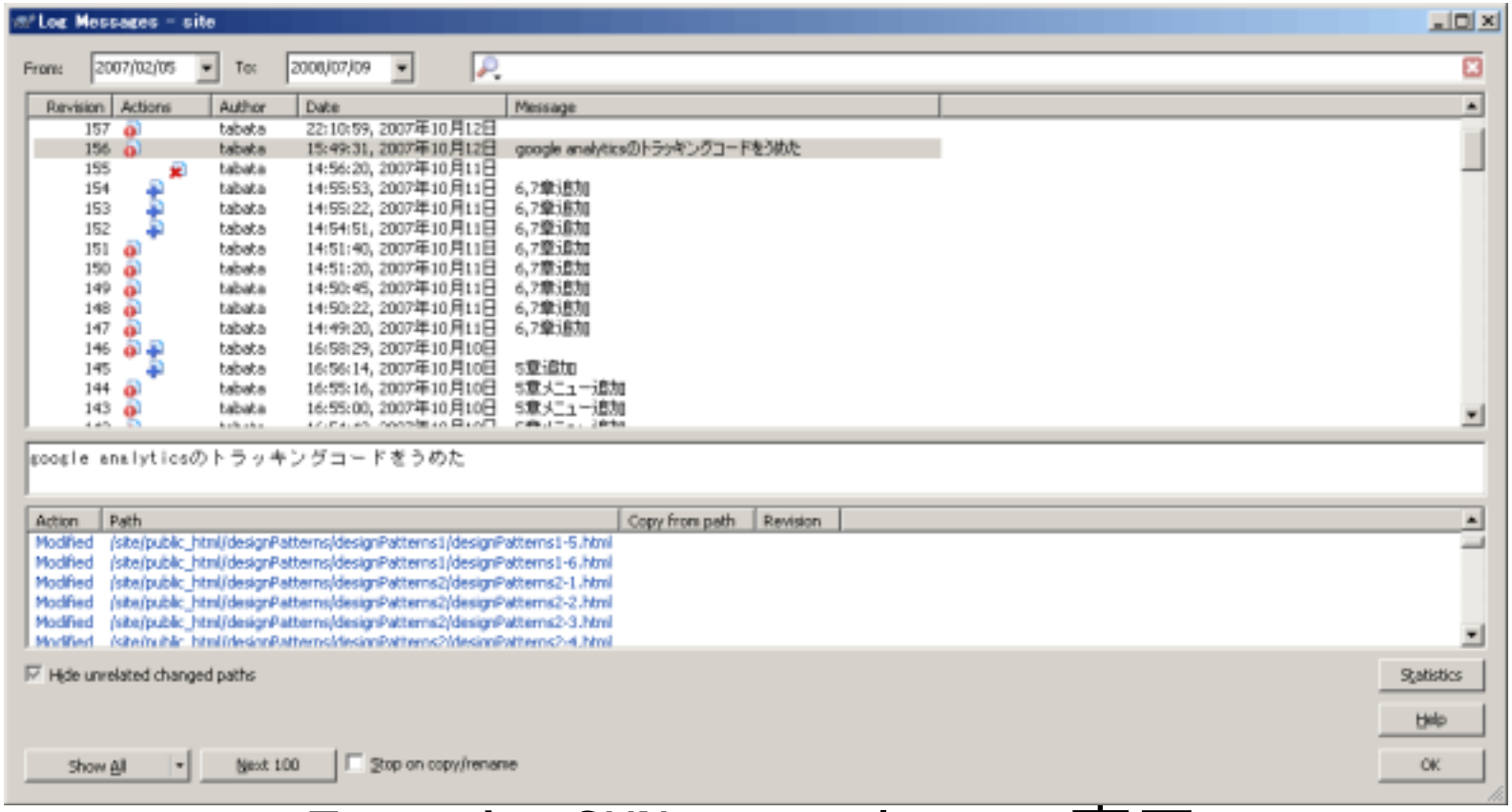


- ログ
- 差分
- 競合(conflict) と解決

# ログ



## ファイル/ディレクトリの更新履歴

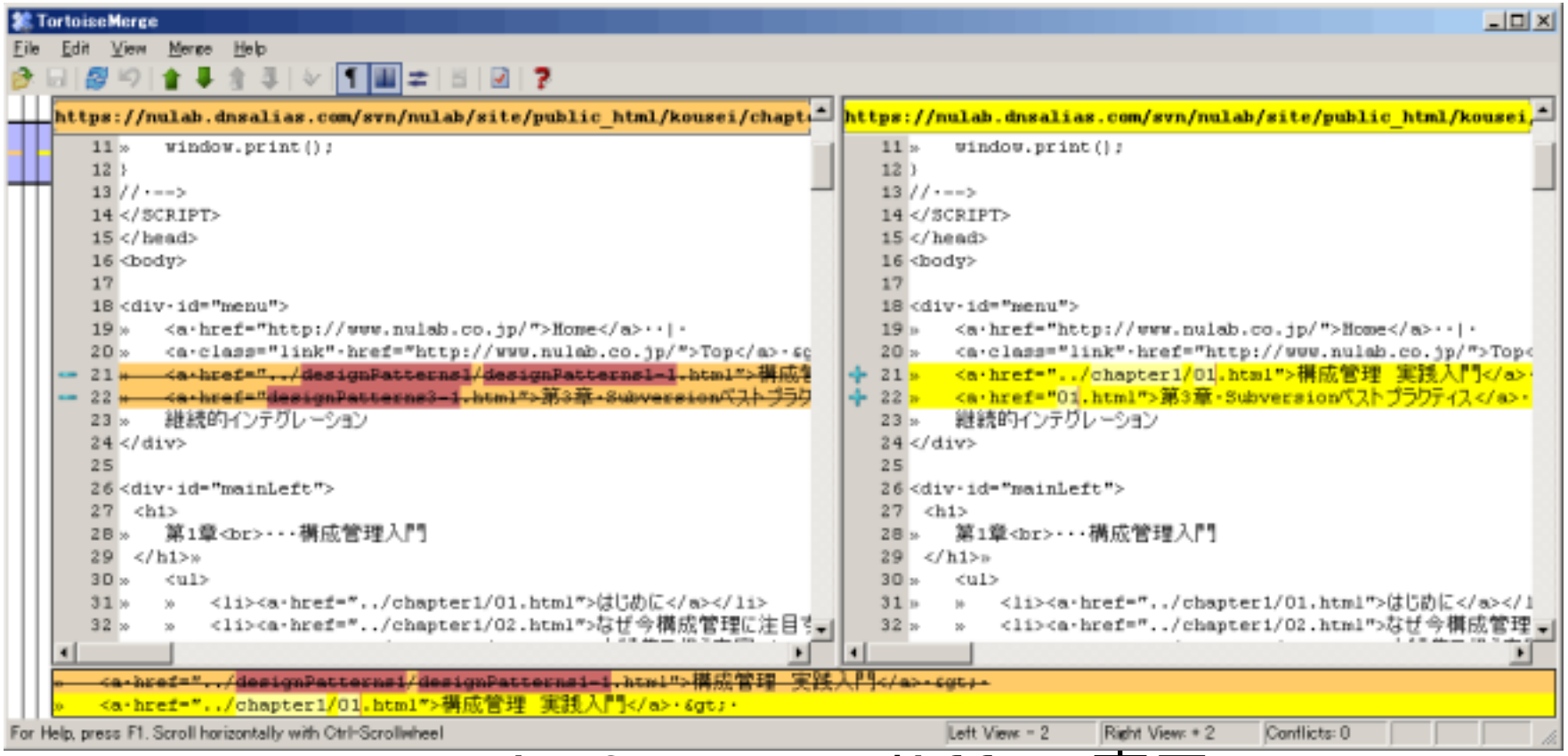



TortoiseSVN での log の表示

# 差分



## ファイル/ディレクトリの違い

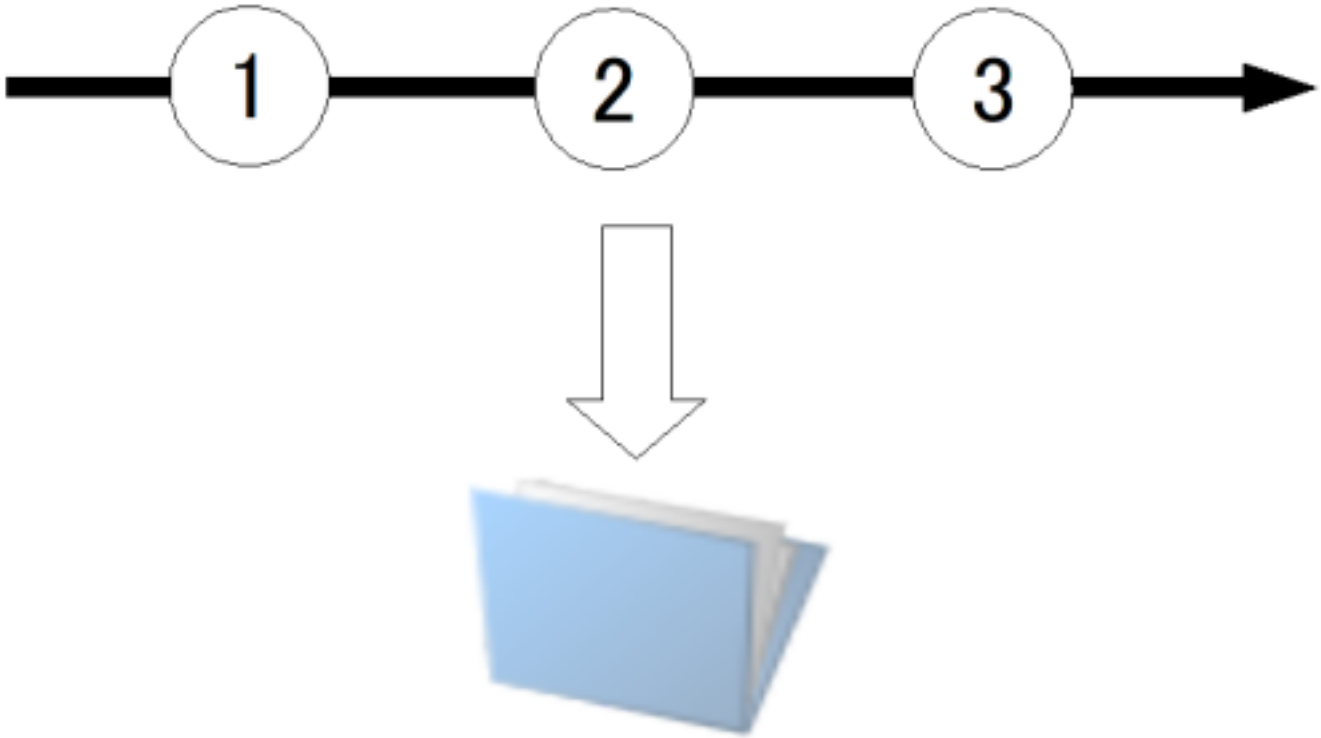



TortoiseSVN での diff の表示

# ログ + チェックアウト



リビジョンを指定してチェックアウト



# 「ログ」と「差分」

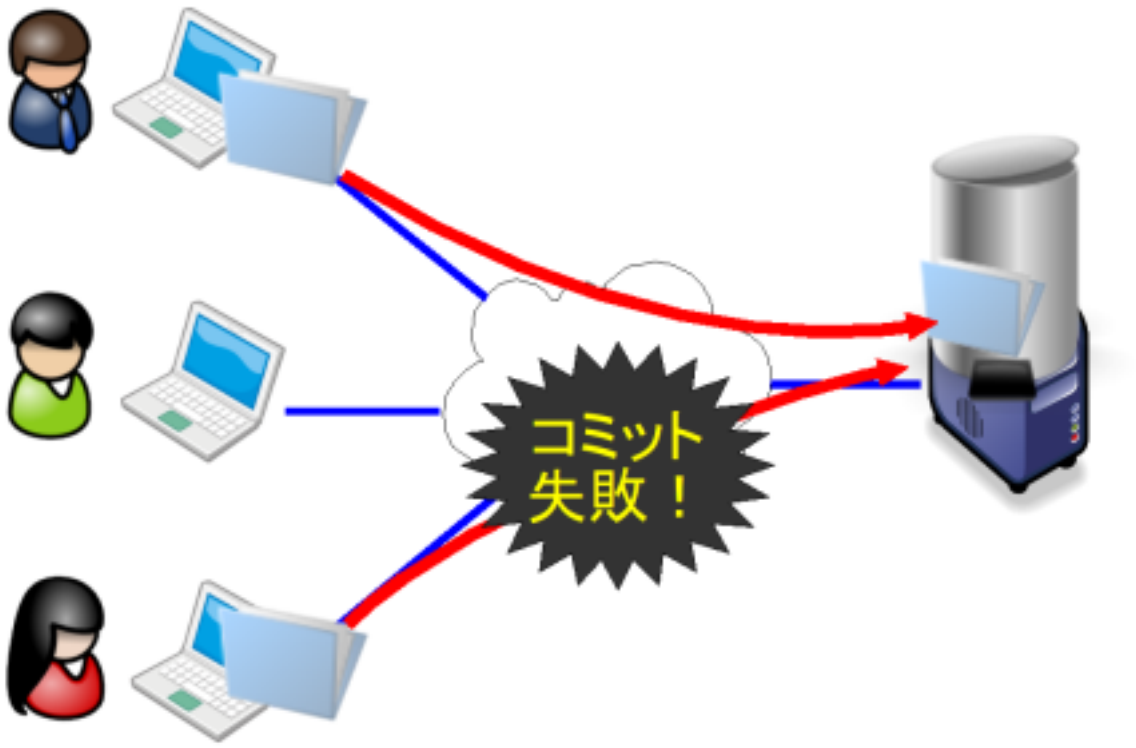


- 「最新」や「いつかの」をすぐ入手できる.
- 変更履歴を参照できる.

# 競合(conflict) と解決



誰かの更新を上書きしない



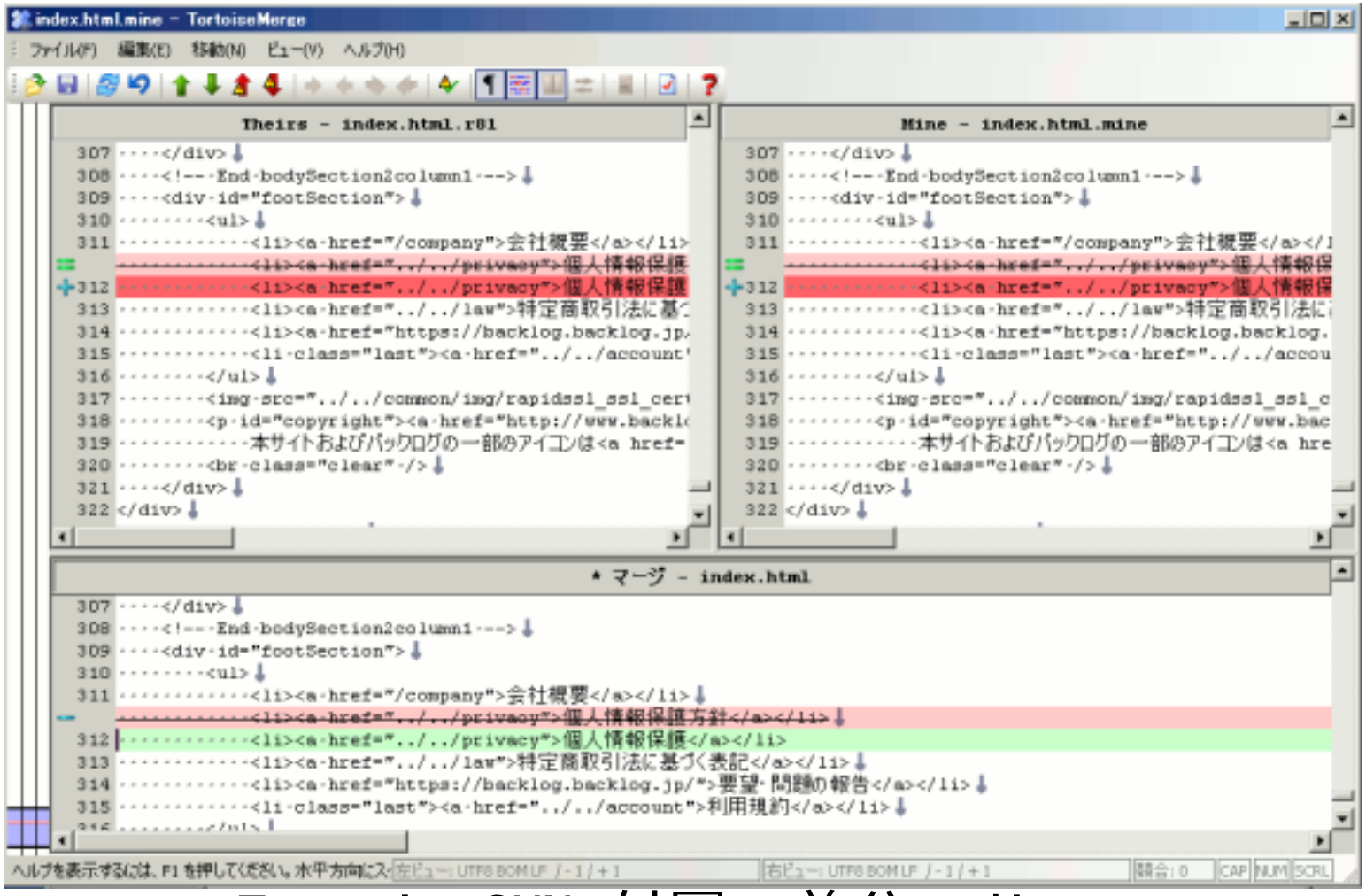
# 競合(conflict) と解決



- コミット前にアップデートしよう
  - 変更箇所が違う場合には
    - Subversion の場合は自動で解決
    - CVS はどうだったっけ?
- 変更箇所が同じ場合には
  - 差分を確認しながら修正
    - 「差分マージ用のエディタ」があったりする.

# 競合(conflict) と解決






TrotoiseSVN 付属の差分 editor

# ちょっと一息

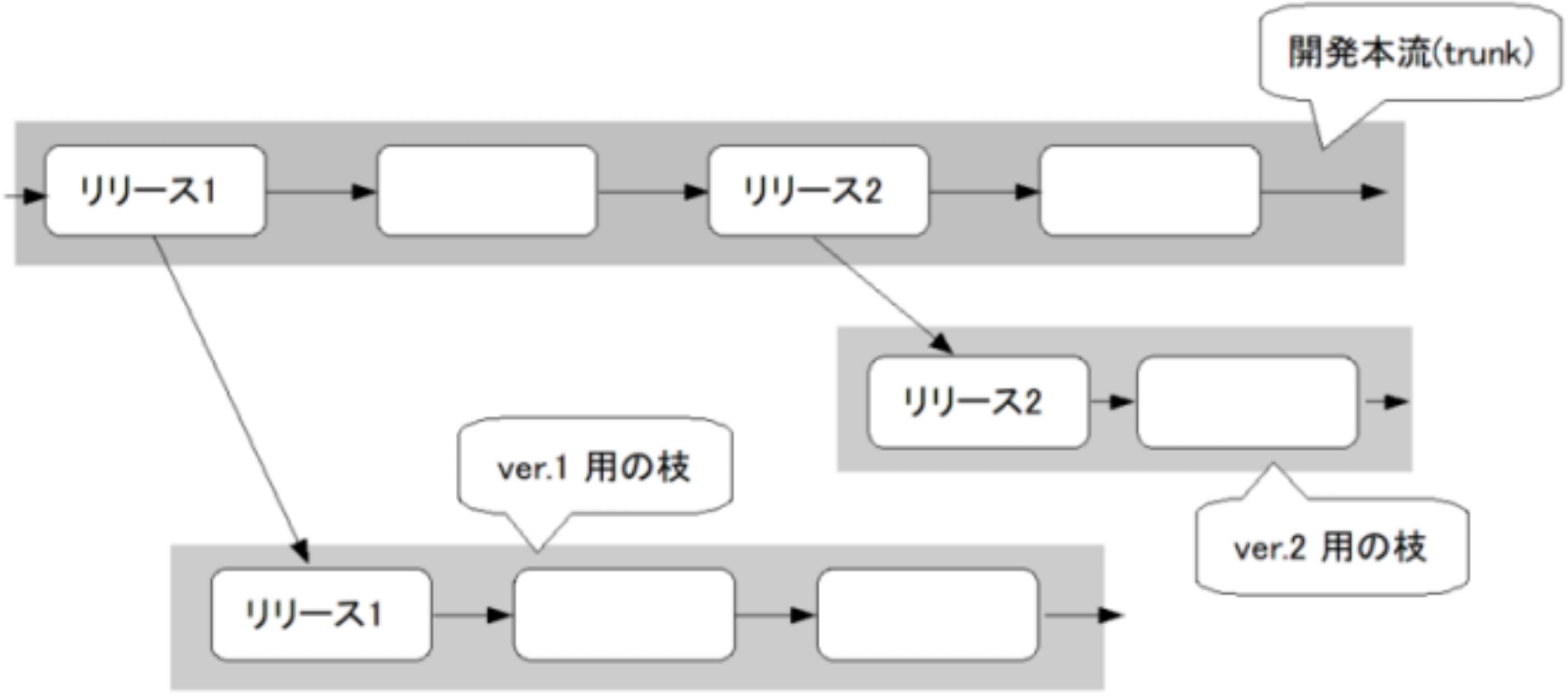


- 出てきた用語
  - リポジトリ,作業コピー
  - インポート
  - チェックアウト
  - コミット(チェックイン)
    - リビジョン番号
  - アップデート

# さらなる一歩



- 開発の流れを複数に分けることが可能
  - メインは幹(trunk),分かれた先は branch(枝)



# Agenda



- バージョン管理システム入門
  - 概要
  - 集中リポジトリ型のワークフロー
    - CVS,Subversion
  - 歴史
- 分散リポジトリ型の紹介
  - arch,Bazaar,Git,Mercurial,Monotone

簡単に

# UNIX 黎明期



- 目的: ファイルの管理
- SCCS: Soruce Code Control System
  - 1972,ベル研の Rochkind,M.J. がIBM System/370 上の OS/MVT 向けに開発.
  - Single UNIX Specification に含まれている
- RCS: Revesion Control System
  - 1985,パデュー大の Tichy,W.F. が開発.
  - データ管理はファイルベース. 直前との差分を保存

# CVS の登場



- 複数ファイルの一括管理
  - 複数のファイル -> 「モジュール」と呼ぶ.
    - 変更をモジュール単位で管理
  - 「リポジトリ」の誕生
    - 管理ディレクトリを別ディレクトリに
- ロックをしない編集
  - 他人の作業を待たなくて良い.
- ネットワーク対応

# Subversion:「より良い CVS」



- ファイル名/ディレクトリの変更に対応
  - CVS では(変更は可能だけれども)履歴が切れる
- Atomic(不可分)なコミット
  - コミット単位で管理 -> リポジトリの保護
- バイナリファイルの扱いが強化
  - バイナリ差分アルゴリズムでデータを管理
    - CVS は RCS ベース(diff で管理)

そして
分散型へ

# Agenda



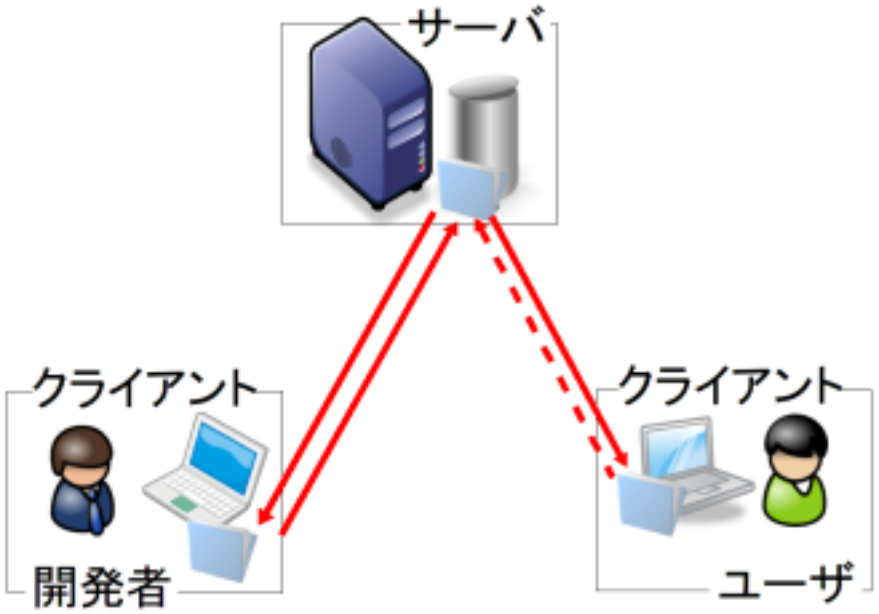
- バージョン管理システム入門
  - 概要
  - 集中リポジトリ型のワークフロー
    - CVS,Subversion
  - 歴史
- 分散リポジトリ型の紹介
  - arch,Bazaar,Git,Mercurial,Monotone

# 集中型 VCS



## ■ 集中型 VCS の構成

- 開発コアメンバーと「外部ユーザ」



# 集中型 VCS

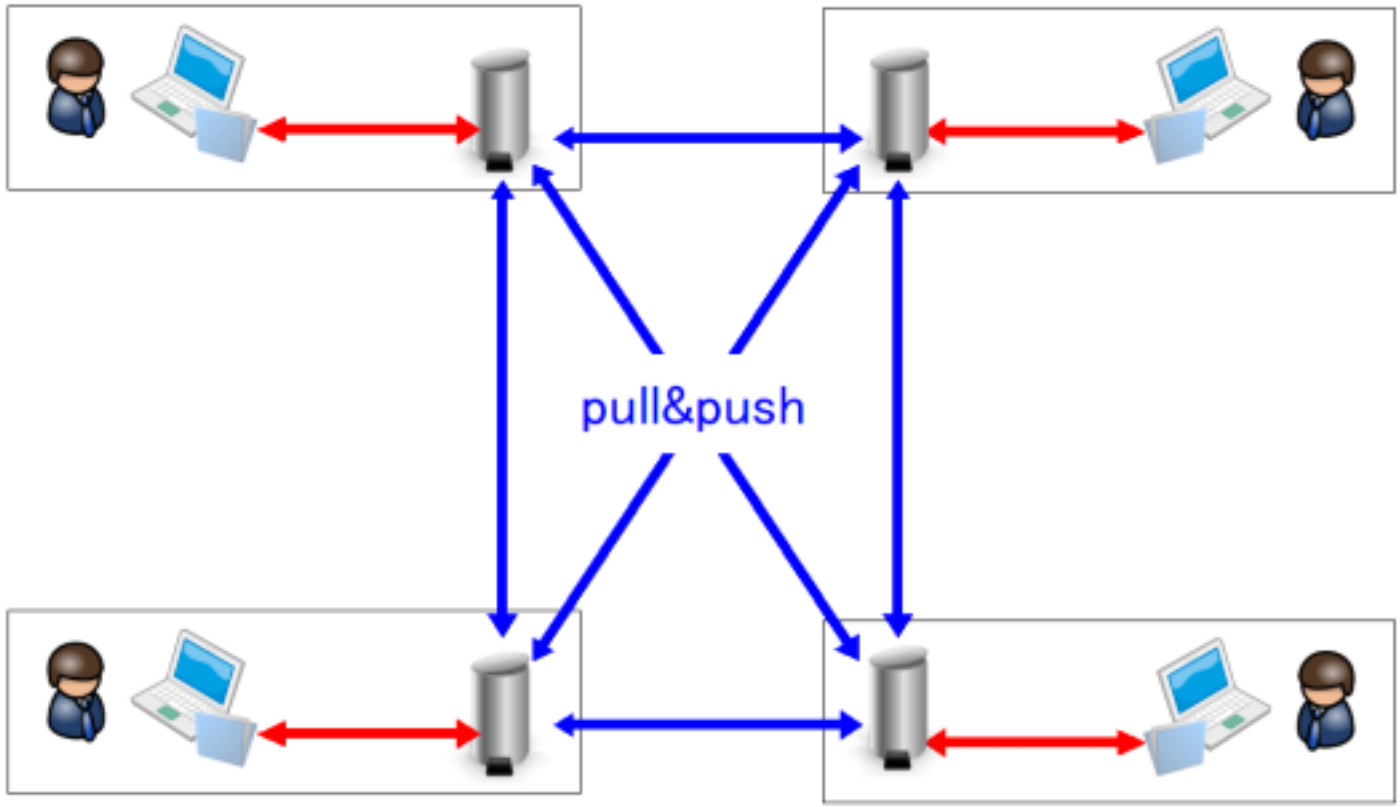


- 集中型 VCS の構成
- 開発「コアメンバー」と「外部ユーザ」
  - コアメンガーがリポジトリを管理
  - 外部ユーザは Read-Only
    - チェックアウトはできる
    - 改善 -> コアメンバーへパッチを送付
    - コアメンバーがリポジトリへコミット

# そして分散型へ



- 分散型 VCS の構成



# 分散型 VCS



- だれもが手元にリポジトリを持つ
  - 変更点のコミットには権限不要
  - だれの変更でもpullで手元に反映可能
    - 公開されていれば
  - 外部リポジトリへの反映(push)には権限が必要
    - 運用ポリシー次第で集中型 VCS としても機能

# 分散 VCS のメリット



- 手元にリポジトリがあること!!
  - ネットワークに繋っていなくても
    - 履歴が参照できる.
    - 差分が取れる
  - コミット時の衝突管理に時間が取られない.
    - 衝突管理は push 時にのみ発生

「気軽に branch が作れて消せる」状態だと思いねぇ.

# 主な分散 VCS



- 歴史的順番はあやふや
  - arch: emacs とか tla(C で書かれた arch) ですね.
  - Bazaar: MySQL とか.
  - Monotone: 良く知りません.
  - Git: linux kernel,X.Org,Rails,
  - Mercurial: Mozilla,Xen,

## Git

[名] ばか者,愚か者,あほ,間抜け,能なし,脳たりん

- 発音は「ぎっと」です.
  - どこかで「じっと」と言ったかもしれません
    - ゴメンナサイ
- 作者: Linus B. Torvalds
  - Linux Kernel の開発管理用に作成
    - 開発スタンス: patch ベース

# コマンド比較



| 機能 | Git | CVS | Subversion |
|---|---|---|---|
| 追加 | git add | cvs add | svn add |
| 削除 | git rm | cvs remove | svn rm |
| 移動 | git mv | なし | svn mv |
| 差分 | git diff | cvs diff | svn diff |
| 履歴 | git log | cvs log | svn log |
| チェックアウト | git clone | cvs checkout | svn checkout |
| コミット | git commit/ | cvs commit | svn commit |

# CVS,Subversion との連携



- リポジトリが CVS/Subversion の場合
  - C/Subversion で作業コピーをチェックアウト
  - 作業コピーを(手元では) git で管理
  - 手元での変更,コミットは git に.
  - 一段落したら CVS/Subversion リポジトリに反映

なんてことも可能です.

おしまい

# 参考文献(1)




- 石曽根 信, 2007: バージョン管理システム, UNIX MAGAZINE Classic, pp.40--51
- 岩松信洋, 小林儀匡, 上川純一, 2008:分散バージョン管理システム Git 徹底活用ガイド, Software Design 2008年 4 月号, pp.115--157
- Rochkind M.J.,1975: The Source Code Control System, IEEE Transactions on Software Engineering SE-1:4, pp. 364--370
- Tichy W.F.,1985 : RCS--A System for Version Control, Software--Practice and Experience. 15, pp. 637-654.

# 参考文献(2)




- 電脳倶楽部の CVS メモ:http://www.gfd-dennou.org/library/cc-env/cvs/
- 高木さん@東大の Subversion メモ:http://www-aos.eps.s.u-tokyo.ac.jp/~takagi/SubversionMemo.html
- 山本竜三, Subversion でバージョン管理を:http://www.slideshare.net/dragon3/subversion-presentation
- 岩松信洋, Git 入門@ rails meeting Tokyo:http://www.ustream.tv/recorded/746377