What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- Btw: "repo"="repository"

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it

- Btw: "repo"="repository"

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it
- when checking back in ("`commit`"), auto-merge differences. If conflict, merge by hand and re-check-back-in.
- Btw: "repo"="repository"

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it
- when checking back in ("`commit`"), auto-merge differences. If conflict, merge by hand and re-check-back-in.
- Btw: "repo"="repository"

Key advantages:

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it
- when checking back in (“ **`commit`**”), auto-merge differences. If conflict, merge by hand and re-check-back-in.
- Btw: “repo”=“repository”

Key advantages:

- if cloud-based, work from anywhere

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it
- when checking back in ("`commit`"), auto-merge differences. If conflict, merge by hand and re-check-back-in.
- Btw: "repo"="repository"

Key advantages:

- if cloud-based, work from anywhere
- maintain complete history

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it
- when checking back in ("`commit`"), auto-merge differences. If conflict, merge by hand and re-check-back-in.
- Btw: "repo"="repository"

Key advantages:

- if cloud-based, work from anywhere
- maintain complete history
- can roll back to any previous commit

What is Version Control?

Allow several people to edit a file.

- Emailing around *the current copy* is dismal!
- Why google-docs not good for coding?

Central Repo Model:

- check out a file to work on it
- when checking back in ("`commit`"), auto-merge differences. If conflict, merge by hand and re-check-back-in.
- Btw: "repo"="repository"

Key advantages:

- if cloud-based, work from anywhere
- maintain complete history
- can roll back to any previous commit
- easy to experiment: clone, change, then trash it!

What is *Distributed* Version Control?

Old-school: RCS, CVS, subversion

New-school: mercurial, **git**: independent repositories

What is *Distributed* Version Control?

Old-school: RCS, CVS, subversion

New-school: mercurial, **`git`**: independent repositories

Git *centralized model*:

- Clone the central repo (one-time only) to your local machine. Gives you:
  - Your working-copy of files, and
  - a local repo (inside `.git/`), with full history.
- Edit working copy, and then commit to local repo.
- When stable, **`push`** changes back to central repo.
- Frequently: **`pull`** changes from central repo.

Centralized workflow mantra

- 1. **`pull`**
- 2. edit
- 3. **`commit`**
- 4. **`pull`**
- 5. **`push`**

Do steps 2-3 often (e.g. every time it compiles sucessfully);

do steps 4-5 whenever you are passing all your tests.

Centralized workflow mantra

- 1. **`pull`** at start of work-session
- 2. edit
- 3. **`commit`**
- 4. **`pull`**
- 5. **`push`**

Do steps 2-3 often (e.g. every time it compiles sucessfully);

do steps 4-5 whenever you are passing all your tests.

Centralized workflow mantra

- 1. **`pull`** at start of work-session
- 2. edit for a short time
- 3. **`commit`**
- 4. **`pull`**
- 5. **`push`**

Do steps 2-3 often (e.g. every time it compiles sucessfully);

do steps 4-5 whenever you are passing all your tests.

Centralized workflow mantra

- 1. **`pull`** at start of work-session
- 2. edit for a short time
- 3. **`commit`** small changes
- 4. **`pull`**
- 5. **`push`**

Do steps 2-3 often (e.g. every time it compiles sucessfully);

do steps 4-5 whenever you are passing all your tests.

Centralized workflow mantra

- 1. **`pull`** at start of work-session
- 2. edit for a short time
- 3. **`commit`** small changes
- 4. **`pull`** again! -- to incorporate others' work
- 5. **`push`**

Do steps 2-3 often (e.g. every time it compiles sucessfully);

do steps 4-5 whenever you are passing all your tests.

Centralized workflow mantra

- 1. **`pull`** at start of work-session
- 2. edit for a short time
- 3. **`commit`** small changes
- 4. **`pull`** again! -- to incorporate others' work
- 5. **`push`** your work-session

Do steps 2-3 often (e.g. every time it compiles sucessfully);

do steps 4-5 whenever you are passing all your tests.

some helpful git commands

Syntax: **`git command-name options`**

Examples:

- **`git clone https://ibarland@bitbucket.org/ibarland/sample.git`**
- **`git pull`**
- **`git status`**

- **`git diff someFile`**

- **`git add othrFile`**

- **`git commit .`**
- **`git push`**

Many more commands and concepts (e.g. forking, pull requests); see tutorials and/or " The Git Book".

some helpful git commands

Syntax: **`git command-name options`**

Examples:

- **`git clone https://ibarland@bitbucket.org/ibarland/sample.git`**
- **`git pull`** no options
- **`git status`**

- **`git diff someFile`**

- **`git add othrFile`**

- **`git commit .`**
- **`git push`**

Many more commands and concepts (e.g. forking, pull requests); see tutorials and/or " The Git Book".

some helpful git commands

Syntax: **`git command-name options`**

Examples:

- **`git clone https://ibarland@bitbucket.org/ibarland/sample.git`**
- **`git pull`** no options
- **`git status`**

  shows which files have changed etc.
- **`git diff someFile`**

  show the before/after versions
- **`git add othrFile`**

- **`git commit .`**
- **`git push`**

Many more commands and concepts (e.g. forking, pull requests); see tutorials and/or " The Git Book".

some helpful git commands

Syntax: **`git command-name options`**

Examples:

- **`git clone https://ibarland@bitbucket.org/ibarland/sample.git`**
- **`git pull`** no options
- **`git status`**

  shows which files have changed etc.
- **`git diff someFile`**

  show the before/after versions
- **`git add othrFile`**

  files not part of repo unless you add them
- **`git commit .`**
- **`git push`**

Many more commands and concepts (e.g. forking, pull requests); see tutorials and/or " The Git Book".

some helpful git commands

Syntax: **`git command-name options`**

Examples:

- **`git clone https://ibarland@bitbucket.org/ibarland/sample.git`**
- **`git pull`** no options
- **`git status`**

  shows which files have changed etc.
- **`git diff someFile`**

  show the before/after versions
- **`git add othrFile`**

  files not part of repo unless you add them
- **`git commit .`** wise to commit per-dir
- **`git push`**

Many more commands and concepts (e.g. forking, pull requests); see tutorials and/or " The Git Book".

some helpful git commands

Syntax: **`git command-name options`**

Examples:

- **`git clone https://ibarland@bitbucket.org/ibarland/sample.git`**
- **`git pull`** no options
- **`git status`**

  shows which files have changed etc.
- **`git diff someFile`**

  show the before/after versions
- **`git add othrFile`**

  files not part of repo unless you add them
- **`git commit .`** wise to commit per-dir
- **`git push`** back to central repo

Many more commands and concepts (e.g. forking, pull requests); see tutorials and/or " The Git Book".