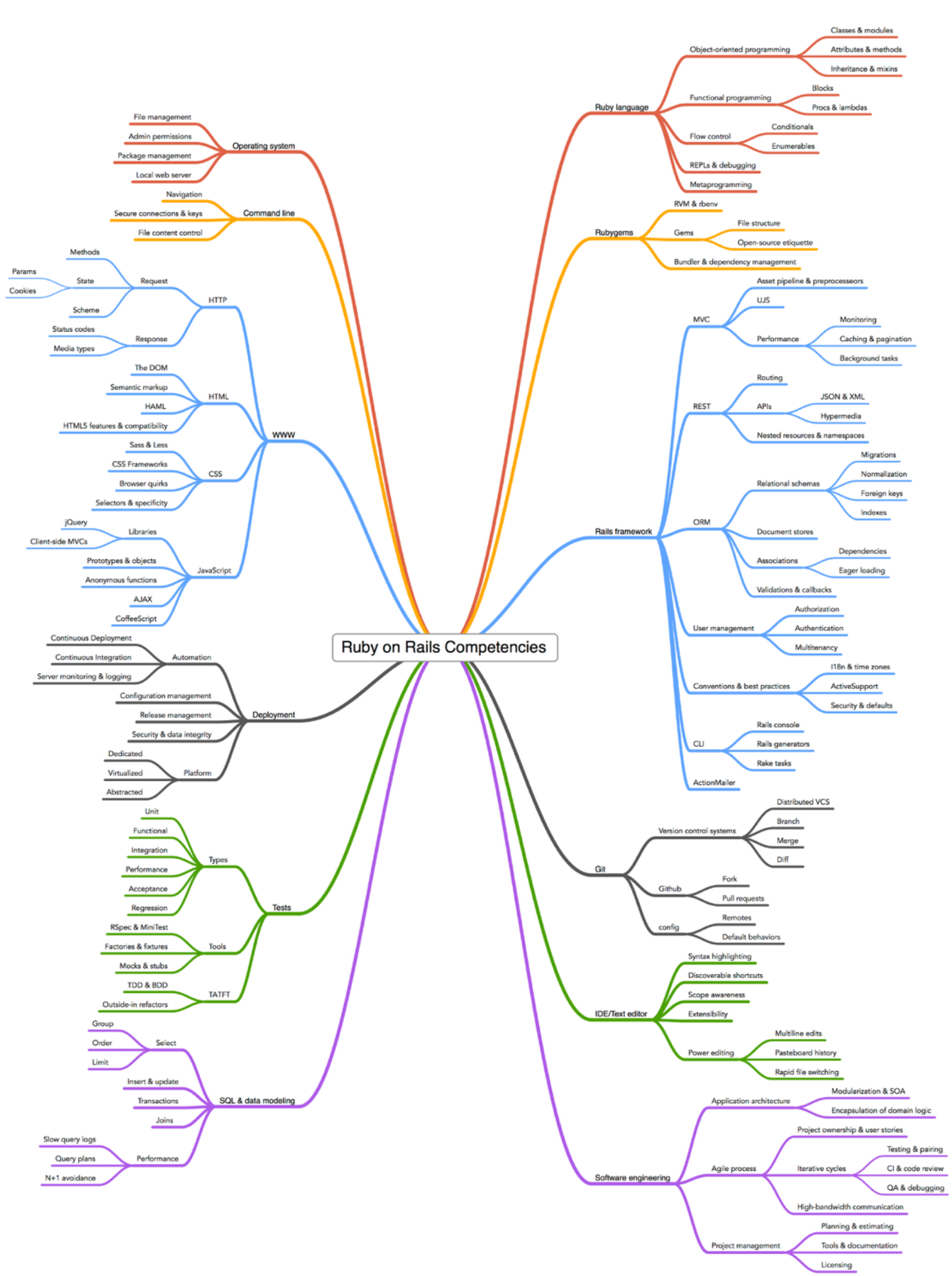
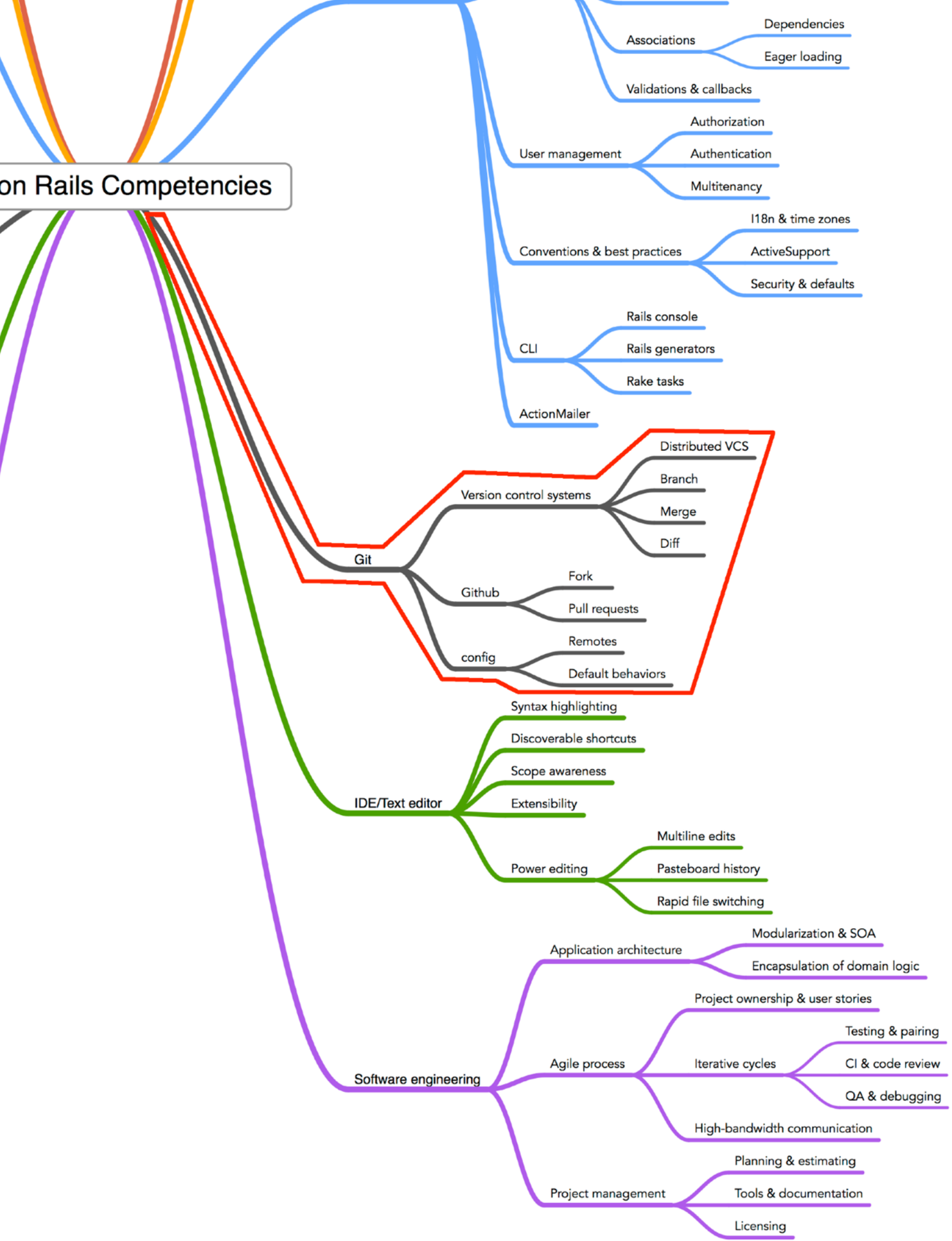
Intro to Git

History & How to Use

Ben Leadholm
b.leadholm@sportradar.com
bigbenny3@gmail.com
da twitters: @bgbn3

On the octopus..





we'll study this tentacle

Frank Herbert, the author of the Dune science fiction series, asked his author friend, Jerry Pournelle if there was a way to save specific drafts of his work, not just a chapter at a time, but different versions of the same chapter, so that he could pursue different plot lines and flip between ideas. This was back in the 1980s (from one of Jerry's articles in Byte magazine).

The technology that Frank Herbert wished he had is now available to you at the low, low price of free. It's already installed on your Mac. It is available through RailsInstaller on Windows.

Before Git

Electronic source code control has only been around since 1972 with Source Code Control System (SCCM).

Geeks being geeks, there were offshoots like Revision Control System (RCS, 1982), which could save histories and branches, but only for a single file. Groups of files would need to be 'locked' in a 'head' branch; and Concurrent Versions System (CVS, 1990) which could store multiple files into one branch and keep track of multiple branches. One would need to merge with a branch before committing changes, and it was centralized, requiring a server repository. CVS allowed developers across the globe to work on the same project through the Internet, as long as there was a central repository.

Before Git

Subversion (SVN, 2000) built on some of the weaknesses of CVS, and was a popular replacement.

Then source code control became distributed, rather than requiring a central repository, BitKeeper (2000) was a proprietary source control used by the Linux kernel team until BitMover (the company that supports BitKeeper) decided to no longer supply a free license to the Linux kernel team once one of the kernel developers reverse-engineered BitKeeper's delta tables, allowing anyone to compare their branch with multiple branches back in history.

Git Development

No longer having free access to BitKeeper, Linus Torvalds and others, developed an open-source source code control that worked in a distributed environment like BitKeeper, but have stronger safeguards against corruption (in case someone's connection to a repository cut-out while they were updating a remote repository).

Git Alternatives

There are other open source distributed source code control projects such as Mercurial (also started in 2005 because BitKeeper removed their free version), and GNU Bazaar (2007) developed as a distributed offshoot of Subversion. Since Rails is opinionated, and to stay on the happy path, we focus on Git in this slide deck.

Git Alternatives

Will there be other source code repository frameworks in the future? Of course. And given that it’s been over 10 years since Git was developed, something's due to come along. However, given that there are a number of popular source code repositories that use Git (GitHub and BitBucket come to mind), it's not out of the question that current frameworks would evolve to meet changing needs.

When to NOT Use Git

While it might be possible to use Git to keep track of binary objects, like image, sound and spreadsheet files, the content is compressed into binary formats that make it impossible for Git to differentiate between versions of the same file. Use something like JFrog's ArtiFactory or Apache Archiva.

How to Use Git

It's not enough to initialize a Git repository for your project and make an initial commit.

For best results, it's good to make a branch for new functionality, rather than update the master branch.

Commit to the feature branch every so often. Back in the day, when word processors wouldn't automatically save your draft, you had to manually hit 'save.' The longer you waited between hitting 'save,' the more risk you took if the laptop lost power or the desktop got the dreaded 'Blue Screen Of Death.'

It’s the same concept with source code control. Commit your changes when it makes sense, and you don't want to have to retype changes.

How to Use Git

Since other developers might be working on your project, and they may have committed to the master branch, it is important to merge the master branch to your branch to ensure your branch will smoothly merge your changes back to the latest master 'trunk.'

How to Use Git

```
git init
```

Once you've got a project within a directory, go to the root of the directory and type 'git init'

This creates a /.git subdirectory within the project directory. It won't be easily visible because of the leading period, but it's there.

That means if you remove the /.git directory, you've removed the Git repository of your project

How to Use Git

```
git add .
```

This will add all the files, recursively, to the project's Git repository. If you want to exclude files (such as log files, or sensitive files like /config/database.yml), you would add these to another dot file

.gitignore

How to Use Git

`git commit -am 'type a descriptive message here'`
`git commit -a` (if you don't want to give yourself a clue)

Commit all your changes to the existing branch. Each time you commit, it helps to add a description of what changes were made.

Once you've added all the files, you can commit the additions with a description 'initial commit.'

How to Use Git

`.gitignore`

Here is a sample of a .gitignore file for a Rails project

```
# Ignore bundler config.
/.bundle

# Ignore the default SQLite database.
/db/*.sqlite3
/db/*.sqlite3-journal

# Ignore all logfiles and tempfiles.
/log/*
!/log/.keep
/tmp
/.idea
```

How to Use Git

```
git status
```

What is Git seeing?

What is the status of your current project?

What files have changed?

Use ‘git status’ to satisfy your curiosity.

How to Use Git

```
git branch
```

You want to view what branches you have available.

Initially, you will have a master branch.

How to Use Git

`git checkout [branch-name]`

You want to switch from your existing branch to [branch-name]

How to Use Git

```
git checkout -b [branch-name]
```

You want to create a new branch from your existing branch. The new branch will be named [branch-name].

How to Use Git

```
git clone [name-of-remote-repository]
```

You want to perform an initial retrieval a project repository from some remote server (like GitHub or BitBucket).

How to Use Git

`git merge [name-of-local-branch]`

You want to integrate a branch into the branch you are currently pointed to. If successful, a list of files will display showing a row of plus and minus characters giving a visual of how much changed.

If unsuccessful, then there are conflicts that have to be resolved, with the file being altered to show the different sections. You have to resolve the differences in order for the changes to commit.

How to Use Git

```
git pull

git pull origin [branch-name]

git pull [repo-name] [branch-name]
```

You want to retrieve the latest changes from a remote repository already on your system. The branch will be the same as the one you already in (so if you are in a branch that you just created, this command will be unsuccessful).

How to Use Git

```
git push

git push origin [branch-name]

git push [repo-name] [branch-name]
```

You want to send your changes to the remote repository from your workstation.

Where Do We Go From Here?

As you may guess, there is more. But we've peeled the first layers of the onion.

We've dealt with individual commands issued from a command line. There are Git graphical front-ends like GitX for the Macintosh. Or SourceTree for Mac and Windows.

As with much in life, your enjoyment of the product depends on actually using it often.

Questions?

References

https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git
http://ericsink.com/vcbe/html/history_of_version_control.html
https://en.wikipedia.org/wiki/Concurrent_Versions_System
https://en.wikipedia.org/wiki/Source_Code_Control_System
https://en.wikipedia.org/wiki/Apache_Subversion
https://en.wikipedia.org/wiki/BitKeeper
https://en.wikipedia.org/wiki/Git_(software)
https://en.wikipedia.org/wiki/GNU_Bazaar
https://en.wikipedia.org/wiki/Binary_repository_manager
https://git-scm.com/download/gui/linux

https://github.com/bclead3/IntroToGitSlides.git