The fundamentals of technical writing
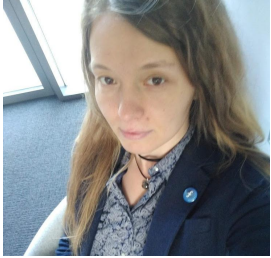
Tooling

Dominika Vagnerova
Technical Writer

Petr Kovář
Documentation Program Manager

About the authors:

Alexandra Nikandrova
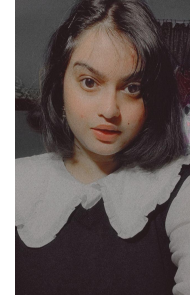
Technical Writer.

Former Devops

Ioanna Gkioka

Technical Writing Manager

Support technical writers

Former university teacher

Kalyani Desai

I am a Technical Writer, Open source contributor,

Fun fact: I tried some strawberry shampoo.

It doesn't taste as good as it smells :P

Arati Ajit Belgaonkar

I am a Technical Writer by profession and personally, I am an athlete. My hobbies are gardening and swimming.

Dominika Vagnerova

Technical Writer

Former journalist

About the authors:

Vendula Ferschmannova
Global Manager

Mayur Patil
Technical Writer 2

RHEL Networking and RDMA

Petr Kovář
Documentation Program Manager

Open source documentation writer and translator.

What we'll discuss today

- Homework follow-up
- Introduction
- Editors
- Languages
  - XML, DocBook
  - Dita toolkit
  - reStructuredText (RTS), Sphinx
  - AsciiDoc
- Docs as code, single sourcing
- Git
- Publishing tools
- Homework

Introduction

Why

- Single source → multiple outputs

- Reusing text

- Unified structure and templating
    - Usability: Structure of each article in documentation must look similar. Must have the same font, highlighting, levelling, and use blank space

How

- Languages and structure created specifically for technical writers

- Templates

- Simplified English

Editors

Editors

- Multiple editors: vi/vim,VS Code, Notepad, Pages, Emacs, etc.

- Choose whatever is comfortable

VS Code preview

13

Vim editor

There are modes: **normal** (default), **insert** and **command line**.

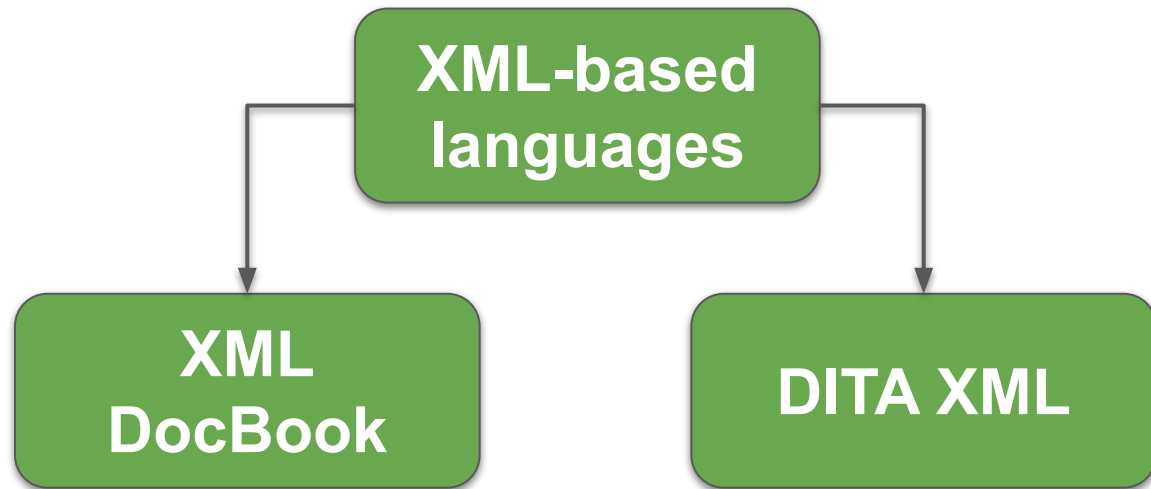| Command	Description
vim `FILE_NAME`	Create or modify the `FILE_NAME` in vim.
:q or :ZQ	Quit the file without saving. Perform in **command line** mode.
:x or :qw!	Save and quit file. Perform in **command line** mode.
dd	Delete the highlighted text or the current line. Perform in **normal** mode.
v	Highlight the text. Use *left* and *right* arrows to expand the text area. Perform in **normal** mode.
y	Copy the highlighted text or the current line. Perform in **normal** mode.
p	Paste the highlighted text or the current line. Perform in **normal** mode.

Vim tutorial: *vimtutor*

Useful links: *vim.org*

Prerequisites for the next class (14th March)

- Install AsciiDoctor on your computer (https://rh-writers.github.io/technical-writing-course-brno/#_introduction_to_asciidoctor)

- Sign up for a GitHub account.

- Set up Git and SSH keys on your computer if you work from the command line (https://rh-writers.github.io/technical-writing-course-brno/#_git_installation.

- Alternatively, use the GitHub web interface.

Languages

```
                    ┌──────────────────┐
                    │    XML-based     │
              ┌─────│    languages     │─────┐
              │     └──────────────────┘     │
              ▼                              ▼
    ┌──────────────────┐           ┌──────────────────┐
    │       XML        │           │                  │
    │     DocBook      │           │     DITA XML      │
    └──────────────────┘           └──────────────────┘
```

- Robust Content Management Systems
- Multi-language support
- Single source
- Modular content management
- Commercial/corporate world

- Easy adoption by community members
- Single source
- Modular content management
- Natural connection with GitHub
- Popular in open source world

Low-syntax markups

Markdown

reStructured Text (RST)

AsciiDoc
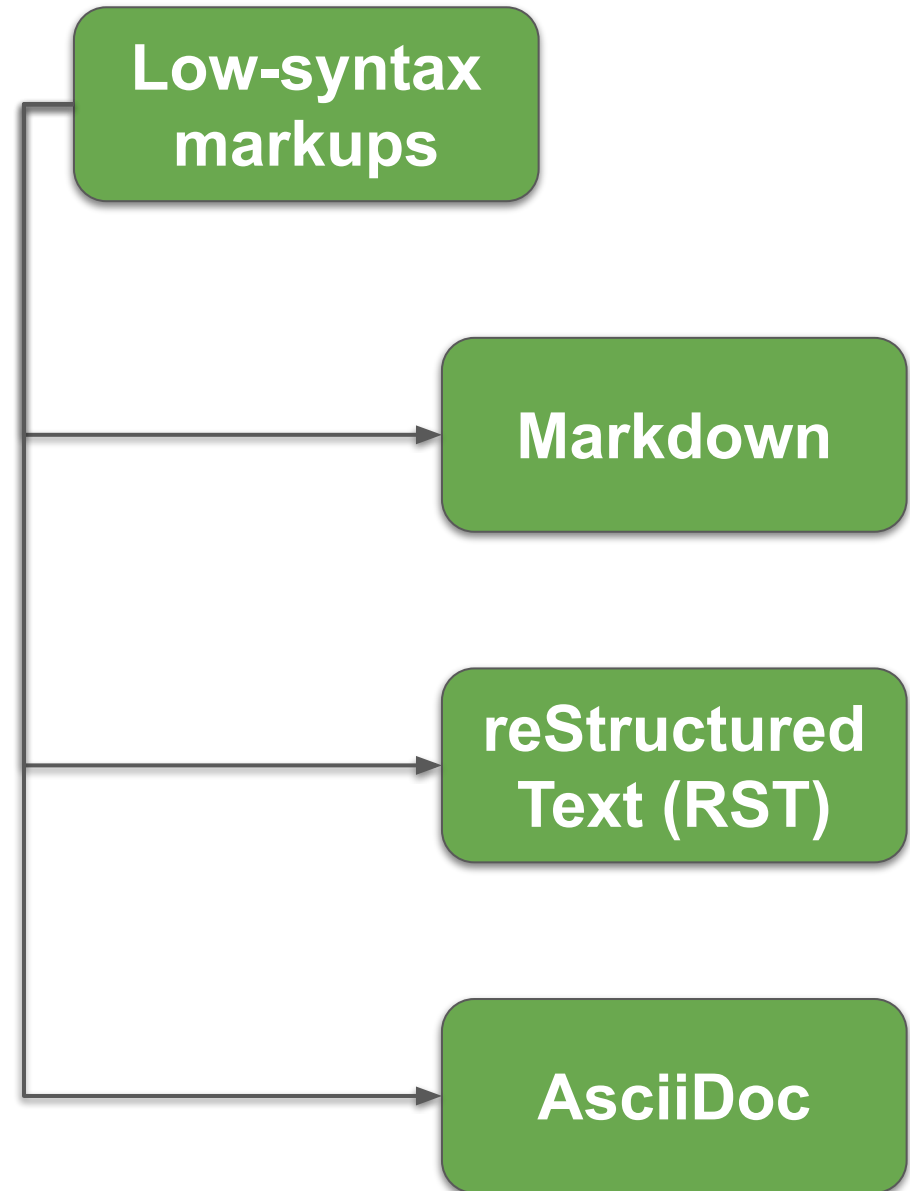
XML

Example

```
<section>

  <title>Extensible Markup
  Language</title>

  <para>Extensible Markup Language
  (XML) is a markup language that marks
  up data content with tags. XML uses
  tags to define content
  structure.</para>

</section>
```

- Extensible Markup Language (XML) is a markup language is a markup language and file format for storing, transmitting, and reconstructing arbitrary data (Wikipedia)
- Define data with pair of tags
  - Human-readable
  - Machine-readable
- Each XML document needs to contain XML header (XML declaration):
  `<?xml version="1.0" encoding="UTF-8"?>`
- Nesting

19

DocBook

Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<book xml:id="simple_book" xmlns="http://docbook.org/ns/docbook" version="5.0">
  <title>Very simple book</title>
  <chapter xml:id="chapter_1">
    <title>Chapter 1</title>
    <para>Hello world!</para>
    <para>I hope that your day is proceeding <emphasis>splendidly</emphasis>!</para>
  </chapter>
  <chapter xml:id="chapter_2">
    <title>Chapter 2</title>
    <para>Hello again, world!</para>
  </chapter>
</book>
```

- Semantic markup language created just for the purpose of technical writing
- XML uses tags to define content structure
- It consists of:
  - Elements - the basic building blocks
  - Attributes - information related to a specific element
  - Entities - representing an item of data instead of using the data itself
- Advantages: self-describing, widely used, extensibility
- Disadvantages: more lengthy, less readable compared to other markup languages, strict syntax rules
- Learn more:

  https://tdg.docbook.org/tdg/4.5/docbook.html

20

DocBook

Example

- Single source, multiple outputs (HTML, PDF, CHM)
- Templates
- Tooling: Oxygen, xMetaL, etc.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<book xml:id="simple_book" xmlns="http://docbook.org/ns/docbook" version="5.0">
  <title>Very simple book</title>
  <chapter xml:id="chapter_1">
    <title>Chapter 1</title>
    <para>Hello world!</para>
    <para>I hope that your day is proceeding <emphasis>splendidly</emphasis>!</para>
  </chapter>
  <chapter xml:id="chapter_2">
    <title>Chapter 2</title>
    <para>Hello again, world!</para>
  </chapter>
</book>
```

Darwin Information Typing Infrastructure (DITA)

- DITA is an XML-based architecture for creating modular content that allows you to reuse content and interchange content from various sources: topic-based authoring
- Advantages:
  - CMS support
  - WYSIWYG
  - Compounding guides based on versions
- DITA Open Toolkit – the open-source publishing engine: https://github.com/dita-ot/dita-ot/

Example

```
<topic xml:lang="en"
id="sample">
  <title>Sample
  title</title>
  <body>
  <p audience="teacher">
  This text is for the
  teacher.</p>
  <p audience="student">
  This text is for the
  student.</p>
  </body>
</topic>
```

DocBook vs. DITA

Language	DocBook	DITA	Markdown	RST	AsciiDoc
Single Source	x	x	x	x	x
Modular content		x		x	x
Reusing content	x	x		x	x
CMS	x	x		x	x
Proprietary / open source	x	x	x	x	x

Markdown

Example

```
Heading
=======

# Alternative heading

Sub-heading
-----------

Block of text with _italic_, **bold**
and `monospace` formatting. This is a
[link](http://example.com).

 1. numbered list
    * bulleted list
    * another bulleted list
 2. another list item

![Image](some-picture.png "picture")
```

- *"Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML)."* John Gruber

- Simplicity: plain text with Markdown syntax but very basic options (e.g. support for tables, modularity)
- Created in 2004, it became the first popular lightweight markup language, especially for blogging, online forums and hosting platforms like GitHub
- Many markup flavors, limited success in standardization
- Learn more: https://daringfireball.net/projects/markdown/

reStructuredText (RST) + Sphinx

Example

```
Chapter 1 Title
===============

.. toctree::
   :maxdepth: 2

   other_documents/included_document

This is a paragraph.

Section 1.1 Title
-----------------

   This is an indented block of text.

::

    This is a block of preformatted text.

* a bullet point

  - a sub-list item

* another bullet point

.. image:: image_folder/included_image.png
```

- reStructuredText (RST) is a lightweight markup language popular in Python-based communities
- Sphinx generates documentation from RST sources using the Docutils Python tools
- Sphinx supports cross-references and including documents in a hierarchy with automated linking, or generating indices
- Supports multiple output formats including HTML, PDF, man pages, etc.
- Can build whole documentation sites, including translations and custom themes
- Supported on GitHub and used by documentation hosting platforms such as https://readthedocs.org/

AsciiDoc vs. Asciidoctor

AsciiDoc is the language

- A plain text markup language
- A text document format with a file extension of `.adoc`
- Articles, Books, Man pages and other document types
- Rich features like use of attributes, conditional text, snippets

Asciidoctor is the processor

- Generates output in HTML, PDF, EPUB, DocBook, and man page
- Open source
- Written in the Ruby programming language

- AsciiDoc Syntax Quick Reference
- AsciiDoc Recommended Practices

AsciiDoc file

AsciiDoc features

- Built-in and user-defined attributes

- Includes

- Conditional text

```
= Adding multiple credentials with
authentication secrets

[role="_abstract"]
Add multiple credentials to {brandname} clusters
with authentication secrets.

.Prerequisites

ifdef::community[]
* Have a `kubectl` or `oc` client.
endif::community[]
ifdef::downstream[]
* Have an `oc` client.
endif::downstream[]

.Procedure

. Create an `identities-batch` file that
contains the commands to add your credentials.
+
[source,yaml,options="nowrap",subs=attributes+]
----
include::yaml/identities_batch.yaml[]
----
```

AsciiDoc includes

- Include directive imports content from a separate file

- AsciiDoc or any other text format

- Useful for reusing content, better organization, and maintenance of the files

AsciiDoc attributes

AsciiDoc source file

```
[NOTE]
====
This procedure shows you how to uninstall a {brandname} deployment on the command line but you
can use the {osweb} instead.
Refer to the {openshiftshort} documentation for specific instructions.
====
```

Attributes file

```
:brandname: Data Grid
:osweb: OpenShift Web Console
:openshiftshort: OpenShift
```

Rendered HTML

> **🖈 Note**
>
> This procedure shows you how to uninstall a Data Grid deployment on the command line but you can use the OpenShift Web Console instead. Refer to the OpenShift documentation for specific instructions.

```
[id='default-jgroups-stacks_{context}']
= Default JGroups stacks

{brandname} provides default JGroups stack files, `default-jgroups-*.xml`, in the `default-configs` directory
inside the `infinispan-core-{ispn_version}.jar` file.

ifdef::remote_caches[]
You can find this JAR file in the `{server_home}/lib` directory.
endif::remote_caches[]

[%header,cols="2,1,3"]
|===
|File name
|Stack name
|Description
```

Data Grid Server Guide

Embedding Data Grid in Java Applications

11.1. Default JGroups stacks

Data Grid provides default JGroups stack files, `default-jgroups-*.xml`, in the `default-configs` directory inside the `infinispan-core-14.0.2.Final-redhat-00001.jar` file.

You can find this JAR file in the `$RHDG_HOME/lib` directory.

File name	Stack name	Description

5.1. Default JGroups stacks

Data Grid provides default JGroups stack files, `default-jgroups-*.xml`, in the `default-configs` directory inside the `infinispan-core-14.0.2.Final-redhat-00001.jar` file.

File name	Stack name	Description

Open Source project, Docs as code, and Single sourcing

Infinispan & Data Grid & Single-sourcing

Infinispan/Data Grid is a high-performance, distributed in-memory data store

Infinispan (https://github.com/infinispan) is a high-performance, distributed in-memory data store. Infinispan is a 100% open source project, under the Apache Public License 2.0.

Data Grid is built from the Infinispan open-source software project and is available to deploy as an embedded library, as a standalone server, or as a containerized application on Red Hat OpenShift Container Platform.

Single-sourcing

Write once, publish everywhere

- Create multiple output formats (HTML, PDF, EPUB, and more)
- Create multiple deliverables by reusing the same content (getting started guide, user manuals, and other publications)
- Reuse content in multiple versions
- Reuse content in multiple products

Docs as code

Docs as code is an approach to writing and publishing documentation with the same tools and processes developers use to create code.
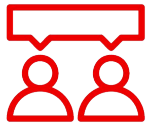
Documentation in the same repository as the source code

Docs developed alongside code at the same pace

Subject Matter Experts use the same or similar workflow to review and maintain documentation as

they do for the code

Reference content part of the code base (Javadoc, schema documentation, contextual help etc.)

Documentation source files spread across multiple repositories

Single sourcing

Write once, publish everywhere

Infinispan uses GitHub actions to synchronize documentation to the project site.

Data Grid uses Jenkins to synchronize content from the Infinispan repositories to GitLab

Multiple source repositories

https://github.com/infinispan/infinispan

https://github.com/infinispan/infinispan-operator

https://github.com/infinispan/infinispan-helm-charts

https://github.com/infinispan/dotnet-client

https://github.com/infinispan/cpp-client

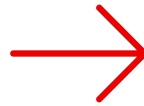https://github.com/infinispan/infinispan-simple-tutorials

Target repository "read-only"

jdg-bot @jdg-bot
-o- Pushed to branch 8.1.x
94d1d677 · Synchronized revision c19a0a8679bee17bffa1d2c7686a8ee68df42dbd

jdg-bot @jdg-bot
-o- Pushed to branch 8.4.x
b5afc3ae · Synchronized revision 5c133d0d8873d2663135494126c5319e3914dddb

Git

Agenda

- Git concept
- Repositories
- Branches
- Git workflow and basic commands

Git Concept

Git is an asynchronous distributed version control system.

- Multiple people can work on the same source both online and offline
- No central repository
- Keeps change history of the files

Image from https://git-scm.com/

Basic Terminology

Repository

The *.git* directory inside the project. Tracks changes and history of a project.

Local repository

Repository stored locally on your computer

Remote repository

Repository stored on a server

Commit

A set of changes you applied at the same time

Hash

A unique identifier of a commit

Project

A set of data, such as repositories, MRs, wiki, etc.

Basic Terminology

Branch

An independent line of development in your project

Clone

Make a copy of a remote repository on your computer

Fork

A copy of the project for any purpose

Pull

Copy data from your remote repository and merge it to the local one

Push

Copy data from your local repository to the remote one

Merge

Join two or more development histories together

Rebase

Move all changes of your current branch atop of another

Repositories

Repositories

Local

- Located on your computer
- You work locally

Remote

- Located on a server (GitHub, GitLab, etc.)
- More than one can be linked to your local repository
- You must synchronize your local repository with the remote one

Local Repository

- A working directory on your computer

- In most cases, you **clone** an existing remote repository to create the local one

- It stores Git metadata and configuration (in the *.git* subdirectory)

    - Contains information about all commits and branches

- You do changes in working directory and the history is tracked in the *.git* directory locally
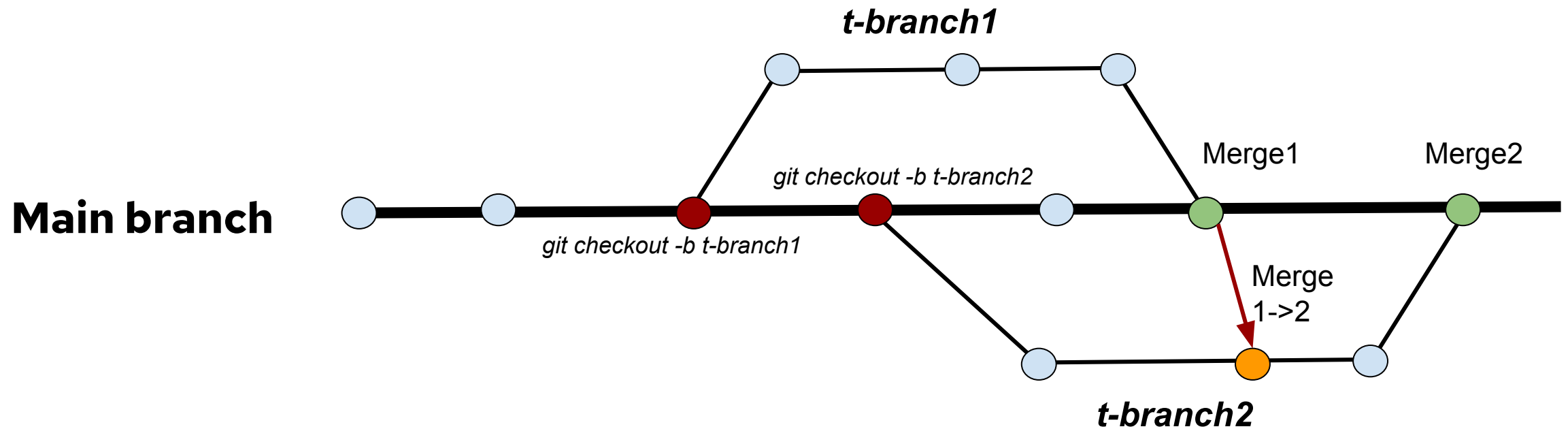
Remote Repository

- Stored on a server that all team members can be synchronized with

- Multiple remote repositories can be linked to your local repository

  - *e.g.* a product repository and your personal fork of a remote repository

- You must synchronize your local changes with a remote repository (fetch/rebase and push)

Branches

Branches

- An independent line of commits in chronological order to the project

- "Alternate history"

- **Topic branch** or **feature branch** is a lightweight branch for a specific purpose (e.g. new feature or bug fix) which could take some time

Branch workflow

Git Workflow

Cloning a Repository

```
$ git clone REPOSITORY_URL
```

- Cloning a repository makes a local copy on your computer

- One time action (per each machine you are working on)

- A remote repository named **origin** is automatically configured

- In most cases, clone the ssh URL from GitHub or GitLab (starts with **git@github.com** or

  git@gitlab)

Listing Branches

```
$ git branch
```

● The current branch is highlighted with an asterisk

Creating a New Branch

```
$ git checkout -b BRANCH_NAME
```

- With this command you create a new branch locally and automatically switch to it
- A new branch will not be created automatically at the remote repository
- A new branch is based on the branch you are switching from

Switching to Another Branch

```
$ git checkout BRANCH_NAME
```

- Git replaces the repository contents with the version of the branch you are switching to

- If the branch does not exist, this command creates a new branch, which is equivalent to:
  - `$ git checkout -b BRANCH_NAME --track REMOTE/BRANCH_NAME`

Showing Current Status

```
$ git status
```

- This command shows the following:

    - The current branch

    - Ready for a commit files

    - Modified files

    - Not tracked by Git files

Updating a Branch From Remote Repository

```
$ git fetch REMOTE_REPOSITORY
$ git rebase
```

- *git fetch* command downloads the status about remote repository:

  - Commits, files, and other metadata

  - Replace *REMOTE_REPOSITORY* with *--all* to get update from all remote repositories that are linked to a local repository

  - Does not apply any changes to your local repository

- The rebase command puts changes in your local branch atop the corresponding remote branch

Preparing for Commit

```
$ git add FILE_1 FILE_2 FILE_N
```

- You can select what changes you want to commit

- If you want to select all the changes you have done, use:
  - `$ git add .`

Committing Changes

```
$ git commit -m "changes description"
```

- Committing creates a revision in the Git history with the changes you have added before

- Every commit has its unique identifier: **hash** (a-z, 0-9)

Pushing Commits

```
$ git push REPOSITORY_NAME BRANCH_NAME
```

- Every commit on the current branch will be uploaded to the corresponding branch in the remote repository
- You must specify remote repository and branch name if you have current branch locally only

Synchronization Flow
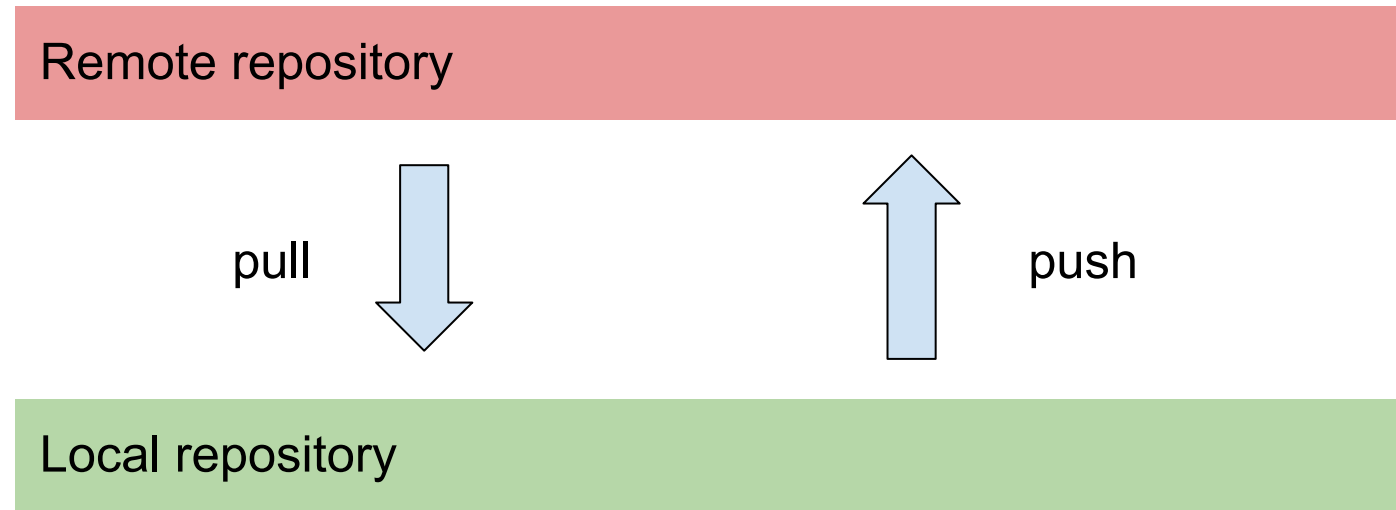
Take-away

- Git is an asynchronous distributed version control system

- There are local and remote repositories

- You clone a remote repository usually once and later synchronize with *git pull* and *git push* commands

- Usually you have a main branch and create a topic branch from it

- You work in your topic branch

- Tools

Publishing tools

Publishing tools platforms

- Introduction

- Purpose

- Types

Publishing toolchains

- Content management systems: Drupal, WordPress, etc.

- Static site generators: Jekyll, Hugo, etc.

  - Publishing platforms for the generators: GitHub Pages, GitLab Pages, etc.

- GitHub Pages

  - Built and publish docs as code from a GitHub repository

    - By default, uses Jekyll to build Markdown source files

    - Other markup languages can be built and published using the GitHub Actions CI

    - Learn more: https://pages.github.com/

Example of the publishing toolchain

1. Edit, commit and submit AsciiDoc files as a Pull Request (PR) against the

   https://github.com/rh-writers/technical-writing-course-brno repo

2. Merging the PR to the main branch triggers GitHub Actions to:

   a. Build the merged files with Asciidoctor

   b. Publish (deploy) them to the GitHub Pages site at

      https://rh-writers.github.io/technical-writing-course-brno/

3. List of triggered build and publish jobs:

   https://github.com/rh-writers/technical-writing-course-brno/actions

Linters: Vale

- Style-checker (linter)
- Supports multiple markup languages
- Command line, browser, IDE or CI jobs
- Custom rules, e.g.: Vale at Red Hat
  (https://github.com/redhat-documentation/vale-at-red-hat)
- Learn more: https://vale.sh/

```
[dvagnero@toolbox topics]$ vale proc_configuring_pom_xml.adoc

proc configuring pom xml.adoc
2:3      warning      Use correct American English    RedHat.Spelling
                      spelling. Did you really mean
                      'Coguring'?
2:25     suggestion   Define acronyms and             RedHat.Definitions
                      abbreviations (such as 'POM')
                      on first occurrence if they're
                      likely to be unfamiliar.
```

ChatGPT

- Language model
- To generate and process text
- Trained on specific datasets
- No real time information
- No fact checking

Which is heavier, steel? or feather?

Steel is generally heavier than a feather. The weight of steel depends on its density and volume, while the weight of a feather depends on its size and composition.

The density of steel is about 7.85 grams per cubic centimeter, while the density of a feather varies depending on the type of bird it comes from. For example, the density of a feather from a small songbird such as a sparrow is typically around 1.3 grams per cubic centimeter, while the density of a feather from a larger bird such as a hawk or eagle is closer to 1.5 grams per cubic centimeter.

However, it's important to note that the weight of an object is not solely determined by its density, but also by its volume. A small piece of steel will weigh less than a large piece of steel, even if they have the same density. Similarly, a small feather will weigh less than a large feather, even if they have the same density.

So a kilogram of steel is heavier than a kilogram of feathers.

Yes, that's correct. A kilogram of steel will generally be heavier than a kilogram of feathers because the density of steel is much higher than the density of feathers.

Q&A