The fundamentals of technical writing

Tooling

Alexandra Nikandrova
Technical Writer

Dominika Borges
Technical Writer

About the authors:

Alexandra Nikandrova
Technical Writer.
Former Devops

Apurva Bhide
Senior RHEL Technical Writer

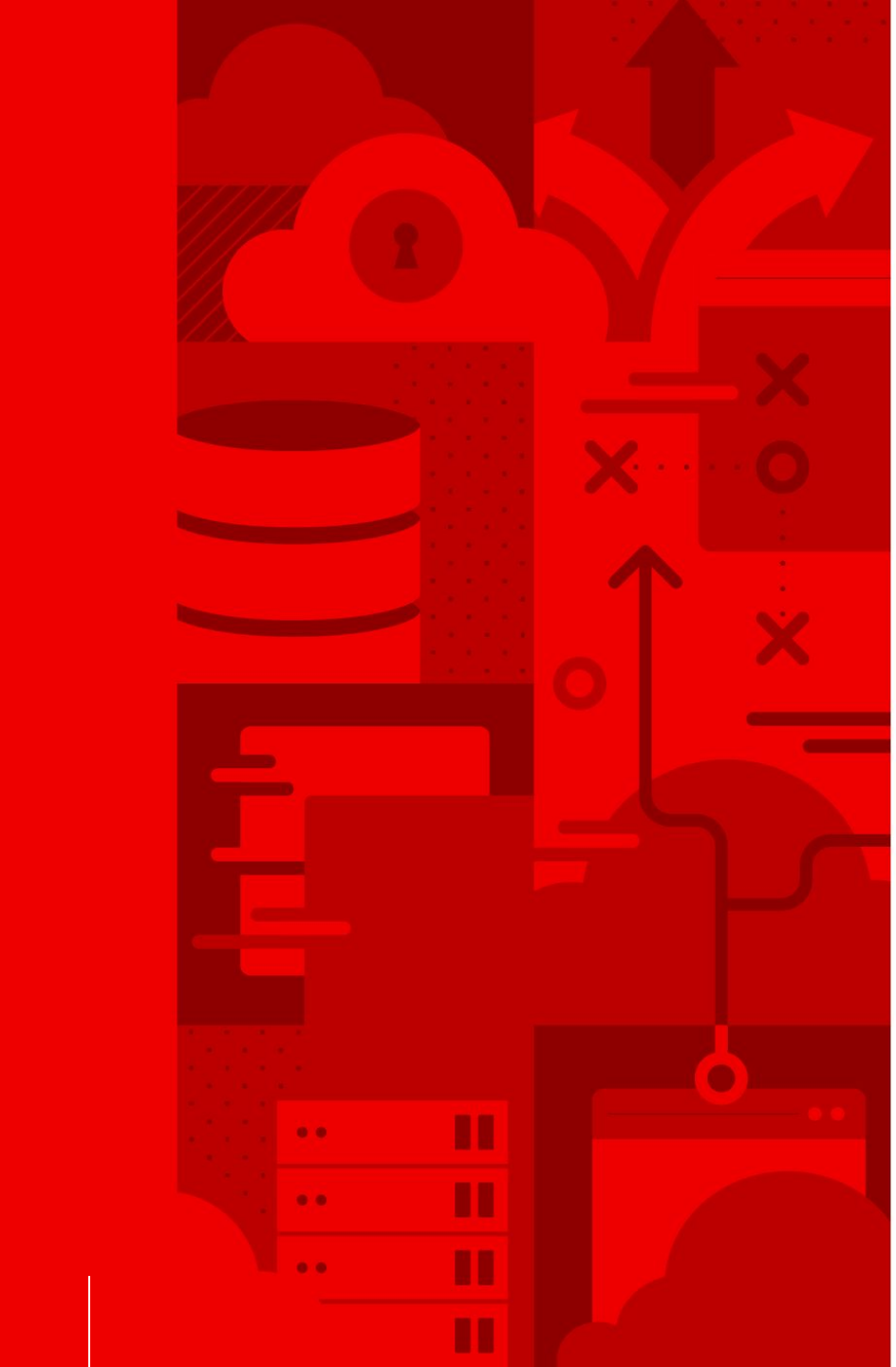
Dominika Borges
Technical Writer
Former journalist

Mayur Patil
Technical Writer 2
RHEL Cloud, Networking, and RDMA

Sagar Dubewar
Senior Technical Writer
RHEL Installer and Cloud

What we'll discuss today

- Homework follow-up
- Introduction
- Languages
  - XML
    - DocBook
    - DITA toolkit
  - reStructuredText (RTS), Sphinx
  - AsciiDoc
  - Markdown
- Editors
- Exercises
- Homework



Introduction

Why?

Single sourcing and content reuse

- To create multiple output formats (HTML, PDF, EPUB, and more)
- Reuse content in multiple versions, multiple products or multiple guides

Platform independence

- Consistent output across different platforms

Unified structure and consistency

- Unified structure and templating

Why?

Docs as code

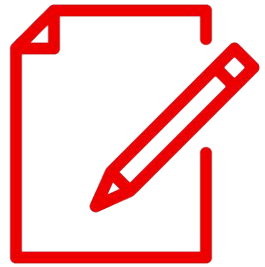
- Alignment with coding practices, versioning, and collaborative workflows

Readability

- Text-based and lightweight

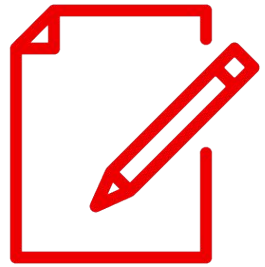
Focus on content

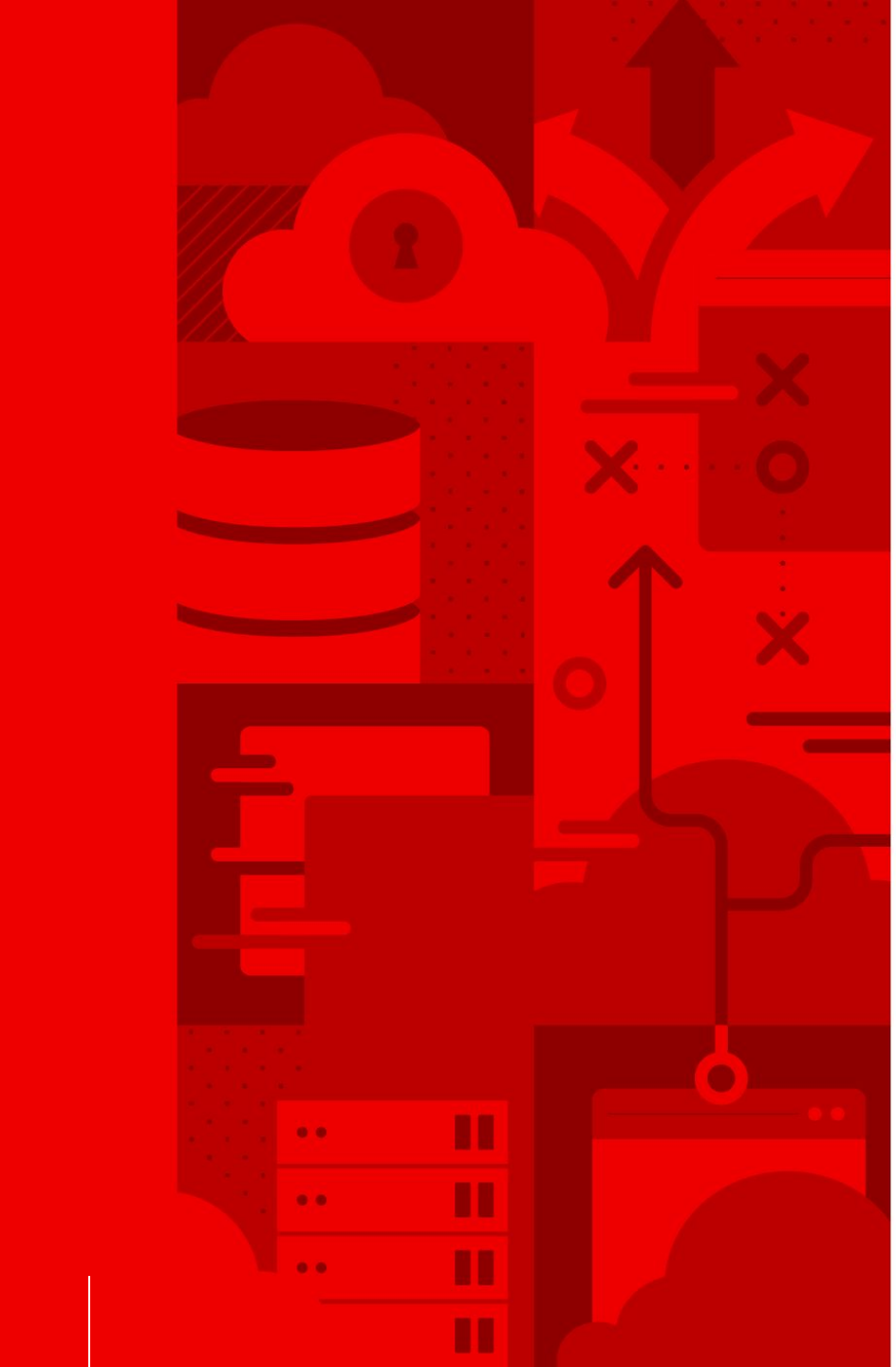
- Separation of content and formatting



How?

- Languages and structure created specifically for technical writers
- Templates
- Simplified English
- Doc-as-Code



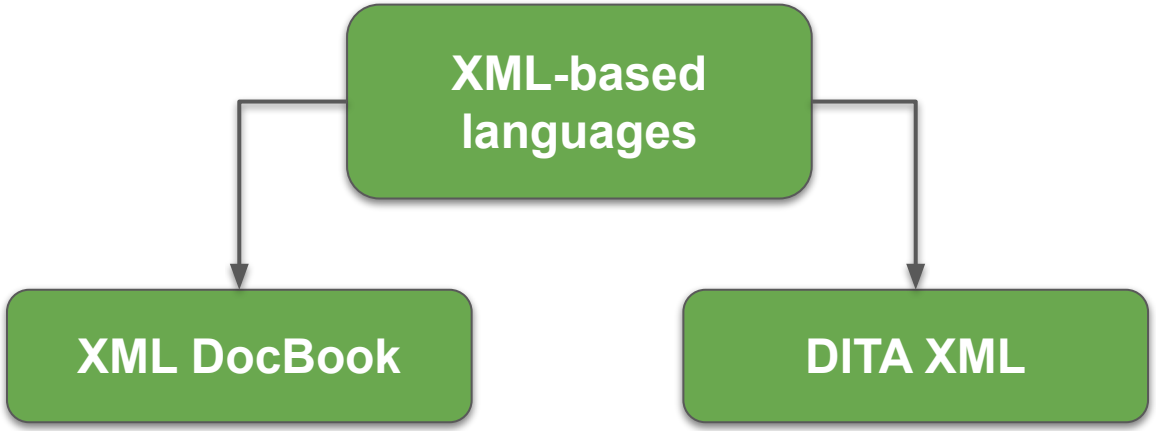


Languages

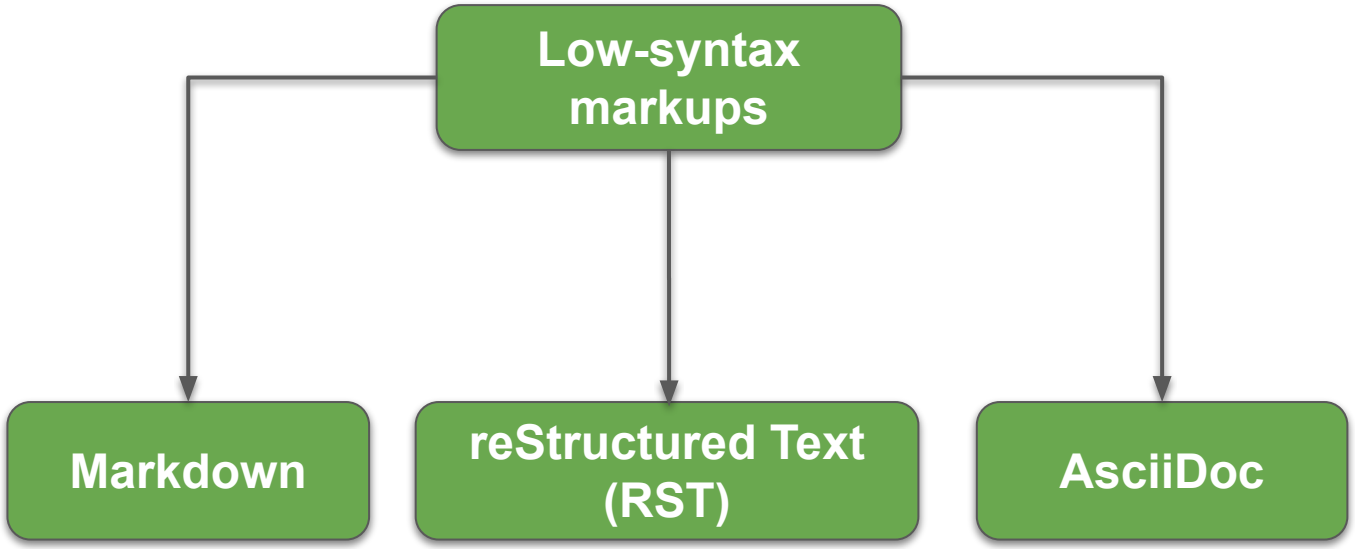
Markup languages

A tool to structure and format documents in a consistent and organized way

- separate content from formatting
- provide semantic meaning to content (accessibility)
- consistent output across different platforms
- unified structure and templating



- 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

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): Markup language and file format for storing, transmitting, and reconstructing arbitrary data.
- 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"?>`
- XML also supports nesting of tags.

DocBook

Advantages:

- self-explanatory, extensibility, single-sourcing

Disadvantages:

- Lengthy, less readable, strict syntax rules

Example

```
<?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 Architecture (DITA)

- DITA: XML-based architecture for creating modular content to reuse and interchange contents from various sources, such as topic-based authoring.

- Advantages:
  - CMS support
  - WYSIWYG
  - Compounding guides based on ver

- DITA Open Toolkit – the open-source publi
  engine: https://github.com/dita-ot/dita-ot/

Example

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

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

```
= Document Title

AsciiDoc is a lightweight and semantic
markup language primarily designed for
writing technical documentation.

.Labeled list
Term 1:: Definition
Term 2:: Definition

[source,python]
----
print("Hello, AsciiDoc!")
----

include::another-document.adoc[]

https://asciidoctor.org[AsciiDoc]
```

- A lightweight markup language
- Multiple outputs: HTML, PDF, EPUB, DocBook, and man page.
- Multiple document types: Articles, Books, Man pages.
- Rich features: Use of attributes, conditional text, snippets
- AsciiDoctor processor

- AsciiDoc Syntax Quick Reference
- AsciiDoc Recommended Practices

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 docs. 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 collaboration platforms like GitHub.
- Many Markdown flavors, limited success in standardization
- Resource: https://daringfireball.net/projects/markdown/

Xml-based vs low-syntax

```
<info>
<title>Creating d
<date>2024-02-20<
</info>
<simpara>Add crede
endpoints.
Before you can acc
user with the {bra
<tip>
<simpara>{brandnar
Create an <litera
<literal>ADMIN</li
</tip>
<itemizedlist>
<title>Prerequisi
<listitem>
<simpara>Download
</listitem>
</itemizedlist>
<orderedlist numer
<title>Procedure<
<listitem>
<simpara>Open a te
</listitem>
<listitem>
<simpara>Create ar
command.</simpara>
<programlisting la
linenumbering="unr
```

1.6. Creating Data Grid users

Add credentials to authenticate with Data Grid Server deployments through Hot Rod and REST endpoints. Before you can access the Data Grid Console or perform cache operations you must create at least one user with the Data Grid command line interface (CLI).

Tip

Data Grid enforces security authorization with role-based access control (RBAC). Create an `admin` user the first time you add credentials to gain full `ADMIN` permissions to your Data Grid deployment.

Prerequisites

- Download and install Data Grid Server.

Procedure

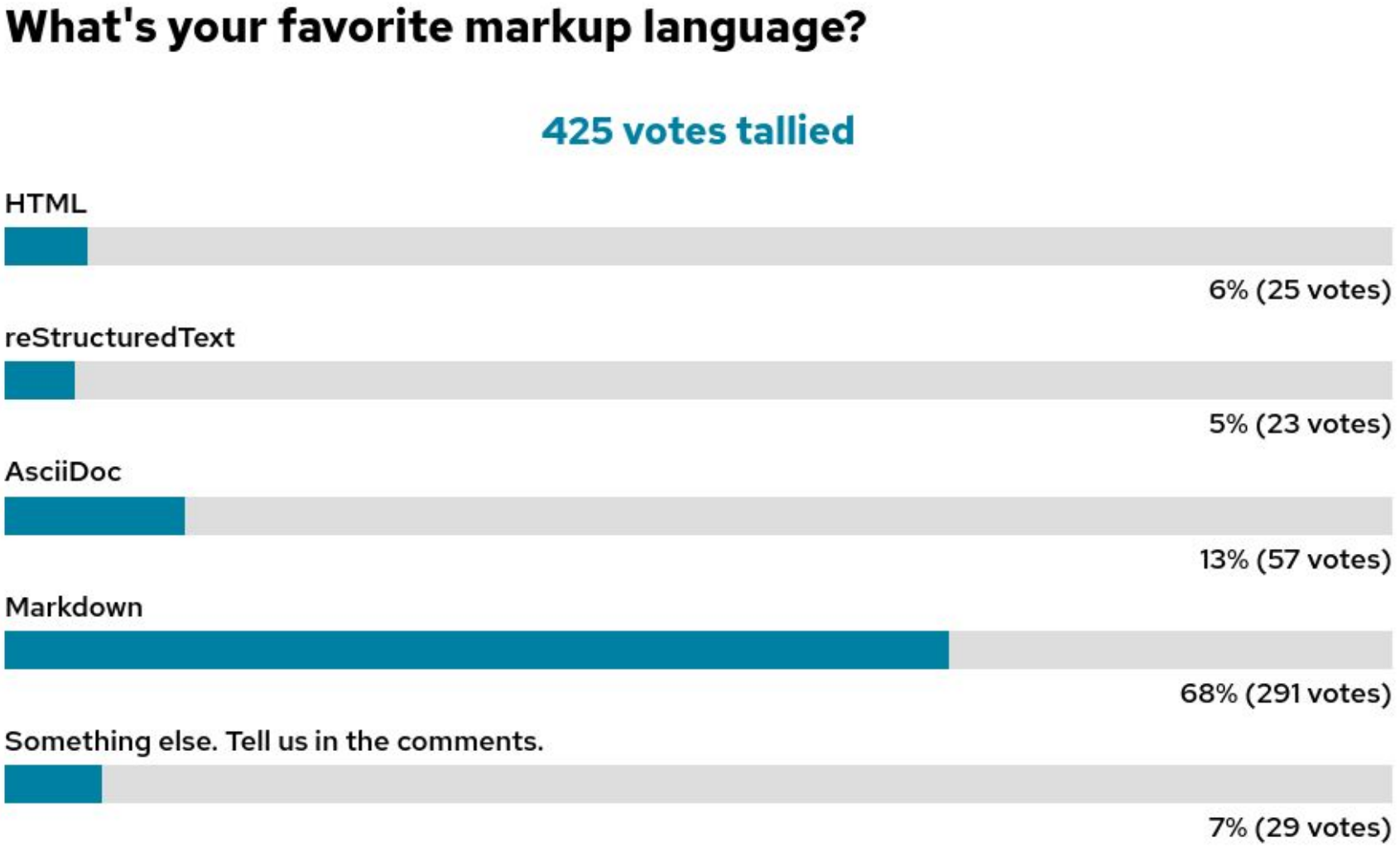
1. Open a terminal in `$RHDG_HOME`.
2. Create an `admin` user with the `user create` command.

```
bin/cli.sh user create admin -p changeme
```

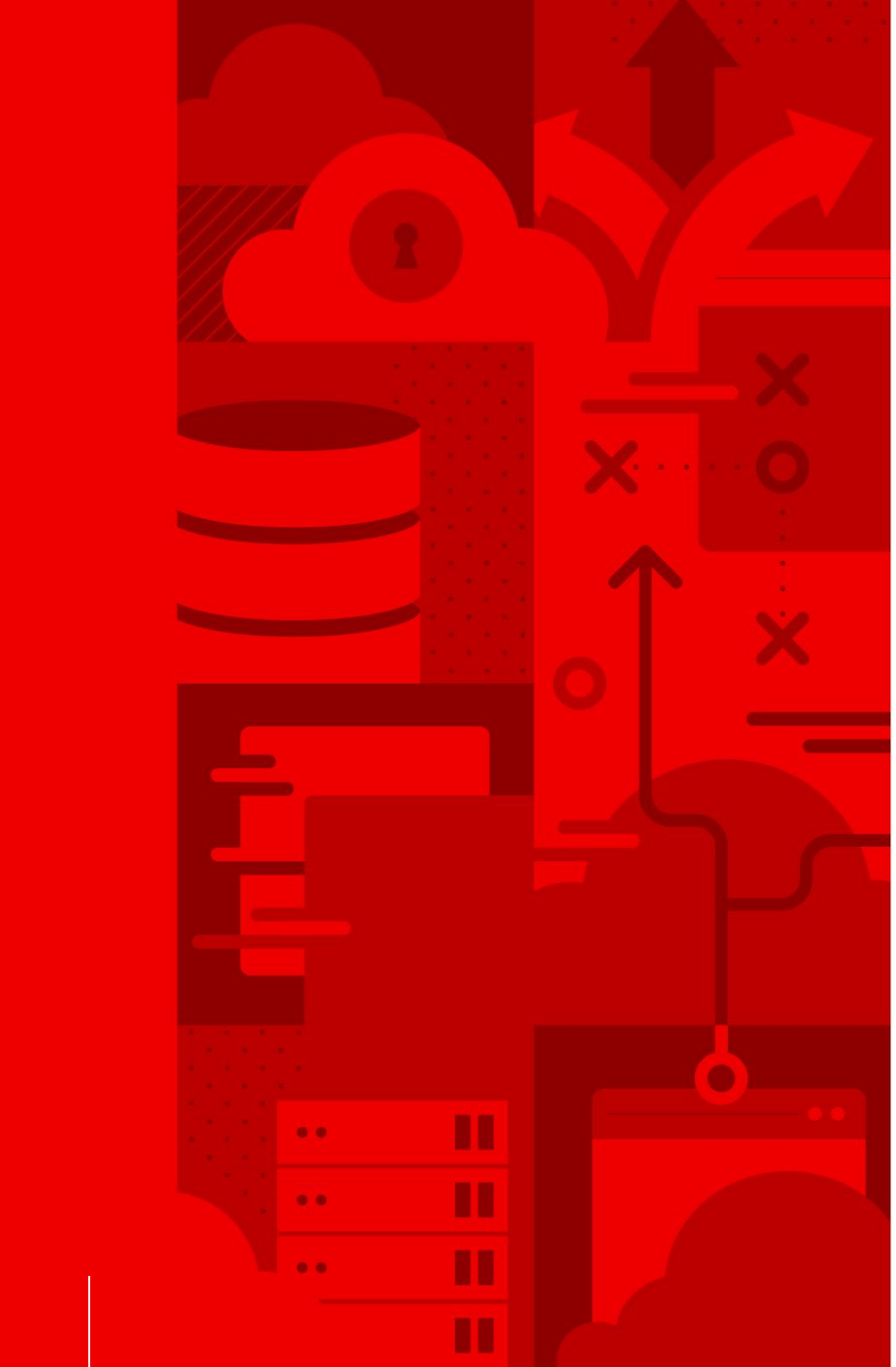
```
Hot Rod and REST

reate at least one

(RBAC).
ull

er create</literal>

g>
```

Summary

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



Source:
https://opensource.com/article/22/12/markup-languages-documentation



Editors

Editors

• **Standalone:**
  ○ Vim
  ○ VSCode
  ○ Notepad++
  ○ Apple Pages
  ○ Emacs
  ○ Sublime Text
	• **Web Based:**
	  ○ Stackedit
	  ○ Pandao
	  ○ Dillinger.IO
		• **Extensions:**
		  ○ Markdown Viewer
		  ○ Markdown Here

- Choose whatever is comfortable

Vim editor

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

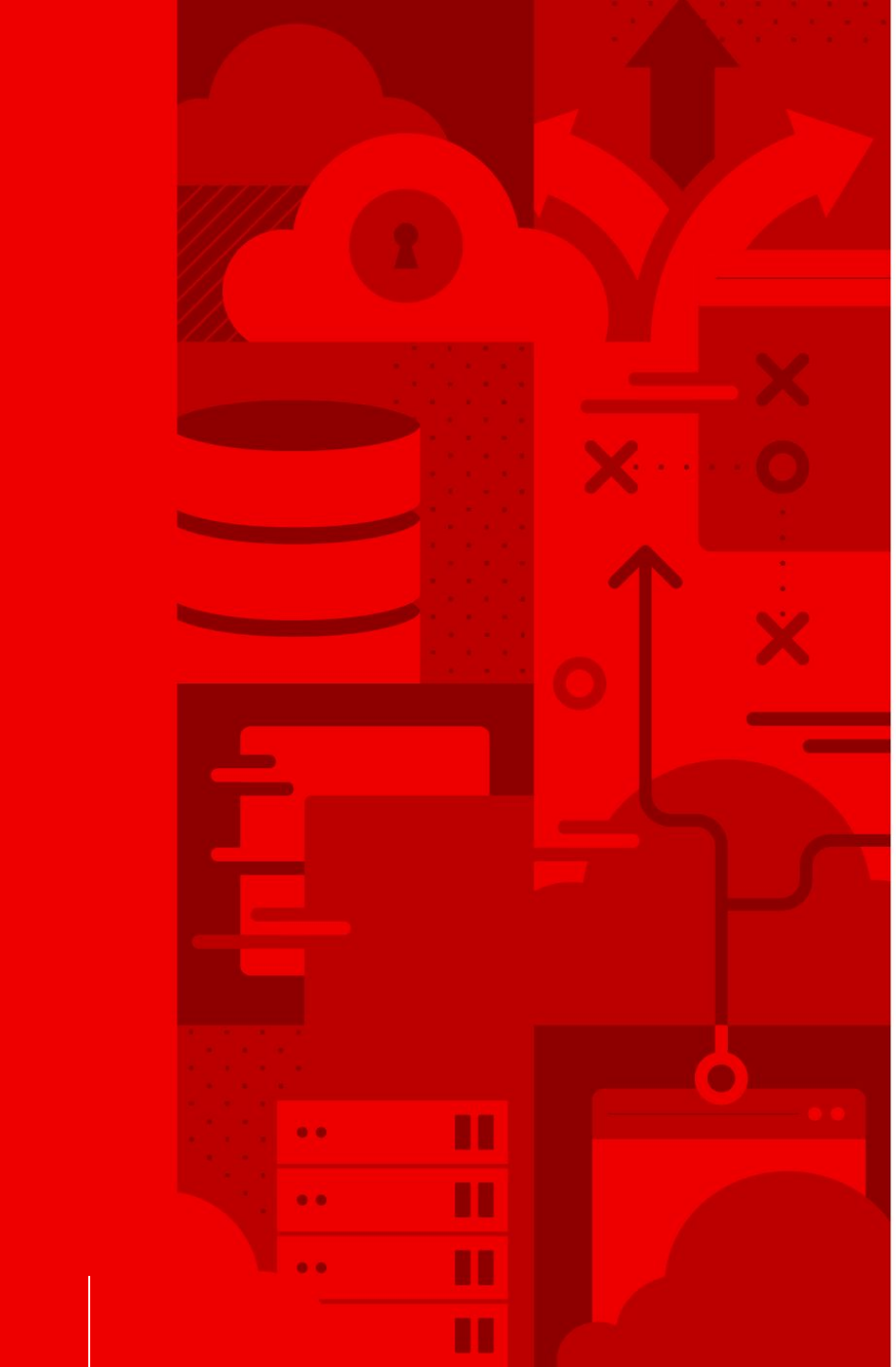
Command	Description
vim FILE_NAME	Create or modify the FILE_NAME in vim.
:q or :ZQ	Quit the file without saving. Perform in **command line** mode.
:x or :qw!	Save and quit file. Perform in **command line** mode.
dd	Delete the highlighted text or the current line. Perform in **normal** mode.
v	Highlight 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*

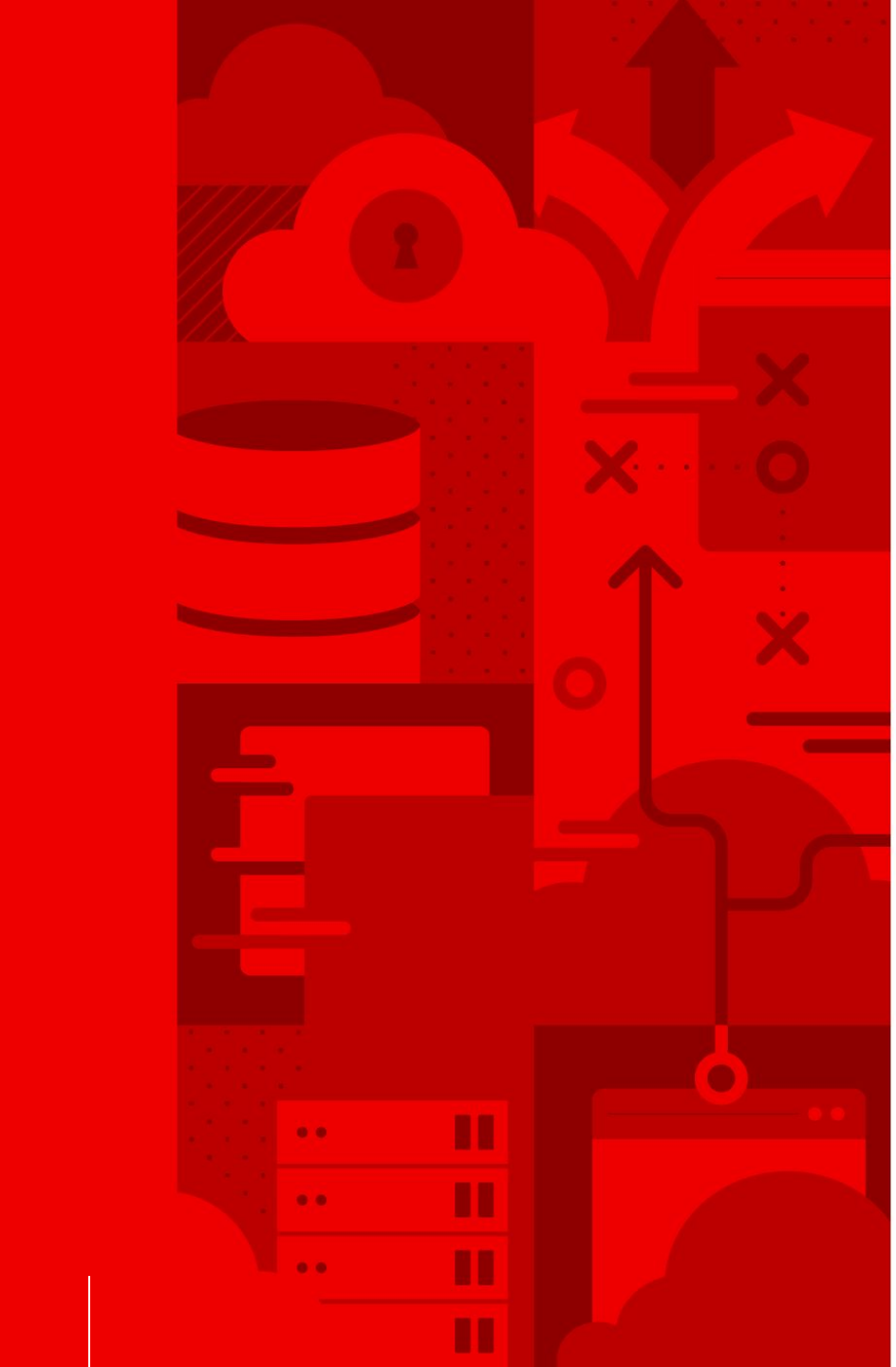
Useful links: *vim.org*

Demo: VSCode preview

Demo: Markdown syntax in VS Code



Q&A



Exercise

Time to practice Markdown

Rewrite and restructure the following text using **Markdown**

You can use editor of your choice, the fastest would be using one of the online editors **dillinger.io**

Duration **20 minutes**

Need help?
markdownguide.org/basic-syntax/
developers.google.com/style/text-formatting

Be creative

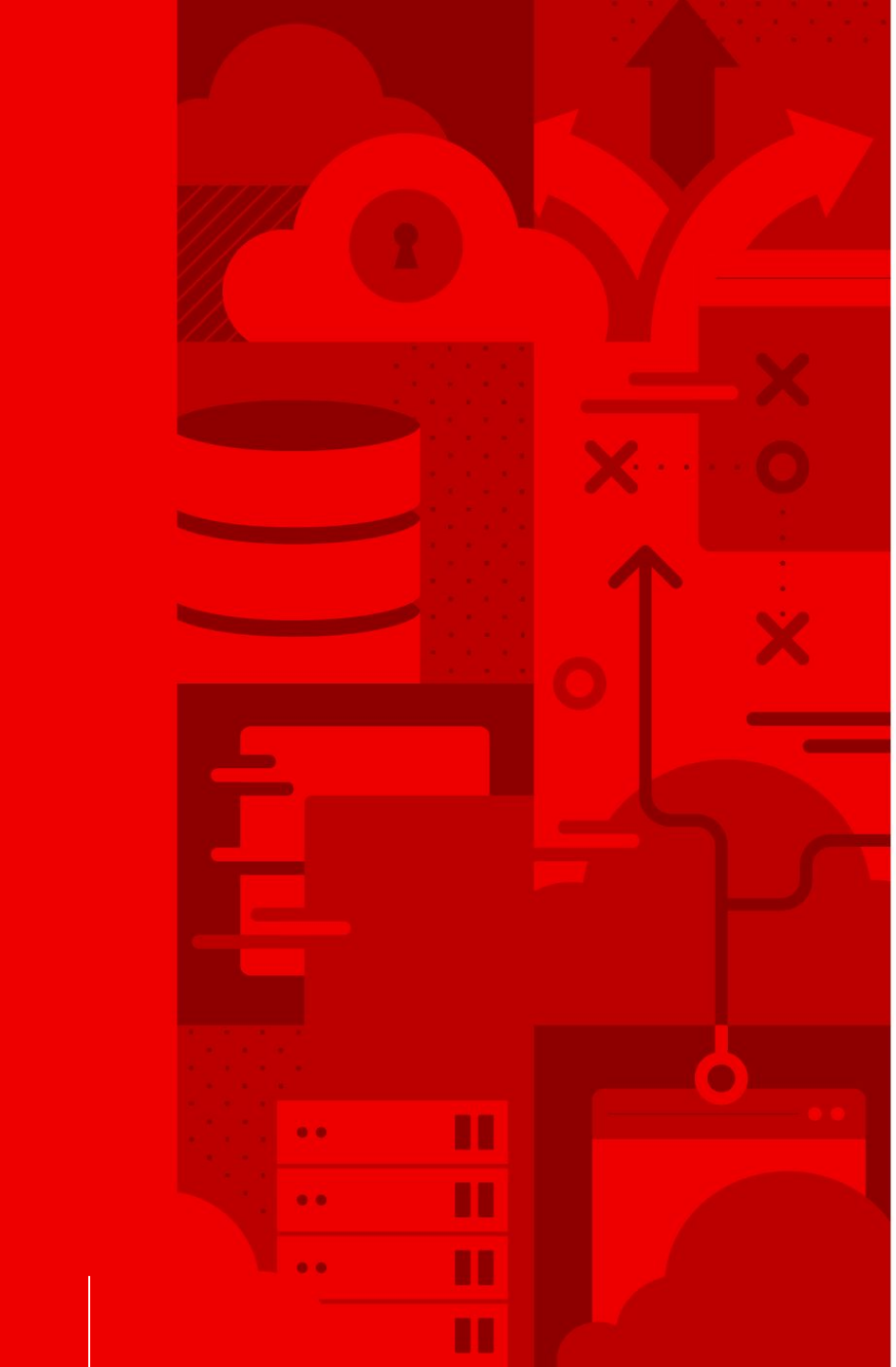
Markdown exercise reflection

Creating a playbook

Configuring Ansible

Takeaways

- Tools help you to do your work efficiently
- Languages have their place
- Most popular language in the open source is Markdown
- Use whatever editor you are comfortable with



Homework