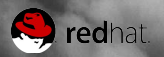
redhat.

TECHNICAL DOCUMENTATION

FOR STUDENTS OF MENDEL UNIVERSITY



WHO ARE WE

...and why we think we know anything about technical documentation

Jaromír Hradílek
Associate Manager, Red Hat

Lucie Maňásková
Associate Technical Writer, Red Hat

Štefan Sitáni
Technical Writer, Red Hat

WHAT YOU CAN EXPECT FROM US

...and what we expect from you in this workshop

Part I: Theory

- What is technical documentation
- What is modular documentation
- How do we write documentation in AsciiDoc

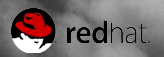
Part II: Exercise

- Let's write some documentation
- Let's talk about how it went

redhat.

WHAT IS TECHNICAL DOCUMENTATION

AND WHAT DO TECHNICAL WRITERS DO



redhat.

WHAT DO WE DO AT RED HAT

- Write customer-facing documentation for our entire product portfolio, that is, everything you can find on the Customer Portal here: https://access.redhat.com/documentation/
- Learn the technologies we document so that we can experiment, test, and come up with solutions.
- Communicate with customer-facing teams to understand the customer needs.
- Communicate with customers directly when possible.
- Contribute to documentation upstream where possible.

WHAT WE DON'T DO AT RED HAT

- Translate existing content. We write original content directly in American English.
- Take what developers write and make it sound better. We work from a user story and provide complete, focused documentation based on it.
- Take what developers write and make it sound worse.
- Write manual pages.
- Get bored.

WHAT IS IN IT FOR US

- An excuse to learn new technologies and test documented procedures to ensure their technical accuracy. This includes products like **OpenShift, OpenStack**, **Ceph Storage**, **Satellite**, **Red Hat Enterprise Linux**, and more.
- An opportunity to master the use of our internal documentation toolchain, including **AsciiDoc**, **Git**, and **GitLab**.
- A need to master the use of issue tracking systems like **Jira** and **Bugzilla**.
- Ability to work in an international team and communicate with subject matter experts from all over the world.
- An opportunity to visit universities and have workshops there.

WHAT ROLES DO WE HAVE

- **Technical Writer**

  Works with software developers, quality engineers, support engineers and other subject matter experts to understand the customer perspective and create effective solutions.

- **Documentation Program Manager**

  Works with Technical Writers and Content Strategists to execute on content plans through resource allocation, capacity planning, status reporting and problem solving.

- **Content Strategist**

  Works with Documentation Program Managers, Technical Writers, as well as product management and product marketing groups to create content strategy for product documentation and ensure it meets the requirements.

- **Other supporting roles**

CHALLENGES WE FACE

- Ever increasing complexity and product portfolio.
- High demand for cross-product documentation.
- Diverse and continually changing target audience.
- Changing trends in documentation and how customers consume it.
- Limited direct exposure to customers.
- The need to maintain thousands of pages of content for evolving products.

WHAT ARE WE DOING ABOUT IT

- Planning and creating content based on user stories.
- Embracing the concepts of minimalism and modular documentation.
- Being more opinionated in our documentation, describing only recommended paths that we can validate.
- Not describing every single thing you can do with a particular component.
- Partnering with Quality Engineering to have our documentation tested.
- Partnering with customer facing teams to understand customer needs.
- Enabling direct documentation feedback on the Customer Portal.

RED HAT TECHNICAL WRITERS IN EUROPE

2010

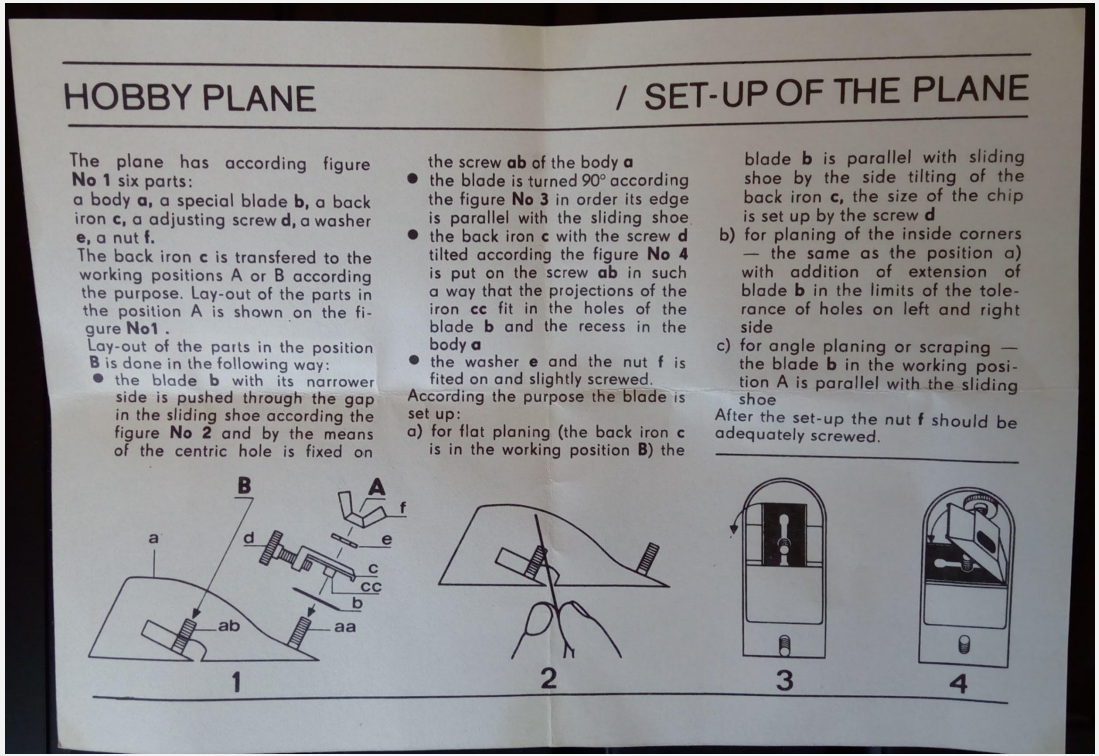
2019

redhat.

A SHORT DIGRESSION...

BECAUSE WE NEED TO TALK ABOUT THIS

A HOBBY PLANE SETUP INSTRUCTIONS



HOBBY PLANE / SET-UP OF THE PLANE

The plane has according figure **No 1** six parts:
a body **a,** a special blade **b,** a back iron **c,** a adjusting screw **d,** a washer **e,** a nut **f.**
The back iron **c** is transfered to the working positions A or B according the purpose. Lay-out of the parts in the position A is shown on the figure **No1 .**
Lay-out of the parts in the position **B** is done in the following way:

- the blade **b** with its narrower side is pushed through the gap in the sliding shoe according the figure **No 2** and by the means of the centric hole is fixed on the screw **ab** of the body **a**
- the blade is turned 90° according the figure **No 3** in order its edge is parallel with the sliding shoe
- the back iron **c** with the screw **d** tilted according the figure **No 4** is put on the screw **ab** in such a way that the projections of the iron **cc** fit in the holes of the blade **b** and the recess in the body **a**
- the washer **e** and the nut **f** is fited on and slightly screwed.

According the purpose the blade is set up:
a) for flat planing (the back iron **c** is in the working position **B**) the blade **b** is parallel with sliding shoe by the side tilting of the back iron **c,** the size of the chip is set up by the screw **d**
b) for planing of the inside corners — the same as the position a) with addition of extension of blade **b** in the limits of the tolerance of holes on left and right side
c) for angle planing or scraping — the blade **b** in the working position A is parallel with the sliding shoe

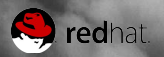
After the set-up the nut **f** should be adequately screwed.

redhat.

MODULAR DOCUMENTATION

IN A NUTSHELL

Technical Documentation for Students of Mendel University – May 6, 2019



redhat.

MODULAR DOCUMENTATION

Why modular documentation:

- Allows the writer to build documentation around a real user story rather than product capabilities and features.
- Allows the documentation to be more opinionated and focus on recommended paths.
- Is significantly less costly to maintain.
- Individual modules can be reused in different context.

Building blocks:

- User story.
- Modules.
  - Concept module.
  - Procedure module.
  - Reference module.

USER STORY

Description:

- Is written from the perspective of a particular user.
- Describes something that the user does to achieve a certain goal.
- Is typically short and fits in a single sentence.
- Common template:

  *As a **[user role]**, I want to **[desired action]** so that **[ultimate goal]**.*

Examples:

- As a system administrator, I need to be able to review available security updates along with their detailed descriptions prior to installing them in the system in order to assess security risks and keep the system as secure as possible.

CONCEPT MODULE

Description:

- Gives the user enough information needed to understand the described product.
- Titles typically start with nouns to indicate no action is performed in this module.
- The optional introductory paragraph explains what the concept module is about and why should the reader invest in reading it.
- The text typically consists of paragraphs, itemized lists, labeled lists, and diagrams. If necessary, it can be further divided in subsections.
- The module can optionally include a section with additional resources to link to sources with more detailed information.

Template:

Concept title

Introductory paragraph.

One or more paragraphs, itemized lists, labeled lists, or diagrams with necessary explanations.

Additional resources

- One or more links to additional resources.

CONCEPT MODULE

Concept title

Introductory paragraph

One or more paragraphs, itemized lists, labeled lists or diagrams with necessary explanations

1.1. Distribution of content in RHEL 8

RHEL 8 content is distributed through the two main repositories: **BaseOS** and **Application Stream** (AppStream).

BaseOS

The BaseOS repository provides the core set of the underlying OS content in the form of traditional RPM packages. BaseOS components have a life cycle identical to that of content in previous Red Hat Enterprise Linux releases.

Application Stream

The Application Stream repository provides content with varying life cycles as both modules and traditional packages. Application Stream contains necessary parts of the system, as well as a wide range of applications previously available as a part of Red Hat Software Collections and other products and programs.

IMPORTANT

Both BaseOS and AppStream are a necessary part of a Red Hat Enterprise Linux system.

PROCEDURE MODULE

Description:

- Provides detailed steps to perform a particular action.
- Titles typically start with gerunds to indicate an action is performed in this module.
- The introductory paragraph explains who should perform the procedure and why. If there are any special considerations and risks, they should be mentioned here.
- Prerequisites list conditions that must be met before performing the procedure.
- The module can optionally include a section with additional resources to link to sources with more detailed information.

Template:

Procedure title

Introductory paragraph.

Prerequisites

- One or more prerequisites.

Procedure

1. Steps required to perform the action.

Additional resources

- One or more links to additional resources.

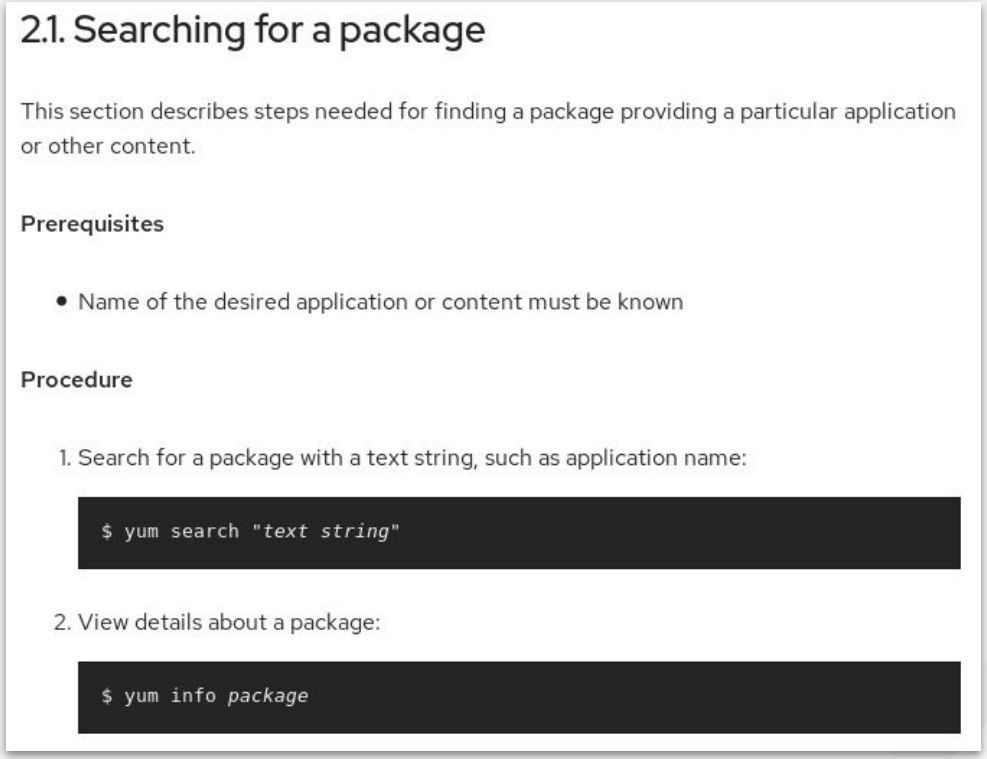
PROCEDURE MODULE

Procedure title

Introductory paragraph

One or more prerequisites

Steps required to perform the action



2.1. Searching for a package

This section describes steps needed for finding a package providing a particular application or other content.

Prerequisites

- Name of the desired application or content must be known

Procedure

1. Search for a package with a text string, such as application name:

```
$ yum search "text string"
```

2. View details about a package:

```
$ yum info package
```

REFERENCE MODULE

Description:

- Provides important information that users might need but don't need to remember.
- Titles typically start with nouns to indicate no action is performed in this module.
- The introductory paragraph explains what the reference module is about and when should the reader invest in reading it.
- The reference module typically contains a single itemized list, labeled list, or a table. If necessary, it can be further divided in subsections.
- The module can optionally include a section with additional resources to link to sources with more detailed information.

Template:

Reference title

Introductory paragraph.

A single itemized list, labeled list, or a table.

Additional resources

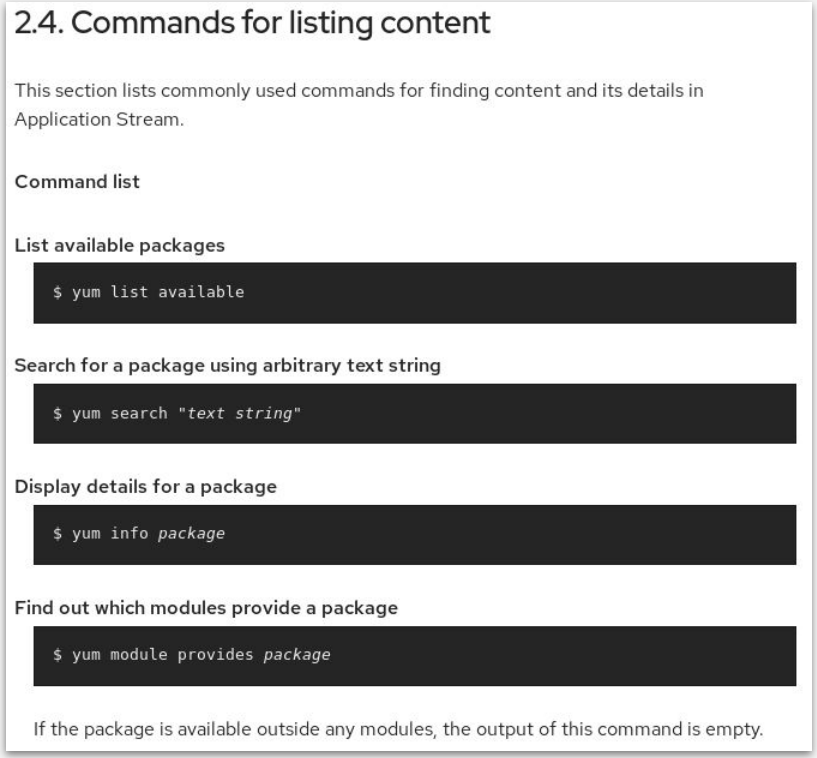
- One or more links to additional resources.

REFERENCE MODULE

Reference title

Introductory paragraph

A single itemized list, labeled list, or a table



2.4. Commands for listing content

This section lists commonly used commands for finding content and its details in Application Stream.

Command list

List available packages

```
$ yum list available
```

Search for a package using arbitrary text string

```
$ yum search "text string"
```

Display details for a package

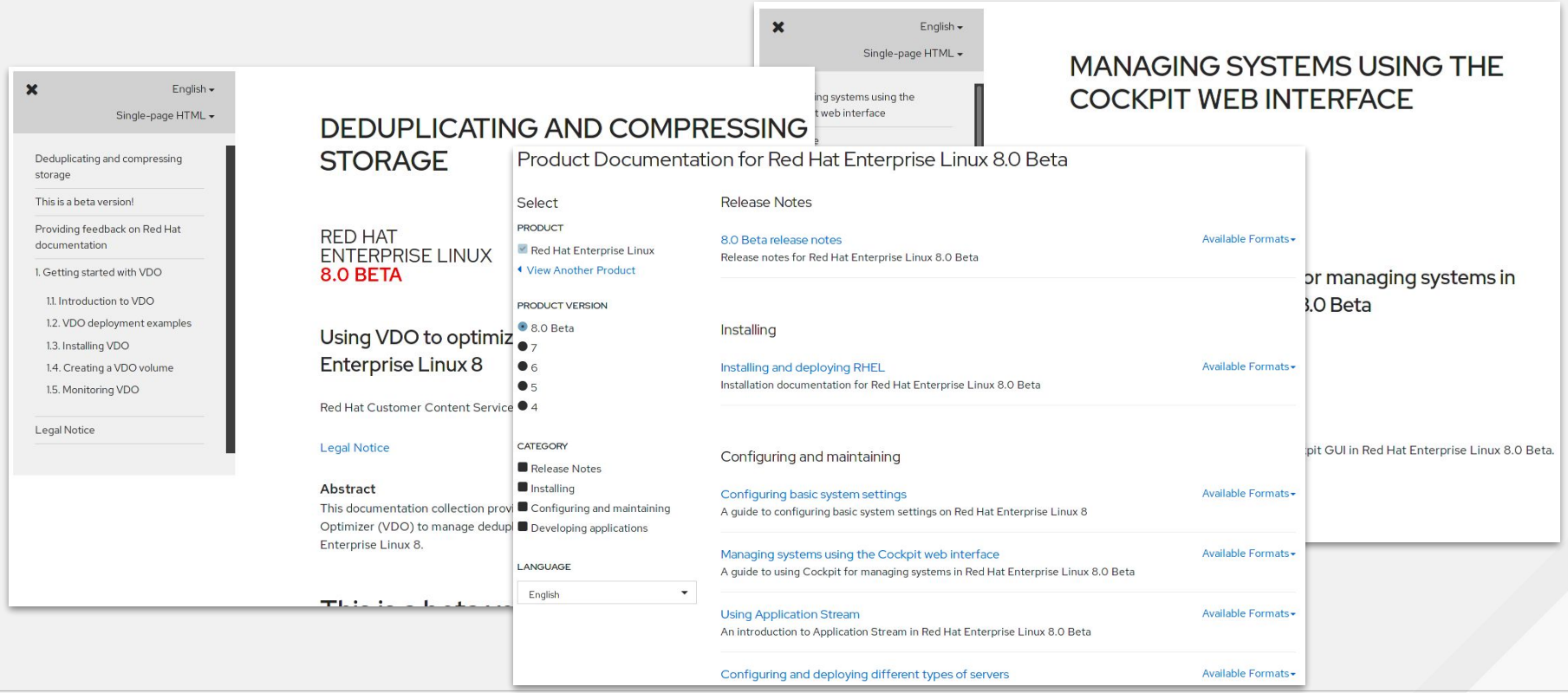
```
$ yum info package
```

Find out which modules provide a package

```
$ yum module provides package
```

If the package is available outside any modules, the output of this command is empty.

WHEN WE PUT ALL TOGETHER



redhat.

ASCIIDOC BASICS

IN THE **ATOM** EDITOR

Technical Documentation for Students of Mendel University – May 6, 2019

redhat.

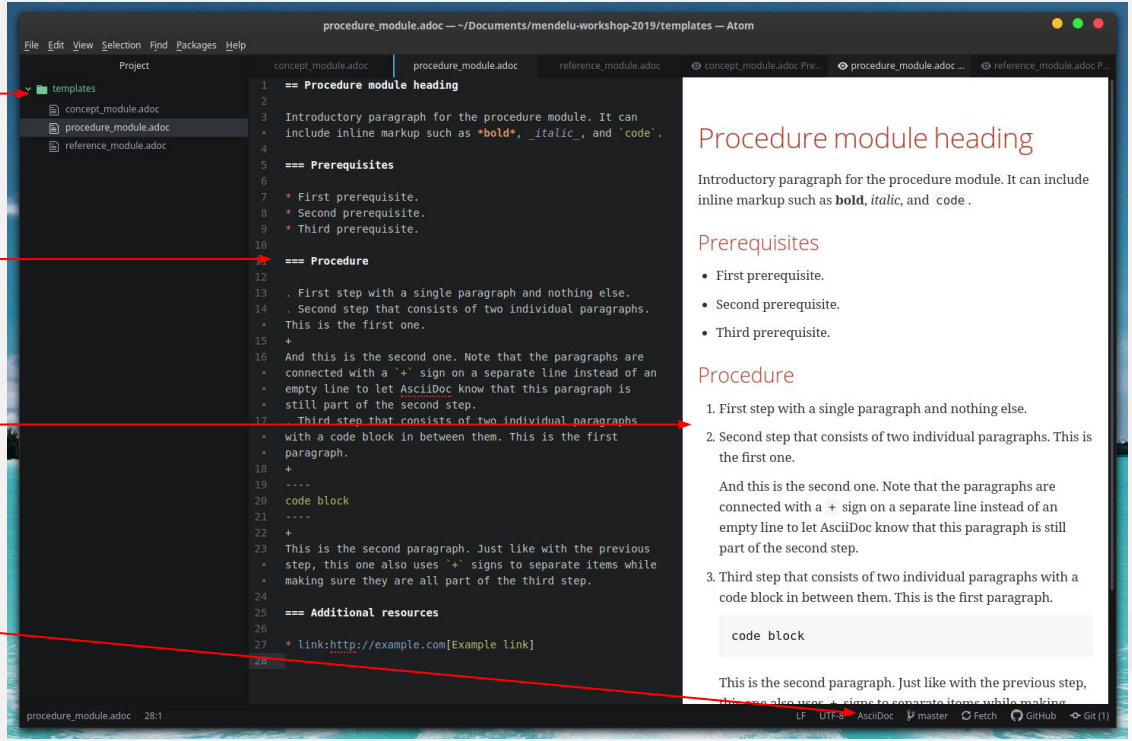
TEXT EDITOR: ATOM

Project view
View → Toggle Tree View
Ctrl + \

Edit area
View → Toggle Soft Wrap

AsciiDoc Preview
Packages → AsciiDoc Preview → Toggle Preview
Ctrl + Shift + A

File Type
AsciiDoc



redhat.

DOCUMENT TITLE

The document title is the first heading in an AsciiDoc document. It starts with the equal sign at the beginning of the line followed by the title text. It must be followed by at least one empty line to separate it from the rest of the document content.

Syntax:

```
= Document title
```

Example:

SECTION HEADINGS

Section headings start with two or more equal signs at the beginning of the line followed by the heading text. The number of equal signs indicates the heading level. Section headings must be followed by at least one empty line to separate them from the rest of the document content.

Syntax:

```
== Section heading (level 1)

=== Subsection heading (level 2)

==== Subsubsection heading (level 3)
```

Example:

PARAGRAPHS

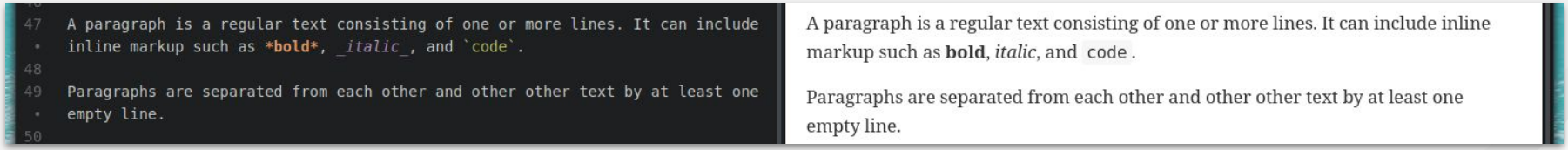
A paragraph is a regular text consisting of one or more lines. It can include inline markup such as *bold*, _italic_, and `code`. Paragraphs are separated from each other and other other text by at least one empty line.

Syntax:

```
First paragraph.

Second paragraph.
```

Example:



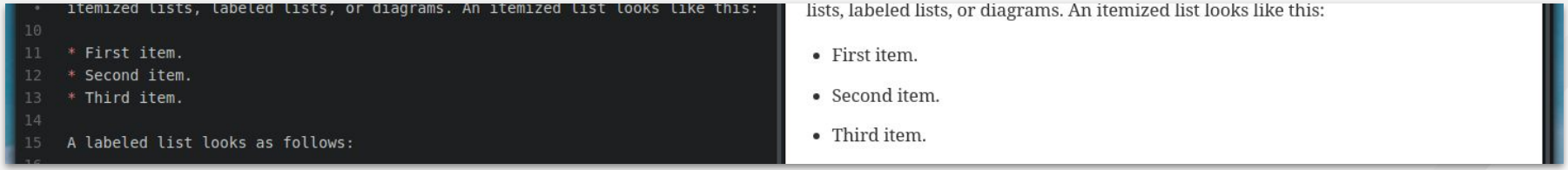
ITEMIZED LISTS

Itemized lists, or unordered lists in AsciiDoc terminology, consist of one or more list items with a asterisk at the beginning of the line followed by the list item text. Itemized lists must be followed by at least one empty line to separate them from the rest of the document content.

Syntax:

```
* First item.
* Second item.
* Third item.
```

Example:



ORDERED LISTS

Ordered lists are typically used in procedures for steps. Ordered lists consist of one or more list items with a dot at the beginning of the line followed by the list item text. Ordered lists must be followed by at least one empty line to separate them from the rest of the document content.

Syntax:

```
. First item.
. Second item.
. Third item.
```

Example:

```
36  === Procedure
37
38  . First step.
39  . Second step.
40  . Third step.
41
42  === Additional resources
43
```

Procedure

1. First step.
2. Second step.
3. Third step.

CODE BLOCKS

Code block begins with four minus signs on a separate line and ends with another for minus signs on a separate line. Anything in between these lines is considered a code block and printed as is, with no special character replacement, with retained indentation, and so on.

Syntax:

```
----
My code block.
----
```

Example:

EXTERNAL LINKS

AsciiDoc automatically recognizes links beginning with http:// or https:///. To use and alternative text for the link, you can follow the link with the text in square brackets. If the link contains spaces or special characters, you may prefix it with the link: keyword without a space.

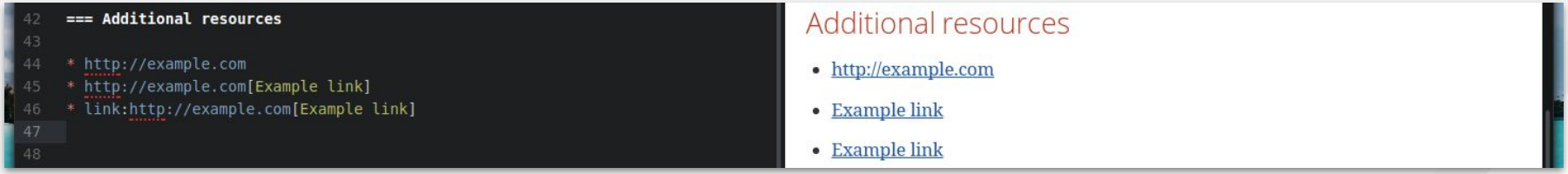
Syntax:

```
http://example.com

http://example.com[alternative text]

link:http://example.com[alternative text]
```

Example:



redhat.

PRACTICAL EXERCISE

redhat.

LET'S WRITE SOMETHING

Task: Write documentation based on one of the following user stories:

- *As a system administrator, I need to configure my machine to boot in text only mode, automatically start the Apache HTTP Server on port 80 and SSH on port 22, and disable unnecessary services like Bluetooth so that I can use it as a simple web server.*
- *As an adult with limited baking experience and minimal decorating skills, I want to create a layered birthday cake for my best friend's upcoming birthday party.*

Use google, experiment with the system in front of you if applicable, and reach out to us if you need advice. We are your subject matter experts here.

Tip: Use templates located in the Git repository created for this workshop:

- https://github.com/jhradilek/mendelu-workshop-2019

REVIEW AND DISCUSSION

LET'S REVIEW WHAT YOU WROTE

Reflect:

- How much did you manage to complete in the time you had?
- Are there any missing concepts, procedures, or reference modules?
- Can each module stand alone?
- How much did you fit in the templates?

Q&A

THANK YOU!