Template version: 1.0

HLD for Core Network

[[doc-subtitle]]

High Level Design

Vodafone
vodafone.de

Document Number: 001
Version: [[doc-version]]
Release Date: [[release-date]]

Document History

Author Name: [[doc-author]]
Author Contact: [[doc-author-email]]

Document Revision

Version Number	Release Date	Status	Change Description	Author
[[doc-version]]	[[doc-release-date]]	Draft	[[change-description]]	[[doc-author-name]]

Intellectual Property Rights

Table of Contents

Table of Tables

Table of Figures

1 Introduction

1.1 References

Referenced documents in creation of this document are listed in the Table 1.

Document Reference	Document Title	Version	Filename
[1]	[[reference-doc-title]]		[[reference-doc-filename]]
[2]			
[3]			
[4]			
[5]			
[6]			
[7]			
[8]			

Table 1: Document references

2 PC-MD-Testing

2.1 PuzzlesCloud MarkDown Test - PC_H1

text

Alternatively, for H1 and H2, an underline-ish style:

2.2 Alt-H1

2.2.1 AAA - PC_H2

text

2.2.2 Alt-H2

2.2.2.1 AAA - PC_H3

Text

2.2.2.1.1 AAA - PC_H4

Text

2.2.2.1.1.1 AAA - PC_H5

Text

2.2.2.1.1.1.1 AAA - PC_H6

Text aaa - PC_Normal

aaa - PC_Bold

bbb - PC_Bold

aaa - PC_Italic

bbb - PC_Italic

```
aaa - PC_BlockCode
```

- *aaa - PC_BulletList*
- *bbb - PC_BulletList*
- *aaa - PC_BulletList*
- *bbb - PC_BulletList*
- *aaa - PC_BulletList*
- *bbb - PC_BulletList*
- *aaa - PC_NumberedList*
- *aaa - PC_NumberedList*
    1. *aaa - PC_BulletList - Unordered sub-list*
    2. *bbb - PC_BulletList - Unordered sub-list*
- *aaa - PC_NumberedList*
    1. *aaa - PC_BulletList - Ordered sub-list*
    2. *bbb - PC_BulletList - Ordered sub-list*
- *[ ] aaa - PC_TaskList*

website PC_Link

Tables	Are	Cool
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

Table: Table Title - PC_TableCaption

-

`code` - PC_Code

```
code block - PC_Code
```

Three or more...

Hyphens

Asterisks

Underscores Three or more#

3 blog

3.0.1 PuzzlesCloud Team Selected for Raising Starts Program

3.0.1.1 Partnership with Raising Starts Program

The PuzzlesCloud Team has been selected to participate in the Raising Starts program designed to support startups in the earliest development stages, which with the support of the Government of Switzerland implements the Science Technology Park Belgrade. Through work with mentors, training and comprehensive professional support, the Program will enable us to acquire knowledge and skills to accelerate the development of our innovation, open the doors of the global market and reach new potential investors.

The program is implemented within the project "Technopark-Serbia 2 - Encouraging exports through the development of technological parks", by which the Government of Switzerland supports the further development of innovations and expansion of the network of technology parks, conducted by the Science and Technology Park Belgrade.

More information about the program is available at https://ntpark.rs/raising-starts/

3.0.2 Why docs-as-code?

3.0.2.1 Introduction

Documentation as Code Concept (docs-as-code) is getting more into the focus. Considering that many organizations, even outside of IT, are moving to Agile and DevOps ways of operation and docs-as-code concept natively supports it, it is completely natural that is getting so much attention.

Documentation as Code (docs-as-code) refers to a philosophy of writing documentation with the same tools as code. The most popular tool nowadays for writing, developing and maintaining code is Git version control system. Git in combination with MarkDown (you probably wrote a lot README.md files) is most convenient for docs-as-code as well.

Documentation written this way is called source and represents single source of truth. In other words, can be converted to any other form, namely HTML, PDF, DOCX, or any other needed. This conversion is usually happening with CI/CD pipelines, therefore automatically.

Considering previous conversion use cases, we could divide the concept into two areas:

Docs-as-code for online documentation (generated static HTML pages out of the source files)
Docs-as-code for offline docs delivery(generated PDF, DOCX or any other documents)

3.0.2.2 Docs-as-code for Online Docs and Blogs

The technical concept of having static content in Git and out of it create static websites (docs, blogs) is supported by Static Site Generators and is called Jamstack. With this concept all documentation is kept as the source files, in one of the lightweight markup languages, such as MarkDown, asciidoc or similar formats. Those files are converted by some of the Static Site Generators to HTML pages, or other formats (e.g PDF)

Even writing blogs could be conducted with the same approach.

This approach could be further enriched by embedding feedback to the HTML documentation by using native Git features, such as issue tracker, comments etc.

The trend in the last couple of years is to create docs websites (docs.example.com), on some platform that natively supports it: GitHub or GitLab with Pages service.

Some examples of the most popular Open Source Static Site Generators (SSG) are: Jekyll (supported by GitHub and GitLab), Hugo, Next, Gatsby (supported by GitLab only). The full list of available SSGs on the market could be found on the following link: https://jamstack.org/generators/.

There are also corresponding Headless CMSs used for the content management. There are Git and API based and both have its own advantages. The list of available Headlass CMSs can be found on the following link: https://jamstack.org/headless-cms/
Nice examples of how Microsoft, GitHub or GitLab have done it could be found on the following links: 1. https://docs.microsoft.com/en-us/teamblog/a-new-feedback-system-is-coming-to-docs 2. https://github.com/github/docs 3. https://gitlab.com/gitlab-org/gitlab-docs

3.0.2.3 Docx-as-code for Offline Docs

DevOps and Agile teams develop also a lot of documentation. Some examples are:

- *API documentation*
- *Internal Knowledge Bases*
- *Internal Operational Procedures*
- *Design and Architecture Documents etc*

Very often they need to work with Tech Writers, Product Teams and Marketers to deliver professional documentation to their end customers. There the problems starts. There are no common tools where all feel comfortable. One side uses Git and Markdown (DevOps and Agile team members) and the other mainly Word/DOCX (Tech Writers,Product Teams and Marketers).
The answer would be docx-as-code. They all work in Git on the source files and Marketers could change DOCX templates on regular basis. All would have available very professional documents for external deliverables continuously delivered, as part of the CI/CD automation process. Examples of the documents that could be delivered this way are the following:

- *Design Documents*
- *User Guides*
- *Product docs*
- *API docs*
- *Code tutorials*
- *Operational Guides*
- *Technical Specifications*

The list is of course not exhaustive, could be many more.
Source documents mentioned in the previous paragraph for Online documentation could also be re-used for this purpose.

3.0.2.4 Benefits of docs-as-code using Git

If you ever used Git, you would know what kind of benefits it brings with its native features. For the documentation writing, it would bring the following benefits:

- *Forget about document re-formatting. All documents are written in simple MarkDown format, which doesn't have great variety of formats/styles supported*
- *Allow seamless documentation writing collaboration with possibility to work in offline mode (locally on your PC)*
- *Native review/approval workflow through branching and pull/merge requests*
- *Simple documentation issue reporting and tracking*
- *Keep track of every single documentation line in version control system*
- *Check the diffs historically and understand changes between two document versions*
- *Native documentation release management (possible also to take continuous publishing approach)*
- *Tag the documents exported to your customer while keep working on them continuously*
- *Someone deleted your document -- no problem, restore it from the previous Git version*

Test the documentation and get error free docs The last mentioned great advantage of using docs-as-code approach is documentation testing possibility (executed during CI/CD pipeline run), which will provide error free documents (functional hyperlinks, correct URL references, correct formatting, great look etc.)

The intention is that Technical Writers, Developers, Product Teams and all others involved in docs production process use the same tool set and collaborate in same environment. This approach would boost productivity drastically.

3.0.2.5 Conclusion

Unblock your productivity and efficiency progress and start practicing docs-as-code now. If you improve your productivity every month 2%, year over year will be 1/4. Think about it.

For further reading we would recommend the following references: 0. https://technology.blog.gov.uk/2017/08/25/why-we-use-a-docs-as-code-approach-for-technical-documentation/ 1. https://docs.microsoft.com/en-us/teamblog/a-new-feedback-system-is-coming-to-docs 2. https://www.writethedocs.org/guide/docs-as-code/ 3. https://docs-as-co.de/ 4. https://idratherbewriting.com/docs-as-code-tools-and-workflows/#/1

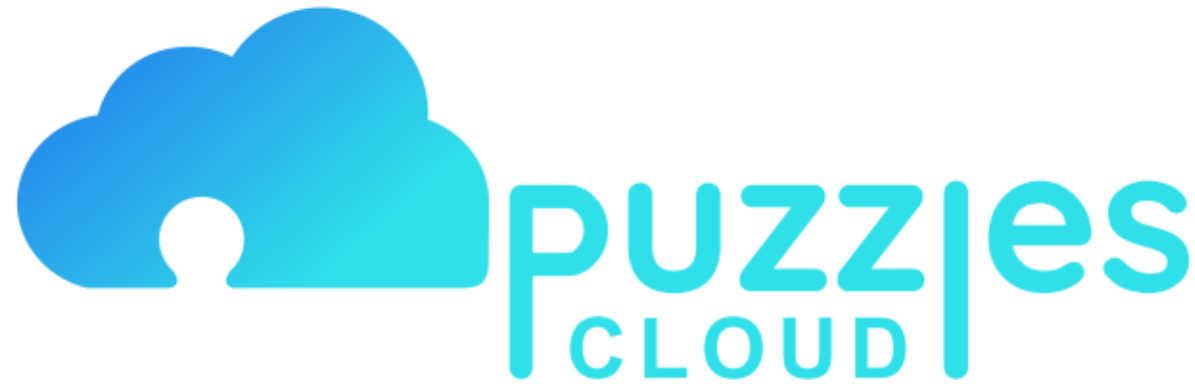

Figure 1: Last Page