



Automated Doc Generation

Jose Miguel Izquierdo

Technical Marketing Engineer @ Juniper Networks

A U T O C O N 2

THE NETWORK AUTOMATION CONFERENCE

Jose Miguel Izquierdo

- 12 years at Juniper
 - 10y as Professional Services Consultant
 - 2y as Technical Marketing Engineer
- Based in Madrid, Spain (EMEA)
-  linkedin.com/in/josemiguelizquierdo



JoseMi

- 12 years at Juniper
 - 10y as Professional Services Consultant
 - 2y as Technical Marketing Engineer
- Based in Madrid, Spain (EMEA)
-  linkedin.com/in/josemiquelizquierdo

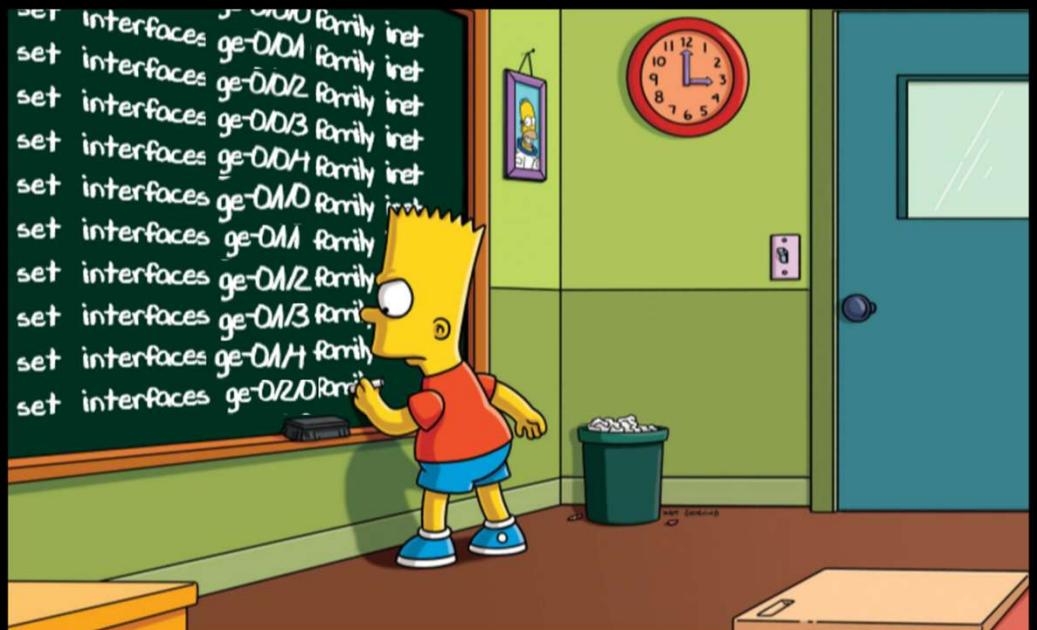


Agenda

- Challenges
- Motivations
- Solution
- Tools & Examples
- Use cases (2x)

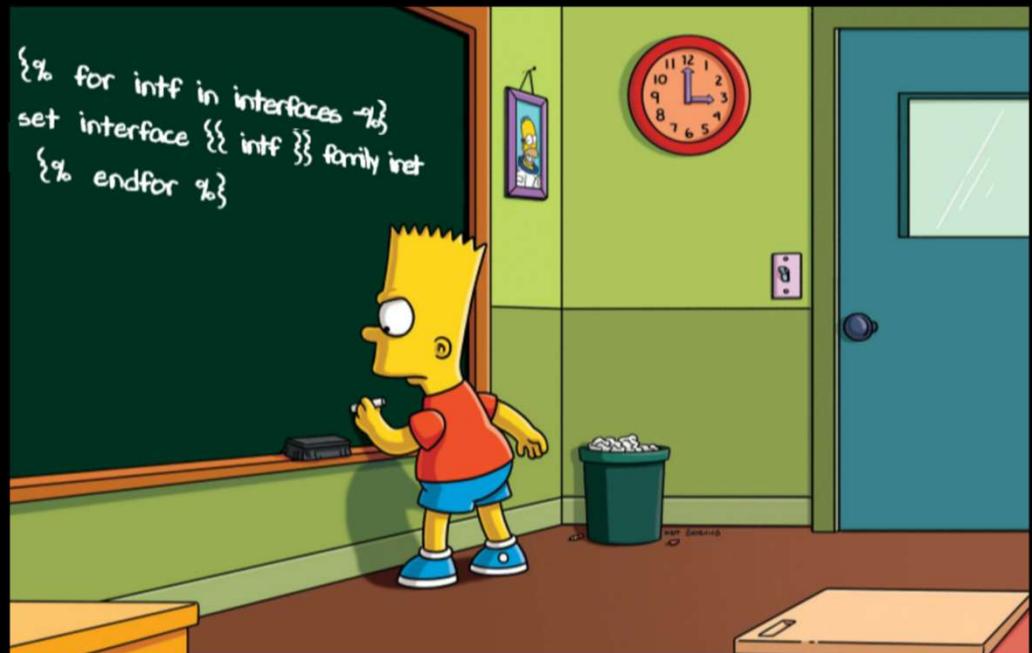
Manual Documentation challenges

- Time-consuming
- Prone to human error
- Inconsistent across iterations and projects
- Difficulty in maintaining version control



Motivations

- Standardization
- Consistent look & feel
- Tracking control
- Change management
- Reusability
- Integration



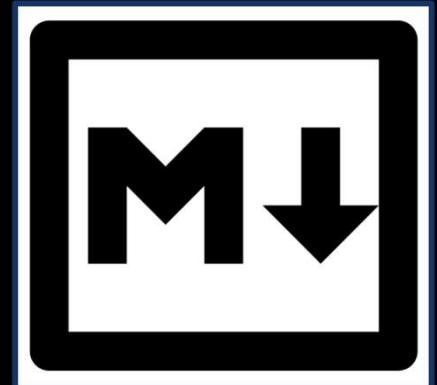
Solution

Combining tools

- Markdown
- Mermaid
- PlantUML
- Jinja2
- Ansible
- Docker
- Jenkins
- Git
- Version Control systems
- Pandoc
- LaTeX
- AI

Markdown

- Lightweight markup language with plain text formatting syntax.
- Easy to write & read (just a few special chars)
- Simplicity
- Portability (easily converted to pdf, word, etc...)
- Works with almost any text editor



Markdown examples I

```
1
2 # Heading 1
3 ## Heading 2
4 ### Heading 3
5
6 [www.google.es](https://www.google.es) – clickable links (URL)
7
8 [x.cfg](./files/x.cfg) – clickable links (open local files)
9
10 ![image](./images/JNPR White Background.png) – images
11
12 ## Lists
13
14 1. Ordered list
15 2.
16
17 - Unordered list
18 -
```

Heading 1

Heading 2

Heading 3

www.google.es – clickable links (URL)

[x.cfg](#) – clickable links (open local files)



Driven by
Experience™

Lists

1. Ordered list
2.
 - Unordered list
 -

Markdown examples II

```
● ● ●  
1  
2 ## Texts  
3  
4 **bold text**  
5  
6 *italicized text*  
7  
8 ~~Strikethrough~~  
9  
10 - [x] Write the press release  
11 - [ ] Update the website  
12 - [ ] Contact the media  
13  
14 ## Tables  
15  
16  
17 Left | Right | Center  
18 :---|-----:|:-----:  
19 A1 | B1 | C1  
20 A2 | B2 | C2  
21 A3 | B3 | C3  
22
```

Texts

bold text

italicized text

Strikethrough

- Write the press release
- Update the website
- Contact the media

Tables

Left	Right	Center
A1	B1	C1
A2	B2	C2
A3	B3	C3

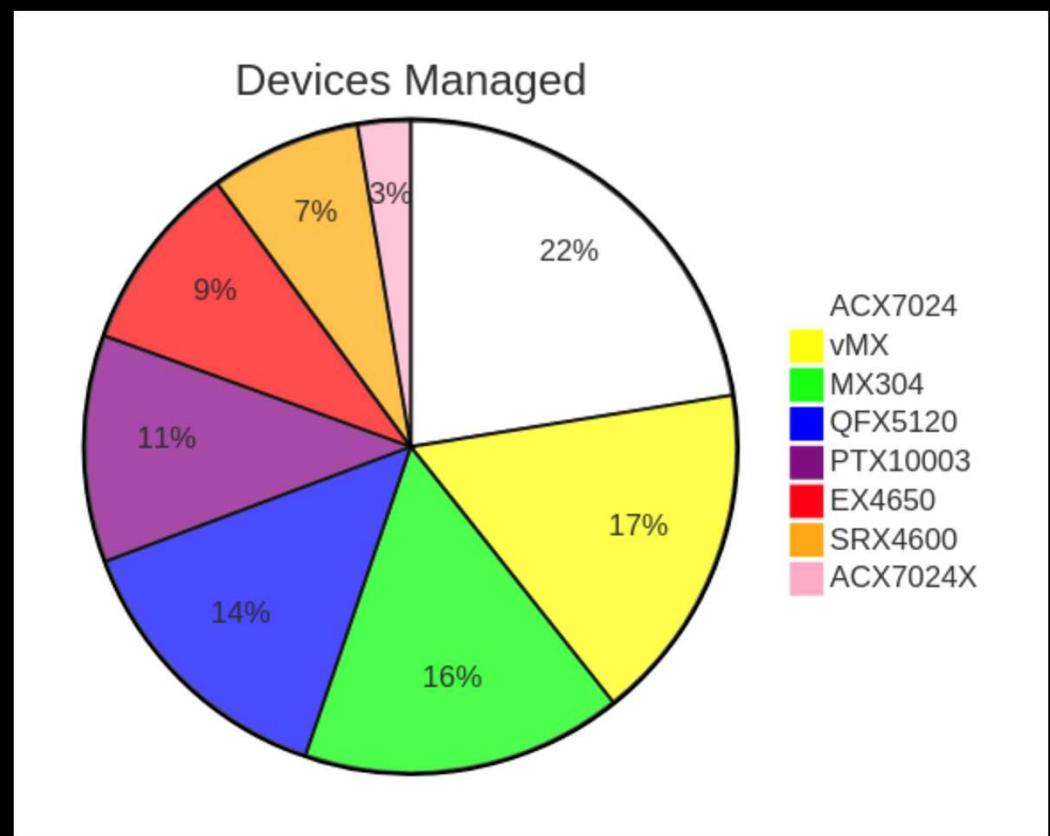
Mermaid

- Diagrams As Code
- Simple
- Dynamic
- Easy to learn, adopt...
- Version control friendly
- Live editors
- <https://mermaid.js.org/>



Mermaid example

```
1  ````mermaid
2  %%{init: {'theme': 'base',
3  'themeVariables': { 'pie1': '#FFFFFF',
4  'pie2': '#FFFF00',
5  'pie3': '#00FF00',
6  'pie4': '#0000FF',
7  'pie5': '#800080',
8  'pie6': '#ff0000',
9  'pie7': '#FFA500'}}}%%
10 pie title Devices Managed
11 "ACX7024" : 120
12 "ACX7024X" : 14
13 "MX304" : 85
14 "QFX5120" : 75
15 "EX4650" : 50
16 "SRX4600" : 40
17 "PTX10003" : 60
18 "vMX" : 90
19 ````
```



PlantUML

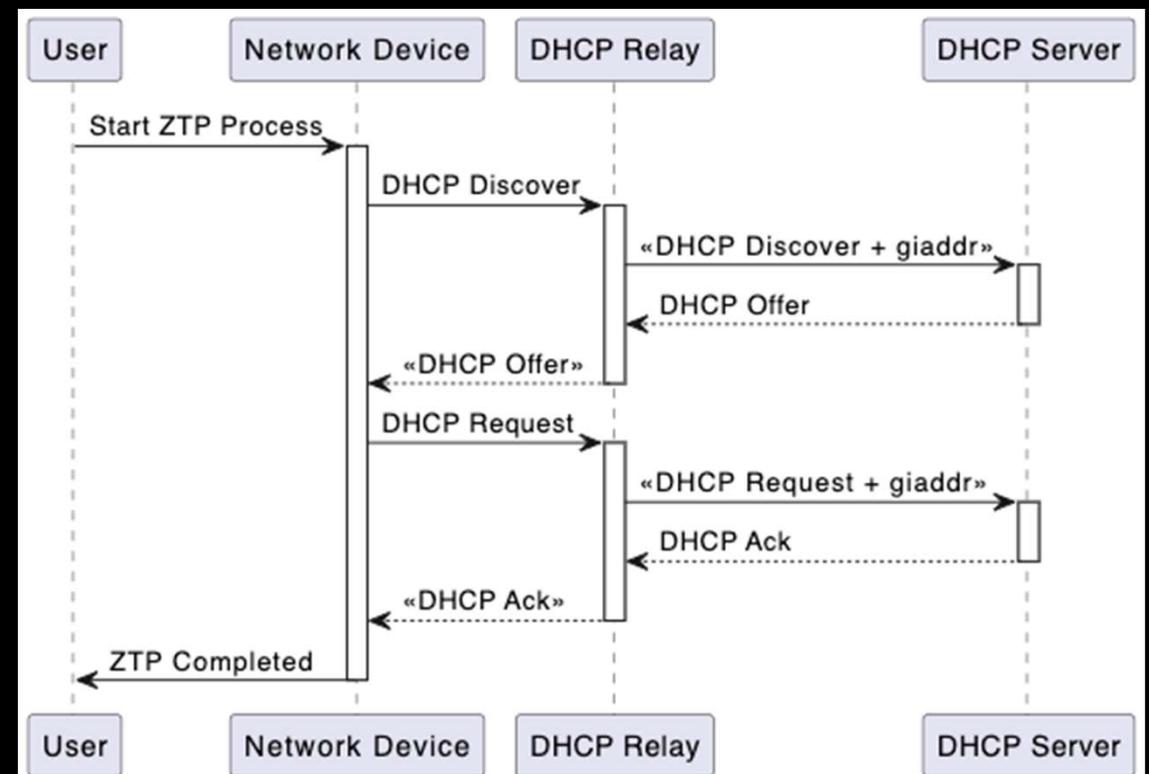
- Diagrams As Code
- Simple
- Dynamic
- Easy to learn, adopt...
- Version control friendly
- Live editors
- <https://plantuml.com/>



PlantUML example

```
```plantuml
@startuml
participant User
participant "Network Device" as ND
participant "DHCP Relay" as PS
participant "DHCP Server" as CM

User --> ND: Start ZTP Process
activate ND
ND --> PS: DHCP Discover
activate PS
PS --> CM: << DHCP Discover + giaddr>>
activate CM
CM --> PS: DHCP Offer
deactivate CM
PS --> ND: << DHCP Offer >>
deactivate PS
ND --> PS: DHCP Request
activate PS
PS --> CM: << DHCP Request + giaddr>>
activate CM
CM --> PS: DHCP Ack
deactivate CM
PS --> ND: << DHCP Ack >>
deactivate PS
ND --> User: ZTP Completed
deactivate ND
@enduml
````
```



Jinja2

- Templating engine
- Templates rendering:
 - variables {{ }}
 - logic { % ... % } (Loops and conditionals)
- Any text artifacts
- Efficiency
- Reusability



Jinja2 examples I

```
{% for interface in interfaces %}
set interfaces {{ interface.name }} description "{{ interface.description }}"
set interfaces {{ interface.name }} unit {{ interface.unit }} vlan-id {{ interface.vlan_id }}
set interfaces {{ interface.name }} unit {{ interface.unit }} family inet address {{ interface.ipv4_address }}
{% endfor %}
```

```
        set interfaces ge-0/0/0 description "Uplink to Core"
        set interfaces ge-0/0/0 unit 0 vlan-id 100
        set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/24

        set interfaces ge-0/0/1 description "LAN interface"
        set interfaces ge-0/0/1 unit 0 vlan-id 200
        set interfaces ge-0/0/1 unit 0 family inet address 192.168.2.1/24

        set interfaces ge-0/0/2 description "WAN interface"
        set interfaces ge-0/0/2 unit 0 vlan-id 300
        set interfaces ge-0/0/2 unit 0 family inet address 192.168.3.1/24
```

Jinja2 examples II

```
● ● ●  
1  <interfaces>  
2  {% for interface in interfaces -%}  
3      <interface>  
4          <name>{{ interface.name }}</name>  
5          <description>{{ interface.description }}</description>  
6          <unit>  
7              <id>{{ interface.unit }}</id>  
8              <vlan-id>{{ interface.vlan_id }}</vlan-id>  
9              <ipv4>{{ interface.ipv4_address }}</ipv4>  
10             {% if interface.ipv6_address -%}  
11                 <ipv6>{{ interface.ipv6_address }}</ipv6>  
12                 {% endif -%}  
13             </unit>  
14         </interface>  
15     {% endfor -%}  
16 </interfaces>
```

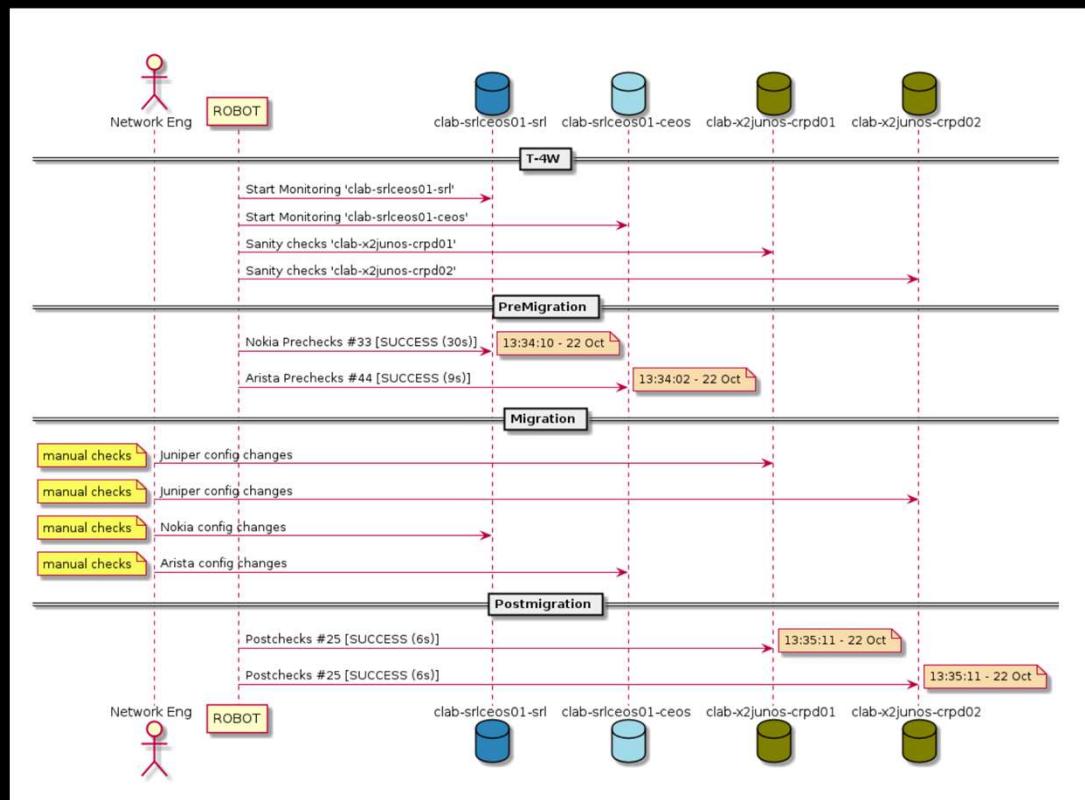
Jinja2 examples III



Jinja2 examples III

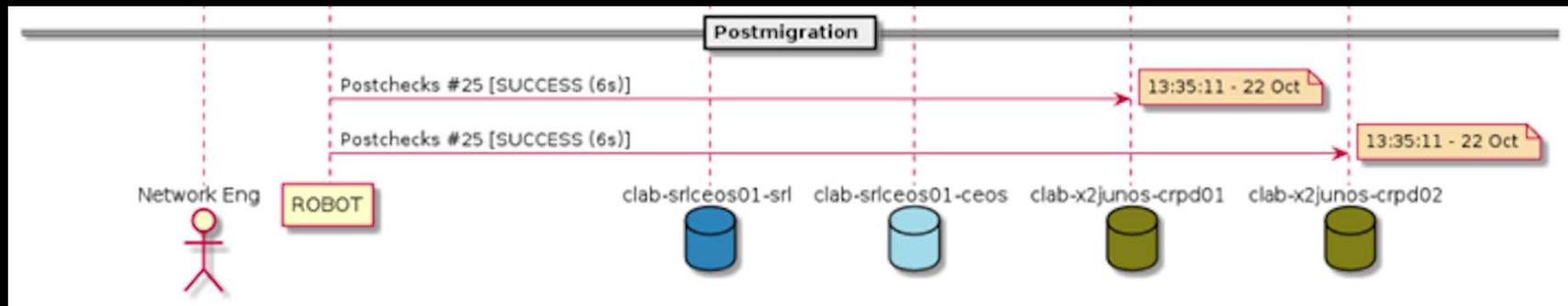
```

1  ````plantuml
2  @startuml
3
4  actor "Network Eng" as ne
5  participant ROBOT
6  database "clab-srlceos01-srl" #steelblue
7  database "clab-srlceos01-ceos" #lightblue
8  database "clab-x2junos-crp01" #olive
9  database "clab-x2junos-crp02" #olive
10
11 == T-4W ==
12
13 ROBOT -> "clab-srlceos01-srl": Start Monitoring 'clab-srlceos01-srl'
14 ROBOT -> "clab-srlceos01-ceos": Start Monitoring 'clab-srlceos01-ceos'
15 ROBOT -> "clab-x2junos-crp01": Sanity checks 'clab-x2junos-crp01'
16 ROBOT -> "clab-x2junos-crp02": Sanity checks 'clab-x2junos-crp02'
17
18 == PreMigration ==
19
20 ROBOT -> "clab-srlceos01-srl": Nokia Prechecks #{{ tcnrsl.matches.0.number }} [{{ tcnrslj_result.matches.0.result }}]
21 note right #heat: {{ 'H:M:S - %d %b' | strftime(tcnrsl.timestamp.matches.0.timestamp[-3]) }}
22 ROBOT -> "clab-srlceos01-ceos": Arista Prechecks #{{ tnceos.matches.0.number }} [{{ tnceosj_result.matches.0.result }}]
23 note right #heat: {{ 'H:M:S - %d %b' | strftime(tnceosj_timestamp.matches.0.timestamp[-3]) }}
24
25 == Migration ==
26
27 ne -> "clab-x2junos-crp01" : Juniper config changes
28 note left: manual checks
29 ne -> "clab-x2junos-crp02" : Juniper config changes
30 note left: manual checks
31 ne -> "clab-srlceos01-srl" : Nokia config changes
32 note left: manual checks
33 ne -> "clab-srlceos01-ceos" : Arista config changes
34 note left: manual checks
35
36 == Postmigration ==
37
38 ROBOT -> "clab-x2junos-crp01": Postchecks #{{ tcnjpr1.matches.0.number }} [{{ tcnjpr1j_result.matches.0.result }}]
39 note right #heat: {{ 'H:M:S - %d %b' | strftime(tcnjpr1.timestamp.matches.0.timestamp[-3]) }}
40
41 ROBOT -> "clab-x2junos-crp02": Postchecks #{{ tcnjpr2.matches.0.number }} [{{ tcnjpr2j_result.matches.0.result }}]
42 note right #heat: {{ 'H:M:S - %d %b' | strftime(tcnjpr2.timestamp.matches.0.timestamp[-3]) }}
43
44 @enduml
45 ````
```



Jinja2 examples III - zoom

```
36 == Postmigration ==
37
38 ROBOT -> "clab-x2junos-crpd01": Postchecks #{{ tcnjnpr1.matches.0.number }} [{{ tcnjnpr1_result.matches.0.result }}]
39 note right #wheat: {{ '%H:%M:%S - %d %b' | strftime(tcnjnprj_timestamp.matches.0.timestamp[:-3]) }}
40
41 ROBOT -> "clab-x2junos-crpd02": Postchecks #{{ tcnjnpr2.matches.0.number }} [{{ tcnjnpr2_result.matches.0.result }}]
42 note right #wheat: {{ '%H:%M:%S - %d %b' | strftime(tcnjnprj_timestamp.matches.0.timestamp[:-3]) }}
43
44 @enduml
45 ````
```



Jinja2 examples III

● ● ● Data

```
1 actor_name = "naf"  
2 nodes = ["node1", "node2"]
```

{ 1x actor
 2x nodes

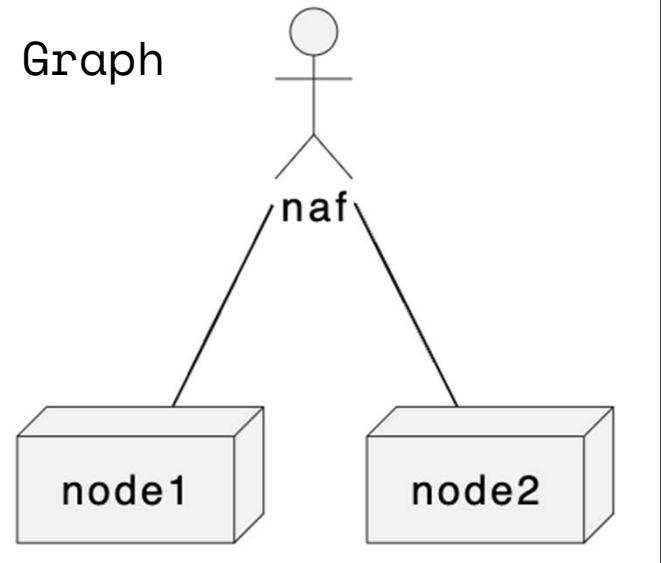
● ● ● Template

```
1 ```plantuml  
2 @startuml  
3 actor {{ actor_name }}  
4 {% for node in nodes -%}  
5 node {{ node }}  
6 {% endfor %}  
7 {% for node in nodes -%}  
8 {{ actor_name }} -- {{ node }}  
9 {% endfor -%}  
10 @enduml  
11 ```
```

● ● ● Code

```
1 ```plantuml  
2 @startuml  
3 actor naf  
4 node node1  
5 node node2  
6  
7 naf -- node1  
8 naf -- node2  
9 @enduml  
10```
```

Graph



Jinja2 examples III



Data

```
1 actor_name = "naf"  
2 nodes = ["node{}".format(i) for i in range(1, 6)]
```

{
 1x actor
 5x nodes



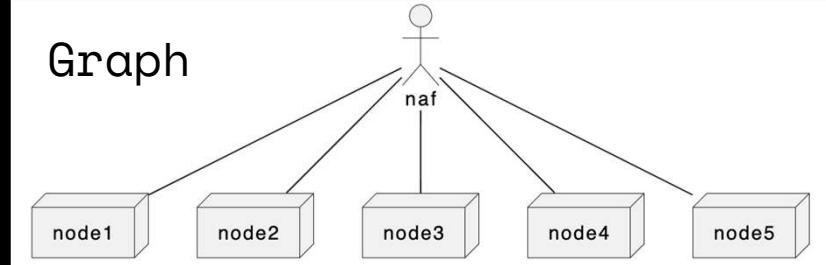
Template

```
1 ```plantuml  
2 @startuml  
3 actor {{ actor_name }}  
4 {% for node in nodes -%}  
5 node {{ node }}  
6 {% endfor %}  
7 {% for node in nodes -%}  
8 {{ actor_name }} -- {{ node }}  
9 {% endfor -%}  
10 @enduml  
11 ```
```

Code

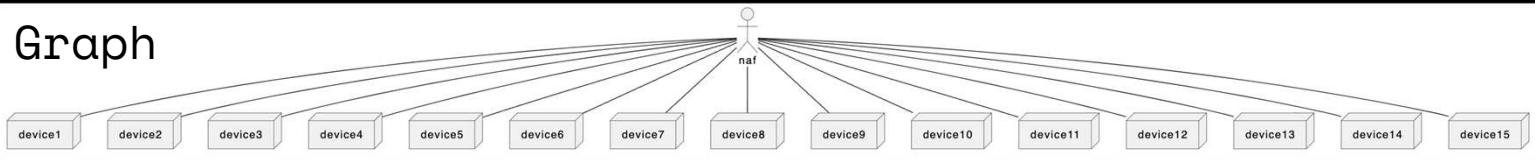
```
1 ```plantuml  
2 @startuml  
3 actor naf  
4 node node1  
5 node node2  
6 node node3  
7 node node4  
8 node node5  
9  
10 naf -- node1  
11 naf -- node2  
12 naf -- node3  
13 naf -- node4  
14 naf -- node5  
15 @enduml  
16 ```
```

Graph



Jinja2 examples III

Graph



Template

```
1 ````plantuml
2 @startuml
3 actor {{ actor_name }}
4 {% for node in nodes -%}
5 node {{ node }}
6 {% endfor %}
7 {% for node in nodes -%}
8 {{ actor_name }} -- {{ node }}
9 {% endfor %}
10 @enduml
11 ````
```



Data

```
1 actor_name = "naf"
2 nodes = ["device{}".format(i) for i in range(1, 16)]
```

{ 1x actor
15x nodes

● ○ ● Code

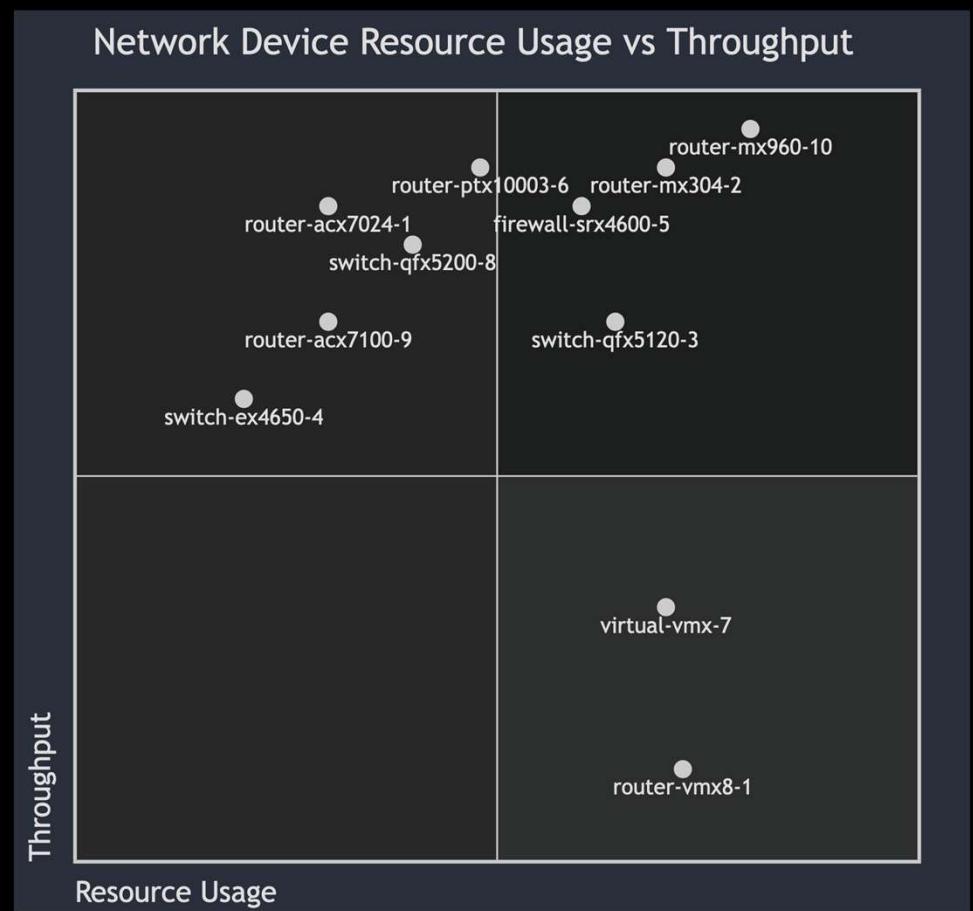
```
1 ````plantuml
2 @startuml
3 actor naf
4 node device1
5 node device2
6 node device3
7 node device4
8 node device5
9 node device6
10 node device7
11 node device8
12 node device9
13 node device10
14 node device11
15 node device12
16 node device13
17 node device14
18 node device15
19
20 naf -- device1
21 naf -- device2
22 naf -- device3
23 naf -- device4
24 naf -- device5
25 naf -- device6
26 naf -- device7
27 naf -- device8
28 naf -- device9
29 naf -- device10
30 naf -- device11
31 naf -- device12
32 naf -- device13
33 naf -- device14
34 naf -- device15
35 @enduml
36 ````
```

Jinja2 examples IV

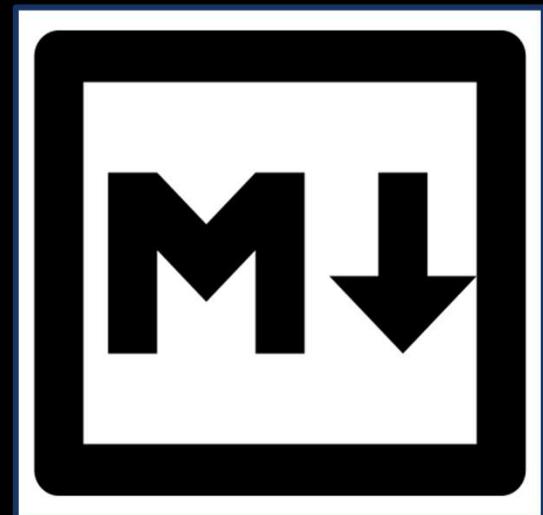


Jinja2 examples IV

```
1  ````mermaid
2  quadrantChart
3  title {{ title }}
4  x-axis {{ x_axis }}
5  y-axis {{ y_axis }}
6  {% for device, coords in devices.items() -%}
7  "{{ device }}": [{{ coords[0] }}, {{ coords[1] }}]
8  {% endfor -%}
9  ````
```



Jinja2 examples V



Jinja2 examples V

```
 1 # {{ title }}
 2
 3 **Author**: {{ author }}
 4
 5 ---
 6
 7 {% for section in sections %}
 8 ## {{ section.name }}
 9
10 {{ section.description }}
11
12 {% if section.items %}
13 ### Items:
14
15 | Name | Description |
16 | ---- | ----- |
17 {% for item in section.items %}
18 | **{{ item.name }}** | {{ item.description }} |
19 {% endfor %}
20 {% endif %}
21 {% endfor %}
```

Network Configuration Report

Author: John Doe

System Information

Basic system info of the network device.

Items:

| Name | Description |
|------------|--------------|
| Hostname | MX960-Router |
| OS Version | JunOS 20.4R3 |

Interfaces

Overview of configured interfaces.

Items:

| Name | Description |
|----------|---------------------------------|
| ge-0/0/0 | Uplink to Core - 192.168.1.1/24 |
| ge-0/0/1 | LAN Interface - 192.168.2.1/24 |

Jinja2 examples VI





Jinja2 examples VI

● ● ●

```
1 ...
2 "query": "SELECT mean(value) FROM \"jm-abc-pe-2a.TenGigE0/2/0/9.input-rate\" where $timeFilter group by time(5m)",
3 "query": "SELECT mean(value) FROM \"jm-abc-pe-2b.TenGigE0/2/0/7.input-rate\" where $timeFilter group by time(5m)",
4 "query": "SELECT mean(value) FROM \"jm-xyz-pe-2a.TenGigE0/5/0/5.input-rate\" where $timeFilter group by time(5m)",
5 "query": "SELECT mean(value) FROM \"jm-xyz-pe-2b.TenGigE0/5/0/6.input-rate\" where $timeFilter group by time(5m)",
6 ...
```

● ● ●

```
1 ...
2 "query": "SELECT mean(value) FROM \"{{ sd }}.{{ s_port }}.input-rate\" where $timeFilter group by time(5m)",
3 ...
```

Ansible

- Automation tool
- Human readable (YAML syntax)
- Works great with jinja2 templates
- Easily integrated
- Built-in modules
- Extensible
- Cross-platform



Ansible examples I

```
1 - name: Retrieve sanity_checks job last build ({{ sanity_checks_jlbn.matches.0.number }}) details.
2   uri: <-->
3     url: https://{{ jenkins.user }}:{{ jenkins.password }}@{{ jenkins.host }}:{{ jenkins.port }}/job/sanity_checks/{{ sanity_checks_jlbn.matches.0.number }}/api/xml? tree
4   method: GET
5   force_basic_auth: yes
6   validate_certs: no
7   status_code: 200
8   return_content: true
9   register: sanity_checks_job_details
10  tags: sanity_checks
11
12 - name: Parse sanity_checks job last build duration.
13   xml: <-->
14     xmlstring: "{{ sanity_checks_job_details.content }}"
15     xpath: //duration
16     content: text
17   register: sanity_checks_job_duration
18   tags: sanity_checks
19
20 - name: Parse sanity_checks job last build result.
21   xml: <-->
22     xmlstring: "{{ sanity_checks_job_details.content }}"
23     xpath: //result
24     content: text
25   register: sanity_checks_job_result
26   tags: sanity_checks
```

Ansible examples II

```
● ● ●  
1   - name: Creating base policy-options config  
2     template: ←  
3       src: policy-options.j2  
4       dest: "{{ tmp_dir }}/policy-options.cfg"  
5  
6   - name: Creating bridge-domains config  
7     template: ←  
8       src: bridge-domains.j2  
9       dest: "{{ tmp_dir }}/bridge-domains.cfg"  
10  
11  - name: Assembling configurations  
12    assemble: ←  
13      src: "{{ tmp_dir }}"  
14      dest: "{{ junos_conf }}"  
15      mode: "{{ conf_file_mode | default('666') }}"  
16      regexp: .+\.cfg
```

Docker

- Containerization platform
- Isolated environments (containers)
- Portability
- Efficiency
- Consistency
- Dockerfile
- DevOps



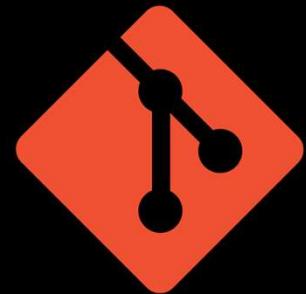
Jenkins

- Automation server
- Aprox. ~1800 plugins (robotframework, slack, gitlab, etc...)
- Customizability
- Jobs (pipelines, tasks, scripts...)
- DevOps



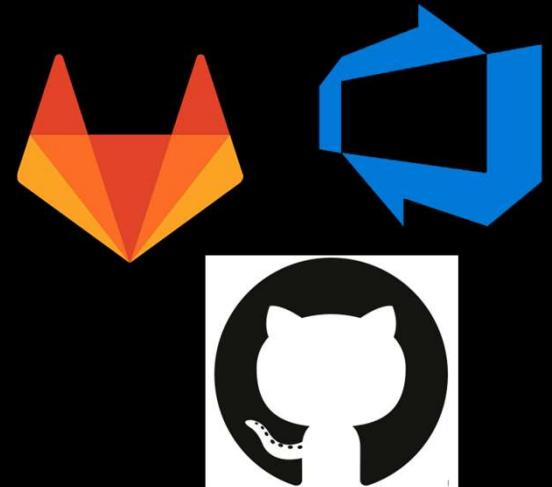
Git

- Track changes in files and directories
- Compare changes
- Different branches
- Commits
- Write your project own story



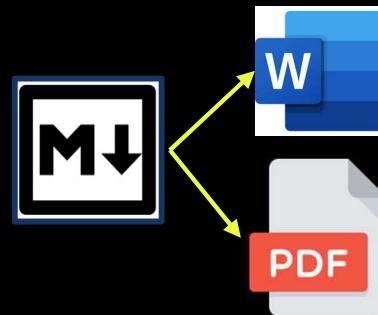
Version Control Systems

- DevOps platforms (GitLab, GitHub, Azure DevOps...)
- Git repo management
- Easily track changes (GUI)
- Easy to collaborate
- Change Management
- CI/CD
- Seamless Markdown integration



Pandoc

- Universal document converter (from ⇔ to)
- Flexibility
- Customizable
- <https://pandoc.org/>



LaTeX

- High quality technical documents
- Consistency
- Typesetting:
 - layout
 - fonts
 - ToC
 - spacing
 - alignment
 - tables
 - figures
- <https://www.latex-project.org>



LATEX



Real World Use Case I

Product/Project documentation (PD)

PD: Workflow

- *jinja2* templates (Markdown + code blocks)
- Graphs (static [Diagrams As Code] + static [pictures])
- Ansible (orchestrator)
- Data (mainly static: *group_vars*, *host_vars*)
- Triggered by a Merge request (MR)



PD: Pipeline - Stage #1

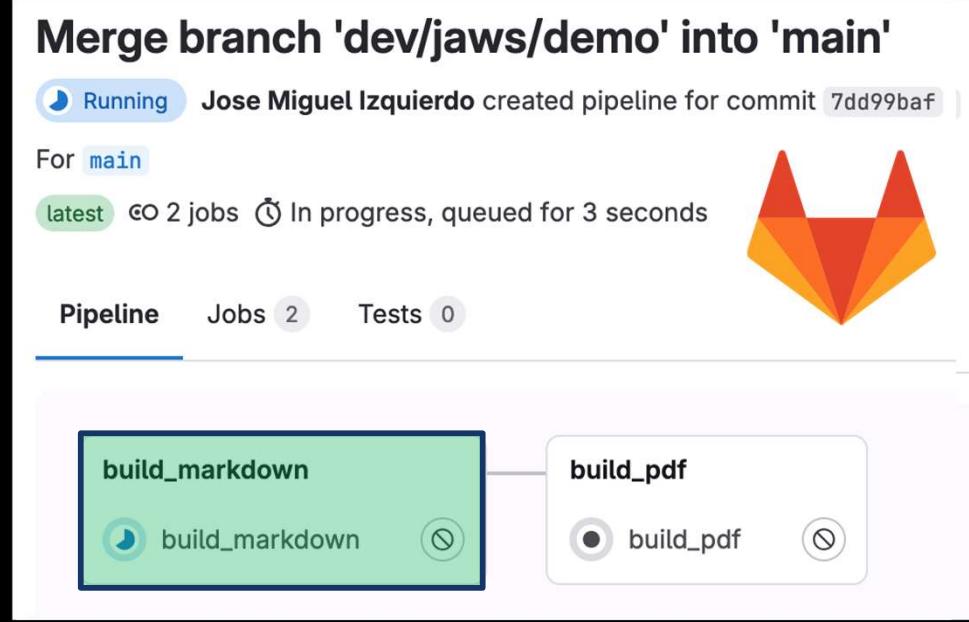
Merge branch 'dev/jaws/demo' into 'main'

Running Jose Miguel Izquierdo created pipeline for commit 7dd99ba... |

For main

latest 2 jobs In progress, queued for 3 seconds

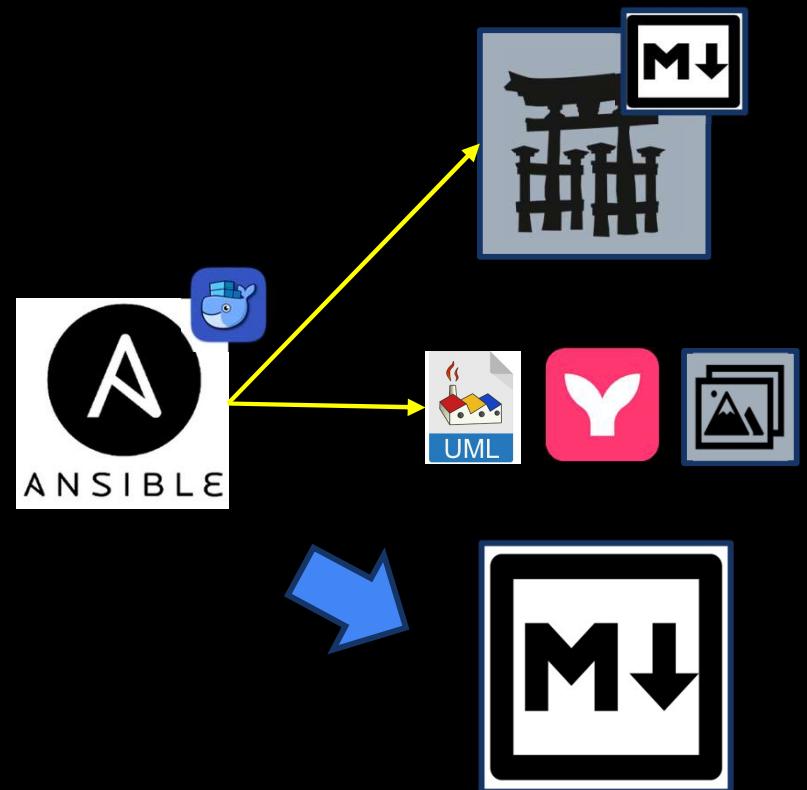
Pipeline Jobs 2 Tests 0



```
graph LR; build_markdown[build_markdown] --> build_pdf[build_pdf]
```

The diagram shows a GitLab CI pipeline. It starts with a green box labeled "build_markdown" containing a "build_markdown" job. An arrow points from this box to a white box labeled "build_pdf" containing a "build_pdf" job.

.gitlab-ci.yml



PD: Pipeline - Stage #2

Merge branch 'dev/jaws/demo' into 'main'

Running Jose Miguel Izquierdo created pipeline for commit 7dd99ba... |

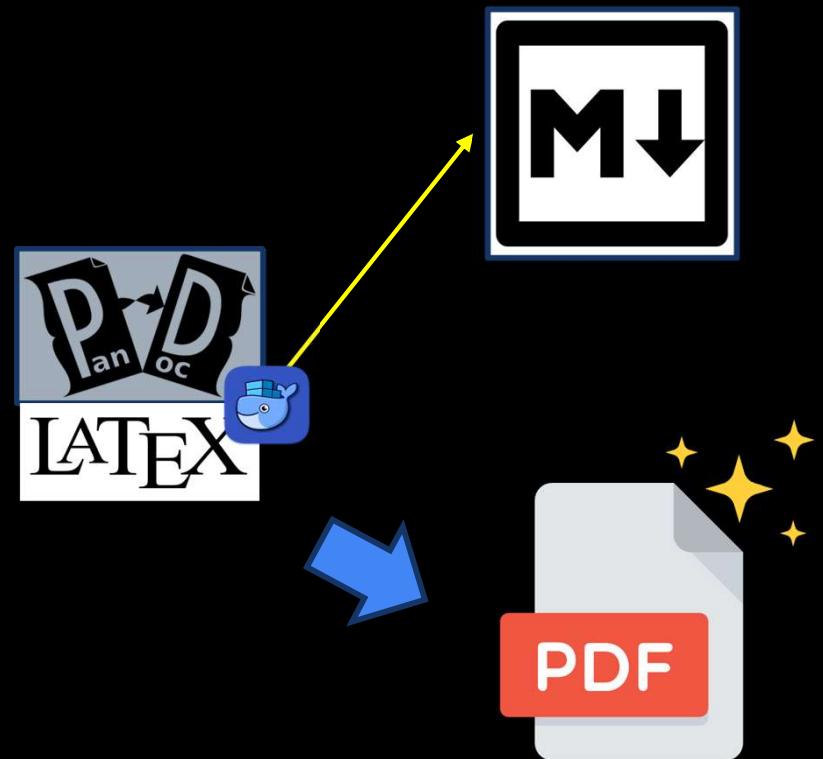
For main

latest 2 jobs In progress, queued for 3 seconds

Pipeline Jobs 2 Tests 0

The screenshot shows a GitLab CI pipeline interface. At the top, it says "Merge branch 'dev/jaws/demo' into 'main'" with a "Running" status and a commit hash. Below that, it says "For main". Under "latest", it shows "2 jobs" with "In progress, queued for 3 seconds". There are tabs for "Pipeline", "Jobs 2", and "Tests 0". The pipeline diagram shows two stages: "build_markdown" (green box) and "build_pdf" (blue box). An arrow points from "build_markdown" to "build_pdf". Each stage has a green checkmark icon and a circular refresh icon.

.gitlab-ci.yml



PD: Pipeline - Artifacts

Merge branch 'dev/jaws/demo' into 'main'

Passed Jose Miguel Izquierdo created pipeline for commit 7dd99baf

For main

latest 2 jobs 54 seconds, queued for 3 seconds

Pipeline Jobs 2 Tests 0

build_markdown build_pdf

.gitlab-ci.yml

Job artifacts ? These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Keep Download Browse

Commit 7dd99baf Merge branch 'dev/jaws/demo' into 'main'

Pipeline #1233 Passed for main

build_pdf

Related jobs → build_pdf

PD: Documentation



Real World Use Case II

Night intervention report (NR)

NR: Workflow

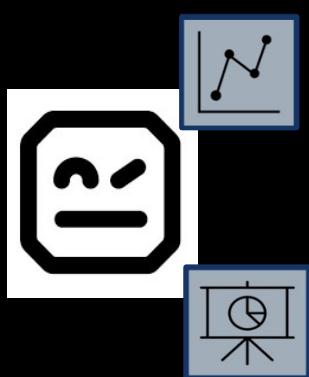
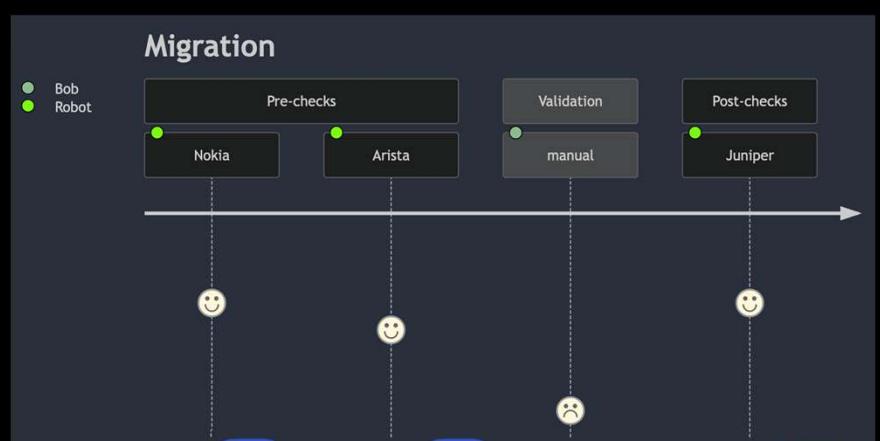
- *jinja2* templates (Markdown + code blocks)
- Graphs (dynamic [diagrams As Code] + static [pictures])
- Ansible (orchestrator)
- Data (3rd party systems: Grafana, Robotframework)
- Artifacts



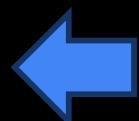
NR: Tests

The Jenkins dashboard displays a list of builds with columns for Status, Workstation, Name, Last Success, Last Failure, Last Duration, Built On, and F. The builds listed are dynamic_report, test_lab_arista_zeos, test_lab_juniper_cruds, and test_lab_nokia_slinux. Each build has a green checkmark icon and a Jenkins logo.

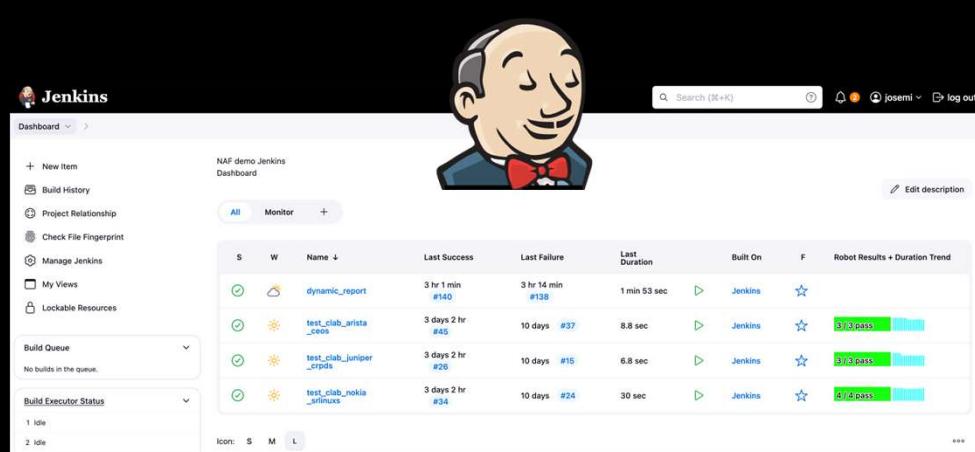
| S | W | Name | Last Success | Last Failure | Last Duration | Built On | F |
|---|-------|------------------------|-----------------|------------------|---------------|----------|---|
| ✓ | cloud | dynamic_report | 3 hr 1 min #140 | 3 hr 14 min #138 | 1 min 53 sec | Jenkins | ☆ |
| ✓ | sun | test_lab_arista_zeos | 3 days 2 hr #45 | 10 days #37 | 8.8 sec | Jenkins | ☆ |
| ✓ | sun | test_lab_juniper_cruds | 3 days 2 hr #26 | 10 days #15 | 6.8 sec | Jenkins | ☆ |
| ✓ | sun | test_lab_nokia_slinux | 3 days 2 hr #34 | 10 days #24 | 30 sec | Jenkins | ☆ |



- Reports...
- Logs...
- Operational command outputs...



NR: Report



Jenkins Dashboard

NAF demo Jenkins Dashboard

Search (K)

josemi log out

New Item

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Lockable Resources

Build Queue

No builds in the queue.

Build Executor Status

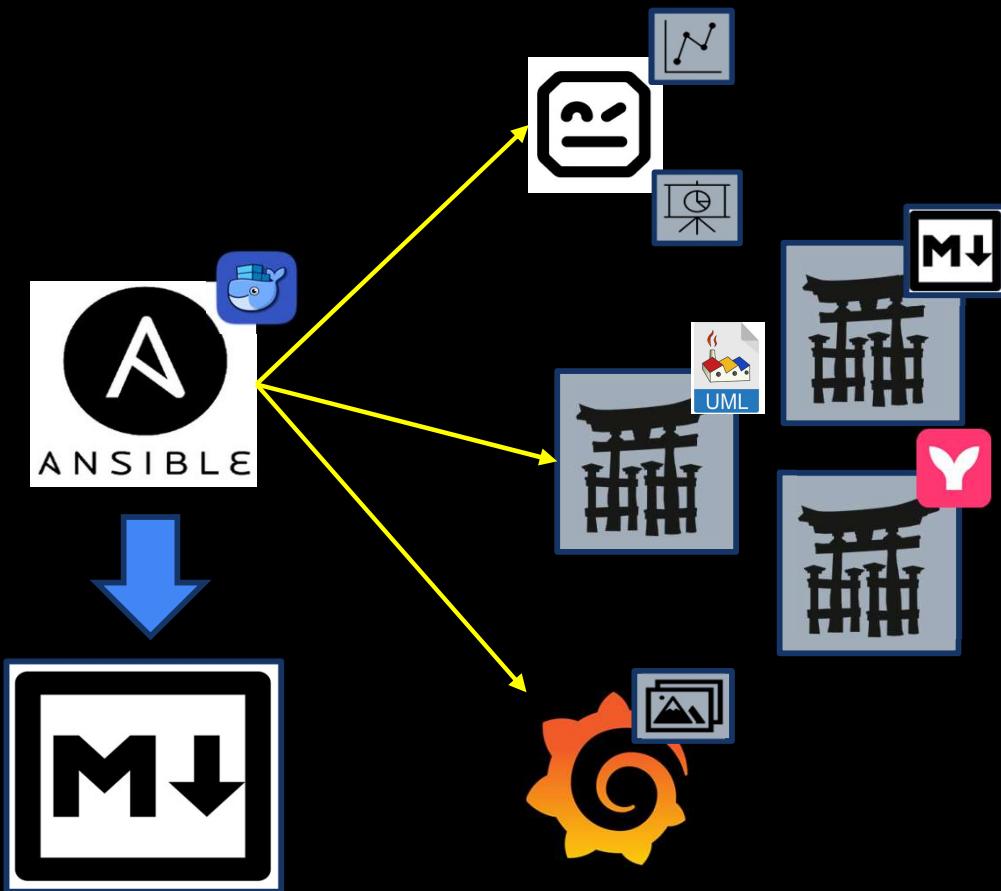
1 idle

2 idle

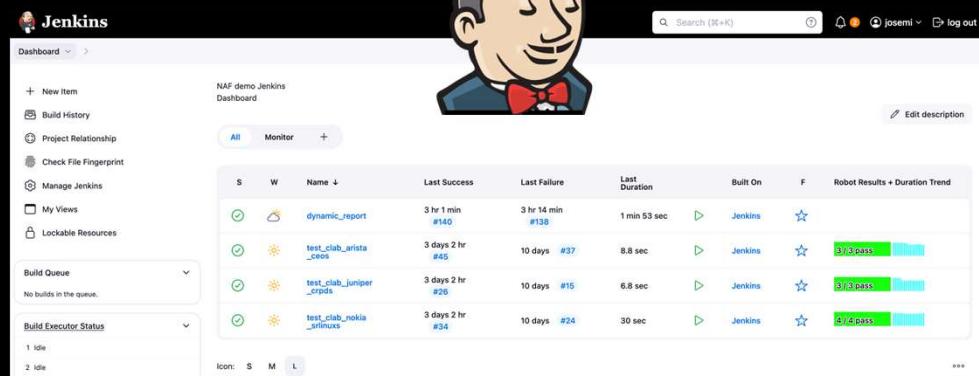
Last Success Last Failure Last Duration Built On F Robot Results + Duration Trend

| S | W | Name | Last Success | Last Failure | Last Duration | Built On | F | Robot Results + Duration Trend |
|-------|-------|------------------------|-----------------|------------------|---------------|----------|---|--------------------------------|
| green | cloud | dynamic_report | 3 hr 1 min #140 | 3 hr 14 min #138 | 1 min 53 sec | Jenkins | ☆ | |
| green | sun | test_ciab_arista_ceos | 3 days 2 hr #45 | 10 days #37 | 8.8 sec | Jenkins | ☆ | |
| green | sun | test_ciab_juniper_crpd | 3 days 2 hr #26 | 10 days #15 | 6.8 sec | Jenkins | ☆ | |
| green | sun | test_ciab_nokia_slimux | 3 days 2 hr #34 | 10 days #24 | 30 sec | Jenkins | ☆ | |

Icon: S M L

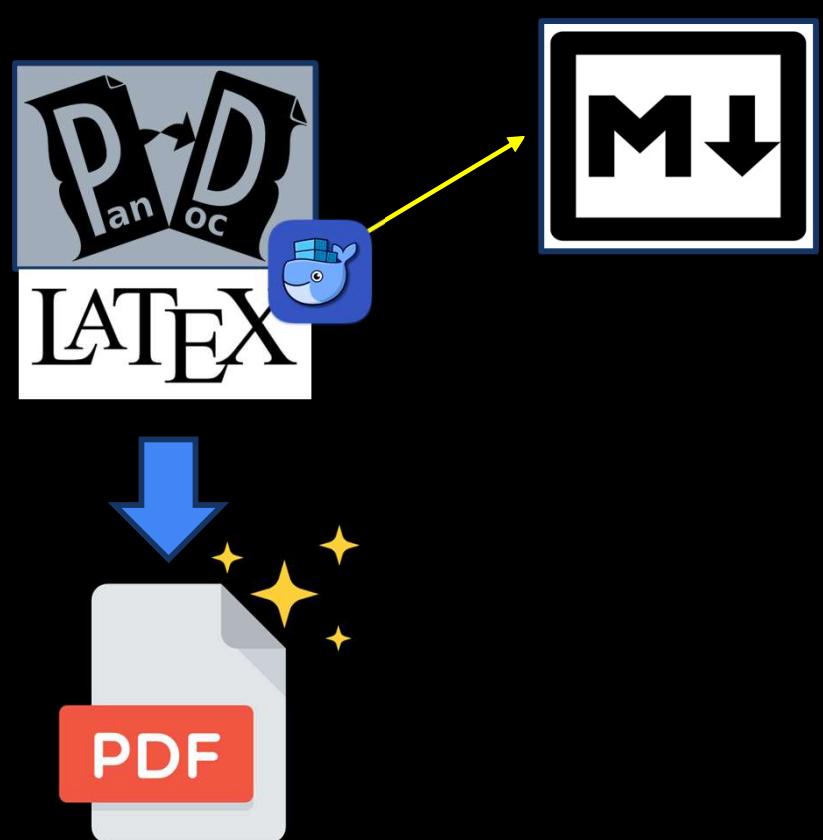


NR: PDF Report

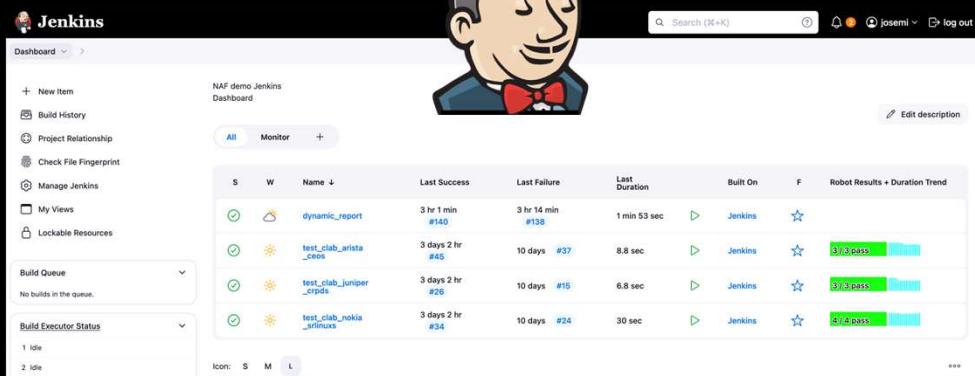


The Jenkins dashboard displays a list of builds under the "Build History" section. The builds shown are:

| S | W | Name | Last Success | Last Failure | Last Duration | Built On | F | Robot Results + Duration Trend |
|---|---|-------------------------|-----------------|------------------|---------------|----------|---|-----------------------------------|
| ✓ | ✓ | dynamic_report | 3 hr 1 min #140 | 3 hr 14 min #138 | 1 min 53 sec | Jenkins | ☆ | <div style="width: 100%;"> </div> |
| ✓ | ✓ | test_ciab_arista_ceos | 3 days 2 hr #45 | 10 days #37 | 8.8 sec | Jenkins | ☆ | <div style="width: 100%;"> </div> |
| ✓ | ✓ | test_ciab_juniper_crpd | 3 days 2 hr #26 | 10 days #15 | 6.8 sec | Jenkins | ☆ | <div style="width: 100%;"> </div> |
| ✓ | ✓ | test_ciab_nokia_slimuxs | 3 days 2 hr #34 | 10 days #24 | 30 sec | Jenkins | ☆ | <div style="width: 100%;"> </div> |

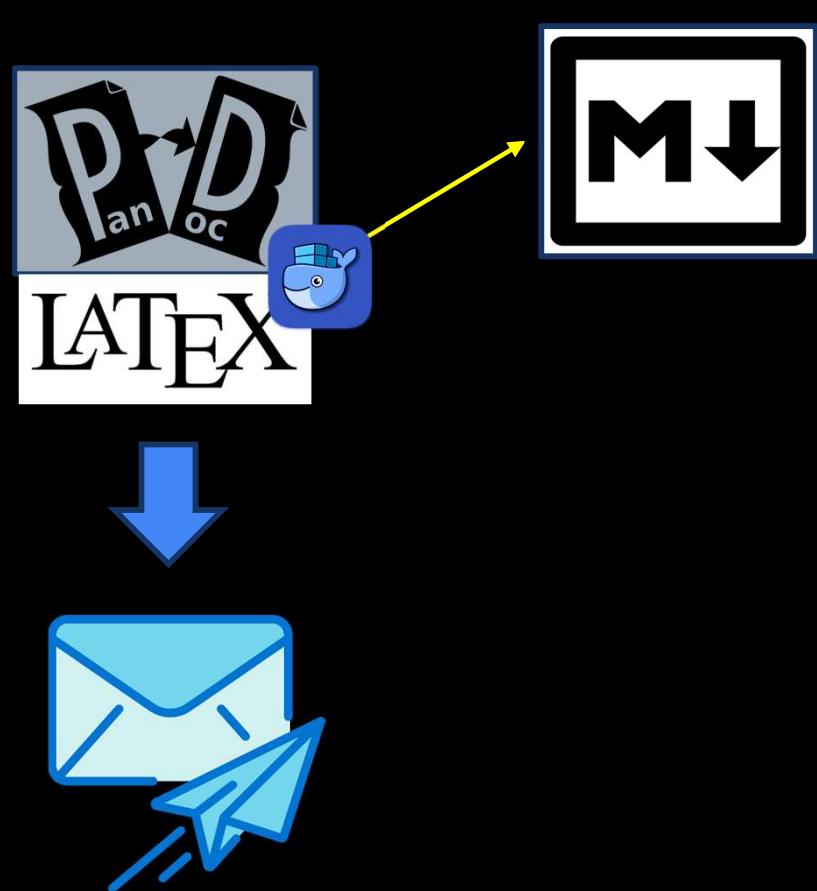


NR: PDF Report



The Jenkins dashboard displays the following information:

| S | W | Name | Last Success | Last Failure | Last Duration | Built On | F | Robot Results + Duration Trend |
|-------|-------|------------------------|-----------------|------------------|---------------|----------|---|--------------------------------|
| green | cloud | dynamic_report | 3 hr 1 min #140 | 3 hr 14 min #138 | 1 min 53 sec | Jenkins | ☆ | |
| green | sun | test_ciab_arista_ceos | 3 days 2 hr #45 | 10 days #37 | 8.8 sec | Jenkins | ☆ | |
| green | sun | test_ciab_juniper_crpd | 3 days 2 hr #26 | 10 days #15 | 6.8 sec | Jenkins | ☆ | |
| green | sun | test_ciab_nokia_slimux | 3 days 2 hr #34 | 10 days #24 | 30 sec | Jenkins | ☆ | |



NR: Documentation

The collage consists of several screenshots from the Juniper Network Automation Forum (NAF) interface, illustrating various documentation and monitoring tools:

- Juniper Documentation**: A vertical sidebar on the left showing a table of contents with sections like "Introduction", "Planning", "Deployment", "Monitoring", "Post-migration", "Console output", "Report", "Nokia PreCheck", "PostCheck", "Command", and "Nokia".
- Table of Contents**: A screenshot of the NAF table of contents.
- IntelliDoc**: A screenshot of the IntelliDoc tool, which generates documentation from code comments.
- Monitoring**: A screenshot of the Monitoring section showing network activity monitoring and KPIs.
- Post-migration**: A screenshot of the Post-migration section showing network configuration and status.
- Console output**: A screenshot of the Jenkins console output for a build named [EnvInject1] - 3.
- Report**: A screenshot of the Report section showing a Jenkins job named "PreChecks" which gathers useful information.
- Nokia PreCheck**: A screenshot of the Nokia PreCheck tool showing a list of manual checks for Juniper, Nokia, and Arista configurations.
- PostCheck**: A screenshot of the PostCheck tool showing a list of manual checks for Juniper, Nokia, and Arista configurations.
- Command**: A screenshot of the Command section showing a list of commands run by Jenkins.
- Nokia**: A screenshot of the Nokia section showing a list of outputs.
- JNPR cR Summary**: A screenshot of the JNPR cR Summary report showing a table of hardware details.
- Test Status**: A screenshot of the Test Status report showing a table of test results.
- Test Details**: A screenshot of the Test Details report showing a table of detailed test results.
- Console**: A screenshot of the Console section showing a Jenkins log entry for a docker run command.
- Reminder**: A screenshot of a Jenkins reminder message: "I did all the job, but if anything is not as you expected, then you can complain to jizquierdo@juniper.net".
- Figure 4: Jenkins**: An illustration of a smiling Jenkins head.
- Network Automation Forum demo.code**: A screenshot of a Jenkins job named "Network Automation Forum demo.code".
- Figure 5: NAF**: A screenshot of the NAF logo.

Summary

- Identify your use case goals:
 - repeatable
 - consistent look & feel
 - etc...
- Select your tools (recommendation start with the first 5)
- No need to be a coding expert
- Define your workflow integrations

The end

Template

```
1
2  {% for job in cast_and_crew -%}
3    {{ "{:<16}".format(job) }} - Jose Miguel Izquierdo
4  {% endfor -%}
5
6  {% if thanks %} 
7  Special thanks to:
8  {% for ack in thanks.acknowledgements %}
9    - {{ ack }}!
10 {% endfor %}
11 All - {{ thanks.msg }}
12 {% endif -%}
13
```

Data

```
1  cast_and_crew:
2    - "Casting"
3    - "Slides writer"
4    - "Designer"
5    - "Makeup"
6    - "Production"
7    - "Presenter"
8
9  thanks:
10   acknowledgements:
11     - "Frank Reimer"
12   msg: "THANK YOU VERY MUCH!!!!"
```

The end



Code

1
2 Casting – Jose Miguel Izquierdo
3 Slides writer – Jose Miguel Izquierdo
4 Designer – Jose Miguel Izquierdo
5 Makeup – Jose Miguel Izquierdo
6 Production – Jose Miguel Izquierdo
7 Presenter – Jose Miguel Izquierdo
8
9 Special thanks to:
10 – Frank Reimer!
12
13 All – THANK YOU VERY MUCH!!!!
14