NFE Networking For Everyone

YAML

YAML. Ain’t Markup Language

- язык сериализации данных
- текстовый формат для записи данных (как CSV, JSON, XML и пр.)
- легко воспринимается для чтения человеком (*)
- широкое распространение (ansible, k8s, nornir, и пр.) для описания настроек

```
*  YAML is a human-friendly data serialization
   language for all programming languages.
```

XML vs YAML vs JSON

srv.XML

```xml
<servers>
  <server>
    <name color="red">storage-001</name>
    <owner>dc-team</owner>
    <status>active</status>
  </server>
  <server>
    <name>zabbix-001</name>
    <owner>lan-team</owner>
    <status>offline</status>
  </server>
</servers>
```

srv.YAML

```yaml
servers:
  server:
    - name:
        "#text": "storage-001"
        "@color": "red"
      owner: "dc-team"
      status: "active"
    - name: "zabbix-001"
      owner: "lan-team"
      status: "offline"
```

srv.JSON

```json
{
    "servers": {
        "server": [
            {
                "name": {
                    "@color": "red",
                    "#text": "storage-001"
                },
                "owner": "dc-team",
                "status": "active"
            },
            {
                "name": "zabbix-001",
                "owner": "lan-team",
                "status": "offline"
            }
        ]
    }
}
```

```
with open("./srv.xml", "r") as file_:
    dict_xml = xmltodict.parse(file_.read())

with open("./srv.json", "r") as file_:
    dict_json = json.load(file_)

with open("./srv.yaml", "r") as file_:
    dict_yaml = yaml.safe_load(file_)

dict_json == dict_xml == dict_yaml
True
```

YAML: structures

key-value

```yaml
fruit: apple
liquid: water
meat: beef
```

array (list)

```yaml
fruits:
  - apple
  - banana
liquids:
  - water
  - soda
Meats:
  - beef
```

map (dictionary)

```yaml
fruits:
  - apple
  - banana
liquids:
  - name: water
    calories: 0
  - name: soda
    calories: 38
meats:
  - beef
```

```
with open("<>", "r") as _file:
    pprint(yaml.safe_load(_file))


{
   'fruit': ‘apple',
   'liquid': ‘water',
   'meat': ‘beef’,
}
```

```
with open("<>", "r") as _file:
    pprint(yaml.safe_load(_file))


{
   'fruits': ['apple', ‘banana'],
   'liquids': ['water', ‘soda'],
   ‘meats': [‘beef’],
}
```

```
with open("<>”, "r") as _file:
    pprint(yaml.safe_load(_file))

{
   'fruits': ['apple', ‘banana'],
   'liquids': [
     {'calories': 0, 'name': 'water'},
     {'calories': 38, 'name': ‘soda’},
   ],
   'meats': [‘beef’],
}
```

YAML: types

key-value

```yaml
boolean1: true
boolean2: yes

key1: value
key2: another value with spaces
key3 with spaces: "yes"

null1: null
null2: ~
null3:

number_dec: 100
number_hex: 0x123 # dec: 291
number_oct: 0123 # dec: 83
```

```
with open("<>", "r") as _file:
    pprint(yaml.safe_load(_file))

{
   'boolean1': True,
   'boolean2': True,

   'key1': 'value',
   'key2': 'another value with spaces',
   'key3 with spaces': 'yes',

   'null1': None,
   'null2': None,
   ‘null3': None,

   ‘number_dec': 100,
   'number_hex': 291,
   'number_oct': 83,
   'number_scientific': '1e+12',
}
```

YAML: spaces

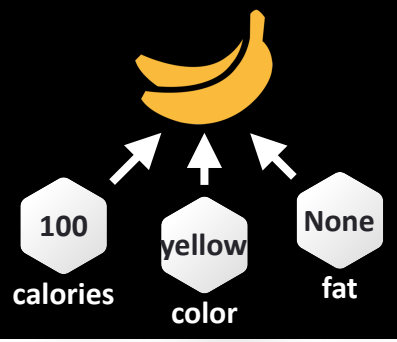
map (dictionary)

```yaml
banana:
  fat:
  calories: 100
  color: yellow
```

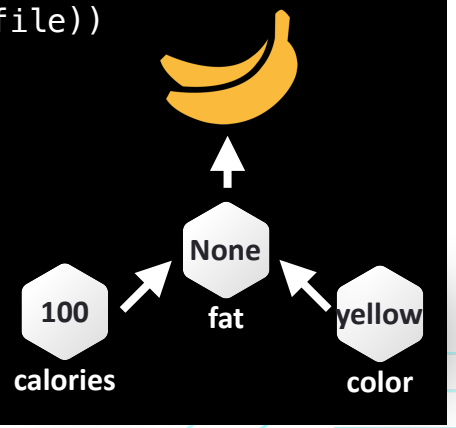
map (dictionary)

```yaml
banana:
  fat:
    calories: 100
    color: yellow
```

```
with open("<>", "r") as _file:
    pprint(yaml.safe_load(_file))


{
   'banana': {
     'calories': 100,
     'color': 'yellow',
     'fat': None,
   },
}
```




```
with open("<>", "r") as _file:
    pprint(yaml.safe_load(_file))


{
   'banana': {
     'fat': {
       'calories': 100,
       'color': 'yellow',
     },
   },
}
```



YAML: ordering

Словарь: неупорядоченная структура

```yaml
server:
  ram: 64gb
  name: db-002
  os: ubuntu
```

```
{
   'server': {
      'name': 'db-002',
      'os': 'ubuntu',
      'ram': '64gb',
   },
}
```

=

```yaml
server:
  name: db-002
  ram: 64gb
  os: ubuntu
```

```
{
   'server': {
      'name': 'db-002',
      'os': 'ubuntu',
      'ram': '64gb',
   },
}
```

Список: упорядоченная структура

```yaml
servers:
  - db-002
  - db-001
  - db-010
```

```
{
   'servers': [
      'db-002',
      'db-001',
      'db-010',
   ],
}
```

≠

```yaml
servers:
  - db-001
  - db-002
  - db-010
```

```
{
   'servers': [
      'db-001',
      'db-002',
      'db-010',
   ],
}
```

YAML: multiline

- Block Style Indicator: определяет, что делать с символами newline внутри строки.
  - literal style (`|`): сохраняет переносы строки как есть
  - folder style (`>`): переносы строки заменяются пробелами, пустые строки заменяются переносами строк.
- Block Chomping Indicator: определяет, что делать с переносами строк в конце текста.
  - clip (default): все переносы (пустые строки) заменяются одним переносом.
  - strip (`-`): все переносы отрезаются.
  - keep (`+`): все переносы сохраняются как есть.
- Indentation Indicator: количество отступов блока. Определяется автоматически, но иногда нужно задать вручную, например если текст должен начинаться с отступом (с красной строки).

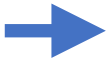
```
example: |4+\n
····Several lines of text,\n
```

YAML: multiline

```
example: |
··Текст созраняется как есть, все переносы строки\n
··остаются на месте.\n
\n
··Строки на более вложенных уровнях так же сохраняют\n
··стрктуру и переносы.\n
····Более глубокий уровень вложенности:\n
·····- список 1\n
·····- список 2\n
\n
\n
\n
```

```
with open("<>", "r") as _file:
    print("'" + yaml.safe_load(_file).get("example", "") + "'")

'Текст созраняется как есть, все переносы строки
остаются на месте.

  Строки на более вложенных уровнях так же сохраняют
  стрктуру и переносы.
    Более глубокий уровень вложенности:
      - список 1
      - список 2
'
```

```
example: >-
··Все переносы строки конвертируются\n
··в пробелы.\n
\n
··А пустые строки конвертируются в переносы строки.\n
\n
····С более глубоким уровнем вложенности работает\n
····правило literal style.\n
\n
\n
\n
```



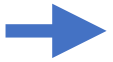
```
with open("<>", "r") as _file:
    print("'" + yaml.safe_load(_file).get("example", "") + "'")

'Все переносы строки конвертируются в пробелы.
А пустые строки конвертируются в переносы строки.

  С более глубоким уровнем вложенности работает
  правило literal style.'
```

YAML: multiline

```
example: |2+
····literal style с началом текста с красной строки.\n
\n
··Количество переносов строки в конце\n
··текста сохраняется.\n
\n
\n
\n
```

```
with open("<>", "r") as _file:
    print("'" + yaml.safe_load(_file).get("example", "") + "'")

'  literal style с началом текста с красной строки.

Количество переносов строки в конце
текста сохраняется.



'
```

```
example: plain text без использования кавычек\n
\n
··работает аналогично\n
··folder style\n
\n
\n
\n
```



```
with open("<>", "r") as file_:
    print("'" + yaml.safe_load(_file).get("example", "") + "'")

'plain text без использования кавычек
работает аналогично folder style'
```

YAML: example

```
env:
  - name: ENV_VAR
    value: >-
      some_secret_key
      my_db_password
      my_redis_password

  - name: BANNER_TOP
    value: >
      <h6 style="margin-bottom:0em">
        Это DEV окружение и все изменения будут <b style=“color:purple">удалены</b>
      </h6>
      <p>
        Подробности в чате <a href=“https://some_link" target=“_blank">chat_name</a>
      </p>
```

YAML: anchors (constants)

Функционал `anchors` позволят дублировать код

Anchor обозначается через `&`

Ссылка на значение задается через `*`

```yaml
- ubuntu: &ubuntu ubuntu-2204-lts
  ram-large: &ram-large 64Gb
  ram-small: &ram-small 8Gb

- name: server1
  os: *ubuntu
  ram: *ram-large
- name: server2
  os: *ubuntu
  ram: *ram-small
```

```
with open("<>", "r") as file_:
    pprint(yaml.safe_load(file_))

[
  {
    'ram-large': '64Gb',
    'ram-small': '8Gb',
    'ubuntu': 'ubuntu-2204-lts',
  },
  {
    'name': 'server1',
    'os': 'ubuntu-2204-lts',
    'ram': '64Gb',
  },
  {
    'name': 'server2',
    'os': 'ubuntu-2204-lts',
    'ram': '8Gb',
  },
]
```

YAML: anchors (merge)

`anchors` может так же использоваться для слияния ключей при помощи оператора `<<:`

```yaml
srv-small-template: &srv-small-template
  os: ubuntu-2204-lts
  ram: 8Gb
  hdd: 512Gb

servers:
  - name: server1
    <<: *srv-small-template
  - name: server2
    ram: 16Gb
    <<: *srv-small-template
```

```
with open("<>", "r") as file_:
    pprint(yaml.safe_load(file_))

{
   'servers': [
      {
         'hdd': '512Gb',
         'name': 'server1',
         'os': 'ubuntu-2204-lts',
         'ram': '8Gb',
      },
      {
         'hdd': '512Gb',
         'name': 'server2',
         'os': 'ubuntu-2204-lts',
         'ram': '16Gb',
      },
   ],
   'srv-small-template': {
      'hdd': '512Gb',
      'os': 'ubuntu-2204-lts',
      'ram': '8Gb',
   },
}
```

yaml

YAML: anchors (example)

```yaml
version: "3"
services:
  web1: &web
    image: nginx
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
      - ./templates:/etc/nginx/templates
    port:
      - 80:80
  web2:
    <<: *web
    port:
      - 81:80
  web3:
    <<: *web
    port:
      - 82:80
```

```
{'services': {'web1': {'image': 'nginx',
                       'port': ['80:80'],
                       'volumes': ['./nginx.conf:/etc/nginx/nginx.conf:ro',
                                   './templates:/etc/nginx/templates']},
              'web2': {'image': 'nginx',
                       'port': ['81:80'],
                       'volumes': ['./nginx.conf:/etc/nginx/nginx.conf:ro',
                                   './templates:/etc/nginx/templates']},
              'web3': {'image': 'nginx',
                       'port': ['82:80'],
                       'volumes': ['./nginx.conf:/etc/nginx/nginx.conf:ro',
                                   './templates:/etc/nginx/templates']}},
 'version': '3'}
```

YAML: extra features

```yaml
# ?\s - начало сложного ключа
? >-
  multiple lines
  key
: value

nested list:
  - - Item
  - - yes
  - - 0.5
  - - - nested sequence
      - collapsed

json_map: { "key": "value" }
json_list: [3, 2, 1]

explicit_string: !!str 0.5
explicit_float: !!float 0.5
datetime_canonical: !!timestamp 2022-12-31T01:23:45.3Z
```

```
with open("<>", "r") as _file:
    print(yaml.safe_load(_file))

{
   'multiple lines key': ‘value',

   'nested list': [
     ['Item'],
     [True],
     [0.5],
     [[
       'nested sequence',
       'collapsed',
     ]],
   ],

   'json_map': {'key': 'value'},
   'json_list': [3, 2, 1],

   'explicit_string': '0.5',
   'explicit_float': 0.5,
   'datetime_canonical': datetime.datetime(2022, 12, 31, 1, 23,
   45, 300000, tzinfo=datetime.timezone.utc),
}
```

YAML: Python

- Разнообразные библиотеки:
  - PyYAML (YAML 1.1, 2020-06-01)
  - ruamel.yaml (YAML 1.2, 2023-06-17)
  - …
- dump(), save_dump()- серриализация объектов Python в формат yaml
- load(), full_load(), save_load(), unsafe_load() - дессериализация данных из yaml в Python объекты

  save - загружаются только теги подмножества YAML (!!int, !!flow, !!set …)

  full - загружаются все теги, но без исполнения кода

  unsafe - загружаются все теги с исполнением кода

YAML: serialization

```python
import yaml


class Person:
    def __init__(self, _name: str, _id: int, _age: int) -> None:
        self._name: str = _name
        self._id: int = _id
        self._age: int = _age

    def __repr__(self) -> str:
        return f"({self._id}){self._name}<{self._age}-yo>"


tom = Person("tom", 4, 33)

with open("./tom.yaml", "w") as _file:
    _file.write(yaml.dump(tom))
```

tom.yaml

```yaml
!!python/object:__main__.Person
_age: 33
_id: 4
_name: tom
```

```python
class Person(yaml.YAMLObject):
    yaml_tag = "!PersonPythonClass"

    def __init__(self, _name: str, _id: int, _age: int) -> None:
        ...
```

tom.yaml

```yaml
!PersonPythonClass
_age: 33
_id: 4
_name: tom
```

YAML: deserialization

mark.yaml

```
!PersonPythonClass
_age: 12
_id: 9
_name: mark
```

```
with open("./mark.yaml", "r") as _file:
    mark = yaml.unsafe_load(_file)


type(mark)
__main__.Person


mark
(9)mark<12-yo>
```

YAML: deserialization

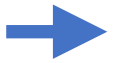
getenv.yaml

```
!!python/object/apply:os.getenv ["HOME"]
```

```
import yaml


yaml.unsafe_load(open("./getenv.yaml"))
'/Users/alexigna'
```

eval.yaml

```
!!python/object/apply:list
- !!python/object/new:map
  - !!python/name:eval
  - [ "5+4", "9+1" ]
```



```
import yaml


yaml.unsafe_load(open("./eval.yaml"))
[9, 10]
```

https://pyyaml.org/wiki/PyYAMLDocumentation

YAML: unhashable key

```
? [0,0]
: "value in dec: 0"
? [0,1]
: "value in dec: 1"
? [1,0]
: "value in dec: 2"
? [1,1]
: "value in dec: 3"
```

```
    raise ConstructorError("while
constructing a mapping", node.start_mark,
            "found unhashable key",
key_node.start_mark)
```

```
? !!python/tuple [0,0]
: "value in dec: 0"
? !!python/tuple [0,1]
: "value in dec: 1”
? !!python/tuple [1,0]
: "value in dec: 2"
? !!python/tuple [1,1]
: "value in dec: 3"
```

```
{(0, 0): 'value in dec: 0',
 (0, 1): 'value in dec: 1',
 (1, 0): 'value in dec: 2',
 (1, 1): 'value in dec: 3'}
```

YAML: summary

- Объемная спецификация
- Неоднозначность типов

```
port_mapping: [22:22, 80:80]
{'port_mapping': [1342, '80:80']}
```

- Сложная структура: якоря/псевдонимы, теги

```
files:
  - /info.txt
  - *.html
```

- Версионность
- Булевы переменные
- Строки vs числа

```
versions: [9.3.1, 10.2]
{'versions': ['9.3.1', 10.2]}
```

- Сложность чтения больших документов
- Мультистроки

https://stackoverflow.com/questions/3790454/how-do-i-break-a-string-in-yaml-over-multiple-lines

YAML: summary

Популярность YAML:

- Вложенная структура vs плоская у INI
- Простота написания vs XML
- Наличие комментариев, переменных/якорей vs JSON

Альтернативы

- TOML
- Улучшение JSON
- Всегда использовать save_ и избегать неоднозначной типизации

YAML: homework

```
# ping -c 2 google.com

# ls -l /
```

```python
import yaml


class Person:
    def __init__(self, _name: str, _id: int, _age: int) -> None:
        self._name = _name
        self._id = _id
        self._age = _age

    def __repr__(self) -> str:
        return f"({self._id}){self._name}<{self._age}-yo>"

    def inc_age(self) -> None:
        self._age += 1

    def dump_to_yaml_file(self, filename: str) -> None:
        with open(filename, "w") as _file:
            _file.write(yaml.dump(self))

tod = Person("tod", 4, 22)
tod.inc_age()
tod.dump_to_yaml_file("./tod.yaml")
```

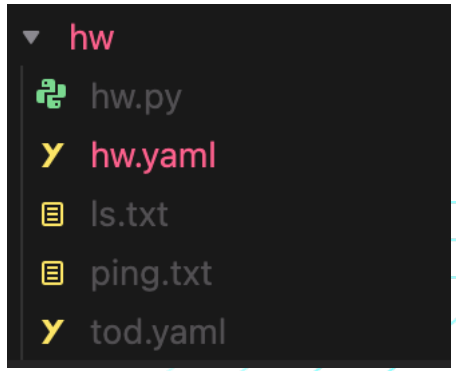
hw.yaml

```
<some yaml text of homework>
```

hw.py

```python
import yaml

with open("./hw.yaml", "r") as _file:
    config = yaml.unsafe_load(_file)
```

```
# python hw.py
```



NFE Networking For Everyone