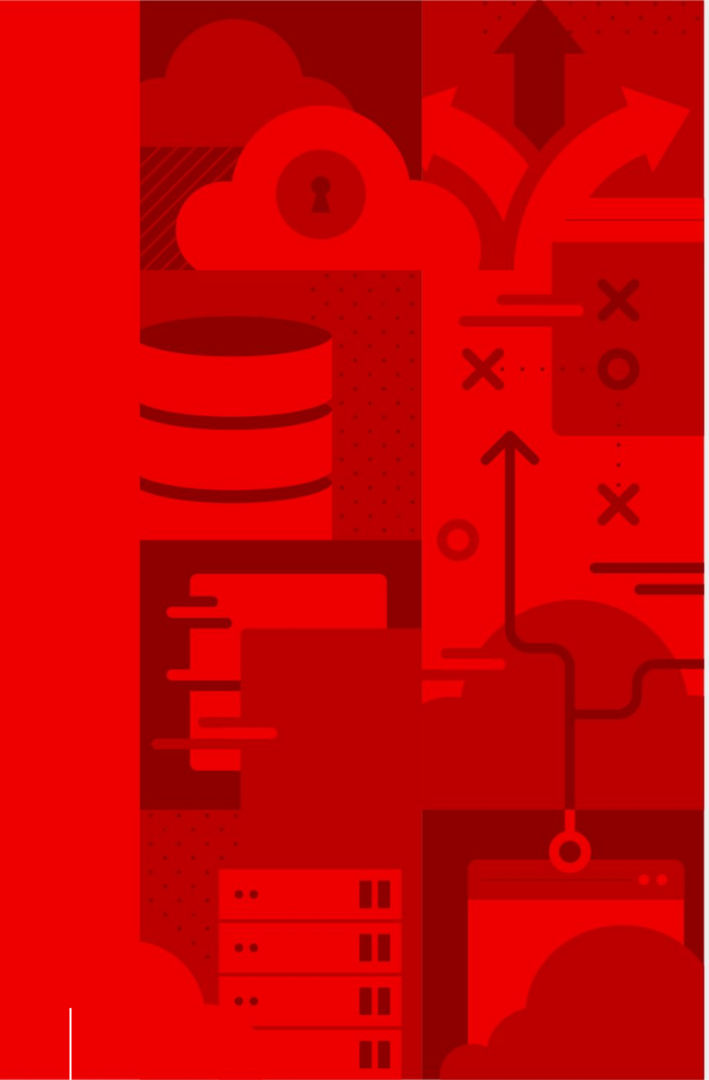


Dashboards as a Code: managing Grafana with Jsonnet

Szymon Datko & **Adrian Fusco Arnejo**

OpenInfra Day Germany – May 15th, 2024

About Us

Szymon Datko
Senior Software Engineer

– Linux enthusiast and free/open source software lover.

– Loves playing board and computer games.

– Teacher at Wrocław University of Science and Technology.

Adrian Fusco
Software Engineer

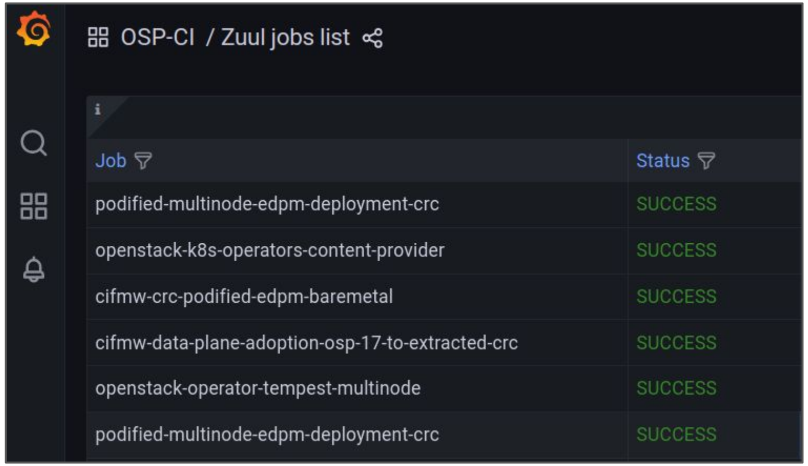
– DevOps soul, hartened with Perl and Bash.

– Passionate for traveling, food and immersions into different cultures.

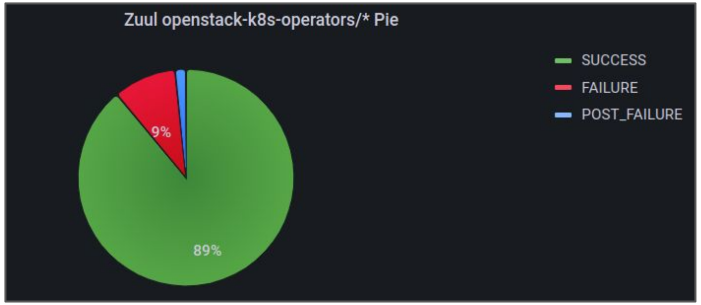
– Fluently speaks Español, English, Galego, Italiano, learning Turkçe.

What is Grafana?

- Visualization & Dashboarding Platform.
- Offers several kinds of visualizations.
    - Tables, graphs, charts, …
- Unifies data from different sources.
    - Agnostic: databases, metrics, APIs…
- Provides built-in alerting system.
- Open Source & Community Driven!
    - Many additional plugins available!

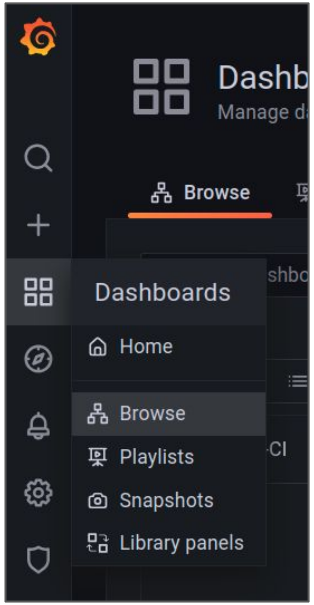

OSP-CI / Zuul jobs list

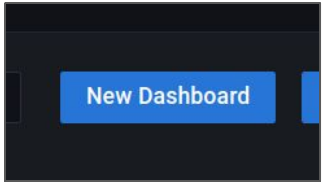
Job	Status
podified-multinode-edpm-deployment-crc	SUCCESS
openstack-k8s-operators-content-provider	SUCCESS
cifmw-crc-podified-edpm-baremetal	SUCCESS
cifmw-data-plane-adoption-osp-17-to-extracted-crc	SUCCESS
openstack-operator-tempest-multinode	SUCCESS
podified-multinode-edpm-deployment-crc	SUCCESS

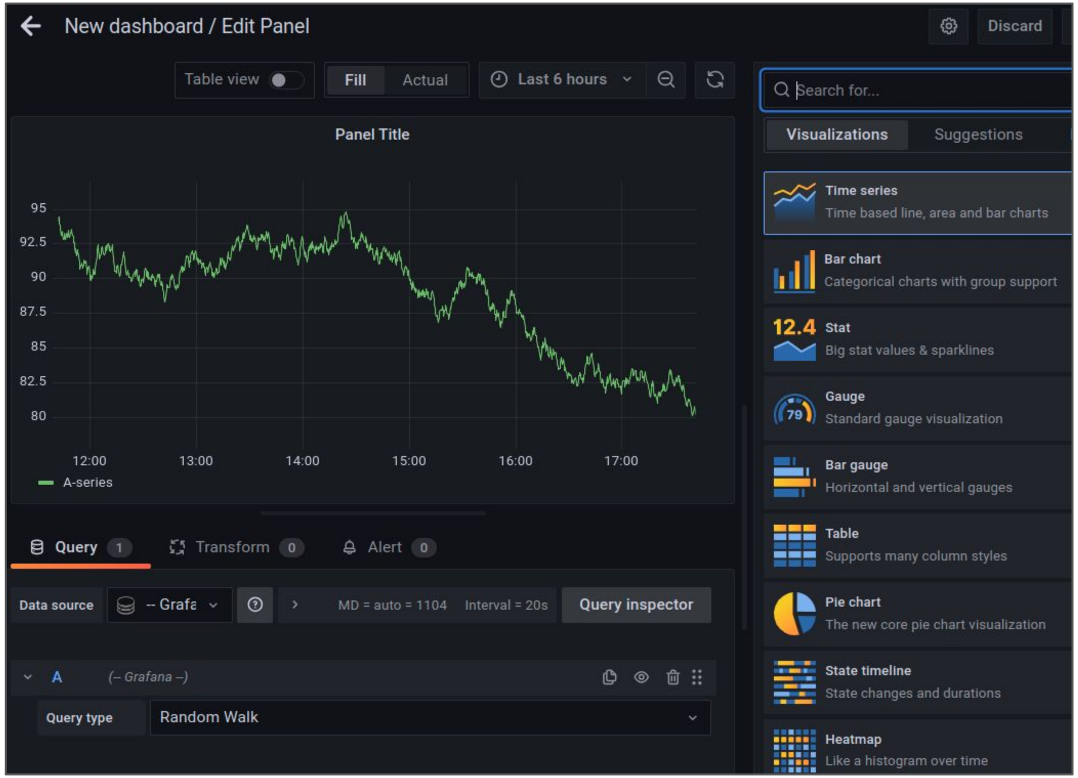


Red Hat

Setting up dashboards...






New dashboard / Edit Panel

Discard

Table view | Fill | Actual | Last 6 hours

Search for...

Visualizations | Suggestions

Time series — Time based line, area and bar charts
Bar chart — Categorical charts with group support
Stat — Big stat values & sparklines
Gauge — Standard gauge visualization
Bar gauge — Horizontal and vertical gauges
Table — Supports many column styles
Pie chart — The new core pie chart visualization
State timeline — State changes and durations
Heatmap — Like a histogram over time

Panel Title

Query 1 | Transform 0 | Alert 0

Data source: -- Grafa

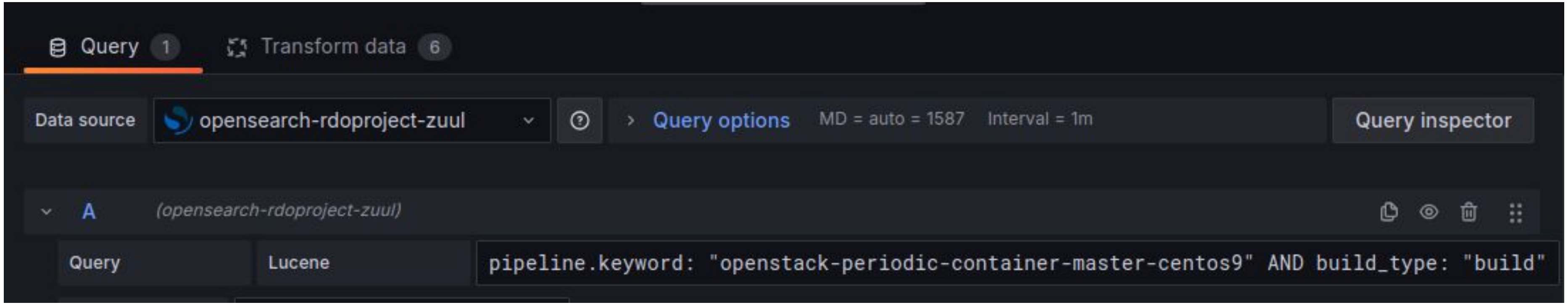
MD = auto = 1104 Interval = 20s

Query inspector

A (-- Grafana --)

Query type: Random Walk

Setting up dashboards... – continued

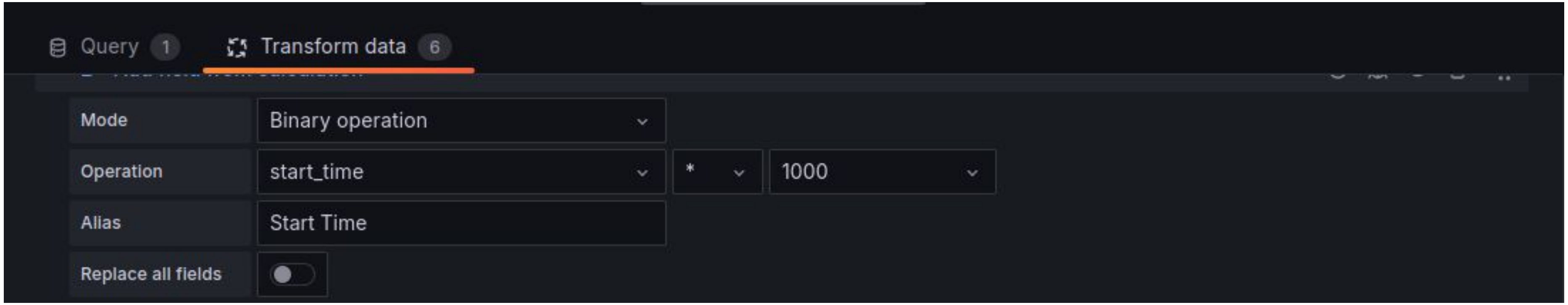


Query 1 | Transform data 6

Data source: opensearch-rdoproject-zuul | Query options | MD = auto = 1587 | Interval = 1m | Query inspector

A (opensearch-rdoproject-zuul)

Query	Lucene	`pipeline.keyword: "openstack-periodic-container-master-centos9" AND build_type: "build"`



Query 1 | Transform data 6

Mode	Binary operation		
Operation	start_time	*	1000
Alias	Start Time		
Replace all fields	off		

Red Hat

Why it is an issue?

- **Scale**:

  Setup new dashboards and visualizations

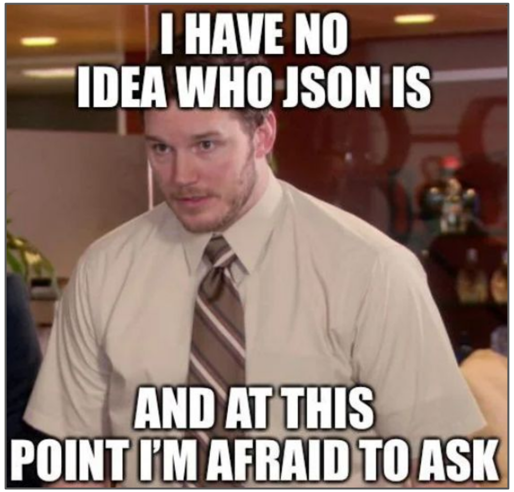
  for monitoring hundreds of pipelines and jobs?

- **Versioning and history**:

  Review a long JSON file with thousands of lines?

- **Safety**:

  Disaster-recovery, backup and update plan.


I HAVE NO IDEA WHO JSON IS
AND AT THIS POINT I'M AFRAID TO ASK

A solution exists!

Remedy for Grafana configuration issues:

– **jsonnet** (configuration language)

– **jb** (jsonnet-bundler = jsonnet package installer)

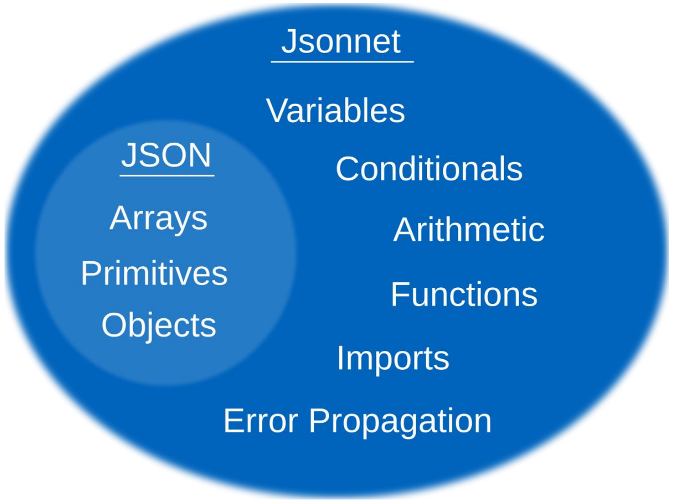
– **grafonnet** (jsonnet library developed by Grafana Labs team)

  – Allows developers to write a code to perform same actions that are usually done through the UI.

What is Jsonnet?

– Pure functional language with object-oriented features.
  – All values are inmutables!
– A simple extension of JSON.
  – Any JSON document is a valid Jsonnet program.
– Designed primarily for configuring complex systems.
– Mainly used for producing JSON files.
  – Supports also INI, XML and YAML outputs.
– Hermeticity: Independence from the Environment.
  – Programs are *pure computations*. Only explicit input.
– Utilized by various popular applications and platforms.
  – Including Openshift, Kubernetes, Grafana, …

```
$ echo '{"hello": "world"}' > hello.jsonnet
$ jsonnet hello.jsonnet
{
   "hello": "world"
}
```



Visit the official
jsonnet documentation
for details.

Example program #1 – using loops

```
$ cat loop.jsonnet

local protocol_information = {
  protocols: [
    {
      port: 3306,
      type: 'mysql',
      description: 'MySQL/MariaDB service',
    },
    {
      port: 443,
      type: 'https',
      description: 'HTTPS (HTTP over SSL/TLS)',
    },
    {
      port: 80,
      type: 'http',
      description: 'Hypertext transfer protocol',
    },
  ],
};

{
  protocol_information: [
    {
      [p.port + '/' + p.type]: p.description,
    }
    for p in protocol_information.protocols
  ],
}
```

⇒

```
$ jsonnet loop.jsonnet

{
   "protocol_information": [
      {
         "3306/mysql": "MySQL/MariaDB service"
      },
      {
         "443/https": "HTTPS (HTTP over SSL/TLS)"
      },
      {
         "80/http": "Hypertext transfer protocol"
      }
   ]
}
```

Example program #2 – generating INI files

```
local app_names = ['api', 'frontend', 'backend'];
std.manifestIni({
  sections: {
    supervisord: {
      user: 'fedora',
      directory: '/tmp',
      logfile: '/tmp/supervisord.log',
      logfile_maxbytes: '75MB',
      logfile_backups: 5,
      loglevel: 'info',
      pidfile: '/tmp/supervisord.pid',
    }
  } + {
    ['program:flask_app_' + app]: {
      command: 'gunicorn flask_app_' + app + '.py',
      directory: '/home/fedora/apps/',
      autostart: true,
      stderr_logfile: 'logs/flas_app_' + app +
'.err.log',
      stdout_logfile: 'logs/flas_app_' + app +
'.out.log',
    } for app in app_names
  }
})
```

⇒

```
[program:flask_app_api]
autostart = true
command = gunicorn flask_app_api.py
directory = /home/fedora/apps/
stderr_logfile = logs/flas_app_api.err.log
stdout_logfile = logs/flas_app_api.out.log
[program:flask_app_backend]
autostart = true
command = gunicorn flask_app_backend.py
directory = /home/fedora/apps/
stderr_logfile = logs/flas_app_backend.err.log
stdout_logfile = logs/flas_app_backend.out.log
[program:flask_app_frontend]
autostart = true
command = gunicorn flask_app_frontend.py
directory = /home/fedora/apps/
stderr_logfile = logs/flas_app_frontend.err.log
stdout_logfile = logs/flas_app_frontend.out.log
[supervisord]
directory = /tmp
logfile = /tmp/supervisord.log
logfile_backups = 5
logfile_maxbytes = 75MB
loglevel = info
pidfile = /tmp/supervisord.pid
user = fedora
```

Example program #3 – generating XML files

```
local xml = import 'xml.libsonnet';
local xmlResponseTemplate = xml.Element('xml', content='') {
  version: '1.0',
  encoding: 'utf-8',
  statusCode:: xml.Element('statusCode', content='200'),
  callID:: xml.Element('id', content=std.extVar('id')),
  parametersData:: xml.Element('parametersData', content='') {
    requestName:: xml.Element('requestName', content='getLocation'),
    has: [
      self.requestName,
      xml.Element('parameters', content='') {
        timeZone:: xml.Element('timeZone', content='Europe/Madrid'),
        has: [self.timeZone],
      },
    ],
  },
  responseData:: xml.Element('responseData', content='') {
    dayOfTheWeek:: xml.Element('dayOfTheWeek', content='Thursday'),
    date:: xml.Element('date', content='05/09/2024'),
    time:: xml.Element('time', content='18:06'),
    has: [self.dayOfTheWeek, self.date, self.time],
  },
  has: [$.statusCode, $.callID, $.parametersData, $.responseData],
};
xml.manifestXmlObj(xmlResponseTemplate)
```

$\Rightarrow$

```
$ jsonnet -S simple_example.jsonnet \
      --ext-str id=$(uuidgen)

<xml encoding="utf-8" version="1.0">
  <statusCode>200</statusCode>
  <id>66534247-2d4f-42b1-8f83-75af89c93dca</id>
  <parametersData>
    <requestName>getLocation</requestName>
    <parameters>
      <timeZone>Europe/Madrid</timeZone>
    </parameters>
  </parametersData>
  <responseData>
    <dayOfTheWeek>Thursday</dayOfTheWeek>
    <date>05/09/2024</date>
    <time>18:06</time>
  </responseData>
</xml>
```

Red Hat

Example program #4 – generating YAML files

```
local containerImage = std.extVar('containerImage');
local containerImageTag = std.extVar('containerImageTag');
local applicationPort = std.parseInt(std.extVar('appPort'));
local replicas = std.parseInt(std.extVar('replicas'));

std.manifestYamlDoc({
  apiVersion: 'apps/v1',
  kind: 'Deployment',
  metadata: { name: 'webserver-deployment', },
  spec: {
    replicas: replicas,
    selector: {
      matchLabels: { app: containerImage }
    },
    template: {
      metadata: { labels: { app: containerImage } },
      spec: {
        containers: [
          {
            name: containerImage,
            image: containerImage + ':' + containerImageTag,
            ports: [{containerPort: applicationPort}],
            livenessProbe: {
              httpGet: { path: '/', port: applicationPort },
            },
        …
})
```

⇒

```
"apiVersion": "apps/v1"
"kind": "Deployment"
"metadata":
  "name": "webserver-deployment"
"spec":
  "replicas": 3
  "selector":
    "matchLabels":
      "app": "nginx"
  "template":
    "metadata":
      "labels":
        "app": "nginx"
    "spec":
      "containers":
      - "image": "nginx:latest"
        "livenessProbe":
          "httpGet":
            "path": "/"
            "port": 80
        "name": "nginx"
        "ports":
        - "containerPort": 80
      …
```

Example #5 – creating kubernetes deployment… ;-)

```
$ jsonnet jsonnet-examples/yaml-files/deployment.yaml.jsonnet \
          --ext-str containerImage=nginx \
          --ext-str containerImageTag=latest \
          --ext-str appPort=80 \
          --ext-str replicas=3 \
          | kubectl apply -f -
deployment.apps/webserver-deployment configured

$ kubectl get deployment
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
webserver-deployment   4/4     4            4           4m21s

$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
webserver-deployment-8555667db5-25bsz   1/1     Running   0          94s
webserver-deployment-8555667db5-8twh6   1/1     Running   0          31s
webserver-deployment-8555667db5-vl8hv   1/1     Running   0          102s
webserver-deployment-8555667db5-vzqrp   1/1     Running   0          91s
```

What is jb?

– **jb** = **j**sonnet-**b**undler

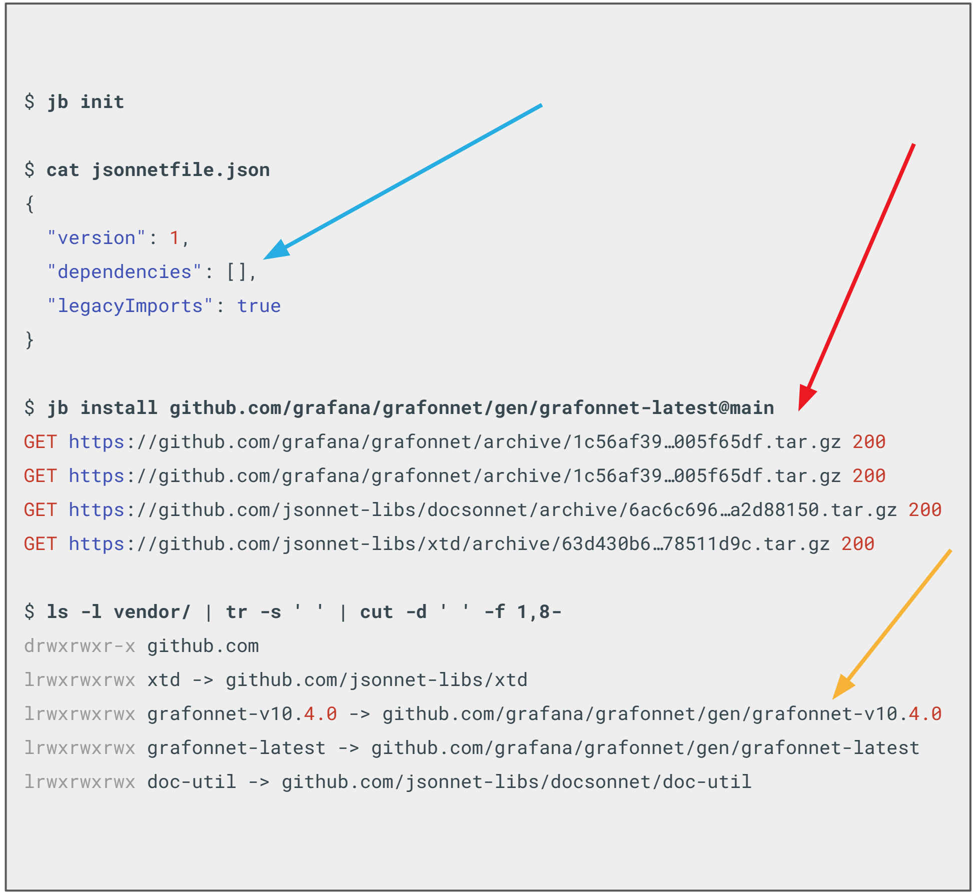
– A Jsonnet package manager.

– Resolves dependencies for libraries.

– Allows specifying version constraints.

– Project still in alpha stage;
  Flags, behavior and design may change…

– jsonnet-bundler repository



```
$ jb init

$ cat jsonnetfile.json
{
  "version": 1,
  "dependencies": [],
  "legacyImports": true
}

$ jb install github.com/grafana/grafonnet/gen/grafonnet-latest@main
GET https://github.com/grafana/grafonnet/archive/1c56af39…005f65df.tar.gz 200
GET https://github.com/grafana/grafonnet/archive/1c56af39…005f65df.tar.gz 200
GET https://github.com/jsonnet-libs/docsonnet/archive/6ac6c696…a2d88150.tar.gz 200
GET https://github.com/jsonnet-libs/xtd/archive/63d430b6…78511d9c.tar.gz 200

$ ls -l vendor/ | tr -s ' ' | cut -d ' ' -f 1,8-
drwxrwxr-x github.com
lrwxrwxrwx xtd -> github.com/jsonnet-libs/xtd
lrwxrwxrwx grafonnet-v10.4.0 -> github.com/grafana/grafonnet/gen/grafonnet-v10.4.0
lrwxrwxrwx grafonnet-latest -> github.com/grafana/grafonnet/gen/grafonnet-latest
lrwxrwxrwx doc-util -> github.com/jsonnet-libs/docsonnet/doc-util
```

Installing grafonnet via jb

What is jb? – tracking dependencies and utilizing them

```
$ cat jsonnetfile.json
{
  "version": 1,
  "dependencies": [
    {
      "source": {
        "git": {
          "remote": "https://github.com/grafana/grafonnet.git",
          "subdir": "gen/grafonnet-latest"
        }
      },
      "version": "main"
    }
  ],
  "legacyImports": true
}
```

```
$ cat my-dashboard.jsonnet
local grafonnet = import 'grafonnet-v10.4.0/main.libsonnet';
local dashboard = grafonnet.dashboard;
dashboard.new('Hello') + dashboard.withDescription('This is a test')
```

```
$ jsonnet -J vendor/ my-dashboard.jsonnet
{
   "description": "This is a test",
   "schemaVersion": 36,
   "time": {
      "from": "now-6h",
      "to": "now"
   },
   "timezone": "utc",
   "title": "Hello"
}
```

Red Hat

What is grafonnet?

– Jsonnet library for generating Grafana dashboards & visualizations.

– Developed by Grafana Labs.

– Resolves the problem of previous grafonnet-lib.

– The library is automatically updated based on the JSON schemas by Grok (Grafana Object development Kit).

– If there is a new version of Grafana, there is a new version of grafonnet.

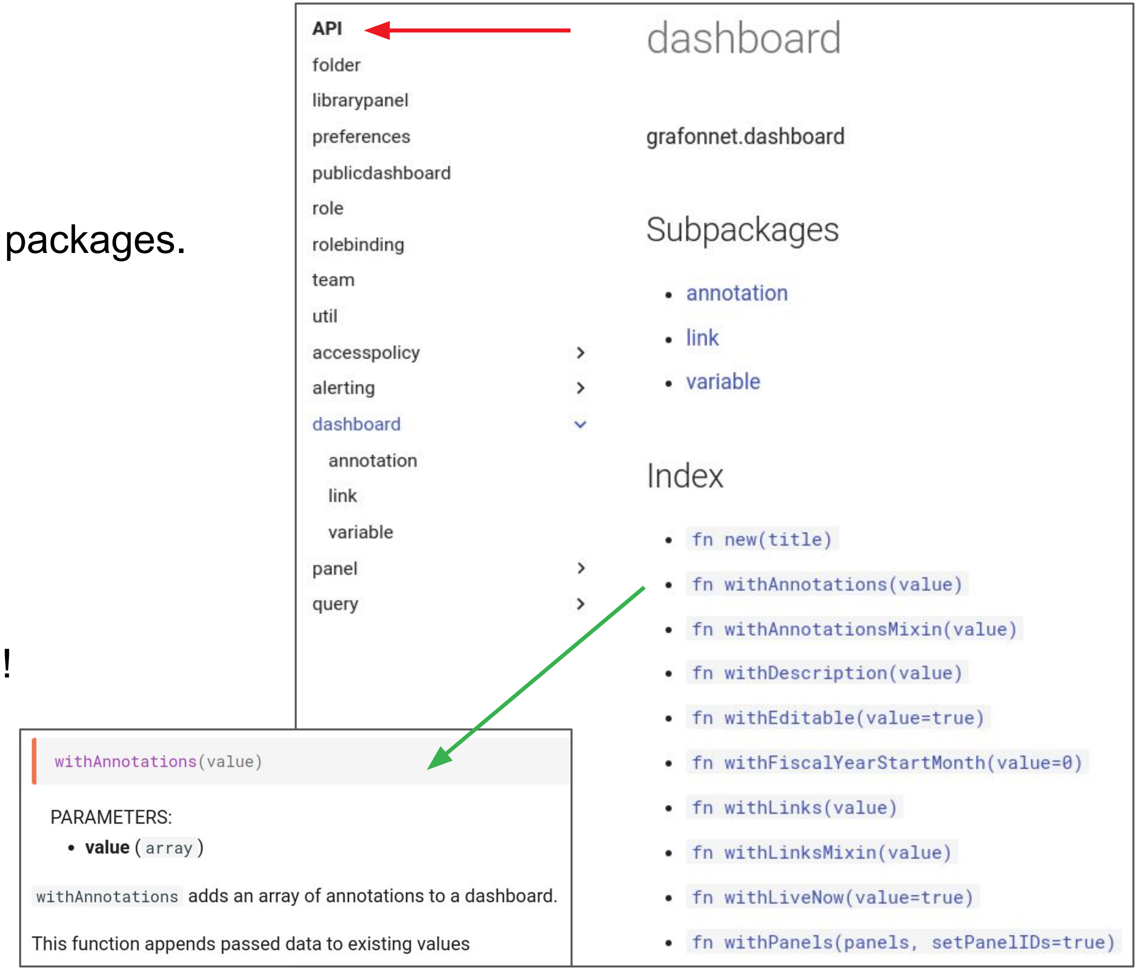
```
$ ls vendor/grafonnet-v10.4.0/
alerting.libsonnet  clean  custom  docs  jsonnetfile.json
main.libsonnet  panel.libsonnet  query.libsonnet  raw
```

```
$ cat vendor/grafonnet-v10.4.0/main.libsonnet
...
{
  ...
  accesspolicy: import 'raw/accesspolicy.libsonnet',
  dashboard: import 'clean/dashboard.libsonnet',
  librarypanel: import 'raw/librarypanel.libsonnet',
  preferences: import 'raw/preferences.libsonnet',
  publicdashboard: import 'raw/publicdashboard.libsonnet',
  role: import 'raw/role.libsonnet',
  rolebinding: import 'raw/rolebinding.libsonnet',
  team: import 'raw/team.libsonnet',
  folder: import 'raw/folder.libsonnet',
  panel: import 'panel.libsonnet',
  query: import 'query.libsonnet',
  util: import 'custom/util/main.libsonnet',
  alerting: import 'alerting.libsonnet',
}
```

Grafonnet library content. Schemas are generated from Grafana project.

Grafonnet API

– Defines all the functions available in packages.
  – Dashboards,
  – visualizations,
  – queries,
  – alerting…
– We can perform the same actions that are possible via Grafana WebUI!
– Grafonnet repository
– Grafonnet documentation



dashboard

grafonnet.dashboard

Subpackages

- annotation
- link
- variable

Index

- `fn new(title)`
- `fn withAnnotations(value)`
- `fn withAnnotationsMixin(value)`
- `fn withDescription(value)`
- `fn withEditable(value=true)`
- `fn withFiscalYearStartMonth(value=0)`
- `fn withLinks(value)`
- `fn withLinksMixin(value)`
- `fn withLiveNow(value=true)`
- `fn withPanels(panels, setPanelIDs=true)`

`withAnnotations(value)`

PARAMETERS:
- **value** (`array`)

`withAnnotations` adds an array of annotations to a dashboard.

This function appends passed data to existing values

Brace yourself...

Dare to experiment on your own!

1 – Clone the repository with our examples and jump into it:

```
git clone https://github.com/adrianfusco/openinfra2024-dashboard-as-a-code.git
cd openinfra2024-dashboard-as-a-code/grafana/
```

2 – Play with the configuration files.

- See: `config/dashboards/OpenInfra/jsonnet/` and other places.
- Build with: `jsonnet -J {JSONNET_VENDOR_PATH} {INPUT_FILE} > {OUTPUT_FILE}`
- Data come from: https://opensearch.rdoproject.org/

3 – Deploy the Grafana in container.

```
docker compose up
```

4 – Open http://localhost:3000/ in your web browser.

Red Hat

Summary

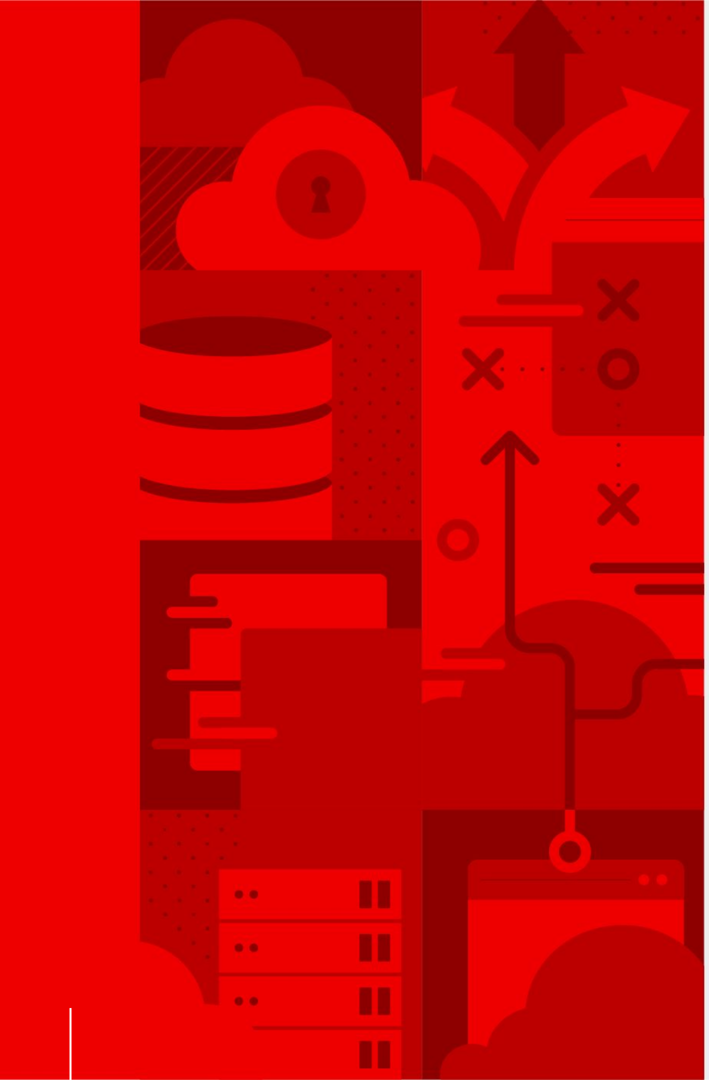
- Grafana managed as Code with trio:
  - **jsonnet** (configuration language)
  - **jb** (jsonnet-bundler = jsonnet package installer)
  - **grafonnet** (jsonnet library developed by Grafana Labs team)
- Especially useful big and complex ecosystems.
- The tooling can be utilized not only for Grafana!
- ”*Jsonnet is extremely powerful. But, the learning curve is pretty serious. It’s basically coding json files with a functional programming approach.*”

  /poweredupfaxmachine, September 2021 in Reddit/
- Visit our repository with examples – play and learn by yourself!
  https://github.com/adrianfusco/openinfra2024-dashboard-as-a-code

Grab the slides!

SCAN ME

https://datko.pl/oi-berlin.pdf



Dashboards as a Code: managing Grafana with Jsonnet

Szymon Datko & **Adrian Fusco Arnejo**

OpenInfra Day Germany – May 15th, 2024