JQ Einführung

JQ der Filter für JSON

> jq is a lightweight and flexible command-line JSON processor

- JQ ist, in bester Unix Manier, ein Filter für JSON Daten
- JQ – https://stedolan.github.io/jq/

Beispiel

Originale Ausgabe

```
└─$ ip --json -4 --brief address show
[{"ifname":"lo","operstate":"UNKNOWN","addr_info":[{"local":"127.0.0.1","prefixlen":8}]},{"ifn
ame":"wlp0s20f3","operstate":"UP","addr_info":[{"local":"10.23.91.219","prefixlen":16}]},{"ifn
ame":"virbr0","operstate":"DOWN","addr_info":[{"local":"192.168.122.1","prefixlen":24}]},{"ifn
ame":"cni-podman1","operstate":"UP","addr_info":[{"local":"192.168.49.1","prefixlen":24}]}]
```

Highlighting mit JQ

Highlighting und Pretty Print

JQ kann fabelhaft dafür genutzt werden, Highlighting und Pretty Printing zu machen.

- `cat foo.json | jq '.'` – für Pretty Print
- `cat foo.json | jq -c '.'` – für Kompakt
- `cat foo.json | jq -C '.' | less -R` – Force Color, damit man die Farben in less sieht

Weitere Parameter

- `cat foo.json | jq -r '.'` – Raw, nimmt die Anführungszeichen aus den Strings
- `cat foo.json | jq -s '.'` – Slurp, Macht aus jeder Eingabezeile einen Array

Filtern

Viele moderne Programme können JSON Output. Einige Andere kann man mit Hilfe von `jc` zu JSON konvertieren.

Ein Beispiel für JC

```
$ jc dig www.google.com

[{"id":33414,"opcode":"QUERY","status":"NOERROR","flags":["qr","rd","ra"],"query_num":
1,"answer_num":1,"authority_num":0,"additional_num":1,"opt_pseudosection":{"edns":{"ve
rsion":0,"flags":[],"udp":65494}},"question":{"name":"www.google.com.","class":"IN","t
ype":"A"},"answer":[{"name":"www.google.com.","class":"IN","type":"A","ttl":87,"data":
"142.250.185.100"}],"query_time":4,"server":"127.0.0.53#53(127.0.0.53)","when":"Fri
Aug 12 18:12:16 CEST 2022","rcvd":59,"when_epoch":1660320736,"when_epoch_utc":null}]
```

- JC – https://github.com/kellyjonbrazil/jc

Komandostruktur

JQ führt ähnlich zu `sed` alle Filter der Reihe nach durch. Man filtert sich also von Objekt zu Objekt und tranformiert ggfs. die Objekte auf dem Weg.

- `.` – das Aktuelle Objekt
- `.kind` – der Inhalt von `kind`
- `.[].kind` – alle `kind`-Inhalte im Array
  - Identisch mit `.[] | .kind`
  - Hierbei werden die nachfolgenden Filter für jedes Arrayelement ausgeführt

Neue Datenstrukturen

JQ kann die vorhanden Objekte auch umbauen.

StorageClass ohne Filter

```
$ minikube kubectl -- get sc standard -o json

{
    "apiVersion": "storage.k8s.io/v1",
    "kind": "StorageClass",
    "metadata": {
        "annotations": {
            "storageclass.kubernetes.io/is-default-class": "true"
        },
        "creationTimestamp": "2022-04-08T18:58:31Z",
        "labels": {
            "addonmanager.kubernetes.io/mode": "EnsureExists"
```

```
        },
        "name": "standard",
    },
    "provisioner": "k8s.io/minikube-hostpath",
    "reclaimPolicy": "Delete",
    "volumeBindingMode": "Immediate"
}
```

Neue Datenstrukturen 2

Nur bestimmte Daten aus der StorageClass

```
$ minikube kubectl -- get sc standard -o json \
    | jq '. | {kind,
            name: .metadata.name,
            provisioner,
            reclaimPolicy}'

{
  "kind": "StorageClass",
  "name": "standard",
  "provisioner": "k8s.io/minikube-hostpath",
  "reclaimPolicy": "Delete"
}
```

Strings

Es ist auch oft hilfreich einfach gewünschte Felder in einen String zusammen zu stellen.

Eine Liste aller Pods

```
$ minikube kubectl -- get \
   pod -n flux-system -o json \
 | jq -r '.items[]
    | "\(.kind) "
    + "\(.metadata.namespace) "
    + "\(.metadata.name) "
    + "\(.spec.containers[].name) "
    + "\(.spec.containers[].image)"' \
  | column -t

Pod flux-system  helm-controller-dfb4b5478-qqc9n       manager  ghcr.io/fluxcd/helm-
controller:v0.18.2
Pod flux-system  kustomize-controller-cd544c8f8-dksh4  manager
ghcr.io/fluxcd/kustomize-controller:v0.22.3
...
```

Alternativen: Das Problem

Fehler bei einzelnen Pods

```
$ minikube kubectl -- get \
    pod helm-controller-dfb4b5478-qqc9n \
    -n flux-system -o json \
  | jq -r '.items[]
  ...

jq: error (at <stdin>:265): Cannot iterate over null (null)
```

Alternativen: Die Lösung

Eine Liste oder ein einzelner Pod

```
$ minikube kubectl -- get \
    pod  helm-controller-dfb4b5478-qqc9n \
    -n flux-system -o json \
  | jq -r '.items[]? // .
    | "\(.kind) "
    + "\(.metadata.namespace) "
    + "\(.metadata.name) "' \
  | column -t

Pod  flux-system  helm-controller-dfb4b5478-qqc9n
```

Mapping

Kubernetes Secrets haben ihre Daten mit Base64 encondiert.

Decoded secrets

```
minikube kubectl -- get secrets -A -o json \
  |jq '.items[]? // .
    | { name: .metadata.name,
        data: .data
        | map_values(@base64d) }'

{
  "ca.crt": "-----BEGIN CERTIFICATE-----\nMIID...kuOpg==\n-----END CERTIFICATE-----
\n",
  "namespace": "default",
  "token": "eyJhb...FSAg"
}
```

Join

Hier möchte ich nur die Namen der Keys sehen, welche in den Screts stehen.

Keys aller Secrets

```
$ minikube kubectl -- get secrets -A -o json \
|jq '.items[]? // .
    | { name: .metadata.name,
        data: (.data | keys | join(", ")) }'

{
  "name": "ttl-controller-token-l4crc",
  "data": "ca.crt, namespace, token"
}
```

Select

Secrets ohne tokens

```
$ minikube kubectl -- get secrets -A -o json \
  |jq '.items[]? // .
    | select(.data.token | not) | { name: .metadata.name,
        data: (.data | keys | join(", ")) }'

{
  "name": "flux-system",
  "data": "identity, identity.pub, known_hosts"
}
{
  "name": "grafana",
  "data": "admin-password, admin-user, ldap-toml"
}
{
  "name": "sh.helm.release.v1.grafana.v1",
  "data": "release"
}
```

Linksammelung

- JQ – https://stedolan.github.io/jq/
- JC – https://github.com/kellyjonbrazil/jc
- Minikube – https://minikube.sigs.k8s.io/docs/