Cilium NetworkPolicies

with Jsonnet

Sebastian Plattner

plattner@puzzle.ch

PUZZLE ITC
changing IT for the better

Nice to meet you

Sebastian Plattner
System Architect
plattner@puzzle.ch

Agenda

- Requirements
- Kubernetes Basics
- JSONNET
- Implementation
- Demo 🤞

Requirements

Requirement

❶ Block Internet Traffic On Deploy Runner but

❷

Allow Traffic To Specified FQDNs

while having it as

3

As Code

Kubernetes Basics

Kubernetes Basics

Least privilege

- By **default** Kubernetes imposes **no restrictions** on network traffic.
- As soon as you implement your **first NetworkPolicy** that selects a Pod, Kubernetes will **start restricting** all traffic to/from that Pod
- If you create a “default deny” NetworkPolicy, that **selects all Pods** and specifies **both Ingress and Egress types,** Kubernetes will start behaving in a **least-privilege** manner

Kubernetes Basics

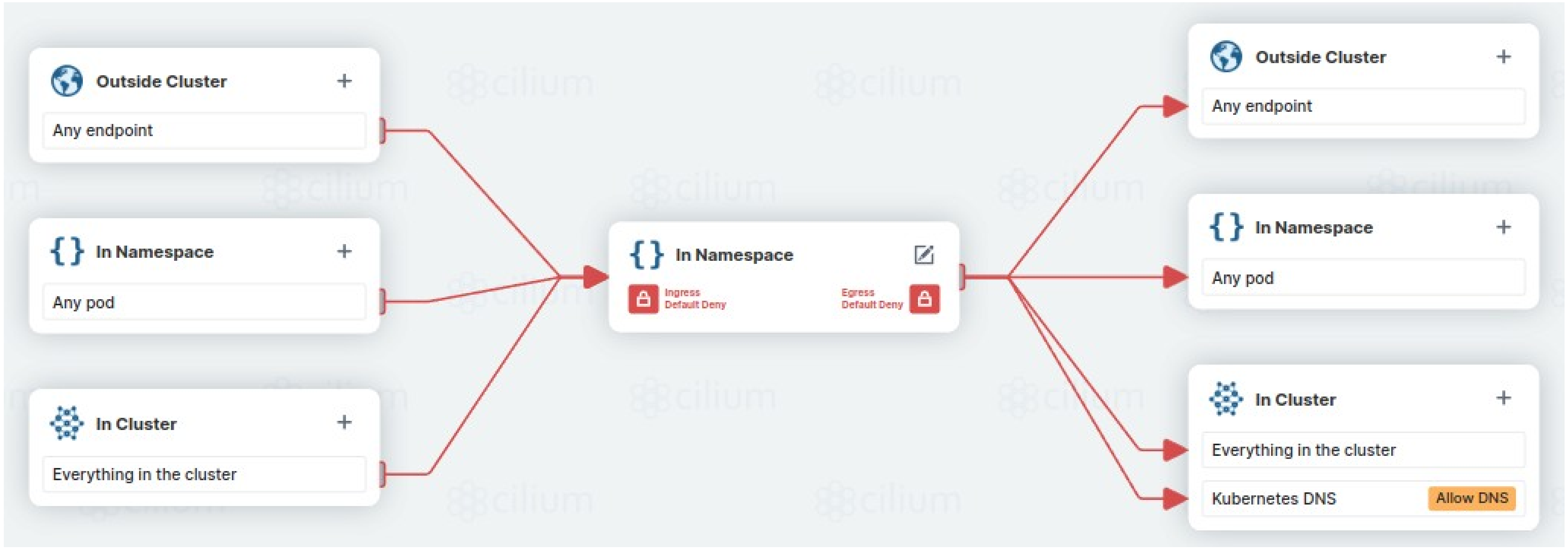
Network Policy

1

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
  namespace: my-ns
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

select all Pods in Namespace

default-deny-all



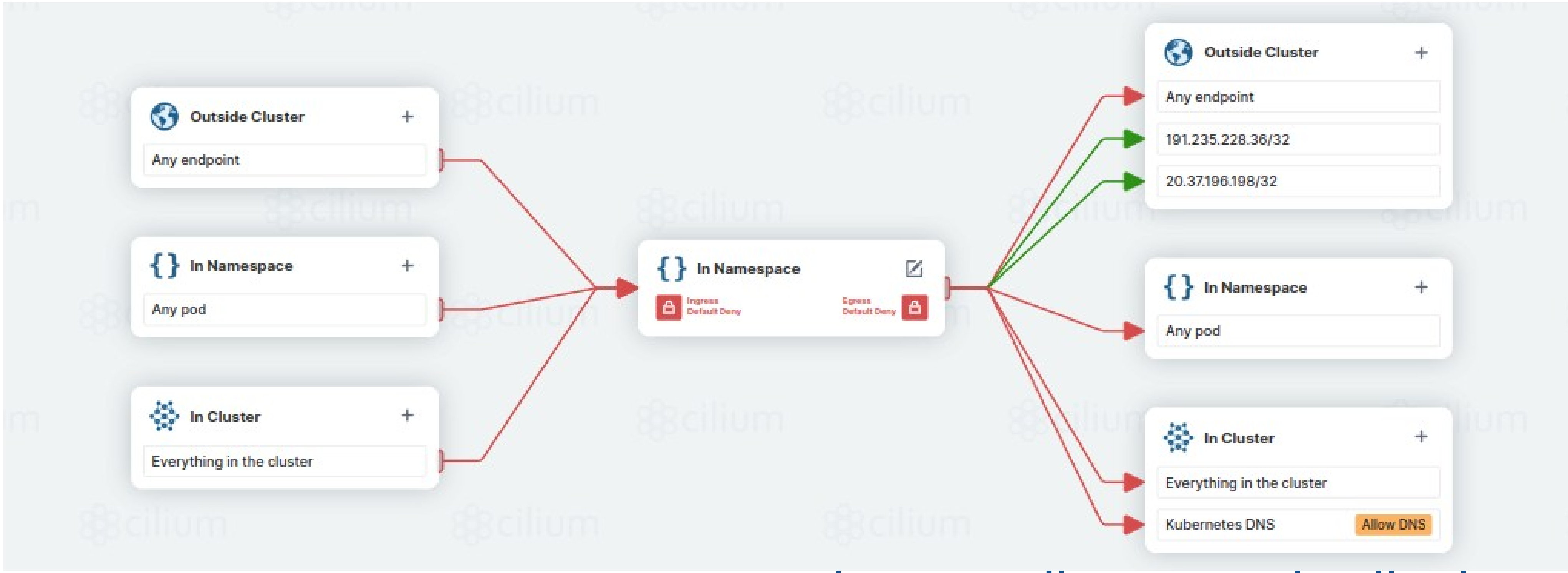
https://editor.networkpolicy.io/

Kubernetes Basics

Network Policy

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-azure
  namespace: my-ns
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress:
  - ipBlock:
      cidr: 191.235.228.36/32
  - ipBlock:
      cidr: 20.37.196.198/32
```

allow-azure



https://editor.networkpolicy.io/

but

```
dig azure.com

;; ANSWER SECTION:
azure.com.   3514 IN A  191.235.228.36
azure.com.   3514 IN A  40.74.100.137
azure.com.   3514 IN A  20.43.132.131
azure.com.   3514 IN A  20.37.196.198
azure.com.   3514 IN A  20.50.2.51
azure.com.   3514 IN A  20.49.104.40
azure.com.   3514 IN A  40.112.243.51
```

Kubernetes Basics

Network Policy

```
apiVersion: networking.k8s.io/v1
kind: N    kPolicy
metadata
  name: a    azur
  namespac     -r
spec:
  podSelecto
  policyType
  - Egress
  egress:
  - ipBl
      ci   191.235    36/32
  - ip
      cidr: 20.37.196.198/32
```

Kubernetes Basics

Network Policy

- Only Layer4/Layer4
  - IP, TCP/UDP, Port
- IP addresses can change and they do!

eBPF-based Networking, Observability, Security

Cilium is an open source, cloud native solution for providing, securing, and observing network connectivity between workloads, fueled by the revolutionary Kernel technology eBPF

cilium

Cilium Network Policy for the rescue

- Layer 3,4 & 7!
- Layer 3:
  - Allow to use FQDN
- Layer 7:
  - HTTP
  - DNS
  - ...

cilium

2

Cilium Network Policy

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-azure
Spec:
  endpointSelector:
    matchLabels: {}
  egress:
    - toFQDNs:
        - matchName:azure.com
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP
```

cilium

YOU SAID ALL I NEEDED WAS A YAML FILE
imgflip.com

JSONNET

JSONNET

A configuration language for app and tool developers

JSONNET

- **It began life early 2014 as a 20% project and was launched on Aug 6**
- **The design is influenced by several configuration languages internal to Google**
- **Jsonnet is not an official Google product (experimental or otherwise), it is just code that happens to be owned by Google.**

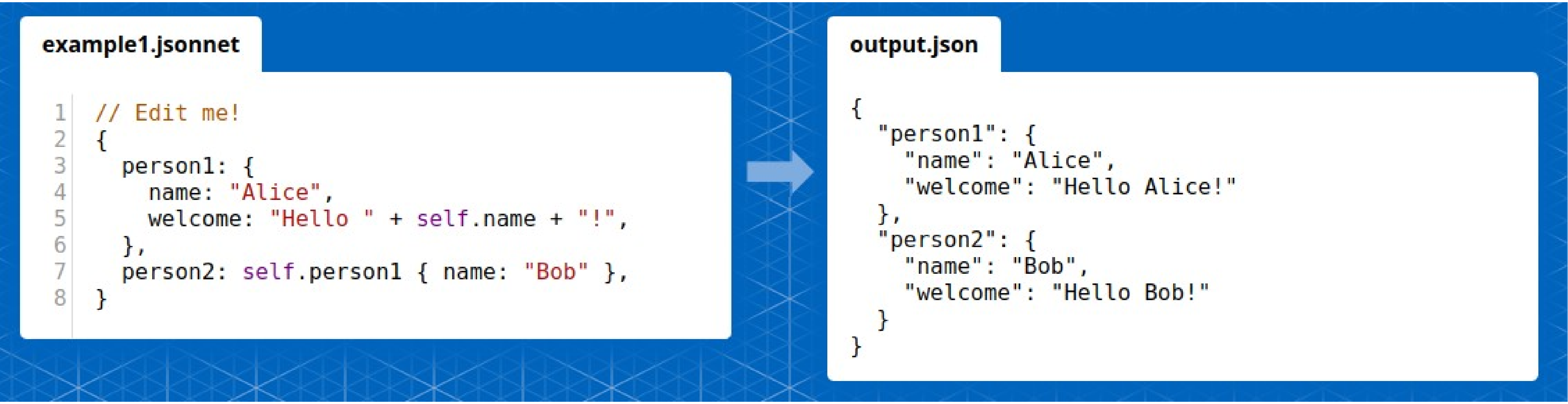
JSONNET

JSONNET
Variables
Conditionals
Arithmetics
Functions
Imports
Error Propagation

JSON
Arrays
Primitives
Objects

JSONNET - Examples

Eliminate duplication with object-orientation



example1.jsonnet

```
// Edit me!
{
  person1: {
    name: "Alice",
    welcome: "Hello " + self.name + "!",
  },
  person2: self.person1 { name: "Bob" },
}
```

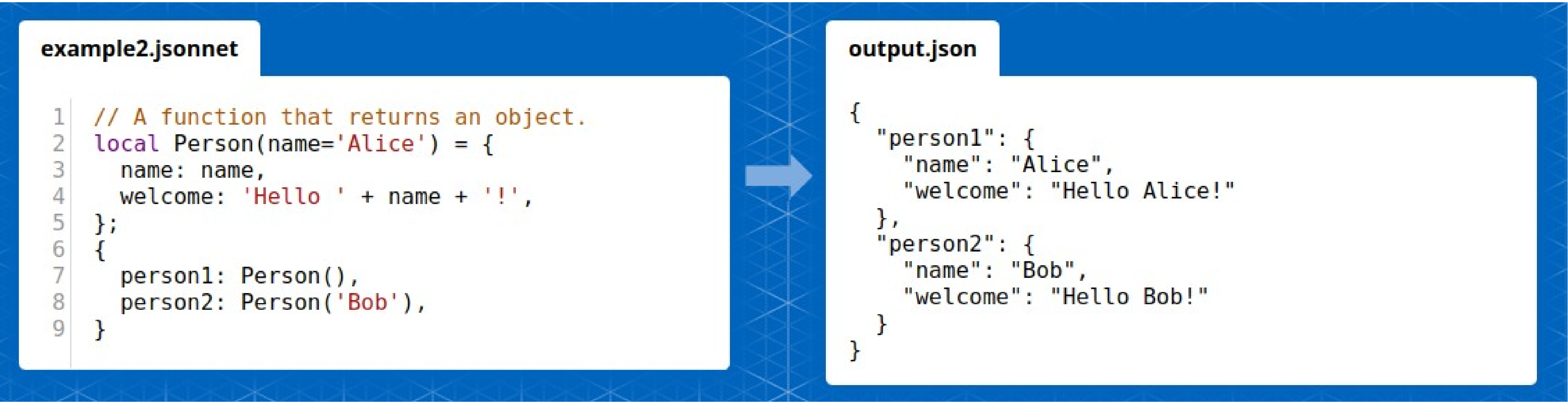
output.json

```
{
  "person1": {
    "name": "Alice",
    "welcome": "Hello Alice!"
  },
  "person2": {
    "name": "Bob",
    "welcome": "Hello Bob!"
  }
}
```

https://jsonnet.org/

JSONNET - Examples

Or, use functions



example2.jsonnet

```
// A function that returns an object.
local Person(name='Alice') = {
  name: name,
  welcome: 'Hello ' + name + '!',
};
{
  person1: Person(),
  person2: Person('Bob'),
}
```

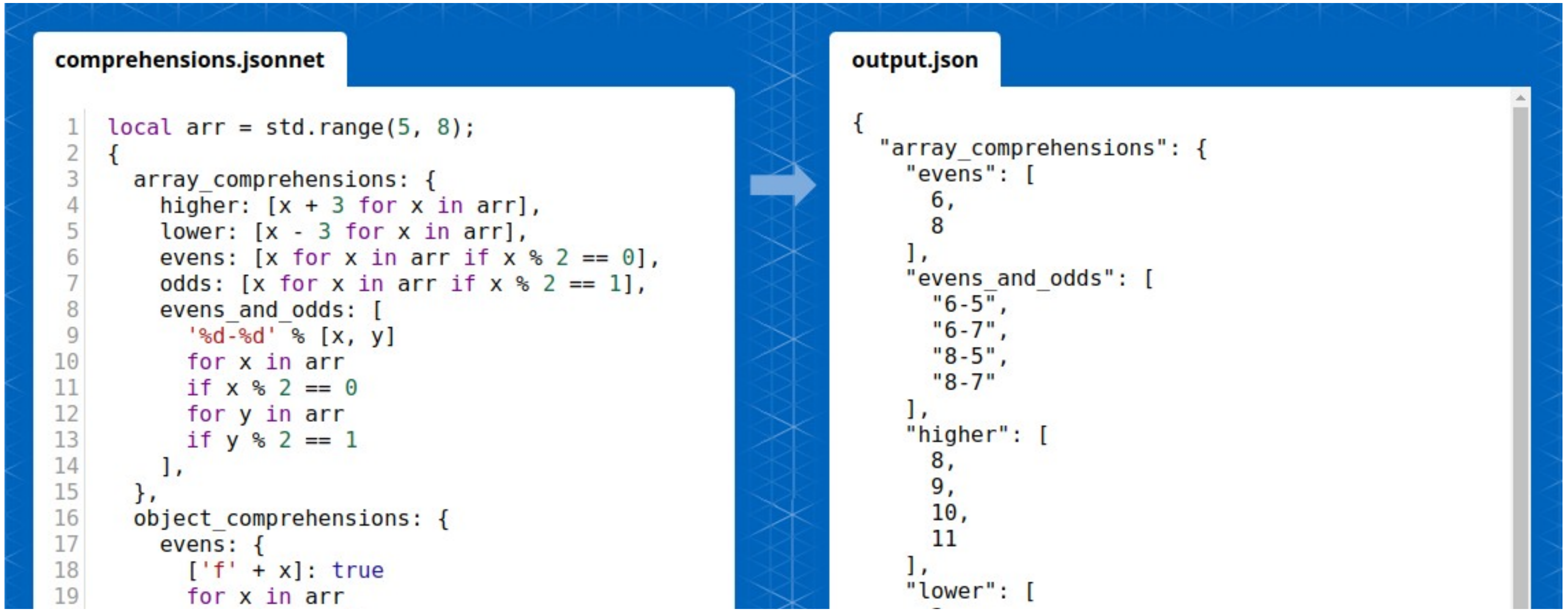
output.json

```
{
  "person1": {
    "name": "Alice",
    "welcome": "Hello Alice!"
  },
  "person2": {
    "name": "Bob",
    "welcome": "Hello Bob!"
  }
}
```

https://jsonnet.org/

JSONNET - Examples

Array/Object comprehensions, conditionals



comprehensions.jsonnet

```
local arr = std.range(5, 8);
{
  array_comprehensions: {
    higher: [x + 3 for x in arr],
    lower: [x - 3 for x in arr],
    evens: [x for x in arr if x % 2 == 0],
    odds: [x for x in arr if x % 2 == 1],
    evens_and_odds: [
      '%d-%d' % [x, y]
      for x in arr
      if x % 2 == 0
      for y in arr
      if y % 2 == 1
    ],
  },
  object_comprehensions: {
    evens: {
      ['f' + x]: true
      for x in arr
```

output.json

```
{
  "array_comprehensions": {
    "evens": [
      6,
      8
    ],
    "evens_and_odds": [
      "6-5",
      "6-7",
      "8-5",
      "8-7"
    ],
    "higher": [
      8,
      9,
      10,
      11
    ],
    "lower": [
```

https://jsonnet.org/

Now Lets Use JSONNET To Block Traffic Towards Internet But Allow Some FQDN›s

CiliumNetworkPolicy – default deny

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: default-deny-allow-dns
spec:
  egress:
    - toEndpoints:
        - matchLabels:
            io.kubernetes.pod.namespace: kube-system
            k8s-app: kube-dns
      toPorts:
        - ports:
            - port: "53"
              protocol: UDP
          rules:
            dns:
              - matchPattern: "*"
  endpointSelector:
    matchLabels: {}
  ingress:
    - {}
```

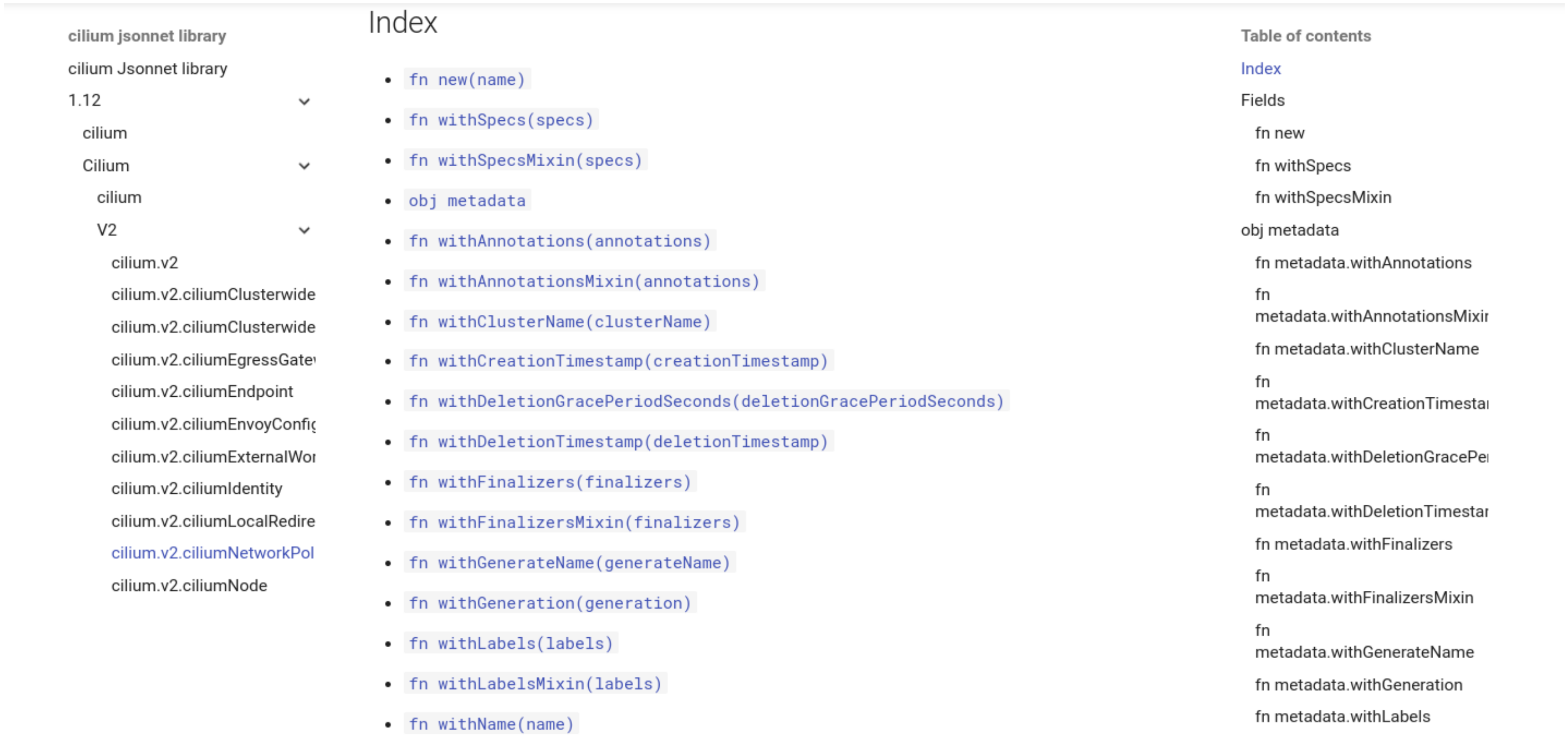
Enables Cilium DNS Proxy L7 Rule

CiliumNetworkPolicy

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-azure
Spec:
  endpointSelector:
    matchLabels: {}
  egress:
    - toFQDNs:
        - matchName: azure.microsoft.com
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP
```

JSONNET - Import

```
local cilium = import 'github.com/jsonnet-libs/cilium-libsonnet/1.12/main.libsonnet';
```



cilium jsonnet library

cilium Jsonnet library

1.12

cilium

Cilium

cilium

V2

cilium.v2

cilium.v2.ciliumClusterwide

cilium.v2.ciliumClusterwide

cilium.v2.ciliumEgressGate

cilium.v2.ciliumEndpoint

cilium.v2.ciliumEnvoyConfi

cilium.v2.ciliumExternalWo

cilium.v2.ciliumIdentity

cilium.v2.ciliumLocalRedire

cilium.v2.ciliumNetworkPol

cilium.v2.ciliumNode

Index

- `fn new(name)`
- `fn withSpecs(specs)`
- `fn withSpecsMixin(specs)`
- `obj metadata`
- `fn withAnnotations(annotations)`
- `fn withAnnotationsMixin(annotations)`
- `fn withClusterName(clusterName)`
- `fn withCreationTimestamp(creationTimestamp)`
- `fn withDeletionGracePeriodSeconds(deletionGracePeriodSeconds)`
- `fn withDeletionTimestamp(deletionTimestamp)`
- `fn withFinalizers(finalizers)`
- `fn withFinalizersMixin(finalizers)`
- `fn withGenerateName(generateName)`
- `fn withGeneration(generation)`
- `fn withLabels(labels)`
- `fn withLabelsMixin(labels)`
- `fn withName(name)`

Table of contents

CiliumNetworkPolicy

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-azure
Spec:
  endpointSelector:
    matchLabels: {}
  egress:
    - toFQDNs:
        - matchName: azure.microsoft.com
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP
```

JSONNET

```
egressPolicyFQDNwithPort(fqdn, port='443', l4proto='TCP'):

local toFQDNwithMatchName =
  if std.length(std.findSubstr('*', fqdn)) > 0 then
    cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.toFQDNs.withMatchPattern(fqdn)
  else
    cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.toFQDNs.withMatchName(fqdn);

local toFQDN =
  cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.withToFQDNs(toFQDNwithMatchName);
```

- Functions & Standard Libray functions
- Variables
- Conditionals

CiliumNetworkPolicy

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-azure
Spec:
  endpointSelector:
    matchLabels: {}
  egress:
    - toFQDNs:
        - matchName: azure.microsoft.com
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP
```

JSONNET

```
local port =
  cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.toPorts.ports.withPort(port);

local protocol =
  cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.toPorts.ports.withProtocol(l4proto);

local port_proto =
  cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.toPorts.withPorts(toPort +
protocol);

local toPorts =
  cilium.cilium.v2.ciliumNetworkPolicy.spec.egress.withToPorts(port_proto);
```

JSONNET

```
// Combine port and FQDN Rule
toPorts + toFQDN,
```

- Arithmetic

CiliumNetworkPolicy

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-azure
Spec:
  endpointSelector:
    matchLabels: {}
  egress:
    - toFQDNs:
        - matchName: azure.microsoft.com
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP
```

JSONNET

```
  // Create a Network Policy with mutlitple toFQDN Rules
  generateNP(name, fqdns=[]):
    local endpointSelector =
cilium.cilium.v2.ciliumNetworkPolicy.spec.endpointSelector.withMatchLabels({});

    local egresses = [
      self.egressPolicyFQDNwithPort(fqdn)
      for fqdn in fqdns
    ];

    cilium.cilium.v2.ciliumNetworkPolicy.new(name) {
      spec: {
        ingress: [{}],
        egress: egresses,
      },
    } + endpointSelector,
```

CiliumNetworkPolicy

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-azure
Spec:
  endpointSelector:
    matchLabels: {}
  egress:
    - toFQDNs:
        - matchName: azure.microsoft.com
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP
```

Now let›s use these functions

JSONNET

```
local allowedBase = [

  {
    name: 'allow-azure',
    allowedFQDNs: ['azure.microsoft.com'],
  }
]
```

- Variables

JSONNET

```
// Output

{
  ['base/' + policy.name + '.json']:
policyGenerator.generateNP(policy.name, policy.allowedFQDNs)
  for policy in allowedBase
}
```

- Object Comprehension
- Function

JSONNET install

3

```
go install
github.com/google/go-jsonnet/cmd/jsonnet@latest

go install -a
github.com/jsonnet-bundler/jsonnet-bundler/cmd/jb@la
test

jb init
jb install github.com/jsonnet-libs/cilium-
libsonnet/1.12/@main
```

Generate Network Policies

3

```
jsonnet -J vendor -m ./networkpolicies -c  -y
networkpolicies.jsonnet

# Rewrite to YAML 🙈
for file in $(find ./networkpolicies -type f -name
"*.json" -print); do; yj --json --yaml $file > $
(echo $file | sed "s/json/yaml/") && rm $file; done
```

Do you mind a bit more?

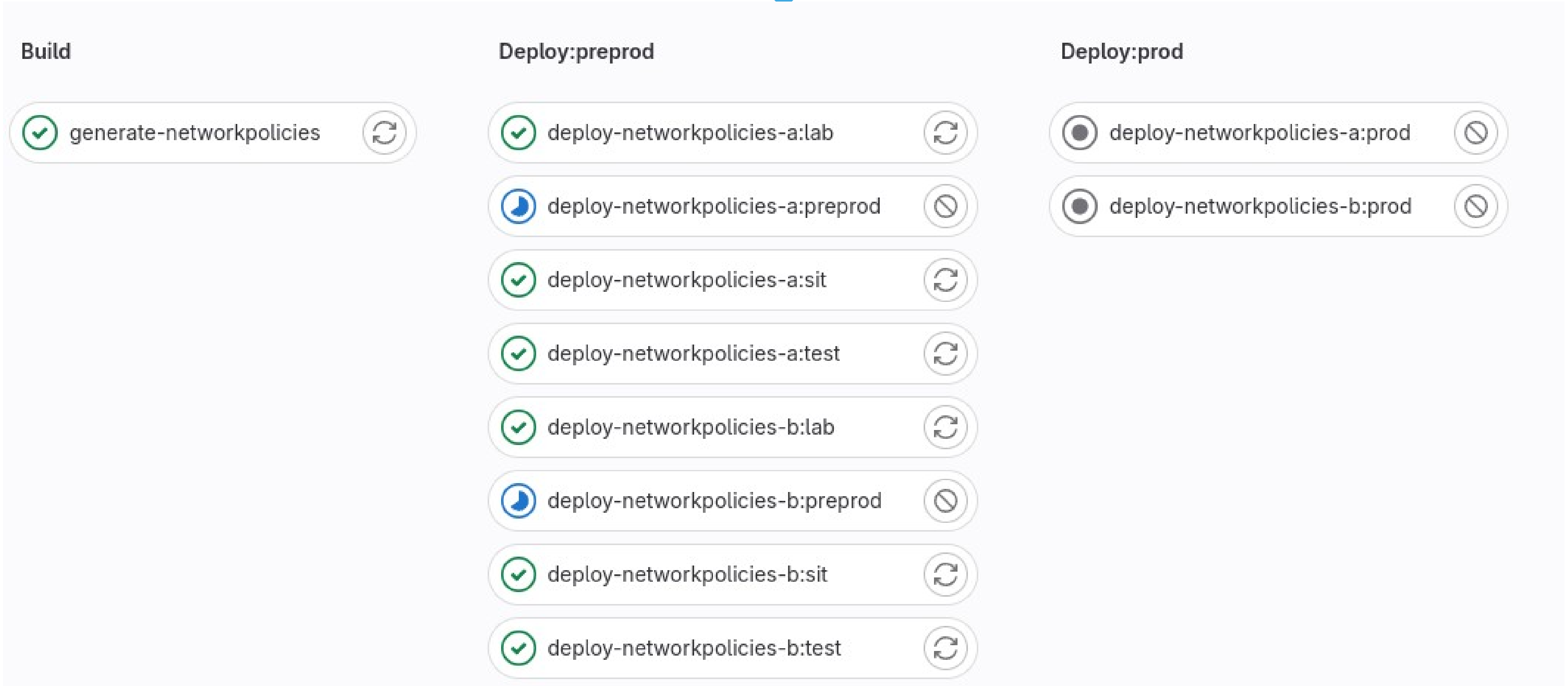
- Bash magic to deploy
- Kustomize with
  - Base
  - Overlays for rules only used by a few applications
  - kustomization.yaml written with JSONNET

Do you mind a bit more?

- Common label contains checksum of the current config
  - Allows for easier deletion of old rules

```
kubectl -n $NAMESPACE delete
ciliumnetworkpolicies.cilium.io
--selector=checksum!=$CHECKSUM
```

Pipeline



Build

generate-networkpolicies

Deploy:preprod

deploy-networkpolicies-a:lab

deploy-networkpolicies-a:preprod

deploy-networkpolicies-a:sit

deploy-networkpolicies-a:test

deploy-networkpolicies-b:lab

deploy-networkpolicies-b:preprod

deploy-networkpolicies-b:sit

deploy-networkpolicies-b:test

Deploy:prod

deploy-networkpolicies-a:prod

deploy-networkpolicies-b:prod

Merci!

Mehr Informationen zu Puzzle:

http://www.puzzle.ch

@puzzleitc

github.com/puzzle