Developer Days
Automation

The bridge to possible

Good Practices in NSO Development

Blaž Dolenc
Software Engineer @Flint SI

“Dobra priprava na delo je polovica uspeha.”

Slovenian proverb

“Good preparation before work is half the success.”

Slovenian proverb

Me working on my first NSO project, circa 2016, colorized

The problem?

- Adhoc test environments
- No control over versions of NSO, dependencies and software
- Testing just on netsim
- No CI/CD

What to do to make NSO development a delightful experience?

$ whoami

- Blaž Dolenc, Software Engineer @ Flint SI
- Working as consultant @Deutsche Telekom for the past 5 years
- CCSI delivering and developing NSO trainings and other learning materials
- Reach out blaz.dolenc@flintmail.com

Tip #1
Lint your code

Lint your code

loopback/src/Makefile

```
all: fxs pylint

...

pylint:
    pylint --rcfile=.pylintrc ../python/loopback
```

```
pylint --generate-rcfile > loopback/src/.pylintrc
```

Lint your code

loopback/src/Makefi
le

```
all: fxs pylint

...

pylint:
    pylint --rcfile=.pylintrc ../python/loopback
```

```
pylint --generate-rcfile > loopback/src/.pylintrc
```

Lint your code

```
developer:~ > make -C ~/loopback/src/ all
make: Entering directory '~/loopback/src'pylint --rcfile=.pylintrc ../python/loopback*************
Module loopback.loopback~/loopback/python/loopback/loopback.py:1:0: C0114: Missing module docstring
(missing-module-docstring)~/loopback/python/loopback/loopback.py:6:0: C0115: Missing class docstring
(missing-class-docstring)~/loopback/python/loopback/loopback.py:8:4: C0116: Missing function or
method docstring (missing-function-docstring)~/loopback/python/loopback/loopback.py:20:8: W0622:
Redefining built-in 'vars' (redefined-builtin)~/loopback/python/loopback/loopback.py:9:49: W0212:
Access to a protected member _path of a client class (protected-
access)~/loopback/python/loopback/loopback.py:29:0: C0115: Missing class docstring (missing-class-
docstring)~/loopback/python/loopback/loopback.py:3:0: C0411: standard import "import ipaddress"
should be placed before "import ncs" (wrong-import-order)-------------------------------------------
-----------------------Your code has been rated at 6.96/10 (previous run: 6.96/10, +0.00)make: ***
[Makefile:32: pylint] Error 20
make: Leaving directory '~/loopback/src'
```

Lint your code

```
developer:~ > make -C ~/loopback/src/ all
make: Entering directory '~/loopback/src'pylint --rcfile=.pylintrc ../python/loopback************
Module loopback.loopback~/loopback/python/loopback/loopback.py:1:0: C0114: Missing module docstring
(missing-module-docstring)~/loopback/python/loopback/loopback.py:6:0: C0115: Missing class docstring
(missing-class-docstring)~/loopback/python/loopback/loopback.py:8:4: C0116: Missing function or
method docstring (missing-function-docstring)~/loopback/python/loopback/loopback.py:20:8: W0622:
Redefining built-in 'vars' (redefined-builtin)~/loopback/python/loopback/loopback.py:9:49: W0212:
Access to a protected member _path of a client class (protected-
access)~/loopback/python/loopback/loopback.py:29:0: C0115: Missing class docstring (missing-class-
docstring)~/loopback/python/loopback/loopback.py:3:0: C0411: standard import "import ipaddress"
should be placed before "import ncs" (wrong-import-order)---------------------------------------------
-----------------------Your code has been rated at 6.96/10 (previous run: 6.96/10, +0.00)make: ***
[Makefile:32: pylint] Error 20
make: Leaving directory '~/loopback/src'
```

Lint your code

```
developer:~ > make -C ~/loopback/src/ all
make: Entering directory '~/loopback/src'pylint --rcfile=.pylintrc ../python/loopback************
Module loopback.loopback~/loopback/python/loopback/loopback.py:1:0: C0114: Missing module docstring
(missing-module-docstring)~/loopback/python/loopback/loopback.py:6:0: C0115: Missing class docstring
(missing-class-docstring)~/loopback/python/loopback/loopback.py:8:4: C0116: Missing function or
method docstring (missing-function-do                                                          2:
Redefining built-in 'vars' (redefined                                                          12:
Access to a protected member _path o
access)~/loopback/python/loo                                                                   ass-
docstring)~/loopback/python/loopback/                                                          '
should be placed before "import ncs"                                                           ------
------------------------Your code has                                                          ***
[Makefile:32: pylint] Error 20
make: Leaving directory '~/loopback/s
```

```python
import ncs
from ncs.application import Service
import ipaddress
```

```python
import ipaddress
import ncs
from ncs.application import Service
```

Tip #2

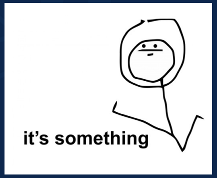
Invest in development and test environments

What to choose?

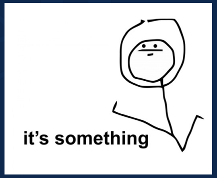
Netsim

Netsim	

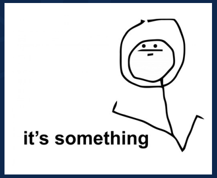
Netsim

Netsim	

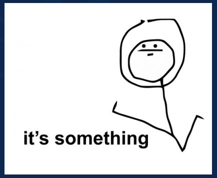
Physical devices

Netsim	
Physical devices	

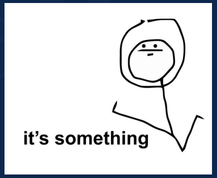
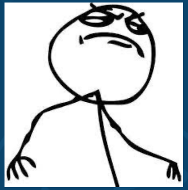
Physical devices

Netsim	
Physical devices	

Virtual routers

Netsim	
Vrnetlab/containerlab	
Physical devices	

Virtual routers

Netsim	
Vrnetlab/containerlab	
Physical devices	

Test with virtual routers

testenvs

```
full
full-netsim
quick
quick-netsim
```

Test environments

testenvs

```
full
    service-config.xml
    topology.json
    test.mk
full-netsim
quick
quick-netsim
```

Test environments

```
testenv
s
full
    service-config.xml
    topology.json
    test.mk
full-netsim
quick
quick-netsim
```

```
<config >
  <loopback xmlns="http://com/example/loopback">
      <name>loopback0</name>
      <device>example-device</device>
      <management-intf>1</management-intf>
       <management-prefix>192.168.1.1/32</manage
      <bgp-intf>2</bgp-intf>
      <bgp-prefix>10.0.0.1/32</bgp-prefix>
  </loopback>
</config>
```

Test environment

testenv
s

```
full
    service-config.xml
    topology.json
    test.mk
full-netsim
quick
quick-netsim
```

```
{ "routers": {
        "PE-1": {
                "function": "PE",
                "type": "vmx",
                "docker_network": "testenv-network
        },
        "CPE-1": {
                "function": "CPE",
                "type": "openwrt",
                "docker_network":
        }
 "p2p": {
        "PE-1": [ "CPE-1" ]
        }
   }
}
```

Test environments

```
testenv
s
full
    service-co
    topology.j
    test.mk
full-netsim
quick
quick-netsim
```

```
start:
    docker run -d --name PE-1 --privileged vr-vmx:R2.23
    docker run -d --name CPE-1 --privileged vr-wrt:11.2
    docker run -d --name vr-xcon --link PE-1 --link CPE-1
                             vr-xcon --p2p PE-1/1--CPE-1/1
    docker run -d --name NSO nso:6.1

configure:
     $(MAKE) loadconf FILE=service-config.xml

test:
    $(MAKE) test-service-plan
    $(MAKE) test-ping
```

Developer workflow

```
~$ git clone git@github.com:/nso-service-dev-practices.git
~$ cd nso-service-dev
~$ make build
~$ make -C testenvs/full start configure
```

Developer workflow

```
~$ git clone git@github.com:/nso-service-dev-practices.git
~$ cd nso-service-dev
~$ make build
~$ make -C testenvs/full start configure
```

```
NSO image with
local packages
```

Developer workflow

```
~$ git clone git@github.com:/nso-service-dev-practices.git
~$ cd nso-service-dev
~$ make build
~$ make -C testenvs/full start configure
```

NSO image with local packages

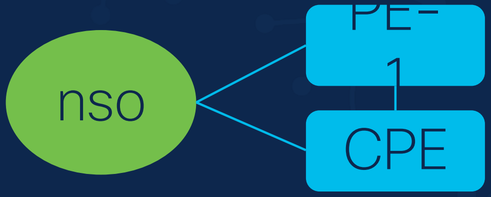
Pull in vrnetlab images and start topology

Developer workflow

```
~$ git clone git@github.com:/nso-service-dev-practices.git
~$ cd nso-service-dev
~$ make build
~$ make -C testenvs/full start configure
```

NSO image with local packages

Pull in vrnetlab images and start topology



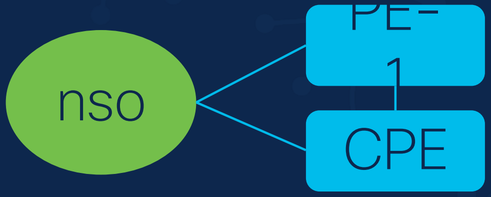
Developer workflow

```
~$ git clone git@github.com:/nso-service-dev-practices.git
~$ cd nso-service-dev
~$ make build
~$ make -C testenvs/full start configure
```

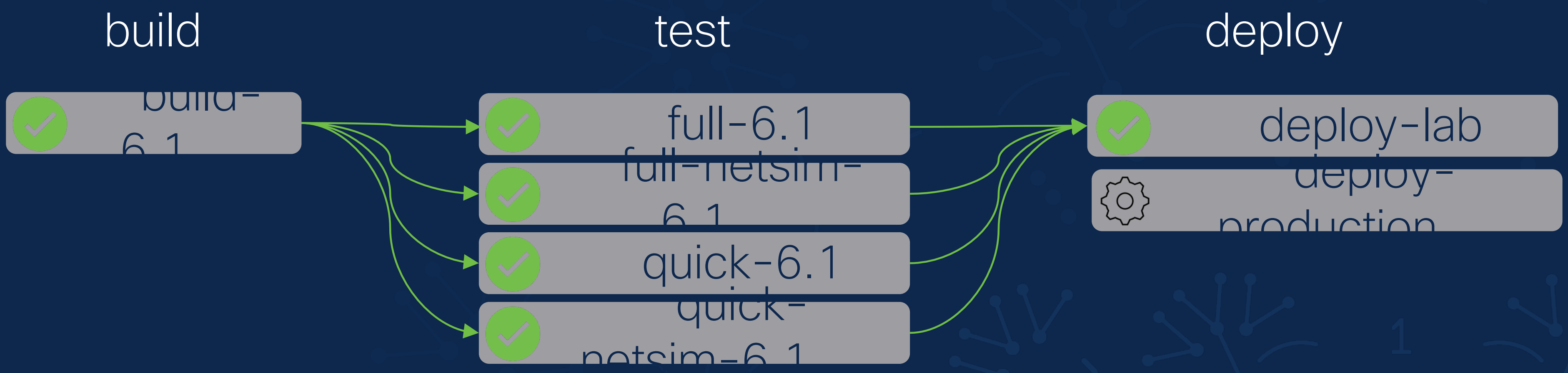
NSO image with local packages

Pull in vrnetlab images and start topology

Configure with test service instances



CI/CD workflow



Want to learn more? https://gitlab.com/nso-developer/nso-docker

Tip #3
Add device configurations to git

Add expected device configurations

```
testenvs
full
    service-config.xml
    topology.json
    test.mk
    expected/
    output/
full-netsim
quick
quick-netsim
```

Add expected device configurations

```
testenv
s
full
    service-conf
    topology.jso
    test.mk
    expected/
    output/
full-netsim
quick
quick-netsim
```

```
configure:
     $(MAKE) loadconf FILE=service-config.xml

test:
    $(MAKE) test-service-plan
    $(MAKE) test-ping
    $(MAKE) save-output

save-output:
    $(MAKE) saveconf FILE=output/devices.xml "devices device"

check-diff:
    diff -c expected/ output/
```

Add expected device configurations

```
testenv
s
full
    service-config.xml
    topology.json
    test.mk
    expected/
    output/devices.xml
full-netsim
quick
quick-netsim
```

```
<devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>PE-1</name>
    <authgroup>PE-1</authgroup>
    <device-type>
      <netconf>
        <ned-id xmlns:juniper-junos</ned-id>
      </netconf>
    </device-type>
    <config>
      <configuration
xmlns="http://xml.juniper.net/xnm/1.1/xnm">
        <apply-groups>admin</apply-groups>
        <interfaces>
          <interface>
            <name>ge-0/0/0</name>
            <description>Backbone
interface</description>
            <mtu>4400</mtu>
```

Add expected device configurations

testenv
s

```
full
    service-confi
    topology.js
    test.mk
    expected/devices.xml
    output/devices.xml
full-netsim
quick
quick-netsim
```

verified working configuration

current test run

Check diff

testenv
s

```
full
    servi
    topol
    test.mk
    expected/devices.xml
    output/devices.xml
full-netsim
quick
quick-netsim
```

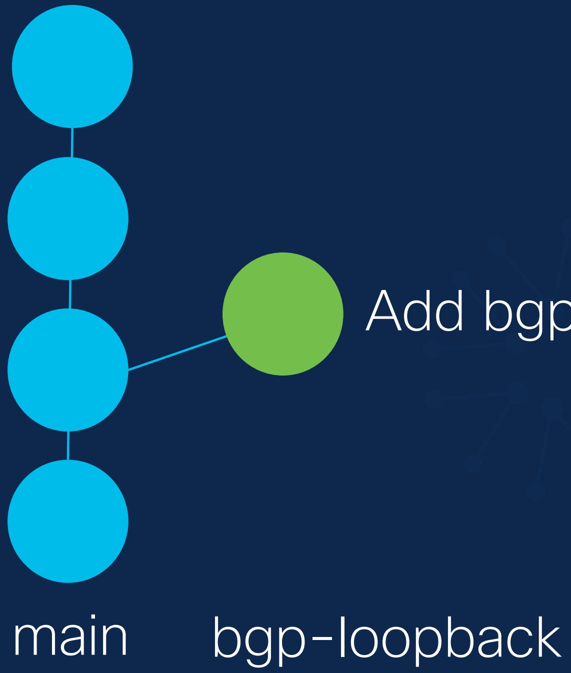
verified working configuration under git

current test run not in git

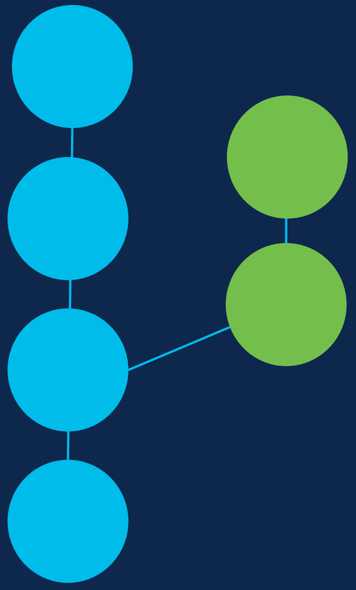
diff

```
<Loopback>
   <id>{/bgp-intf}</id>
-      <ipv4>
-        <address>
-          <ip>{$BGP_ADDRESS}</ip>
-        <mask>255.255.255.255</mask>
-        </address>
-        </ipv4>
</Loopback>
```

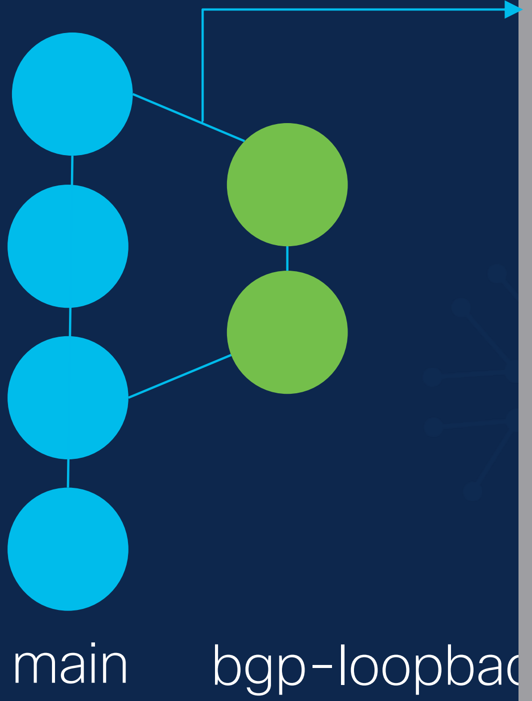
It gets even better



It gets even better



Merge request



Merge request

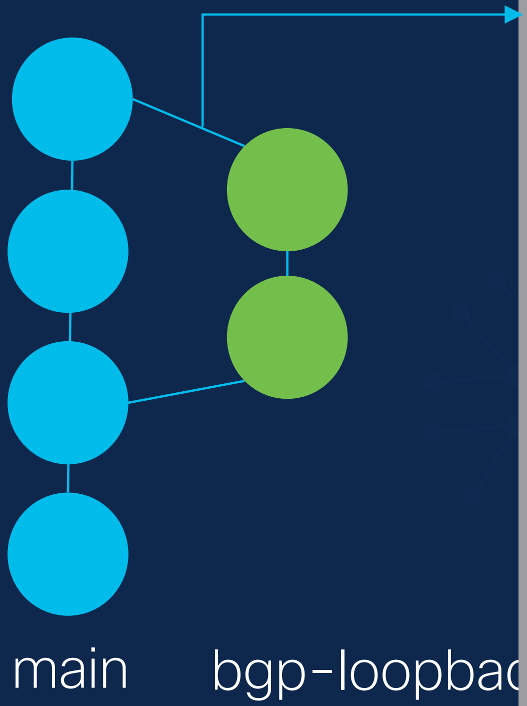
packages/loopback/python/loopback.py

```
+   bgp_prefix = service.bgp_prefix
+   self.log.debug(f‘bgp-prefgix leaf is {bgp_prefix}‘)
+   net =ipaddress.IPv4Network(bgp_prefix)
```

testenvs/full/expected/devices.xml

```
<Loopback>
   <id>{/bgp-intf}</id>
+     <ipv4>
+           <address>
+           <ip>{$BGP_ADDRESS}</ip>
+           <mask>255.255.255.255</mask>
+           </address>
+           </ipv4>
</Loopback>
```

Device configuration is reviewed



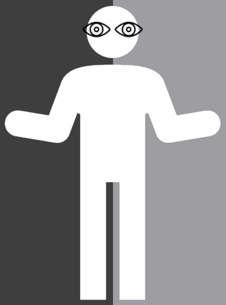
Merge request

packages/loopback/python/loopback.py

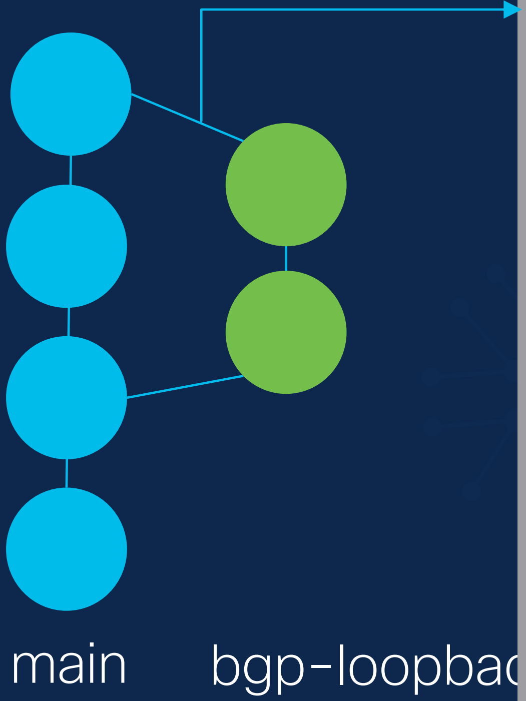
```
+   bgp_prefix = service.bgp_prefix
+   self.log.debug(f'bgp-prefgix leaf is {bgp_prefix}')
+   net =ipaddress.IPv4Network(bgp_prefix)
```

testenvs/full/expected/devices.xml

```
<Loopback>
   <id>{/bgp-intf}</id>
+     <ipv4>
+           <address>
+           <ip>{$BGP_ADDRESS}</ip>
+           <mask>255.255.255.255</mask>
+           </address>
+           </ipv4>
</Loopback>
```



Device configuration is reviewed



Merge request

packages/loopback/python/loopback.py

```
+   bgp_prefix = service.bgp_prefix
+   self.log.debug(f‘bgp-prefgix leaf is {bgp_prefix}‘)
+   net =ipaddress.IPv4Network(bgp_prefix)
```

testenvs/full/expected/devices.xml

```
<Loopback>
   <id>{/bgp-intf}</id>
+     <ipv4>
+           <address>
+           <ip>{$BGP_ADDRESS}</ip>
+           <mask>255.255.255.255</mask>
+           </address>
+           </ipv4>
</Loopback>
```

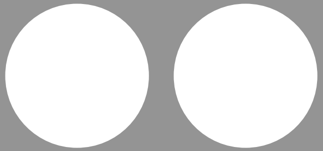
This will burn down our network please fix

Tip #4
Do not reinvent the wheel

NSO service package

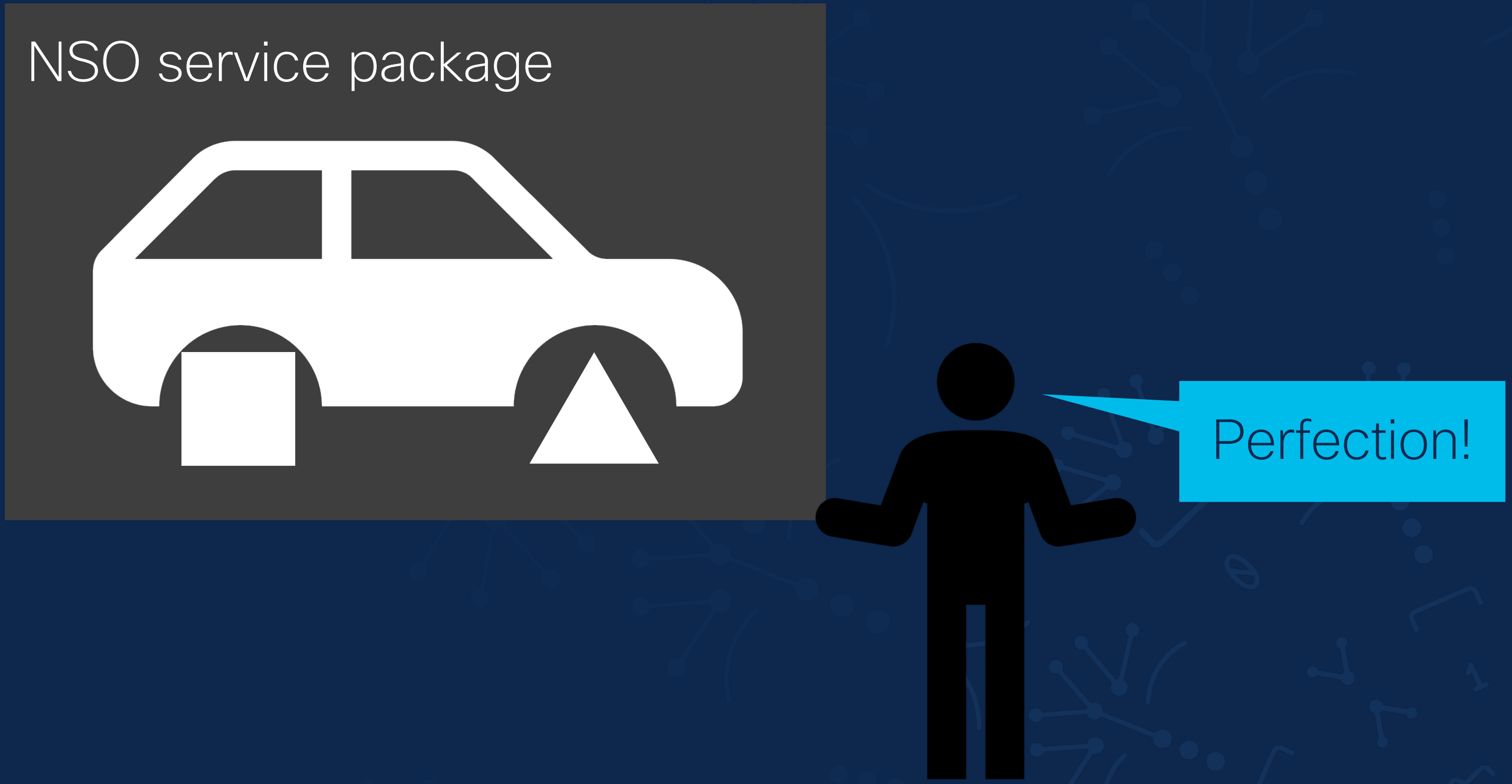
NSO service package

NSO device-automaton



NSO service package
NSO device-automaton
This is to complex, I will develop my own thing

NSO service package



device-automaton

- Add devices to the NSO in a declarative way
- Multiple management-endpoint support
- device type and NED detection
- configuration sync-management and more
- *https://gitlab.com/nso-developer/device-automaton*

bgworker

- Run background worker processes in NSO
- Periodical polling of devices for operational state
- Checking if services are in sync
- HA, restarts, config changes all handled!
- *https://gitlab.com/nso-developer/bgworker*

nso-docker

- Everything you need for running NSO in Docker
- Development and CI testing
- Skeletons for building NEDs, packages and projects
- *https://gitlab.com/nso-developer/nso-docker*

Good NSO service development practices LAB

https://github.com/NSO-developer/nso-service-dev-practices

NSO
NSO

The bridge to possible