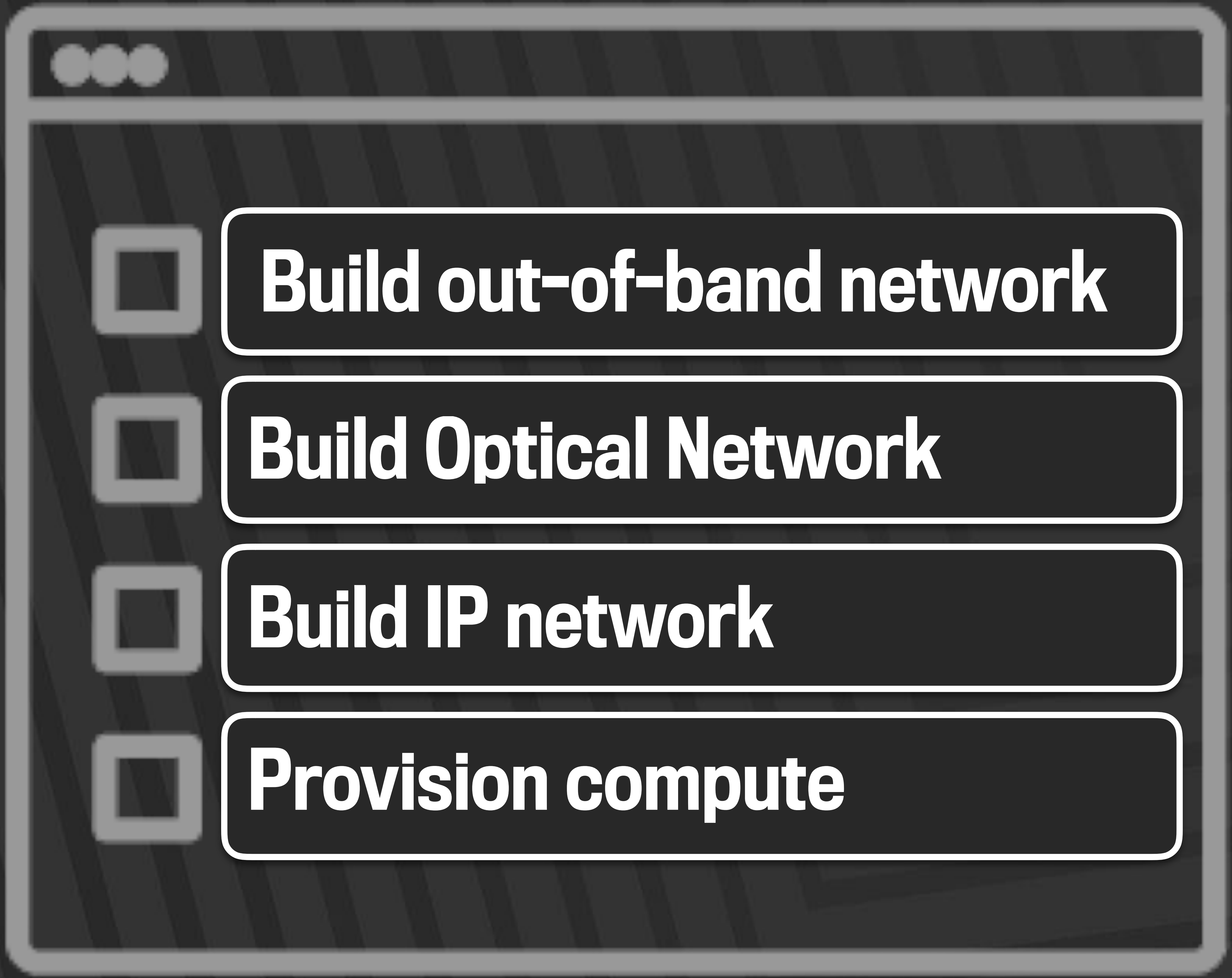
Scaling the Facebook backbone through Zero Touch Provisioning (ZTP)

David Swafford

Network Engineer

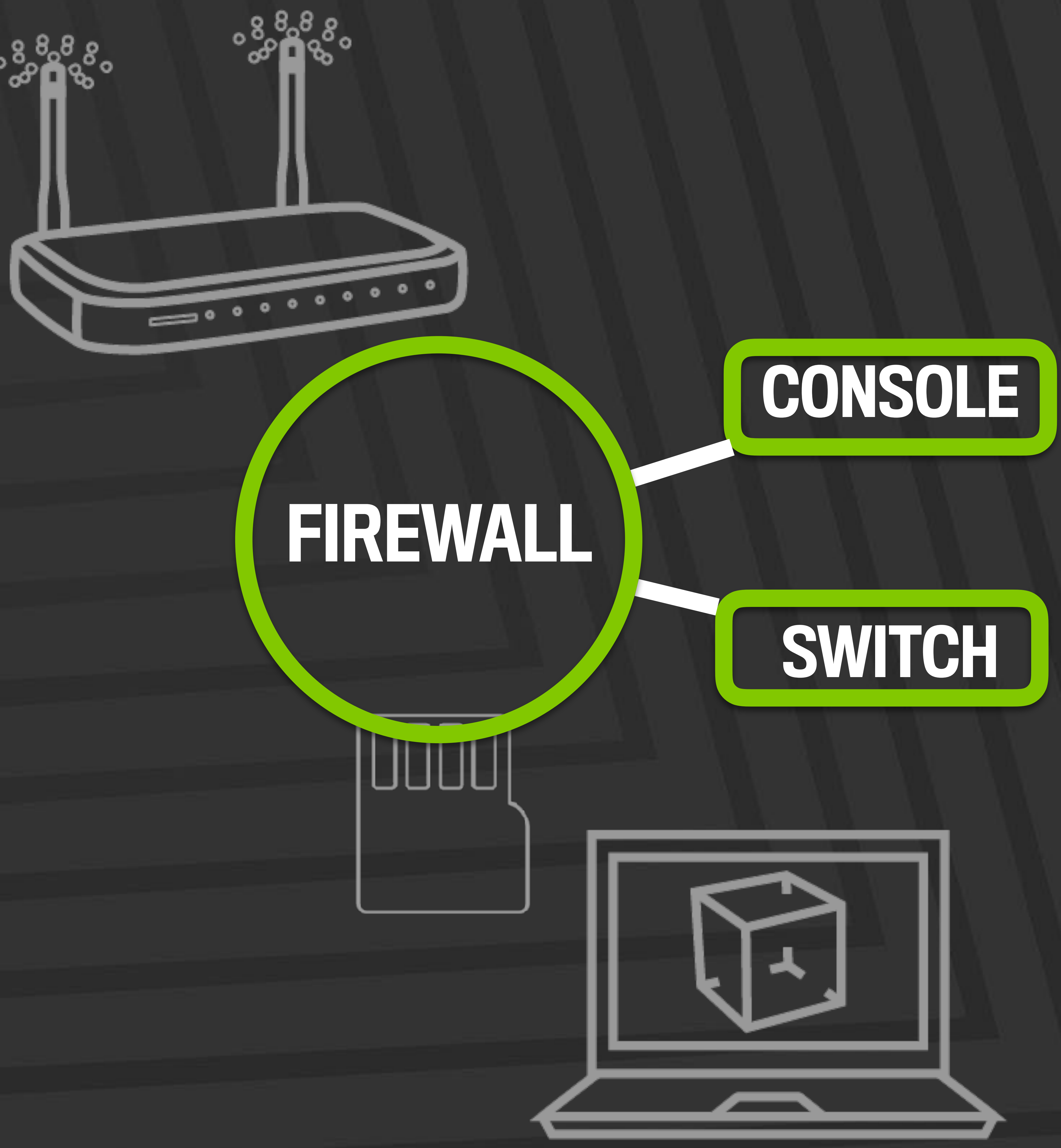
PANDUIT

Building a new POP (Point of Presence)



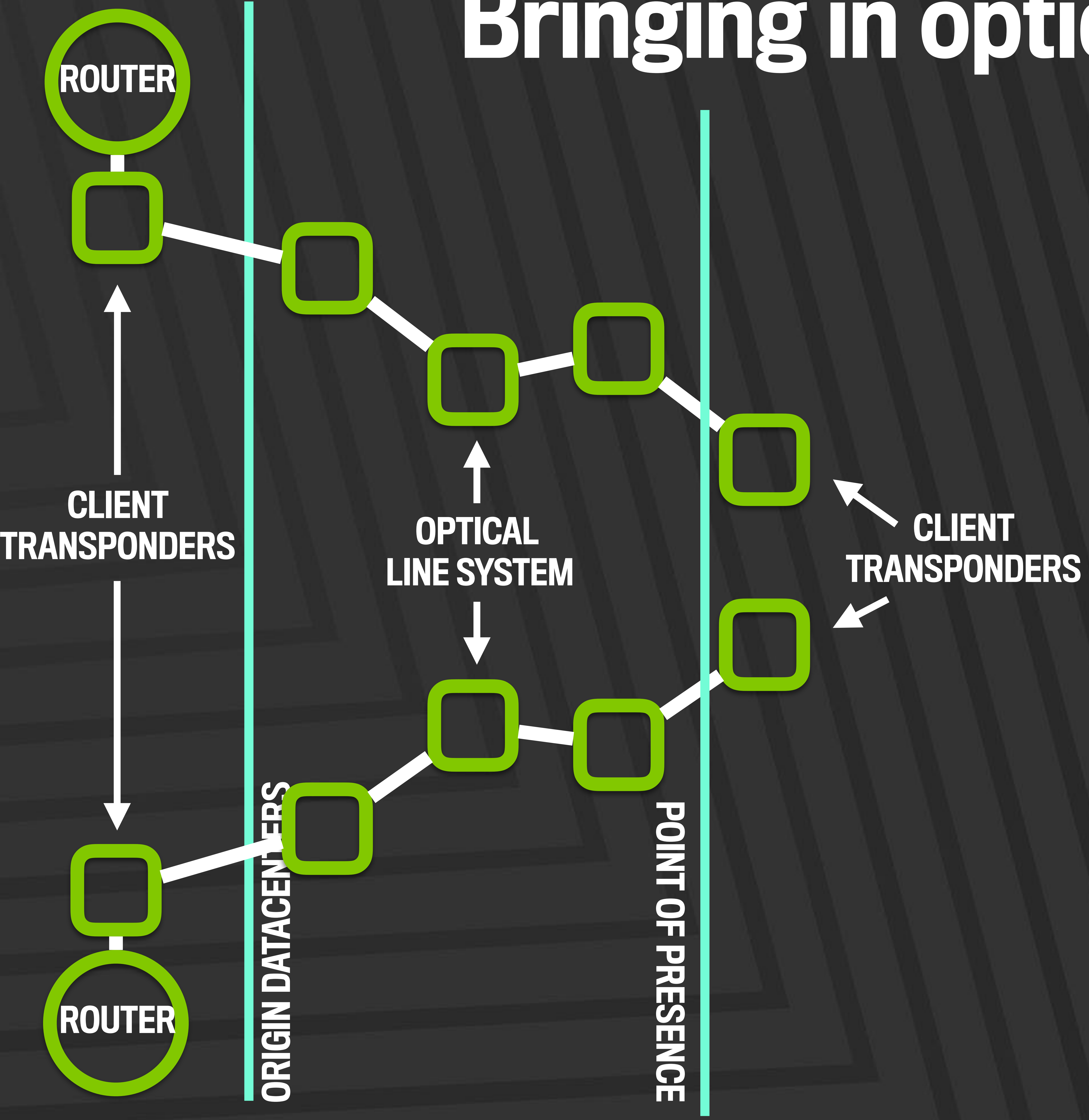
- Build out-of-band network
- Build Optical Network
- Build IP network
- Provision compute

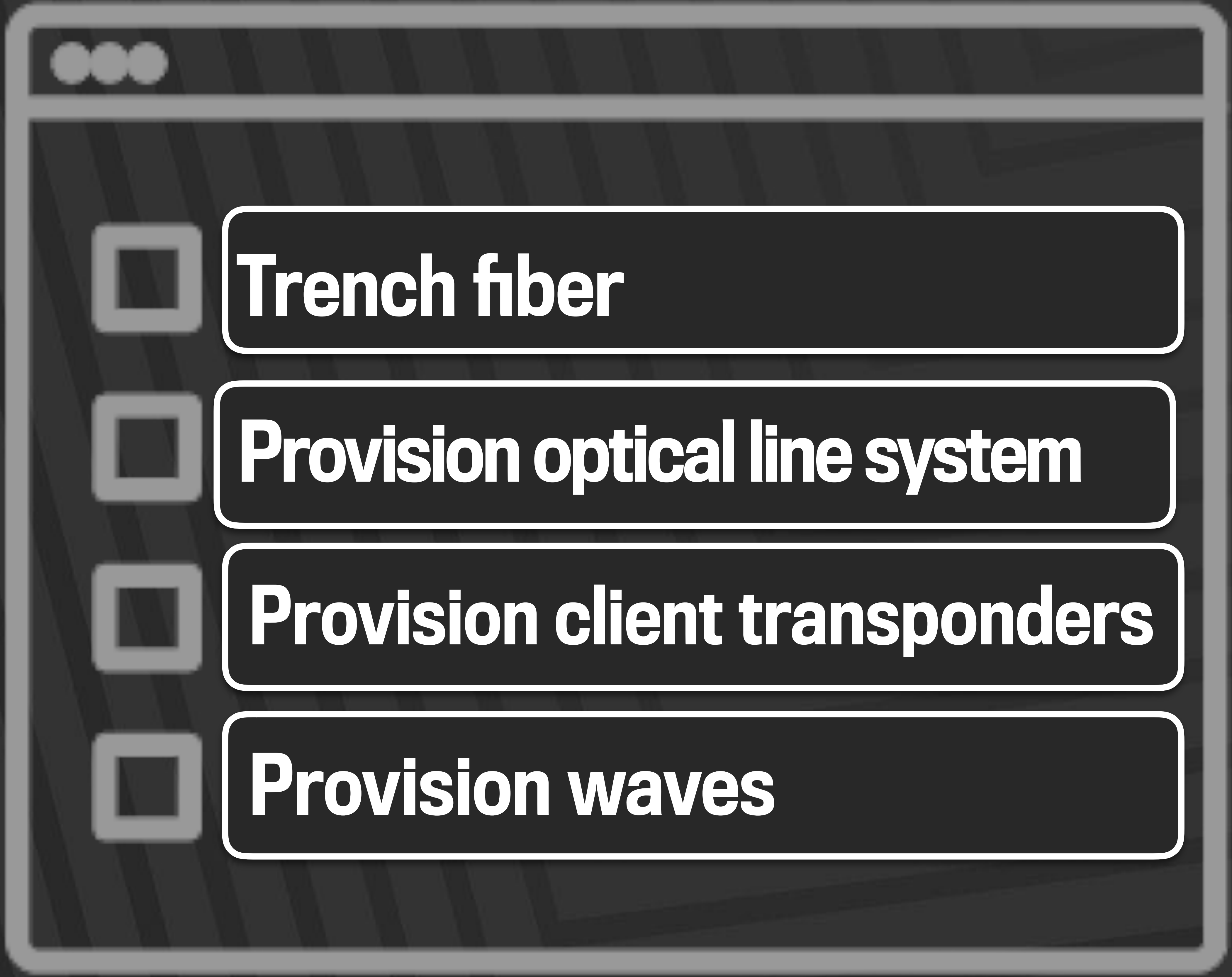
Building the out-of-band network



- [ ] Install Internet service
- [ ] Provision firewall
- [ ] Provision console servers
- [ ] Provision management switches

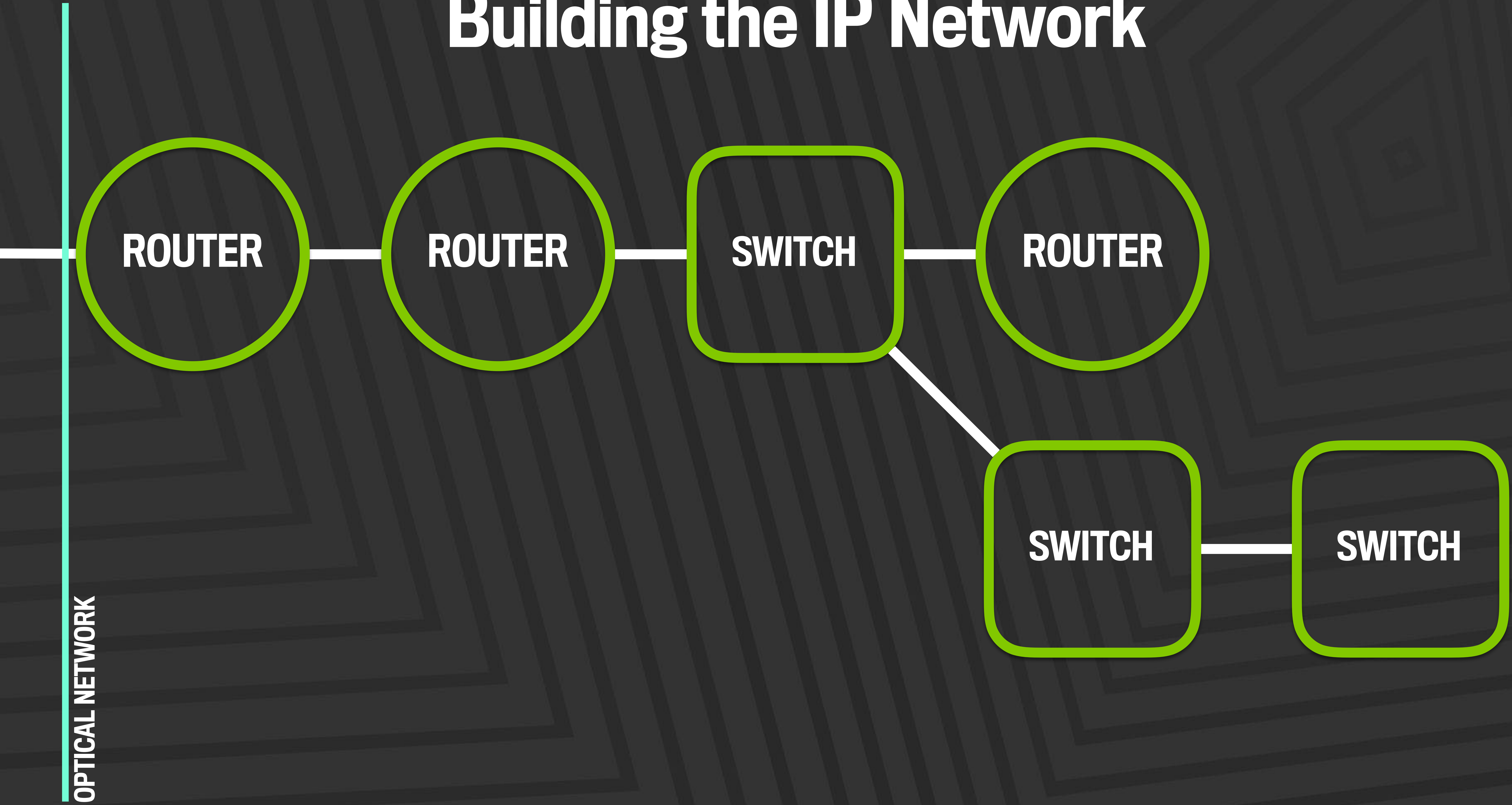
Bringing in optical connectivity



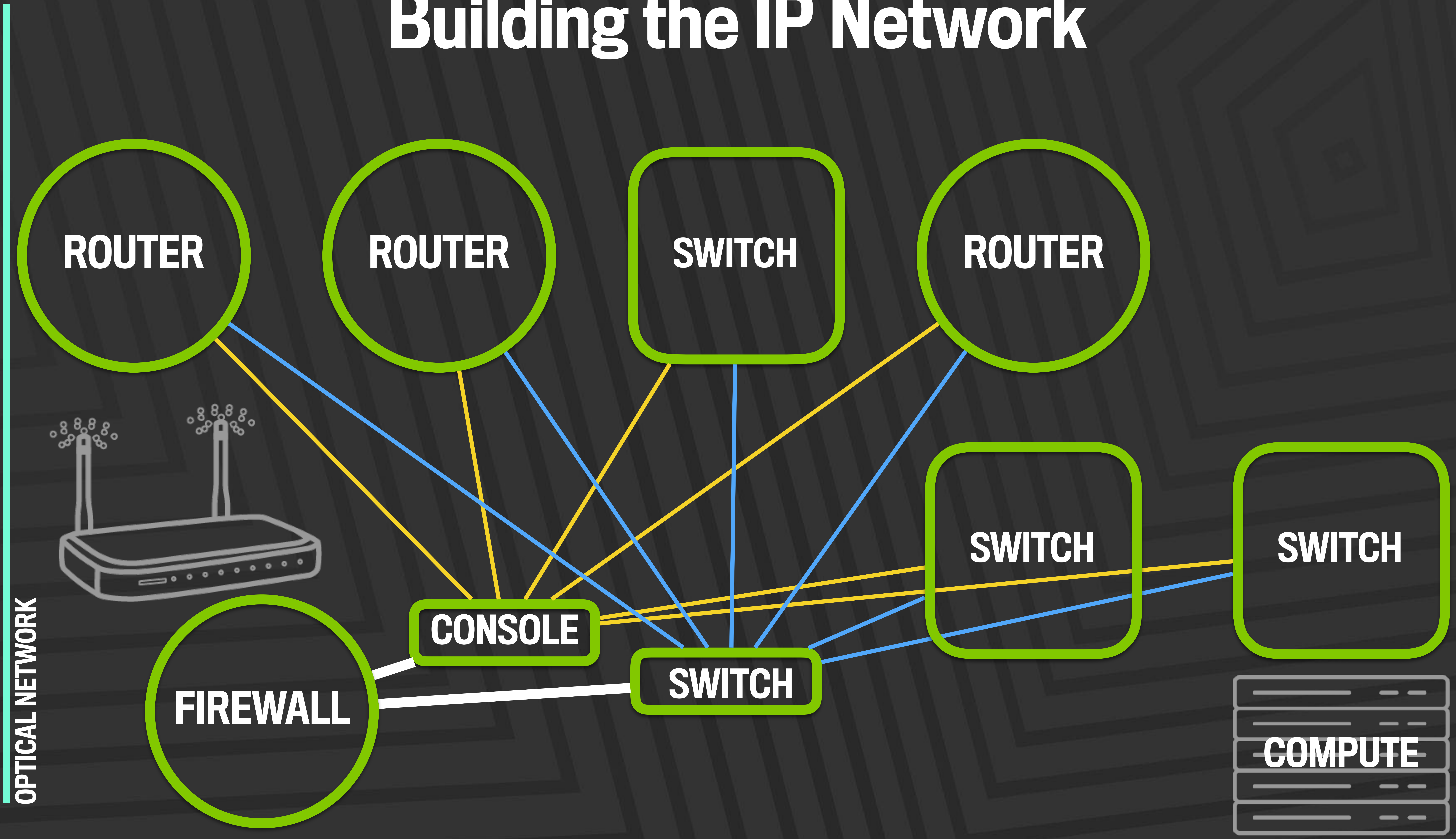


- [ ] Trench fiber
- [ ] Provision optical line system
- [ ] Provision client transponders
- [ ] Provision waves

Building the IP Network



Building the IP Network



Provisioning one of our edge routers

- Letting People Know
- Rack and Stack
- Cabling
- Management IP assignment
- Config Generation
- Software Upgrades
- Loading Config
- Validating Config
- Validating Hardware (Fans, Power Supplies, Linecards)
- Validating Physical Connectivity (LLDP and Light Levels)
- Validating Logical Connectivity (Protocols)
- Updating External Systems (Location Data, Status)
- Undraining Traffic

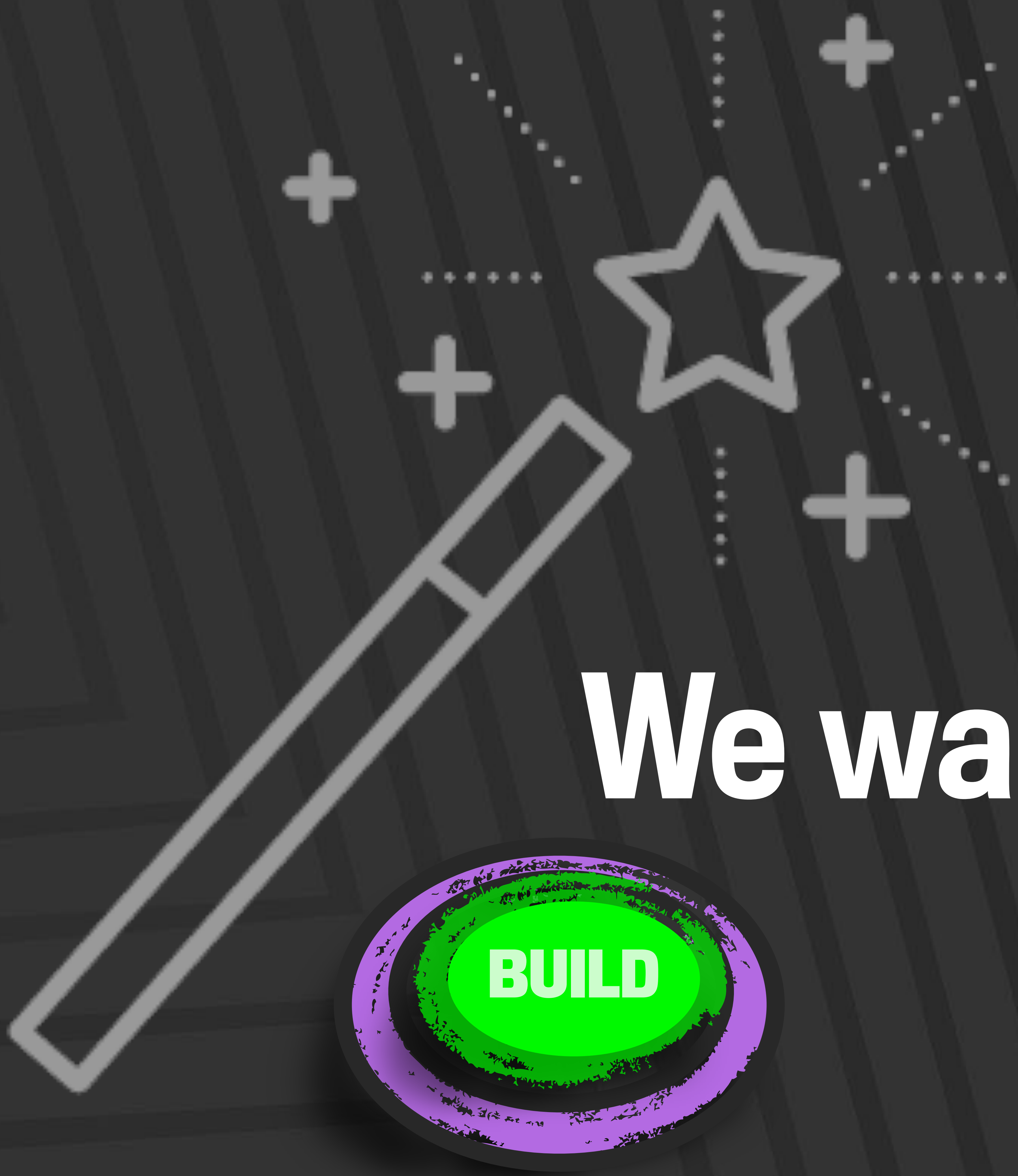
What was already solved?

- Letting People Know
- Rack and Stack
- Cabling
- Management IP assignment
- Config Generation
- Software Upgrades
- Loading Config
- Validating Config
- Validating Hardware (Fans, Power Supplies, Linecards)
- Validating Physical Connectivity (LLDP and Light Levels)
- Validating Logical Connectivity (Protocols)
- Updating External Systems (Location Data, Status)
- Undraining Traffic

30 steps involving 10+ tools...

MOPs?

We wanted push button!



Major Pieces Needed

A method to quickly and reliably:

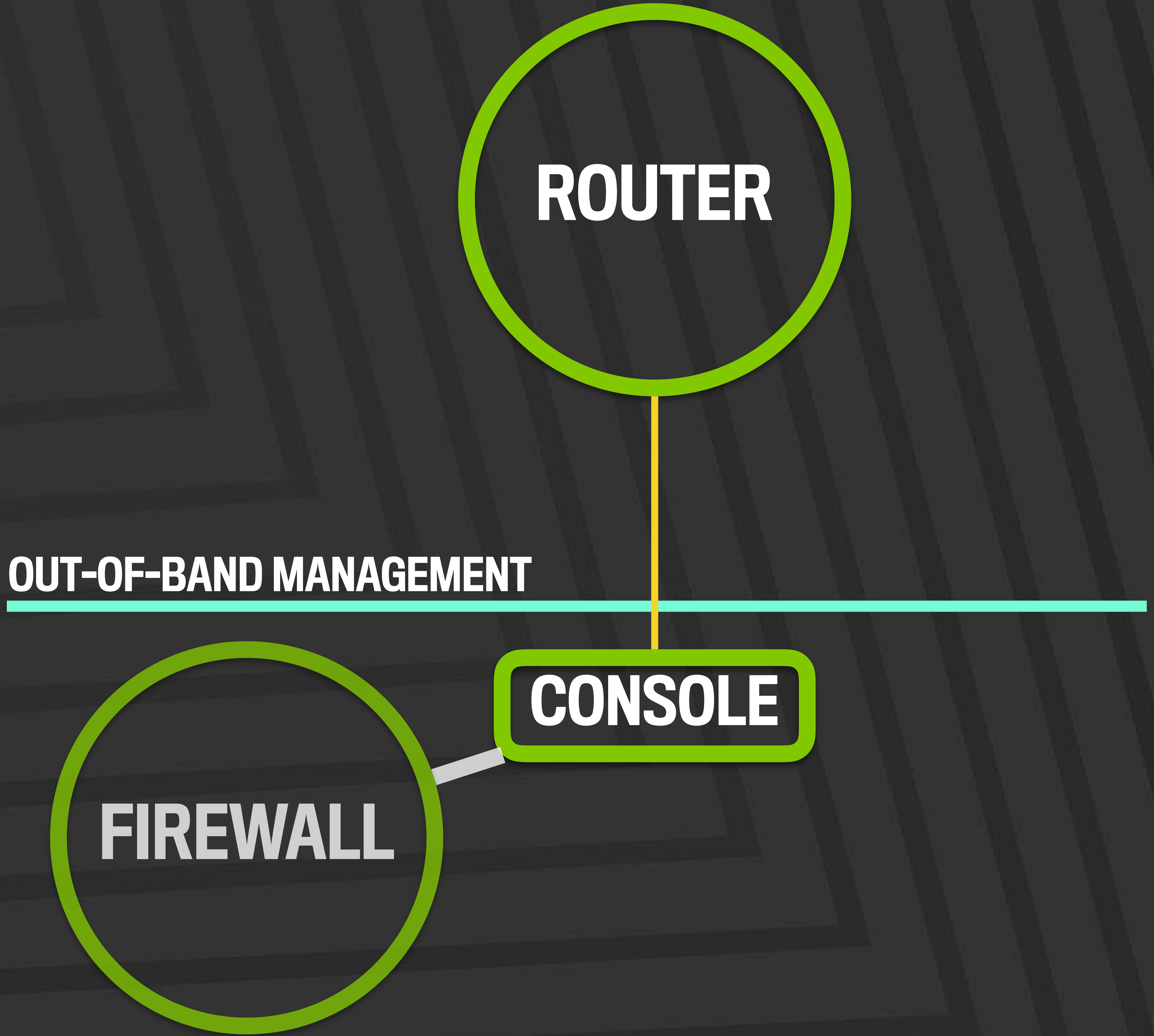
- apply configuration to a blank device
- upgrade software

Software for:

- notifying people
- checking hardware
- updating our asset management system
- changing BGP policy to enable traffic

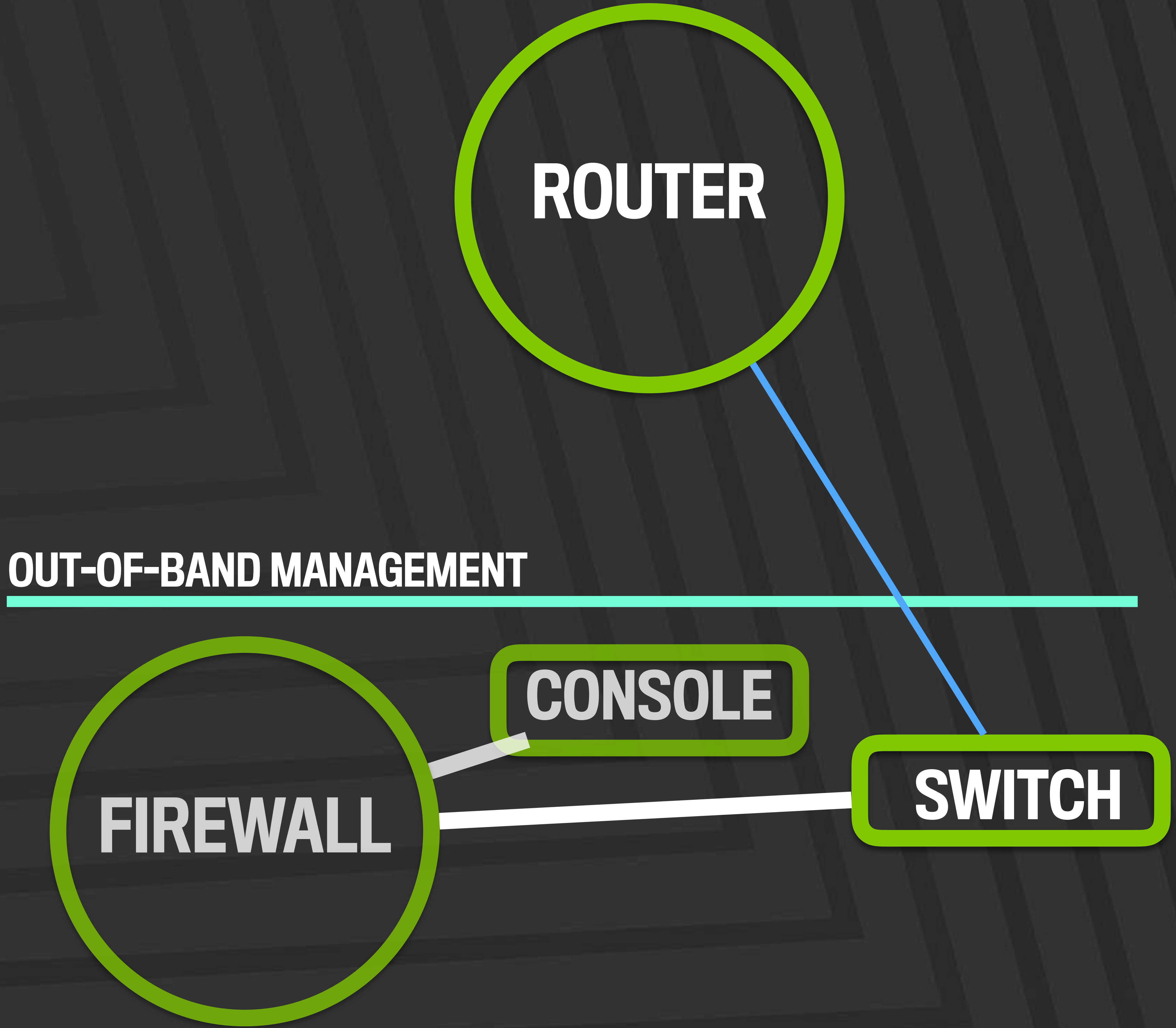
Empower and enable our engineers!

Options for loading configuration



```
CONSOLE >>  '\r'
CONSOLE <<  'login:'
CONSOLE >>  'root\r'
CONSOLE <<  'password:'
CONSOLE >>  '\r'
CONSOLE >>  'router>'
CONSOLE <<  'enable\r'
CONSOLE >>  'router#'
CONSOLE >>  'config t\r'
CONSOLE <<  'router(config)#'
...
```

Options for loading configuration



```
ETH0 >> DHCPDISCOVER
ETHO << DHCPOFFER
ETHO >> DHCPREQUEST
ETH0 << DHCPACK
ETH0 >> HTTP-REQUEST
ETH0 << HTTP-RESPONSE
ETH0 >> HTTP-REQUEST
ETH0 << HTTP-RESPONSE
...
```

Replacing MOPs with Vending Machine

Automating the MOPs?

- We needed to write a LOT of code.
- We needed a workflow automation system
- We needed to replace the MOPs

How? Divide and conquer!

- The system was built for the network engineer
- We removed the barriers
- We empowered our peer network engineering teams

Building for the network engineer

- Small, independent pieces of code written in any programming language
- Steps should do only one thing
- Knowledge of "the system" should not be required

How? Isolate "the system" from the workflow

- Units of work are called Steps
- A Step is a compiled piece of code that is executed as a binary
- Testing and development reduced to only your step

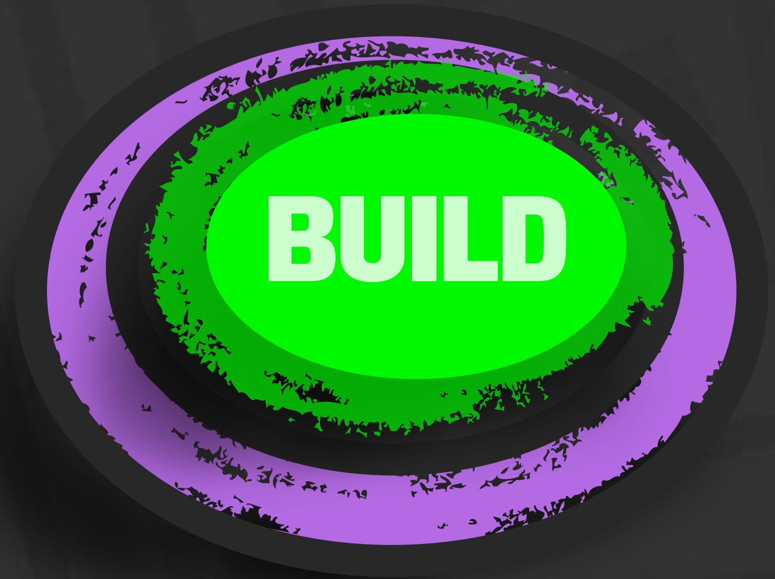
Giving the system a name

- We named it Vending Machine!
- Vending Machine is a purpose-built workflow automation system created around Zero Touch Provisioning
- Stability in step-level isolation

Provisioning redefined

- ~~Letting People Know~~
- Rack and Stack
- Cabling
- ~~Management IP assignment~~
- ~~Config Generation~~
- ~~Software Upgrades~~
- ~~Loading Config~~
- ~~Validating Config~~
- ~~Validating Hardware (Fans, Power Supplies, Linecards)~~
- ~~Validating Physical Connectivity (LLDP and Light Levels)~~
- ~~Validating Logical Connectivity (Protocols)~~
- ~~Updating External Systems (Location Data, Status)~~
- ~~Undraining Traffic~~

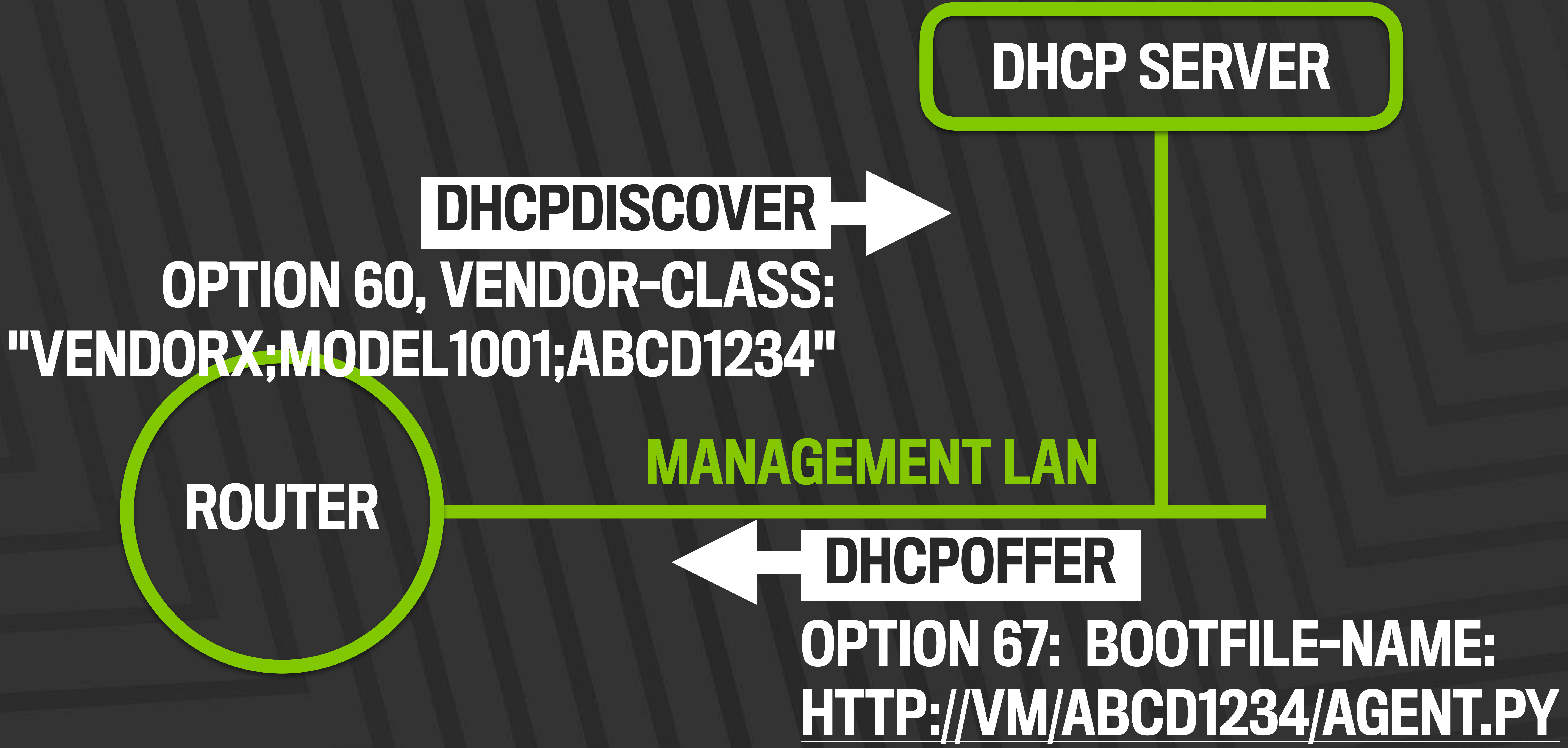
~~MOPs~~



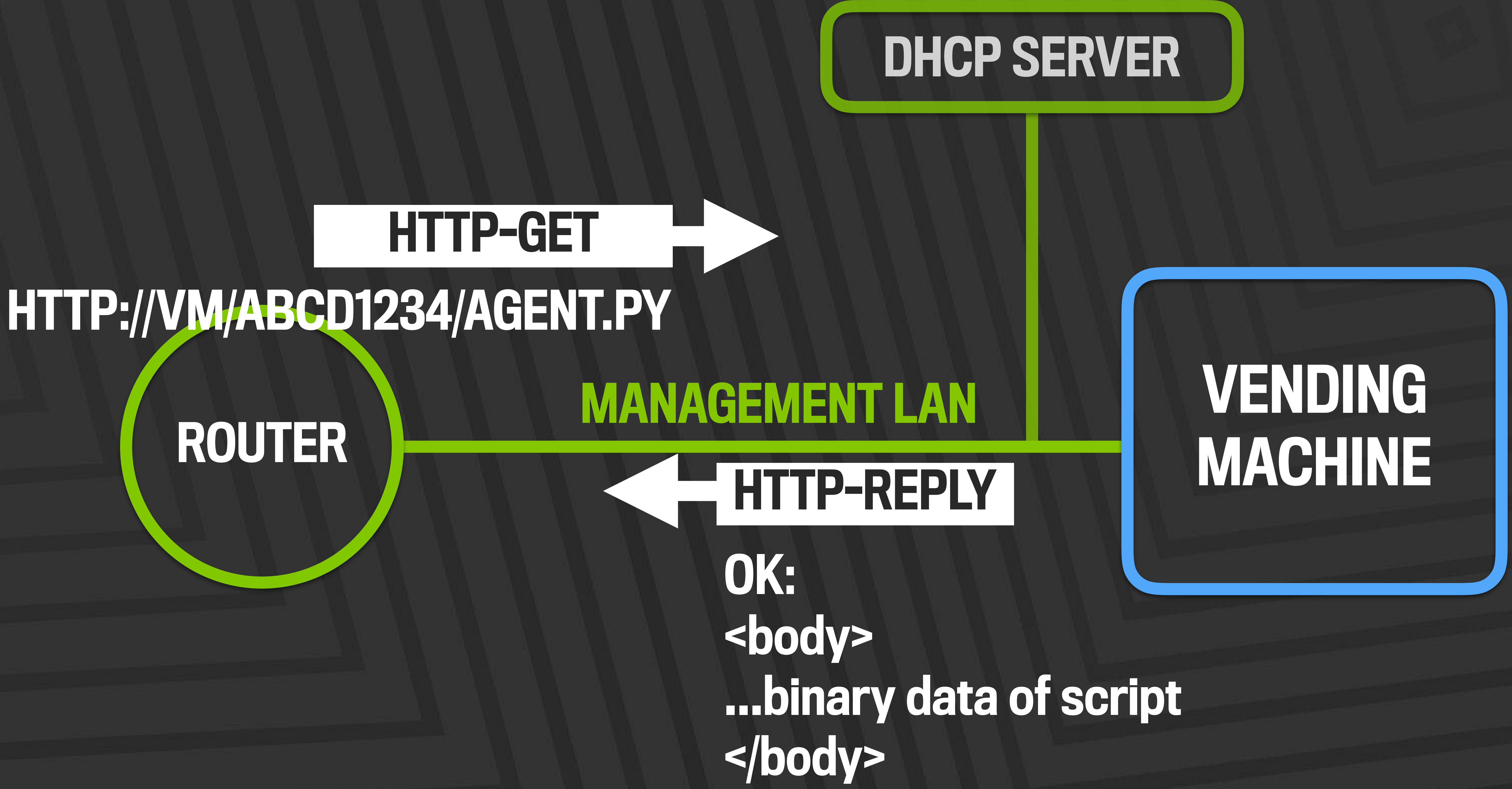
```
vm configure <name>
```

Zero Touch Provisioning

Requesting a ZTP agent over DHCP

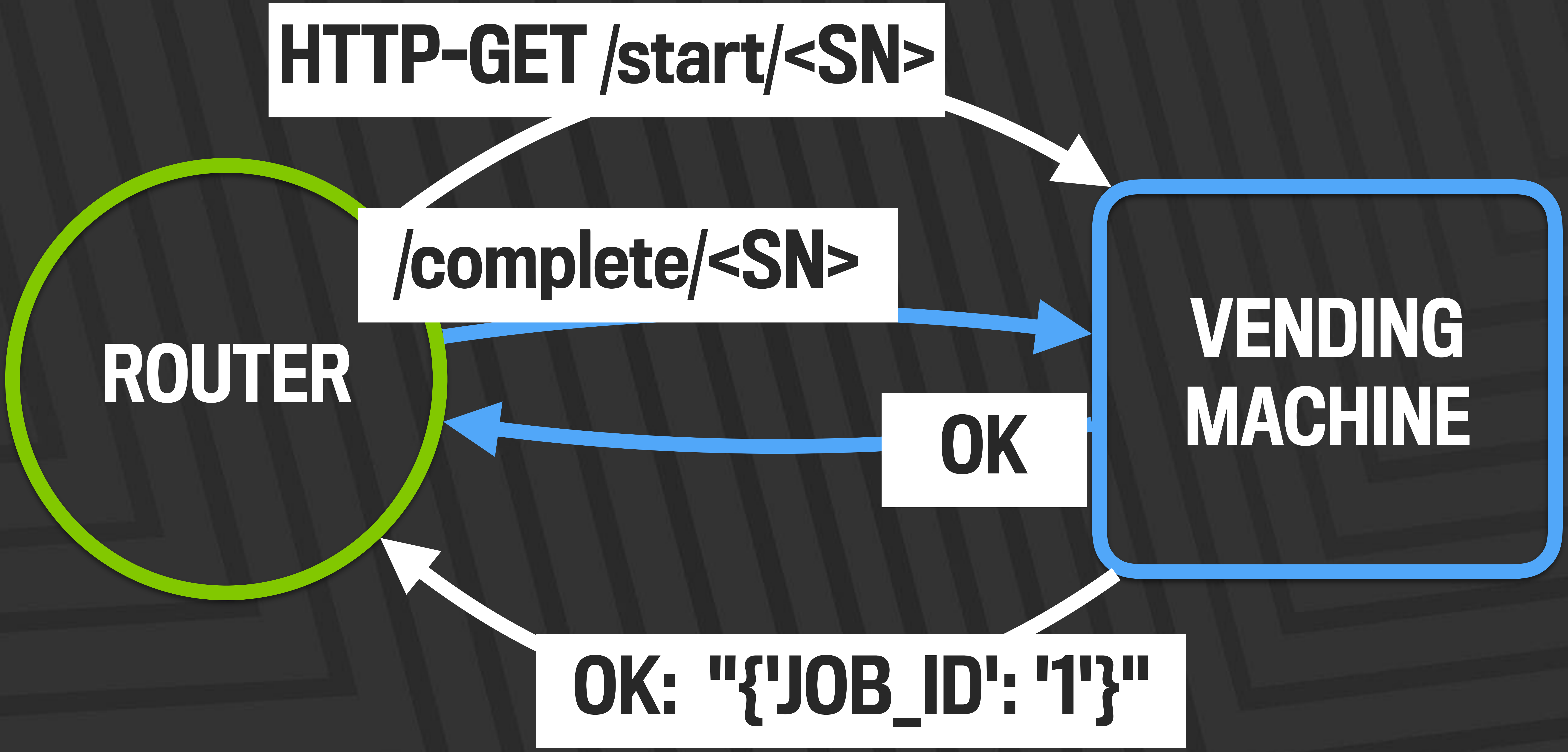

DHCP SERVER
DHCPDISCOVER
OPTION 60, VENDOR-CLASS:
"VENDORX;MODEL1001;ABCD1234"
ROUTER
MANAGEMENT LAN
DHCPOFFER
OPTION 67: BOOTFILE-NAME:
HTTP://VM/ABCD1234/AGENT.PY

Requesting a ZTP agent over DHCP

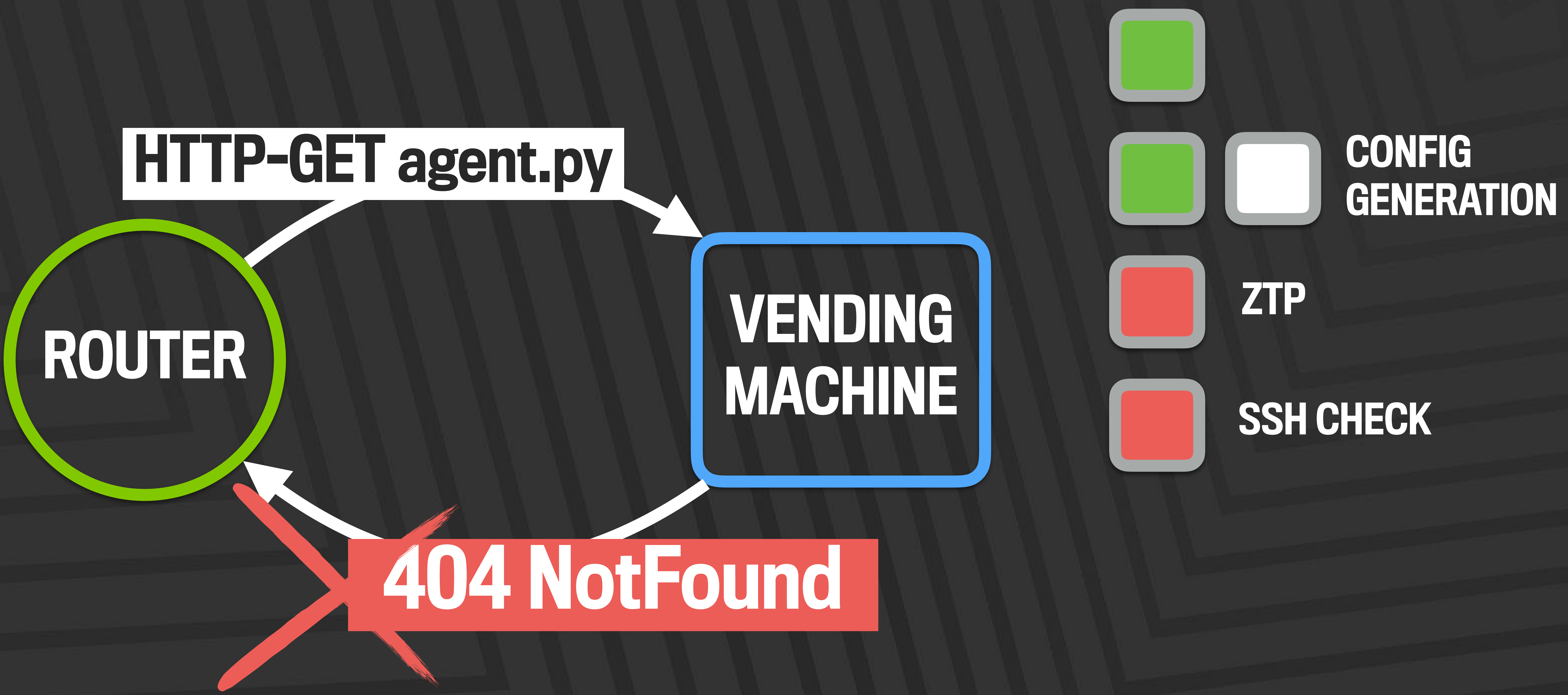


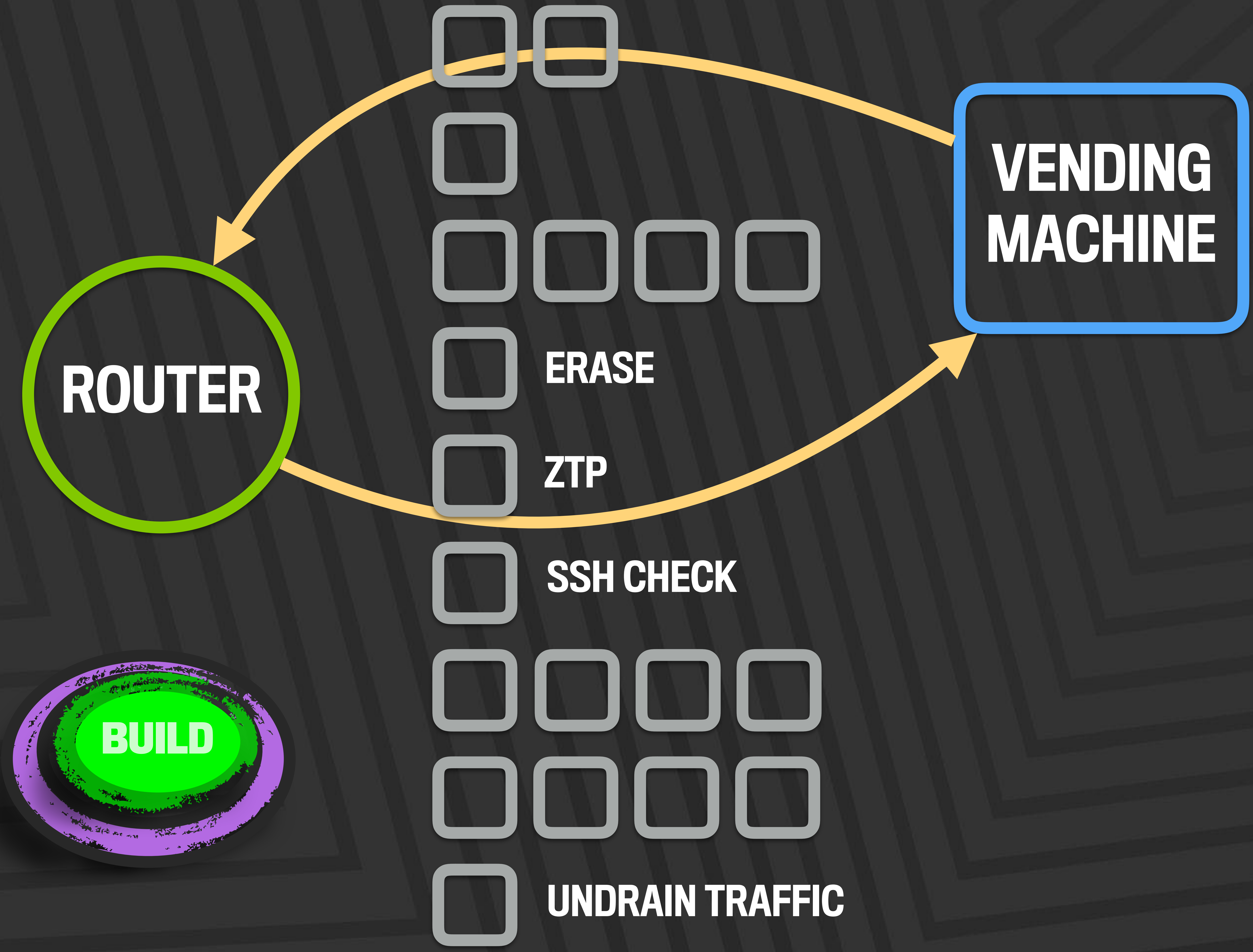
```
OK:
<body>
...binary data of script
</body>
```

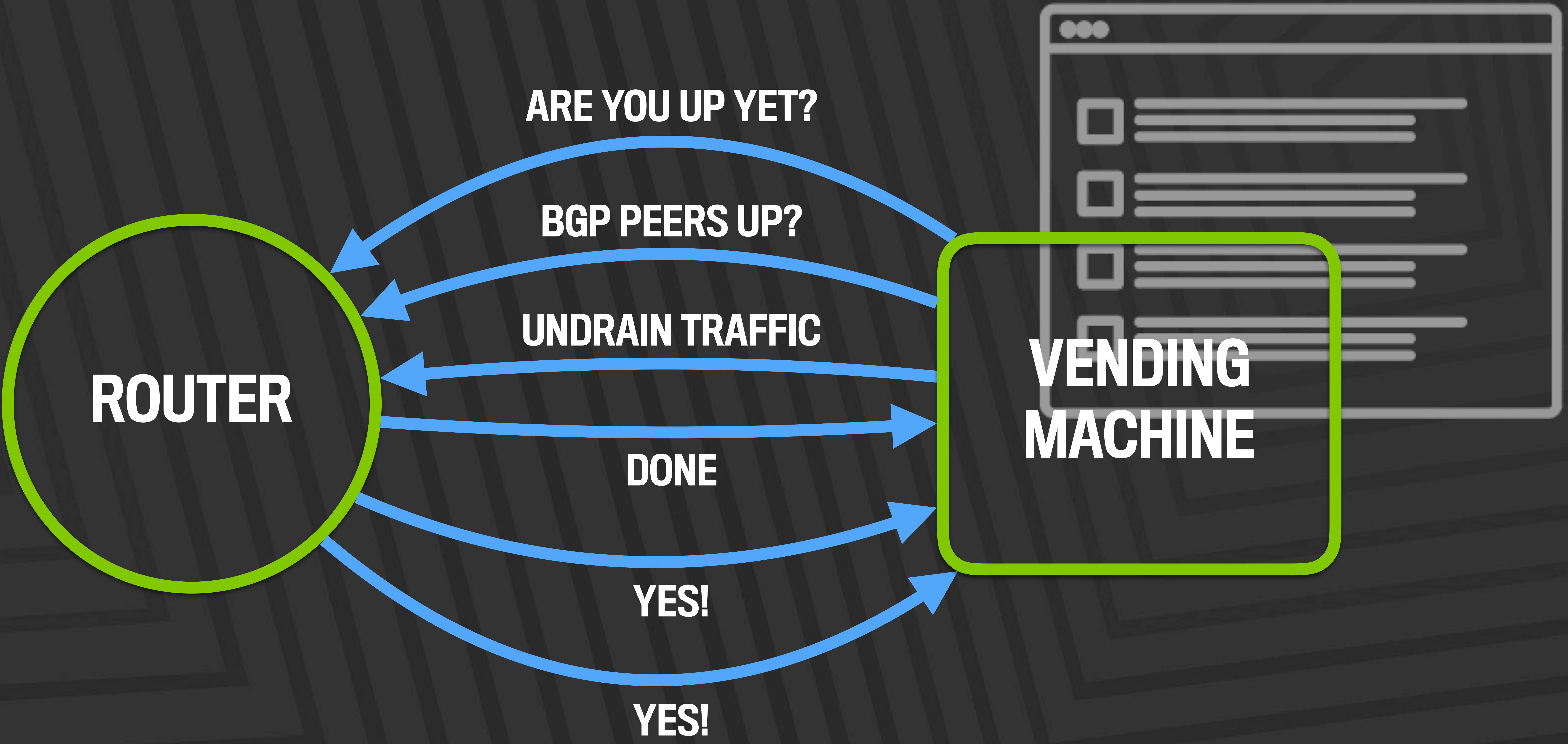
Building a feedback loop



Delaying ZTP while running other Steps




ROUTER
VENDING MACHINE
ERASE
ZTP
SSH CHECK
UNDRAIN TRAFFIC
BUILD


ARE YOU UP YET?
BGP PEERS UP?
UNDRAIN TRAFFIC
ROUTER
VENDING MACHINE
DONE
YES!
YES!

Writing a Vending Machine Step

Config Generation

```
#!/usr/bin/python3

import json
import logging
import sys


def main():

    stdin = sys.stdin.read().strip()
    input = json.loads(stdin)
    hostname = input['hostname']

    logging.info(f'Generating configs for {hostname}')

    build_configs(hostname)

    verify_configs(hostname)
```

Config Generation

```
#!/usr/bin/python3

from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from configservice import ConfigGenerationService
from configservice.ttypes import ConfigGenerationResult


def build_configs(self, hostname):
    transport = TSocket('localhost', 9090)
    transport = TTransport.TBufferedTransport(transport)
    protocol = TBinaryProtocol.TBinaryProtocol(transport)

    with ConfigGenerationService.Client(protocol) as client:
        result = client.generate_configs(hostname)
        if result.status == ConfigGenerationResult.SUCCESS:
            logging.info('Generated new configs!')
        else:
            logging.info('Configs are already up-to-date.')
```

Apache Thrift's client example:
http://thrift-tutorial.readthedocs.io/en/latest/usage-example.html

Config Generation

```
#!/usr/bin/python3

import urllib3

VM_VIP = '2a03:2880:f101:83:face:b00c:0:25de'


def verify_configs(self, hostname):

        with urllib3.PoolManager() as http:
            url = f'http://{VM_VIP}/{hostname}/config.conf')

            response = http.request('GET', url)

            if response.status == 200:
                logging.info(
                    f'Successfully fetched config from {url}')
                sys.exit(0)

        logging.error(
            f'Failed to fetch config from {url}')
        sys.exit(1)
```

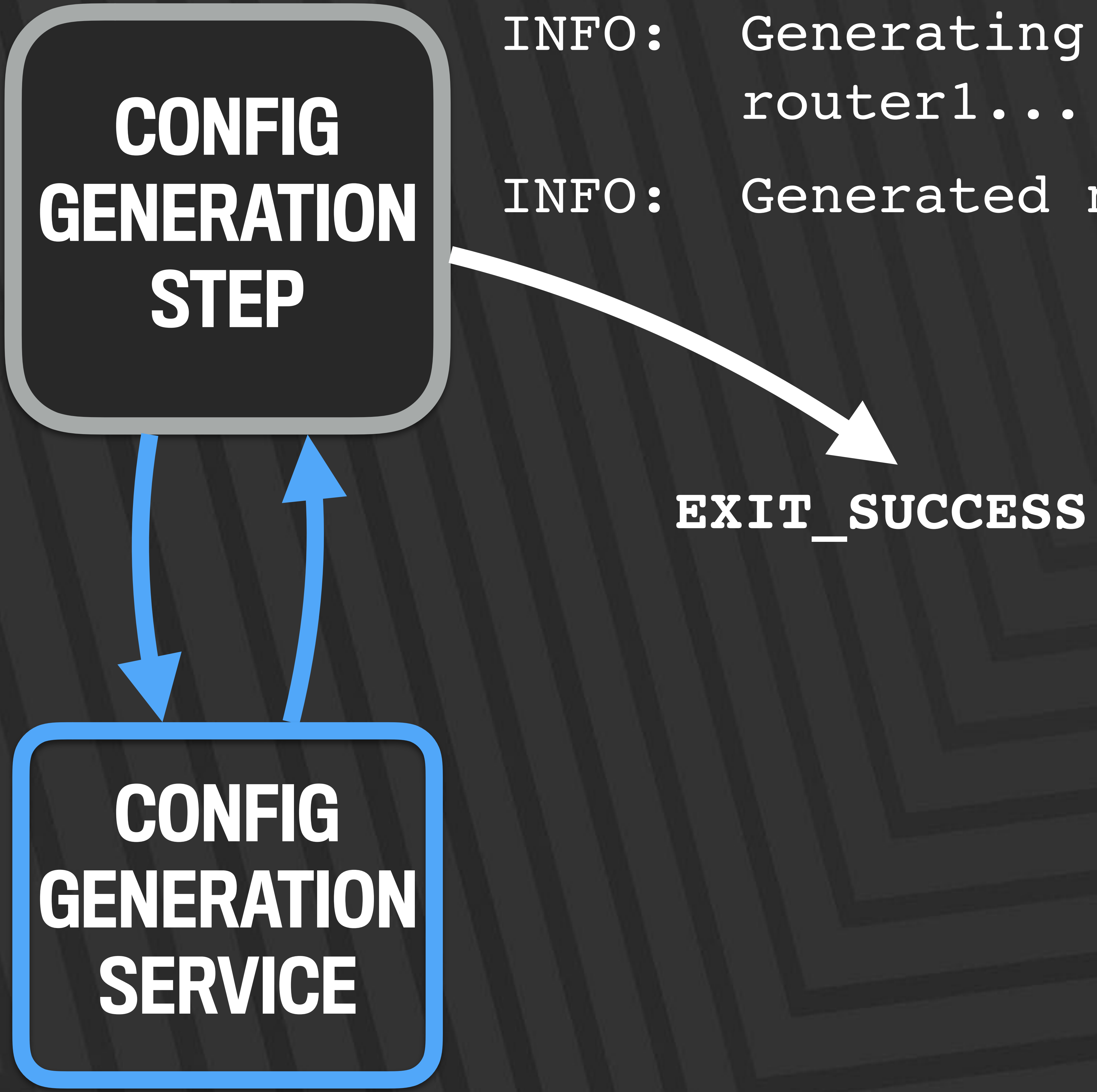
Config Generation

STDIN:

```
'{"asset_id": "10001",
  "hostname": "router1",
  "serial": "AAEF0016",
  "job_id": "1",
  "attempt_id": "1"}'
```

STDERR:

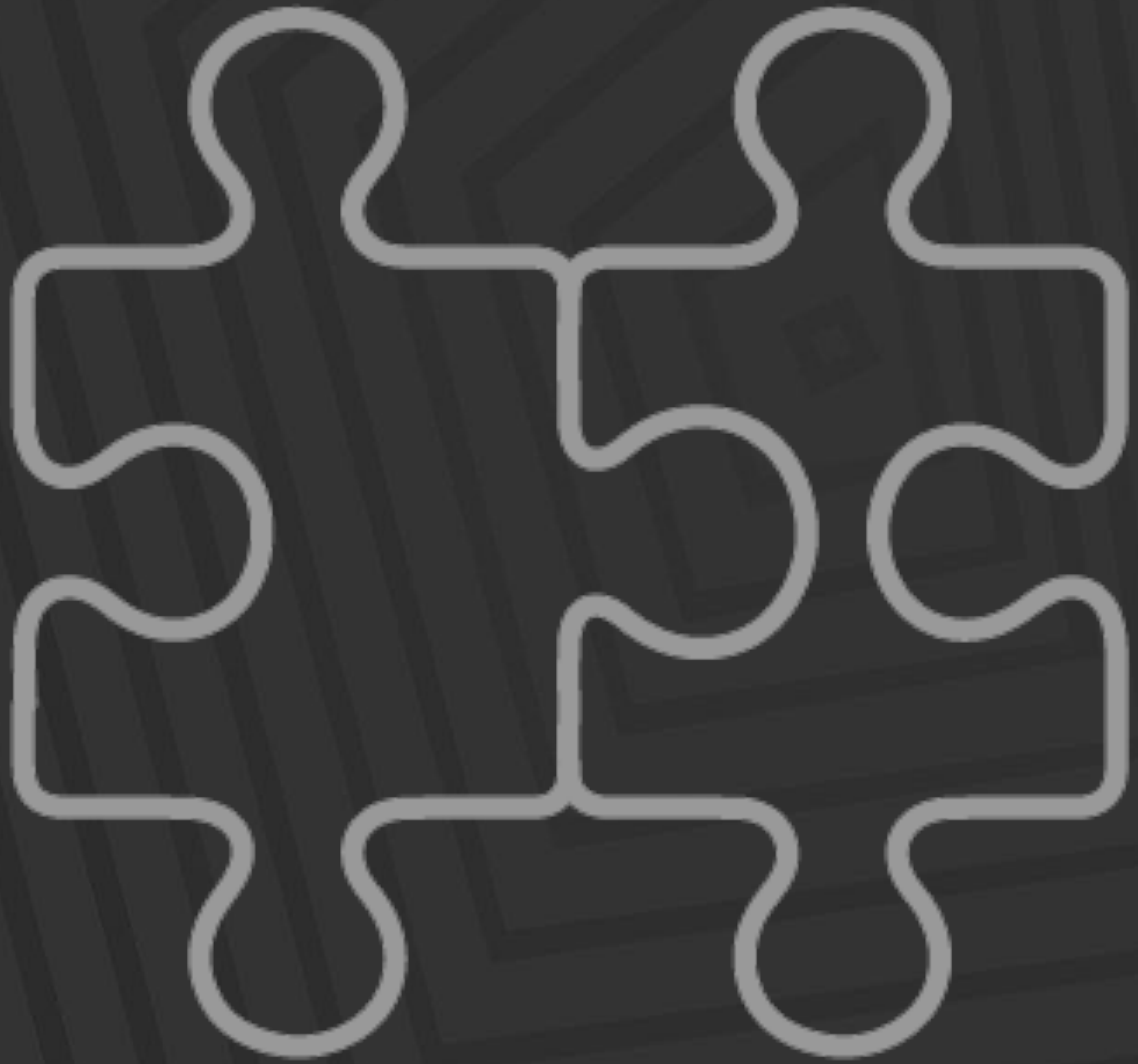
```
INFO:  Generating configs for
       router1...

INFO:  Generated new configs!
```



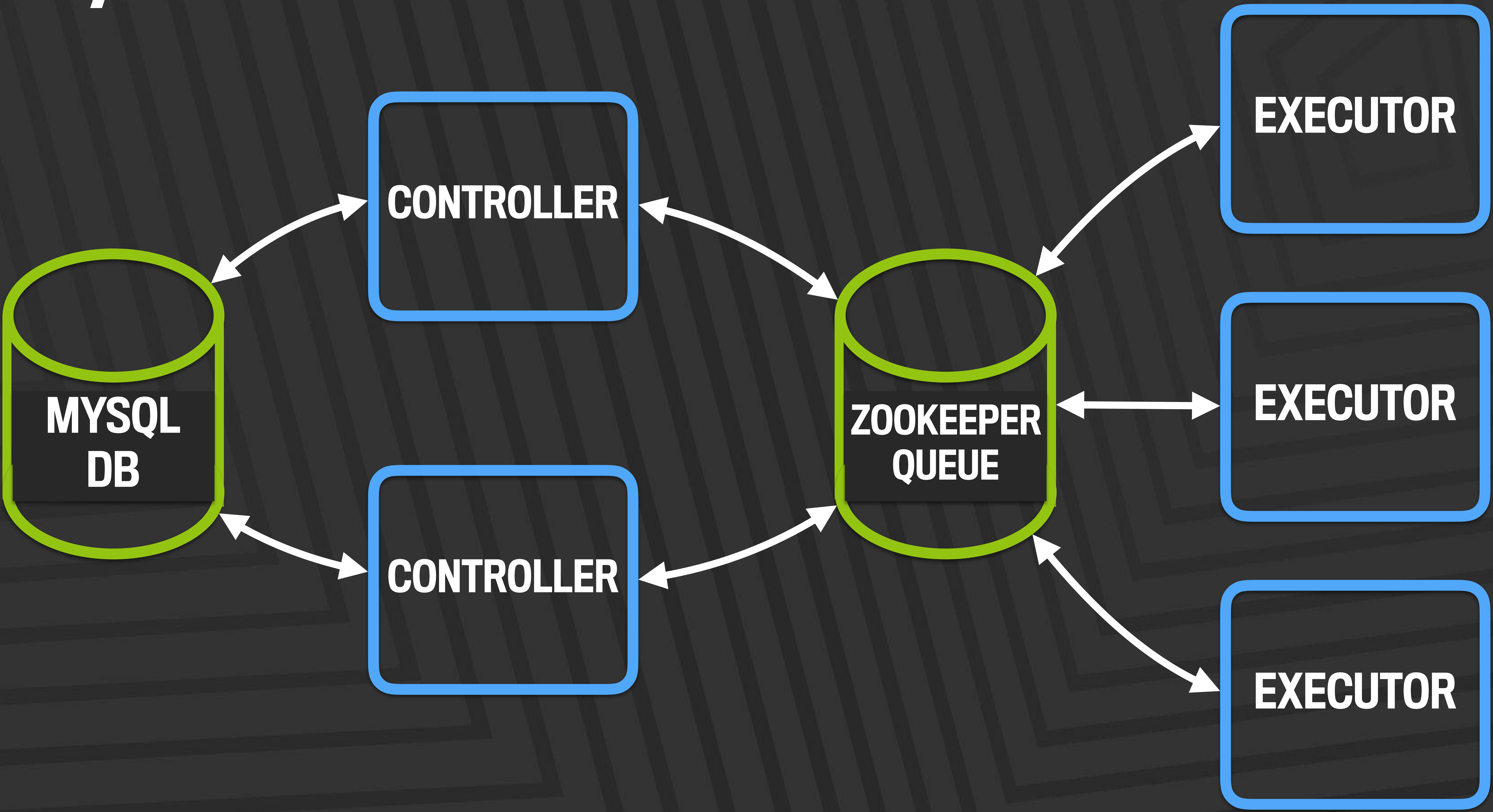
Vending Machine Internals

Design Goals

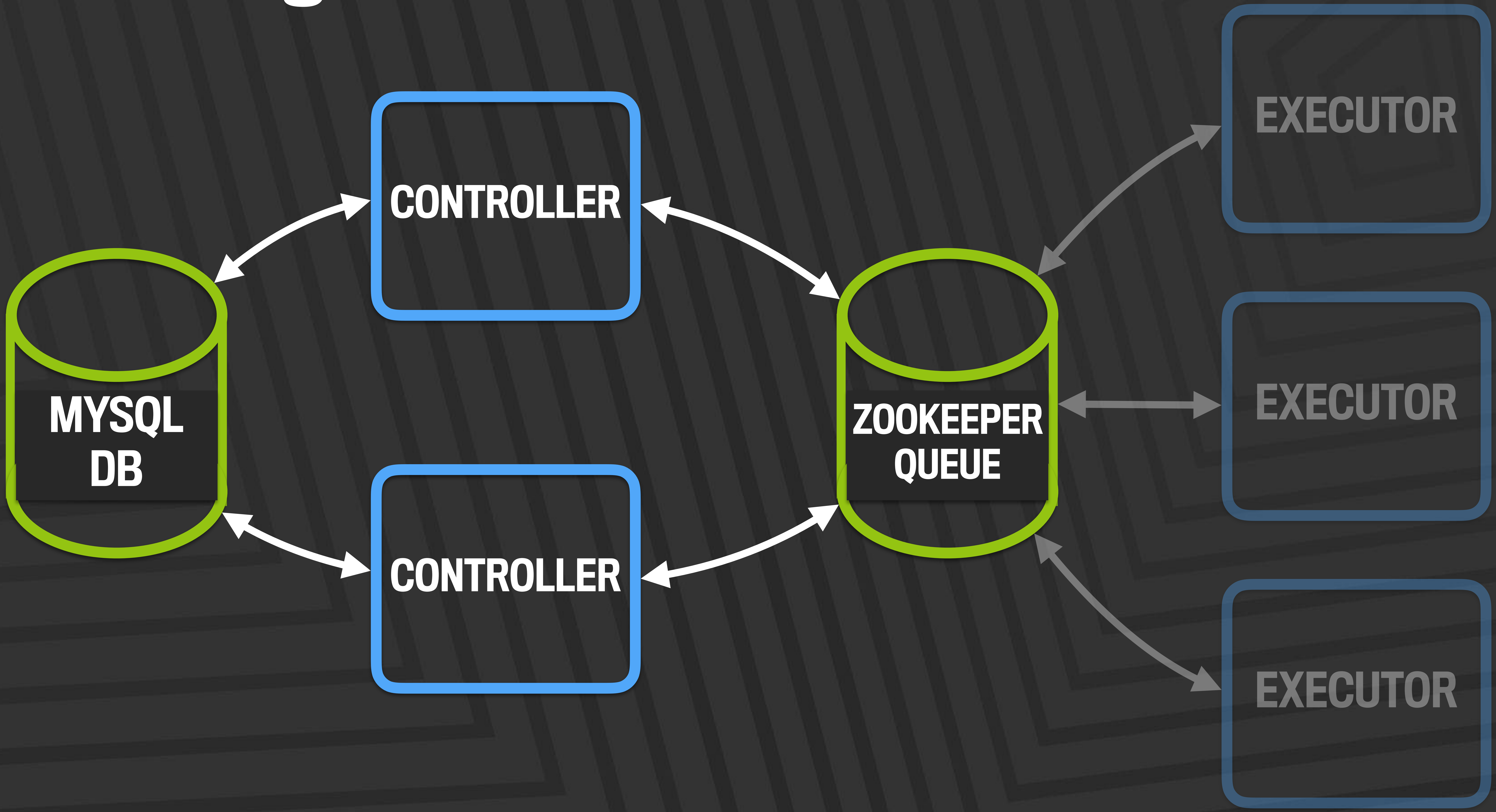
- Flexibility and Rapid Development
- Scalable
- Fast
- Resilient
- Predictable



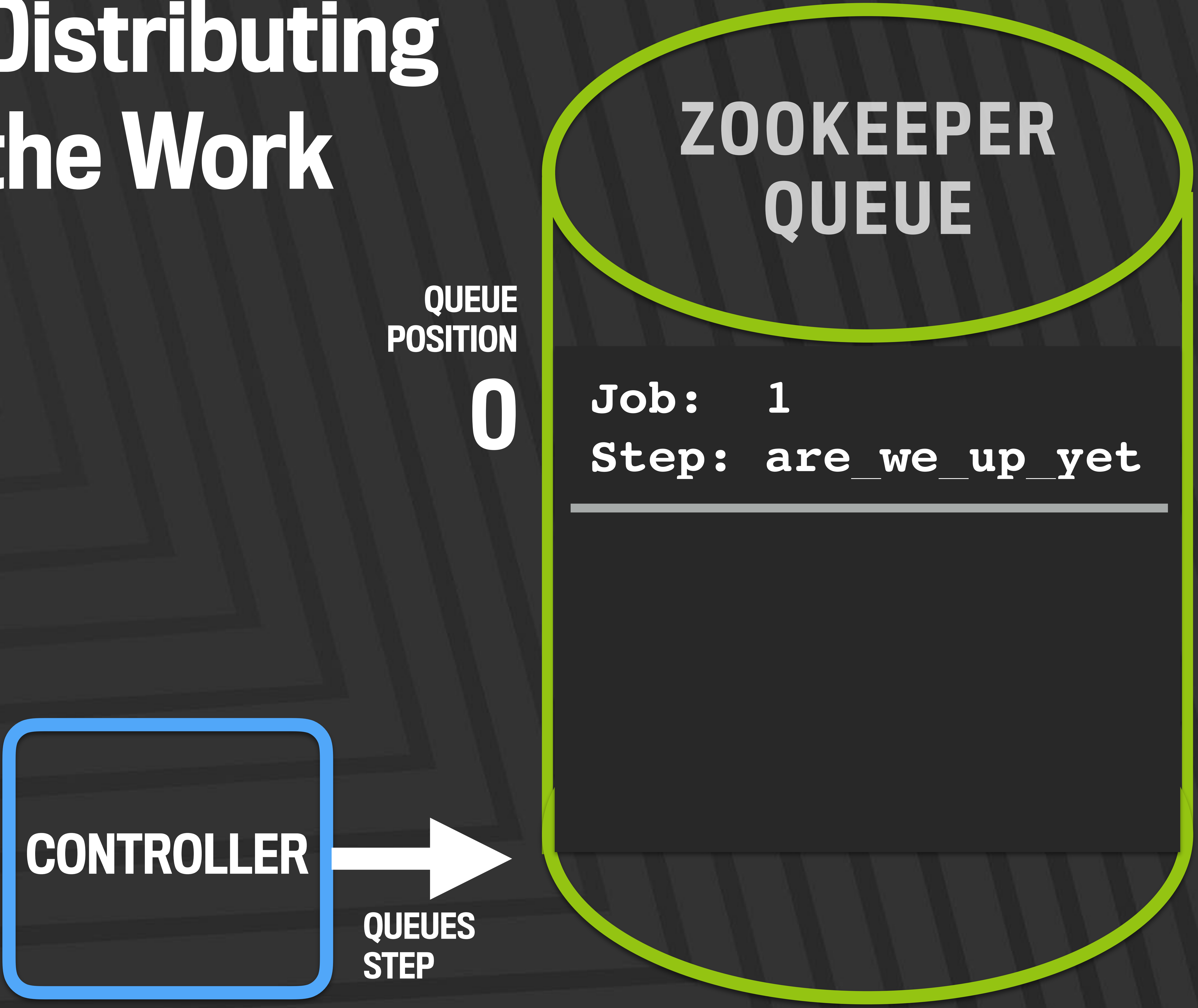
The System


CONTROLLER
MYSQL DB
CONTROLLER
ZOOKEEPER QUEUE
EXECUTOR
EXECUTOR
EXECUTOR

Coordinating Jobs

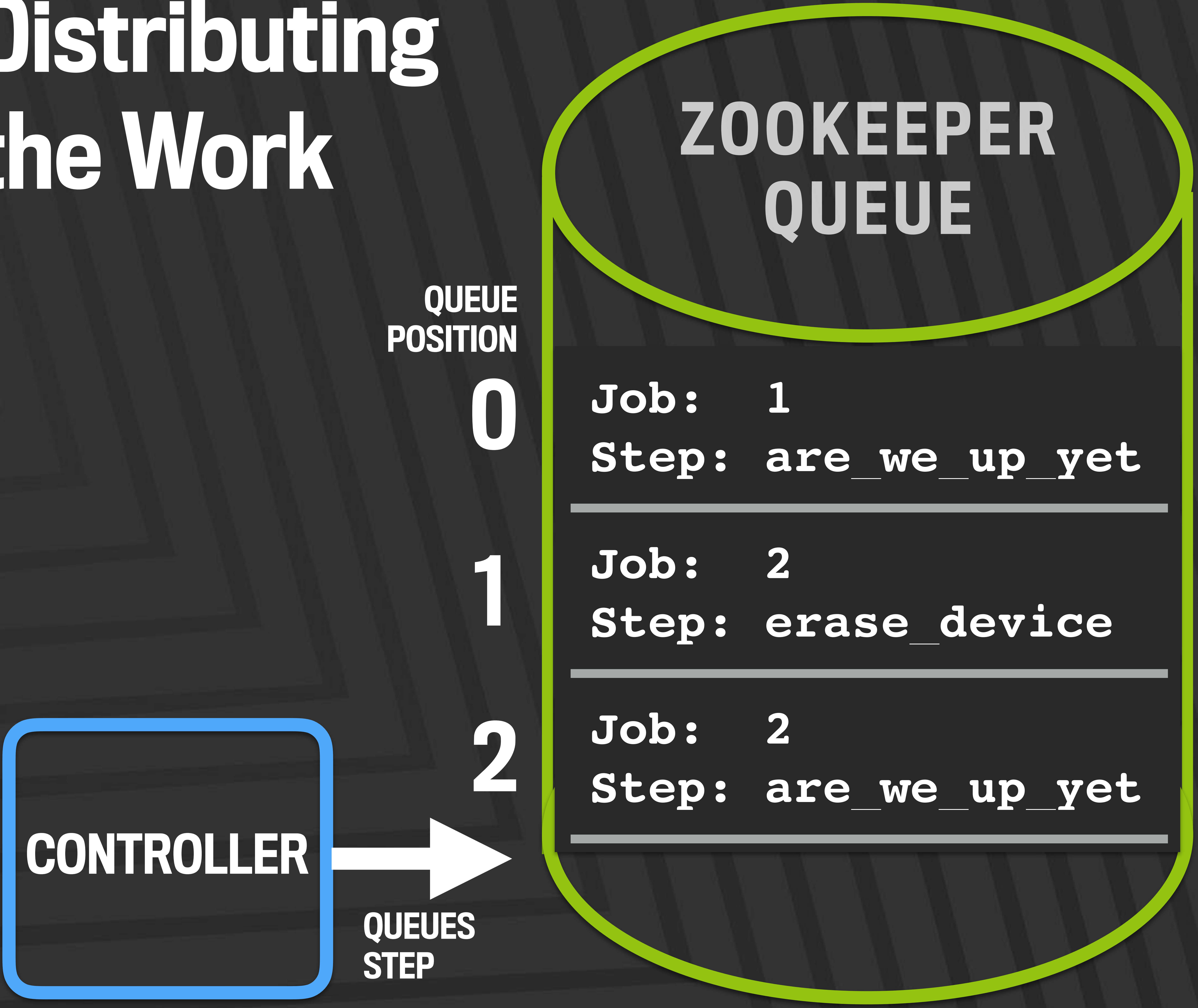


Distributing the Work

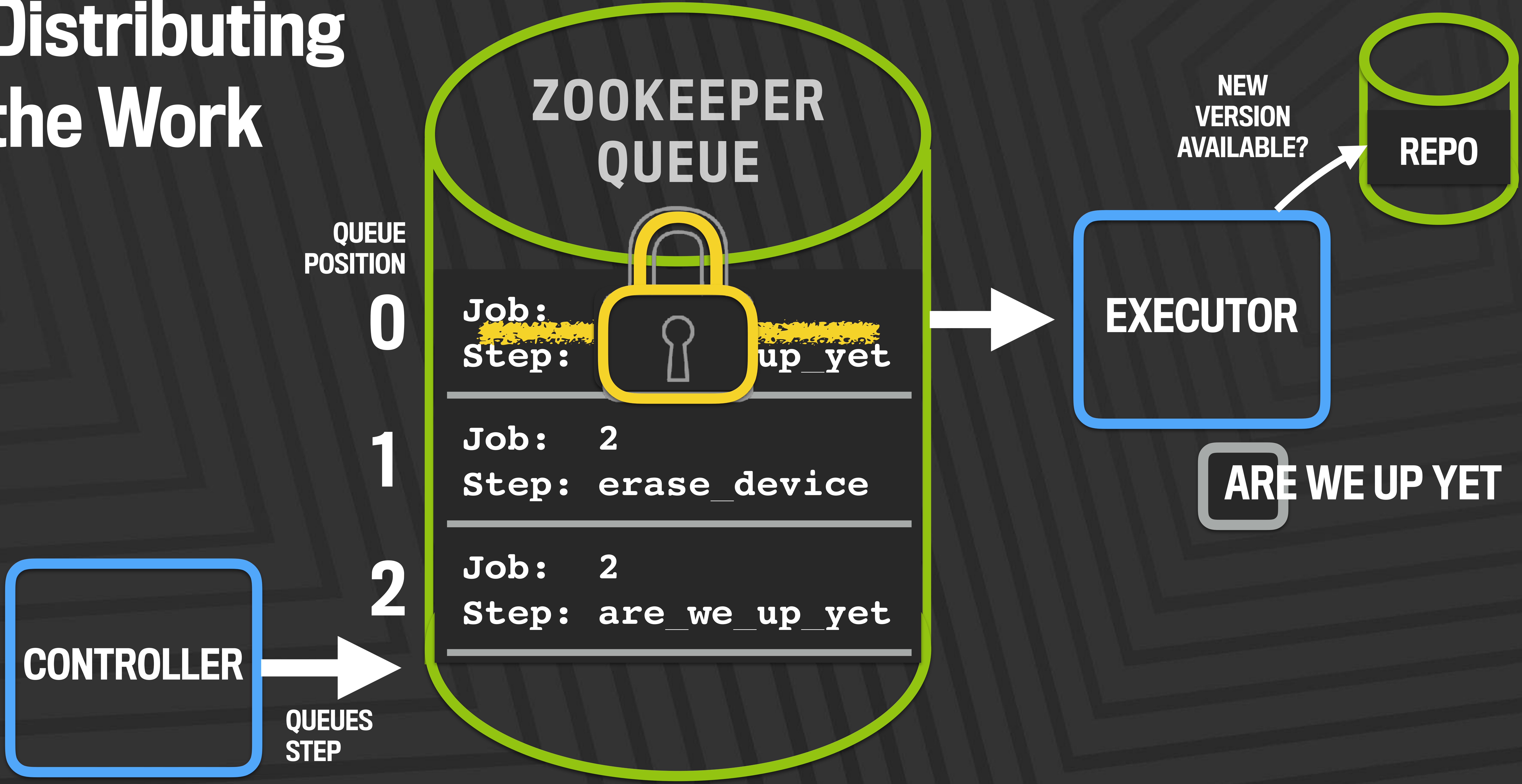


```
Job:   1
Step: are_we_up_yet
```

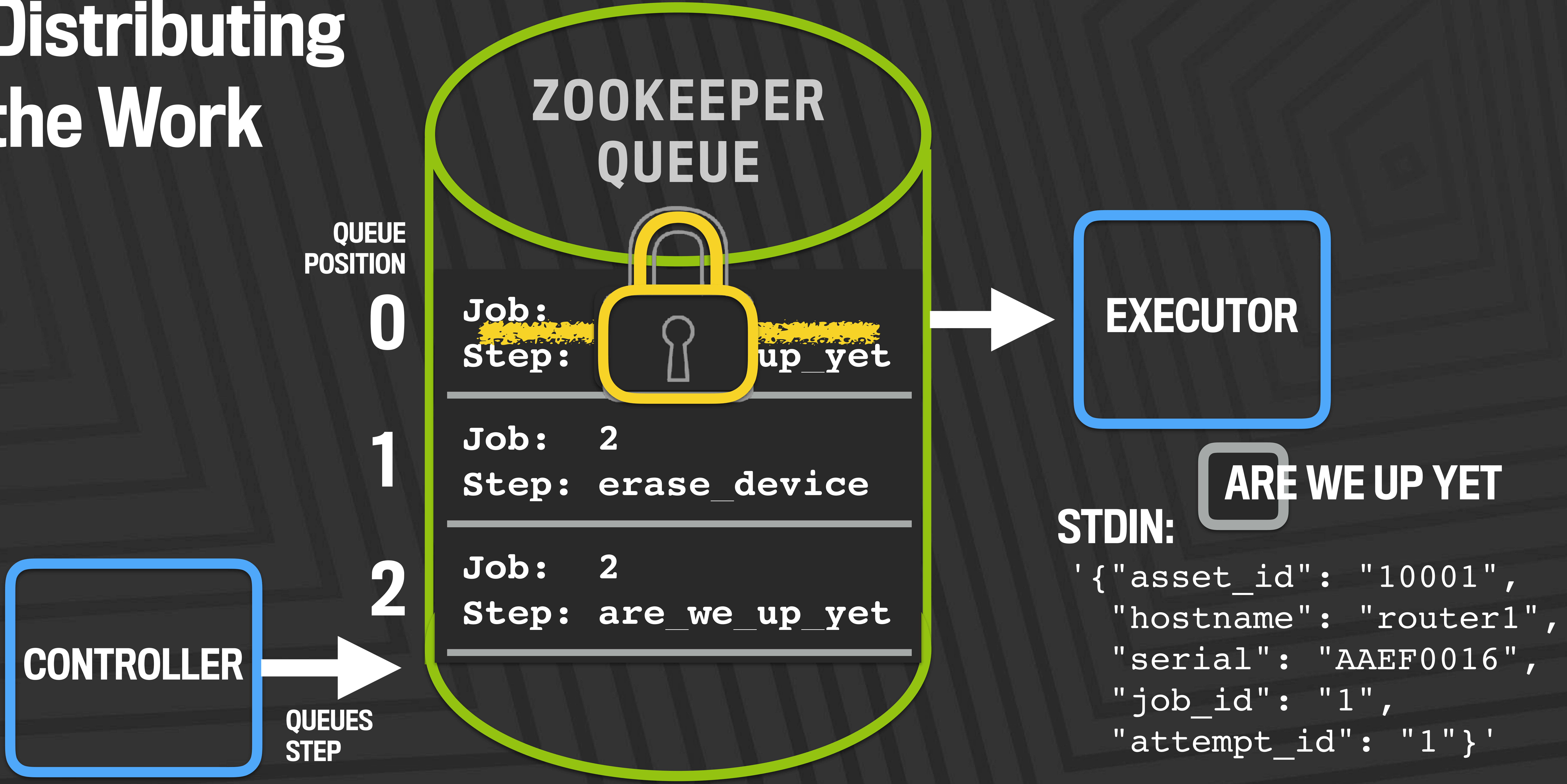
Distributing the Work



Distributing the Work

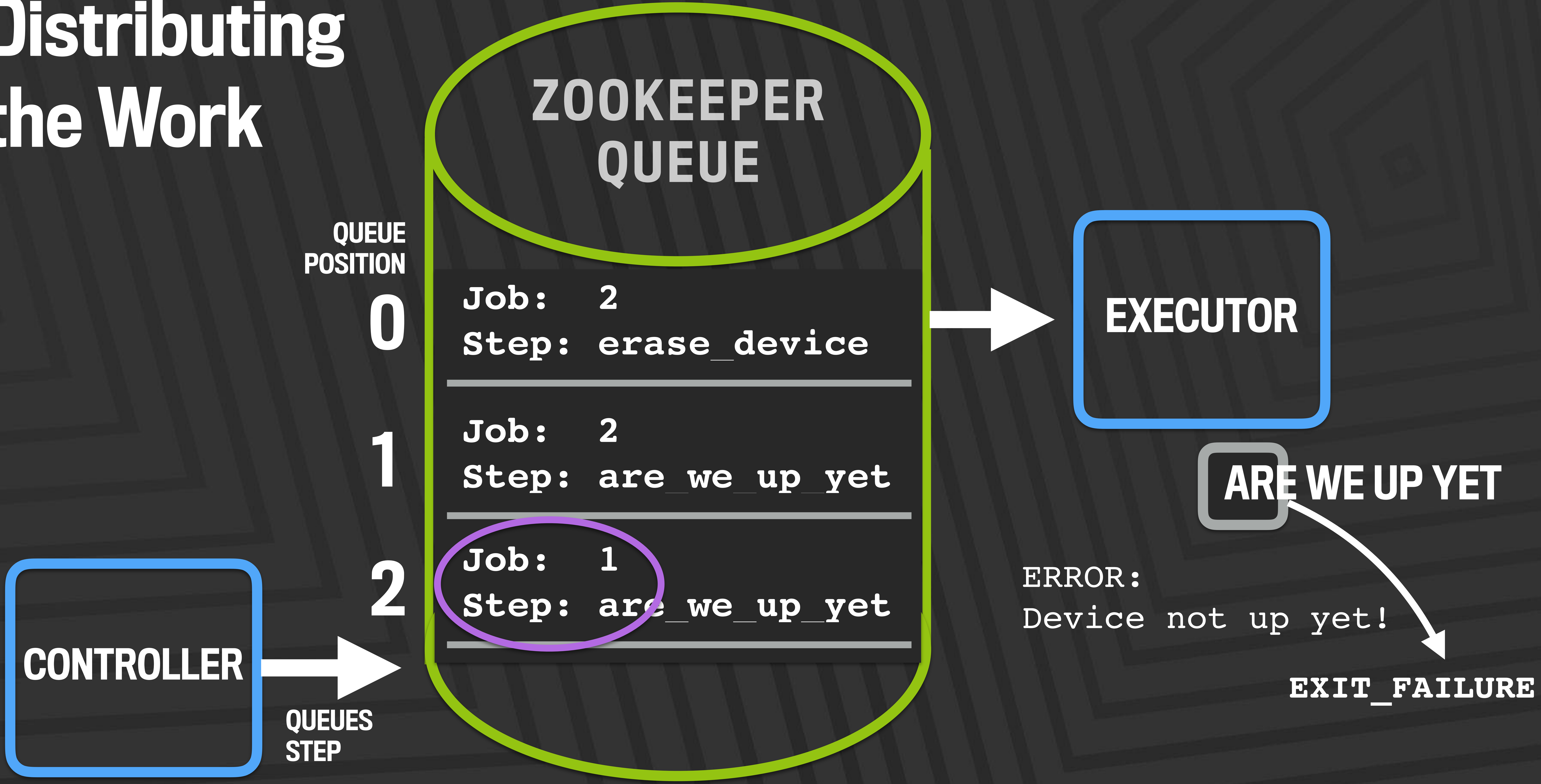


Distributing the Work



```
'{"asset_id": "10001",
  "hostname": "router1",
  "serial": "AAEF0016",
  "job_id": "1",
  "attempt_id": "1"}'
```

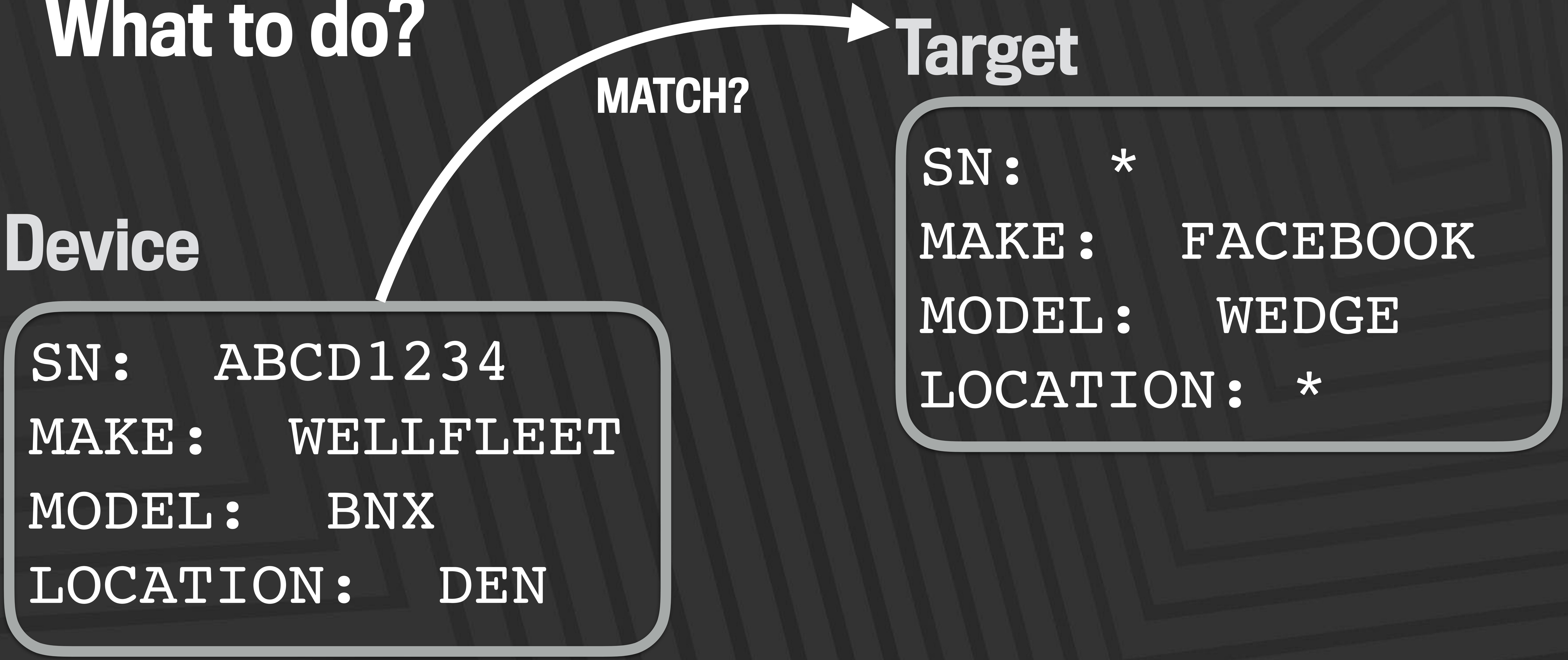
Distributing the Work



Transient Failures

```
Traceback (most recent call last): File
"<stdin>", line 1, in <module> File "/
usr/lib64/python2.7/socket.py", line
224, in meth return
getattr(self._sock,name)(*args)
socket.error: [Errno 111] Connection
refused
```

What to do?



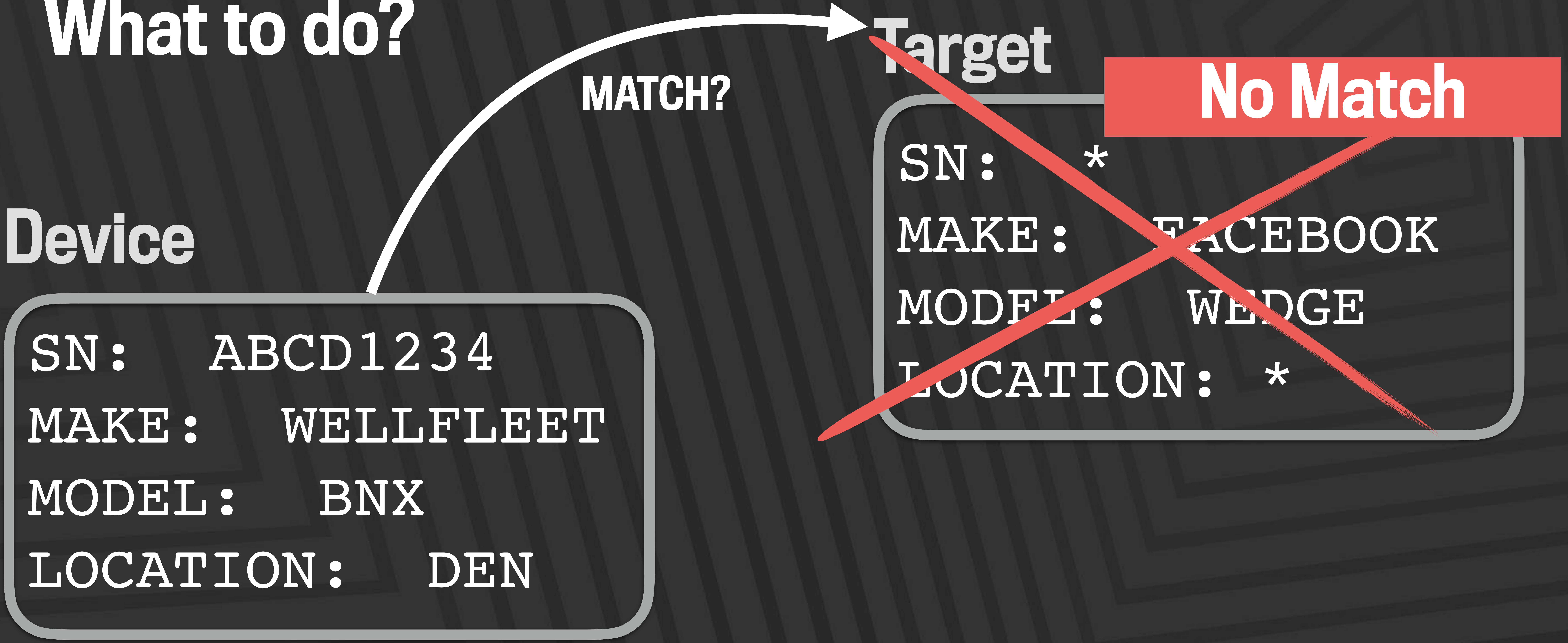
Target

```
SN:   *
MAKE:   FACEBOOK
MODEL:   WEDGE
LOCATION:  *
```

Device

```
SN:   ABCD1234
MAKE:   WELLFLEET
MODEL:   BNX
LOCATION:   DEN
```

What to do?



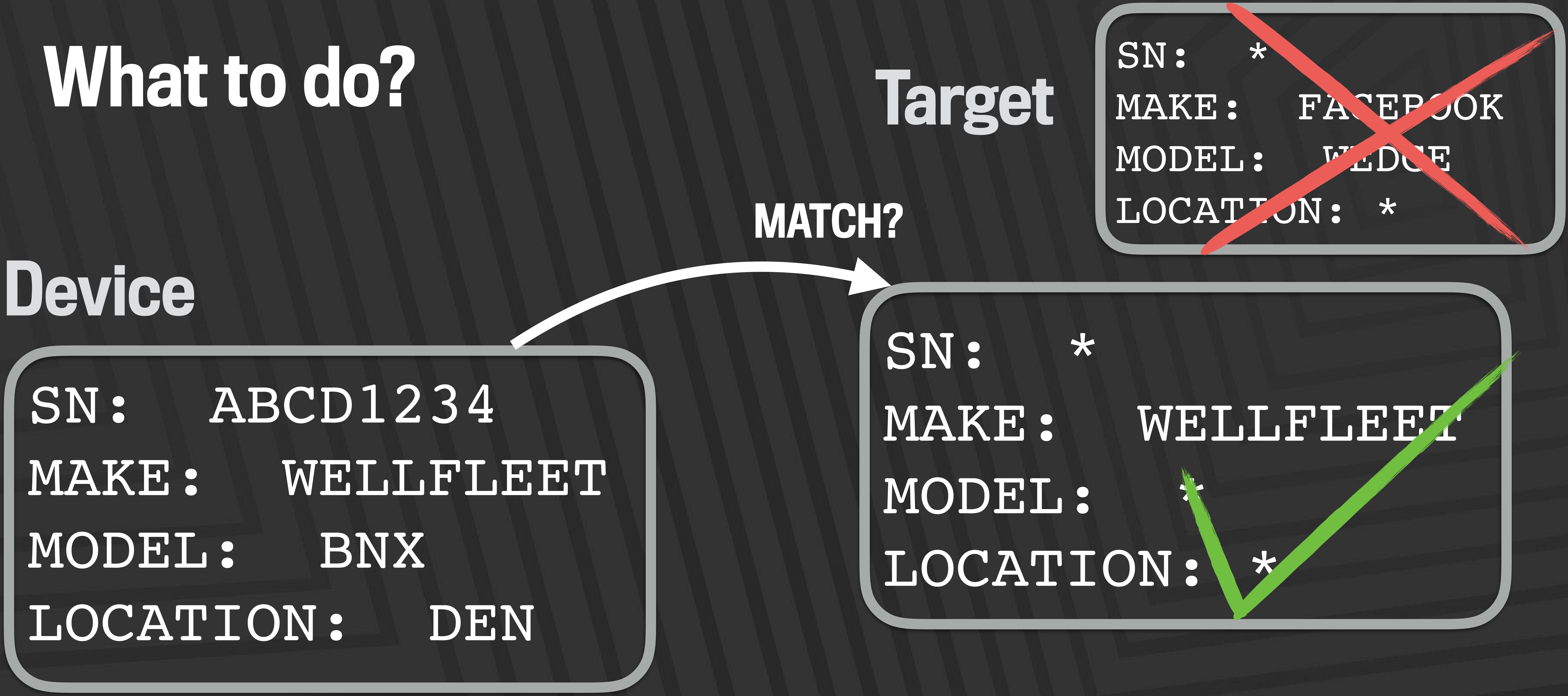
Device

SN: ABCD1234
MAKE: WELLFLEET
MODEL: BNX
LOCATION: DEN

Target

~~SN: *~~
~~MAKE: FACEBOOK~~
~~MODEL: WEDGE~~
~~LOCATION: *~~

What to do?



Device

```
SN:    ABCD1234
MAKE:    WELLFLEET
MODEL:    BNX
LOCATION:    DEN
```

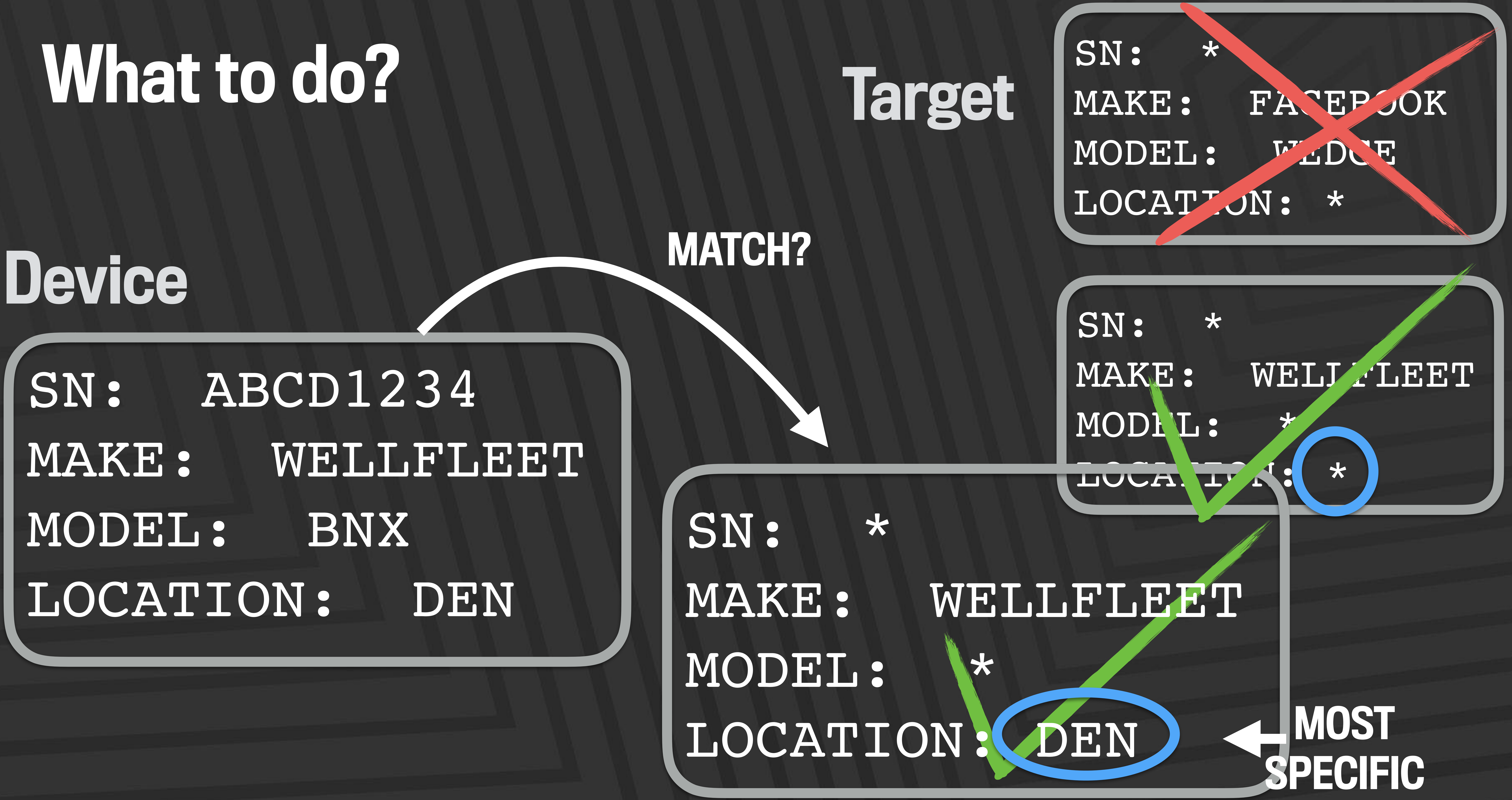
Target

~~SN: *~~
~~MAKE: FACEBOOK~~
~~MODEL: WEDGE~~
~~LOCATION: *~~

```
SN:    *
MAKE:    WELLFLEET
MODEL:    *
LOCATION:  *
```

✓

What to do?



Device

```
SN:    ABCD1234
MAKE:    WELLFLEET
MODEL:    BNX
LOCATION:    DEN
```

Target

```
SN:    *
MAKE:    FACEBOOK
MODEL:    WEDGE
LOCATION:  *
```

```
SN:    *
MAKE:    WELLFLEET
MODEL:    *
LOCATION:  *
```

```
SN:    *
MAKE:    WELLFLEET
MODEL:    *
LOCATION:  DEN
```

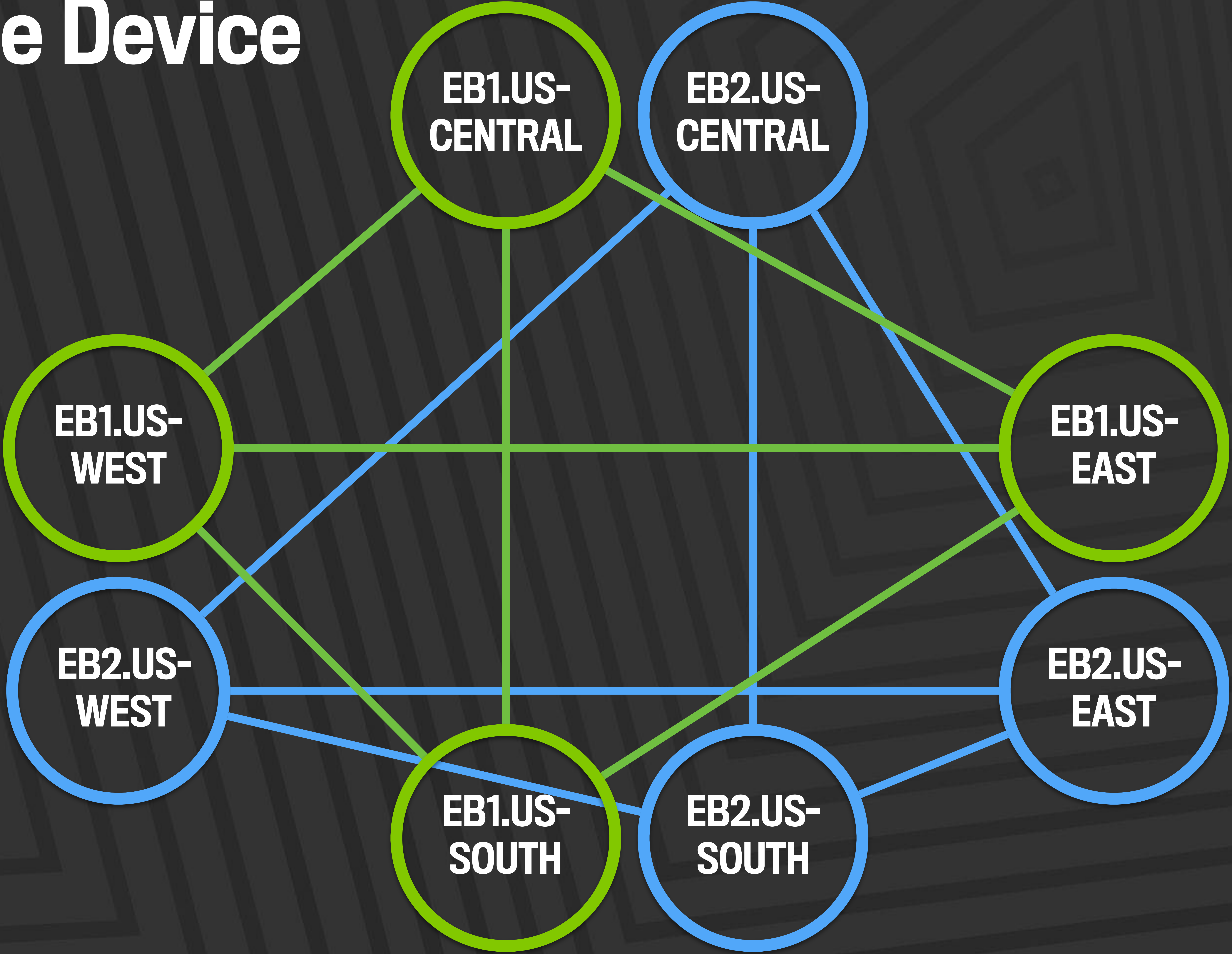
Going Beyond the Device

☐ DRAIN PLANE 2

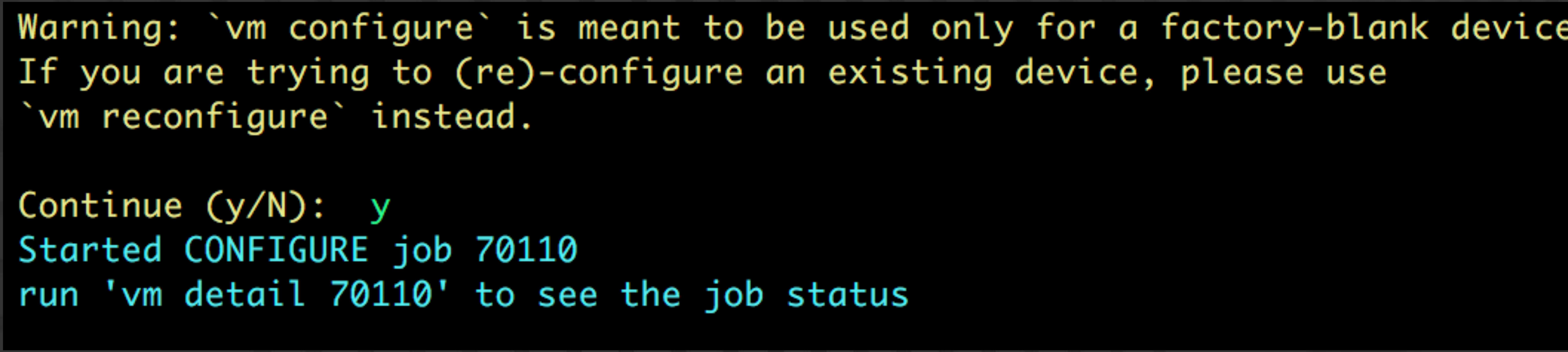
☐ ☐ ☐ ☐

REBUILD:
EB2.US-WEST
EB2.US-CENTRAL
EB2.US-SOUTH
EB2.US-EAST

☐ UNDRAIN PLANE 2

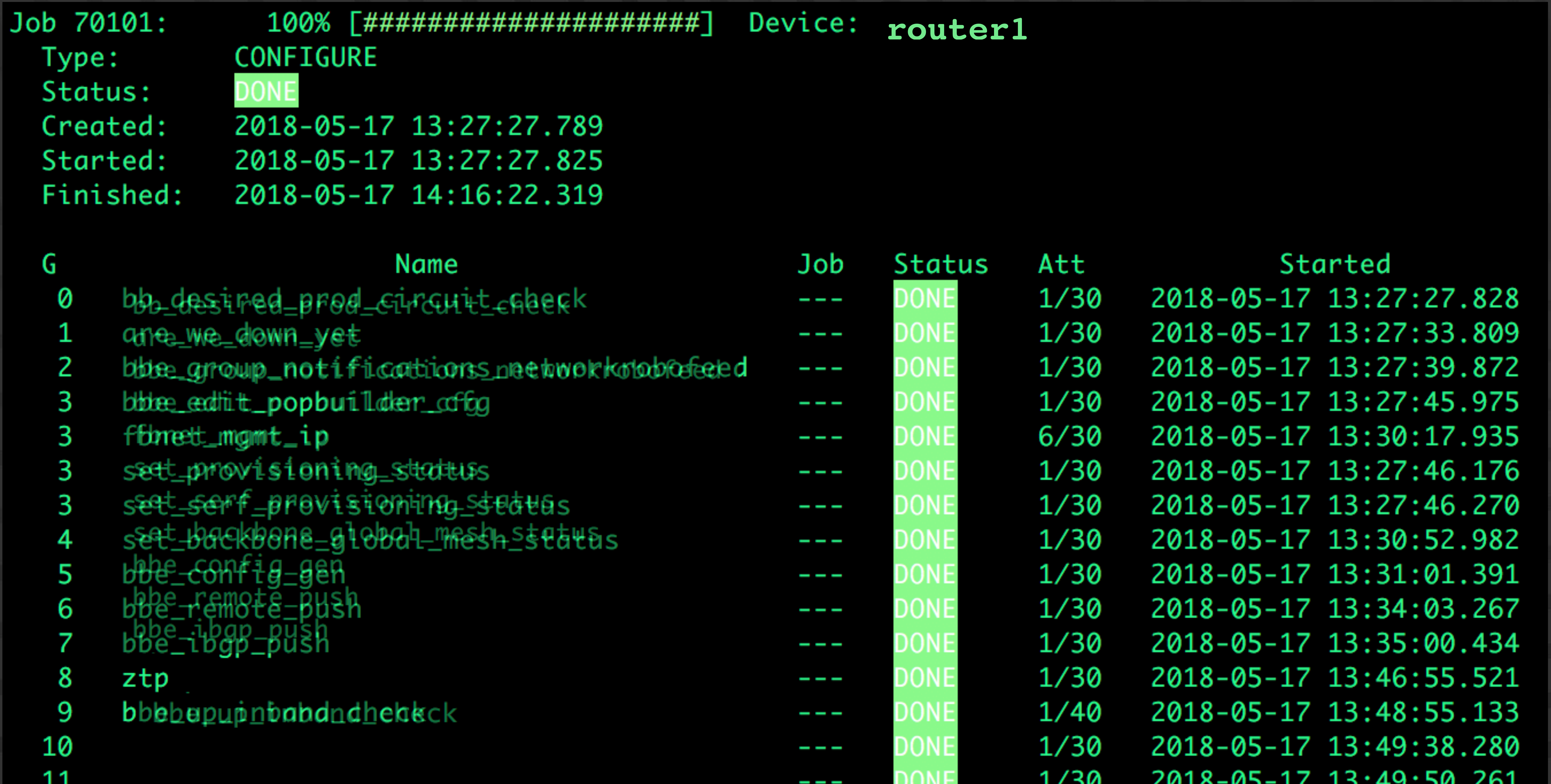


vm configure router1



```
Warning: `vm configure` is meant to be used only for a factory-blank device
If you are trying to (re)-configure an existing device, please use
`vm reconfigure` instead.

Continue (y/N):  y
Started CONFIGURE job 70110
run 'vm detail 70110' to see the job status
```

vm detail



```
Job 70101:       100% [######################]  Device:  router1
  Type:         CONFIGURE
  Status:       DONE
  Created:      2018-05-17 13:27:27.789
  Started:      2018-05-17 13:27:27.825
  Finished:     2018-05-17 14:16:22.319

  G                    Name                       Job    Status    Att           Started
   0   bb_desired_prod_circuit_check              ---    DONE      1/30   2018-05-17 13:27:27.828
   1   are_we_down_yet                            ---    DONE      1/30   2018-05-17 13:27:33.809
   2   bbe_group_notifications_networkrobofeed    ---    DONE      1/30   2018-05-17 13:27:39.872
   3   bbe_edit_popbuilder_cfg                    ---    DONE      1/30   2018-05-17 13:27:45.975
   3   fbnet_mgmt_ip                              ---    DONE      6/30   2018-05-17 13:30:17.935
   3   set_provisioning_status                    ---    DONE      1/30   2018-05-17 13:27:46.176
   3   set_serf_provisioning_status               ---    DONE      1/30   2018-05-17 13:27:46.270
   4   set_backbone_global_mesh_status            ---    DONE      1/30   2018-05-17 13:30:52.982
   5   bbe_config_gen                             ---    DONE      1/30   2018-05-17 13:31:01.391
   6   bbe_remote_push                            ---    DONE      1/30   2018-05-17 13:34:03.267
   7   bbe_ibgp_push                              ---    DONE      1/30   2018-05-17 13:35:00.434
   8   ztp                                        ---    DONE      1/30   2018-05-17 13:46:55.521
   9   bbe_inband_check                           ---    DONE      1/40   2018-05-17 13:48:55.133
  10                                              ---    DONE      1/30   2018-05-17 13:49:38.280
  11                                              ---    DONE      1/30   2018-05-17 13:49:50.261
```

vm log tail

```
   INFO   [bb_connectivity_check_mpls#1] Checking "show mpls lsp egress" command output o
   INFO   [bb_connectivity_check_mpls#1] Checking "show mpls lsp ingress" command output
   INFO   [bb_connectivity_check_mpls#1] LSPs Up: 363 egress and 317 ingress
   INFO   [bb_connectivity_check_mpls#1] LSPs Down: 0 egress and 0 ingress
   INFO   [bb_connectivity_check_mpls#1] LSPs Configured: 363 egress and 317 ingress
   INFO   [bb_connectivity_check_mpls#1] MPLS Health Checks Pass for router1
   INFO   [bb_connectivity_check_mpls#1] executor: attempt succeeded
   INFO   Job 70101 finished.
```

WHAT WOULD
YOU DO IF
YOU WEREN'T
AFRAID?