



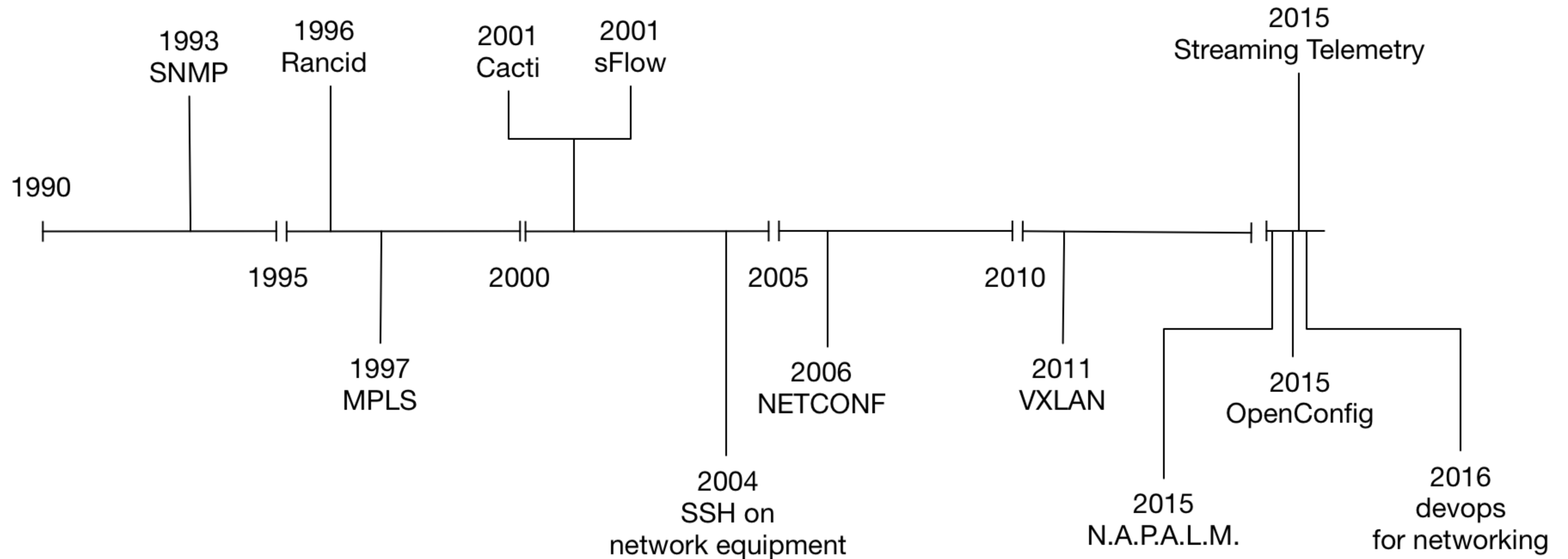
# PAST, PRESENT, AND **FUTURE** OF **NETWORK OPERATIONS**

---

David Barroso <dbarroso@**fastly**.com>

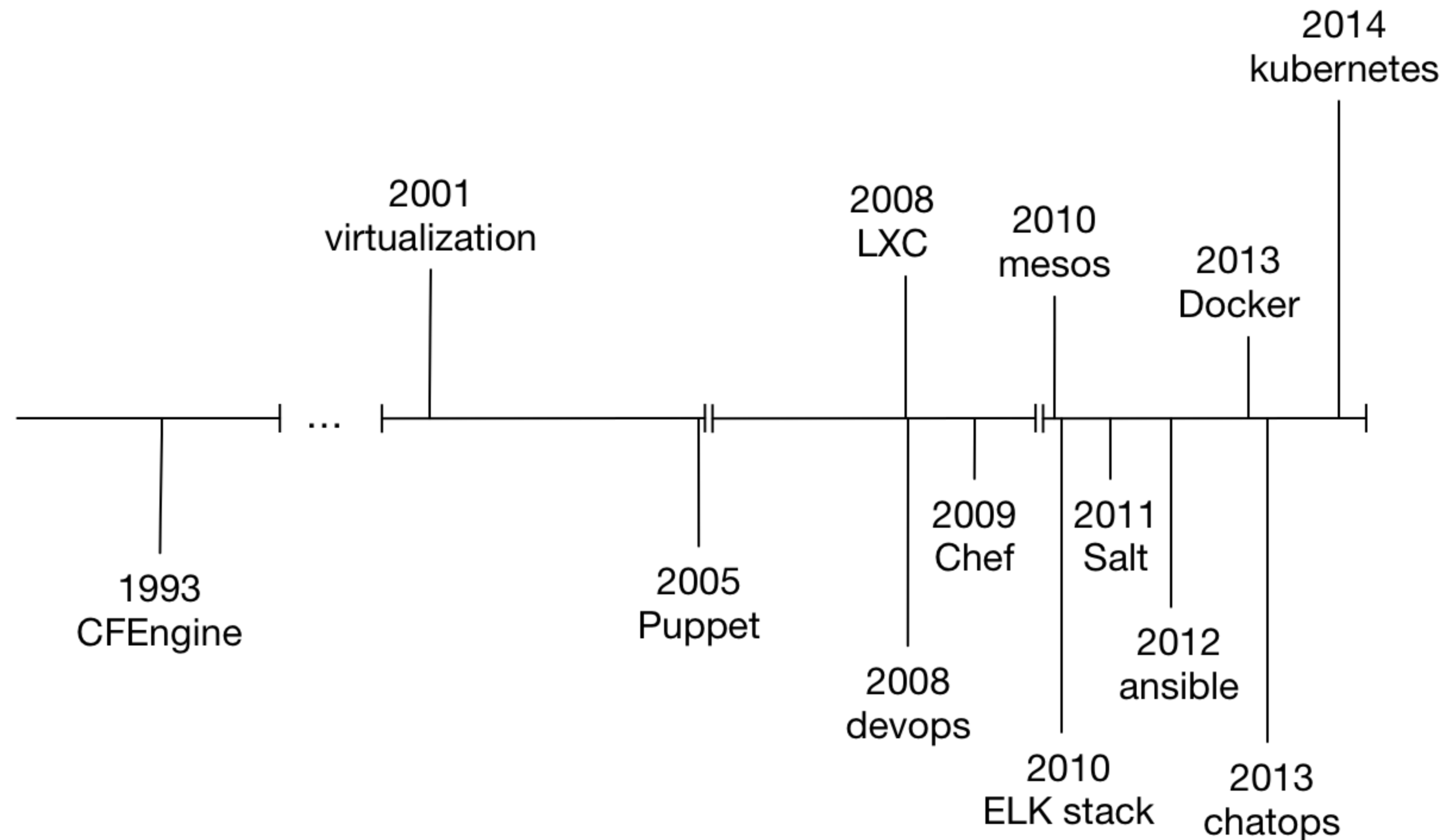
**A BIT OF HISTORY**  
**WHERE DO WE COME FROM?**

# EVOLUTION OF NETWORK OPERATIONS



From an ops perspective nothing minimally **interesting** happened for almost 10 years

# MEANWHILE IN THE **SYSTEMS** WORLD



Focus was on **breaking down** the app as much as possible

# WHY OH WHY?

- The network is a complex **distributed system**
  - Distributed database with **eventual consistency**? I call it {OSPF | IS-IS}
  - **Gossip** protocols? I call it BGP
- Dev guys kept asking us to enable st%#&d and **unique knobs** in the network to fix the app. Repeatable configurations down the toilet
- Focus was on complex protocols to **move packets** faster and build even more complex networks
- CCIE's can't learn anything but IOS and TCL and **love typing** command after command



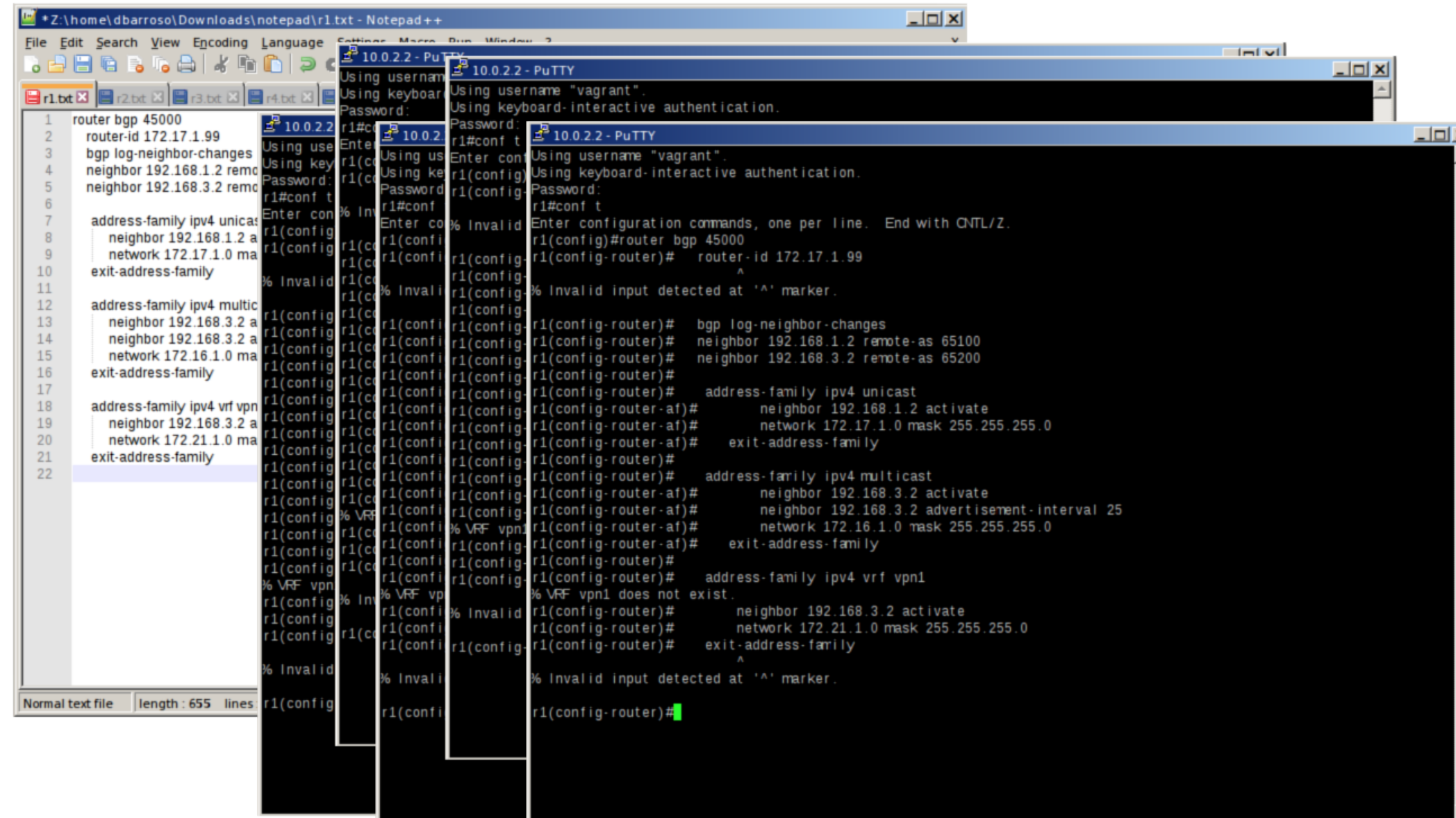
# VENDORS!!!



They think all the problems can be solved with yet another **AFI** or **encapsulation** mechanism



# BIG OPERATORS!!!



The image shows a Notepad++ window on the left with a configuration file named 'r1.txt'. The file contains the following configuration for a Cisco router:

```
1 router bgp 45000
2 router-id 172.17.1.99
3 bgp log-neighbor-changes
4 neighbor 192.168.1.2 remote-as 65100
5 neighbor 192.168.3.2 remote-as 65200
6
7 address-family ipv4 unicast
8 neighbor 192.168.1.2 activate
9 network 172.17.1.0 mask 255.255.255.0
10 exit-address-family
11
12 address-family ipv4 multicast
13 neighbor 192.168.3.2 activate
14 neighbor 192.168.3.2 advertisement-interval 25
15 network 172.16.1.0 mask 255.255.255.0
16 exit-address-family
17
18 address-family ipv4 vrf vpn1
19 neighbor 192.168.3.2 activate
20 network 172.21.1.0 mask 255.255.255.0
21 exit-address-family
22
```

On the right, there are three overlapping PuTTY terminal windows. The top window shows the login process for a user named 'vagrant'. The middle window shows the start of the configuration process, including the command 'router bgp 45000' and 'router-id 172.17.1.99'. The bottom window shows the configuration of the BGP neighbors and address families, including 'address-family ipv4 unicast', 'address-family ipv4 multicast', and 'address-family ipv4 vrf vpn1'. The terminal output shows the configuration commands being entered and the resulting configuration on the router.

Major Tier 1 service providers still manage the network **manually** causing Internet **outages** and taking **3 months** for enabling a port

**/ RANT**

**BECAUSE THERE IS HOPE**



## VENDOR ABSTRACTIONS

- **N.A.P.A.L.M.** - One API to rule them all
- **OpenConfig** - Vendor-neutral, model-driven network management designed by users

# STREAMING TELEMETRY

Pub / Sub service to **stream network telemetry** using vendor-neutral models.

# ROOT ACCESS TO UNDERLYING **LINUX** OS



Even to the **ASIC** via SDKs

# DEVOPS **FOR NETWORKING**



# USE CASES

- Network **abstractions** || Infra as code
- Running **code** on the network
- **CI pipeline** for TE changes



**NETWORK ABSTRACTIONS || INFRA AS CODE**

# OPERATIONS ARE **ABSTRACTED**

```
(ansibly) → ansible git:(master) X inv --list
Available tasks:
  deploy.base          Deploy Base Role. Check Base role documentation for
  deploy.faild        Deploy faild. Check faild role documentation for f
  ...
  dev.ipamd_start     Start docker container with ipamd.
  ...
  ops.peer_disable    Disable a peer on a device.
  ops.peer_enable     Enable a peer on a device.
  ops.pop_drain        Drain a POP for maintenance.
  ops.pop_undrain     Put POP in production.
  ...
  package.install     Deploy a package. Check metarole_service role docu
  package.list        Get a list of overridden packages.
  ...
  preflight_checks.all Run all the preflight checks.
  preflight_checks.ipv6 Run preflight checks for IPv6 role.
  ...
  show.bgp_neighbors  Get BGP neighbor state and statistics from the dev
  show.pop            Check POP health
  ...
```

# OPERATIONS ARE **EXPOSED** VIA API'S AND CHATOPS

The image shows a split-screen view. On the left is an Ansible terminal window with the following commands and output:

```
(indirection_preso) → ansible # add a link
(indirection_preso) → ansible curl -k -H "Content-Type: application/json" -X POST https://172.28.128.3/api/v1/webhooks/add_link -H "St2-API-Key: NmYyZWVmNmYxYmZhM" --data '{"left": "eos.spine1", "left_port": "eth3", "right": "eos.spine2", "right_port": "eth3"}'
(indirection_preso) → ansible # deploy fabric
(indirection_preso) → ansible curl -k -H "Content-Type: application/json" -X POST https://172.28.128.3/api/v1/webhooks/deploy_fabric -H "St2-API-Key: NmYyZWVmNmYxYmZhM" --data '{}'
```

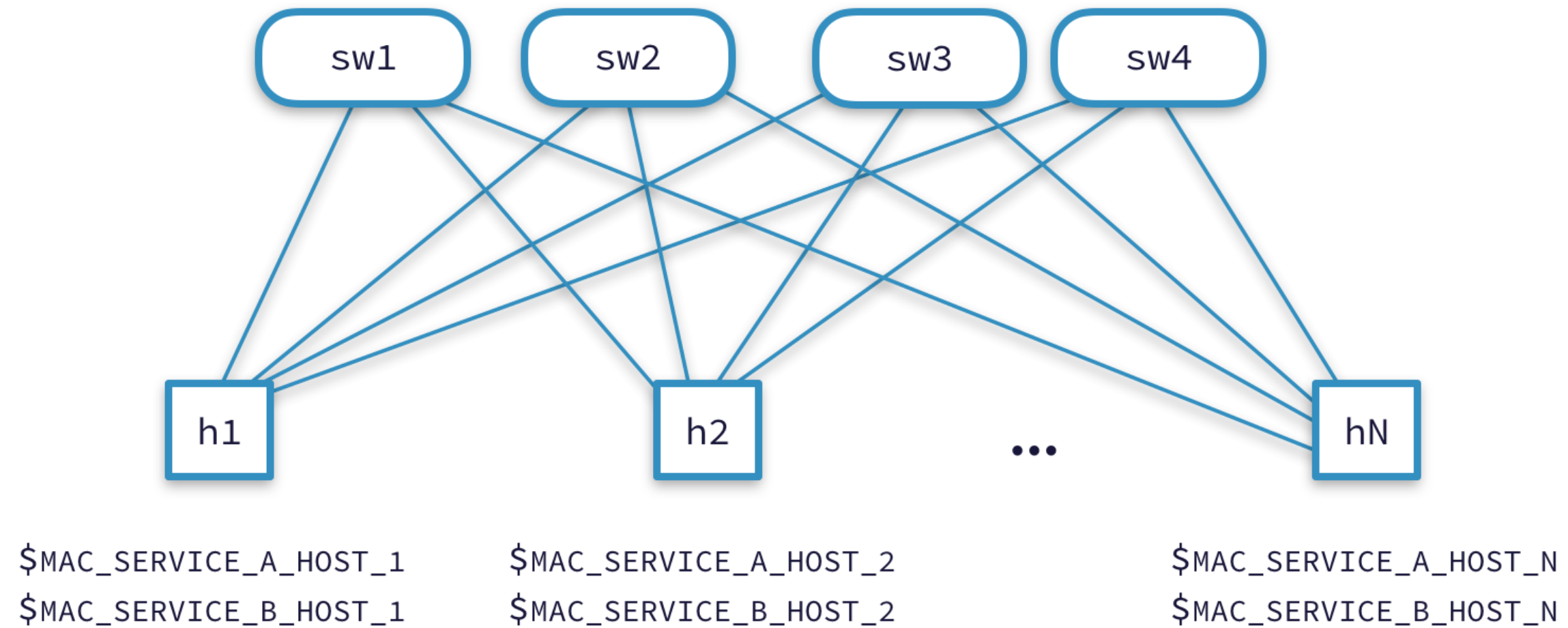
On the right is a Slack chatops window titled "chatops\_demo\_dbarroso" with 2 members. A bot named "hal" has sent a message:

@channel Awaiting for confirmation for task 57482818421aa93ec7bd89b7:

```
# Deploy Configuration
*****
*****
* vmx.core2          - changed=False -- -----
* vmx.core1          - changed=False -- -----
* eos.spine2         - changed=True  -----
@@ -67,10 +67,13 @@
neighbor 2001:db8:c4f3:4:: maximum-routes 12000
neighbor 2001:db8:c4f3:5:: remote-as 65001
neighbor 2001:db8:c4f3:5:: maximum-routes 12000
+ neighbor 2001:db8:c4f3:6:: remote-as 65001
+ neighbor 2001:db8:c4f3:6:: maximum-routes 12000
address-family ipv6
neighbor 2001:db8:c4f3:3:: activate
neighbor 2001:db8:c4f3:4:: activate
neighbor 2001:db8:c4f3:5:: activate
+ neighbor 2001:db8:c4f3:6:: activate
network 2001:db8:b33f::7/128
!
management api http-commands
* eos.spine1          - changed=True  -----
```

**RUNNING CODE ON THE NETWORK**

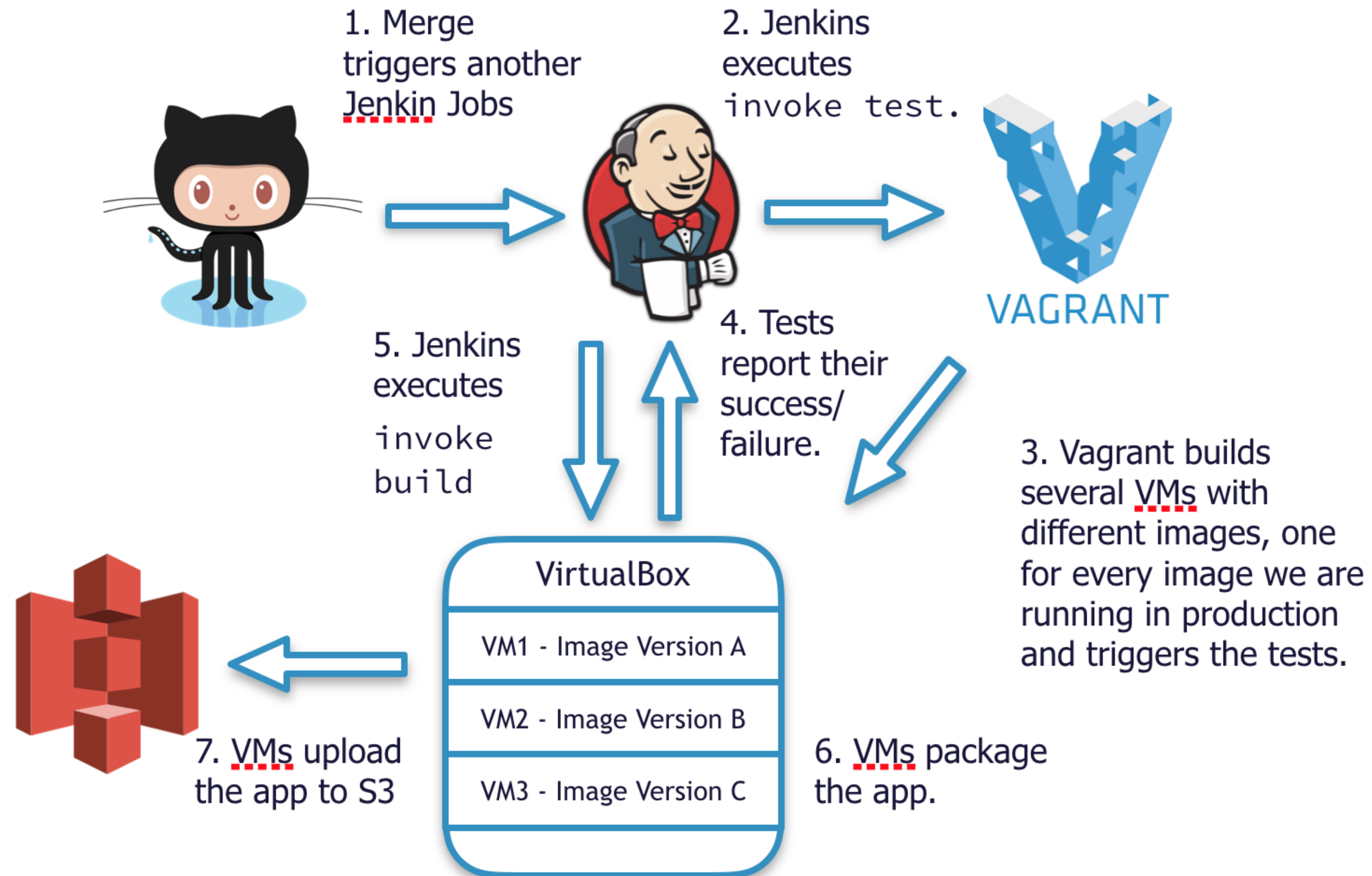
**FAILED**



- Turns switches into hardware load balancers
- An agent running on the switch communicates with hosts
- The agent programs the ASIC
- The application is not really the point of this talk

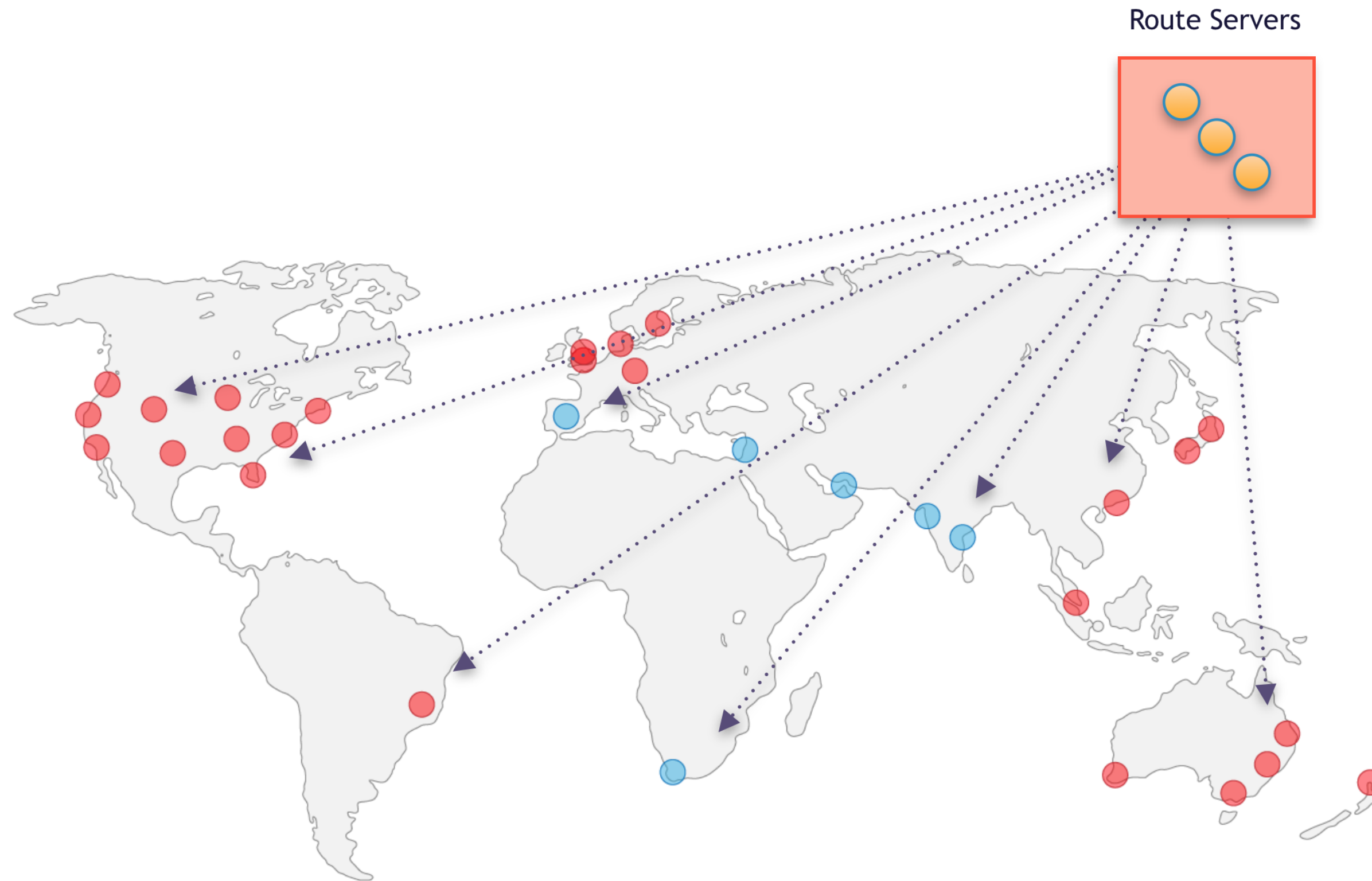


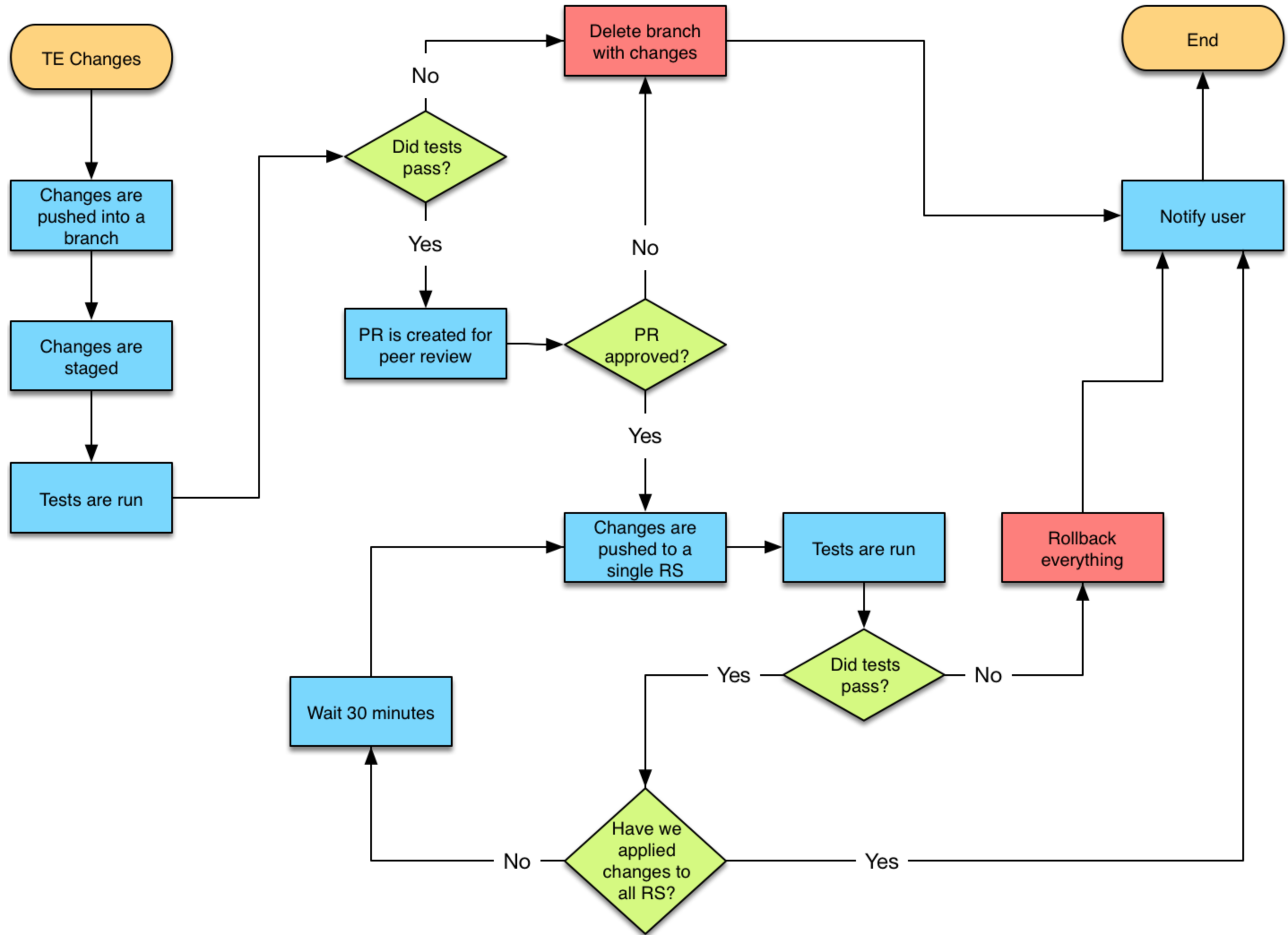
# WRITING NETWORK SOFTWARE



# CI PIPELINE FOR TE CHANGES

# BISHOP - TRAFFIC ENGINEERING PIPELINE





## SUMMARY

- The network is late to the party but some folks are **catching up**
- Do not reinvent the wheel. Use the **same tools & principles** that `sysadmin/developers` have been using for years
- **Do not listen** to `$vendors` that won't give you `root` access because they are lying
- The sole purpose of this is to make your job **more interesting**. A `python` script is not going to steal your job



**QUESTIONS?**