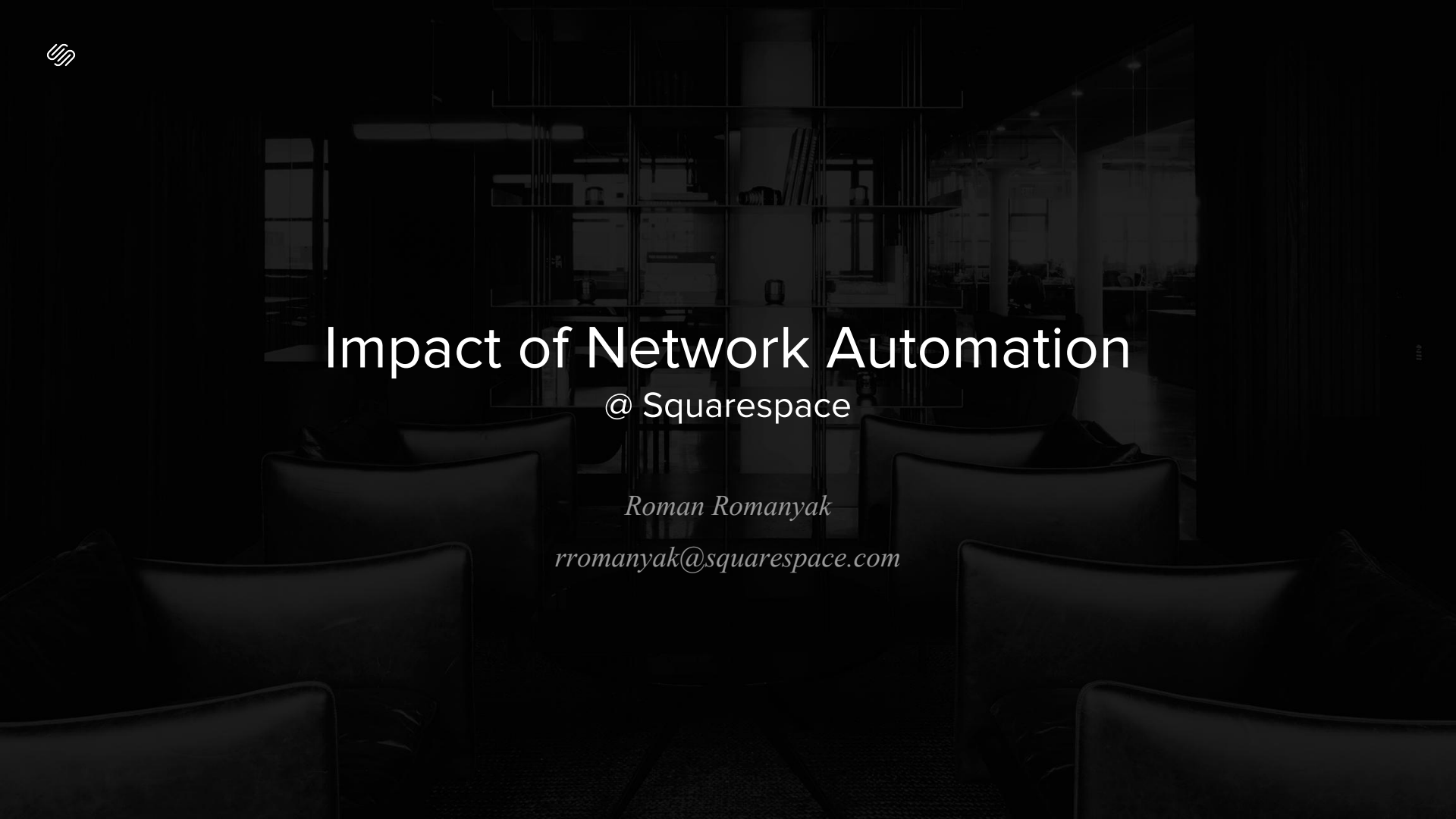


Impact of Network Automation

@ Squarespace

Roman Romanyak

rromanyak@squarespace.com

Agenda

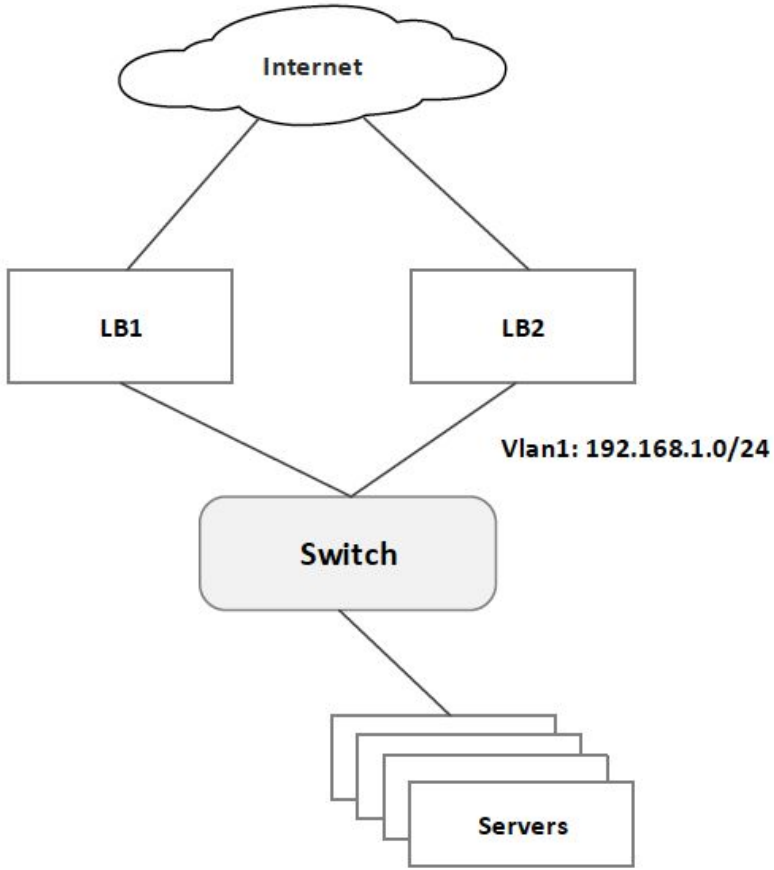
- *Evolution of Network Architecture*
- *Network Automation use-cases*
- *Using route-servers for traffic engineering*
- *Monitoring*

Terminology

- ***BGP*** *- exterior routing protocol used for inter-Autonomous System (AS) routing.*
- ***Autonomous System*** *- a network under one administrative domain with single routing policy*
- ***ASN*** *- autonomous system number assigned to AS (globally unique and private)*
- ***Neighbors / Peering*** *- the relationship between routers that exchange routing information over BGP*
- ***iBGP / EBGP*** *- neighbors belong to the - same AS / different AS*
- ***AS-Path*** *- well known mandatory attribute. When a route propagates from network to network the ASn is added to the AS Path. Used to prevent loops*
- ***BGP community*** *- labels attached to BGP routes and advertised to other neighbors.*
- ***Route-Server*** *- networking device that does not participate directly in routing, but carries an entire routing table.*

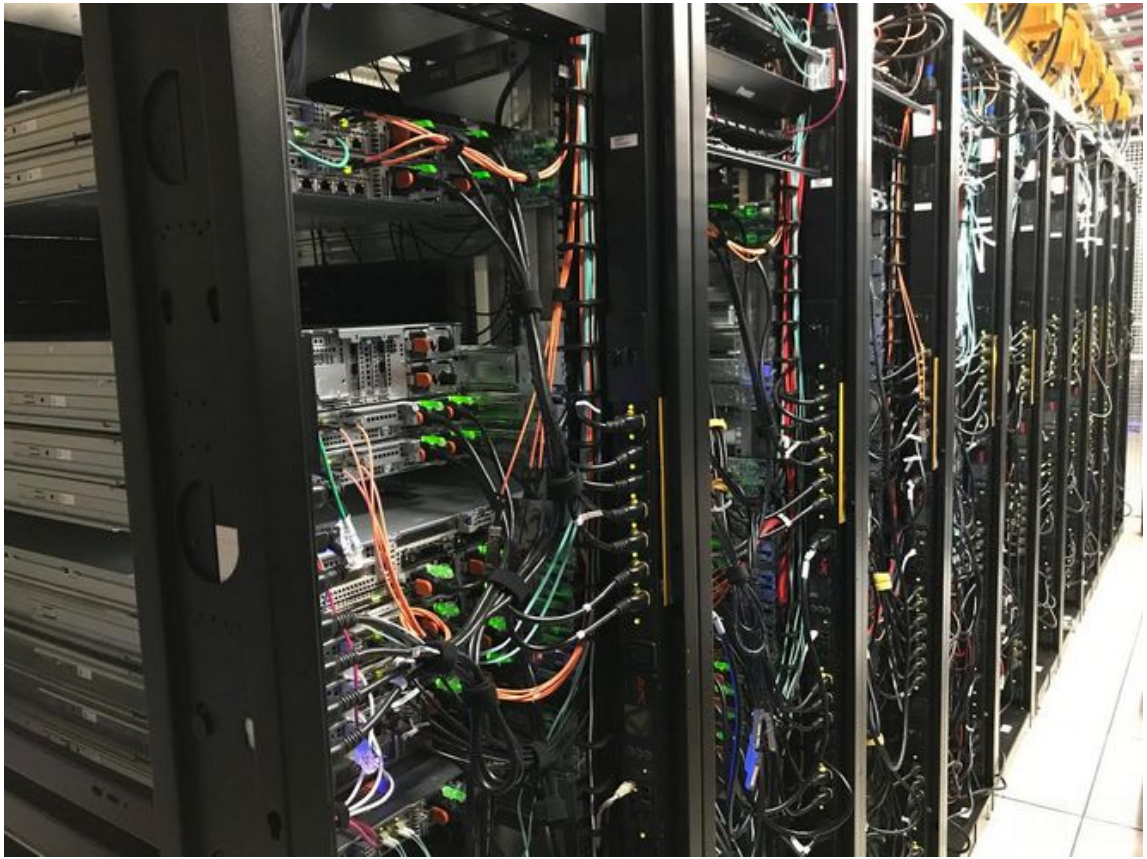
- 2012

- *A pair of LBs with direct transit links*
- *Single core switch*
- *One production Vlan*



2013-2015

- *Multi-datacenter*
- *Dynamic routing*
- *Backbone, WAN & LAN*
- *Layer-3 / CLOS topology*



2015: New Office

New headquarters network:

- *Based on Layer-3*
- *Dynamic routing based on BGP*
- *Security policies managed via Ansible*

2015: New Office

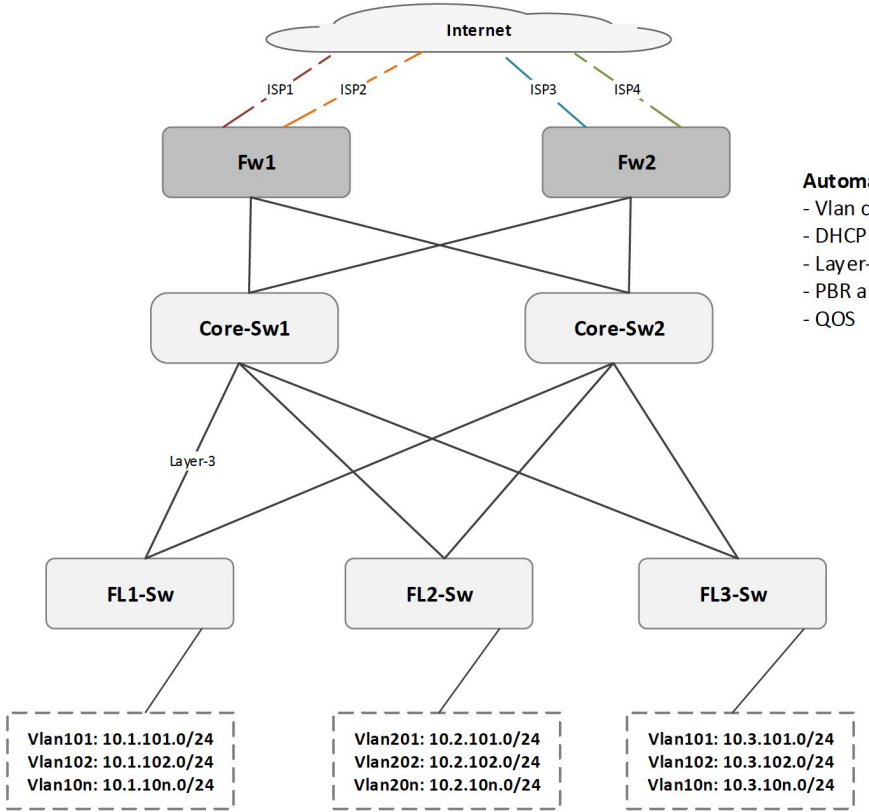
New headquarters network:

- *Based on Layer-3*
- *Dynamic routing based on BGP*
- *Security policies managed via Ansible*

2015: New Office

New headquarters network:

- *Based on Layer-3*
- *Dynamic routing based on BGP*
- *Security policies managed via Ansible*



Automated:
- Vlan configuration
- DHCP Pools & Static bindings
- Layer-2 and Layer-3 access lists
- PBR and internet failover
- QOS

Automation. Low risk tasks.

- *Low risk, simple problems*
- *Learn how to efficiently reuse the code*
- *Use available developers tools: Git, Unit-testing, CI-Tools*

First use-cases: syslog, snmp, system authentication, ntp, dns, bgp communities, etc.

Automation Workflow

- *Generate configuration based on the variables and templates*
- *Run a playbook to push changes in the dry-run mode*
- *Review diff files*
- *Run a playbook to push changes to the network nodes*

Automation Workflow: templates

```
    <system operation='replace'>
       <host-name>{{ host_basename }}</host-name>
       <domain-search>{{ domain }}</domain-search>
       <time-zone>{{timezone}}</time-zone>
       <domain-name>{{ domain_name }}</domain-name>
{% for dns_server in dns_servers %}
       <name-server>
          <name>{{ dns_server  }}</name>
       </name-server>
{% endfor %}
{% for radius_server in radius_servers %}
       <radius-server>
          <name>{{ radius_server['ip_addr']  }}</name>
          <secret>{{ radius_server['secret']  }}</secret>
       </radius-server>
{% endfor %}

....
    </system>
```

Automation Workflow: build playbook

```yaml
---

# ============================================================
#  Build system configuration on Prod Node
# ============================================================

- name: Build system configuration on Prod Node
  hosts: [redacted]
  vars_files:
    - "{{ inventory_dir }}/group_vars/corp_network/prod_network.yml"
  connection: local
  gather_facts: no

  roles:
    - {role: net-system }


# ============================================================
#  Finish by assembling config snippets into single file
# ============================================================

- name: Create final Configuration
  hosts: [redacted]
  connection: local
  gather_facts: no
  tasks:
    - name: assembling configurations
      assemble: src={{ build_dir }} dest={{ config_file }}

    - name: wrapping xml
      shell: "{{ inventory_dir }}/scripts/wrap_xml.sh {{ config_file }}"
      tags:
        - always

    - name: validate XML
      shell: "{{ inventory_dir }}/scripts/validate_xml.py {{ config_file }}"
      tags:
        - always
```

Automation Workflow: deploy playbook

```
---
- name: Deploy configurations
  hosts: leaf_switches
  connection: local
  gather_facts: no
  tasks:
    - name: Deploy configuration
      install_config:
        host={{ inventory_hostname }}
        overwrite=false
        replace=true
        timeout=60
        file={{ config_dir  }}/config.xml
        logfile={{ log_dir }}/{{ inventory_hostname }}.log
        diffs_file={{ diffs_dir }}/{{ inventory_hostname }}.diff
```


Firewall configuration automation

Firewall configuration automation

- *Making sure configuration is consistent across multiple firewalls*
- *Optimizing operations and workflows*
- *Delegating and spreading workload*

Firewall address book and security policies

To effectively manage firewalls, we developed a set of ansible roles that have similar functions to commercial security software.

These ansible roles include:

- *Address-book entries and address-sets are build automatically from ansible inventory files*
- *Security policies (from zone to zone rules) are kept in files, grouped per device*
- *Application and Application Set to refer to the name of one or more applications in the policy*

Firewall address book and security policies

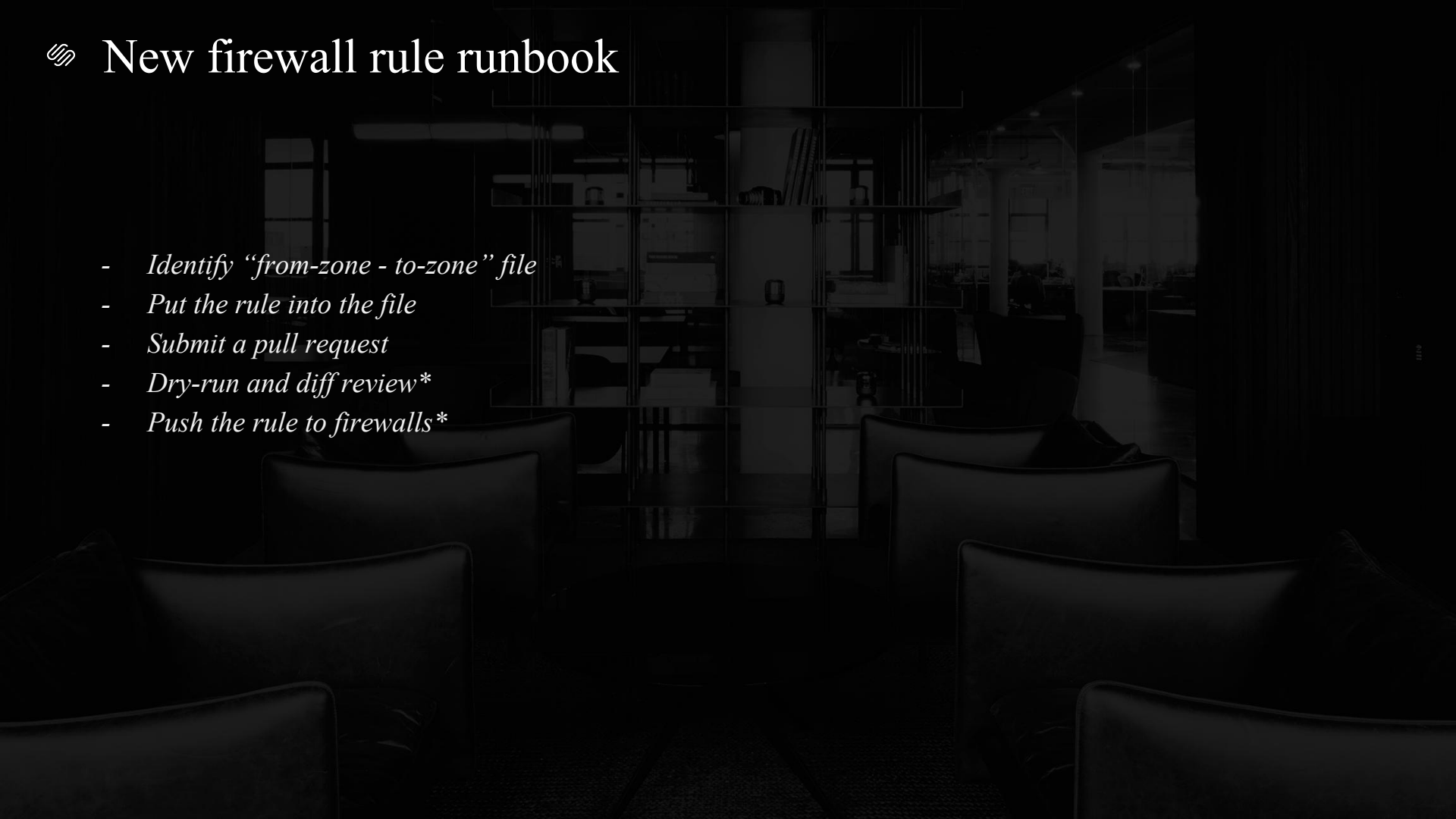
```
---
device_groups:
   dal_fw:
      devices:
         - da-fw001
         - da-fw002
      zone_policies:
         - TRUST_2_TRUST
         - TRUST_2_UNTRUST
         - TRUST_2_VPN
         - MGT_2_TRUST
         - MGT_2_STAGING
         - MGT_2_VPN
         - TRUST_2_MGT
   ewr_fw:
      devices:
         - nj-corp-fw001
         - nj-corp-fw002
      zone_policies:
         - PROD_2_PROD
         - PROD_2_MGT
         - PROD_2_UNTRUST
         - PROD_2_STAGING
```

```
rromanyak@box ~/git/ansible/roles/net-srx-security-policies/vars/security-zones/da_fw> ls
MGT_2_STAGING.yml
MGT_2_TRUST.yml
MGT_2_VPN.yml
TRUST_2_MGT.yml
TRUST_2_TRUST.yml
TRUST_2_UNTRUST.yml
TRUST_2_VPN.yml
UNTRUST_2_TRUST.yml
```

```
---
MGT_2_TRUST:
   policies:
      - PERMIT-TELEMETRY:
           source_address:
              - SRC_ADDRESS
           destination_address:
              - DEST_ADDRESS
           application:
              - JUNOS_TELEMETRY
           action: permit
           log: no
      - DENY-ALL:
           source_address:
              - any
           destination_address:
              - any
           application:
              - any
           action: deny
           log: yes
```

```
---
applications:
   VCENTER:
      - app: vcenter-tcp
      - app: vcenter-udp
      - app: vcenter-console-tcp

   JUNOS_TELEMETRY:
      - app: UDP_50000
      - app: UDP_50020
      - app: UDP_6000

   NETFLOW:
      - app: UDP_9964
      - app: UDP_9965

   VNC:
      - app: TCP_5800
      - app: TCP_5900
```

New firewall rule runbook



- *Identify “from-zone - to-zone” file*
- *Put the rule into the file*
- *Submit a pull request*
- *Dry-run and diff review**
- *Push the rule to firewalls**

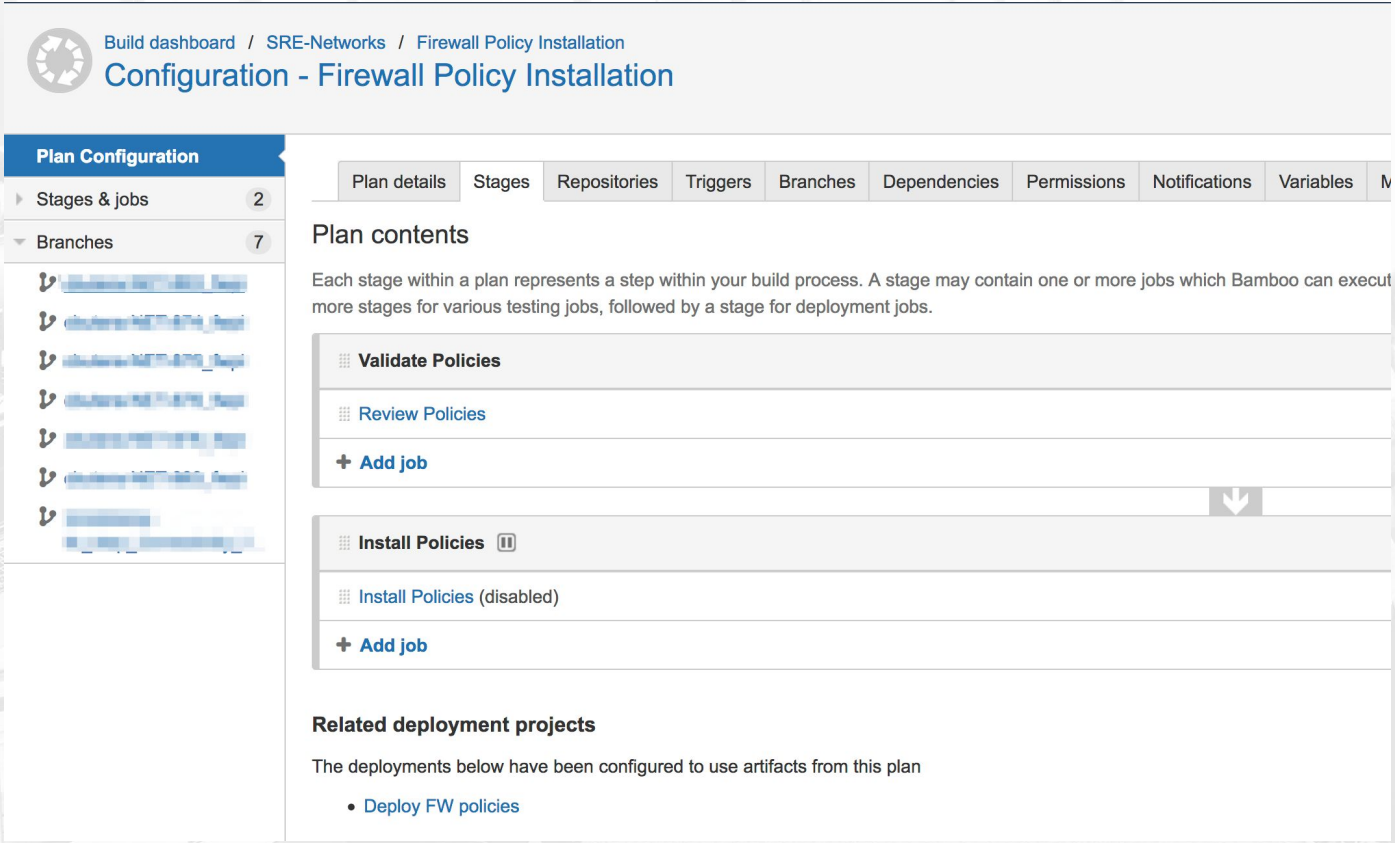
New firewall rule runbook

- *Identify “from-zone - to-zone” file*
- *Put the rule into the file*
- *Submit a pull request*

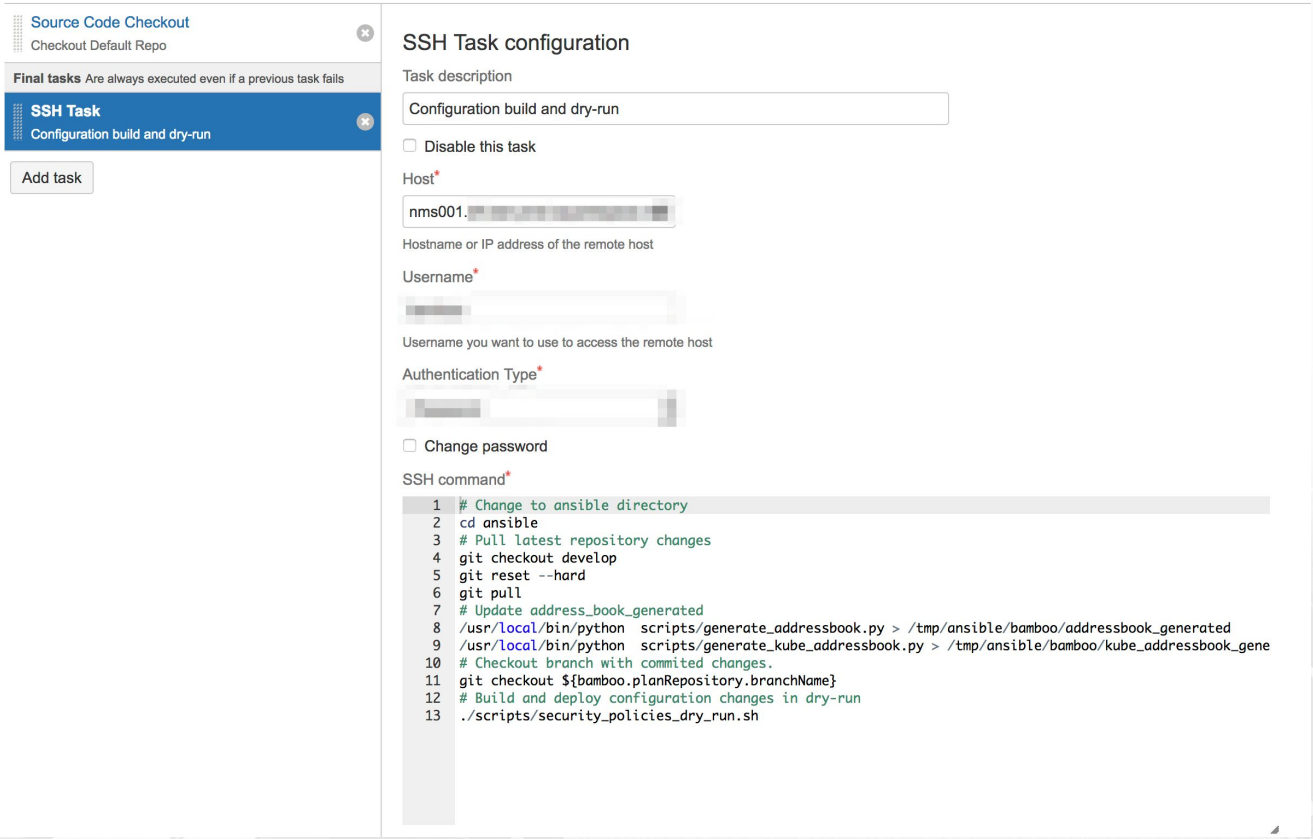
These tasks are the most time consuming and are automated using Bamboo:

- ***Dry-run and diff review****
- ***Push the rule to firewalls****

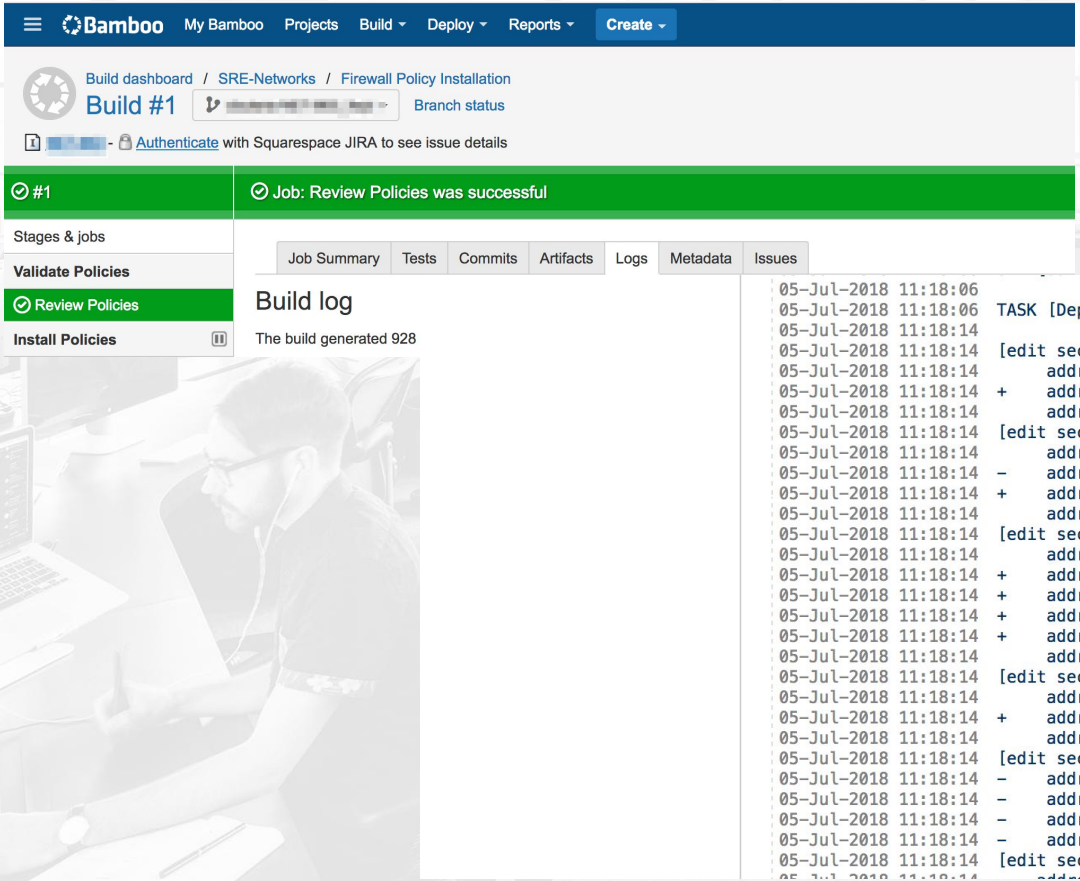
Bamboo plan to push a firewall rule



Bamboo plan to push a firewall rule



Bamboo plan to push a firewall rule



Spine and Leaf Network Overview and Automation

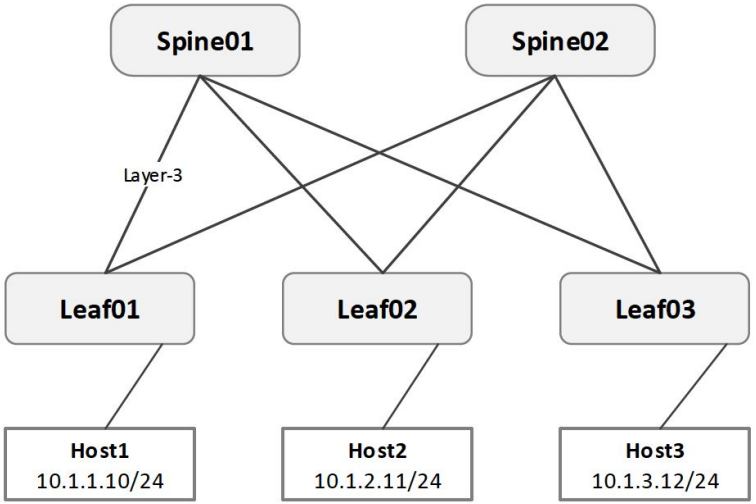
Spine and Leaf Network

Previous data-center network built on layer-2 with Core and Top-of-rack switches

Then we moved to using layer-3 clustered architecture:

- *high throughput and low oversubscription ratio*
- *built with small identical switches*
- *good for east-west traffic*
- *all routers configured in active state and use of ECMP*

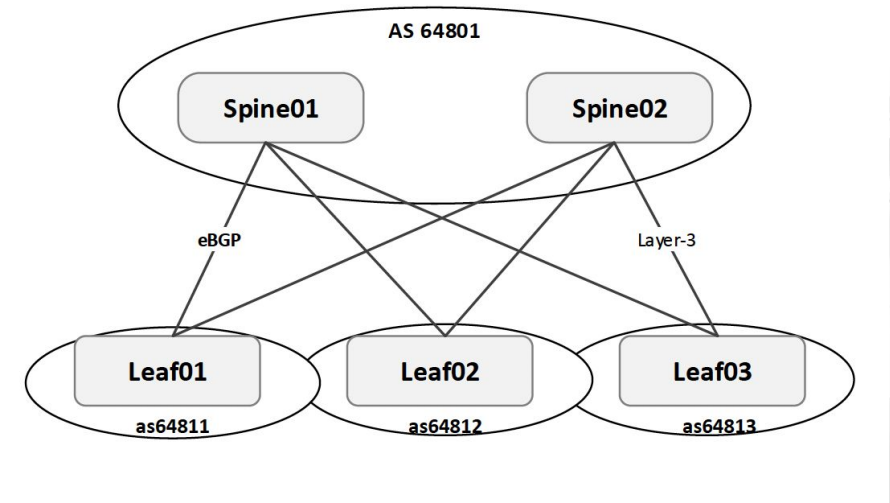
Network based on Layer-3 - Spine-and-Leaf (SL)



Layer-3 in CLOS:

- All network interconnects are layer-3
- Size of layer-2 domain is limited to a leaf switch
- Predictive latency - every node is 3 hops away
- Heavy use of equal cost multipath (ECMP)
- BGPv4 the only routing protocol
- No layer-2 overlays

Spine-and-Leaf POD

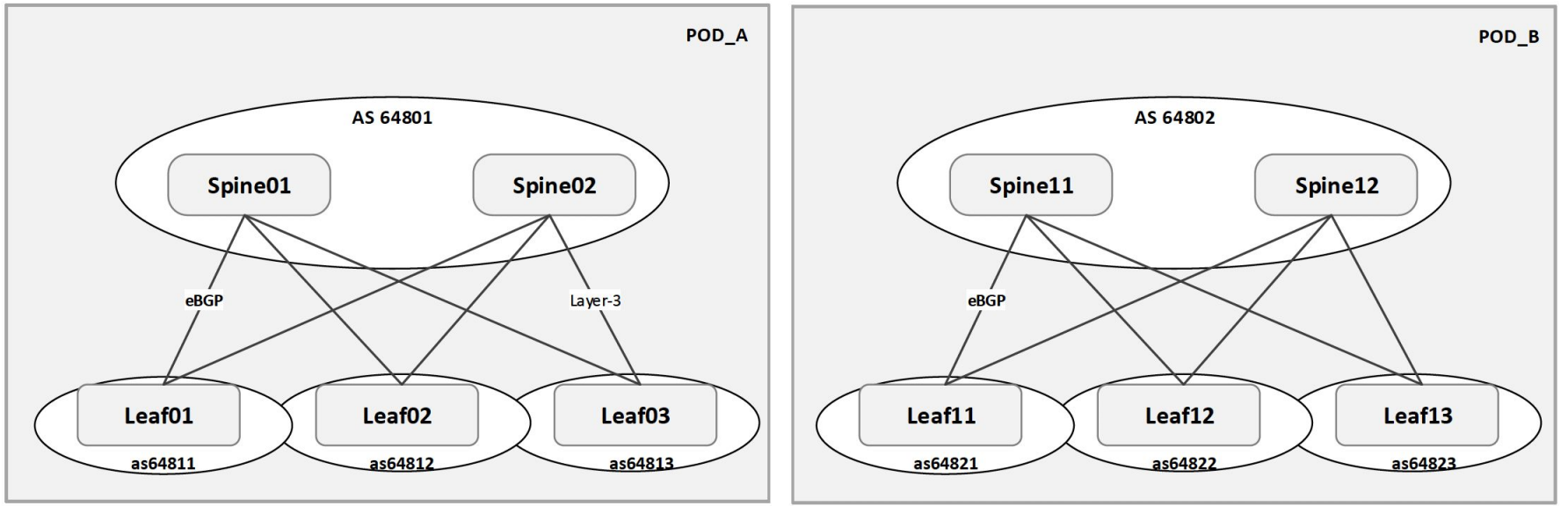


eBGP as a routing protocol of choice:

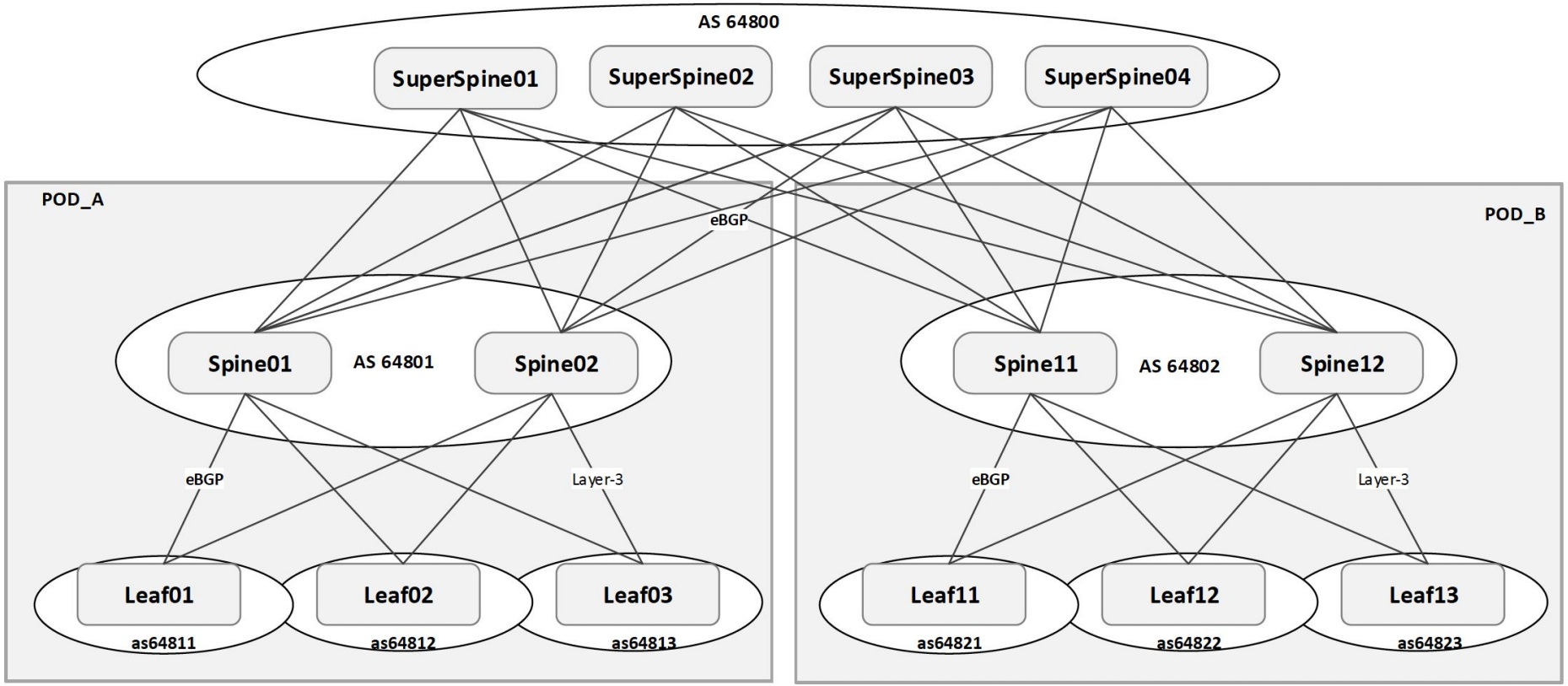
- Less complexity in protocol design
- Relies on TCP rather than adjacency formation
- Control of routing advertisements
- Built-in loop prevention via AS_PATH
- Better support for traffic engineering

Scaling out SL network

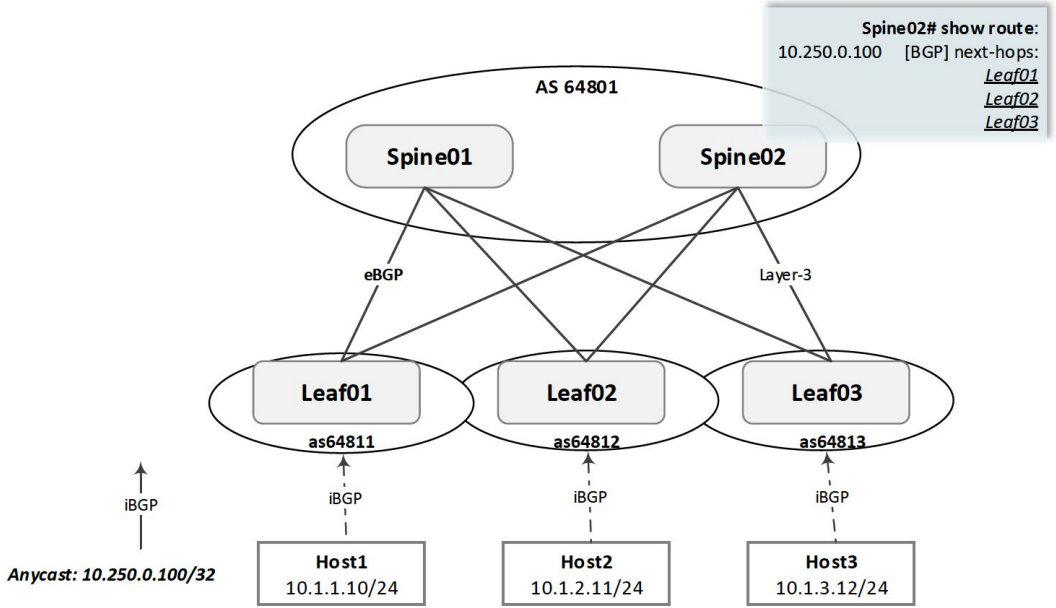
Modular network - multiple PODS



Scaling out SL network

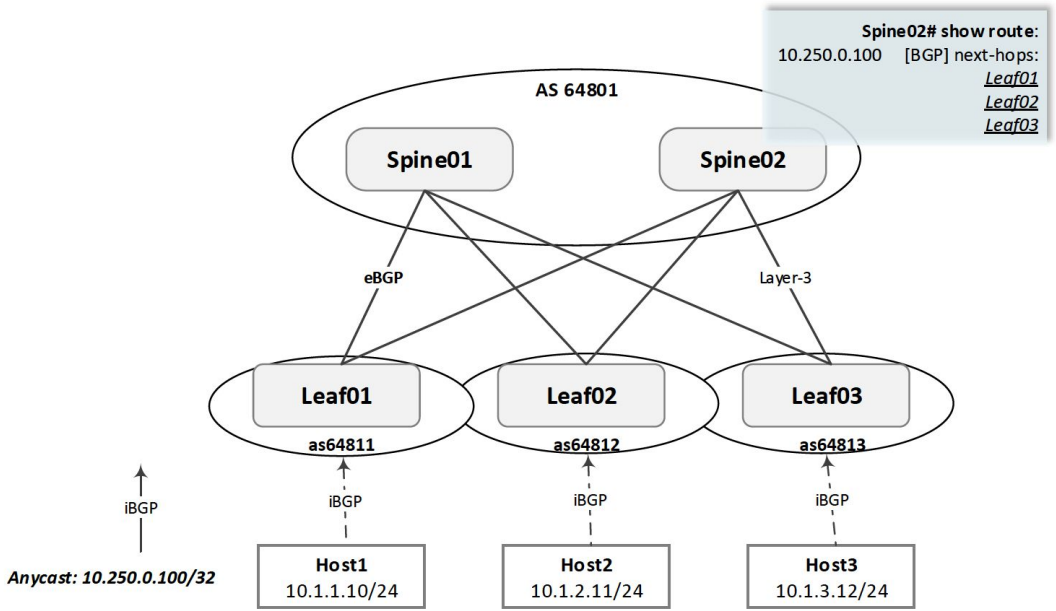


Routing in SL network - Anycast



The same IP address is assigned to n servers and distributed across the datacenter.

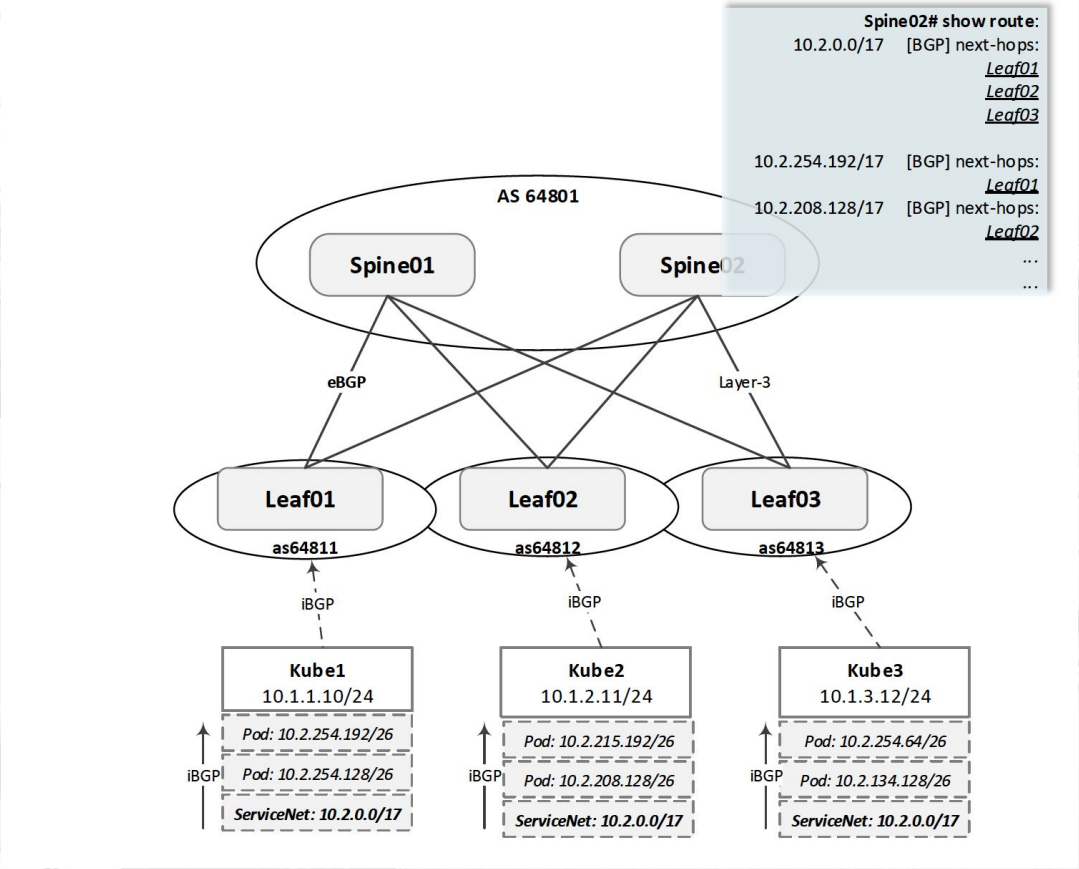
Routing in SL network - Anycast



Services using anycast:

- LDAP
- Radius
- Various observability tools
- Kafka
- Ceph

Routing in SL network - Kubernetes



Operational and scaling issues

- *Managing p2p links*
- *Routing protocols management.*
- *Network policies.*
- *Network assignments / allocations*
- *Security*
- *Deliver configuration changes at scale*

Spine and Leaf Network Automation

- *Modular structure*
- *Policies and prefix-lists, aggregate routes, anycast*
- *Routing protocol (BGP) and peering with servers and Kubernetes nodes*
- *Route Servers integration*

Spine and Leaf Ansible Role

Global Configuration of multistage SL-Net

```
# Spine and Leaf Networks data
eqx.dal:
 summary_routes:
   prod:
     prefixes: [10.22.0.0/15]
   stage:
     prefixes: [10.12.160.0/20]
 service_routes:
   loopbacks: [10.12.0.0/24]
   anycast:
     prod:
       # originates only in the current
datacenter
       intradc: [10.22.0.0/24]
       # originates in one or more datacenters
       interdc: [10.25.0.0/20]
     stage:
       intradc: [10.12.87.0/24]
       interdc: [10.25.16.0/20]
     corp:
       intradc: [10.12.77.0/24]
       interdc: [10.25.32.0/20]
 kubernetes_networks:
   prod:
     aggregates: [10.13.0.0/16]
     service: [10.13.0.0/17]
     direct: [10.13.128.0/17]
   stage:
     aggregates: [10.18.0.0/16]
```

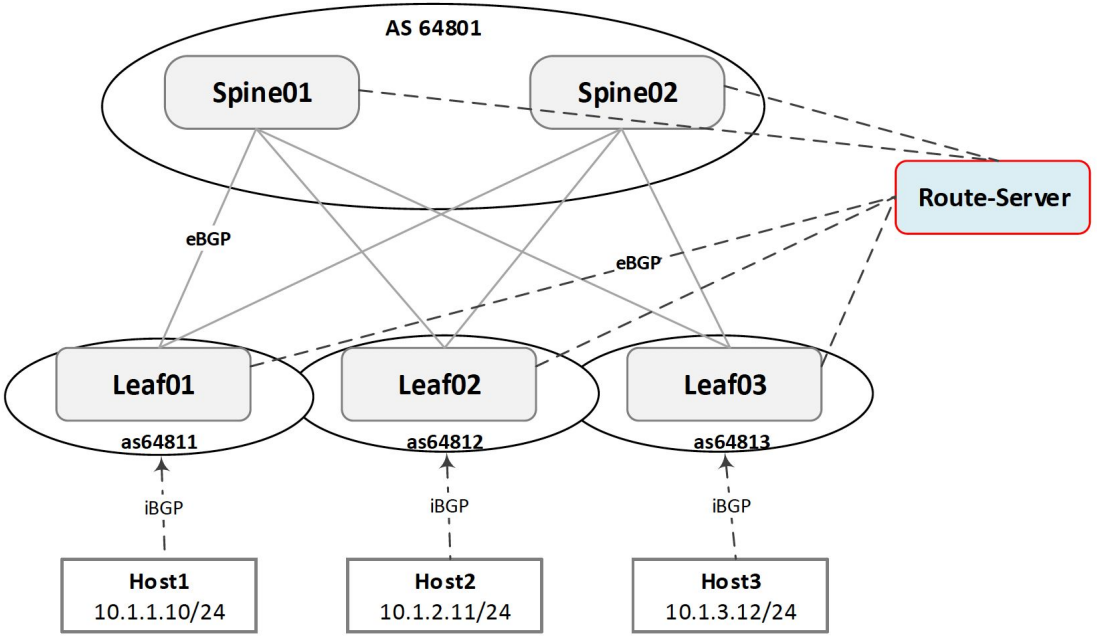
Spine and Leaf Ansible Role

POD-specific variables - used to build leafs and spines interfaces, VRFs, routing protocol, etc.

```
spine_nodes:
  - name: da-spine-sw001
    asn: 64701
    lo: 10.12.0.221
    tag: A
    RS_PEERS:
      - sl-route001
      - sl-route002
    peers:
      - peer_name: da-leaf-sw201
        local_if: et-0/0/0
        peer_if: et-0/0/27
        net_addr: 192.168.210.0/31

      - peer_name: da-leaf-sw202
        local_if: et-0/0/6
        peer_if: et-0/0/27
        net_addr: 192.168.210.4/31
```

```
leaf_nodes:
  - name: da-leaf-sw201
    asn: 64711
    lo: 10.120.0.201
    leaf_networks:
      prod: 10.12.1.0/24
      corp:  10.12.177.0/24
      stage: 10.12.161.0/24
    RS_PEERS:
      - sl-route001
      - sl-route002

  - name: da-leaf-sw202
    asn: 4712
    lo: 10.12.0.202
    leaf_networks:
      prod: 10.12.2.0/24
    RS_PEERS:
      - sl-route001
      - sl-route002
```

Routing in SL network - Route-Servers



- Peers over eBGP to all network nodes
- Private looking glass
- Drain traffic

Internal Route Servers

```
#BIRD Route Server configuration snippet:

function EXPORT_TO_SPINE(string drain){
       if (drain = "drain" && net = 10.0.192.120/32 )
then {
            # adding no-advertise community
            bgp_community.add((65535, 65282));
              return true;
       }
       else
         return false;
}


#_Default_:
protocol bgp INBOUND_da_spine_sw001 from
INBOUND_PEERS{
       neighbor 10.12.0.221 as 64701;
       export where EXPORT_TO_SPINE("none");
       import none;
}

#_Drain_Spine_:
protocol bgp INBOUND_da_spine_sw001 from
INBOUND_PEERS{
       neighbor 10.12.0.221 as 64701;
       export where EXPORT_TO_SPINE("drain");
       import none;
}
```

```
#Spine Switch configuration snippet

da-spine-sw001> show configuration policy-options policy-statement
ROUTE_OFF_MED100
from condition ROUTE_OFF;
then {
    metric 100;
    next policy;
}

{master:0}
da-spine-sw001> show configuration policy-options condition
ROUTE_OFF
if-route-exists {
    10.0.192.120/32;
    table inet.0;
}


> show configuration protocols bgp group SUPERSPINE export
export [ ROUTE_OFF_MED100 da-ip-fabric-specifics INTER-DC-ANYCAST
REJECT-ALL ];
```

Edge Traffic Manipulation

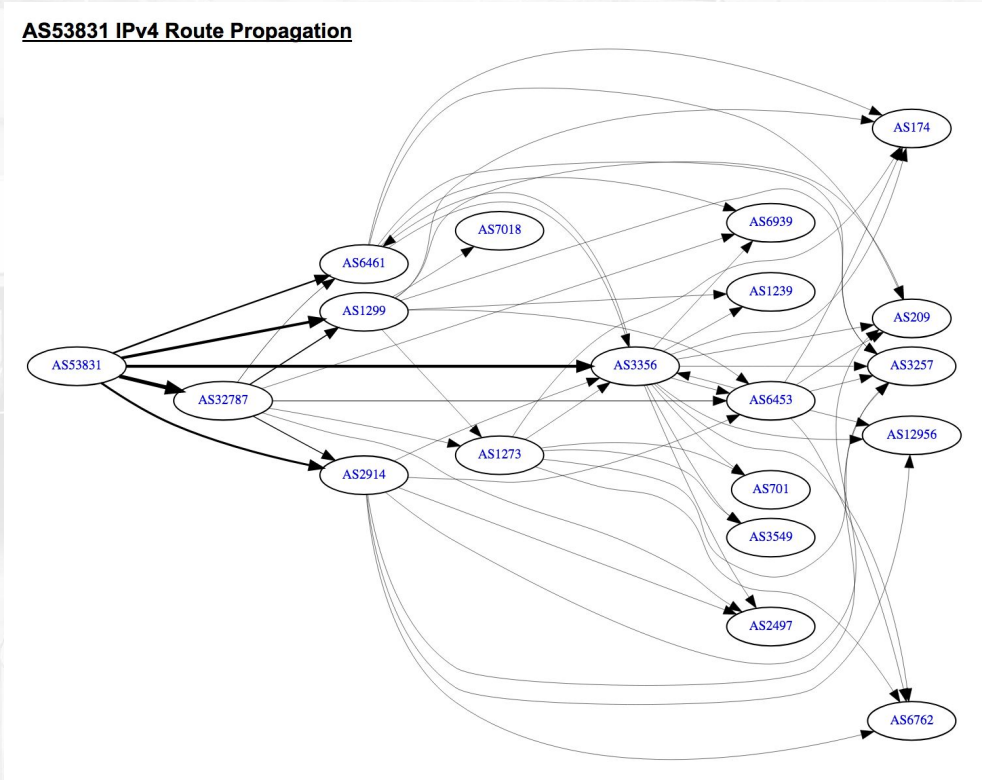
With BGP route server

What problem are we solving?

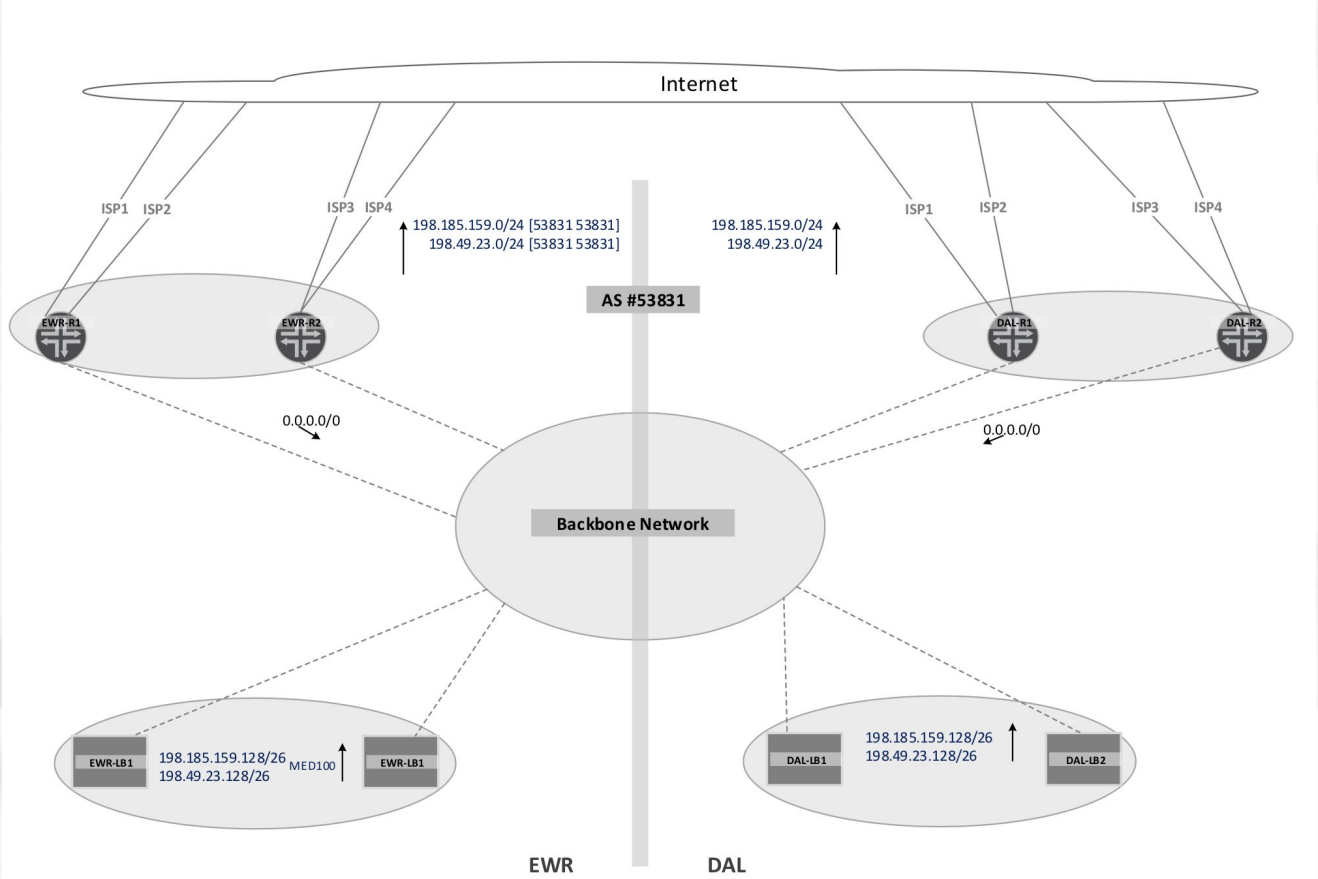
- *Datacenter failover*
- *Draining transit providers links*
- *Routing on/off DDOS protection service*

Squarespace Edge/Internet graph

All of the listed tasks require to start announcing or withdrawing a prefix/subnet via BGP to internal or external Peers.

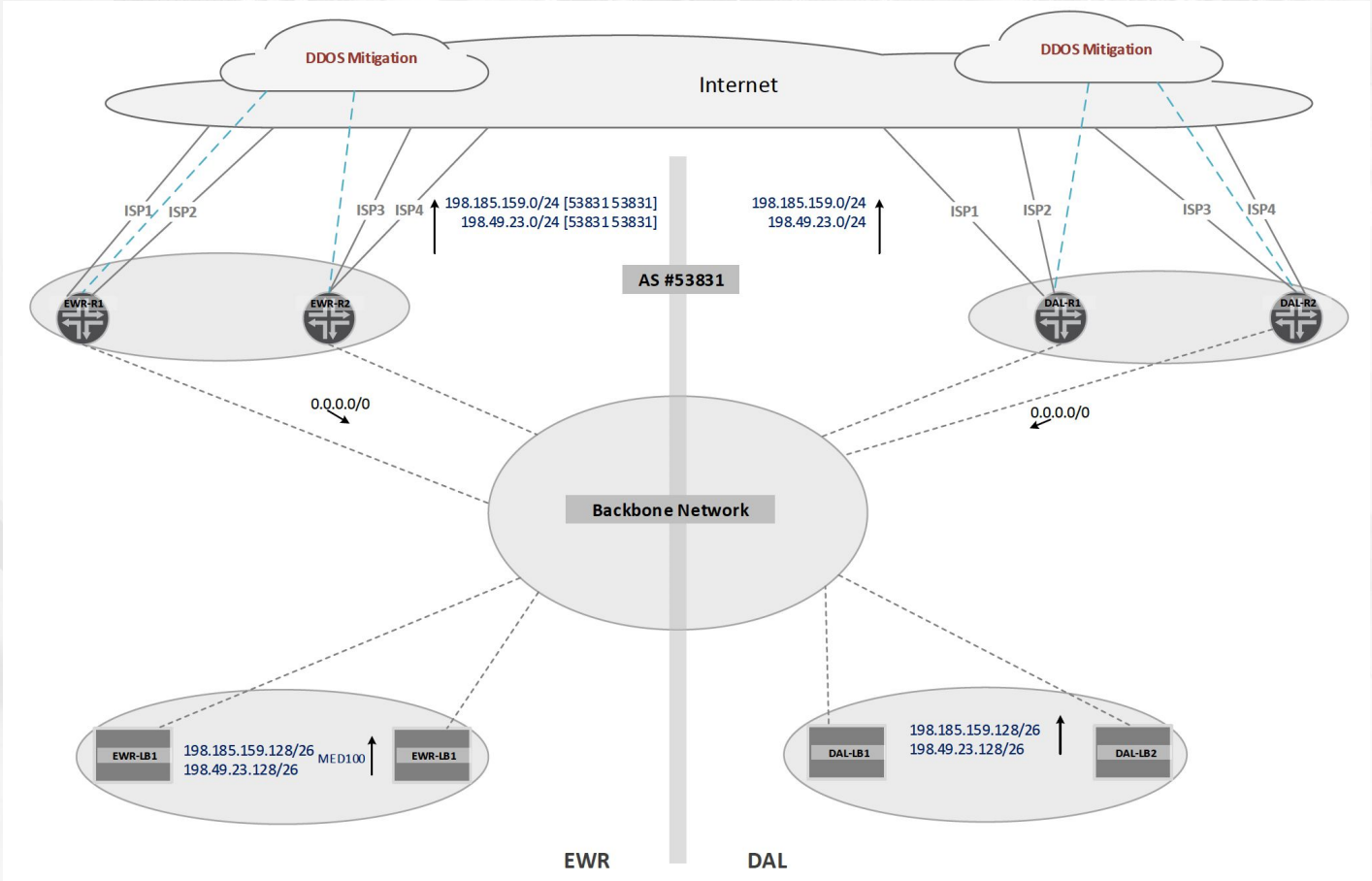


Squarespace Edge Network



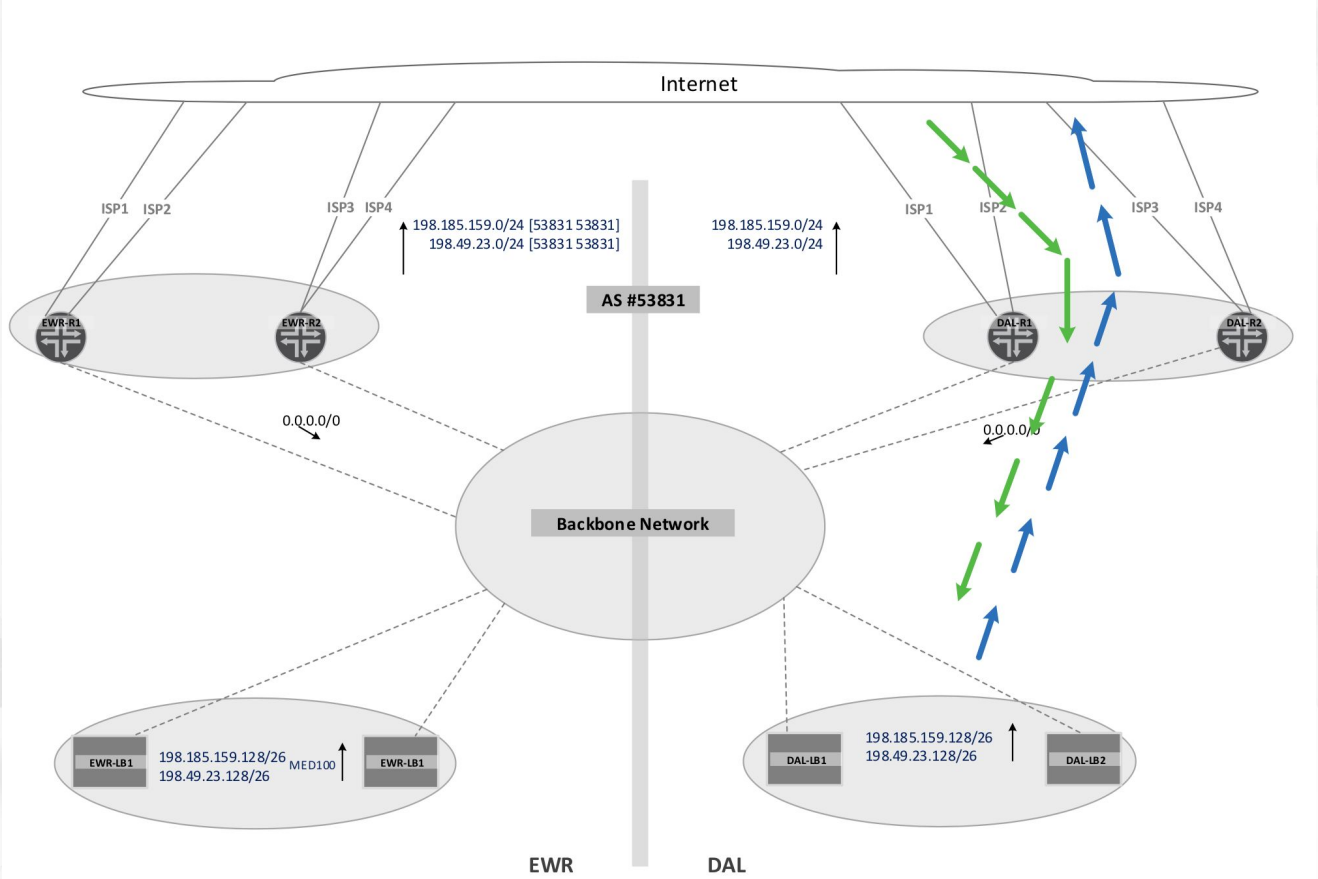
Active / Standby

Squarespace Edge Network



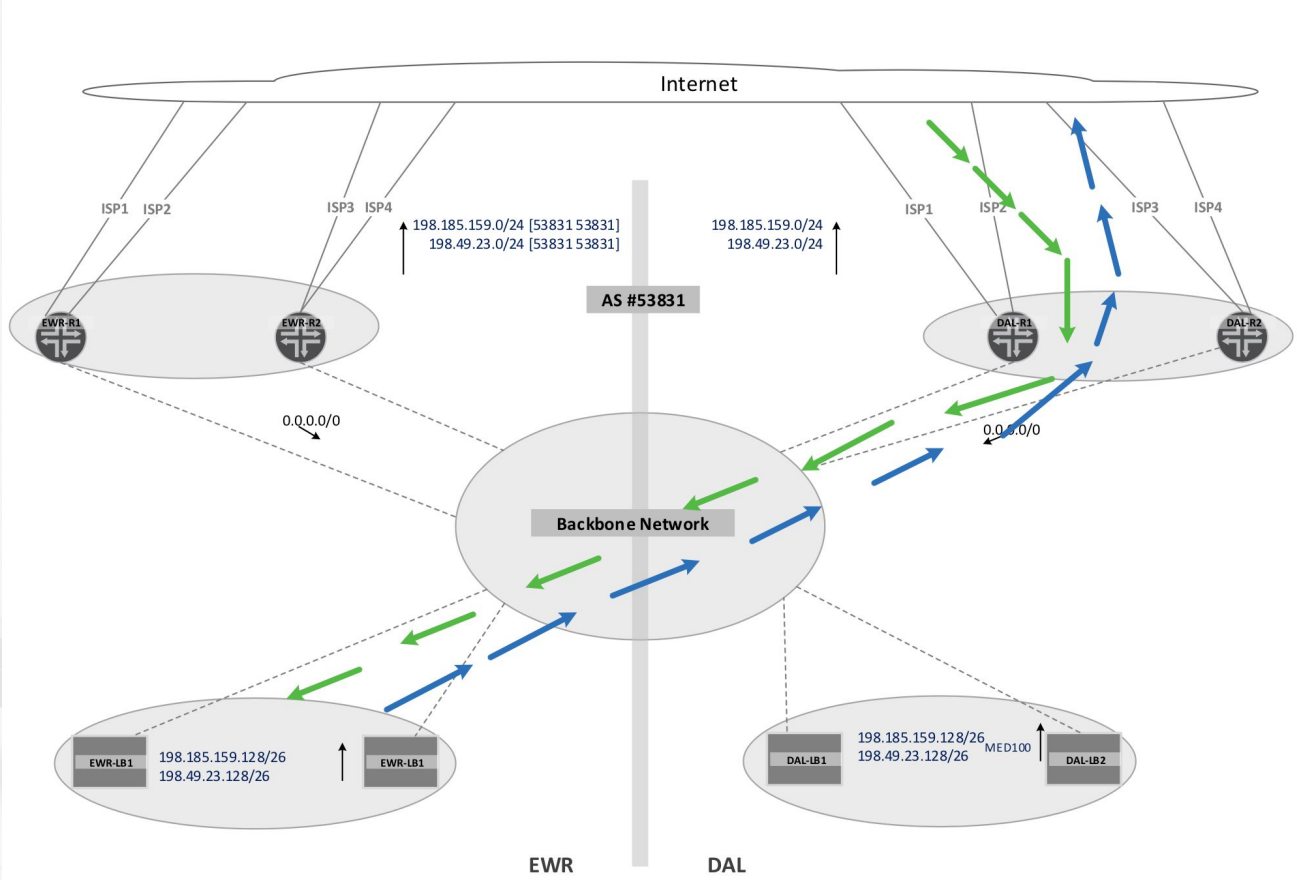
Inbound through DDOS Mitigation Service

Squarespace Edge Network



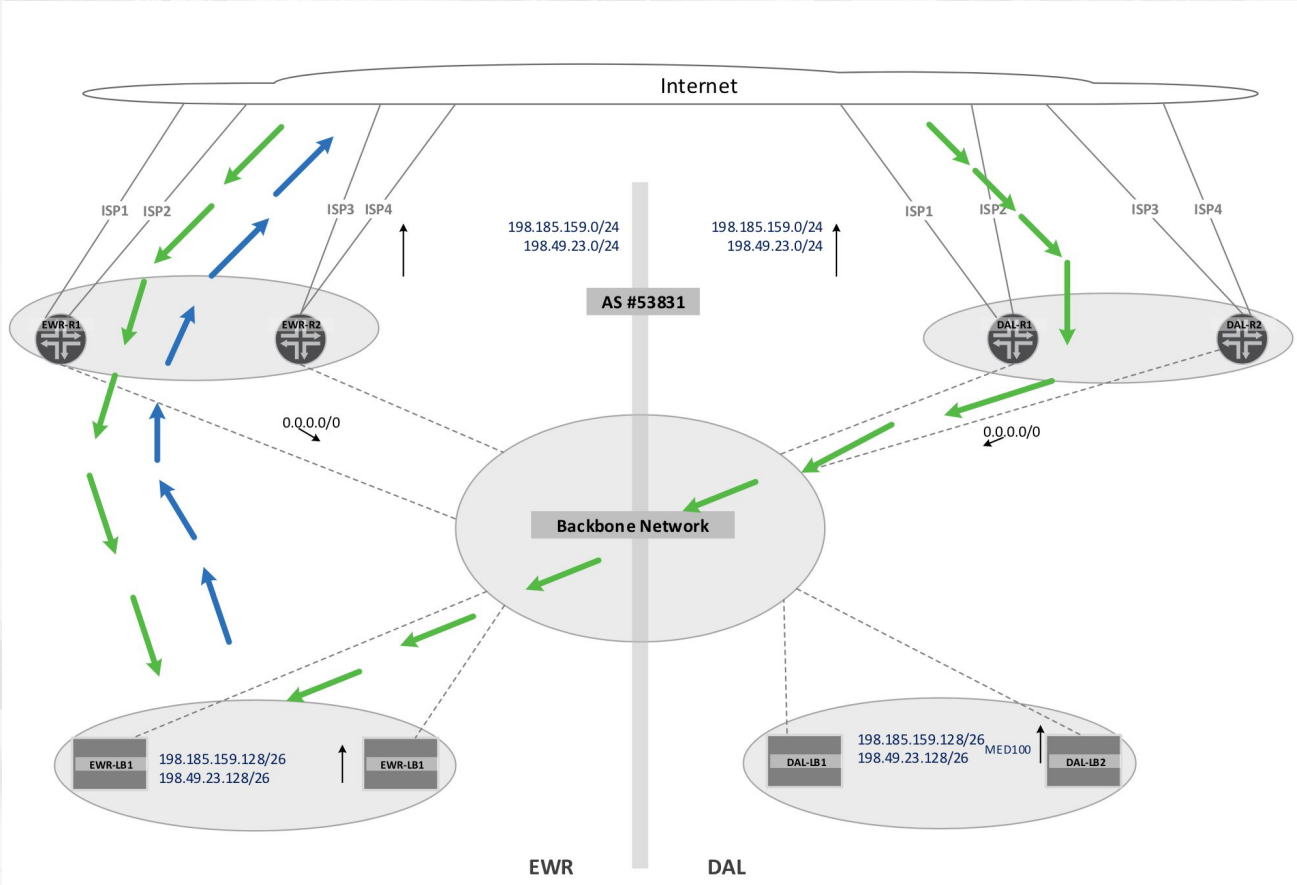
DAL - Active

Squarespace Edge Network



Ingesting in DAL and sending across backbone

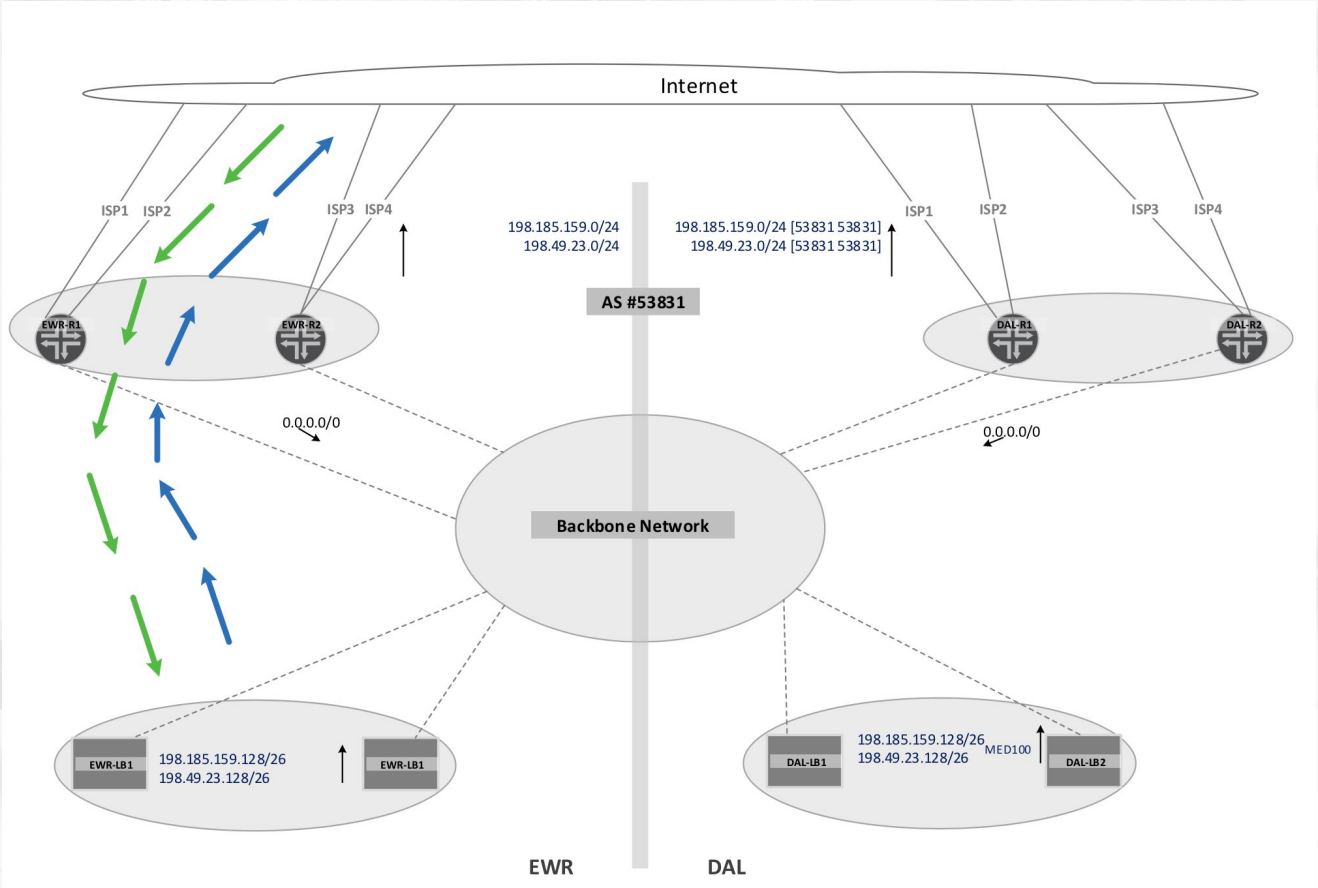
Squarespace Edge Network



Active / Active

50% of traffic across backbone

Squarespace Edge Network



EWR Active

Failover completed

Failover steps manually

or through ansible

```
bash-3.2$ ls DA_VARS/ | grep DA_EDGE
DA_EDGE_PLX_50_50.yml
DA_EDGE_PLX_50_50_AS_PATH_ADD.yml
DA_EDGE_PLX_ASHBURN_ADD_PATH.yml
DA_EDGE_PLX_ASHBURN_OFF.yml
DA_EDGE_PLX_AS_PATH_ADD.yml
DA_EDGE_PLX_DA_EDGE_R1_ADD_PATH.yml
DA_EDGE_PLX_DA_EDGE_R2_ADD_PATH.yml
DA_EDGE_PLX_ISP_OFF.yml
DA_EDGE_PLX_OFF.yml
```

```
  PROLEXIC:
    EXPORT_POLICY:
      - sqs-net-198.185.159.0_24
      - sqs-net-198.49.23.0_24
```

```
bash-3.2$ cat DA_VARS/DA_EDGE_PLX_50_50.yml
EDGE_ROUTERS:
  da-edge-r1:
    TRANSIT:
      Telia:
        export:
          - sqs-net-198.185.159.0_24
          - sqs-net-198.49.23.0_24
          -
          -
          -
          -
          -
          - rejectAll
      ZAYO:
        export:
          - sqs-net-198.185.159.0_24
          - sqs-net-198.49.23.0_24
          -
          -
          -
          -
          -
          - rejectAll
```

```
bash-3.2$ cat DA_VARS/DA_EDGE_PLX_ON.yml
EDGE_ROUTERS:
  da-edge-r1:
    TRANSIT:
      Telia:
        export:
          - sqs-net-198.185.159.0_24_TELIA_LOWEST_LP
          - sqs-net-198.49.23.0_24_TELIA_LOWEST_LP
#
          - rejectAll
      ZAYO:
        export:
          - sqs-net-198.49.23.0_24_ZAYO_LOWEST_LP
          - sqs-net-65.39.205.0_24_ZAYO_LOWEST_LP
#
          - rejectAll
```

- Config change on all 4 edge routers to change policies responsible for route-advertisements to ISPs and update AS-Path prepending.

- Config change on all 4 Load-Balancers to update MED accordingly.

Failover steps manually

or through ansible

```
bash-3.2$ ls DA_VARS/ | grep DA_EDGE
DA_EDGE_PLX_50_50.yml
DA_EDGE_PLX_50_50_AS_PATH_ADD.yml
DA_EDGE_PLX_ASHBURN_ADD_PATH.yml
DA_EDGE_PLX_ASHBURN_OFF.yml
DA_EDGE_PLX_AS_PATH_ADD.yml
DA_EDGE_PLX_DA_EDGE_R1_ADD_PATH.yml
DA_EDGE_PLX_DA_EDGE_R2_ADD_PATH.yml
DA_EDGE_PLX_ISP_OFF.yml
DA_EDGE_PLX_OFF.yml
```

```
  PROLEXIC:
    EXPORT_POLICY:
      - sqs-net-198.185.159.0_24
      - sqs-net-198.49.23.0_24
```

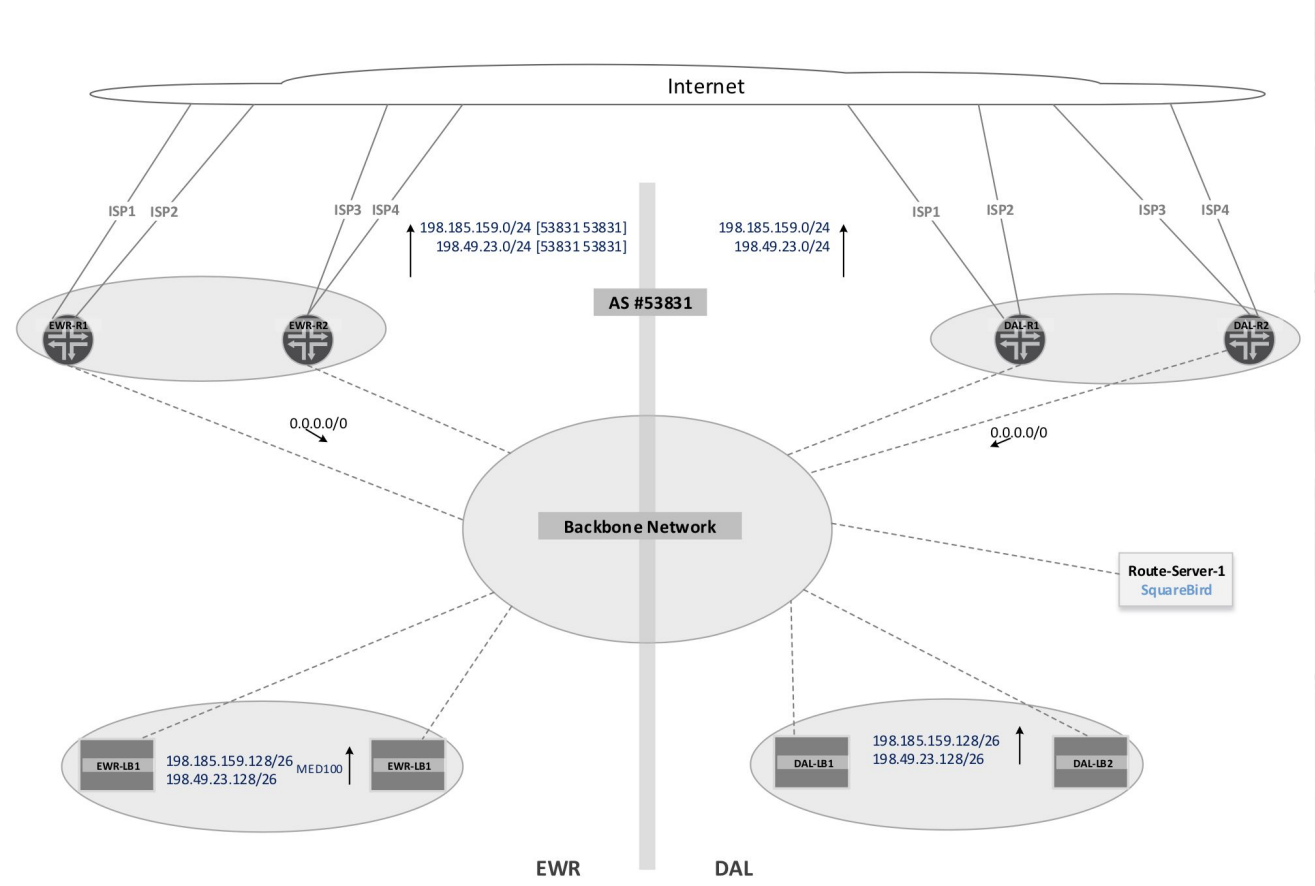
```
bash-3.2$ cat DA_VARS/DA_EDGE_PLX_50_50.yml
EDGE_ROUTERS:
  da-edge-r1:
    TRANSIT:
      Telia:
        export:
          - sqs-net-198.185.159.0_24
          - sqs-net-198.49.23.0_24
          - [redacted]
          - rejectAll
      ZAYO:
        export:
          - sqs-net-198.185.159.0_24
          - sqs-net-198.49.23.0_24
          - [redacted]
          - rejectAll
```

```
bash-3.2$ cat DA_VARS/DA_EDGE_PLX_ON.yml
EDGE_ROUTERS:
  da-edge-r1:
    TRANSIT:
      Telia:
        export:
          - [redacted]
          - sqs-net-198.185.159.0_24_TELIA_LOWEST_LP
          - sqs-net-198.49.23.0_24_TELIA_LOWEST_LP
#         - [redacted]
          - rejectAll
      ZAYO:
        export:
          - [redacted]
          - sqs-net-198.49.23.0_24_ZAYO_LOWEST_LP
          - sqs-net-65.39.205.0_24_ZAYO_LOWEST_LP
#         - [redacted]
          - rejectAll
```

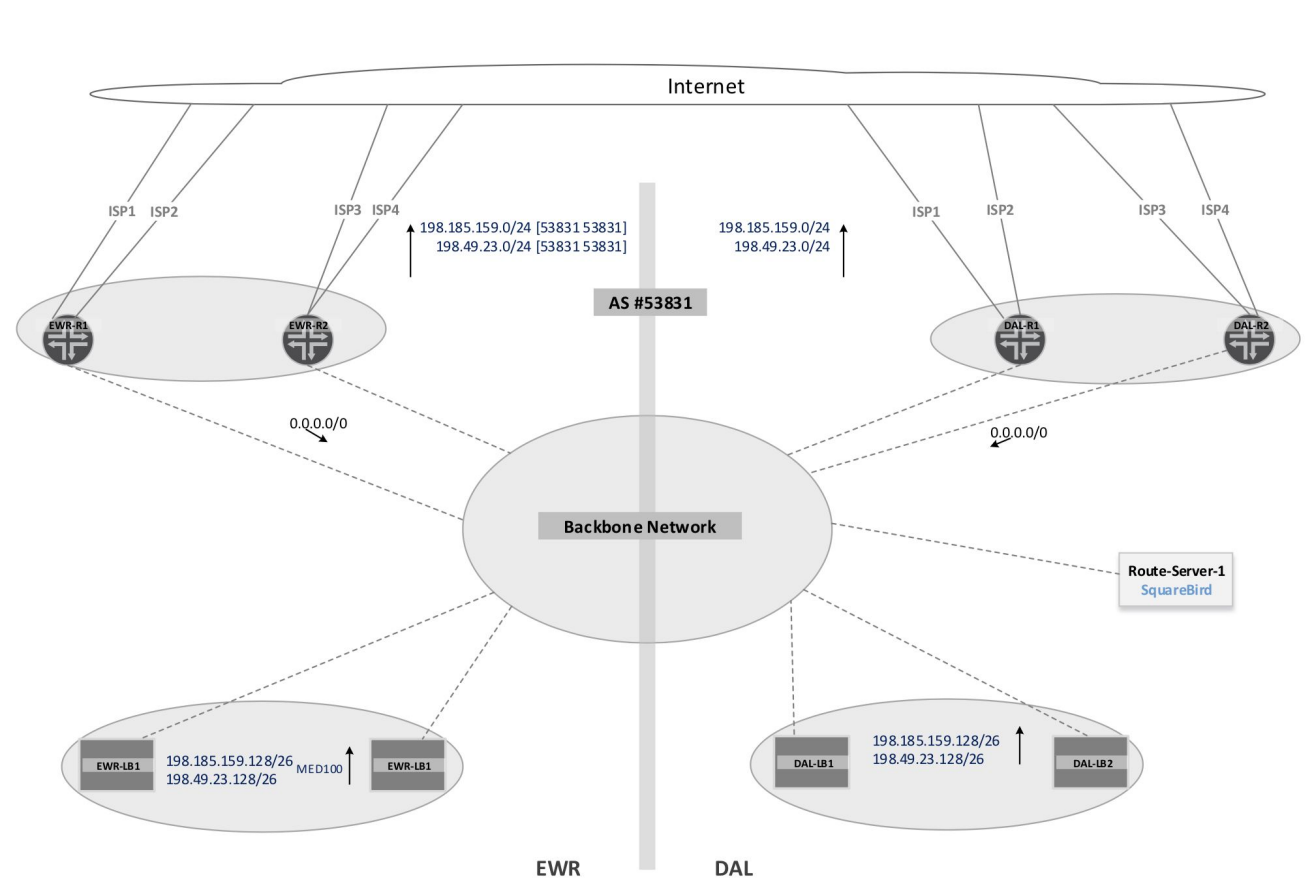
Using ansible to route off of DDOS mitigation:

```
ansible-playbook playbooks/build_edge.yml
--tags 'bgp' -e 'ROUTING=DA_EDGE_PLX_OFF.yml'
ansible-playbook  playbooks/deploy_edge.yml
```

Squarespace Edge Network - Edge Route Server (EdgeRS)



Squarespace Edge Network - Edge Route Server (EdgeRS)



- iBGP with every network node
- Traditional route-reflector for Edge routers
- Originates prefixes that are originated by LBs and Edge Routers
- Supports a set of functions that attach a specific community to a route to trigger a respective action (for example as-path prepending or withdrawal of a route) on a router or load-balancer

Interaction between EdgeRS and Edge Routers is based on BGP community attribute

BGP communities are values attached to a route that is sent to peers.

BGP community has the following format: **<asn>:<number>**.

Asn = 53831, since its our public AS

The second part of the community string (<number>) has the following format: **<[site] [router] [isp] [action]>.**

Interaction between EdgeRS and Edge Routers

Community is the key.

<[site] [router] [isp] [action]>

Configuration on External-RS && Edge router policies are generated by the same ansible role.
For example, when a new transit provider is provisioned, we don't have to manually write hundreds of lines of policies

```
---
### Route Server Variables
SquarespaceASN: 53831
re_community_base: 1000
community_TE: 9482

## Routing actions
TE_Actions:
 Suppress: 6    # Match prefix and Reject
 Prepend: 1     # Match prefix and do AS-PATH prepends
 Default: 9     # Announce prefix without AS-PATH prepends

Transit_ISPs:
 - name: "All"
   id: 9
 - name: "Level3"
   id: 1
 - name: "Telia"
   id: 4
 - name: "PLX"
   id: 7
TE_Sites:
 da: 2
 nj: 3
```

Interaction between EdgeRS and Edge Routers

```
bird> show route export nj_square_edge_r1 all 192.187.26.0/24
192.187.26.0/24    blackhole [static_BGP 2018-04-10] * (200)
        Type: static unicast univ
        BGP.origin: IGP
        BGP.as_path: [redacted]
        BGP.next_hop: [redacted]
        BGP.local_pref: 100
        BGP.community: (53831,9482) (53831,3996) (53831,3096)
bird>
```

<[site] [router] [isp] [action]>

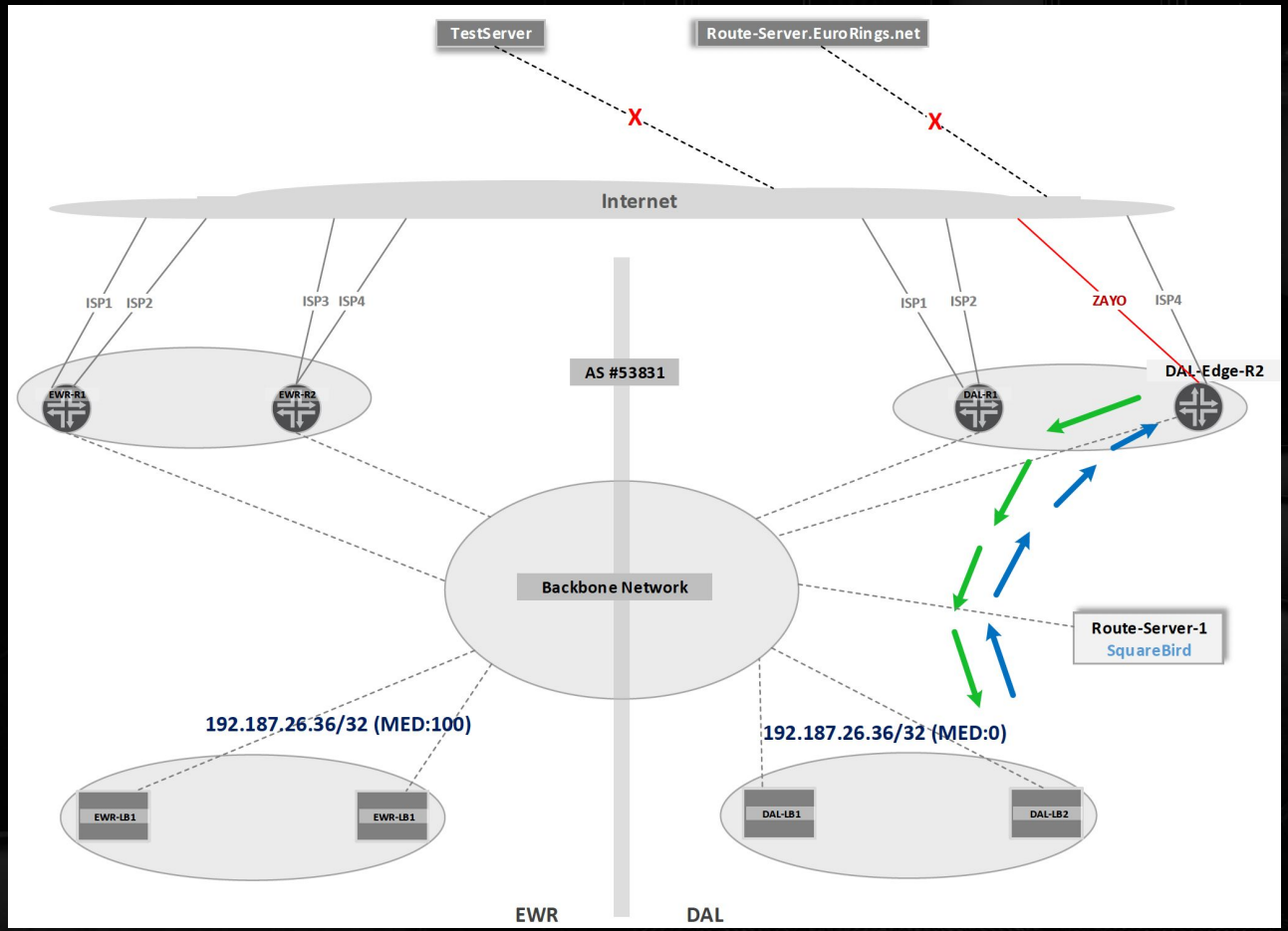
3996: 3=NJ datacenter; 9=all routers; 9=all ISPs; 6=Suppress

```
rromanyak@nj-edge-r1> show configuration | display set | match 3996
set policy-options community SquareBird-nj-All-isp_All-Suppress members 53831:3996

rromanyak@nj-edge-r1> show configuration policy-options policy-statement SquareTE-Export-NTT term Suppress-192.187.26.0/24
from {
    community [ SquareBird-nj-r1-isp_NTT-Suppress SquareBird-nj-r1-isp_All-Suppress SquareBird-nj-All-isp_All-Suppress ];
    route-filter 192.187.26.0/24 exact;
    condition 192.187.26.0/24-SquareTE;
}
then reject;

rromanyak@nj-edge-r1> show configuration policy-options policy-statement SquareTE-Export-NTT | display set | count
Count: 168 lines

rromanyak@nj-edge-r1> show configuration policy-options condition 192.187.26.0/24-SquareTE
if-route-exists {
    192.187.26.0/24;
    table SquareBird.inet.0;
}

rromanyak@nj-edge-r1>
```

Recorded Demo

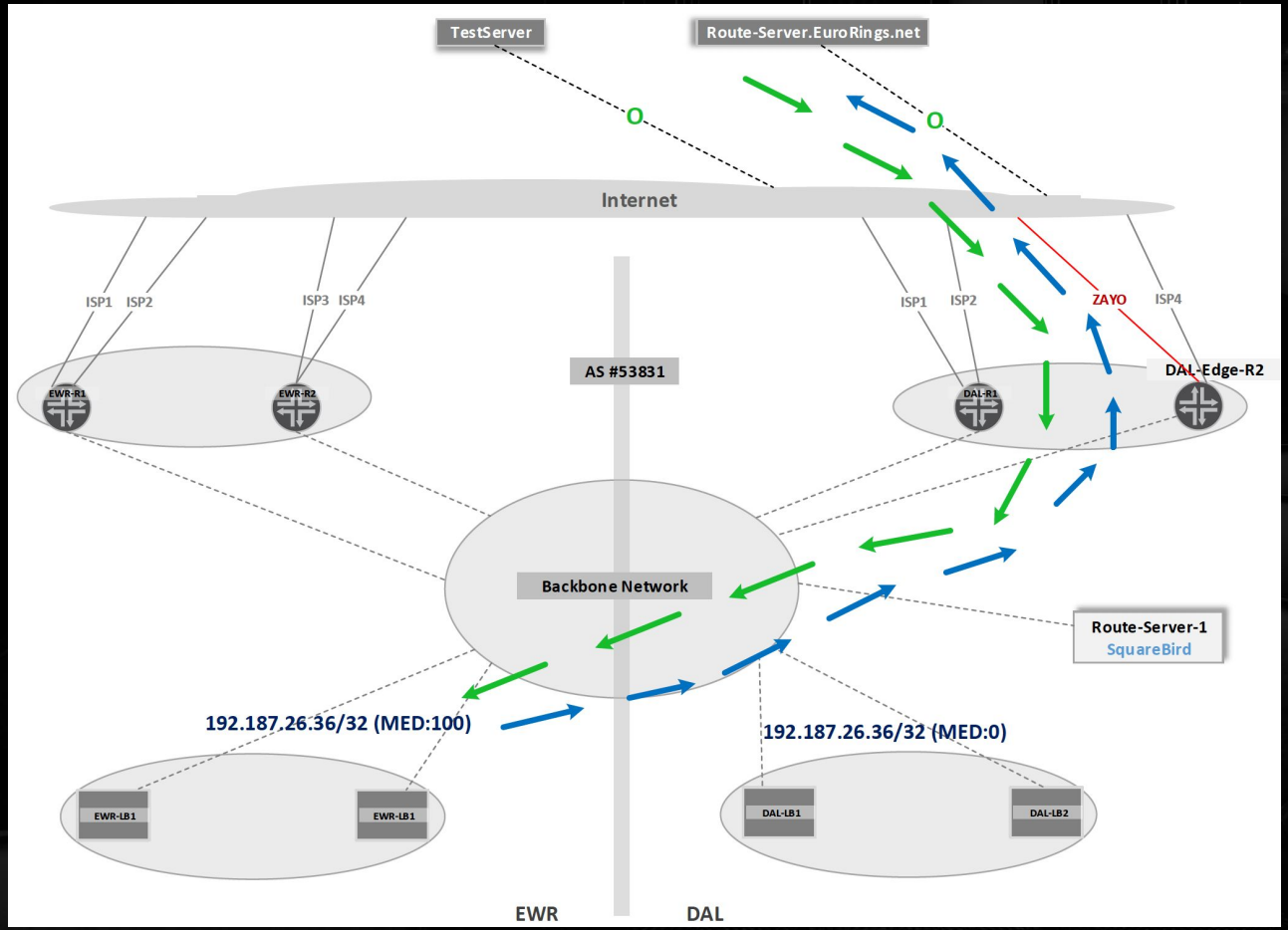
- *Enabling and suppressing BGP advertisements on a transit link*
- *Moving an endpoint on load-balancer to the remote datacenter*

Recorded Demo - Before



- *Latency between da-edge-r2 and VIP = 0.5ms*
- *No connection between Internet and VIP*

Recorded Demo - After

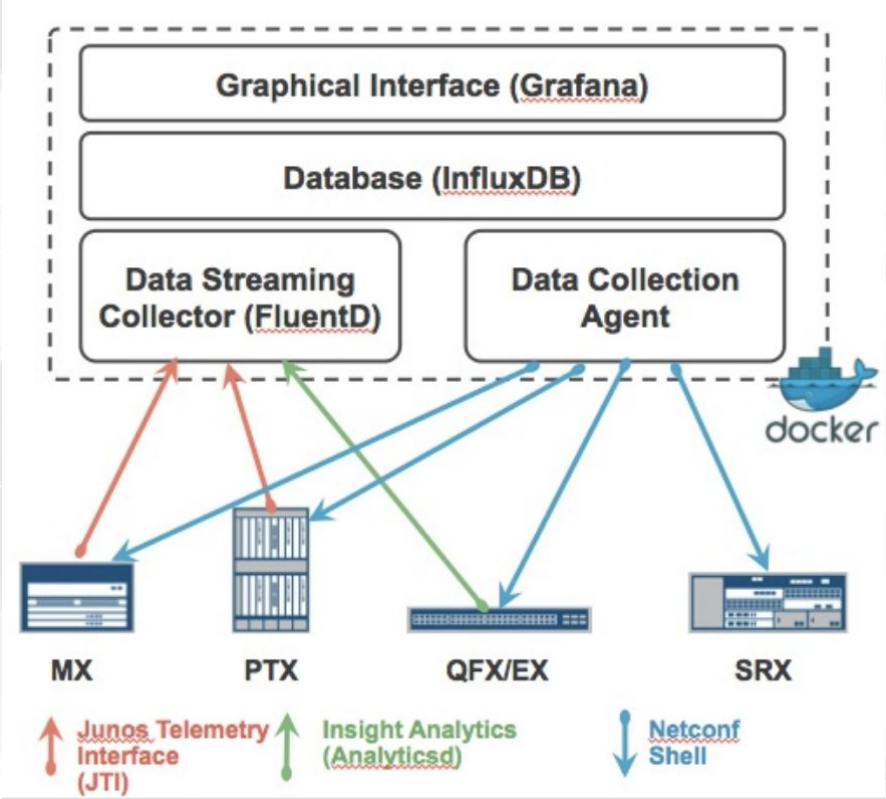


- *Latency between da-edge-r2 and VIP = 38 ms (over DCIs)*
- *VIP accessible from internet*

Monitoring based on OpenNTI

OpenNTI:

- Streaming telemetry
- Grafana
- InfluxDB
- Fluentd



Monitoring - Routing-off transit link

```
36      -   if BGP_to_SquareBird(nj_site, R_All, ISP_All, TE_Default, 0, 0, EWR_LocalWan, "active", "Routed_Off_PLX") then {
        36  + if BGP_to_SquareBird(nj_site, R1, ISP_NTT, TE_Prepend, 0, 0, EWR_LocalWan, "active", "Routed_Off_PLX") then {
37      37      accept;
38      38  }
```

Generating Grafana dashboards

Dashboard for all leaf switches uplinks in two Pods:

```
---
group_dashboards:
  - title: "DAL - Network - Leafs (PodA)"
    server: "telemetry-net-ewr.squarespace.net"
    type: "group"
    dash_file: "opennti-da_leafs_PODA-dashboard.json"
    annotations:
      state: enable
    graph_options:
      fill: 3
      color: ["#f4ce42", "#4286f4"]
    device_groups:
      - name: Prod (DA) - Network - Leafs (PodA)
        device_group: da_leafs_podA
        stats: bps
        direction: duplex
        interfaces: [et-0/0/27, et-1/0/27]

  - title: "DAL - Network - Leafs (PodB)"
    server: "telemetry-net-ewr.squarespace.net"
    type: "group"
    dash_file: "opennti-da_leafs_PODB-dashboard.json"
    annotations:
      state: enable
    graph_options:
      fill: 3
      color: ["#f4ce42", "#4286f4"]
    device_groups:
      - name: Prod (DA) - Network - Leafs (PodB)
        device_group: da_leafs_podB
        stats: bps
        direction: duplex
        interfaces: [et-0/0/27, et-1/0/27]
```

Generating Grafana dashboards

Grafana

TELEMETRY

- *Storing Network configuration in a source control*
- *Source Control is the single source of truth*
- *Minimize direct interaction via CLI*
- *Infrastructure and Network as a code*

SQUARESPACE

Questions?

Thank you!

Roman Romanyak

rromanyak@squarespace.com / @rromanyak

squarespace.com/careers