Network Automation using modern tech

Egor Krivosheev 2degrees

Telemetry

Discovery

Config. Mgmt.

Inventory

Streaming Platform

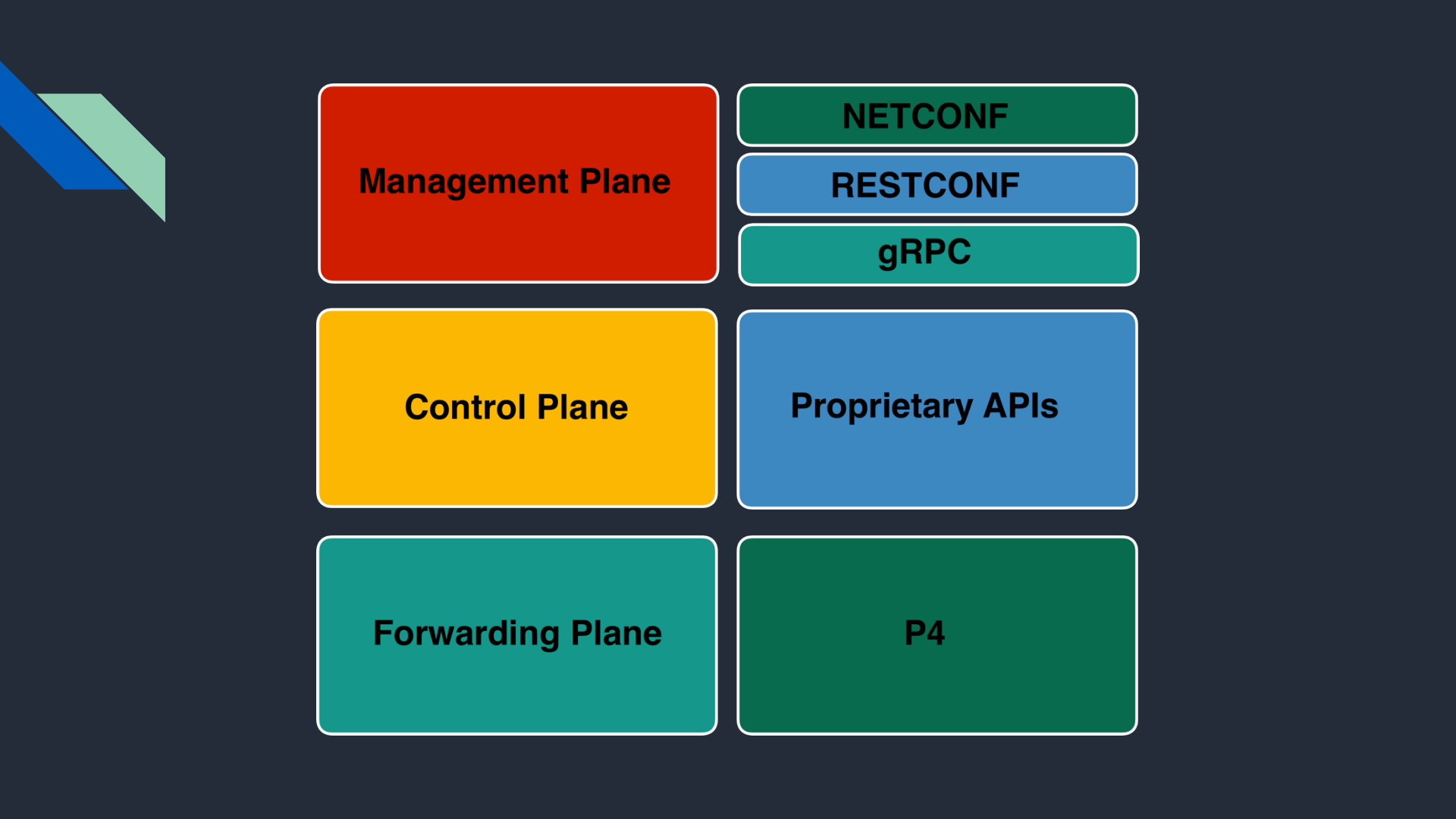
TS DB

Monitoring

Analytics and Visualisation

Key parts of network automation today

- Streaming Telemetry
- APIs
- SNMP and screen scraping are still around


Management Plane
NETCONF
RESTCONF
gRPC
Control Plane
Proprietary APIs
Forwarding Plane
P4

NETCONF

- RFC6241
- XML encoding
- Most implementations use SSH as transport
- There are libraries for many languages Python, Go, C, Java, PHP

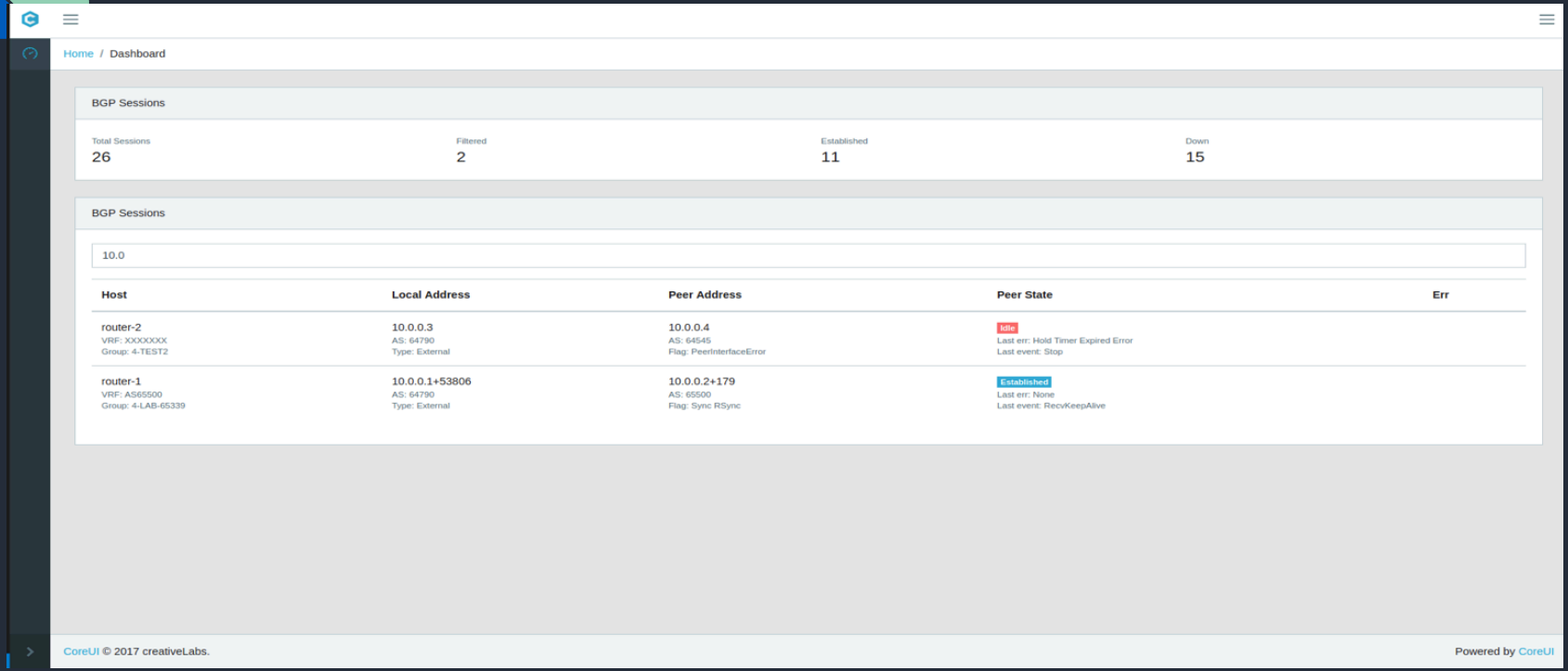
Vendor support

- Nokia
- Cisco
- Juniper
- Brocade
- Arista

Can be used for:

- Configuration management
- Operational data collection

BGP session viewer



- github.com/vasya4k/gojun

RESTCONF

- RFC8040
- XML or JSON as encoding
- Configuration management
- Operational data
- HTTP POST, GET, PUT, DEL to manage configuration or collect operational data

RESTCONF VS NETCONF

In RESTCONF you need one request to change an interface config another one to change BGP and so on. No verification, instead each request succeeds or fails

In NETCONF you lock the config make all the changes, verify, then commit and unlock

gRPC

- gRPC - is an open source remote procedure call system
- Uses HTTP/2 for transport
- Protocol Buffers as the interface description language
- Bi-directional streaming
- Client libraries for more than ten languages

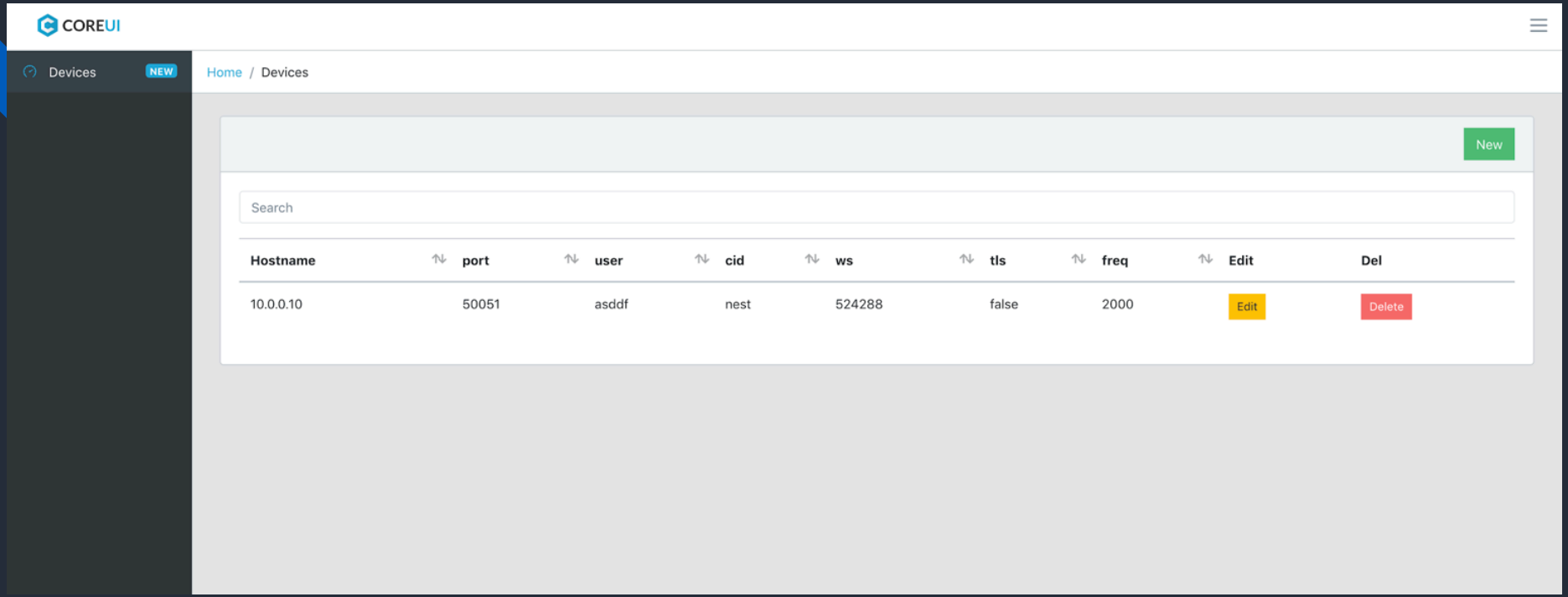
```
syntax = "proto3";

package telemetry;

service OpenConfigTelemetry {
    rpc telemetrySubscribe(SubscriptionRequest)                 returns (stream OpenConfigData) {}
    rpc cancelTelemetrySubscription(CancelSubscriptionRequest)  returns (CancelSubscriptionReply) {}
    rpc getTelemetrySubscriptions(GetSubscriptionsRequest)      returns (GetSubscriptionsReply) {}
    rpc getTelemetryOperationalState(GetOperationalStateRequest) returns (GetOperationalStateReply) {}
    rpc getDataEncodings(DataEncodingRequest)                   returns (DataEncodingReply) {}
}

message OpenConfigData {
    string system_id                                = 1;
    uint32 component_id                             = 2;
    uint32 sub_component_id                         = 3;
    string path                                     = 4;
    uint64 sequence_number                          = 5;
    uint64 timestamp                                = 6;
    repeated KeyValue kv                            = 7;
    repeated Delete delete                          = 8;
    repeated Eom eom                                = 9;
    bool sync_response                              = 10;
}
```

gRPC - network interfaces

- Two interfaces: Open Config Telemetry and gNMI
- First only supports data streaming
- Seconds can be used for both configuration management and streaming telemetry
- Vendor support - Cisco, Juniper, Nokia, Arista

gRPC Example

- A streaming telemetry collector with a web interface and a simple API
- Default collection frequency is 2 sec
- Packaged in three Docker containers Grafana, Collector and Influx DB
- Found here github.com/vasya4k/nest



https://github.com/vasya4k/nest

Control Plane - Proprietary APIs

- Juniper JET and Cisco IOS-XR Service Layer both gRPC based
- Provide direct access to things like firewall policies or CoS without a need for a configuration change
- Much faster than dealing with management plane
- Cisco: BFD, Interfaces, MPLS, Routing
- Juniper: Interfaces, MPLS, Routing, BGP, CoS and Firewall

Use cases

- Adjusting CoS policy on a microwave link
- Adding a lot of static routes in BGP for testing
- Quickly create an IP interface for automatic testing

Forwarding plane API

- With P4 programing language
- P4 Runtime supported by at least Arista, Cisco, Juniper
- Available on many whitebox switches running Barefoot Tofino ASICs
- Capabilities depend on hardware

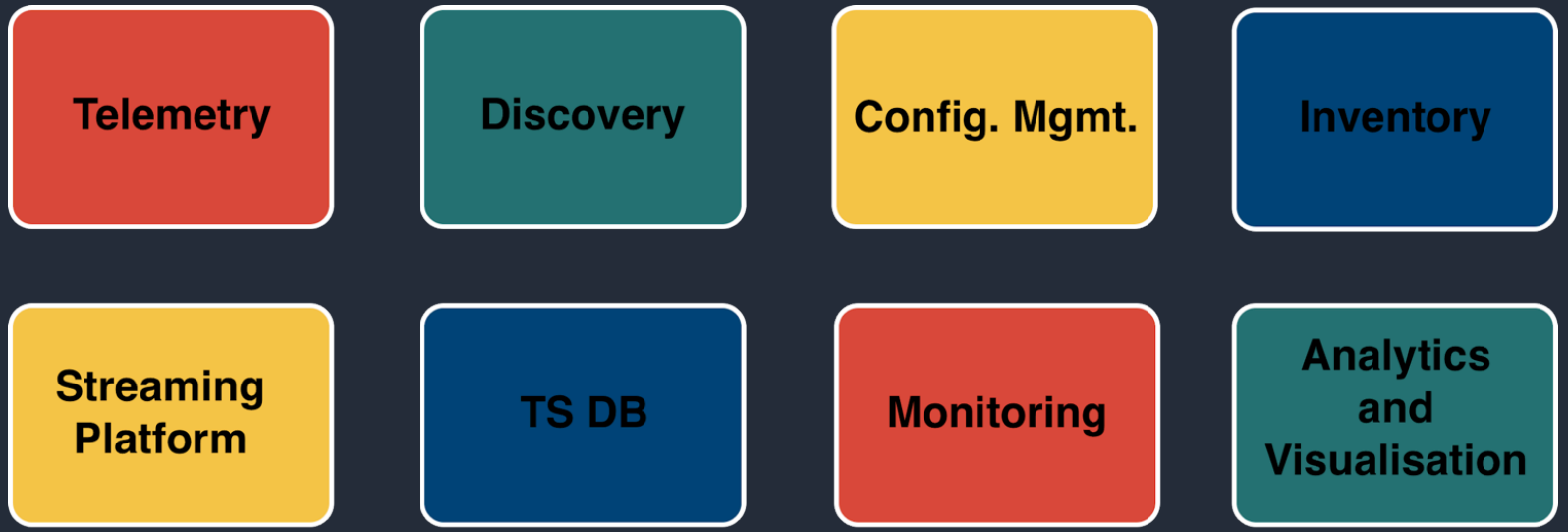
What is P4?

- A domain specific language
- Open-source
- Protocol independent, Target independent
- Match-action pipelines, packet forwarding can be split into series of table lookups and corresponding header manipulations

Can be used to:

- Implement a new encapsulation
- Update forwarding plane code of existing routers

Build your own automation system



Inventory and Discovery

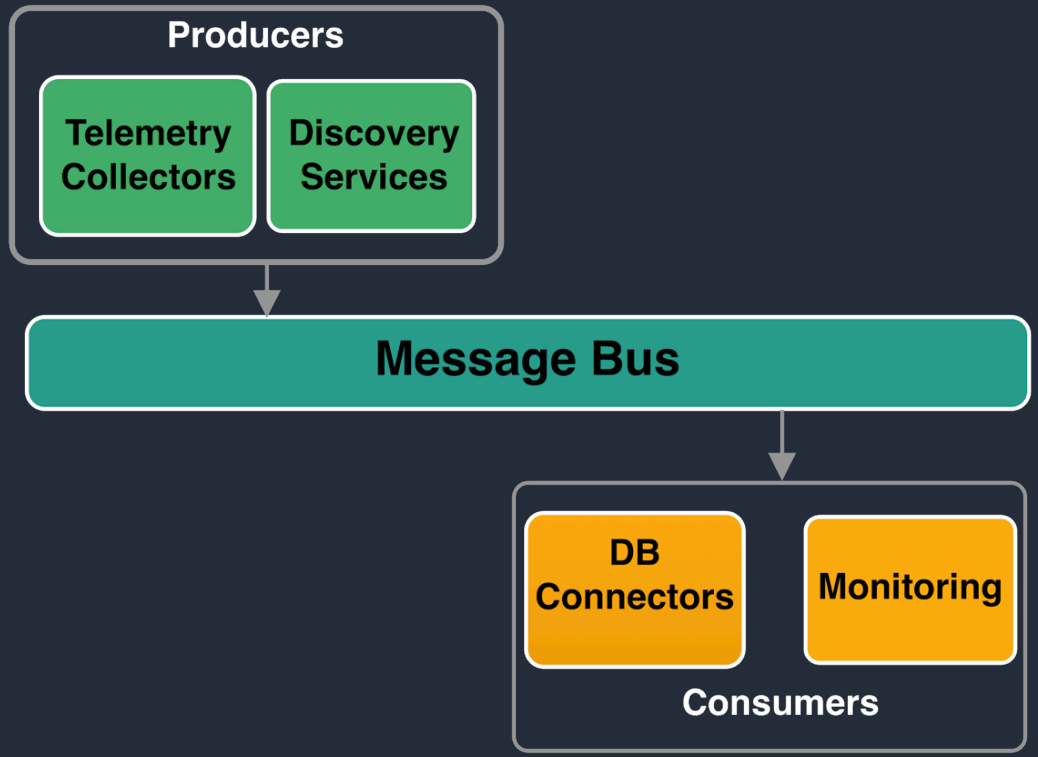
Inventory can be in the form of REST or gRPC APIs with a web interface. The API is intended to be used by Configuration Management, Monitoring and many other systems as a source of truth.

Discovery service is a program which detects changes in hardware or software and sends updates to a streaming platform.

If inventory is updated manually, it also sends updated data into a streaming platform.

Data Streaming Platform

- Also known as message bus
- Most popular choices are Apache Kafka, RabbitMQ and NSQ
- My personal favourite is Kafka as it provides built-in redundancy, persistent storage and is horizontally scalable
- It also allows many systems to subscribe for the same data


Producers
Telemetry Collectors
Discovery Services
Message Bus
DB Connectors
Monitoring
Consumers

Configuration management

- Few choices available Ansible, Saltstack and of cause you can always build your own
- There is also at least one attempt to create a python automation framework called Nornir
- I think a combination of ready to use tools and purposely written systems works the best

Ansible

- Agentless
- Easy to install and use
- Inventory variables are stored in txt files
- YAML syntax for playbooks
- Written in Python
- Extensible by writing y modules in any language
- AWX is an open source version of Ansible Tower

Saltstack

- Needs an agent called proxy minion to manage network devices
- Uses NAPALM under the hood
- Has an API
- Built-in inventory and data cache
- Provides event-driven infrastructure capabilities

Telemetry

- Protocol Buffers as encoding
- Collectors get data using gRPC, NEtCONF, SNMP or even screen scraping and publish stream of records into one or more Kafka topics
- Consumers subscribe to topics and process the streams of records
- Few open-source collectors exist fluentd, Open-nti, Telegraf

Having metrics in Kafka allows to

- Check state to verify configuration changes
- Get events and react to them
- Provide a feedback loop

Time Series DB

- Elasticsearch to store structured data and logs
- InfluxDB for “flat” times series data
- Both have high adoption rate well documented and open-source
- With Influx DB you do not get horizontal scaling in free version

Analytics and Visualisation

- Grafana to present graphs, dashboards
- KIbana to provide GUI and build reports for logs and events
- There is a very interesting project called Yandex.ClickHouse for data analytics

Chatbot as a universal interface

- Using a chatbot makes it easy to abstract multiple actions into one and share information between teams
- For example you can have a bot which goes and fetches an image of a graph from Grafana which can be discussed by a team within same chat
- You can build a bot which can shut down a faulty link or make changes to you BGP policy

People are the most important part

- Automation is hard if you do not have a buy-in from your network engineers
- It takes a lot of time to change mentality
- It took a long time to change mentality of my own team in my previous organisation even though my bosses were all in with automation

Links

- Juniper control plane API www.juniper.net/documentation/en_US/jet17.1/topics/concept/jet-service-apis-overview.html
- Cisco control plane API http://xrdocs.io/cisco-service-layer/apidocs/modules.html
- gRPC https://grpc.io/docs/quickstart/go.html
- My github account with examples https://github.com/vasya4k
- Golang NETCONG library https://github.com/Juniper/go-netconf
- Python NETCONG https://github.com/ncclient/ncclient