Using Ansible as a catalyst for digital transformation
The journey from network engineering to network development

Bart Dworak
Software Engineering Manager, Services Platform Engineering, Network, Microsoft
Our mission is to empower every person and every organization on the planet to achieve more.

Microsoft’s mission statement
Overview/Agenda

What is network engineering?
What is network development?
How do I get from engineering to development?
What is network engineering?
The Network Engineering Cycle

- Architect
- Maintain
- Deploy
Traditionally network hardware is shipped to a site partially pre-configured. Once the device is plugged in, a network engineer is informed and performs the tasks to bring the device online and into a fully configured state.
Maintenance

Network Devices send alerts

Network devices need upgrades

Version 1.11 – Released Dec 30, 2017
Version 1.12 – Released Jan 6, 2018

1.11 is vulnerable to CVE-2017-5754

Network Devices need to be replaced
What is network development?

The primary difference between network engineering and network development is the mindset, method, and tools used to solve network problems, deploy new solutions, and maintain a stable performant system.
Values of network development

Instead of configuring by hand, we automate.

Instead of manually responding to alerts and events, we write automation that diagnoses problems and attempts self-healing.

Instead of being reactive, we are proactive and use data to make decisions and to predict our future.

Instead of configuring devices, we enforce network-wide desired state configuration.
Using a network developer mindset, we pivot the final steps of a network deployment from manual to automated deployment using automation tools and event based workflows.
Network Development Goals

- Automation
- Event response & self-healing
- Fact Collection
- Desired State Configuration
Why network development?

Networks are growing

Moving to the cloud

IOT

Configuration Complexity
Tools of network development

- Git
- GitHub
- Azure
- Ansible
- Azure ML
- Kubernetes
- Azure Cosmos DB
- Python
- Helm
- Azure Serverless
- Azure Service Bus
- And many more!
The road towards network development

- Automation
- Event Response
- Fact Collection
- Desired State
The first step towards network development is establishing automation and automation practices to enable future success.
Automation

Automate simple and repetitive tasks

Check everything into a version control system

Adopt a workflow to manage changes to code-base

Build - Test - Release

FEATURE → DEVELOP → MASTER

ANSIBLEFEST
Using git for version control allows developers to collaborate and peer review each others work before it ever makes it into production.
Having a well known and agreed upon workflow to get code into production allows for quality and control along the way.
Zero-to-Hero or how we introduced automation

Start with simple playbooks and introduce git

Add complexity with re-usable roles and testing

Encourage learning via PR and community discussion

The road to network development takes time and new concepts should be introduced gradually to build a solid foundation for your team.
Event Response

Once you have an automation framework in place it is time to automate event response and implement self-healing where applicable.
Event based automation allows us to action on events triggered by the network while collecting more telemetry on the events.
Using event based automation, we can insert our automation platform in the middle of the incident pipeline and trigger automation on incidents in-flight to the incident management system.
Self-Healing Example

Workflow:
- **AP Online? Connected? Powered?**
  - Yes → Attempt Self-Healing
  - No → Engage an Engineer

- **AP Online? Connected? Powered?**
  - Yes → Resolve Incident
  - No → Engage an Engineer
Fact Collection

In order to automate more, you will need some data and telemetry. It is time to collect facts from the network to help make data-driven decisions on the fly.
Fact Collection

Single Source of Truth

Collected & entered data

Use data to configure and make decisions

Opens the door for machine learning and insights
Creating a single source of truth allows for mastery of data in a single location. Hot data is made available to machines while historical data takes the cold path into the insights platforms.
Collecting telemetry to gain insights

Collecting a huge amount of telemetry from the network opens the door for running advanced analytics scenarios such as anomaly detection and machine learning.
Using our event pipeline, we can collect historical incident data, pass it through the insights pipeline to gain additional insights. In the above example, a device can be having intermittent issues, a problem that can be caught in the incident pipeline using historical data.
The true goal is to establish a desired state configuration mechanism that enforces that state on the network.
In order to build a desired state configuration, developers create network code and check it into a version control system. That code is used to create and build the models that will be used to apply configurations to the network.
Code created by developers is re-used, similar to libraries today. The higher you move in the network code stack the more obfuscated the code becomes giving the end user more power to create and enforce network configurations and policies.
Using data to create configuration

<table>
<thead>
<tr>
<th>Service/Configuration Definitions</th>
<th>Meta-Data from Fact-Store</th>
<th>Device configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>&quot;Snmp&quot;: [</td>
<td>&quot;Snmp&quot;: [</td>
<td>&quot;Snmp&quot;: [</td>
</tr>
<tr>
<td>{</td>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>&quot;password&quot;: &quot;string&quot;,</td>
<td>&quot;password&quot;: &quot;ansible-fest&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;read&quot;: &quot;boolean&quot;,</td>
<td>&quot;read&quot;: &quot;true&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;write&quot;: &quot;boolean&quot;</td>
<td>&quot;write&quot;: &quot;false&quot;</td>
<td></td>
</tr>
<tr>
<td>},</td>
<td>},</td>
<td>{</td>
</tr>
<tr>
<td>&quot;interfaces&quot;: [</td>
<td>&quot;password&quot;: &quot;ansible-fest-write&quot;,</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>&quot;read&quot;: &quot;true&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;name&quot;: &quot;&lt;interfaces.name&gt;&quot;,</td>
<td>&quot;write&quot;: &quot;true&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;Description&quot;: &quot;&lt;neighbor.name&gt;-&lt;self.name&gt;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

Using our fact collection and network code repositories, we are able to combine code with meta-data to drive device configurations.
As we move to enforce full circle desired state, network code is combined with meta-data in the fact store to create a desired state configuration. That configuration is pushed to the device and then continuously streamed back into the platform for validation.
The full desired state configuration circle requires network code to be converted to desired state configuration which is continuously streamed back from network devices via streaming telemetry. This allows for full enforcement of desired state.
Thank you

#ANSIBLEFEST

YouTube.com/user/RedHatVideos
LinkedIn.com/company/Red-Hat
Azure.com/RedHat

FaceBook.com/AnsibleAutomation
Twitter.com/Ansible

Azure.com/RedHat