JOURNEY TO
Network Reliability Engineering (NRE)
and DevNetOps

James Kelly
NETWORK AUTOMATION ≠ AUTOMATED NETWORKS
CONSUMPTION IS THE CONTEMPORARY CHALLENGE

Mistakes of the past
• Forcing engineers’ rebirth as a “developer”
• The “pornography of tech” has left behind transformation of people & processes

Ironically...
humans are the heroes
in the journey to automated NetOps

BRIGHT SPOT:

82% of data center NetOps are still manual

NRE
Network reliability engineer(ing)
SPEED, AGILITY, EFFICIENCY & OTHER SUCCESSES ARE INCIDENTAL FOR THE NRE THAT HOLDS RELIABILITY PREREQUISITE TO OTHER USEFUL ECONOMIES
FOR THE NRE OPERATIONS SIMPLICITY IS CREATED THROUGH ENGINEERING
INSPIRATION

DevOps is Coined 2009

DevOps Handbook 2015

SRE is published by Google 2016
DevNetOps
OR
NRE?
WHAT IS (AN) NRE?

**Def^n An NRE**
The professional that implements network reliability engineering

**Def^n NREing**
*Engineering* an automated network to deliver measurable *reliability* (SLO/SLA of MTTF, MTTR, etc.) under measurable conditions (scale, rate of change, performance, etc.)

**Def^n DevNetOps**
Like NRE, *engineering* automated network, but more explicitly says:
- Take a developer (software engineering) approach
- The application of the approach is to NetOps
- Focus on shorter cycles and lead time in code-to-prod pipeline
- Our work begins in pre-production and follows CI/CD/CR
WHY N- RELIABILITY -E

1. **Reliability is at the base of the hierarchy of needs**
   It’s prerequisite to security, velocity, agility or efficiency.

2. **Reliability must be ensured before acceleration**
   “It’s not how fast you drive, it’s how you drive fast”

3. **Reliability forces us to automate and simplify**
   Encompassing NetOps goals: resiliency, security, metrics…

**Higher-order DevOps principles…**

1. Eliminate toil and technical debt with automation
3. Allow for failure; Iterate and evolve with Agile; then triangulate...
4. Continuous improvement: turn local lessons into global ones
5. Continuous learning (kaizen)
WHY NR - ENGINEER

1. **It’s not just about network automation or technology:**
   “Network automation does not an automated network make.”

2. **NRE focus sidesteps DevOps vs. DevNetOps confusion**
   There are clear NetOps projects outside of software teams, but some confusion on terms remains. NRE is more straightforward.

3. **Engineers are builders with structure and rigor**
   Engineering picks up where vendors leave off

4. **More to it than in-production NOC dashboards**
   More creative, more satisfying, more money

5. **By building, testing, stressing, staging what you build...**
   you prepare for better operations and better outcomes
5-STEP JOURNEY
AUTOMATED NETOPS 5-STEP

1. Manual Ops
   - NetOps at the device or system UI
   - Engineers are more technicians than technologists

2. Automated Workflows
   - Automate the design of ops: workflows
   - Focus on frequent troubleshooting or read-only tasks before config management

3. Automation, tests and networks as code
   - Connect actions to triggers and think test-driven
   - Rethink troubleshooting as testing
   - Everything (even configs) is code to be tested

4. Continuous processes, continuous pipeline
   - CI-CD-CR
   - A “DevNetOps” pipeline for accuracy and agility
   - Fast feedback/fail, small changes, safe/canary deployments
   - Automate analytics response for regulation

5. Engineering Outcomes
   - Management by higher-order metrics
   - NRE outcomes with service-level objectives, indicators and agreements (SLO/SLI/SLA)
   - Use error budgets, toil budgets
   - Manage, don’t maximize, reliability
   - Manage dependencies, separation of concerns

People: Network Reliability Engineers (NRE)
Process: “DevNetOps” And NRE’ing
Technology: Autonomous and automated NetOps
# Technology Landscape

**STEP 1**

- Device CLI
- Product GUI
- Workflow pseudocode
- Mistral

**STEP 2**

- `{ _ () ; }` Code
- Markup/Modeling
- Ansible

**STEP 3**

- `{ }` Markup
- Chef
- Puppet
- Salt

**STEP 4**

- gRPC
- Jenkins
- GitLab

**STEP 5**

- Bamboo
- Jenkins Pipelines
- Spinnaker

---

*It's not what you use, it's how you use it.*

---

*Sample only, not exhaustive*
# A DEVNETOPS PIPELINE

<table>
<thead>
<tr>
<th>Network as Code</th>
<th>TOOLING</th>
<th>PROCESSES</th>
<th>PEOPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Repos of config, secrets, artifacts + gitOps</td>
<td>branching, reviewing, pairing, Agile</td>
<td>Code skills (not necessarily programming)</td>
</tr>
<tr>
<td>Pipeline Orchestration</td>
<td>Pipeline CI/CD tools, test frameworks</td>
<td>TDD, measurement judgements</td>
<td>Build and debug skills, pipeline pros</td>
</tr>
<tr>
<td>Micro &amp; Immutable Architecture</td>
<td>Baking deliveries for ZTP, vendor refactors</td>
<td>Small-step commits/deployment</td>
<td>Hands off CLI/TTY</td>
</tr>
<tr>
<td>Orchestrated Upgrades</td>
<td>ZTD, virtualization, labs, traffic draining</td>
<td>Staging and simulation, canary analysis</td>
<td>In-hours maintenance, maybe roll forward</td>
</tr>
<tr>
<td>Resiliency Design and Drills</td>
<td>Traffic generation, DoS, chaos monkey</td>
<td>Chaos windows, document limits</td>
<td>Force failure for understanding</td>
</tr>
<tr>
<td>Continuous Measurement</td>
<td>Big data analytics, ML, ITops integration</td>
<td>Incident playbooks, capacity planning</td>
<td>Management by stats, metrics, efficiency</td>
</tr>
<tr>
<td>Continuous Response</td>
<td>Auto-remediation, FaaS, predictive stats</td>
<td>Supervise self-driving</td>
<td>Drink tea, meditate</td>
</tr>
<tr>
<td>Continuous Improvement</td>
<td>Upgrades, features, fixes, changes</td>
<td>Record local lesson into global knowledge</td>
<td>Active open-mindedness, post-mortems</td>
</tr>
</tbody>
</table>

© 2018 Juniper Networks, Inc. All rights reserved
THANK YOU