Why Ansible vs. Salt vs. Nornir is the Wrong Question to be Asking

A practical guide to tool choice

@itdependsnet

Ken Celenza

ken@networktocode.com

Introduction

Ken Celenza

- Managing Director at Network to Code
- Traditional network engineer by day, coder by night
- Converted full time ***network automator*** in 2016
- 20 years in the industry, primarily supporting enterprises

Current Landscape (per social media)

Current Landscape (per social media)

- Engineers getting into automation, want to know, “what’s the best tool” or “Ansible vs Nornir”
  - The community responds with their personal preference
  - There is **no context** on use cases

Current Landscape (per social media)

- Engineers getting into automation, want to know, “what’s the best tool” or “Ansible vs Nornir”
  - The community responds with their personal preference
  - There is **no context** on use cases

- There is a strong correlation in responses based on the responders
  - Technical ability
  - Industry

Current Landscape (per social media)

- Engineers getting into automation, want to know, “what’s the best tool” or “Ansible vs Nornir”
  - The community responds with their personal preference
  - There is **no context** on use cases
- There is a strong correlation in responses based on the responders
  - Technical ability
  - Industry
- Blogs comparing tools primarily speak about
  - Engineering challenges (speed)
  - Why a tool is easier ***for me***

Current Landscape (per social media)

- Engineers getting into automation, want to know, “what’s the best tool” or “Ansible vs Nornir”
  - The community responds with their personal preference
  - There is **no context** on use cases
- There is a strong correlation in responses based on the responders
  - Technical ability
  - Industry
- Blogs comparing tools primarily speak about
  - Engineering challenges (speed)
  - Why a tool is easier ***for me***
- Well… what is the right question to ask?
  - ***What requirements does this solution fulfill?***

Current Landscape (per social media)

- Engineers getting into automation, want to know, “what’s the best tool” or “Ansible vs Nornir”
  - The community responds with their personal preference
  - There is **no context** on use cases
- There is a strong correlation in responses based on the responders
  - Technical ability
  - Industry
- Blogs comparing tools primarily speak about
  - Engineering challenges (speed)
  - Why a tool is easier ***for me***
- Well… what is the right question to ask?
  - ***What requirements does this solution fulfill?***
  - Better yet: Here are my requirements, how does this solution address them?
  - ** **Disclaimer**… ok, ok, this is still a simplistic question to really get the answers

Technical Evaluation

Ansible

Pros
- Commercial support
- Large adoption and training material
- Simple to get started

Cons
- Complex workflows are difficult
- Speed to run is slow
- Memory & CPU Intensive

Salt

Pros
- Native event bus
- Native API built into core of the product
- Commercial support

Cons
- More difficult to get started
- Smaller community base
- Less (network) training

Nornir

Pros
- Supports complex workflows
- Uses industry standard Python for all capabilities
- Performant

Cons
- Lack of API
- Lack of commercial support
- Less training available

Use Cases and Requirements

Use Cases and Requirements

- Engineers tend to describe requirements, not in underlying use cases

Use Cases and Requirements

- Engineers tend to describe requirements, not in underlying use cases
- A functional requirement without a use case is not a valid requirement

Use Cases and Requirements

- Engineers tend to describe requirements, not in underlying use cases
- A functional requirement without a use case is not a valid requirement
- Requirement: “We need sub minute failover of automation platform”
  - Is this valid? …. it depends
    - Gaming company has a requirement to update adjust BGP configurations based on actual traffic flow continuously 
    - Large financial may not be able to commit change, change windows are no less than one-hour 

Use Cases and Requirements

- Engineers tend to describe requirements, not in underlying use cases
- A functional requirement without a use case is not a valid requirement
- Requirement: “We need sub minute failover of automation platform”
  - Is this valid? …. it depends
    - Gaming company has a requirement to update adjust BGP configurations based on actual traffic flow continuously ✓
    - Large financial may not be able to commit change, change windows are no less than one-hour ✗
- Use cases help identify requirements that are not needed
  - Use Case: “We need a workflow to remove configuration”
  - Actual Solution: Deploy configuration declaratively

Speed Requirements

Speed Requirements

Daily Jobs

Examples
- Backup configurations
- Configuration auditing

Cadence
- Daily

Time to delivery requirement

Speed Requirements

Daily Jobs

Examples
- Backup configurations
- Configuration auditing

Cadence
- Daily

Time to delivery requirement

Bulk Global Changes

Examples
- Global NTP, DNS, etc changes
- ACL updates

Cadence
- 1-4 times a year

Time to delivery requirement

Speed Requirements

Daily Jobs

Examples
- Backup configurations
- Configuration auditing

Cadence
- Daily

Time to delivery requirement

Bulk Global Changes

Examples
- Global NTP, DNS, etc changes
- ACL updates

Cadence
- 1-4 times a year

Time to delivery requirement

Change Management

Examples
- Firewall rule change
- Load Balancer change

Cadence
- Daily

Time to delivery requirement

Actual Drivers

Actual Drivers

When evaluating your primary network Automation platform (Salt, Nornir, Ansible, etc..) what is the number one driver in that choice from the following options? (https://twitter.com/itdependsnet/status/1218932237103661058)

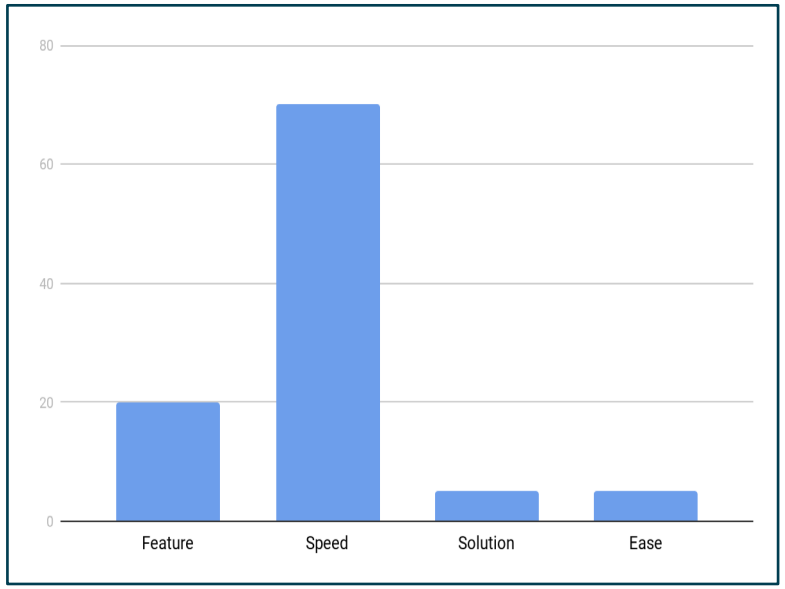
- Feature completeness
- Speed to connect to devices
- Speed to build solutions
- Ease of getting started

Actual Drivers

When evaluating your primary network Automation platform (Salt, Nornir, Ansible, etc..) what is the number one driver in that choice from the following options? (https://twitter.com/itdependsnet/status/1218932237103661058)

- Feature completeness
- Speed to connect to devices
- Speed to build solutions
- Ease of getting started

Hype

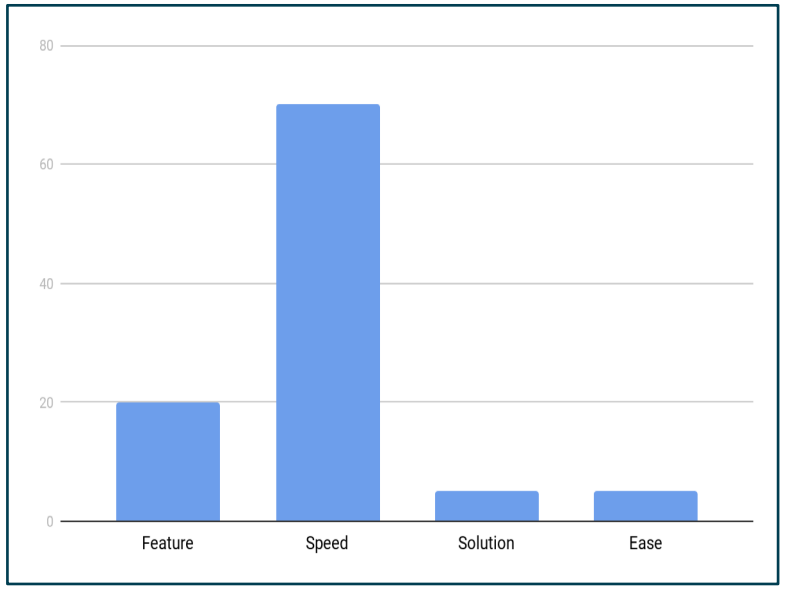


Actual Drivers

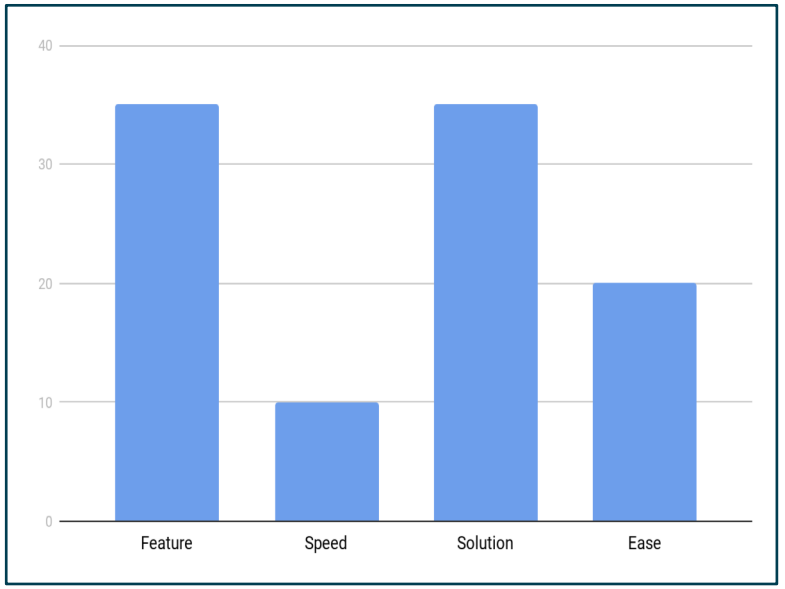
When evaluating your primary network Automation platform (Salt, Nornir, Ansible, etc..) what is the number one driver in that choice from the following options? (https://twitter.com/itdependsnet/status/1218932237103661058)

- Feature completeness
- Speed to connect to devices
- Speed to build solutions
- Ease of getting started

Hype



Expected

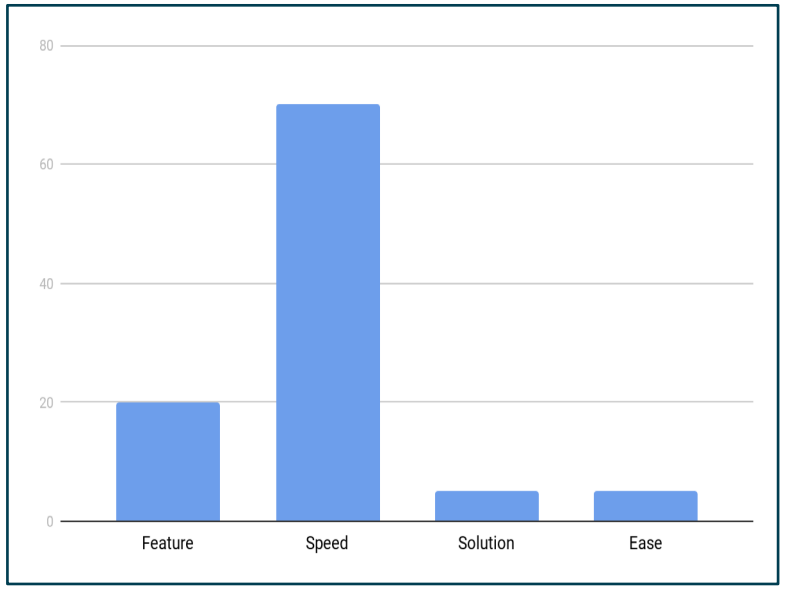


Actual Drivers

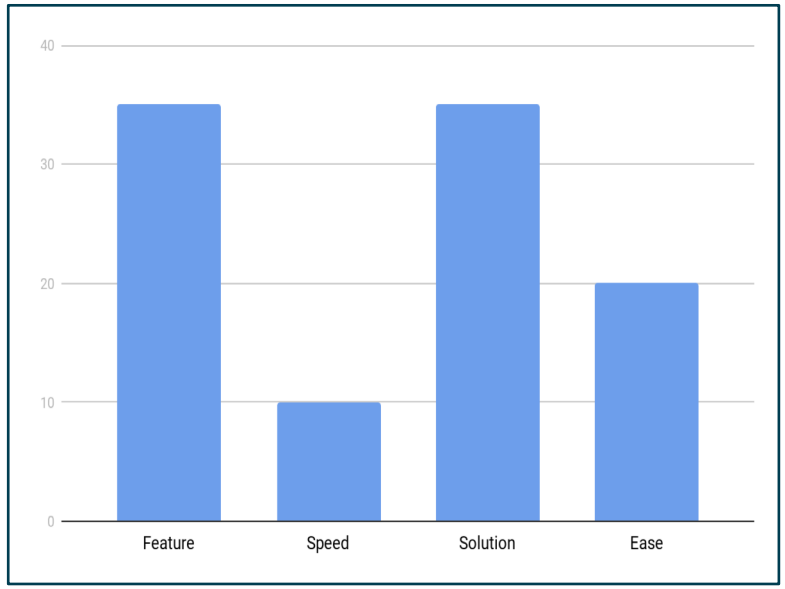
When evaluating your primary network Automation platform (Salt, Nornir, Ansible, etc..) what is the number one driver in that choice from the following options? (https://twitter.com/itdependsnet/status/1218932237103661058)

- Feature completeness
- Speed to connect to devices
- Speed to build solutions
- Ease of getting started

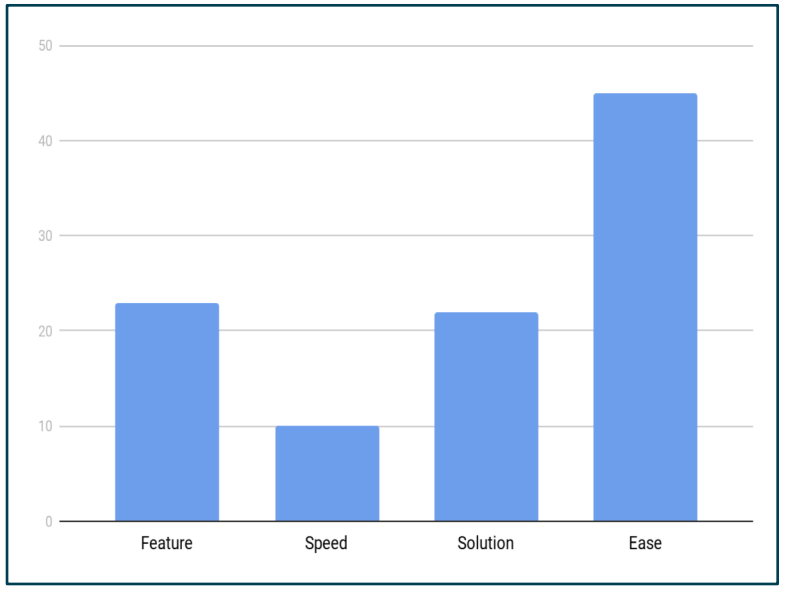
Hype



Expected



Actual



Score Sheet

Score Sheet

Functionality	Priority	<tool #1>	<tool #2>	<tool #3>
Accessibility (for beginner)	5	✓✓✓	✓✓✓	✓✓✓
API	7	✗	✓✓✓	✗
Authentication	7	✗	✓✓✓	✗
Complex Workflow	8	✓✓	✓✓	✓✓✓
Commercial Support	5	✓✓	✗	✓✓
Credential Management	6	✗	✗	✓✓✓
Extensibility	9	✗	✗	✓✓
High Availability	10	✓✓✓	✓✓	✓✓✓
Job Isolation	4	✓	✓✓	✓✓
Logging	7	✓✓✓	✓✓	✓
Network Device Support	8	✓✓✓	✓	✗
Non-network support	9	✓	✓	✗
RBAC	4	✓✓✓	✓✓	✓
Scheduler	10	✓	✓✓✓	✓✓
Scalability	6	✗	✓✓✓	✓✓
Speed	7	✓✓	✗	✓✓✓
Traceability	8	✓✓	✓✓✓	✗
User Interface	4	✓	✓✓✓	✓✓
Workflow	3	✓✓✓	✓	✓

Parting Thoughts

Parting Thoughts

- *The haves and the have nots*
  - Catering to the few high-performers (who often spend their own time) may be detrimental to larger group
  - Solving most use cases simply, can allow engineers to be self-sufficient

Parting Thoughts

- *The haves and the have nots*
  - Catering to the few high-performers (who often spend their own time) may be detrimental to larger group
  - Solving most use cases simply, can allow engineers to be self-sufficient

- Writing the automated workflow and code, is generally the smaller part
  - Building out capabilities for non-functional requirements
  - Maintaining systems and data

Parting Thoughts

- *The haves and the have nots*
  - Catering to the few high-performers (who often spend their own time) may be detrimental to larger group
  - Solving most use cases simply, can allow engineers to be self-sufficient
- Writing the automated workflow and code, is generally the smaller part
  - Building out capabilities for non-functional requirements
  - Maintaining systems and data
- Programming with handcuffs is valuable
  - Even is using more primitive, try to keep with best practices
  - *Just because you can, doesn't mean you should*

Parting Thoughts

- *The haves and the have nots*
  - Catering to the few high-performers (who often spend their own time) may be detrimental to larger group
  - Solving most use cases simply, can allow engineers to be self-sufficient
- Writing the automated workflow and code, is generally the smaller part
  - Building out capabilities for non-functional requirements
  - Maintaining systems and data
- Programming with handcuffs is valuable
  - Even is using more primitive, try to keep with best practices
  - *Just because you can, doesn't mean you should*
- It is costly to introduce new tools, but also valuable to learn from each tool
  - Code that isn't used in the final implementation, isn't "wasted time"

Parting Thoughts

- *The haves and the have nots*
  - Catering to the few high-performers (who often spend their own time) may be detrimental to larger group
  - Solving most use cases simply, can allow engineers to be self-sufficient
- Writing the automated workflow and code, is generally the smaller part
  - Building out capabilities for non-functional requirements
  - Maintaining systems and data
- Programming with handcuffs is valuable
  - Even is using more primitive, try to keep with best practices
  - *Just because you can, doesn't mean you should*
- It is costly to introduce new tools, but also valuable to learn from each tool
  - Code that isn't used in the final implementation, isn't "wasted time"
- We don't need tool wars
  - Not everything has to be competition

Parting Thoughts

- *The haves and the have nots*
  - Catering to the few high-performers (who often spend their own time) may be detrimental to larger group
  - Solving most use cases simply, can allow engineers to be self-sufficient
- Writing the automated workflow and code, is generally the smaller part
  - Building out capabilities for non-functional requirements
  - Maintaining systems and data
- Programming with handcuffs is valuable
  - Even is using more primitive, try to keep with best practices
  - *Just because you can, doesn't mean you should*
- It is costly to introduce new tools, but also valuable to learn from each tool
  - Code that isn't used in the final implementation, isn't "wasted time"
- We don't need tool wars
  - Not everything has to be competition

What requirements does this solution fulfill?

End