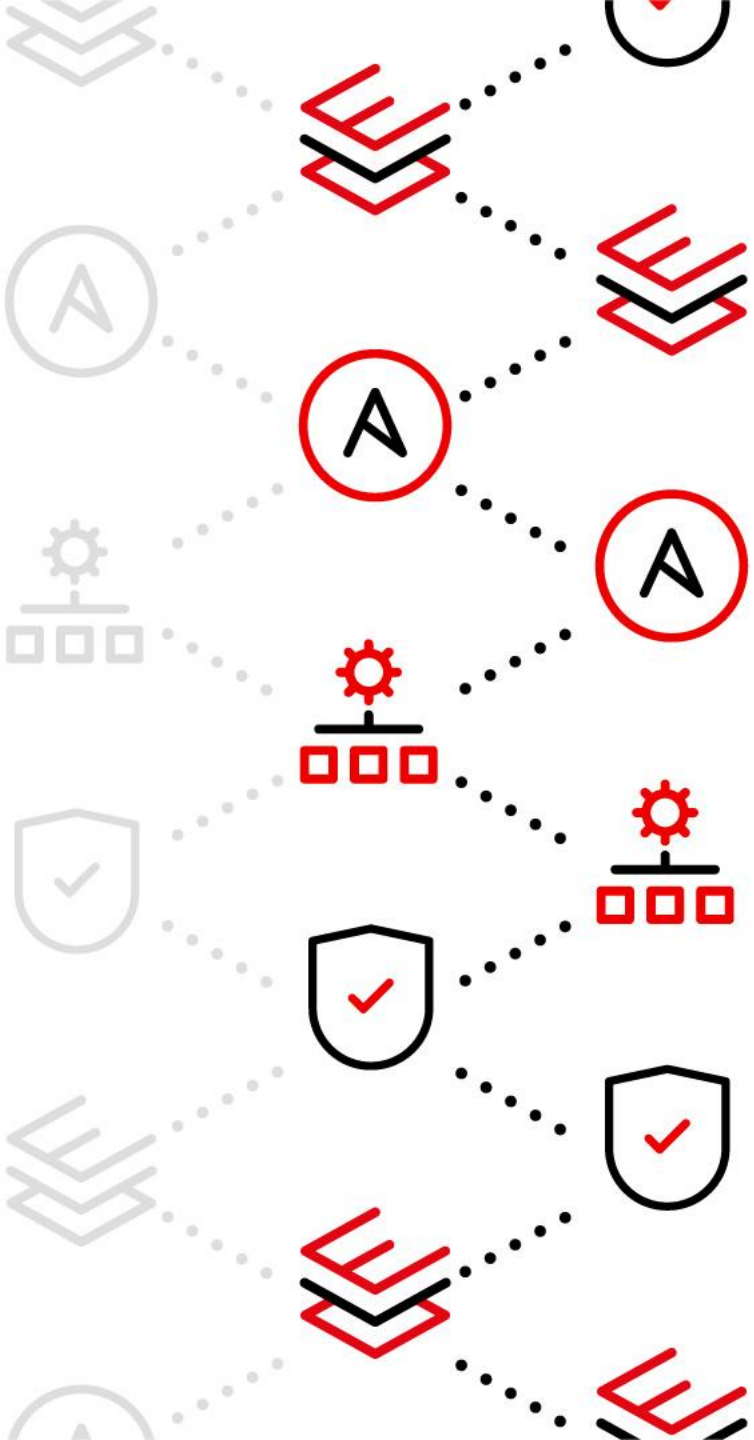


Tech Journey

Massimizza i tuoi investimenti con un’automazione autonoma (EDA)

Marcello Panci
Ansible Sales Specialist

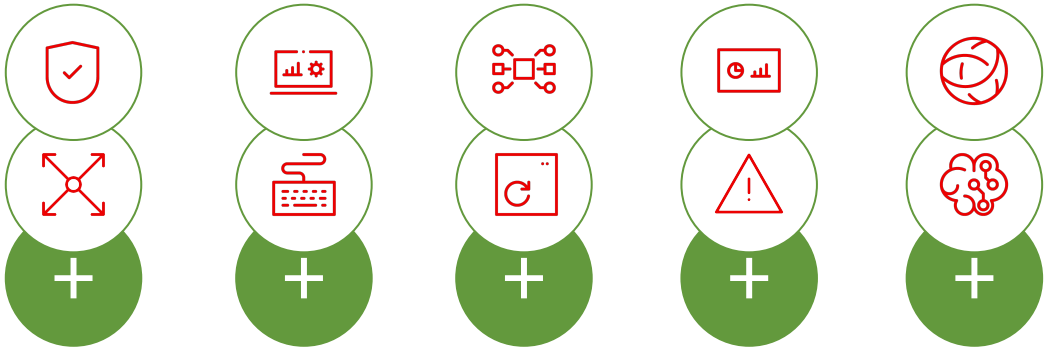
Fabio Alessandro “Fale” Locati
Principal Specialist Solution Architect

Red Hat

Many organizations share the same challenge.

Differing skills, roles, and responsibilities

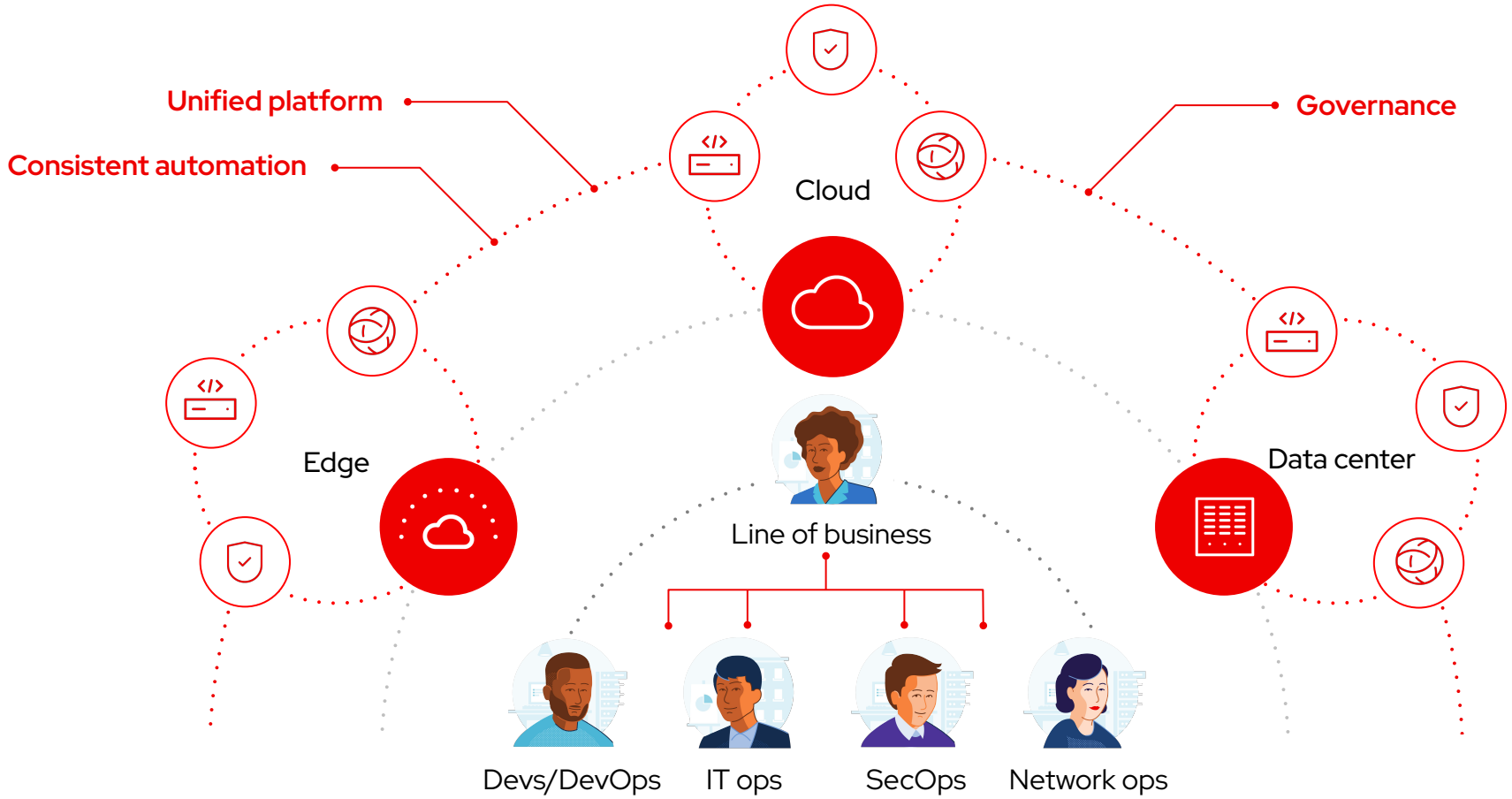
Growing number of use cases



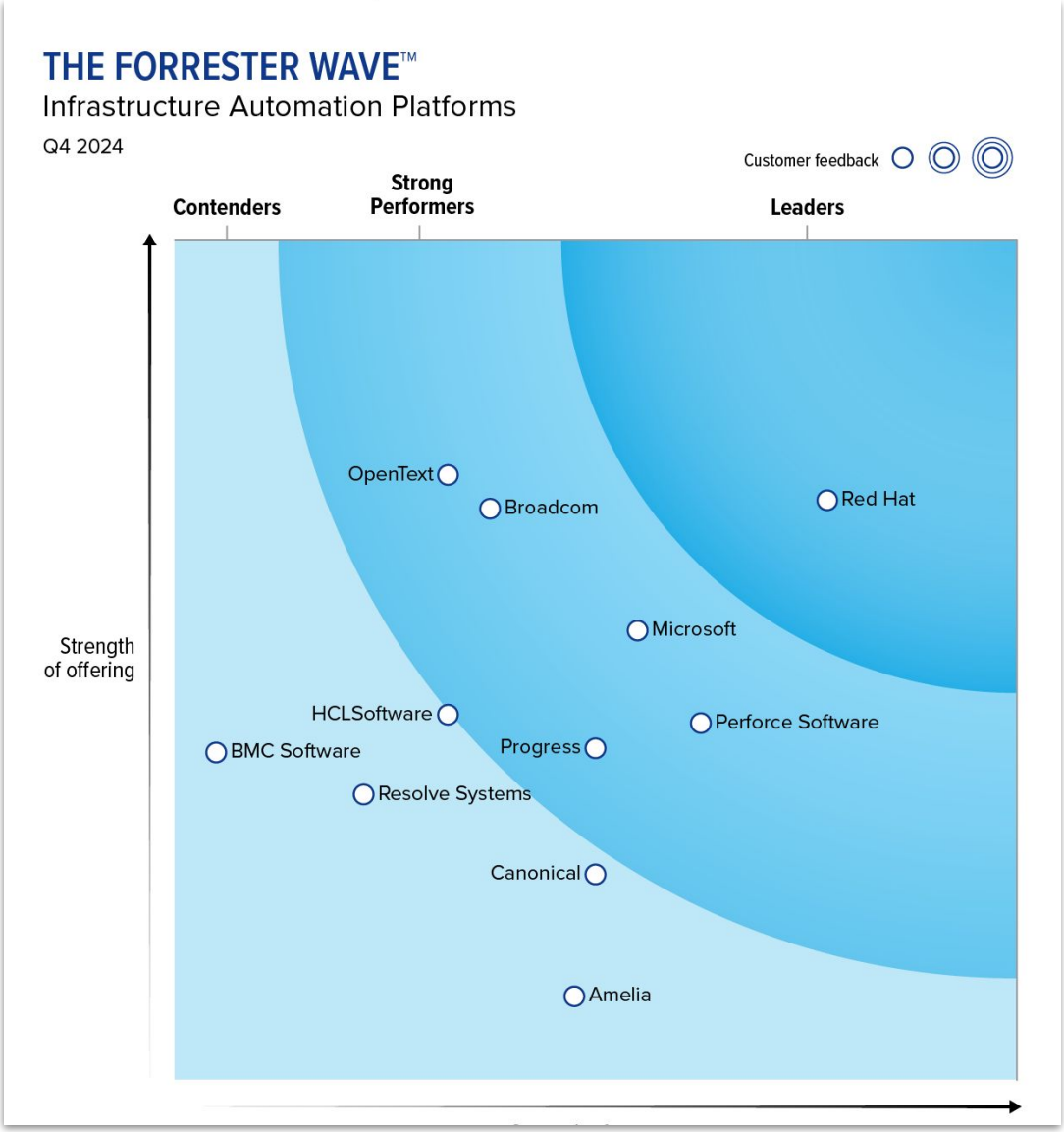
Multiple platforms and locations

Across multiple domains

The solution? Break down the silos.



Red Hat is *the leader* in the 2024 Forrester Wave™: Infrastructure Automation Platforms



Key takeaways from vendor profile

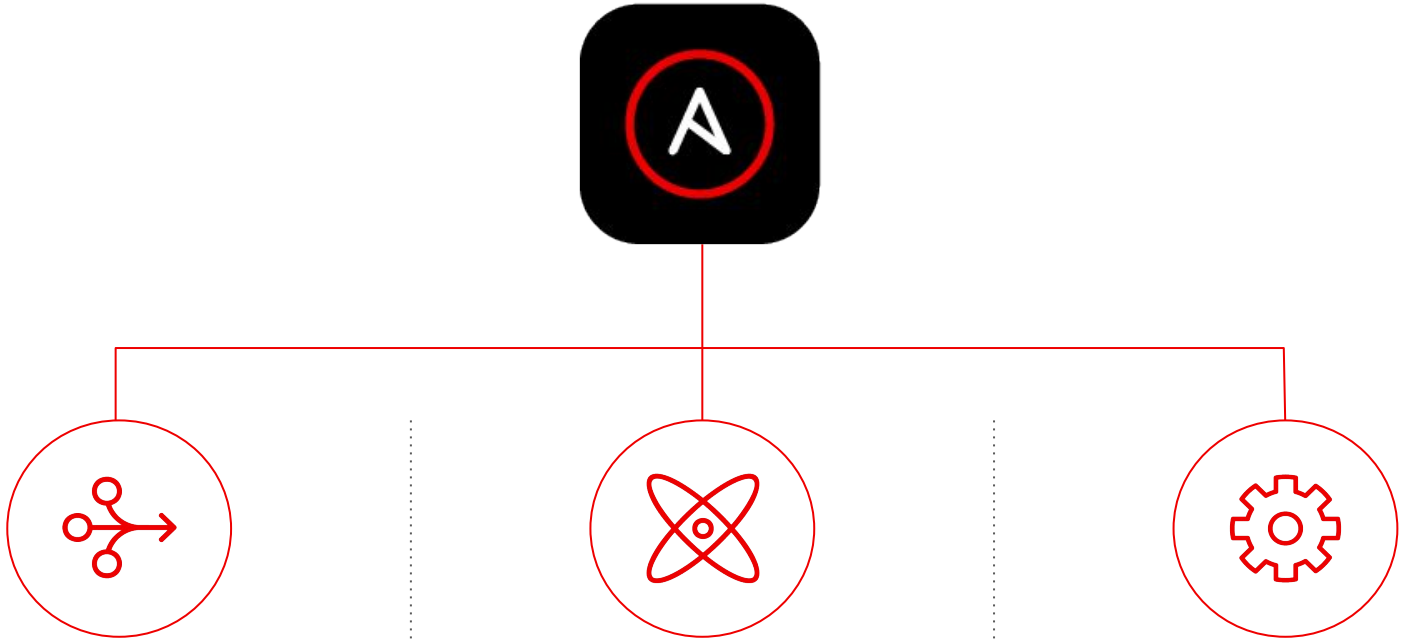
- "The Ansible Automation Platform is a comprehensive solution for various use cases, expanding into event-driven automation to reduce human intervention and reduce the burden of writing code for each scenario."

- "Red Hat's strategy for a collective and inclusive user community is spurring growth. Its vision and partner ecosystem are both strengths…supported by the community, Red Hat Ansible addresses numerous automation use cases."

- "Red Hat offers a good all-around automation tool for organizations that aim to enable users to develop mature automation and integrate technology silos."

Source: Forrester Research, "The Forrester Wave™: Infrastructure Automation Platforms, Q4 2024," 2024.

Ansible is the de facto automation language.



Simple

Human-readable language with quick adoption

Powerful

Universal language spanning multiple IT domains

Agentless

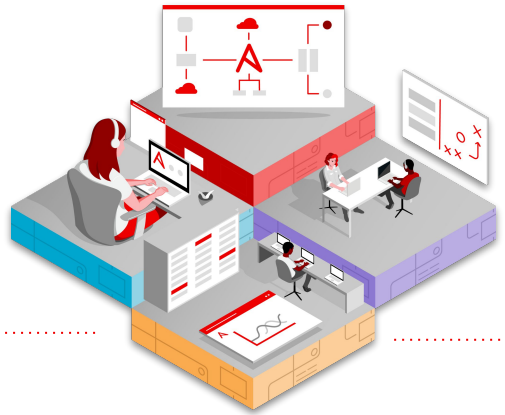
Easily integrate with hybrid environments

Red Hat Ansible Automation Platform

A comprehensive IT automation solution

Ansible is the **de facto** language of automation.

Simple | Human readable | YML



Ansible Automation Platform is engineered to help IT teams **create, manage, and scale** their automation.

Applications

- DevOps
- CI/CD
- GitOps

Network

- Network visibility
- Configuration management
- Network operations

Cloud

- Orchestration
- Operationalization
- Governance

Security

- Investigation enrichment
- Threat hunting
- Incident response

Infrastructure

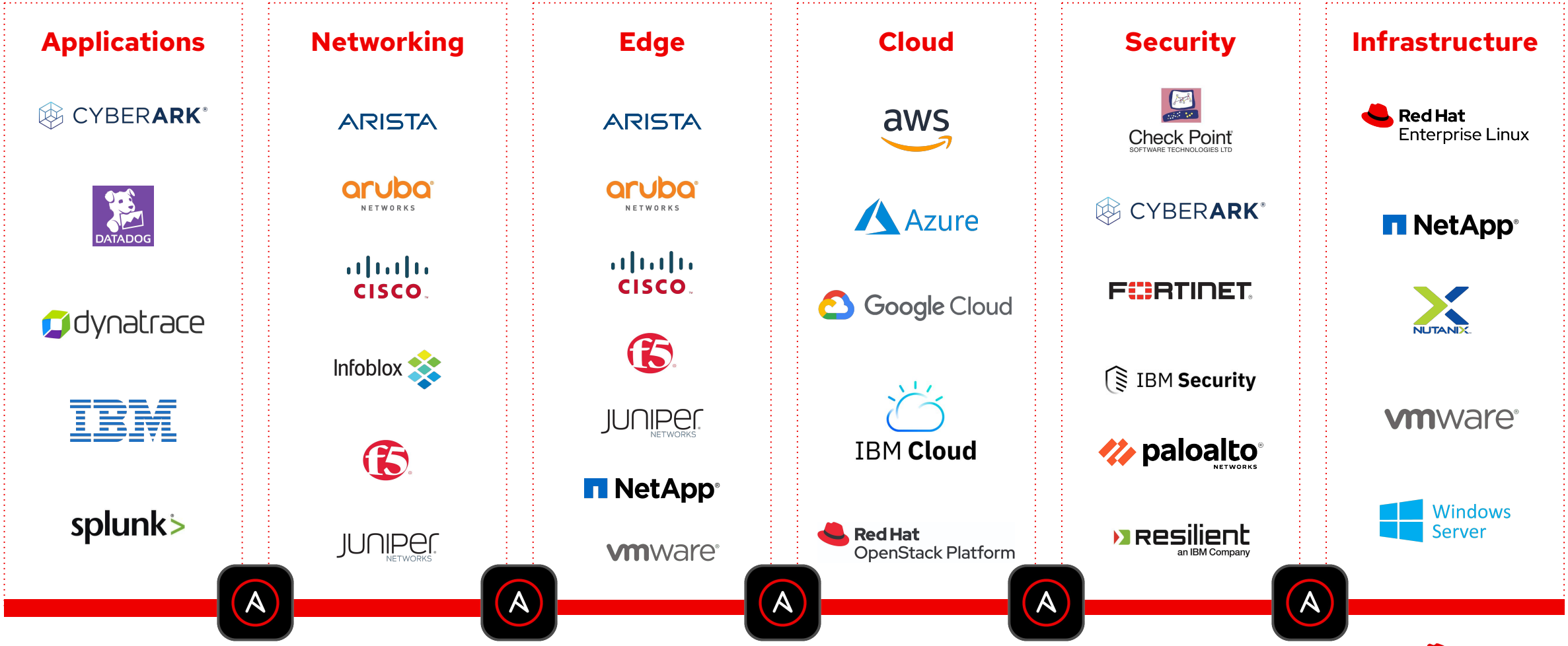
- Deployment
- Provisioning
- Management

Edge

- Infrastructure, network and security extension
- Industrial/IoT device automation
- Manufacturing/Retail remote site management

Seamless integration with other mission critical tools

Building mission critical workflows across the open hybrid cloud ecosystem



Red Hat

Supported and certified content you can trust.

170+

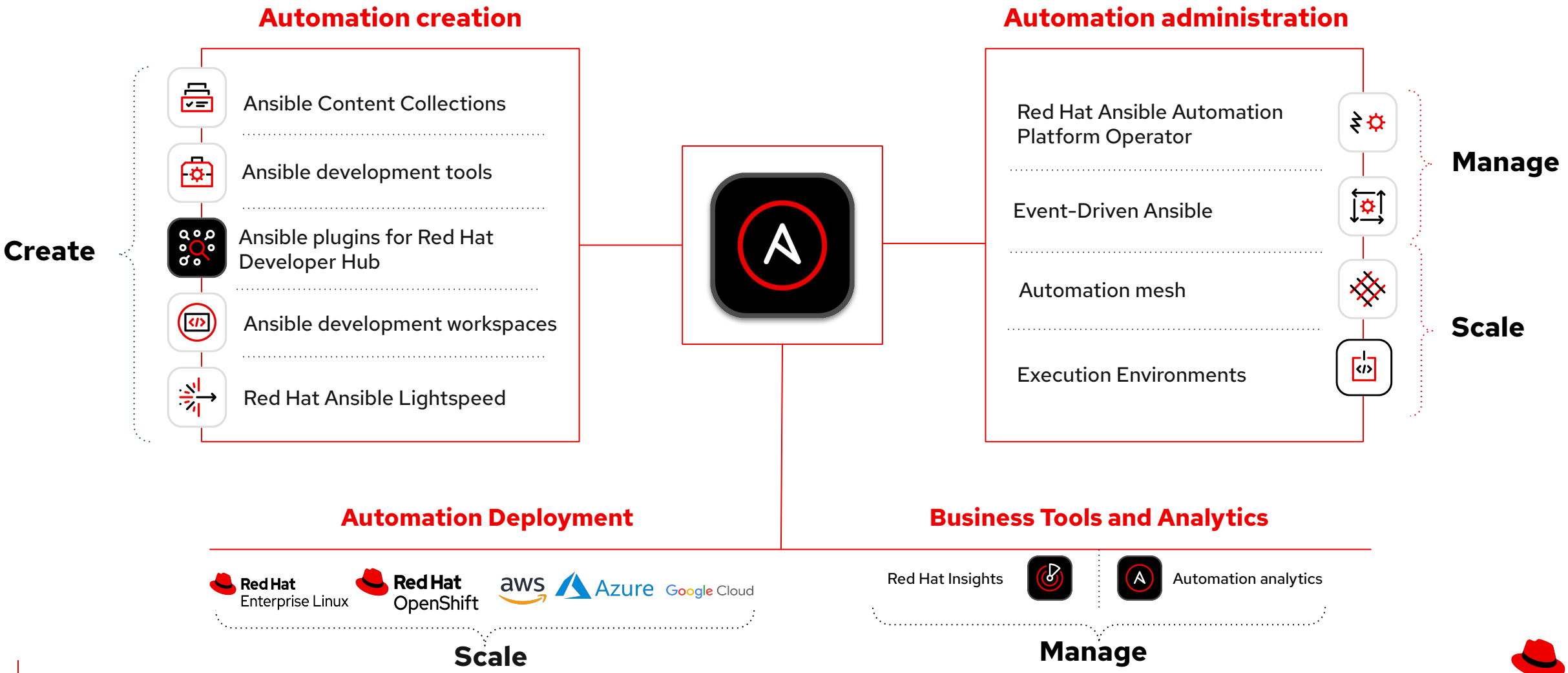
Certified and Validated Content Collections

70+

Certified technology partners

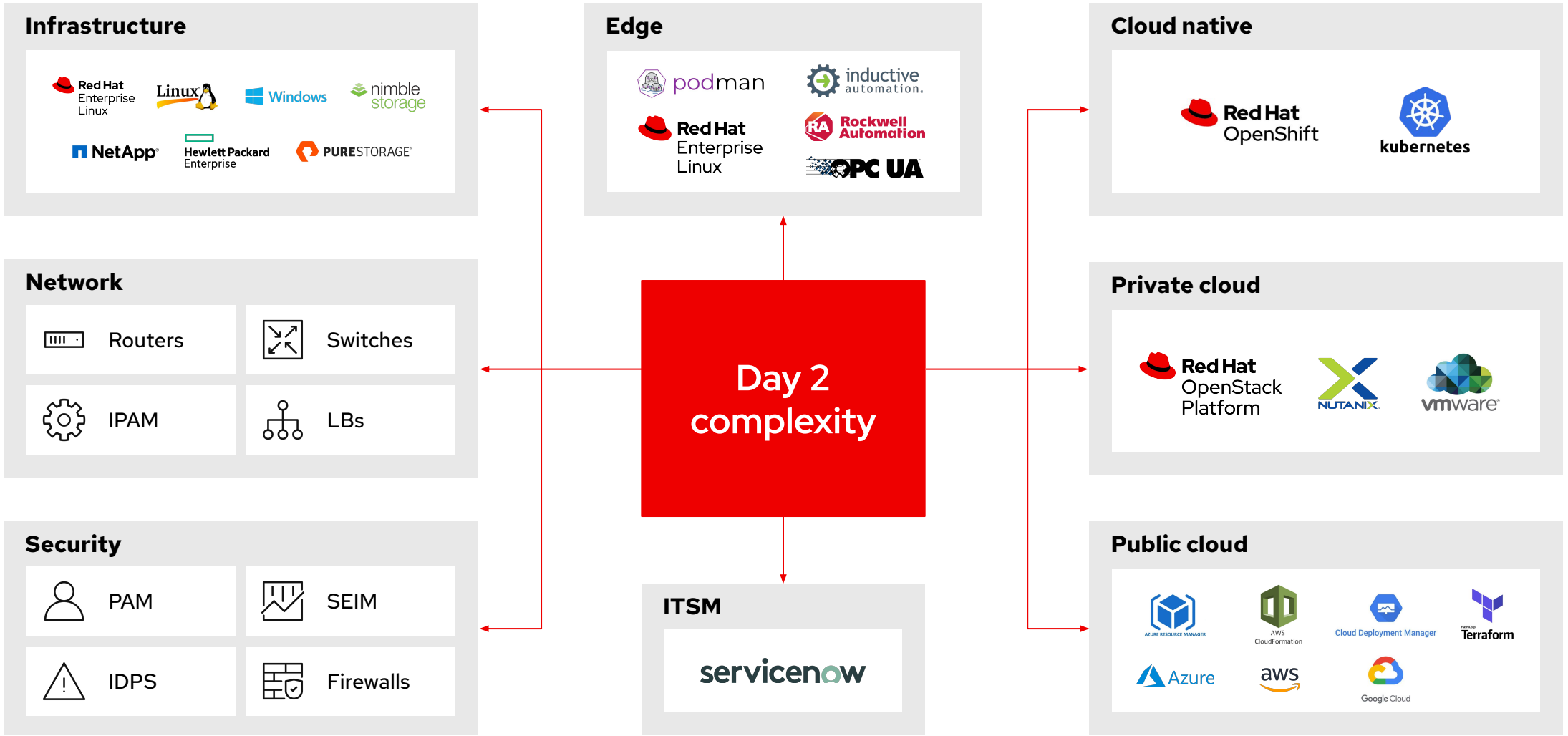
Infrastructure | Cloud | Network | Security | Edge

An integrated solution for the enterprise.

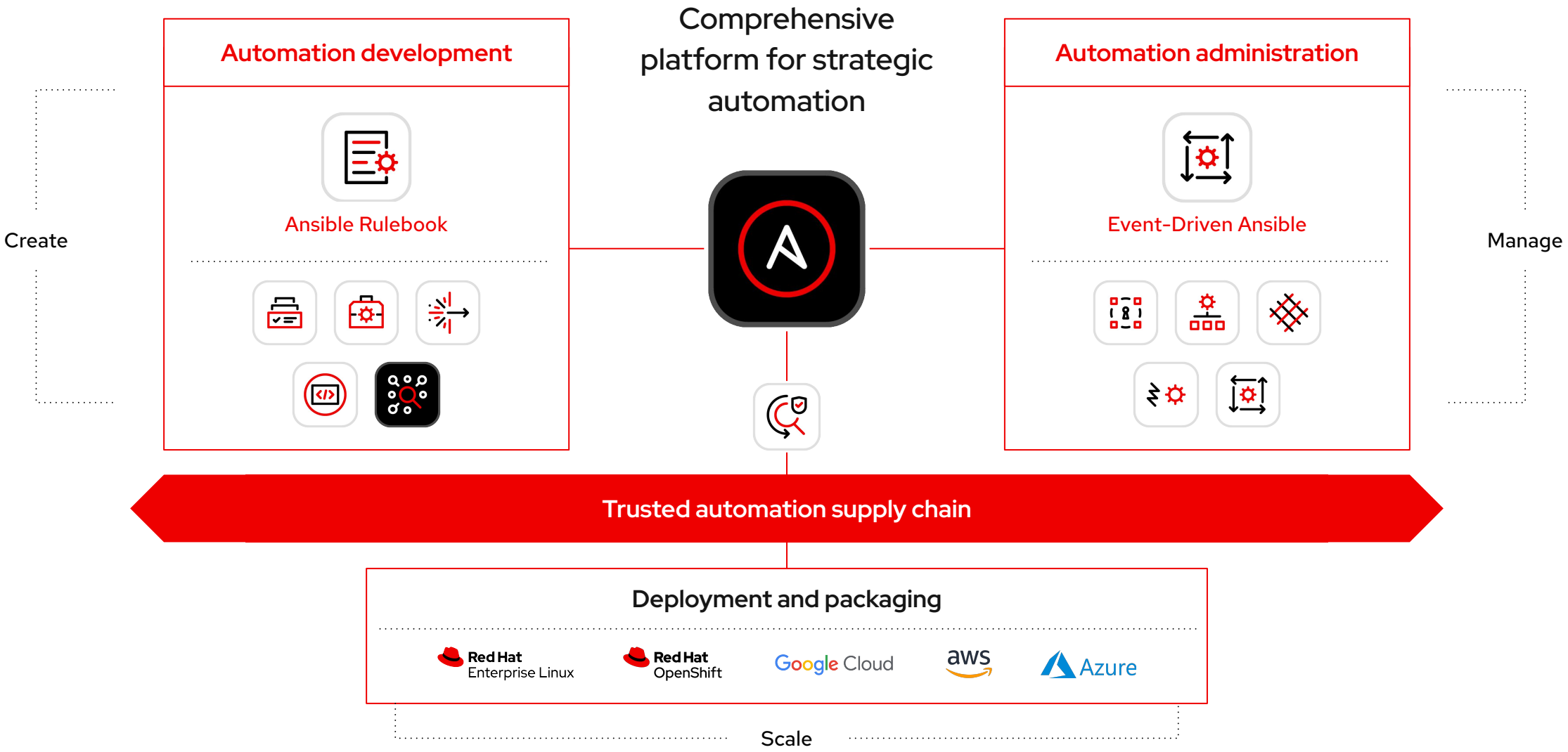


Event Driven Automation

The reality of hybrid infrastructure

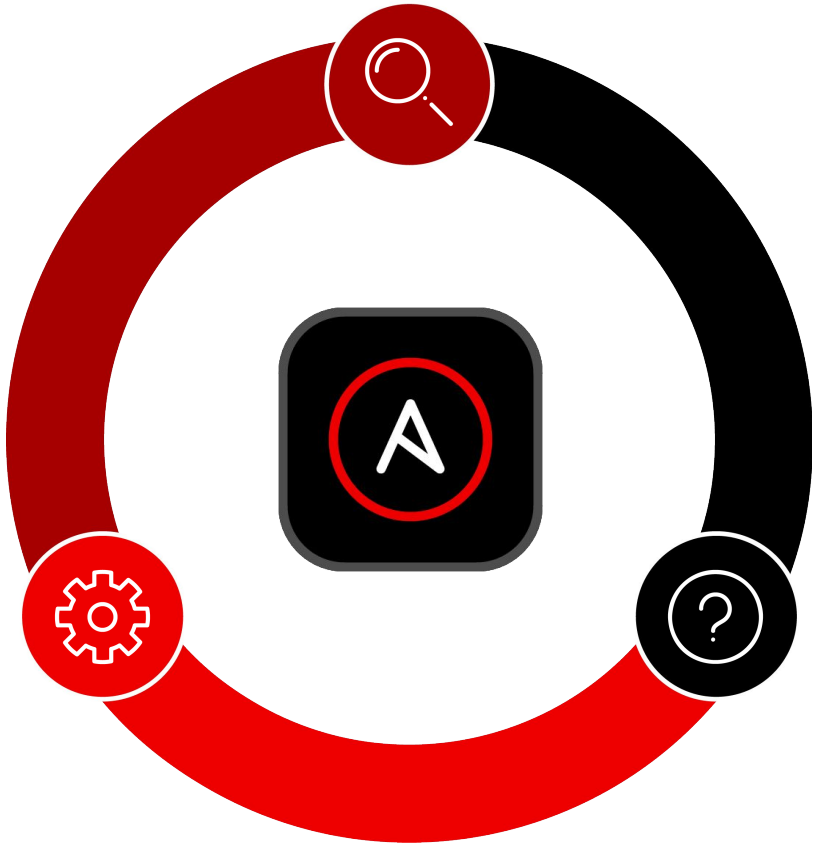


Part of Red Hat Ansible Automation Platform



Automate actions and responses for mission-critical workloads

Event-Driven Ansible receives alerts and responds automatically when conditions are met



Event

Observe

- Watch data / streaming data
- Identify event
- With or without notification and ticketing integration

Decision

Evaluate

- Routed for remediation
- Identify known problem
- Trigger required workflow

Automation

Respond

- No action required
- Automated resolution triggered
- Remediation action completed
- Shorter MTTR

Event-Driven Ansible is partner friendly

Use case and tool friendly event-driven automation

INSTANA an IBM Company

dynatrace

kentik

paloalto NETWORKS

servicenow

IBM Turbonomic

LogicMonitor

CROWDSTRIKE

CYBERARK

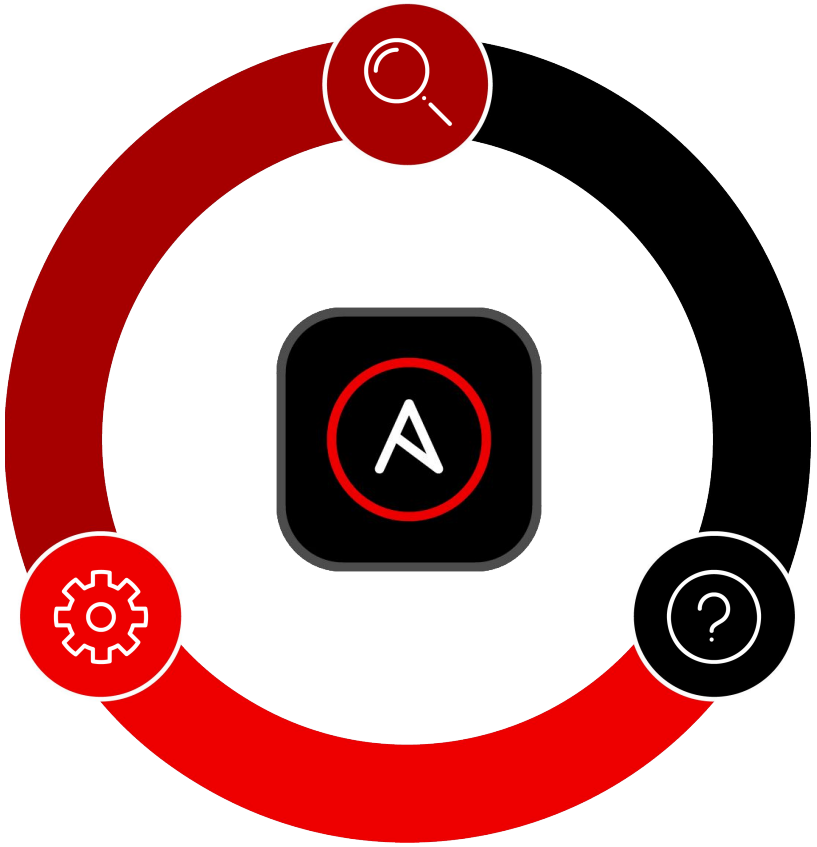
BigPanda

ZABBIX

f5

Red Hat Insights

ansible.eda



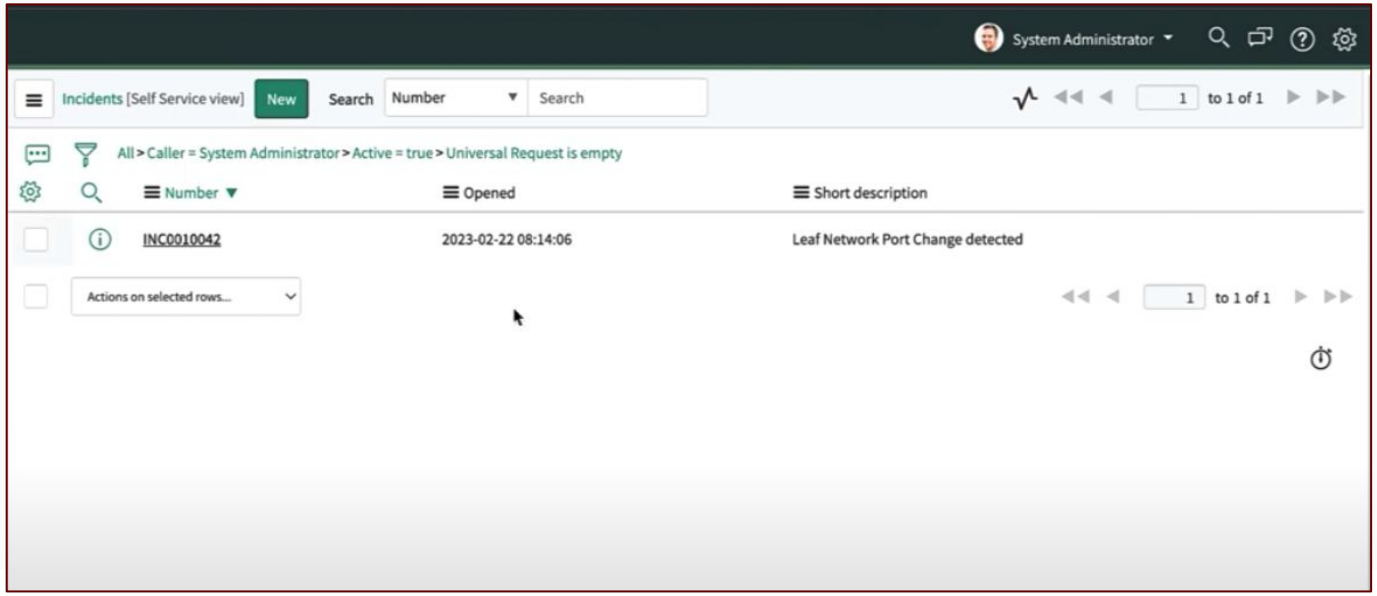
[Blog: custom plugins]

Event Observation as code

Observe first, remediate later

```
- name: Port state events
  hosts: leaf-switches
  sources:
    - ansible.eda.kafka:
        host: kafka-broker.acme-corp.com
        port: 9092
        topic: campus-net
  rules:
    - name: Port is down
      condition: event.fields.admin_status == "DOWN"
      action:
        run_playbook:
          name: ticket_escalation_event.yml
```

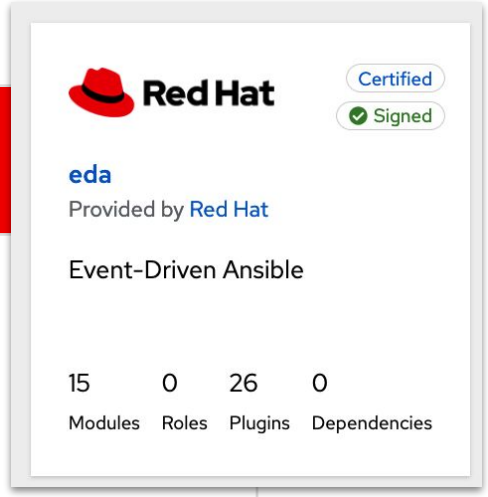
- **Infrastructure awareness from events**
  - Gather critical information around events to assist your teams in remediation
  - Observe changes in infrastructure and applications from events.
  - Ensure troubleshooting steps are consistent



What is in the ansible.eda collection?

ansible.eda certified Content Collection

- AWS SQS
- AWS CloudTrail
- Azure Service Bus
- Kafka (AMQ Streams)
- Prometheus/Alertmanager
- Webhooks
- watchdog (file system watcher)
- url_check (url status check)
- range (event generation plugin)
- file (loading facts from yaml)
- journald
- Tick
- Ability for Event-Driven Ansible to automate itself



ansible.eda

Custom event source plugins blog

Rule Overview

Rules >> Rule sets >> Rulebook

- **Rules require**
  - Conditions to match
  - and an Action(s)
- **Rulesets require**
  - An event source
  - Conditions to match
  - and an Action(s)
- **Rulebooks require**
  - Rulesets

Rules

Executing actions from matched conditions

- **Rules consist of conditional statements**
  - Event-Driven Ansible uses rules to determine if an action or actions should take place
  - Rules can have single and multiple actions
  - Rules are organized into Rulesets
  - Getting started – Ansible Rulebook Documentation

```
rules:
  - name: An remediation rule with 1 action
    condition: event.outage == true
    action:
      run_playbook:
        name: remediate_outage.yml

  - name: A remediation rule with multiple actions
    condition: event.outage == true
    actions:
      - run_playbook:
          name: remediate_outage.yml
      - print_event:
          pretty: true
```

Matching Event conditions

Smart automation from conditional rules

Event Source

```
...

rules:
  - name: Check for folder modification
    condition: event.type == "DirModifiedEvent"
    action:
      run_playbook:
        name: folder_permission_restore.yml
        post_events: true
  - name: Check for file Modification
    condition: event.type == "FileModifiedEvent"
    action:
      run_playbook:
        name: file_permission_restore.yml
        post_events: true
```

- Event sources are processed through conditional rules
- Actions are triggered once rule conditions are met

Action

Rulesets

Rulesets: from source to action

- **Rulesets require:**
  - A unique name
  - A defined event source(s)
  - Hosts similar to Ansible Playbooks
  - A list of defined rules
- **Conditional management of actions to events**
  - Simple YAML structure for logical conditions
  - Events can trigger different types of actions:
    - run_playbook, run_template, run_module
    - set_fact, post_event, print_event
    - retract_fact, shutdown, debug
- **Rulesets run separate sessions in the Rules Engine**
  - Events and Facts are kept separate for each ruleset
  - Actions allow a Ruleset to post events or facts to itself or other Rulesets in a Rulebook

```
- name: My unique ruleset
  hosts: all
  sources:
    - name: range
      range:
        limit: 5
  rules:
    - name: First rule
      condition: event.i == 1
      action:
        debug:

    - name: Host specific rule
      condition:
        all:
          - fact.ansible_os_family == "linux"
          - fact.meta.hosts == "my-host"
          - event.i == 4
      action:
          debug:
```

Rulebooks

Ansible Playbook has many plays, Ansible Rulebook has many rulesets

- **Rulebooks comprise of rulesets**
  - Rulebooks can contain multiple Rulesets
  - Rules trigger based on conditions and actions can be carried out by the rules engine
  - Multiple sources can be defined in a Rulebook
  - Rulebooks can have a similar structure to a Playbook with multiple plays.

```yaml
- name: My ruleset 01
  hosts: all
  sources:
    - name: range
      range:
        limit: 5
  rules:
    - name: first rule
      condition: event.i == 1
      action:
        debug:

- name: My ruleset 03
  hosts: all
  sources:
    - name: range
      range:
        limit: 5
  rules:
    - name: first rule
      condition: event.i == 1
      action:
        debug:
```

Working with Conditions

If-else-then

- **Conditions can use information from**
  - Events Received
  - Previously saved events within a rule
  - Longer term facts about a system
  - Variables provided by vars

- **When creating conditions**
  - Use the **event** prefix to access data from the current event
  - Use the **fact** prefix to access data from the set_facts action
  - Use the **events** prefix to assign variables and accessing data in the rule
  - Use the **facts** prefix to assign variables and access data within the rule
  - Use the **vars** prefix to access variables passed via - vars and - env-vars

- **A condition can contain**
  - One condition
  - Multiple conditions where all of them have to match
  - Multiple conditions where any one of them has to match
  - Multiple conditions where not all but one of them have to match

- **Supported condition data types**
  - Integers
  - Strings
  - Booleans
  - Floats
  - null

*Supported operators

Condition Examples

Matching conditions

Single Condition

```
condition:
  event.outage == true
action:
```

Multiple Conditions needing to match

```
condition: (Multiple Events use ALL and list events)
  all:
    - event.target_os == "linux"
    - event.tracking_id == 345
    ----------------------------------------

condition: (Single Event, use AND operator)
  event.host == host1 and event.outage == true
action:
```

Multiple Conditions where any of them has to match

```
condition:
  any:
    - event.target_os == "linux"
    - event.target_os == "windows"
action:
```

Multiple Conditions with facts and events

```
condition:
 all:
  - facts.first << fact.custom.expected_index is defined
  - event.i == facts.first.custom.expected_index
action:
```

Events, Facts and Persistence

Using valuable event data

Events vs Facts

- **Events and Fact represent the same data**
  - Events are short lived and discarded once processed
  - Facts offer persistence and can come as a result of an action or be set with **set_fact**
  - Facts are kept in memory until explicitly removed
  - Events and Facts cannot be part of the same condition
  - Variables provided by vars
  - A matched event or events are sent to the playbook through **extra_vars** under the **ansible_eda** namespace
  - Facts are not sent to playbooks

```
condition:
  event.outage == true and fact.beta_enabled != true
action:
```

Use the **all** operator to combine events and facts in a condition

```
condition:
  all:
   -  event.outage == true
   -  fact.beta_enabled != true
action:
```

Calling events in playbooks

ansible_eda

```
...playbooks/say_thanks.yml
---
- name: say thanks
  hosts: localhost
  gather_facts: false
  tasks:
    - debug:
        msg: "Thank you, {{ ansible_eda.event.sender }}!"
```

Action: run_playbook

Call event data

```
ansible-rulebook --rulebook kafka-example.yml -i inventory.yml --print-events
{   'message': 'Ansible is cool',
    'meta': {   'received_at': '2023-04-16T08:27:31.343577Z',
                'source': {   'name': 'ansible.eda.kafka',
                              'type': 'ansible.eda.kafka'},
                'uuid': 'ab9ec116-799c-4a80-9e9d-5ec48f4fc214'},
    'sender': 'Nuno and Colin'}

PLAY [say thanks] **************************************************************

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "Thank you, Nuno and Colin!"
}

PLAY RECAP *********************************************************************
localhost                  : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

Variables in Event-Driven Ansible

Rulebooks and variable handling

- **Variable files --vars**
  - Importing of variables from JSON or YAML requires the **--vars** argument
  - Using --vars preserves the data type
- **Environment Variables --env-vars**
  - Importing of variables from the environment requires the **--env-vars** argument
  - Variables from --env-vars are always treated as strings
  - Environment variables take precedence
- **Using Variables in a Rulebook**
  - In conditions variables supplied at runtime are placed in the **vars** namespace

    ```
    condition:
      event.i == vars.my_var
    ```
  - When accessing variables outside of conditions use Jinja substitution

```yaml
---
- name: Testing vars
  hosts: all
  sources:
    - ansible.eda.range:
        limit: "{{ src_range_limit }}"
  rules:
    - name: Say Hello
      condition: event.i == vars.match_this_int
      action:
        debug:
          msg: "Hi, I'm {{ MY_NAME }}."
```

Extra Variables and Actions

Playbook and Template actions requiring extra_vars

- **Both run_playbook and run_job_template actions have an optional parameter of extra_vars**
  - Playbooks or templates might require additional variables, using **extra_vars** in the action all provide these variables at runtime.
  - The rulebook engine will automatically insert the event(s) into the top level key **ansible_eda**

- **Limiting the hosts which are targeted by the action**
  - Configure the meta data of the event to provide hosts. This can be comma delimited with hosts

```
{
    "i": 0,
    "meta": {"hosts": "localhost"}
}
```

Rulebook Action:

```
action:
  run_job_template:
    name: Trigger Provisioning
    organization: Default
    job_args:
      extra_vars:
          os_version: "{{ Linux_OS }}"
```

Playbook Tasks:

```
tasks:
    - name: Print variable foo set by user
      ansible.builtin.debug:
        msg: '{{ foo }}'
    - name: Print variable event
      ansible.builtin.debug:
          msg: '{{ ansible_eda.event }}'
```

Filtering Events

Clearing out the extra data and defining what is relevant

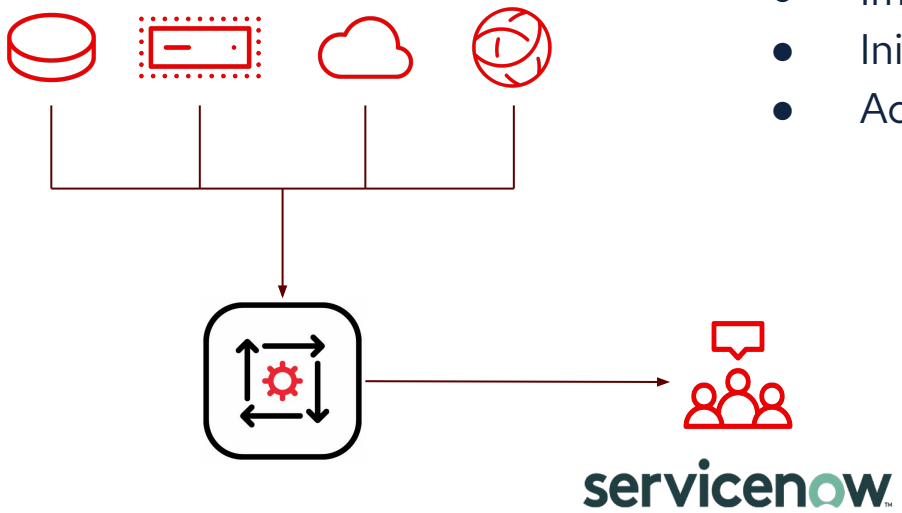
- **Filters provided in the Event-Driven Ansible Collection**
  - Include and exclude keys from the event object with **json_filter**
  - Change dashes in all keys in the payload to underscores with **dashes_to_underscores**
  - Filters can be chained one after the other
  - Event filters must be defined in the source block of the ruleset

```
ansible-rulebook --rulebook kafka-example.yml -i inventory.yml --print-events
{   'message': 'Ansible is cool',
    'meta': {   'received_at': '2023-04-16T08:27:31.343577Z',
                'source': {   'name': 'ansible.eda.kafka',
                              'type': 'ansible.eda.kafka'},
                'uuid': 'ab9ec116-799c-4a80-9e9d-5ec48f4fc214'},
    'sender': 'Nuno and Colin'}
```

```
filters:
      - json_filter:
          include_keys: ['clone_url']
          exclude_keys: ['*_url', '_links', 'base']
        - dashes_to_underscores:
```

- **Meta key is used to store metadata about events**
  - Each event has the **eda.builtin.insert_meta_info** filter added by ansible-rulebook
  - This filters the origin source, type, uuid for the event and event payload information

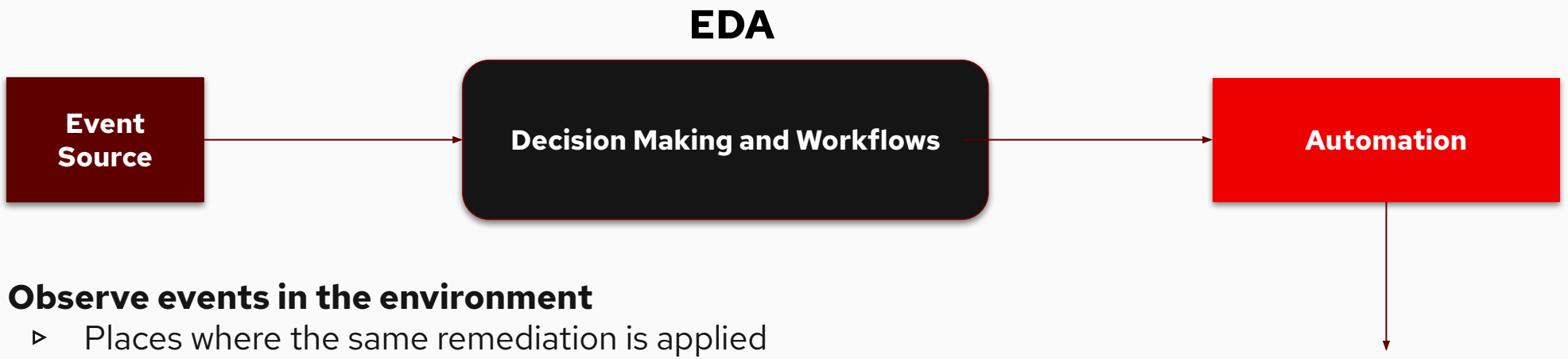
Fact and Ticket Enrichment



- Enhance observation with automated fact gathering
- Improve troubleshooting consistency
- Initiate change management
- Act on events safely without impacting systems

Event-Driven Ansible and ServiceNow ITSM integration

Events to human observation



- **Observe events in the environment**
  - Places where the same remediation is applied again and again.
- **Use events to trigger ITSM ticket escalation**
  - Critical system or infrastructure failure triggers an action to create an incident on ServiceNow for human intervention.
- **Update ServiceNOW CMDB**
  - Infrastructure changes can be observed and used to trigger ServiceNow to update its inventory

Ansible Certified Content Collection for ServiceNow ITSM 2.0

API for Red Hat Ansible Automation Platform Certified Content Collection*

API for Red Hat Ansible Automation Platform Certified Content Collection and Event-Driven Ansible Notification Service

Integrates EDA and Ansible Certified Content Collection for ServiceNow ITSM into your ServiceNow instance

Available at
store.servicenow.com

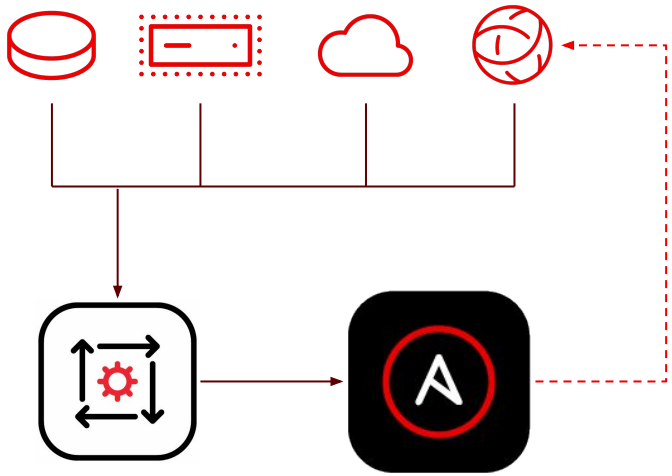
Ansible Certified Content Collection for ServiceNow ITSM

Helps create new automation workflows faster; while establishes a single source of truth in the ServiceNow CMDB.

Available at
console.redhat.com

**For ServiceNow ITSM releases Tokyo and later*

Automatic remediation



- Remediate high occurrence low complexity events
- Free up crucial skills within teams and remove toil
- Improve Mean time to remediation
- Self-healing workloads and infrastructure

Simple Automation Mastered

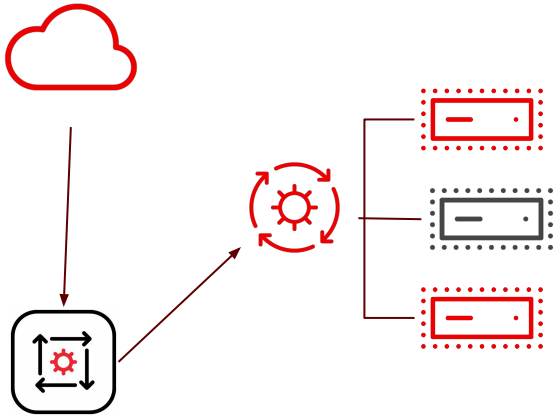
Remediate without hassle

- **Maintain applications and services**
  - Restart, redeploy applications and services
  - Remediate system issues without interrupting your technical teams
  - Self-healing infrastructure
- **Automate business functions**
  - Configure new user access to resources
  - Automate CMDB updates

```
- name: Check Web Applications
  hosts: web_servers
  sources:
    - ansible.eda.url_check:
        urls:
-         - http://mobileapp.acme-corp.com
          - http://banking.acme-corp.com
        delay: 10
  rules:
    - name: Web-applications are down
      condition:
        event.url_check.status == "down"
      action:
        run_playbook:
          name: redeploy_mobile_app.yml
```

- **Automate remediation on critical events**
  - Ensure security configurations are maintained
  - Limit configuration drift
  - Maintain Cloud infrastructure footprint and hygiene
  - Enrich log capturing and security responses

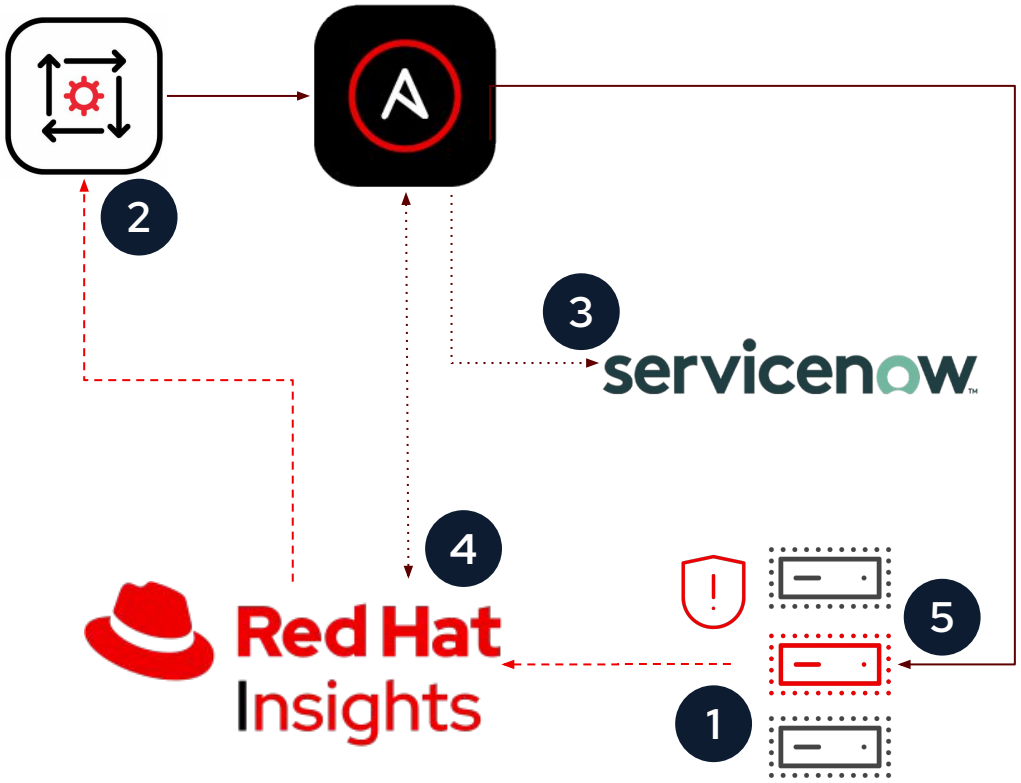
Insights and Event-Driven Ansible



- Utilize Red Hat Insights to remediate configuration drift
- Event-Driven Security and Compliance automation
- Improve recommendation to remediation time
- Manage vulnerability, compliance and malware at scale

Event-Driven response to Insights Events

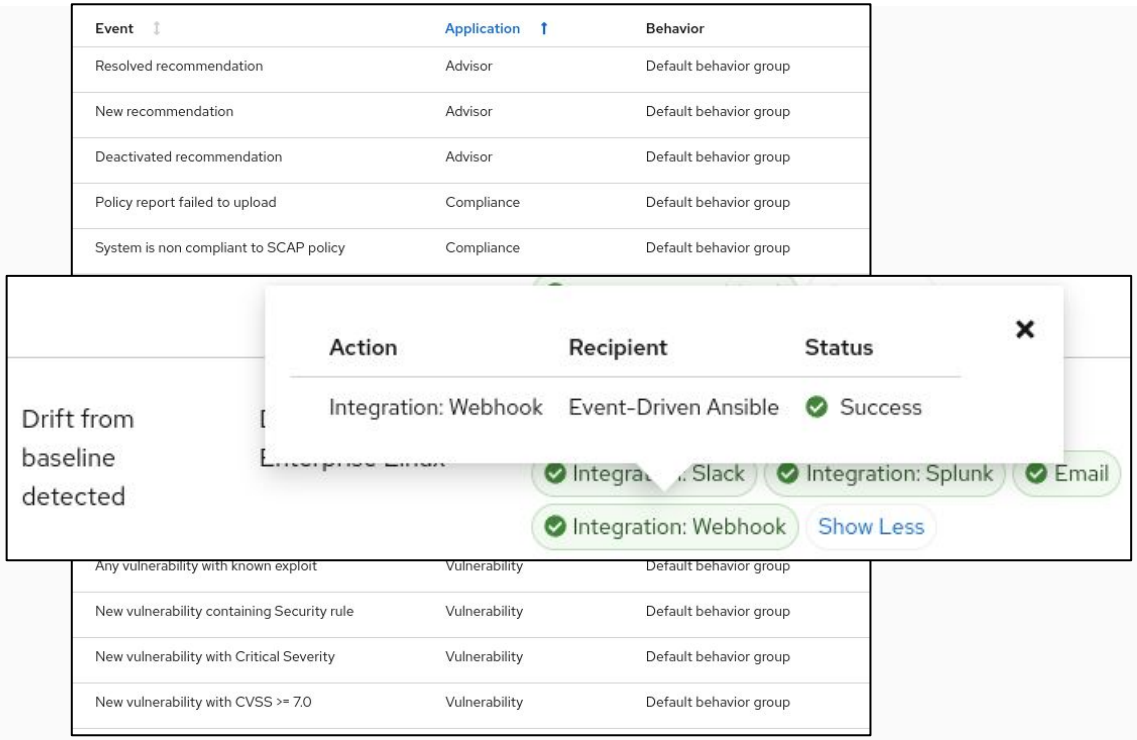
Orchestratrate Self Healing with Insights and Ansible Automation Platform



- **Red Hat Insights and Ansible Automation Platform provide:**
  - **Vulnerability and Malware Management**
    - Automate vulnerability scanning
    - Gather CVE/Malware data from Insights
    - Create Inventories to target affected hosts
    - Create remediation Playbooks and import them into Automation Controller
    - Run remediation and update Insights
  - **Compliance Management**
    - Schedule daily compliance evaluations at scale
    - Automate compliance remediation playbooks
    - Remedy organizational and governance compliance at scale.

From Insights to Remediation

Audit systems and respond




```yaml
---
- name: Listen for events on a webhook
  hosts: all
  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
  rules:
    - name: Handle Red Hat Insights event
      condition: event.payload is defined
      action:
        debug:
          msg: "Received: {{ event.payload }}"
```

- **Red Hat Insights as a source**
  - Gather all recommendations from Insights via payload
  - Filter for relevant recommendations and create actions to remediate or configure your systems to the best possible standard

Red Hat

Tech Journey

Esplora, scopri e innova: la tua serie di webinar e demo

PROSSIMI INCONTRI

27 maggio, 2025 (1 ora)
Webinar
Semplifica l'automazione della tua rete ora.

10 giugno, 2025 (1 ora)
Webinar
Ansible Lightspeed - automatizza con la potenza dell'IA

24 giugno, 2025 (1 ora)
Webinar
Red Hat Ansible Automation Platform: l'automazione con un focus sulla sicurezza

Scopri altri viaggi tecnologici

Build your automation strategy

Adopt and scale AI

One platform. Unlimited potential

Straightforward migration. Lasting modernization.

Accelerate your application modernization journey

Modernize your network with automation

Thank you

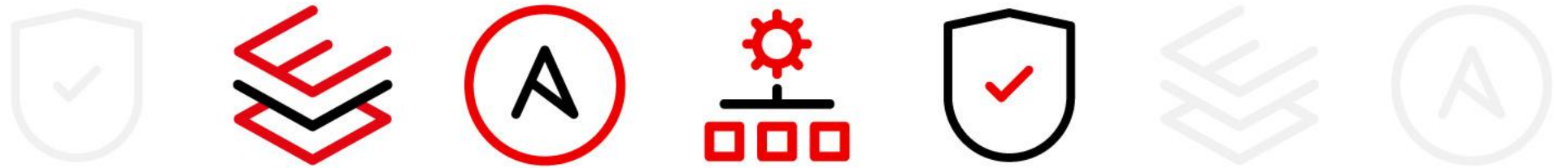
Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

x.com/RedHat



Red Hat