Leverage Event Driven Ansible to reduce your automation reaction time

Fabio Alessandro "Fale" Locati
EMEA Associate Principal Specialist Solutions Architect, Red Hat

Red Hat

TOC

How Event-Driven Ansible can help

How does EDA work

Takeaways

About me

- Working in IT since 2004, mostly in operations roles
- Ansible user since 2013
- Author of 5 books, 4 of which on Ansible
- EMEA Associate Principal Specialist Solution Architect for Ansible @ Red Hat

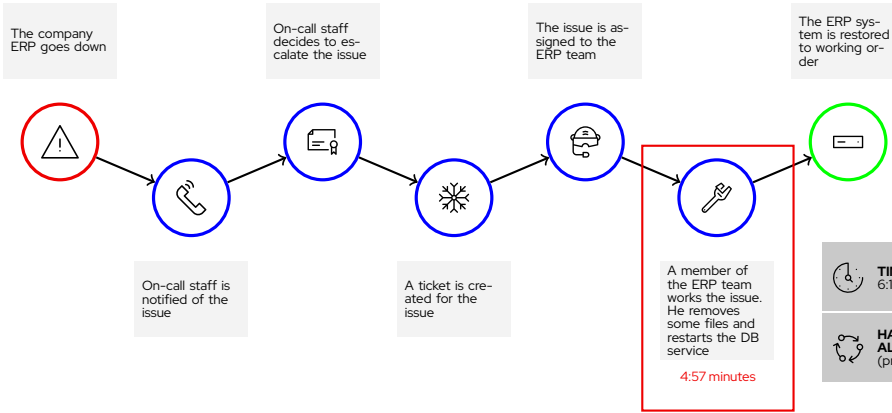
Disclaimer

Everything we will discuss today is fully Open Source.

It works in the same way on both Community and Enterprise editions.


PowerEdge R300

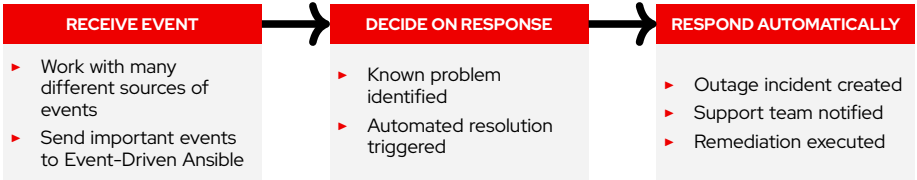
What happened



Red Hat

How Event-Driven Ansible can help

Automate event workflow



WORK ACROSS MULTI-DOMAIN AND MULTI-VENDOR IT OPERATIONS

Work flexibly and well with multi-domain and multi-vendor monitoring and other solutions across the event driven architecture with appropriate approvals, controls and awareness

Red Hat

Event-Driven Ansible advantages

Flexible event-driven automation

Flexible from source to rule to action with multiple event sources. Create and change automation easily.

IT environment friendly

Automate any IT use case quickly and simply. Jumpstart with many content collections available.

Robust automation handling

Scalable decisioning and implementation with flexible actions.

Single automation platform

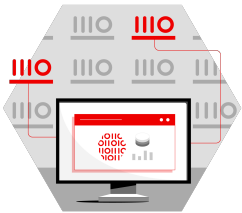
Choose your automation style, leverage existing automation content and extend skills.

Common use cases

- **Networking**: port events, route events
- **Infrastructure**: resource limits events
- **Security**: IDS events, user creation events
- **Applications**: service events
- **Cloud**: scaling events, service bus events
- **Logs**: logs enrichment

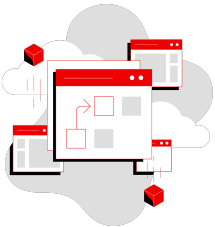
How does EDA work

Key building blocks in EDA



Sources

All the sources of event data you want to use



Rules

What you will create using Event-Driven Ansible

Actions

When a condition or event is met, the Ansible Rulebook executes

Events sources

- Common
  - file (loading facts from yaml)
  - file_watch
  - journald
  - range
  - url_check (url status check)
  - webhooks
- Clouds
  - AWS CloudTrail
  - AWS SQS
  - Azure Service Bus
- Specific software
  - CrowdStrike
  - F5
  - IBM Instana
  - IBM Turbonomic
  - LogicMonitor
  - Kafka (AMQ Streams)
  - Palo Alto Networks
  - PostgreSQL PubSub
  - Prometheus/Alertmanager
  - Red Hat Insights
  - Zabbix
- Brying your own

Event source: webhook example

```
sources:
  - ansible.eda.webhook:
      host: 0.0.0.0
      port: 5000
```

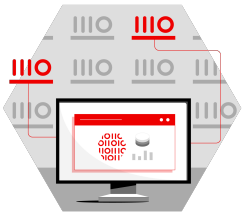
Events Filters

- Clearing out the extra data and defining what is relevant
- Provided filters:
  - 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`
- Each event has the `eda.builtin.insert_meta_info` filter added by ansible-rulebook
- Filters can be chained one after the other
- Bring your own filters!

Filters: an example

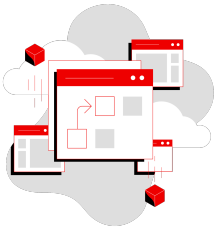
```
filters:
  - json_filter:
      include_keys: ['clone_url']
      exclude_keys: ['*_url', '_links', 'base']
  - dashes_to_underscores:
```

Key building blocks in EDA



Sources

All the sources of event data you want to use



Rules

What you will create using Event-Driven Ansible

Actions

When a condition or event is met, the Ansible Rulebook executes

Rules

- Event-Driven Ansible uses rules to determine if an action or actions should take place
- Can have a single or multiple conditions
- Can have a single or multiple actions

Rules Conditions

- Conditions can use information from:
  - Received event
  - Previously saved events within a rule
  - Longer term facts about a system
  - Variables provided by 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
- Supported condition data types: integers, strings, booleans, floats, `null`
- Is possible to set `facts` and `events` in rules

Rules Actions

- 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`

Rules: a couple of examples

```
rules:
  - name: A remediation rule with one condition and one action
    condition: event.outage == true
    action:
      run_playbook:
        name: remediate_outage.yml


  - name: A remediation rule with multiple conditions and actions
    condition:
      all:
        - event.outage == true
        - fact.ansible_os_family == "linux"
    actions:
      - run_playbook:
          name: remediate_outage.yml
      - print_event:
          pretty: true
```

Rules throttling

- Group events by attributes
- Possible to run the first time in a timeframe with `once_within`
- Possible to collect the events in the timeframe and then run with `once_after`
- Time units are milliseconds, seconds, minutes, hours, days

Rules throttling: a couple of examples

```
rules:
  - name: Throttle example reactive
    condition: event.outage == true
    throttle:
      once_within: 5 minutes
      group_by_attributes:
        - event.meta.hosts
        - event.code
    action:
      run_playbook:
        name: notify_outage.yml


  - name: Throttle example passive
    condition: event.outage == true
    throttle:
      once_after: 5 minutes
      group_by_attributes:
        - event.meta.hosts
        - event.code
    action:
      run_playbook:
        name: notify_outage.yml
```

Rulesets

- A ruleset requires:
  - A unique name
  - A defined event source(s)
  - Hosts (similarly to Ansible Playbooks)
  - A list of defined rules
- 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

Rulesets: an example

```
---
- name: My ruleset
  hosts: all
  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
  filters:
    - json_filter:
        include_keys: ['clone_url']
        exclude_keys: ['*_url', '_links', 'base']
  rules:
    - name: My remediation rule
      condition: event.outage == true
      action:
        run_playbook:
          name: remediate_outage.yml
```

Rulebooks

- Rulebooks are made of one or more rulesets
- Multiple different sources can be defined in a Rulebook
- Rulebooks can have a similar structure to a Playbook with multiple plays.

Rulebooks: an example

```
---
- name: My ruleset 1
  hosts: all
  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
  rules:
    - name: My remediation rule
      condition: event.outage == true
      action:
        debug:
- name: My ruleset 2
  hosts: all
  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5001
  rules:
    - name: My remediation rule
      condition: event.outage == true
      action:
        debug:
```

Takeaways

Takeaways

- Triggering automation from events can help reduce the outages time
- Event-Driven Ansbile allows to trigger Ansible automation from many different events sources
- Event-Driven Ansible is featureful yet straightforward to implement

Q&A

Let’s continue the conversation:

- Fediverse: @fale@fale.io
- Email: fale@redhat.com

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

twitter.com/RedHat

Red Hat