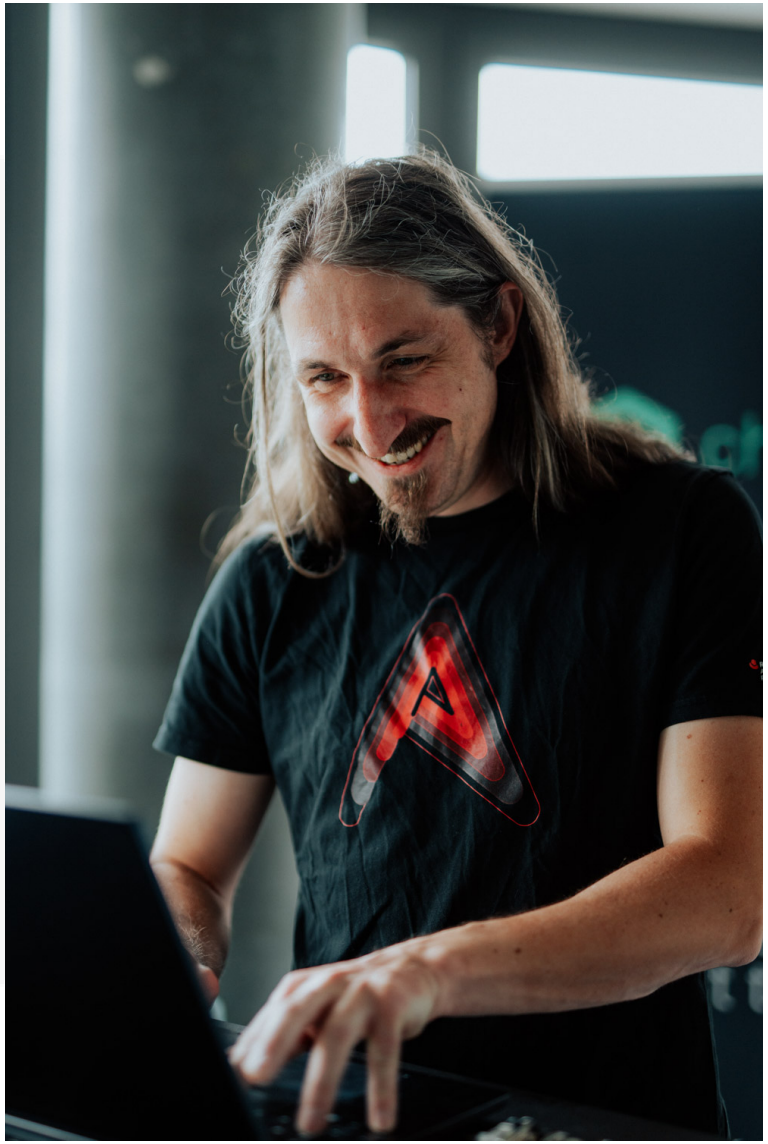
Self-healing with Checkmk and Event-Driven Ansible

How to resolve issues automatically

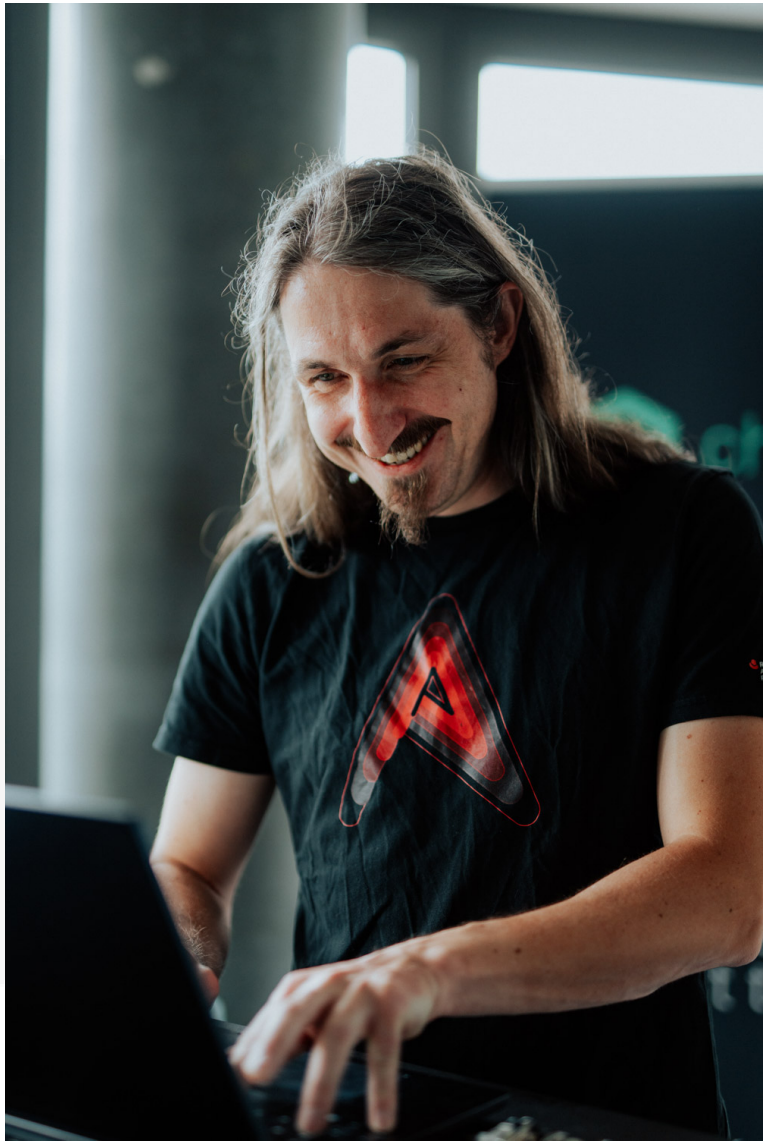
About me

- René Koch
- Self-employed consultant for:
  - Red Hat Ansible (Automation Platform)
  - Red Hat Enterprise Linux
  - Red Hat Satellite
  - Red Hat Identity Management (IPA)
- Experienced monitoring user (Nagios, Icinga, Checkmk)



About me

- René Koch
  - rkoch@rk-it.at
  - +43 660 / 464 0 464
  - https://www.linkedin.com/in/rk-it-at
  - https://github.com/rk-it-at
  - https://github.com/scrat14



Agenda

- Monitoring: Short introduction
- Ansible: Short introduction
- What is Event-Driven Ansible (EDA)?
- Live Demonstration
- Use cases and best practices
- Q&A

Monitoring: Short introduction

Monitoring: Typical workflow

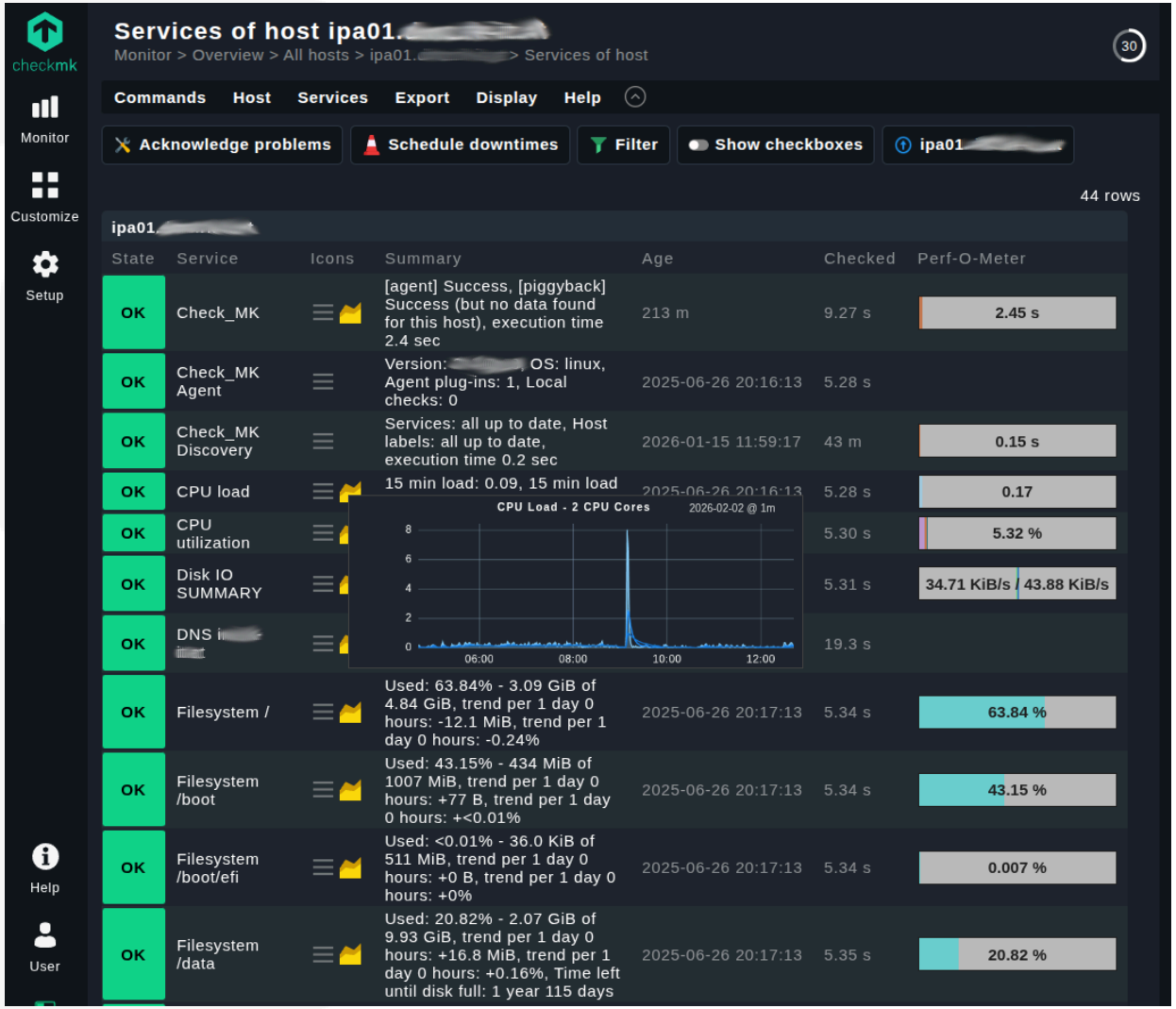
- 🚨 2005: Received email alerts from Nagios 2 for issues with Solaris machines
- 🧩 Manual workflow:
  - 📩 Read email
  - 🔐 Log in to the system
  - 🔎 Check if issue still exists
  - 🛠️ Fix the issue
  - 🤬 **Repeat the same procedure over and over again**
- 🚨 2026: Still the same workflow?

Monitoring: Typical issues

- 💥 DNS server crashes in the middle of the night
- 🐢 Even with redundant DNS servers, systems respond more slowly due to DNS timeouts
- 🔁 Typical workflow:
  - 🔎 Monitoring detects the issue
  - 📟 The on-call engineer is informed
  - 🔐 Log in to the system and restart the service
- 🤔 **Why wake up the on-call engineer instead of fixing it automatically?**

What Is Checkmk?

- **Infrastructure and application monitoring platform**
- Combines data collection, health checks, and alerting in one system
- Supports both on-premises and cloud environments



Checkmk Core Capabilities

- 🧭 **Unified monitoring** for hosts, services, apps, containers and network devices
- 🔌 **Broad data collection** via agent checks, SNMP, APIs and special integrations
- 🚨 **Alerting and event handling** with rules, notifications and escalation paths
- 📊 **Visualization and analytics** through dashboards, BI views and service-level perspectives

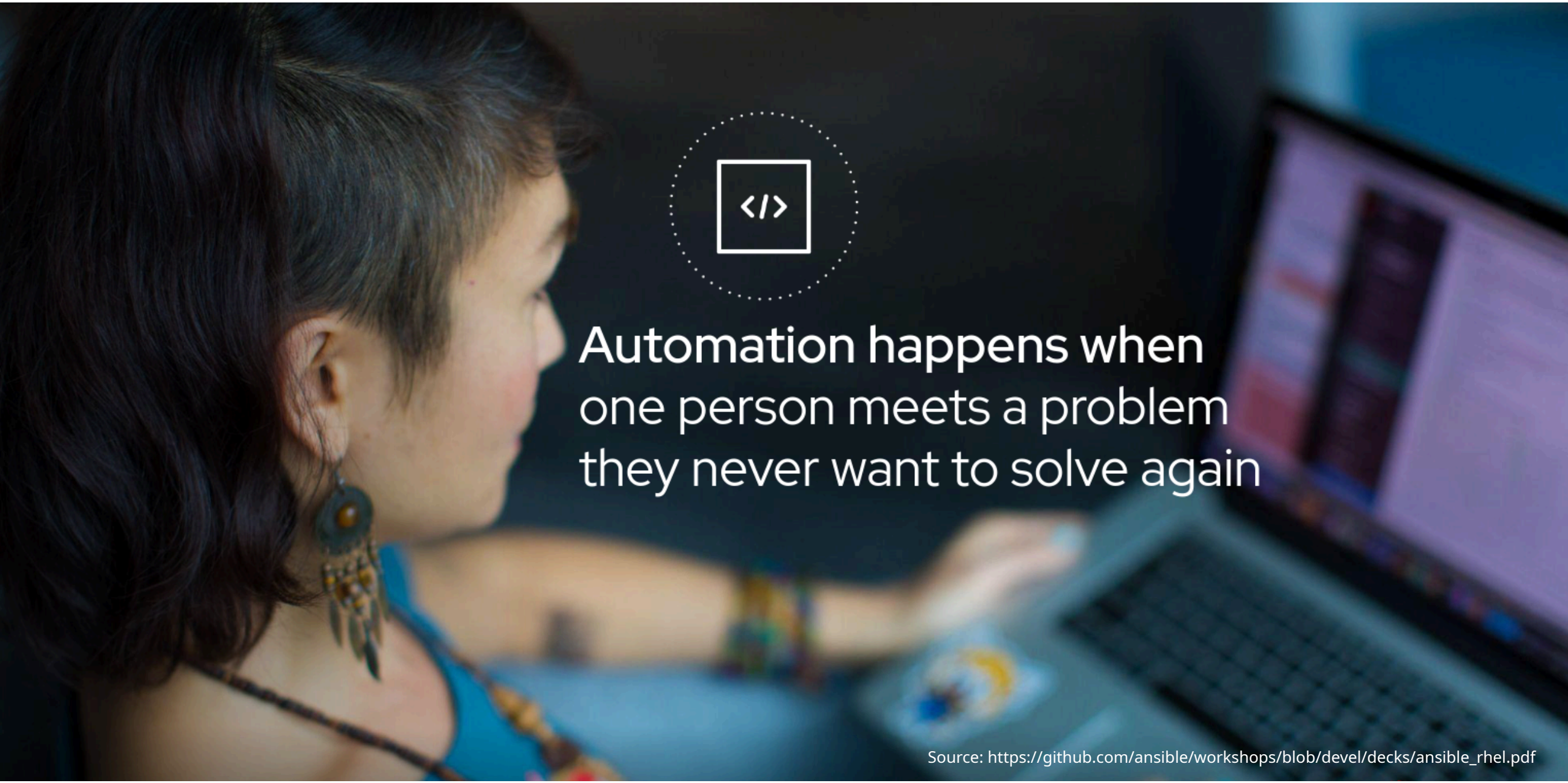
Checkmk Models

- ☁️ **Checkmk SaaS (Cloud)**: hosted by Checkmk; fastest onboarding, less platform maintenance
- 🏙️ **Checkmk Self-Hosted**: self-managed installation; full control over data, upgrades and infrastructure

Checkmk Editions

- 🆓 **Checkmk Community** (formerly Raw): open source edition for smaller environments
- 💼 **Checkmk Pro** (formerly Enterprise): commercial edition with higher performance and advanced features
- 🏭 **Checkmk Ultimate** (formerly Cloud self-hosted): for hybrid and cloud-native infrastructure
- 🏘️ **Checkmk Ultimate with Multi-Tenancy** (formerly MSP): optional add-on for strict tenant separation

Ansible: Short introduction



Source: https://github.com/ansible/workshops/blob/devel/decks/ansible_rhel.pdf

What Is Ansible?

- ⚙️ Automates provisioning, application deployment and configuration management
- 🧩 No agent is required on the target machine
- 🔌 Uses SSH, WinRM and APIs
- 🚀 Executes tasks in parallel on multiple machines
- 🟦 Uses an easy-to-read automation language (YAML)

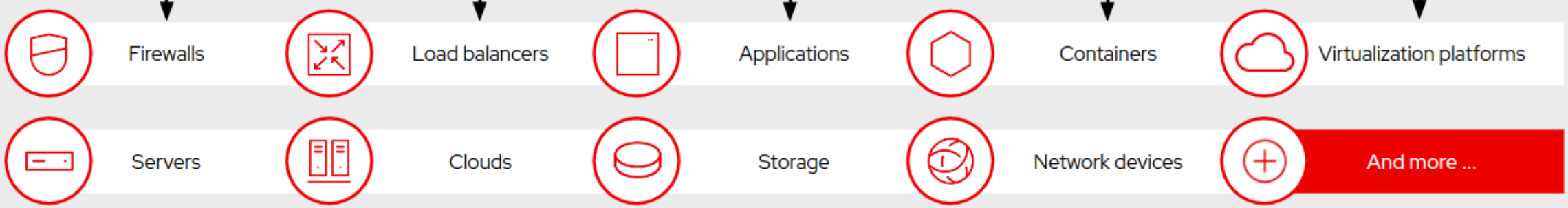
Automate the deployment and management of automation

Your entire IT footprint

Do this...

- Orchestrate
- Manage configurations
- Deploy applications
- Provision / deprovision
- Deliver continuously
- Secure and comply

On these...



Defining community (or free) Ansible, AWX, and Red Hat Ansible Automation Platform

Community Ansible

Free, unsupported open source command line tool for automation.

AWX

Free, unsupported open source software. A GUI and API tool for wrapping around community Ansible.

Red Hat Ansible Automation Platform

Subscription enterprise product. Combines 20+ community projects into a fully supported automation platform for your enterprise.

Ansible Automation Platform

- 🖥️ **Web UI and API** for operating automation from a central interface
- 🧭 **Central control plane** for automation instead of running playbooks from many hosts
- 🔐 **RBAC and credential management** with controlled access to inventories, projects and secrets
- ✅ **Standardized execution** via Execution Environments (same dependencies across teams and stages)
- 🔌 **Integrations** with enterprise tools (for example Splunk, Elastic, ServiceNow)
- 📦 **Automation Hub** for private content and container images

Ansible Automation Platform

- ⚡ **Event-Driven Ansible** included for event-based automation decisions
- 📅 **Operational features**: scheduling, approvals, workflows, notifications and retries
- 📜 **Auditability and governance**: job history, logs and traceable change execution
- 📈 **Scalability**: queueing and distributed execution for larger environments
- 🛠️ **Deployment options**: install on-premises or consume as a managed cloud offering
- 🤝 **Red Hat support** for enterprise operations and troubleshooting

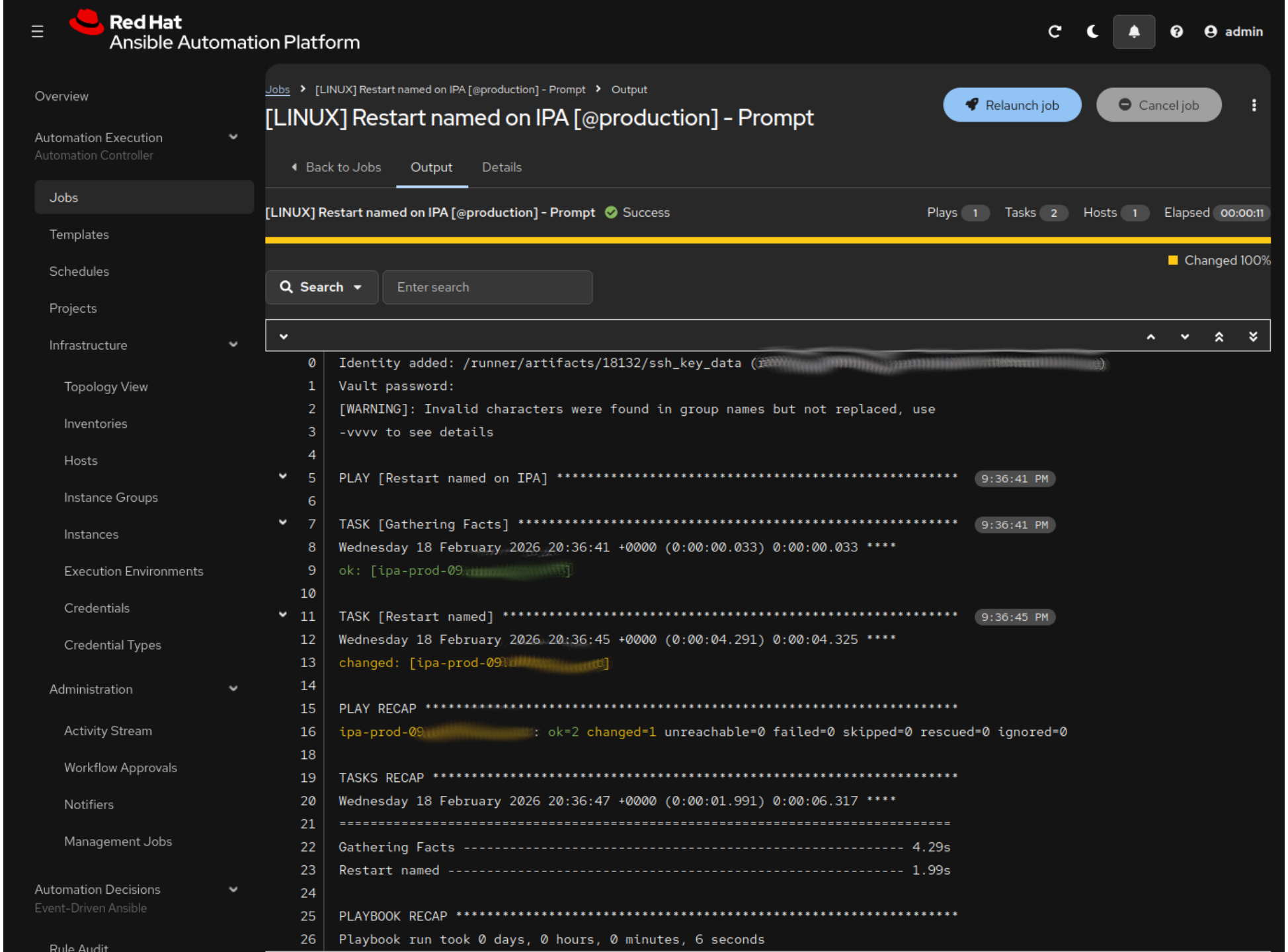
Solve DNS issue with Ansible

- Create a playbook
- Run playbook via
  - Command line
  - Ansible Automation Platform

```
$ ansible-playbook restart_named.yml
```

Solve DNS issue with Ansible

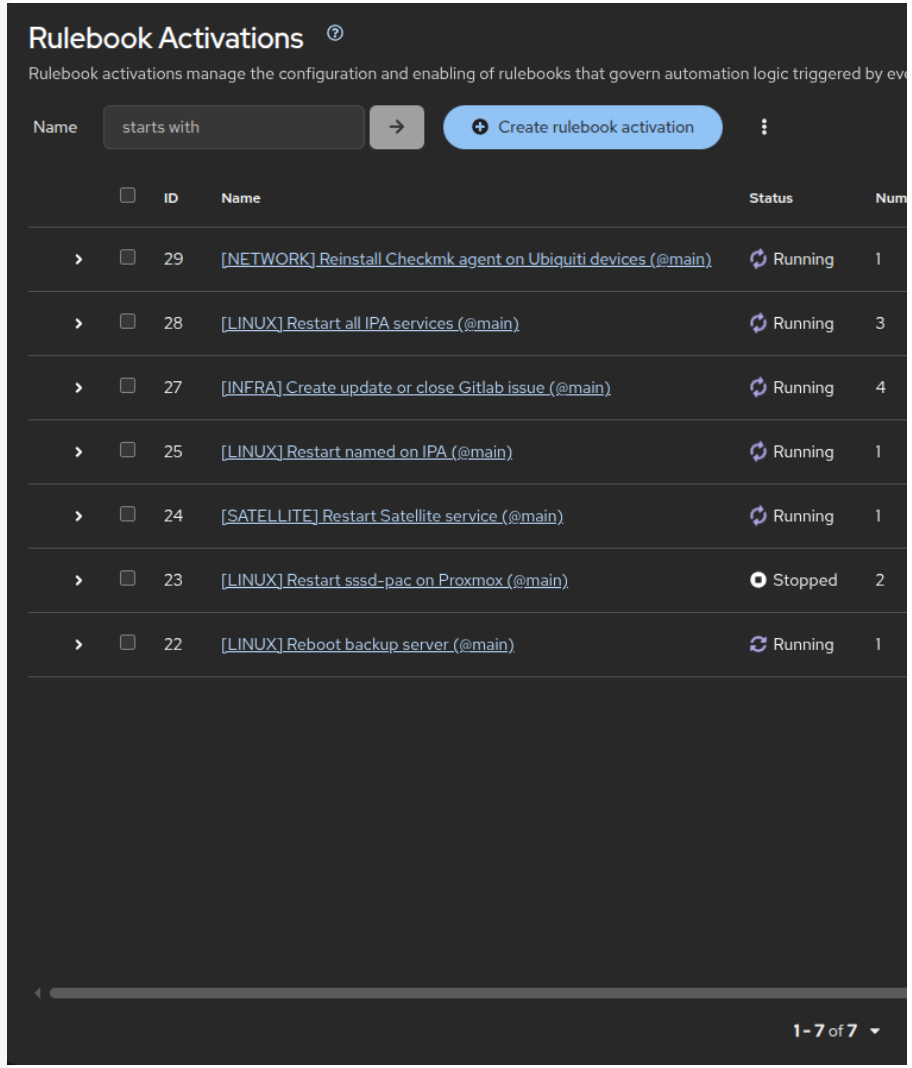
```
---

- name: Restart named on IPA
  hosts: all
  become: true
  gather_facts: true

  tasks:
    - name: Restart named
      ansible.builtin.service:
        name: named
        state: restarted

...
```


Red Hat
Ansible Automation Platform
admin
Overview
Automation Execution
Automation Controller
Jobs
Templates
Schedules
Projects
Infrastructure
Topology View
Inventories
Hosts
Instance Groups
Instances
Execution Environments
Credentials
Credential Types
Administration
Activity Stream
Workflow Approvals
Notifiers
Management Jobs
Automation Decisions
Event-Driven Ansible
Rule Audit
Jobs > [LINUX] Restart named on IPA [@production] - Prompt > Output
[LINUX] Restart named on IPA [@production] - Prompt
Relaunch job
Cancel job
Back to Jobs
Output
Details
[LINUX] Restart named on IPA [@production] - Prompt Success
Plays 1
Tasks 2
Hosts 1
Elapsed 00:00:11
Changed 100%
Search
Enter search
0 Identity added: /runner/artifacts/18132/ssh_key_data ([redacted])
1 Vault password:
2 [WARNING]: Invalid characters were found in group names but not replaced, use
3 -vvvv to see details
4
5 PLAY [Restart named on IPA] ********** 9:36:41 PM
6
7 TASK [Gathering Facts] ********** 9:36:41 PM
8 Wednesday 18 February 2026 20:36:41 +0000 (0:00:00.033) 0:00:00.033 ****
9 ok: [ipa-prod-09[redacted]]
10
11 TASK [Restart named] ********** 9:36:45 PM
12 Wednesday 18 February 2026 20:36:45 +0000 (0:00:04.291) 0:00:04.325 ****
13 changed: [ipa-prod-09[redacted]]
14
15 PLAY RECAP **********
16 ipa-prod-09[redacted] : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
18
19 TASKS RECAP **********
20 Wednesday 18 February 2026 20:36:47 +0000 (0:00:01.991) 0:00:06.317 ****
21 ==========
22 Gathering Facts ---------- 4.29s
23 Restart named ---------- 1.99s
24
25 PLAYBOOK RECAP **********
26 Playbook run took 0 days, 0 hours, 0 minutes, 6 seconds

What Is Event-Driven Ansible (EDA)?

What Is Event-Driven Ansible?

- **EDA is automation that reacts to events**, not schedules
- Events can come from monitoring, webhooks, message queues, logs, or cloud services
- Rules decide **when** to run Ansible actions
- Goal: **faster response** and **consistent remediation**



What Is an "Event" (vs a Source Action)?

- ⚡ **Event**
  - a *state change* or *signal* that matters (e.g., alert fired, service down)
  - often noisy and hard to filter
  - not every event triggers an action
- 🔁 **Source action**
  - a *routine trigger* (e.g., "on every commit")
  - predefined target/action
- 🧪 Examples:
  - 🚫 *Update an AAP project after each commit* (not EDA)
  - ✅ *Send all monitoring alerts to a webhook; EDA decides what to do* (EDA)

Event-Driven Ansible vs. Ansible Playbook

- ⚡ **EDA** keeps a listener running and reacts to events in near real time
- 🧠 **EDA** evaluates event payloads and triggers automation only when rules match
- 📜 **Ansible Playbooks** do not listen for events out of the box
- 🔔 Without EDA, the monitoring source must trigger playbook runs directly
- 🧱 This increases load and complexity on the monitoring system
- 🗂️ This often requires many notification rules in the monitoring system
- 🛑 It can be hard to quickly disable notification rules (for example, across multiple teams)

Event-Driven Ansible vs. AAP Controller

- ⚡ **EDA** keeps a listener running and reacts to events in near real time
- 🌐 **AAP Controller** can start jobs via webhook or API, but each job is a full run lifecycle
- 🐢 Job startup overhead (container start, project sync, inventory/collections) adds latency
- 🚦 Job execution may queue behind other jobs, which delays reaction time
- 🎯 Use EDA for fast event decisions; use Controller for governed execution of the actual remediation

Core Building Blocks

- 📡 **Event Sources**: where events originate (ansible.eda plugins for webhooks, Kafka, Alertmanager, etc.)
- 📘 **Rulebook**: Rule sets with conditions + actions
- 🔎 **Conditions**: Determine if a rule fires
- 🛠️ **Actions**: run playbooks, run job templates, run modules, etc.
- 🧭 **Controller** (optional): central execution and governance

ansible.eda: Supported Event Sources

- 📥 `alertmanager` : receive alerts via Alertmanager webhooks
- ☁️ `aws_cloudtrail` : ingest AWS CloudTrail events
- 📬 `aws_sqs_queue` : consume messages from an AWS SQS queue
- ☁️ `azure_service_bus` : receive events from Azure Service Bus
- 📂 `file` : load events from files
- 👀 `file_watch` : watch files and emit events on changes
- 🧪 `generic` : generate test events from static payload data

ansible.eda: Supported Event Sources

- 📝 `journald` : read events from systemd journal logs
- 📨 `kafka` : consume events from Kafka topics
- 🐘 `pg_listener` : listen for PostgreSQL `NOTIFY` events
- 🔢 `range` : generate a finite sequence of test events
- ⏱️ `tick` : generate periodic timer events
- 🌐 `url_check` : poll URLs and emit status change events
- 🪝 `webhook` : receive events via HTTP webhook
- ℹ️ As of `ansible.eda` collection version `2.11.0`

Common EDA Actions

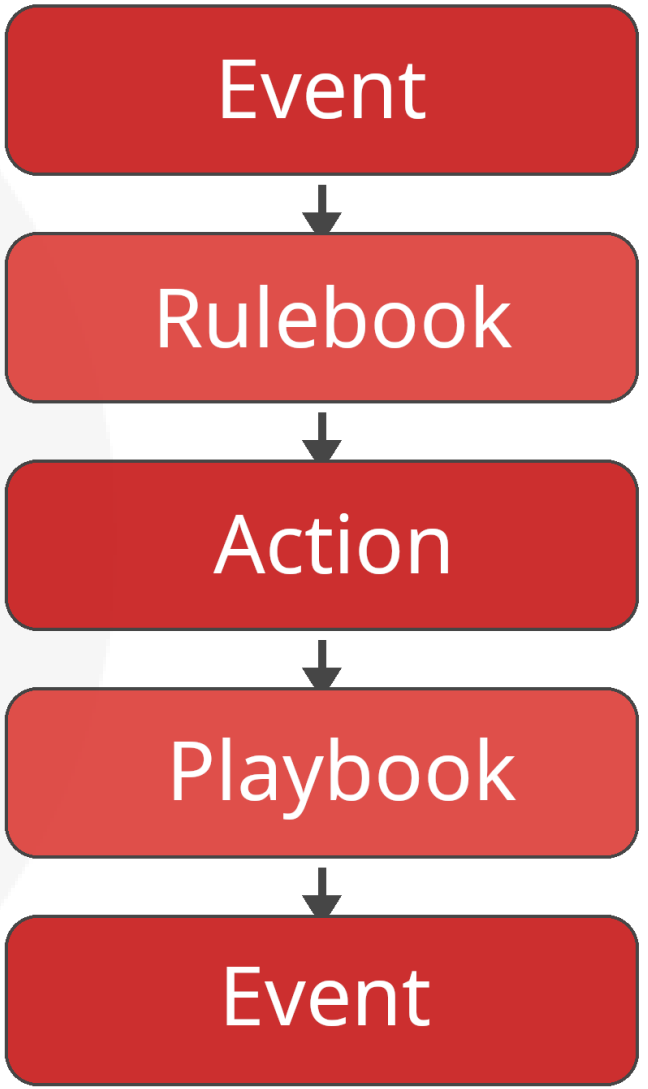
- ▶️ `run_playbook` : run an Ansible playbook directly
- 🧱 `run_module` : execute a specific Ansible module
- 🚀 `run_job_template` : start an AAP Controller job template
- 🛤️ `run_workflow_template` : start an AAP workflow template
- 📣 `post_event` : emit a new event for follow-up processing

Common EDA Actions

- 🧹 `retract_fact` : remove facts previously added to event context
- 🔔 `print_event` : write the event payload to output/logs (useful for debugging)
- 🧩 `set_fact` : add or modify event data for later conditions/actions
- 🔲 `shutdown` : stop rulebook execution
- 🐞 `debug` : print debugging information for rule/action evaluation
- ⛔ `none` : intentionally do nothing (for matched-but-ignored cases)

Event-Driven Workflow

1. Event arrives from a source
2. Rulebook evaluates conditions
3. Matching rule triggers an action
4. Action runs playbook or other automation
5. Results can emit **new events** or update systems



Rulebook: Restart named

```
---

- name: Restart named on IPA server
  hosts: all
  gather_facts: false

  sources:
    - name: Listen on port 5000 for Checkmk events
      ansible.eda.webhook:
        port: 5000

  rules:
    - name: Restart named
      condition: >-
        event['payload']['servicename'] == "DNS example.com" and
        event['payload']['servicestate'] == "CRITICAL"
      action:
        run_job_template:
          name: "[LINUX] Restart named on IPA [@production] - Prompt"
          organization: "Default Organization"
          job_args:
            limit: "{{ event.payload.hostname }}"
```

Checkmk Notification Script

```
#!/usr/bin/env bash

HEADER="X-Checkmk-Token"
TOKEN="${NOTIFY_PARAMETER_1}"
URL="${NOTIFY_PARAMETER_2}"

JSON=`cat <<EOF
{
  "hostname": "${NOTIFY_HOSTNAME}",
  "hostoutput": "${NOTIFY_HOSTOUTPUT}",
  "hoststate": "${NOTIFY_HOSTSTATE}",
  "servicename": "${NOTIFY_SERVICEDESC}",
  "serviceoutput": "${NOTIFY_SERVICEOUTPUT}",
  "servicestate": "${NOTIFY_SERVICESTATE}",
  "date": "${NOTIFY_SHORTDATETIME}",
  "type": "${NOTIFY_NOTIFICATIONTYPE}",
  "what": "${NOTIFY_WHAT}"
}
EOF
`

curl -X POST -H "Content-Type: application/json" -H "${HEADER}: ${TOKEN}" -d "${JSON}" ${URL}
exit $?
```

Playbook: Restart named

```
---

- name: Restart named on IPA
  hosts: all
  become: true
  gather_facts: true

  tasks:
    - name: Restart named
      ansible.builtin.service:
        name: named
        state: restarted
```

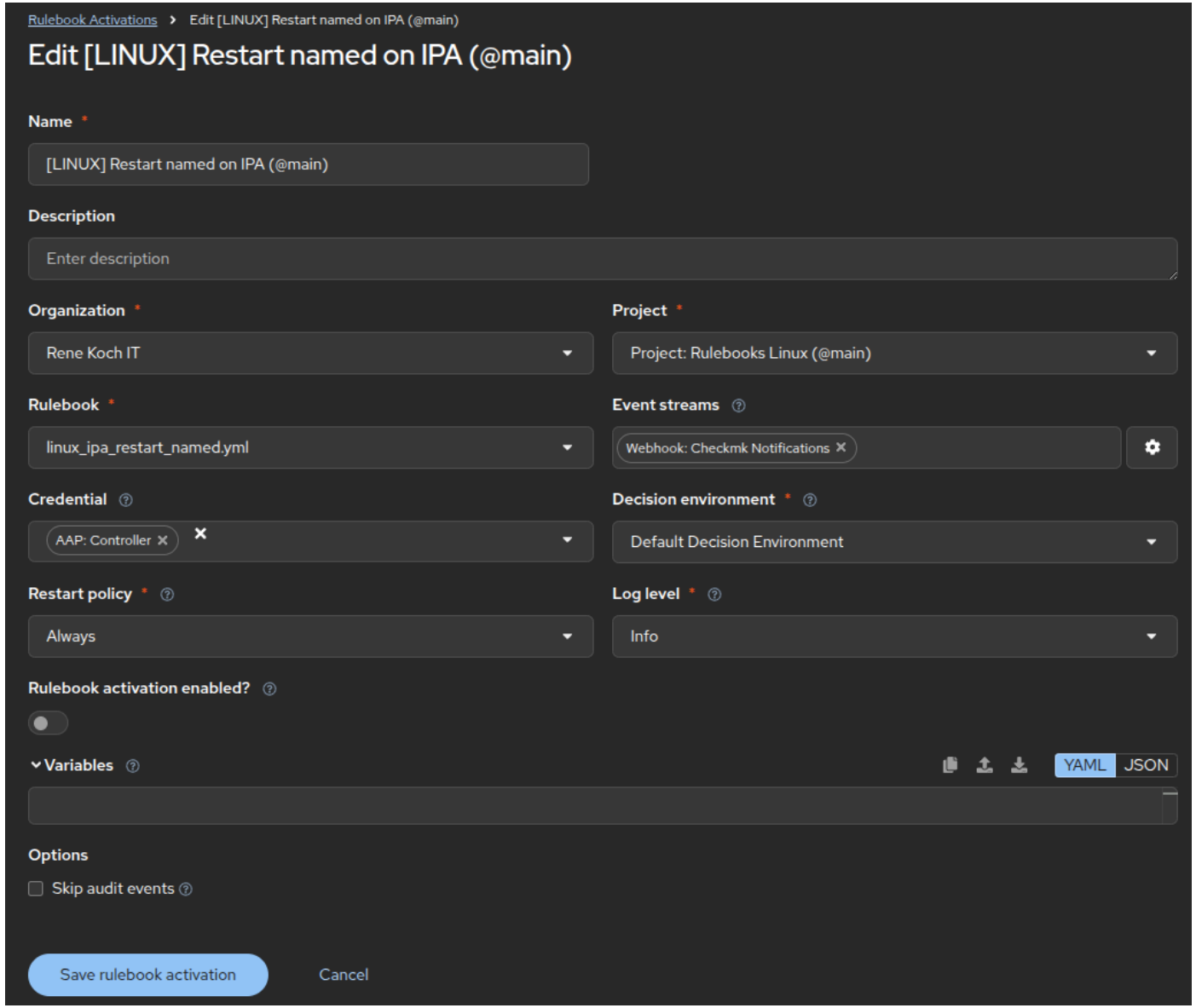
Run a Rulebook (CLI)

```
$ ansible-rulebook -r rulebooks/restart_named.yml -i localhost

PLAY [Restart named on IPA] ****************************************************

TASK [Gathering Facts] *********************************************************
ok: [ipa01.example.com]

TASK [Restart named] ***********************************************************
changed: [ipa01.example.com]

PLAY RECAP *********************************************************************
ipa01.example.com : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

❗ Use the **run_playbook** action (instead of **run_job_template**) when running with `ansible-playbook` outside Ansible Automation Platform.

Ansible Automation Platform Integration

- 📦 **Projects**: Git repository configuration
- 🧪 **Decision Environments**: Container images to run rulebooks
- 🔐 **Credentials**: Secrets for Git, Controller, Hub, tokens, etc.
- 📡 **Event Streams**: Entry points for events (mapped to source definition in rulebook)
- 🚀 **Rulebook Activations**: Rulebook runs

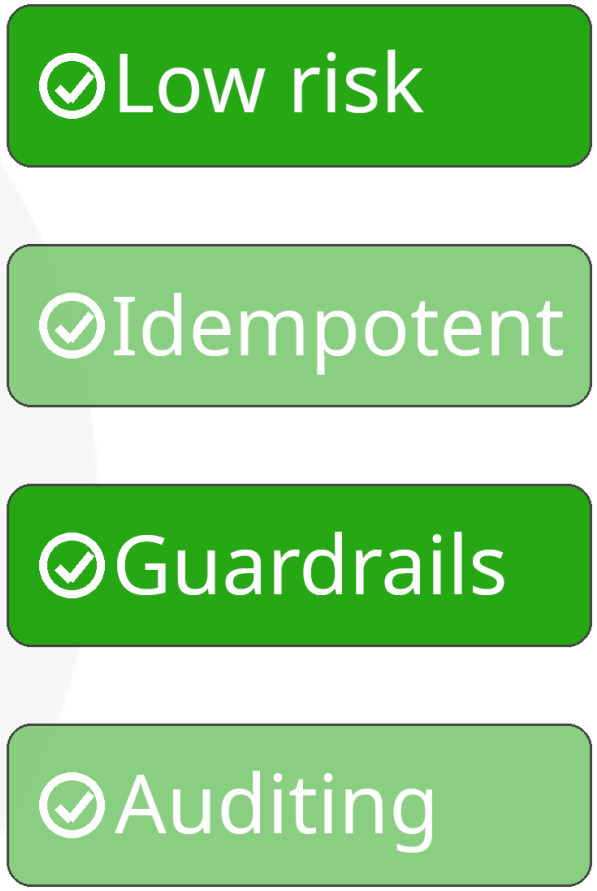

Rulebook Activations
Edit [LINUX] Restart named on IPA (@main)
Edit [LINUX] Restart named on IPA (@main)
Name *
[LINUX] Restart named on IPA (@main)
Description
Enter description
Organization *
Rene Koch IT
Project *
Project: Rulebooks Linux (@main)
Rulebook *
linux_ipa_restart_named.yml
Event streams
Webhook: Checkmk Notifications
Credential
AAP: Controller
Decision environment *
Default Decision Environment
Restart policy *
Always
Log level *
Info
Rulebook activation enabled?
Variables
YAML
JSON
Options
Skip audit events
Save rulebook activation
Cancel

Live Demo: Fix DNS issue

Use cases and best practices

Self-Healing Best Practices

- Start with **low-risk** automations
- Use **idempotent** playbooks (if possible)
- Add **guardrails** (approvals, maintenance windows, downtimes)
- Emit **metrics and logs** for auditing



Challenges with Self-healing

- 🔊 Triggering on noisy events (missing filtering)
- 🧪 Insufficient monitoring coverage
- 🧯 Healing the wrong host (issue caused by a backend dependency)
- 📚 Lack of knowledge or rulebooks
- 🕒 Triggering during maintenance windows due to missing downtime
- 😟 **Fear**

Event-Driven Ansible Use Cases

- 🚨 **Monitoring alerts**: run remediation playbooks
- 🏗️ **Infrastructure events**: auto-scale or restart services
- 🔐 **Security findings**: isolate hosts or rotate credentials
- 🎟️ **Ticketing**: enrich and open incidents automatically
- 📘 **Documentation**: update asset database or documentation system

Other Event Sources for EDA

- 📈 **Zabbix**: trigger EDA from problem/recovery events
- 🛡️ **Wazuh**: trigger EDA from security detections and compliance alerts
- 📊 **Prometheus + Alertmanager**: trigger EDA from metric-based alert rules
- 🔍 **Elastic (Elasticsearch/Kibana)**: trigger EDA from log and SIEM detections
- 🌐 **Switch SNMP traps**: trigger EDA to update NetBox documentation automatically
- ☁️ **Cloud-native services**: trigger EDA from event buses and webhook integrations

Additional Information

- **Products**:
  - Ansible Automation Platform: https://urlr.me/WcXZgR
  - Checkmk: https://urlr.me/BPJs98
- **Product Documentation**:
  - Ansible: https://urlr.me/2x4HfW
  - Automation Decisions: https://urlr.me/HejEDB
  - Rulebooks: https://urlr.me/Qb4TsB

Summary

- 🛰️ Checkmk **monitors** your IT landscape and **notifies** on state change
- ⚡ EDA turns these events into **real-time automation**
- 🤝 It complements traditional Ansible by **reacting** instead of **scheduling**
- 🎯 Start small, measure impact and iterate

Thank you!

René Koch

Freelancer

Secure Linux Administration Conference 11.05.2026

Slides: https://urlr.me/Q5MSWy