

WHITEPAPER · OCTOBER 2026

Network Automation Without Python

Rehearse every change. Prove every change.

How a visual canvas, a lab cloned from your own topology, an on-premises execution agent and signed compliance evidence turn a network change, its rehearsal and its proof into one artifact. Written for the network engineer who wants the canvas and the security approver who has to sign off.

Regnor™



The platform. Design on a canvas of 60+ node types; no Python required.

Valdis™



The agent on your network. Runs every change; the cloud never holds a plaintext credential.

Tavrin™



The compliance engine. 18 DISA STIG packs, signed append-only evidence.

CONTENTS

Executive summary	2	Tavrin compliance: rule packs, remediation, sealed evidence	8
The problem	3	The security model, and what we do not claim	9
The approach: four stages, one loop back	4	A worked example: 14 switches, one NTP change	11
Architecture: cloud design plane, on-premises execution plane	5	How to start	12
Lab cloning	7		

Executive summary

Automate your network without writing Python.

Regnor™ by AutomateNetOps is a visual network automation platform. You build a change as a workflow on a canvas of more than 60 node types (63 today, growing with each release) and rehearse it against a lab cloned from your own topology. The Valdis™ agent runs it on your network, and the platform never holds a device credential in plaintext. The Tavrin™ compliance engine records the outcome as signed, append-only evidence.



A visual canvas

60+ node types in 10 categories: Netmiko, NAPALM, Scrapli, NETCONF, RESTCONF, gNMI, Ansible, Terraform, approval gates, asserts, AI. A workflow in minutes, not a Python project.

2

Rehearse before you ship

Lasso the devices on the topology map, clone them into a lab on your own agent host, and run the same workflow there first.



Security says yes

Device credentials live only in the Vault on your host (Valdis). Every scan leaves a signed, append-only evidence row your auditor can verify with openssl (Tavrin).

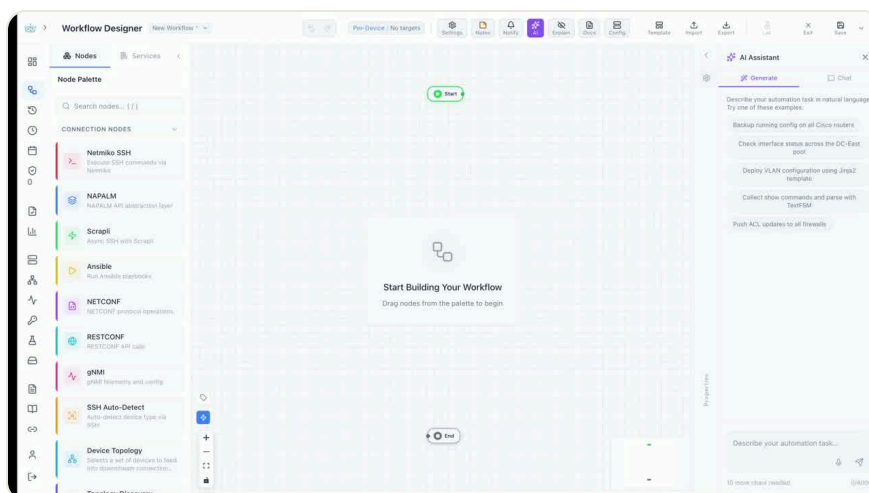
If you are the network engineer

You want the canvas. Sections 2, 4 and 7 show how the pieces work: the approach, lab cloning, and one change taken end to end.

If you are the security or compliance approver

You have to sign off. Sections 3, 5 and 6 state what the architecture does and does not guarantee, including what we do not claim.

The technical claims in this paper were checked against agent release 1.8.261 in October 2026. Where a capability is partial, the paper says so.



The Workflow Designer: an AI-drafted workflow of 11 nodes, no Python written.

1 The problem

Most network changes still go in by hand, one SSH session at a time, and the record of what changed is typed into a ticket afterwards. Three failures follow from that, and they share one root cause.

1 Manual changes

A change window is a maintenance night, a terminal, and a text file of commands pasted device by device. Nobody rehearsed the exact change anywhere, so production is the first real test. When something breaks, the rollback is also typed by hand, under pressure.

2 Scripts only one engineer understands

Teams that do automate usually do it with Python, Netmiko or Ansible written by one motivated engineer. The script works, so it spreads; then that engineer changes roles, and nobody else can read, run or trust it. Automation that one person can maintain is a risk the organization has quietly taken on, not a capability it owns.

3 Audit evidence built by hand

When the auditor asks how a change was approved, what it did, and whether the device is still compliant, the answer is assembled after the fact: screenshots, pasted show output, a spreadsheet of STIG checks ticked by eye. That evidence proves someone did paperwork. It does not prove what ran.

The security and compliance approver sees the same three problems from the other side: credentials sitting in scripts and spreadsheets, change procedures that cannot be reviewed before they run, and evidence that cannot be verified after.

The root cause is the same in every case: the change, its rehearsal and its proof are three separate artifacts produced by three separate manual efforts. Regnor's premise is that they should be one.

TODAY · THREE ARTIFACTS, THREE MANUAL EFFORTS

The change

commands pasted device by device,
rollback typed under pressure

The rehearsal

none, or a one-off lab
that is not your topology

The proof

screenshots and a spreadsheet,
assembled after the fact



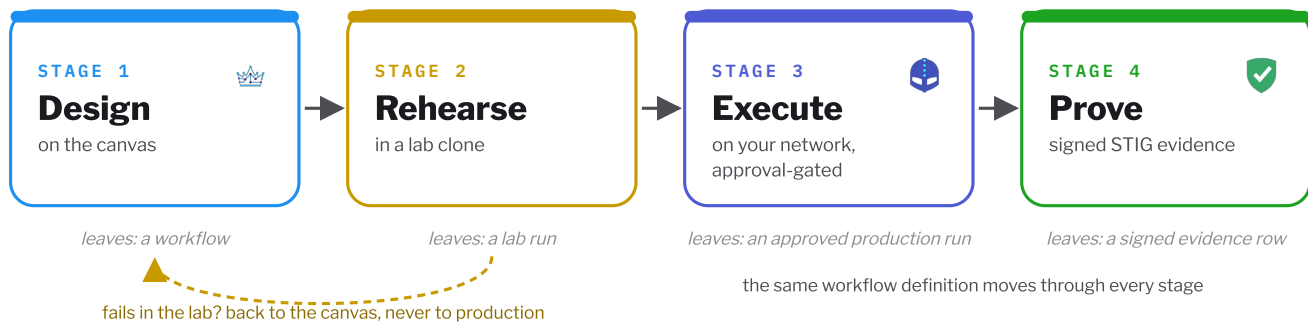
One artifact: a workflow that is designed, rehearsed, approved and proven

the same definition moves from the canvas to the lab clone to the approved production run to the signed evidence row

Today the change, its rehearsal and its proof are three artifacts, each assembled by hand. Regnor makes them one workflow that is designed, rehearsed, approved and proven.

2 The approach

Regnor treats a change as one artifact that passes through four stages, and each stage leaves something the next can use: a workflow, a lab run, an approved production run, and signed compliance evidence.



Four stages, one loop back. Each stage hands the next the same workflow definition. A change that fails in the lab sends the engineer back to the canvas, not to production.

Design on the canvas. The Workflow Designer is a drag-and-drop canvas with more than 60 node types in 10 categories. Connection nodes cover Netmiko, NAPALM, Scrapli, NETCONF, RESTCONF, gNMI, Ansible and Terraform. Flow-and-control nodes provide approval gates, conditions, iteration, try/catch, assert and wait-until. The rest are processing, data, ITSM, event, compliance and AI nodes. A device change is a form: pick the connection library and platform, type the commands, toggle configuration mode. The AI assistant can draft a workflow from a plain-English description, and a Python node exists for teams that want one; nothing requires it.

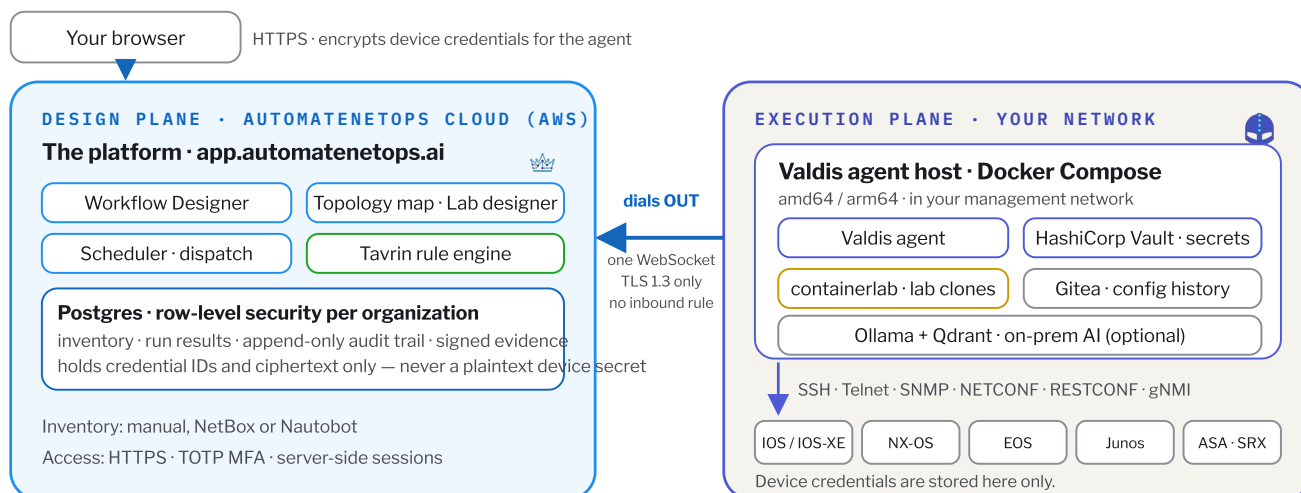
Rehearse in a lab clone. Select the devices on the topology map and clone them into a lab on your own agent host, then run the same workflow against the clone from the canvas with Test in Lab. Section 4 covers what the clone carries and what it does not.

Execute on your network; the platform never holds a device credential in plaintext. Cloud-side device execution is disabled platform-wide: the list of node types allowed to run in the cloud is empty, and every device-touching node is dispatched to the Valdis agent inside your network. A task reaches the agent carrying credential IDs, and the agent resolves the secrets from its local Vault. Regnor calls this its zero-trust credential model and means it narrowly: the platform is never trusted with a plaintext device secret, so a breach of the platform's database exposes no device credential. Approval gates park a run until a user with the right permission approves it, and the parked state survives agent restarts and updates. Backups, diffs, asserts and rollback are nodes you place on the canvas, and dry run returns simulated results without opening a device session.

Prove it. Tavrion evaluates the fleet against DISA STIG rule packs on your schedule and seals each scan's summary as a signed, append-only evidence row. Section 5 covers what is sealed, what is not, and how an auditor checks it.

3 Architecture

Regnor is two planes. The design plane, called the platform in this paper, runs in the AutomateNetOps cloud on AWS. The execution plane is the Valdis agent you run on your own network, and it only ever dials out.



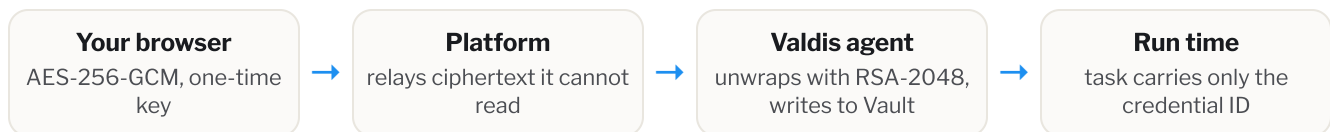
The platform never opens a connection into your network. Tasks carry credential IDs; the agent resolves the secret from its own Vault.

The browser talks to the platform; the agent talks to the platform; the platform never opens a connection into your network. Device secrets are stored in one place, the HashiCorp Vault on your agent host.

The platform holds the Workflow Designer, topology map and lab designer, the scheduler and task dispatch, the Tavrin rule engine, and a Postgres database with row-level security, meaning the database itself filters every query by organization. The database stores inventory, run results, the audit trail and signed evidence. You reach the platform over HTTPS at app.automatenetops.ai. Inventory comes from manual entry or from a NetBox or Nautobot source of truth.

The Valdis agent is a Docker Compose stack you install on a host inside your network with a one-time install code; images are built for linux/amd64 and linux/arm64. It opens a single long-lived WebSocket (a persistent, encrypted two-way connection) to the platform over TLS 1.3, refusing older versions, and makes outbound HTTPS calls for activation and updates. No inbound firewall rule is needed. The stack also runs a HashiCorp Vault sidecar (a companion container) for device credentials, an optional Gitea sidecar (a self-hosted Git server) for versioned configuration backups, containerlab (an open-source tool that runs network operating systems as containers) for lab clones, and optional Ollama and Qdrant so that AI features can run entirely on your premises. From that host the agent reaches your devices over SSH, Telnet for legacy devices, SNMP, NETCONF, RESTCONF and gNMI, so the host needs management-plane reachability. Place it in your management network. It listens on the LAN for its own health endpoint, Vault, Gitea and a peer cache that other agents in your organization fetch from over TLS with a token; none of these should be exposed beyond the management network. Vault is unsealed automatically at start from a key kept on the host's own volume, so protecting that host is part of your side of the model (section 6).

Credentials never sit in the platform. Device credential types (SSH, Telnet, SNMP) can only be stored on-premises; the platform refuses to create one in cloud storage. When you enter a credential in the UI, your browser generates a one-time AES-256-GCM key, encrypts the credential with it, and wraps that key to the agent's RSA-2048 public key with RSA-OAEP. The platform relays ciphertext it cannot read, and the agent decrypts it straight into its Vault. A task sent to the agent carries only the credential ID and vault path; the agent resolves the secret locally at run time. Because the platform never holds a plaintext device credential, no Regnor API can return one. This protects device credentials against a breach of the platform's database or storage. It does not, on its own, defend against a hostile platform operator, because the agent's public key is distributed through the platform.



What does travel. Node output, including the output of a `show running-config` if a workflow asks for it, console sessions you open, and backup content you ask to view are returned through the platform over the same encrypted channel so you can read them in the UI. Integration secrets you choose to keep in the platform (ITSM and S3 API keys, AI provider keys) are encrypted at rest and sent to the agent over the TLS channel only when a node needs them. AI nodes run on the agent and call the provider you configure (Claude, OpenAI, Grok, or a local Ollama); with Ollama and Qdrant, prompts and documents never leave your premises.

Enrollment and identity. An administrator generates an install code, valid for 24 hours by default. The agent activates with it over HTTPS and receives an agent API key that is bcrypt-hashed in the database and stored on the host with owner-only file permissions. Every connection re-authenticates with that key and the agent ID. Revoked or decommissioned agents and suspended organizations are refused. Each agent belongs to exactly one organization and receives tasks only for that organization, routed by site.

Configuration history. When the Gitea sidecar is enabled, one agent per organization hosts it. Every configuration backup is committed there, and the platform keeps only the metadata (hash, size, diff summary, git SHA), fetching the content on demand when you open it.

STAYS ON YOUR SIDE

- Device credentials, in the Vault on your agent host
- The Vault unseal key and the agent host itself
- Lab clones on containerlab
- Configuration history in the Gitea sidecar
- Prompts and documents, when AI runs on Ollama and Qdrant

TRAVELS TO THE PLATFORM, OVER TLS

- Credential ciphertext the platform cannot read
- Node output you asked for, console sessions, backups you open
- Run results, the audit trail, signed evidence
- Integration secrets you chose to keep in the platform, only when a node needs them
- Backup metadata: hash, size, diff summary, git SHA

4 Lab cloning

Regnor's topology map lets you select the devices a change will touch and clone them into a lab running on your own Valdis agent. The first run of a change happens on a lab built from your topology, not on production.

From map to lab in one gesture. The topology map is built by the agent itself. It runs LLDP, CDP and ARP discovery on Cisco IOS, IOS-XE and NX-OS, Arista EOS and Juniper Junos, and you can add links by hand or import a topology from CSV or OpenConfig YAML. Lasso-select the devices on the canvas, or right-click one, and choose Clone to Lab. With nothing selected, the button offers to clone every visible device. A review step, the Fidelity Advisor, lets you name the lab and drop devices before you click Materialize lab. A clone takes up to 100 devices and needs the lab-create permission and an online agent with containerlab.

What the clone carries. Each device's hostname, vendor, model and OS version are mapped to a lab image, and the discovered links are rebuilt between the lab nodes. The clone is a topology, not a configuration. Running configs are not copied automatically, and interface names containerlab cannot represent, such as GigabitEthernet0/1, become eth1, eth2 and so on. You can attach a startup configuration to any lab node in the Lab Designer before you deploy.

Images. The lab runs on containerlab on the agent host. Cisco images run through vrnetlab, which packages VM images such as IOSv so containerlab can run them, from images you upload. Arista cEOS, Nokia SR Linux, Juniper cRPD and Linux nodes run as containers.

LAB NODE KIND	VENDOR	RUNS AS
IOL, IOSv, CSR1000v, Catalyst 9000v and 8000v, XRV9000, Nexus 9000v, FTDv	Cisco	vrnetlab, image uploaded by you
cEOS	Arista	Container
SR Linux	Nokia	Container
cRPD	Juniper	Container
Linux	Any	Container

Test in Lab from the canvas. In the Workflow Designer, Test in Lab runs the open workflow against an existing lab, such as the clone you just made, or against a lab auto-provisioned from the device types the workflow references, with optional tear-down afterwards. Regnor provisions the lab, deploys it, waits for SSH, runs the workflow, and shows the same results view you will see in production. The rehearsal uses the same workflow, nodes and agent. The lab image is not your hardware, so the clone catches logic, ordering and syntax errors; hardware-specific behavior still deserves a production canary.

5 TavrIn compliance

An auditor asks two things of a compliance tool: did you scan everything, and can I check the answer without you in the room. TavrIn™, the compliance engine inside Regnor, is built to answer both.

It evaluates device configurations against DISA STIG rule packs on a schedule you set, records how many in-scope devices each run covered, and seals each scan's summary as a signed, append-only evidence row that your auditor can verify with openssl and no Regnor account.

Rule packs. Regnor ships 18 DISA STIG packs with 759 rules. Each rule carries its DISA V-ID, rule ID and CCI references, and severity maps to CAT I, II and III. Rules that only a person can verify are marked "needs review" instead of a false pass and export to STIG Viewer as Not Reviewed; the agent-side half of that behavior ships in the next agent release. You can also author your own rules (seven rule types, including regex and hierarchical match), test them against a device, and import rule templates.

PACK FAMILY	PLATFORMS	PACKS	RULES	SOURCE
DISA STIG, Cisco	IOS and IOS-XE router and switch, NX-OS, ASA	11	429	Official DISA XCCDF benchmarks, source hash recorded
DISA STIG, Juniper	SRX, EX	7	330	DISA XCCDF (EX) and MITRE InSpec baselines (SRX)
Custom	Any platform Regnor connects to	Yours	Yours	Authored in the Golden Configs page

Auto-dispatched validation. A compliance scan is a schedule (cron, interval, or one-time) bound to your organization's compliance workflow. At run time the platform resolves which profiles, that is rule sets, apply to which devices; the Valdis agent on your network evaluates them locally and returns per-rule results. Every scheduled run also writes a signed coverage-manifest record stating how many of the in-scope devices were actually scanned, so a partial run cannot pass as a complete one. "Run STIG Audit Now" does the same on demand.

Remediation with a safety net. For failing rules whose fix is concrete CLI (130 STIG rules today), TavrIn proposes a remediation. It never applies anything on its own: the run parks at an approval gate until a user who holds the workflow-execute permission approves it. On approval the agent backs up the configuration, shows the blast-radius diff (every line the fix would change), applies the fix inside a self-reverting window (IOS `configure terminal revert timer`, Junos `commit confirmed`), and confirms only if a one-rule re-check passes. Otherwise the device reverts. A signed remediation-closure record then links the finding, the approver and the passing re-check. Launches are limited to one vendor and 25 devices at a time, and an organization can require a second approver.

Sealed evidence. Each scan writes one row to an evidence table that database triggers hold append-only. The row carries the counts, the score, the failed V-IDs, the SHA-256 of the full result set, the rule-set hash and the device-config hash, and it is signed with an Ed25519 key. The per-rule detail rows are not themselves sealed; the hash in the summary row makes a later change to them detectable, not impossible. The trigger refuses any update and refuses to delete a row younger than 90 days. A retention job removes rows older than the configured period, seven years by default. Once a day an anchor hash over the previous day's evidence is written to the audit trail, in the platform's own database rather than an external notary. The seal is enforced in the database and by signature, not by a storage-level object lock.

✓ **One signed evidence row, per scan**

counts · score

failed V-IDs

SHA-256 of the full result set

rule-set hash

device-config hash

coverage: scanned / in scope

Ed25519 signature · key ID = SHA-256 of the public key

Append-only by database trigger · daily anchor hash into the audit trail · verifiable with openssl and no Regnor account

The append-only guarantee is ours to operate; the signature is yours to check.

Export for auditors. The signed-verification bundle ships each evidence payload with its `.sig` file, the public keys, and a `VERIFY.md` with the openssl steps. Verification proves each row is the one Regnor signed and that it has not changed since. The key ID is the SHA-256 of the public key, so an auditor can pin it, and the keys are also served by the platform without login. Reports export as PDF (with an evidence-integrity section and classification banner), CSV, executive summary, and STIG Viewer `.ck1b` and `.ck1`. Export needs the compliance export permission, and every download is itself audited.

6 The security model

The security model rests on three controls an approver can check: tenants are isolated in the database itself, the agent self-updates only to releases the platform signed, and the actions listed below land in an append-only audit trail.

Tenant isolation. Postgres row-level security is enabled and forced on every organization-scoped table. The application connects as a role that owns no tables and cannot bypass the policies, and the organization context is set per query from the authenticated session. Platform administration by AutomateNetOps staff uses a separate role that bypasses row-level security. That is the operator access an approver should assume exists in any SaaS, and it is one reason device credentials are kept out of the platform entirely. Agents, devices, device pools, credentials and workflows all carry an organization ID, and there is no organization switcher. Within a tenant, role-based access control provides six system roles plus custom roles. Device access is granted per pool with separate view, target, edit, console and configuration rights; organization administrators bypass pool permissions entirely.

Signed agent releases. Each release is registered with its image digest (the image's SHA-256 fingerprint) and an Ed25519 signature over the version and digest, made with a release key held by AutomateNetOps. The platform refuses to register a release without one. Before self-updating, the agent pulls the image, checks that the digest matches the registered one exactly, verifies the signature against a public key built into the running image, and refuses the update on any mismatch. The initial install is verified by digest pin only; signature verification begins with the first self-update. Releases are built for amd64 and arm64 and rolled out from a platform fleet console, and an organization watches its own agents on its Agent Dashboard.

Audit trail. Authentication events, credential operations, agent lifecycle events, approvals, compliance actions and evidence exports are written to an audit table with actor, organization, action type, resource, source IP, user agent, outcome and timestamp. Roughly 400 action types are allow-listed in the schema itself, so a new action type must be added by migration before code can log it. A database trigger blocks update and delete on the table. The trigger binds the application role, which owns no tables; the database owner role, held by the platform operator and never by the application, could remove it. That is why the evidence you can hold yourself, the signed bundle, matters more than any promise about our database. The daily evidence anchor (section 5) is written into the same trail, and the trail exports as CSV, JSON or PDF to holders of the audit export permission.

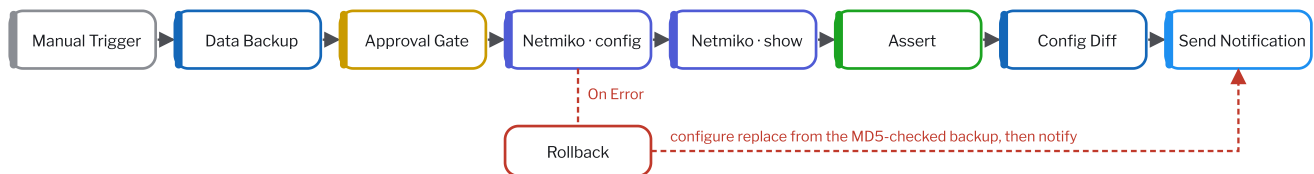
Accounts and sessions. TOTP multi-factor authentication stores its secrets encrypted and issues hashed backup codes. Enforcement is on for administrators by default; enforcement for every user is a platform-wide setting. Sessions are server-side records with sliding expiry capped at seven days, and revocation takes effect on live connections. API keys are bcrypt-hashed, support IP allow-lists and expiry, and logins are rate-limited. Single sign-on is not yet available.

Shared responsibility. Two things stay with you. First, the agent host: it holds the Vault and its unseal key, and the agent container has access to the host's Docker socket, which is equivalent to root on that host. Place it in a management network and restrict LAN access to its ports. Second, workflow edit rights: the Python node runs code on the agent host, so treat the right to edit a workflow as the right to execute code there, and grant it through roles accordingly.

What we do not claim. AutomateNetOps does not yet hold SOC 2 or ISO 27001 certification, has no FedRAMP authorization, and has not yet completed an independent penetration test. We will not imply otherwise. The controls above are the ones you can verify today, in the product and in your own signed evidence bundle.

7 A worked example

A campus team has to retire an NTP server pair and point 14 Cisco IOS access switches at the replacements. Here is the change taken end to end: canvas, lab, production, evidence.



The workflow as built on the canvas. The On Error edge of the configuration step leads to Rollback, then to the same notification.

Design (the engineer, on the canvas). The workflow is eight nodes. A Manual Trigger starts it. A Data Backup node snapshots each switch's running configuration into the Gitea sidecar. An Approval Gate parks the run until the change manager approves it. A Netmiko node in configuration mode removes the two old `ntp server` lines and adds the two new ones. A second Netmiko node runs `show running-config | include ntp server`, and an Assert node checks that the output lists both new servers and neither old one. A Config Diff node compares the pre-change backup with the post-change configuration, and a Send Notification node posts the result to the team channel. The Netmiko node's On Error edge leads to a Rollback node, which restores the backup with `configure replace` after checking its MD5, then to the same notification. No line of Python was written.

Rehearse (the engineer, in the lab). On the topology map the engineer lassoes the 14 switches and clicks Clone to Lab. The Fidelity Advisor confirms the list, and Materialize lab brings up 14 Cisco IOL (IOS on Linux) nodes on containerlab on the agent host, linked as discovered. The engineer attaches each switch's latest backup as its startup configuration, opens the workflow and runs Test in Lab against the clone. Twelve pass. Two fail the Assert: their existing lines read `ntp server vrf MGMT 192.0.2.5`, so the plain `no ntp server 192.0.2.5` matched nothing and the old server survived. The engineer adds a Condition node that uses the VRF form of the removal for those two, re-runs the lab, and all 14 pass. Production has not been touched.

Execute (the engineer and the approver, in production). The engineer runs the workflow against the campus-access device pool. Backups are committed, and the run parks at the Approval Gate. The change manager, who holds the target right on that pool and the workflow-execute permission, opens the run, reads the lab result and the exact commands, and approves. The agent resolves the switch credentials from its Vault, applies the change, and the Assert and Config Diff confirm that only the NTP lines changed. The notification carries the diff. Had a switch failed, the Rollback node would have restored its backup and the notification would have said so. The approval, the run and every device result are in the audit trail.

Prove (the approver, afterwards). That night, Tavrin's scheduled STIG scan covers the campus-access pool. Its coverage-manifest record shows 14 of 14 scanned, and the time-source rules pass on every switch. The scan's summary is a signed, append-only evidence row. The approver downloads the signed-verification bundle, follows the openssl steps in its VERIFY.md on a machine with no Regnor access, and confirms the signatures. The change, its rehearsal and its proof now live in one system: each step audited, the configuration history in Git, and the compliance evidence signed. Nobody assembled it by hand.

8 How to start

Regnor is free during the beta, and one engineer can start without a procurement cycle.

- 1 **Join the beta** at automatenetops.ai. Signups are open: you receive an invite code by email and onboard in an upcoming cohort. The full platform is free during the beta, beta members have a direct line to the founding team, and early-adopter pricing is locked in at launch.
- 2 **Learn in the Regnor Academy**, free structured training at automatenetops.ai/tutorials/academy: 10 courses and 45 lessons across four tracks, from Getting Started and Workflow Designer Fundamentals to Change Safety and Compliance Monitoring. The fuller catalog is inside the app.
- 3 **Take the companion Udemy course** (a paid course), *Network Automation with Python, Ansible and Workflows*, which builds the same workflows you have read about here.
- 4 **Watch it work** in three short videos:
 - [Regnor in 7 minutes](#)
 - [Build a network automation workflow without writing Python](#)
 - [Automated DISA STIG compliance for network devices, with evidence you can verify](#)

For the approver. Bring this paper to the security review together with a signed-verification bundle from a real scan. The bundle verifies with openssl and no Regnor account.



About AutomateNetOps. AutomateNetOps LLC is a South Carolina company founded by Curtis Junevicius, a network engineer with more than 25 years in the field. Regnor is patent pending: 11 U.S. provisional patent applications have been filed.



Regnor™



Valdis™



Tavrin™

Regnor, Valdis, and Tavrin are trademarks of AutomateNetOps LLC. Technical claims checked against agent release 1.8.261, October 2026. © 2026 AutomateNetOps LLC.