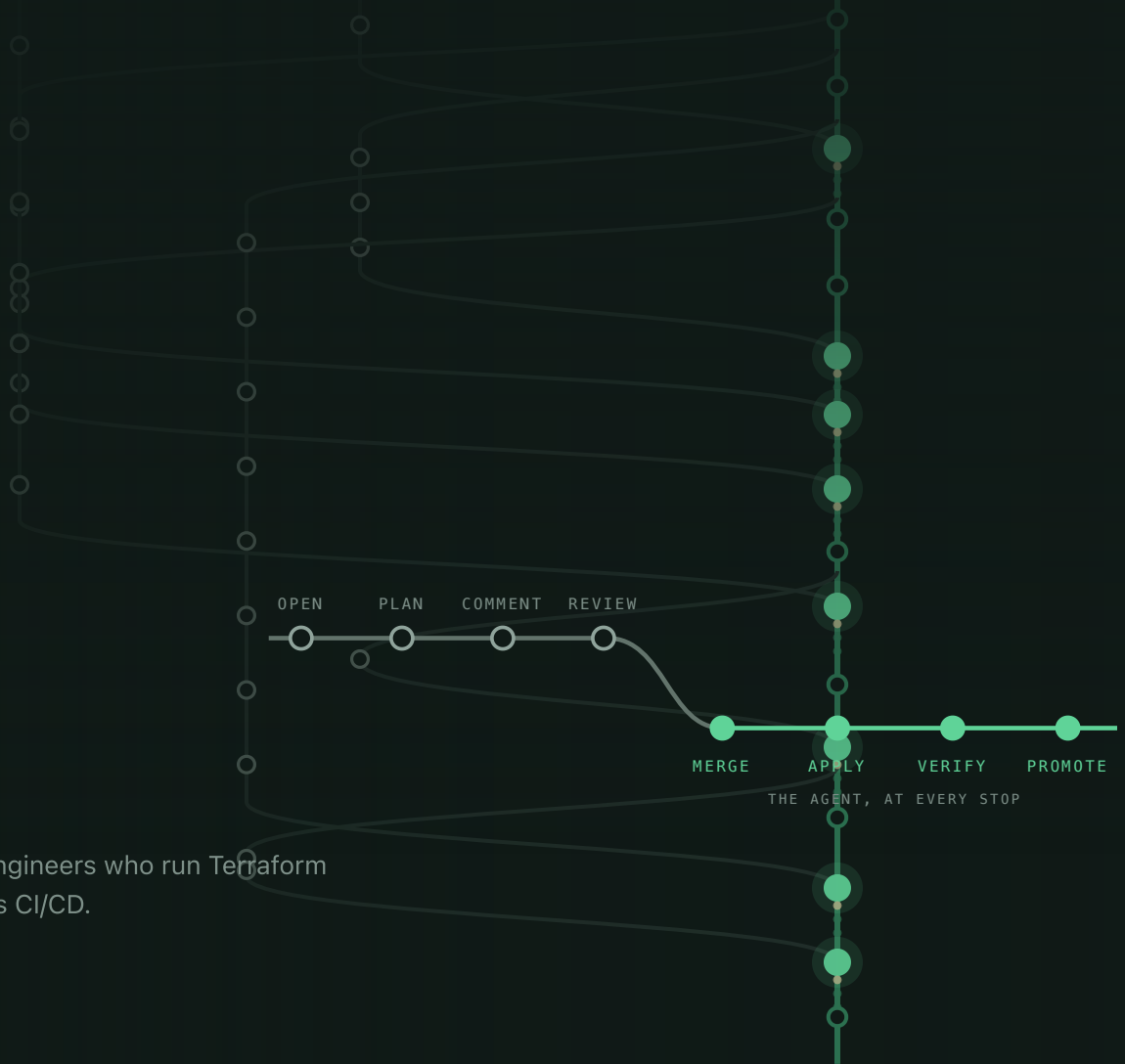
turf.build

REFERENCE ARCHITECTURE · V0.4 · NOVEMBER 2026

Pull Request to Production

Terraform delivery on GitHub Agentic Workflows, with the agent attending every apply.



For platform engineers who run Terraform through GitOps CI/CD.

THE PIPELINE

Terraform in your CI/CD pipeline today

HashiCorp's tutorial draws the pipeline most teams run: Terraform® plans on every pull request and applies on every merge, through GitHub® Actions. Two things go wrong with it.

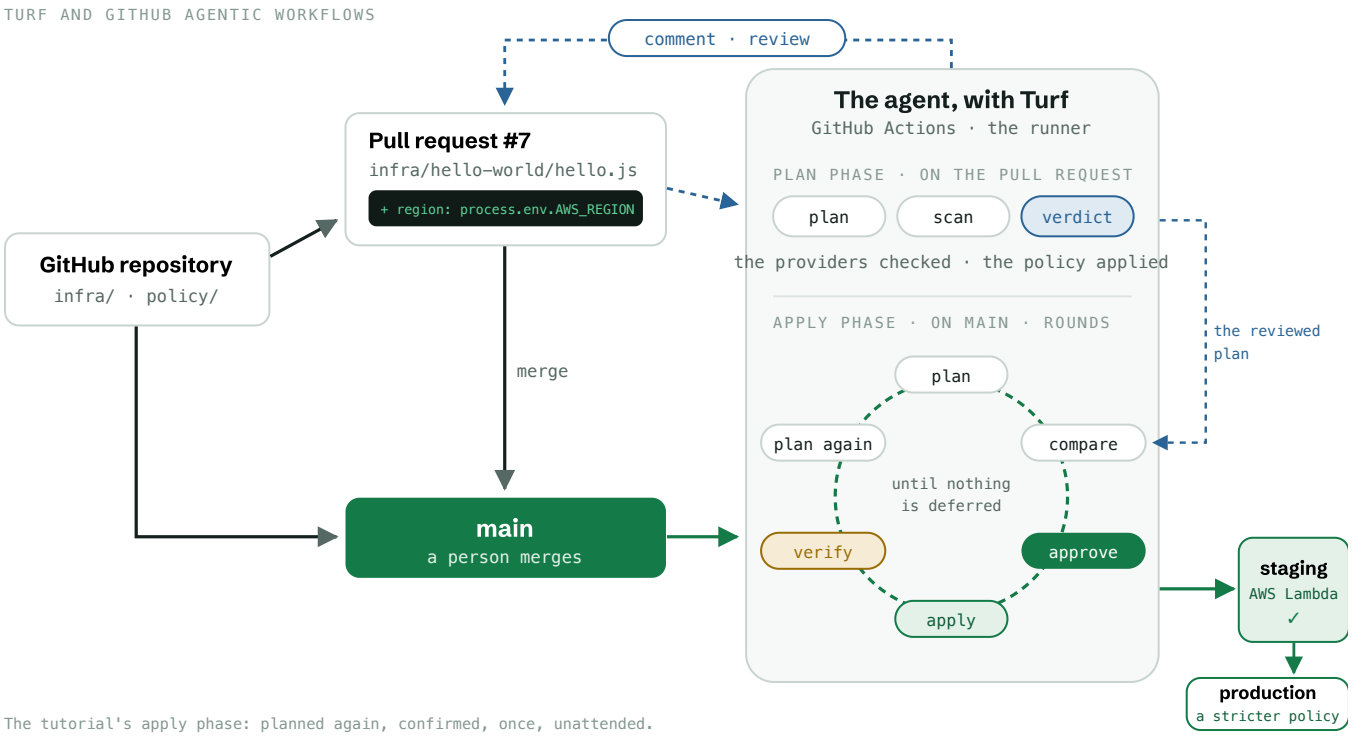
The plan you review is not the plan that applies. The pull request's plan is “speculative so it cannot modify infrastructure”. At merge, the apply workflow “uploads the Terraform configuration” again and plans again, against whatever changed in between.

The apply is unattended, too. It is confirmed on its own, with no one reading the plan that runs and nothing checking the providers it loads. An unexpected replacement leads to an outage, and the alert is how you hear of it.

> **Turf™ is an autonomous infrastructure-as-code tool for agentic workflows, built on OpenTofu™.**

Here Turf runs beside the agent in a GitHub Agentic Workflow. The agent reviews every pull request. After the merge it compares the new plan with the one you reviewed, approves that exact plan under your policy, and stays through every round.

“The apply is attended by the agent.”



The tutorial's apply phase: planned again, confirmed, once, unattended.

Figure 1. The pipeline, with the agent in it. The same stops as the tutorial draws them; the agent answers on the pull request, then runs the apply as rounds, holding the reviewed plan against the one that runs; a halt goes back to the pull request as a fix.

IN SHORT

What you stop worrying about

Four things a platform engineer no longer checks by hand. Each line below is the agent's own, from a run on 2026-10-31.

The plan you reviewed is the plan that runs.

At merge the agent plans again, compares with the plan on the pull request, and approves that exact plan or refuses with the difference. Production's plan is judged on its own, under the stricter policy.

```
Matches the reviewed plan: the changes and updates are the same as in the earlier plan comment.
```

→ Chapter 5

The apply is attended by the agent.

The Terraform configuration asks the agent to verify the rollout from inside the apply. It fetches every function, checks the answer, and the engine goes on, or halts, says why, and the fix is a pull request.

```
GET …533apcrq… → 200, message "Hello from staging!", environment "staging" (turf-hello-staging-0)
```

→ Chapters 5 and 6

Your policy is a paragraph, and every verdict quotes it.

One paragraph per GitHub Environment, in the repository, read by the agent on every run. Changing the policy is a pull request.

```
Refused under the staging approval rule (policy/approval.md): the plan would destroy the shared execution role aws_iam_role.lambda_exec.
```

→ Chapters 4 and 7

A vulnerable provider never gets as far as a plan.

Before anything opens, every provider is checked against GitHub's Advisory Database, through the GitHub MCP Server, at the version the lock file pins.

```
hashicorp/vault 5.4.0 has a high-severity advisory (GHSA-gmm6-j2g5-r52m / CVE-2025-13357) that is fixed only in 5.5.0, above the pinned version. Nothing was opened or planned.
```

→ Chapter 7

Part I is the host: GitHub Agentic Workflows, and Turf on the runner. **Part II** walks the pipeline, one stop at a time. **Part III** is what it guarantees.

PART I
THE HOST

01

OPEN
PLAN
COMMENT
REVIEW
MERGE
APPLY
VERIFY
PROMOTE

IN GITHUB'S WORDS

GitHub Agentic Workflows

GitHub's definition: “AI-powered repository automations that you define in markdown and run as GitHub Actions.” Generally available since October 28, 2026.

- **A new kind of GitHub workflow** that puts an agent into the pipeline, as a job.
- **The agent follows Markdown instructions** and can use MCP tools, inside a sandbox with a read-only token.
- **Turf provides the tools** to plan and apply Terraform configurations (your HCL, as it is): `init`, `open`, `up`, `show`, `apply`, `output`.
- **Here, an agentic workflow for pull requests** that carry infrastructure changes: one file, its frontmatter below, its prompt a runbook the three workflows share.

```
---
on: pull_request affecting infra/**
environment: staging
engine:
  id: claude
  model: claude-sonnet-5-5
imports:
  - shared/turf.md
mcp-servers:
  turf:
    type: http
    url: http://localhost:3000/mcp
    headers:
      Authorization: "Bearer ${{ secrets.TURF_MCP_AUTH_TOKEN }}"
    allowed: [skill_deployment, read_skill_file, init, open, up, show, status, cancel]
safe-outputs:
  add-comment:
    max: 1
    hide-older-comments: true
  submit-pull-request-review:
    allowed-events: [COMMENT, REQUEST_CHANGES]
---

# Turf plan

You are the plan reviewer for a pull request. This run **plans only** …

{{#runtime-import .github/workflows/shared/turf-runbook.md}}
```

`.github/workflows/turf-plan.md`, trimmed. The runbook it imports is the same for all three workflows.

PART I
THE HOST

02

OPEN
PLAN
COMMENT
REVIEW
MERGE
APPLY
VERIFY
PROMOTE

ON THE RUNNER

Turf on the runner

Turf runs beside the agent on the Actions runner: the agent gets Turf's tools, and never a cloud credential.

- **Before the agent starts**, a pre-agent step mints a one-hour AWS session from the job's OIDC token, for the Environment's role, and starts the engine and the MCP server with it.
- **The gateway enforces the tool list**: a plan run can plan, an apply run can apply, and no run can destroy, taint, import or invoke. Only the gateway holds the server's bearer token, so the agent cannot reach the server around it.
- **State lives in S3**, one key per Environment, with its lock; the lock is released when the server stops.

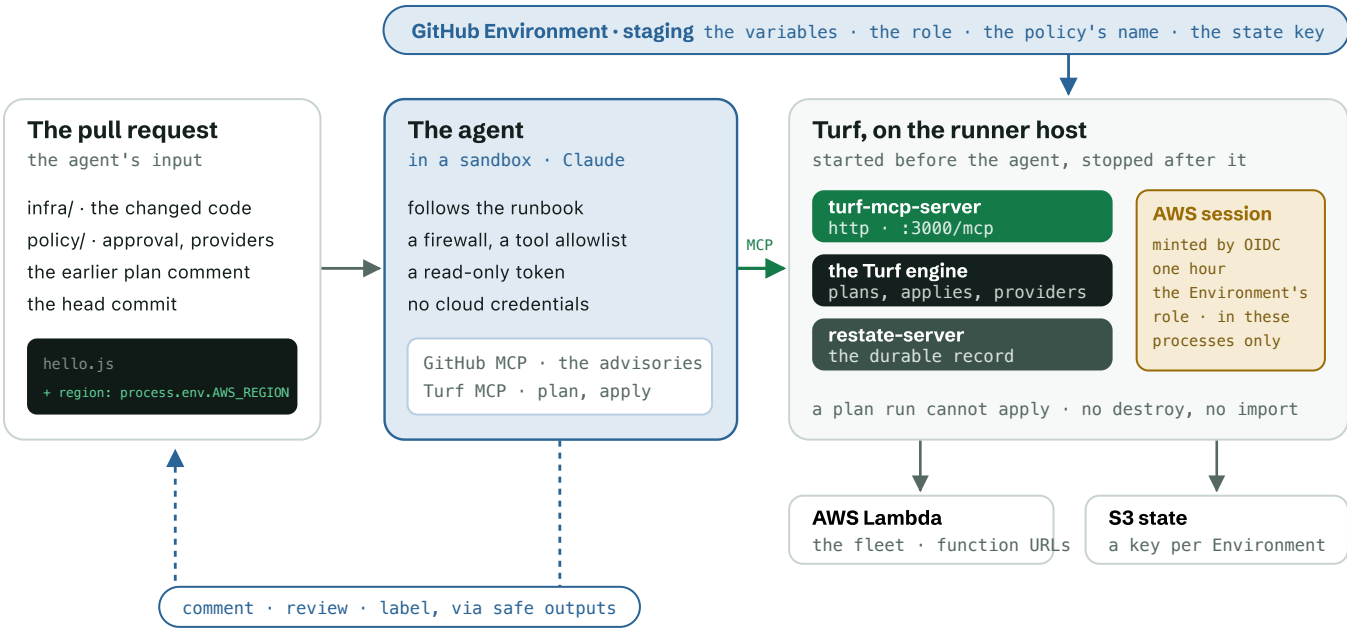


Figure 2. The runner. The pull request is the input, Turf is the engine beside the agent, and the credentials never cross into the sandbox.

RUN, 2026-10-31	AGENT TIME	COST	TURNS	JOB
Plan, on the pull request	44 s	$0.178	13	2 min 21 s
Apply, staging	68 s	$0.254	18	2 min 43 s
Apply, production	84 s	$0.233	19	3 min 5 s

Measured on Claude Sonnet 5.5, from the runs' own summaries; the job column is the whole Actions job, Turf's start-up included. Run links in the Sources.

PART II
THE LOOP

03

- OPEN
- PLAN
- COMMENT
- REVIEW
- MERGE
- APPLY
- VERIFY
- PROMOTE

THE PULL REQUEST

A review on every pull request

Change the Terraform configuration in `infra/`, and the agent answers on the pull request: the plan, a providers check, and a verdict under your policy.

One comment, one review. Ready to merge is a `COMMENT`; a refusal requests changes and says why. Nothing is applied: the plan run cancels the convergence it made.

```
--- a/infra/hello-world/hello.js
+++ b/infra/hello-world/hello.js
@@ -19,6 +19,7 @@ module.exports.handler = async (event) => {
       message,
       function: process.env.AWS_LAMBDA_FUNCTION_NAME,
       environment: process.env.ENVIRONMENT,
+      region: process.env.AWS_REGION,
     }),
   };
```

`gh pr diff 7`: one line in the handler.

TOOL CALLS · TURF, THEN GITHUB

```
✓ init · infra
providers = [
  { hashicorp/archive  2.8.1  ✓ }
  { hashicorp/aws      6.71.0 ✓ }
]
✓ list_global_security_advisories
  ecosystem: go
  affects: …provider-aws@v6.71.0
→ []
✓ list_global_security_advisories
  affects: …provider-archive@v2.8.1
→ []
✓ open · s3 · staging/…
✓ up
status = awaiting_approval
plan = { 0 add, 3 change, 0 destroy }
```

TURF PLAN · PR #7 — THE COMMENT, THEN THE REVIEW

Turf plan: 0 to add, 3 to change, 0 to destroy

Provider	Version	Advisories	Verdict
registry.opentofu.org/hashicorp/archive	2.8.1	none	allowed
registry.opentofu.org/hashicorp/aws	6.71.0	none	allowed

```
~ aws_lambda_function.hello[0]
    source_code_hash: BgdFgZjt7YLjXpiX… → hfdhdiss6ibmMnYm…
~ aws_lambda_function.hello[1]
    source_code_hash: BgdFgZjt7YLjXpiX… → hfdhdiss6ibmMnYm…
~ terraform_data.release
    input.code: BgdFgZjt7YLjXpiX… → hfdhdiss6ibmMnYm…
```

The handler code changed. The function code hash is the only input that differs. Greeting, environment and URLs are unchanged. There are no output changes and no drift.

Verdict: approve. The `staging` policy approves any plan unless it destroys `aws_iam_role.lambda_exec` or deletes every function. This plan has no deletes, and the role and both functions remain.

This run plans only. The convergence is cancelled afterwards, so the state lock is released and nothing was applied.

```
COMMENT · The plan is ready to merge under the staging policy.
Merging will apply 0 to add, 3 to change, 0 to destroy.
```

From run 41251741913, 2026-10-31, trimmed: the invocations line and the closing `turf` block are omitted; the review's first two sentences are quoted.

PART II
THE LOOP

04

- OPEN
- PLAN
- COMMENT
- **REVIEW**
- MERGE
- **APPLY**
- VERIFY
- PROMOTE

THE POLICY

Policy in plain language

Your policy is a paragraph in the repository. The GitHub Environment picks which one.

Here, staging approves almost anything, with two named exceptions, and production approves creates and in-place updates, and nothing else. The verdict always names the paragraph that decided it, and changing the policy is a pull request like any other.

```
## `staging`

Approve any plan, including replacements and deletes, except one that would:

- destroy the shared execution role (aws_iam_role.lambda_exec), or
- delete every function in the fleet (a plan whose deletes leave no
  aws_lambda_function.hello instance).

## `production`

Approve only a plan made of creates and in-place updates:

- no replacement (replace must be zero),
- no delete of any kind, including a shrink of the fleet,
- the first rollout's creates are allowed, since nothing exists yet.

Any other plan is refused: report it with the rule that refused it, say that a person
has to run the apply for this environment by hand, cancel the convergence, and apply
nothing.
```

policy/approval.md, the two policies, lightly trimmed.

```
3. Advisories. For every provider, look up the GitHub Advisory Database for its module
   github.com/hashicorp/terraform-provider-<name> at the resolved version.
   - An advisory of severity high or critical whose fixed version is above the
     resolved version, or that has no fixed version, is a deny.
   - Anything of lower severity, or already fixed, is reported and does not block.
```

policy/providers.md, rule 3, trimmed. Rules 1 and 2 are the source namespace and the pin.

staging GITHUB ENVIRONMENT

TURF_APPROVAL_POLICY	staging
AWS_ROLE_ARN	…:role/…-staging
TF_STATE_KEY	staging/lambda-fleet.tfstate
TF_VAR_fleet_size	2
approves	almost anything; two exceptions

production GITHUB ENVIRONMENT

TURF_APPROVAL_POLICY	production
AWS_ROLE_ARN	…:role/…-production
TF_STATE_KEY	production/lambda-fleet.tfstate
TF_VAR_fleet_size	3
approves	creates and in-place updates

Figure 3. The Environment selects the policy. Its variables, its role, its state key, and the paragraph the run obeys.

PART II
THE LOOP

05

- OPEN
- PLAN
- COMMENT
- REVIEW
- **MERGE**
- **APPLY**
- VERIFY
- PROMOTE

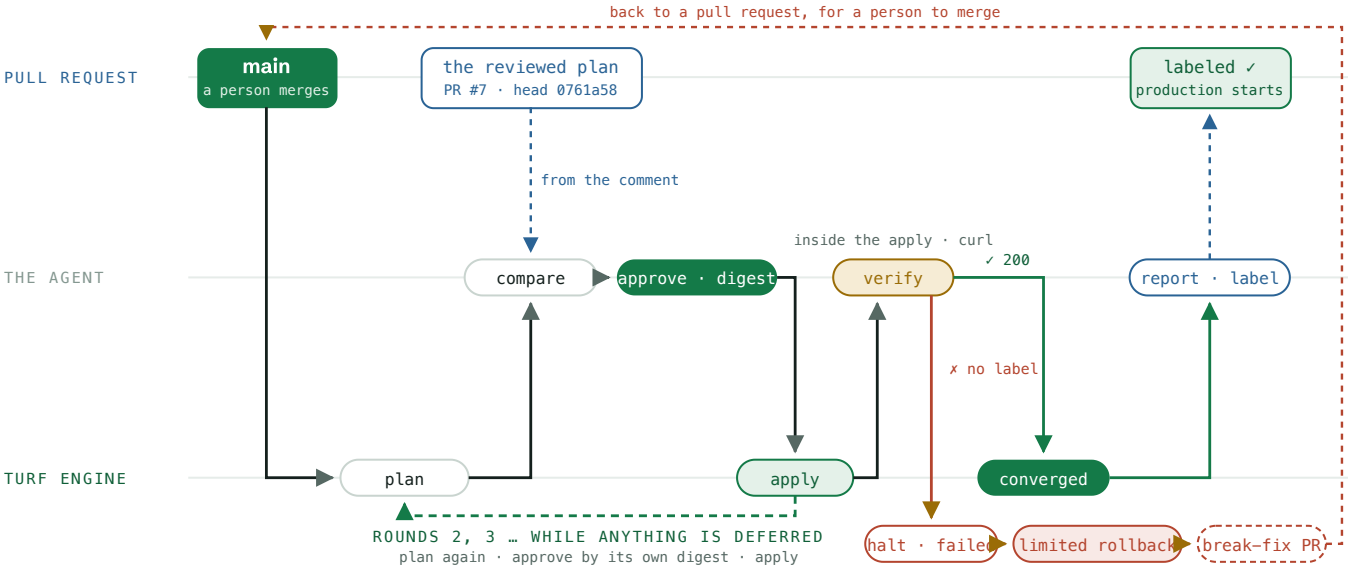
THE MERGE

A well-governed apply

The apply is attended by the agent.

- **The reviewed plan, and no other.** At merge the agent plans again, compares with the plan on the pull request, and approves that exact plan by its fingerprint; a plan that changed is refused, with the difference.
- **Supervised through every round.** A value known only after an apply is planned in the next round and approved by its own digest; `verify` runs inside the apply (chapter 6), and a failed check halts the convergence, the report says why, the label is withheld.
- **The fix is a limited rollback, as a pull request, at once.** A Terraform configuration is not rolled back by reverting the commit: a `moved` block, an import, an upgrade or a delete has already happened. The agent reverts what failed, keeps the rest, says which is which, and a person merges.

FIGURE 1'S APPLY PHASE, UNROLLED · STAGING



PR #7 matched the reviewed plan, was approved as sha256:f1d9d24f…, and converged in one round. PR #28's verify failed; the run ended as PR #29.

Figure 4. The apply phase of Figure 1, unrolled. Round by round the engine plans and applies, the agent approves each plan by its own digest and verifies from inside; a convergence ends as a label, a halt ends as a pull request.

TURF APPLY (STAGING) · PR #29 — THE BREAK-FIX PULL REQUEST · RUN 41292134365 · TRIMMED

Revert the response field rename of #28: staging verify failed

What failed. The staging apply of #28 applied 4 changes, then the `verify` turf action failed and halted `terraform_data.rollout`: both URLs answered 200, but under the key `"greeting"` instead of `"message"`.

Reverted. `infra/hello-world/hello.js`: the response field is `message` again, as `rollout.turf.hcl` verify expects.

Kept. The rename `terraform_data.release` → `terraform_data.rollout`, with its `moved` block, and the `action_trigger` target. The state is already at the new address; reverting them would plan a create of the old address and a destroy of the new one, a replacement.

PART II
THE LOOP

06

- OPEN
- PLAN
- COMMENT
- REVIEW
- MERGE
- **APPLY**
- **VERIFY**
- PROMOTE

TURF ACTIONS

Verified from inside

The agent actively participates in verifying the Lambda fleet as it rolls out: a turf action is a step the HCL asks of the agent, carried out with the agent's own tools, inside the apply.

- **After the fleet is created or updated, `verify` fires.** The engine pauses with the prompt and the inputs; the agent fetches every function URL with curl, checks the body, and answers.
- **A failed check halts the convergence**, the report says why, the label production waits for is never set, and the fix is a pull request before anyone is paged (chapter 5).

```
action "turf_action" "verify" {
  config {
    prompt = <<-EOT
      The fleet was just deployed or patched. Fetch every URL in inputs.urls
      with curl. Each must answer HTTP 200 with a JSON body whose "message"
      equals inputs.greeting and whose "environment" equals inputs.environment. …
    EOT
    inputs = { greeting = caller.output.greeting, environment = caller.output.environment,
               urls = caller.output.urls }
  }
}
action_trigger "verify" {
  target     = terraform_data.release
  events     = [after_create, after_update]
  actions    = [action.turf_action.verify]
  on_failure = halt
}
```

infra/rollout.turf.hcl, the prompt trimmed. Turf lays it over main.tf; tofu and terraform ignore it by its name.

```
TOOL RESULT · apply(plan_digest)
outcome = awaiting_actions
action_requests[0] = {
  key = "action/1/action.turf_action.verify@after_update@…"
  if_failed = "A failure halts terraform_data.release …"
  inputs = { environment = "staging",
             greeting = "Hello from staging!",
             urls = [ "https://533apcrq…", "https://op3tstef…" ]
}
BASH · ONE PLAIN CURL PER URL
$ curl -sS -m 15 -w '\n%{http_code}' https://533apcrq….on.aws/
{"message":"Hello from staging!","function":"…-staging-0",
 "environment":"staging","region":"us-west-2"}
200
TOOL CALL · apply(action_results)
action_results = { "action/1/…@…": { succeeded: true,
  progress: [ "GET …533apcrq… → 200, message "Hello from
    staging!", environment "staging" (turf-hello-staging-0)",
    "GET …op3tstef… → 200, … (turf-hello-staging-1)" ] } }
outcome = converged · rounds[0] = { applied 3, invoked 1 }
```

THE AGENT — IN THE REPORT

Apply
Round 1: applied 3, deferred 0

PART II
THE LOOP

07

- OPEN
- **PLAN**
- COMMENT
- **REVIEW**
- MERGE
- APPLY
- VERIFY
- PROMOTE

THE VERDICT

Policy enforcement

A refusal is a review: the agent requests changes and quotes the policy that decided. Two pull requests show both policies at work.

- **#5 renames the shared execution role.** A rename forces a replacement, and the staging policy will not destroy that role. Nothing applied, the lock released.
- **#6 adds `hashicorp/vault` at 5.4.0.** The providers policy finds GHSA-gmm6-j2g5-r52m, fixed in 5.5.0, and refuses before `open`.

```
--- a/infra/main.tf
@@ -18,7 +18,7 @@ resource "aws_iam_role" "lambda_exec" {
-   name = "${var.name_prefix}-exec"
+   name = "${var.name_prefix}-lambda-exec"
```

TURF PLAN · #5 · THE DIFF BLOCK, TRIMMED

```
Plan: 2 to add, 2 to change, 2 to destroy (2 replaced)
∓ aws_iam_role.lambda_exec (must be replaced: name)
    name: …-staging-exec → …-staging-lambda-exec
    arn: …:role/…-staging-exec → (known after apply)
∓ aws_iam_role_policy_attachment.lambda_policy
    role: …-staging-exec → …-staging-lambda-exec
~ aws_lambda_function.hello[0]
    role: …:role/…-staging-exec → (known after apply)
~ aws_lambda_function.hello[1]
    role: …:role/…-staging-exec → (known after apply)

Verdict: refuse
```

TURF PLAN · PR #5 — THE REVIEW

Refused under the staging approval rule (`policy/approval.md`): the plan would destroy the shared execution role `aws_iam_role.lambda_exec`. Its name changes from `turf-hello-staging-exec` to `turf-hello-staging-lambda-exec`, which forces a replacement. The policy attachment is replaced and both functions are updated to the new role. The plan is 2 to add, 2 to change, 2 to destroy. The providers rule passed (no advisories). A person needs to review and apply this change by hand, or keep the existing role name.

REQUEST_CHANGES

From run 41238948486, 2026-10-31; the review quoted whole. The diff above is `gh pr diff 5`, trimmed.

```
--- a/infra/providers.tf
@@ -8,6 +8,10 @@ terraform {
+    vault = {
+      source  = "hashicorp/vault"
+      version = "5.4.0"
+    }
```

TURF PLAN · #6 · THE PROVIDERS TABLE

Provider	Version	Advisories	Verdict
hashicorp/archive	2.8.1	none	allow
hashicorp/aws	6.71.0	none	allow
hashicorp/vault	5.4.0	1 (high)	deny
GHSA-gmm6-j2g5-r52m / CVE-2025-13357, fixed in 5.5.0			

```
Refused by providers rule 3 (Advisories).
The workspace was not opened, no plan was made,
and no state lock was taken.
```

TURF PLAN · PR #6 — THE REVIEW

Refused by `policy/providers.md` rule 3 (Advisories). `hashicorp/vault` 5.4.0 has a high-severity advisory (GHSA-gmm6-j2g5-r52m / CVE-2025-13357) that is fixed only in 5.5.0, above the pinned version. Nothing was opened or planned. Bump the vault provider to 5.5.0 or later and update `.terraform.lock.hcl`.

REQUEST_CHANGES

From run 41238989000, 2026-10-31; the review quoted whole. The diff is `gh pr diff 6`, trimmed: the lock file and `discovery.tf` are omitted.

08

- OPEN
- PLAN
- COMMENT
- REVIEW
- MERGE
- APPLY
- VERIFY
- PROMOTE

THE GUARANTEES

Safety, in layers

Two systems, each with its own layers, and the agent between them with the least it needs.

GitHub's layers hold the agent: what it may reach, what it may call, what it may write, and with which token. Turf's layers hold the change: which plan, under which policy, with which credentials, and what the HCL may ask. Nothing here is a feature flag. Each is a line in a file in the repository, and the paper has shown most of them.

GitHub's layers

AROUND THE AGENT

- ✓ A read-only token. The agent cannot write to the repository, the pull request or the issue tracker.
  `permissions: contents: read · pull-requests: read`
- ✓ A sandbox behind the Agent Workflow Firewall: only the domains the frontmatter allows.
  `network.allowed: defaults, github, host.docker.internal, the function URLs`
- ✓ Safe outputs: the comment, the review and the label are posted by a separate job, after threat detection.
  `safe-outputs: add-comment · submit-pull-request-review · add-labels`
- ✓ The MCP gateway enforces the tool list, and only it holds the server's token: a plan run cannot call `apply`, through the gateway or around it.
  `mcp-servers.turf.allowed · headers`
- ✓ GitHub Environments hold the variables, the role and the policy's name, per environment.
  `environment: staging · environment: production`
- ✓ The workflow and its compiled lock file are both committed, and reviewed like any code.
  `gh aw compile · .lock.yml`

Turf's layers

AROUND THE CHANGE

- ✓ Approval of one exact plan: `apply` names the plan it approves, and no other plan runs.
  `apply({ convergence_id, plan_digest })`
- ✓ A policy per environment, in plain language, and the verdict written with the paragraph that decided.
  `policy/approval.md · TURF_APPROVAL_POLICY`
- ✓ The providers policy: source, pin and advisories, checked before anything is opened.
  `policy/providers.md`
- ✓ Cloud credentials minted by OIDC for one hour, in the engine's process, never in the agent's.
  `assume-role-with-web-identity · the pre-agent step`
- ✓ The deployment skill's rules underneath: never a destroy plan, never a warning you cannot explain.
  `skill_deployment`
- ✓ One convergence per run, cancelled on refusal, the lock released on the way out; `destroy`, `taint`, `import` and `invoke` are not the workflow's to run.
  `cancel({ cause }) · the post-step`

Figure 5. Safety, in layers. Each line names the file or the call where the layer lives.

UNDERNEATH

The Turf engine

One engine runs every apply in this paper, on the Actions runner beside the agent; the same engine runs on a laptop or in a cluster.

A laptop, a cluster, a CI runner

Runs on a workstation, as a service in Kubernetes, or on the runner for the length of a job, as here.

Durable execution with a full record

Every effect is recorded as it happens, on restate-server, and an interrupted run picks up where it stopped.

Built-in OpenTelemetry

Traces and metrics for every change and every effect.

Operator console

Every workspace, run and approval in one place.

The Terraform language

Your HCL as it is, with registry modules and providers, the lock file honoured at every plan.

Terraform actions, and turf actions

Lifecycle triggers and failure handling; a turf action is the step the agent carries out.

Policy and governance

Plans export for your policy tools, or for an agent reading a paragraph. Nothing runs until one exact plan is approved.

An MCP server and skills

For any agent: Claude Code, Copilot, Codex, or the agent in a GitHub Agentic Workflow.

Encrypted at rest

State and the engine's store.

Convergence

Plans what it can, defers the rest, and plans again until nothing is left, in one run.

Whole convergence, in one run. The engine plans what it can know, the agent approves that exact plan, the engine applies; a value known only after that apply is planned in the next round, in the same run, under the same policy. The engine refreshes before it plans and reports drift beside the plan; the agent says whether it is the provider catching up or a change made outside the Terraform configuration.

Failure, on the record. A verify that answers `succeeded: false` halts the run, the label is withheld, production waits, the report names the reason, and the fix is already a pull request. The lock is taken by `open` and released when the server stops.

Run it

The pipeline in this paper is public: **github.com/turfbuild/turf-agentic-workflows**. Fork it, and:

1. Apply `setup/` with Turf, from your laptop: the two GitHub Environments with their variables, one AWS role per environment trusting the repository's OIDC subject, and the state bucket, as Terraform.
2. Set two repository secrets: `TURF_MCP_AUTH_TOKEN`, any random string, which the gateway sends to the Turf server; and `ANTHROPIC_API_KEY`, or switch the engine to Copilot or Codex.
3. `gh aw compile`, commit the workflows with their lock files, and open a pull request touching `infra/`.

Or import the component into a workflow of your own, since gh aw imports resolve across repositories: `imports: [turfbuild/turf-agentic-workflows/.github/workflows/shared/turf.md@main]`.

GET TURF

Download it. Run it.

Turf is available now, as a download. No form, no login. Free for noncommercial use.

```
$ curl -fsSL https://get.turf.build/install.sh | bash
```

Commercial use, including a company's internal use, requires a license. Companies evaluate Turf under a free pilot: turf.build/licensing. Turf sends anonymous usage telemetry to Turf LLC by default: the version, the platform, counts of resources, changes and runs, and an anonymous installation id. It never sends resource contents, credentials, state, variable values or names. Turn it off with `turf telemetry off`, or with `TURF_TELEMETRY=off` or `DO_NOT_TRACK=1` in the engine's environment; the full field list and the endpoint are at turf.build/telemetry.

About Turf

Turf™ is an autonomous infrastructure-as-code tool for agentic workflows, built on OpenTofu™. Every change, from a person or an agent, is a plan that is checked, approved and recorded, and the agent attends the apply. Turf runs on a laptop, in a cluster, or on a CI runner, and any agent drives it through its MCP server.

turf.build

Turf™ is a trademark of Turf LLC. Terraform® is a registered trademark, and HashiCorp Vault a trademark, of HashiCorp, Inc. OpenTofu™ is a trademark of the Linux Foundation. GitHub®, GitHub Actions, GitHub Copilot and GitHub Agentic Workflows are trademarks of GitHub, Inc. AWS and AWS Lambda are trademarks of Amazon.com, Inc. or its affiliates. Claude is a trademark of Anthropic, PBC. Turf is not affiliated with, endorsed by, or sponsored by HashiCorp, IBM, the Linux Foundation, the OpenTofu project, GitHub, Amazon or Anthropic; every mark is used only to describe interoperability. Quotations from HashiCorp's tutorial “Automate Terraform with GitHub Actions” and from GitHub's documentation are attributed in the text. All other marks belong to their owners.

Pull Request to Production, v0.4, 2026-11-01.