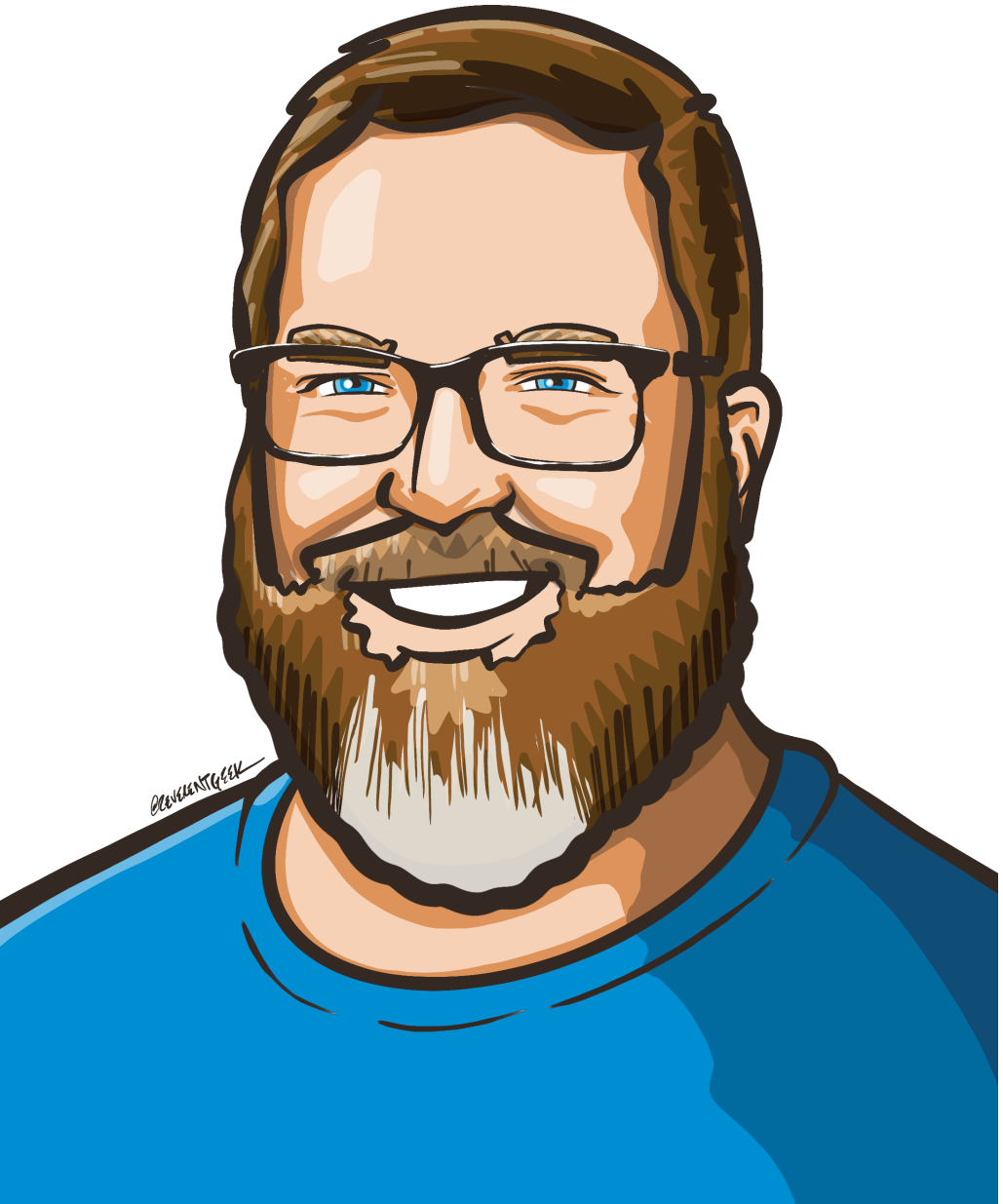
CI/CD with GitHub Actions

Chris Ayers

https://github.com/codebytes/github-actions-demos



Chris Ayers

Principal Software Engineer
Azure EngOps AzRel
Microsoft

BlueSky: @chris-ayers.com

LinkedIn: - chris-l-ayers

Blog: https://chris-ayers.com/

GitHub: Codebytes

Mastodon: @Chrisayers@hachyderm.io

~~Twitter: @Chris_L_Ayers~~

YAML

Yet Another Markup Language

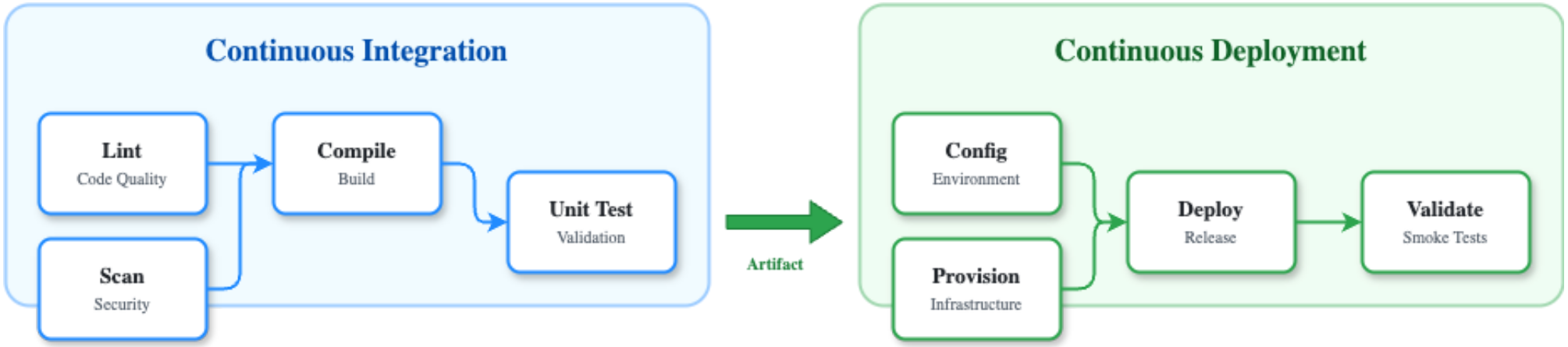
GitHub uses YAML for workflows

Demo: Online Parser

Feature	Description
Lists	Start with a –
Key-Value	Key: value
Objects	Objects: Properties of objects

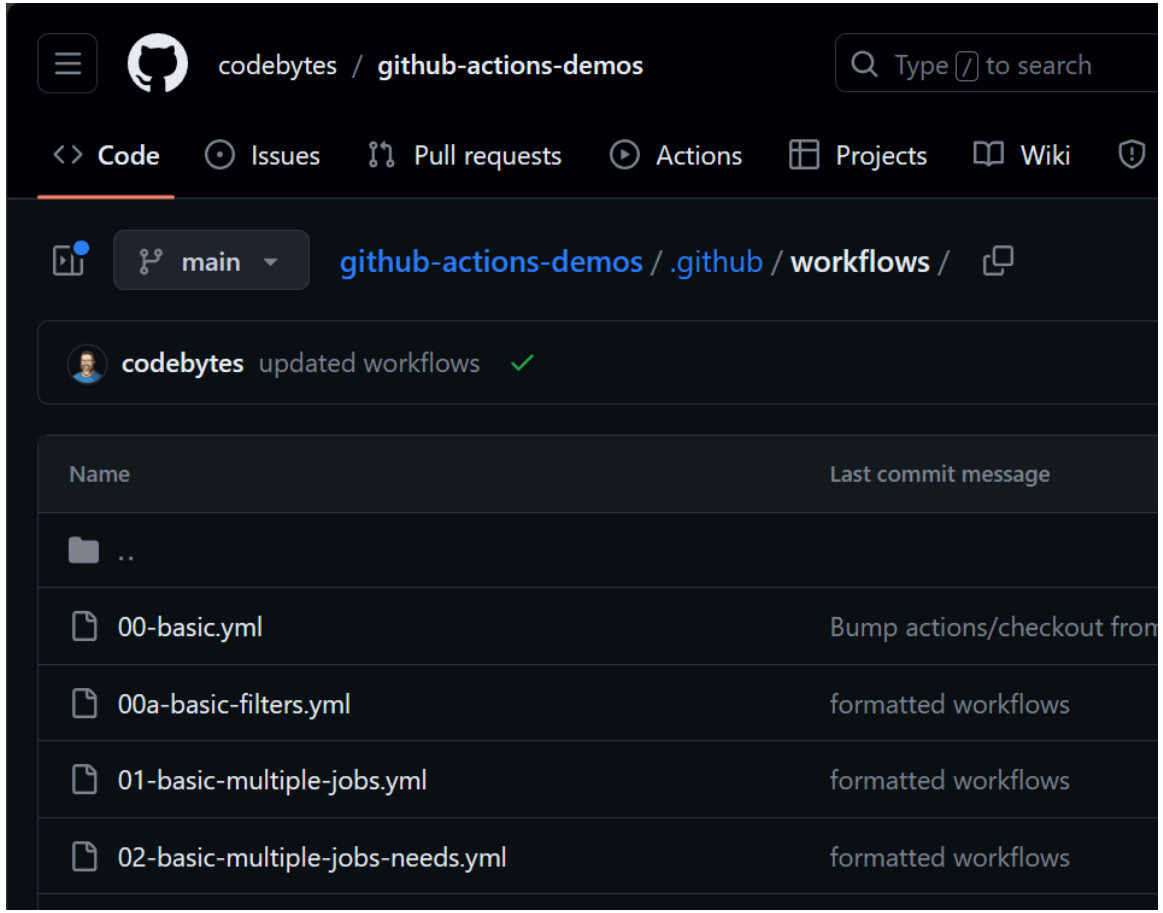
Workflows & Pipelines

From code commit to production deployment



Actions Overview

- Live in the `.github/workflows` folder
- Workflows are defined in YAML
- Workflows are Event Driven



Workflow Syntax

```yaml
name: CI Pipeline
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
  workflow_dispatch:
```

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Run tests
        run: npm test
      - name: Build
        run: npm run build
```

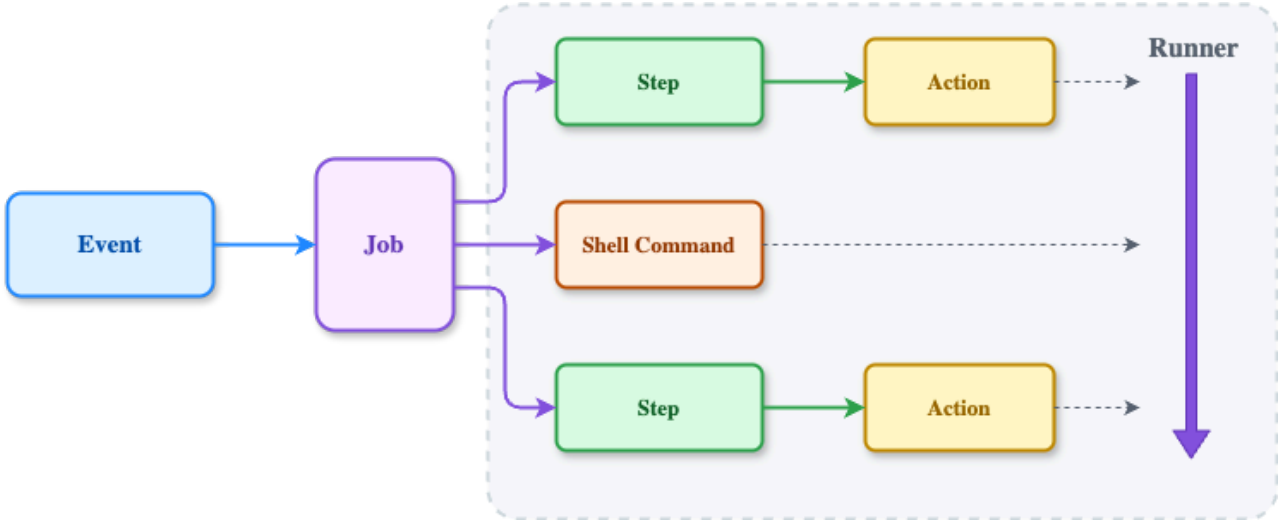
Events that trigger workflows

Events docs

- branch_protection_rule
- checks
- create / delete
- deployment
- discussion
- fork
- issue_comment
- issues
- label
- page_build
- pull_request
- pull_request_review
- pull_request_review_comment
- push
- release
- schedule / status
- workflow_call / workflow_dispatch

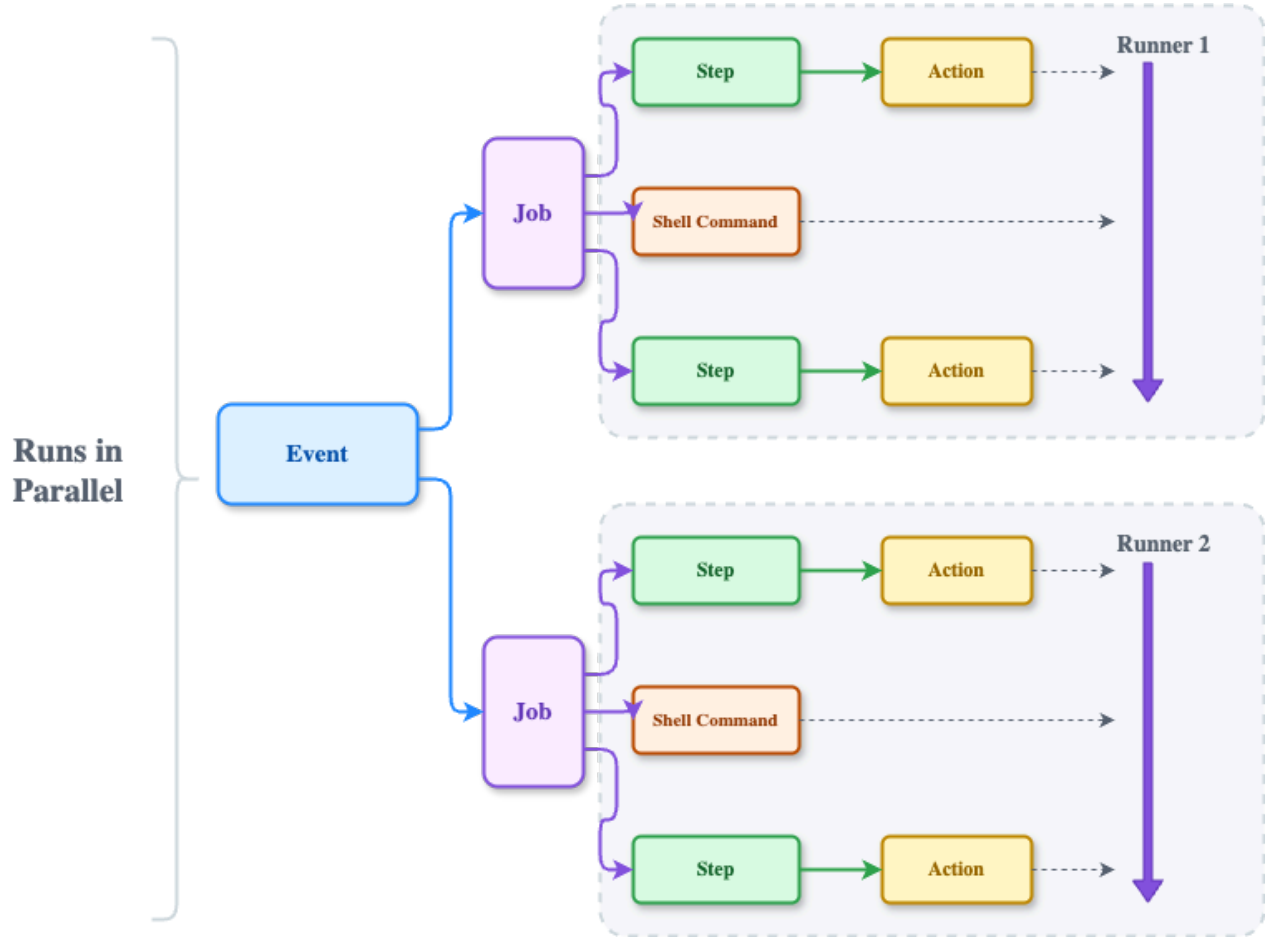
Workflows

- Events trigger workflows
- Workflows contain jobs
- Jobs contain steps
- Steps are commands or actions



Jobs

- Workflows can contain multiple jobs
- Jobs run in parallel by default
- Each job runs on a Runner
- Steps and Shell Commands run in sequence



Steps

`run:` — Shell commands

```yaml
steps:
  - name: Single line
    run: echo "Hello"

  - name: Multi-line
    run: |
      echo "Building..."
      npm ci
      npm run build

  - name: Use a different shell
    run: Get-Process
    shell: pwsh
```

`uses:` — Actions

```yaml
steps:
  - uses: actions/checkout@v4

  - uses: actions/setup-node@v4
    with:
      node-version: 20
      cache: 'npm'

  - name: Step with output
    id: version
    run: echo "tag=v1.0" >>
         $GITHUB_OUTPUT

  - run: echo ${{ steps.version
         .outputs.tag }}
```

Job Dependencies & Outputs

`needs:` — Sequencing jobs

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps: [...]

  test:
    needs: build  # waits for build
    runs-on: ubuntu-latest
    steps: [...]

  deploy:
    needs: [build, test]
    runs-on: ubuntu-latest
```

Passing data between jobs

```yaml
jobs:
  version:
    outputs:
      tag: ${{ steps.v.outputs.tag }}
    steps:
      - id: v
        run: echo "tag=v1.2.3" >>
             $GITHUB_OUTPUT
  deploy:
    needs: version
    steps:
      - run: echo ${{ needs
             .version.outputs.tag }}
```

Runners

- Specify the type of runner with `runs-on` (e.g., `ubuntu-latest`).
- GitHub provisions a new VM for each job.
- Steps in a job share information using the runner's filesystem.
- VM is decommissioned after job completion.

Supported runners and hardware

- GitHub-hosted runner application is open source.
- 🪟 **Windows** · 🐧 **Linux** · 🍎 **macOS**
  - Runners include preinstalled software, updated weekly.
  - There are also Large Hosted Runners
- 🏠 Self-Hosted Runners
- You can install additional software on runners.

Environment Variables & Contexts

Scoping: workflow → job → step

```yaml
env:
  APP_ENV: production
jobs:
  build:
    env:
      NODE_ENV: test
    steps:
      - run: echo $APP_ENV
        env:
          LOG_LEVEL: debug
```

Contexts provide runtime info

Context	Example
`github.*`	`github.sha`, `github.ref`
`env.*`	`env.APP_ENV`
`secrets.*`	`secrets.API_KEY`
`runner.*`	`runner.os`
`matrix.*`	`matrix.node-version`

Path Filters & Concurrency

📂 Path Filters — Monorepo support

```yaml
on:
  push:
    paths:
      - 'src/**'
      - 'package.json'
    paths-ignore:
      - 'docs/**'
      - '*.md'
```

Only triggers when relevant files change

🔁 Concurrency — Avoid duplicate runs

```yaml
concurrency:
  group: ${{ github.workflow }}-
    ${{ github.ref }}
  cancel-in-progress: true
```

- ✅ Cancels stale PR runs
- ✅ Saves runner minutes
- ✅ Ensures only latest commit deploys

Expressions & Conditionals

Expressions `${{ }}`

```yaml
steps:
  - name: Greet
    run: echo "SHA is ${{ github.sha }}"

  - name: Only on main
    if: github.ref == 'refs/heads/main'
    run: echo "Deploying..."

  - name: Skip on fork
    if: github.event.pull_request.head
        .repo.fork == false
    run: npm run deploy
```

Common Functions

Function	Use
`contains()`	Check strings/arrays
`startsWith()`	Branch matching
`format()`	String templates
`toJSON()`	Debug contexts
`success()`	Previous step OK
`failure()`	Previous step failed
`always()`	Run regardless

Matrix Strategies

Test across multiple configurations simultaneously

```yaml
jobs:
  test:
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest, macos-latest]
        node-version: [18, 20, 22]
    runs-on: ${{ matrix.os }}
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.node-version }}
      - run: npm test
```

🎯 Creates **9 parallel jobs** (3 OS × 3 versions)

Secrets & Permissions

Using Secrets

```yaml
steps:
  - name: Deploy
    env:
      TOKEN: ${{ secrets.DEPLOY_TOKEN }}
    run: ./deploy.sh
```

⚠️ Never `echo` secrets in logs

⚠️ Never use structured data as a secret

GITHUB_TOKEN Permissions

```yaml
permissions:
  contents: read
  pull-requests: write
  packages: write
```

🔐 Always use **least-privilege**

Default token is scoped per-repo

Real-World Pipeline — .NET CI/CD

Build, Test & Publish

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-dotnet@v4
        with:
          dotnet-version: '10.0.x'
      - run: dotnet restore
      - run: dotnet build -c Release
      - run: dotnet test -c Release
      - run: dotnet publish -c Release
             -o ./webapp
      - uses: actions/upload-artifact@v4
        with:
          name: webapp
          path: ./webapp
```

Real-World Pipeline — .NET CI/CD

Deploy to Azure

```yaml
  deploy:
    needs: build
    runs-on: ubuntu-latest
    environment:
      name: production
      url: https://myapp.azurewebsites.net
    steps:
      - uses: azure/login@v2
        with:
          client-id: ${{ secrets.
            AZURE_CLIENT_ID }}
          tenant-id: ${{ secrets.
            AZURE_TENANT_ID }}
      - uses: actions/download-artifact@v4
        with:
          name: webapp
      - uses: azure/webapps-deploy@v3
        with:
          app-name: myapp
```

📂 Demo: `10-dotnet.yml`

Artifacts & Caching

📦 Artifacts — Share between jobs

```yaml
- uses: actions/upload-artifact@v4
  with:
    name: build-output
    path: dist/

# In another job:
- uses: actions/download-artifact@v4
  with:
    name: build-output
```

⚡ Caching — Speed up builds

```yaml
- uses: actions/setup-node@v4
  with:
    node-version: 20
    cache: 'npm'
```

Or manual cache control:

```yaml
- uses: actions/cache@v4
  with:
    path: ~/.npm
    key: ${{ runner.os }}-npm-
         ${{ hashFiles('**/
         package-lock.json') }}
```

DEMOS

Workflow Basics

ACT

Run Actions Locally

- nektos/act
- SanjulaGanepola/github-local-actions

Reusable Workflows & Composite Actions

🔄 Reusable Workflows

```yaml
# .github/workflows/ci.yml
on:
  workflow_call:
    inputs:
      environment:
        type: string
        required: true
```

Called from another workflow:

```yaml
jobs:
  deploy:
    uses: ./.github/workflows/ci.yml
    with:
      environment: staging
```

🧩 Composite Actions

```yaml
# .github/actions/setup/action.yml
name: 'Project Setup'
runs:
  using: 'composite'
  steps:
    - uses: actions/setup-node@v4
      with:
        node-version: 20
        cache: 'npm'
    - run: npm ci
      shell: bash
```

Used in workflows:

```yaml
- uses: ./.github/actions/setup
```

Composite Actions vs Reusable Workflows

	🧩 Composite Action	🔄 Reusable Workflow
Scope	Runs as a **step** inside a job	Runs as an entire **job** (or jobs)
Location	`action.yml` in any directory	Must be in `.github/workflows/`
Sharing	Can publish to Marketplace	Shared via `workflow_call` trigger
Secrets	Inherits caller's context	Must be passed explicitly (or `inherit`)
Runners	Uses the **caller's** runner	Can specify its **own** runner
Outputs	Step-level outputs	Job-level outputs

✅ Use Composite Actions for:

- Bundling related steps (setup, lint)
- Reusable across many workflows
- Publishing to the Marketplace

✅ Use Reusable Workflows for:

- Complete CI/CD pipelines
- Multi-job orchestration
- Enforcing org-wide standards

Environments & Deployment

🌍 Environments

- **Protection rules** — require approvals
- **Wait timers** — delay before deploy
- **Branch restrictions** — only `main` → prod
- **Environment secrets** — scoped per env

```yaml
jobs:
  deploy-prod:
    environment:
      name: production
      url: https://myapp.com
    runs-on: ubuntu-latest
```

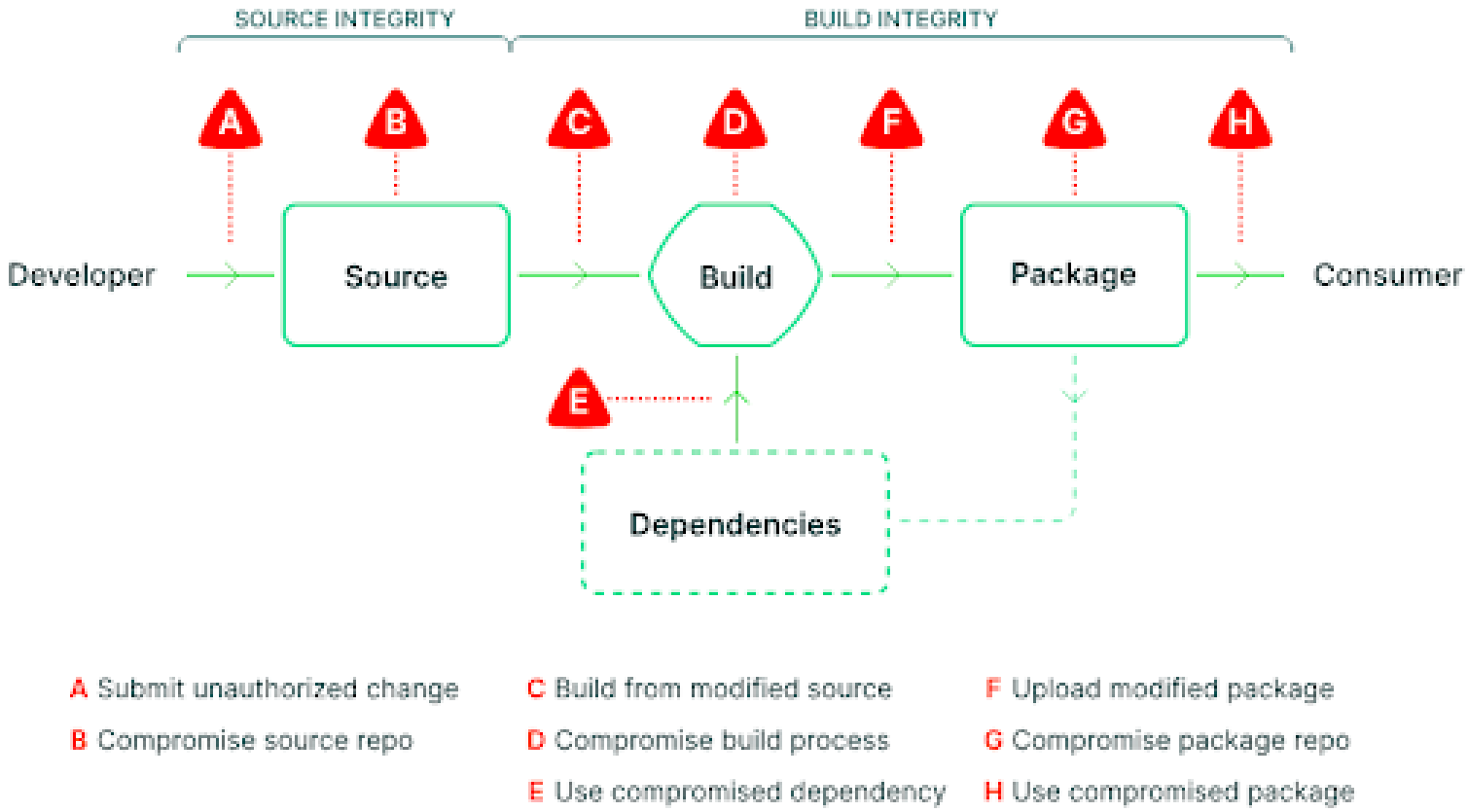
🔑 OIDC for Cloud Auth

No stored secrets — federated identity

```yaml
permissions:
  id-token: write
  contents: read

steps:
  - uses: azure/login@v2
    with:
      client-id: ${{ secrets.
        AZURE_CLIENT_ID }}
      tenant-id: ${{ secrets.
        AZURE_TENANT_ID }}
      subscription-id: ${{ secrets.
        AZURE_SUBSCRIPTION_ID }}
```

Supply Chain Attacks



Security

- Never use structured data as a secret
- Register all secrets used within workflows
- Audit how secrets are handled
- Use credentials that are minimally scoped
- Audit and rotate registered secrets
- Consider requiring review for access to secrets
- Use an action instead of an inline script (recommended)
- Use an intermediate environment variable
- Use OpenID Connect to access cloud resources
- Pin third-party actions to a full length commit SHA

https://docs.github.com/en/actions/security-guides

Actions Updates - Dependabot

- Actions are regularly updated for enhanced automation.
- Dependabot keeps GitHub Actions references in workflow.yml up-to-date.
- If newer action versions exist, Dependabot sends an update pull request.
- Dependabot also updates git references for reusable workflows.

`.github/dependabot.yml`

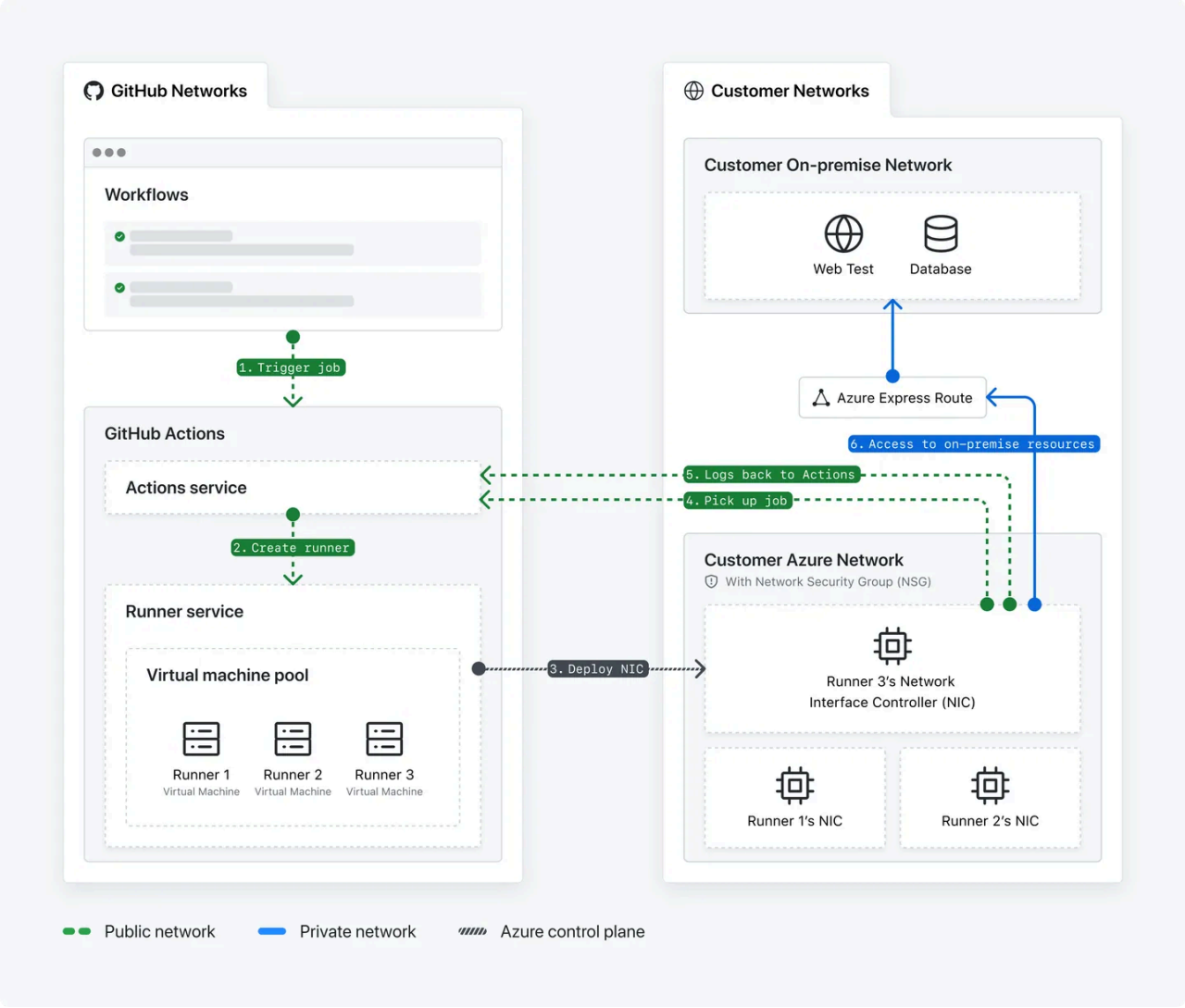
```yaml
version: 2
updates:
  # See documentation for possible values
  - package-ecosystem: "github-actions"
    # Location of package manifests
    directory: "/"
    schedule:
      interval: "weekly"
```

DEMOS

Security & Deployment

Bonus - Private Networking

- GitHub Actions triggers, creating a runner.
- Runner service deploys the runner's NIC into your Azure VNET.
- The runner agent picks up the workflow job.
- Runner sends logs back; NIC accesses private resources.



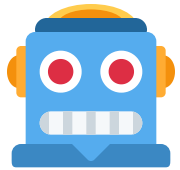
GitHub Well-Architected Framework

- **Community-driven guide** for deploying GitHub effectively.
- Design principles
- Framework pillars
- Actionable, prescriptive advice

GitHub Well-Architected Framework

Key Principles of the Framework

- 🔐 Security
- 📈 Scalability
- ⚙️ Automation
- 🤝 Collaboration
- 👁️ Observability
- 🚀 Performance
- 🏛️ Governance
- 💡 Innovation



Agentic Workflows

AI-powered automation with natural language

@Chris_L_Ayers - https://chris-ayers.com

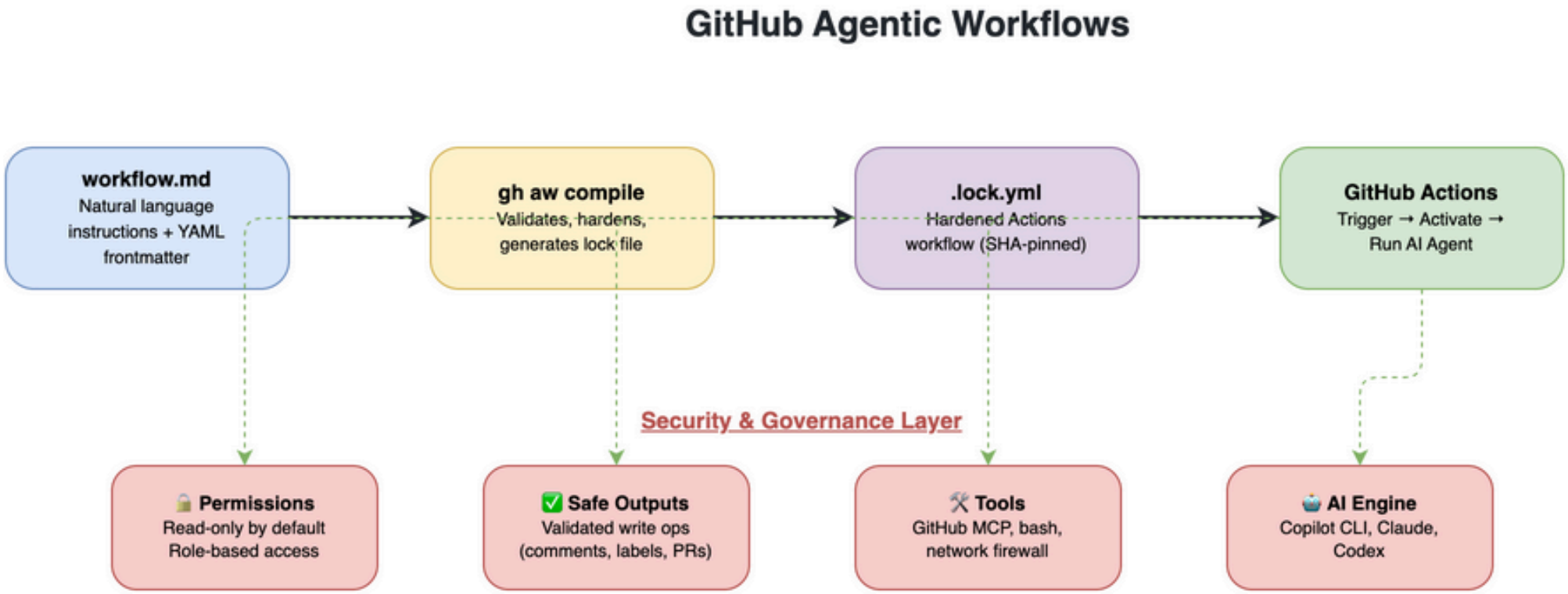
What are Agentic Workflows?

- **AI-powered** GitHub Actions
- Write automation in **markdown**
- Agents **understand context**, make decisions, and act
- Compiled to hardened `.lock.yml` files

Traditional vs Agentic

Traditional	Agentic
Fixed if-then rules	Natural language
Explicit scripting	Context-aware AI
Brittle to change	Adapts flexibly

How It Works


GitHub Agentic Workflows
workflow.md
Natural language instructions + YAML frontmatter
gh aw compile
Validates, hardens, generates lock file
.lock.yml
Hardened Actions workflow (SHA-pinned)
GitHub Actions
Trigger → Activate → Run AI Agent
Security & Governance Layer
Permissions
Read-only by default
Role-based access
Safe Outputs
Validated write ops (comments, labels, PRs)
Tools
GitHub MCP, bash, network firewall
AI Engine
Copilot CLI, Claude, Codex

Workflow Structure

Frontmatter (YAML config)

```yaml
---
on:
  issues:
    types: [opened]
permissions: read-all
tools:
  github:
    toolsets: [issues, labels]
safe-outputs:
  add-comment:
  add-labels:
    allowed: [bug, feature, question]
---
```

Body (natural language instructions)

```
# Issue Triage Agent

Analyze new issues and categorize
them with the appropriate label.

Skip issues that:
- Already have labels
- Have been assigned to a user

After adding a label, comment
mentioning the author with your
reasoning.
```

📝 The AI agent reads and executes these instructions at runtime

Key Security Features

🔒 Permissions

- **Read-only** by default
- Role-based access (`roles:`)
- `strict: true` enforced

✅ Safe Outputs

- Validated write operations
- Scoped to specific actions

🛡️ Lockdown Mode

- Custom token required
- Integrity filtering
- Bot & role filtering

🛠️ Tooling

- `gh aw compile` — Build & validate
- `gh aw audit` — Analyze runs
- `gh aw secrets` — Manage tokens

Example: Issue Triage Agent

What it does:

- Lists unlabeled issues
- Analyzes title & body with AI
- Adds appropriate labels
- Comments with reasoning

Triggers:

- Schedule (weekday afternoons)
- Manual dispatch

Safe outputs:

- `add-labels` (scoped allowlist)
- `add-comment`

Security:

- `strict: true`
- `lockdown: true`
- Role-based access

Questions

@Chris_L_Ayers - https://chris-ayers.com

Resources

Links

- https://docs.github.com
- https://skills.github.com
- codebytes/github-actions-demos
- Agentic Workflows Docs
- Agentics Collection
- Learn: Automate with Actions

Follow Chris Ayers

BlueSky: @chris-ayers.com

LinkedIn: - chris-l-ayers

Blog: https://chris-ayers.com/

GitHub: Codebytes

Mastodon: @Chrisayers@hachyderm.io

~~Twitter: @Chris_L_Ayers~~

Thank you!

Please connect