**Creating DevOps pipeline using GitHub Copilot**

End-to-end **DevOps pipeline** using **GitHub Copilot** + **GitHub Actions**. This guide is cloud-agnostic, with concrete prompts and YAML that Copilot can scaffold for you. You can adapt it for AWS/Azure/GCP/K8s easily.

**0) Prereqs (one-time)**

* GitHub repo (enable **GitHub Copilot** and **Copilot Chat** in VS Code).
* A sample app (Node/Java/Python—anything with tests).
* Optional cloud target:
  + **AWS**: create an OIDC role for your repo (least privilege).
  + **Azure**: Federated credentials for a Service Principal.
  + **GCP**: Workload Identity Federation setup.
* Optional container registry (GHCR/AzureCR/ECR/GCR).

**1) Decide your pipeline goals**

Most teams start with:

1. Lint → Test → Build
2. (Optional) Containerize → Push image
3. (Optional) IaC plan/apply (Terraform/OpenTofu)
4. Deploy to **dev** automatically; **stage/prod** require approval
5. Security gates: SAST, dependency scan, IaC scan

Create docs/pipeline.md and describe your goals; Copilot uses this as context.

**Copilot prompt (in docs/pipeline.md):**

“Design a CI/CD pipeline for a Node + Docker app with GitHub Actions: lint, unit tests, build, container scan, push to GHCR, deploy to dev on merge to main, manual approval for stage/prod, Terraform plan on PR, apply on protected envs using OIDC. Add caching and artifacts.”

**2) Branch & environments**

* Branch strategy: main (prod), develop (dev), feature branches.
* GitHub **Environments**: dev, stage, prod (set required reviewers for apply/deploy).

**3) Scaffold project files with Copilot**

Open VS Code, create empty files and let Copilot propose content.

**3.1 Node example (adjust for your stack)**

* package.json test/lint scripts:

{

"scripts": {

"lint": "eslint .",

"test": "npm run test:unit",

"test:unit": "vitest run --coverage"

}

}

**Copilot prompt:**

“Write minimal package.json scripts for linting with eslint and tests with vitest.”

* Add a Dockerfile:

# minimal Dockerfile for Node app

FROM node:20-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

CMD ["npm","start"]

**Copilot prompt:**

“Create a production-ready Dockerfile for Node 20 Alpine using npm ci and a non-root user.”

**4) CI workflow (PR validation)**

Create .github/workflows/ci.yml and type the comment below; accept Copilot’s draft then refine.

# CI: runs on PR (lint, test, build), uploads artifacts, runs SAST/deps scan

name: CI

on:

pull\_request:

branches: [ main, develop ]

workflow\_dispatch:

jobs:

build\_test:

runs-on: ubuntu-latest

permissions:

contents: read

security-events: write

steps:

- uses: actions/checkout@v4

- name: Setup Node

uses: actions/setup-node@v4

with:

node-version: 20

cache: 'npm'

- name: Install

run: npm ci

- name: Lint

run: npm run lint

- name: Unit tests

run: npm test

- name: Build (if applicable)

run: npm run build || echo "no build step"

- name: Upload build artifacts

uses: actions/upload-artifact@v4

with:

name: app-dist

path: |

dist/\*\*

coverage/\*\*

!\*\*/node\_modules/\*\*

**Optional scans (add to the same job):**

- name: Dependency Review

uses: actions/dependency-review-action@v4

- name: CodeQL init

uses: github/codeql-action/init@v3

with:

languages: javascript

- name: CodeQL analyze

uses: github/codeql-action/analyze@v3

**Copilot prompt (top of file as comments):**

“CI workflow for Node: checkout, setup node with caching, npm ci, eslint, vitest, upload dist+coverage, dependency review, CodeQL scan. Trigger on PR and manual.”

**5) Build & push container (on merge to main)**

Create .github/workflows/cd-image.yml.

name: Build and Push Image

on:

push:

branches: [ main ]

workflow\_dispatch:

jobs:

docker:

runs-on: ubuntu-latest

permissions:

contents: read

packages: write

steps:

- uses: actions/checkout@v4

- name: Log in to GHCR

uses: docker/login-action@v3

with:

registry: ghcr.io

username: ${{ github.actor }}

password: ${{ secrets.GITHUB\_TOKEN }}

- name: Extract metadata (tags, labels)

id: meta

uses: docker/metadata-action@v5

with:

images: ghcr.io/${{ github.repository }}

- name: Build and push

uses: docker/build-push-action@v6

with:

push: true

tags: ${{ steps.meta.outputs.tags }}

labels: ${{ steps.meta.outputs.labels }}

**Copilot prompt:**

“Create a GitHub Actions workflow to build and push a Docker image to GHCR using docker/metadata-action for tags and labels.”

**6) IaC: Terraform plan on PR, apply on env approval**

Create infra/ with Terraform (VPC/AKS/EKS/GKE/etc.). Let Copilot draft based on comments.

**Copilot prompt (in infra/README.md):**

“Terraform stack: provider, remote backend, workspaces (dev/stage/prod), network, one compute/service, outputs. Follow least privilege and tagging.”

**6.1 Terraform PR Plan**

.github/workflows/iac-plan.yml

name: Terraform Plan

on:

pull\_request:

paths:

- 'infra/\*\*'

workflow\_dispatch:

jobs:

plan:

runs-on: ubuntu-latest

permissions:

id-token: write

contents: read

pull-requests: write

steps:

- uses: actions/checkout@v4

- name: Setup Terraform

uses: hashicorp/setup-terraform@v3

# Example: AWS OIDC assume-role (adjust for Azure/GCP)

- name: Configure AWS credentials

uses: aws-actions/configure-aws-credentials@v4

with:

role-to-assume: ${{ secrets.AWS\_ROLE\_ARN }}

aws-region: us-east-1

- name: Terraform Init

working-directory: infra

run: terraform init -input=false

- name: Terraform Plan (dev)

working-directory: infra

run: |

terraform workspace select dev || terraform workspace new dev

terraform plan -input=false -out=plan-dev.tfplan

- name: Upload plan

uses: actions/upload-artifact@v4

with:

name: plan-dev

path: infra/plan-dev.tfplan

**6.2 Terraform Apply with Environment Approval**

.github/workflows/iac-apply.yml

name: Terraform Apply

on:

workflow\_dispatch:

inputs:

environment:

description: 'Environment to deploy (dev|stage|prod)'

required: true

default: 'dev'

jobs:

apply:

runs-on: ubuntu-latest

environment: ${{ inputs.environment }} # Require approval in GitHub Environments

permissions:

id-token: write

contents: read

steps:

- uses: actions/checkout@v4

- uses: hashicorp/setup-terraform@v3

- name: Configure AWS credentials

if: ${{ inputs.environment == 'dev' || inputs.environment == 'stage' || inputs.environment == 'prod' }}

uses: aws-actions/configure-aws-credentials@v4

with:

role-to-assume: ${{ secrets.AWS\_ROLE\_ARN }}

aws-region: us-east-1

- name: Terraform Init

working-directory: infra

run: terraform init -input=false

- name: Select Workspace

working-directory: infra

run: terraform workspace select ${{ inputs.environment }} || terraform workspace new ${{ inputs.environment }}

- name: Terraform Apply

working-directory: infra

run: terraform apply -auto-approve -input=false

**Copilot prompts (add as top comments in each file):**

“Plan on PR; apply only via workflow\_dispatch with environment approvals. Use OIDC to assume cloud role.”

**7) Deploy app to dev (auto), stage/prod (manual)**

Choose one target:

**Option A: Direct VM/K8s deploy**

.github/workflows/deploy.yml

name: Deploy App

on:

push:

branches: [ main ]

workflow\_dispatch:

jobs:

deploy\_dev:

runs-on: ubuntu-latest

environment: dev

steps:

- uses: actions/checkout@v4

- name: Deploy to dev

run: |

echo "kubectl set image ... or helm upgrade --install ... or ssh to VM and docker pull ghcr.io/OWNER/REPO:sha"

**Option B: Serverless/Platform deploy (examples)**

* **AWS**: SAM/CDK deploy
* **Azure**: az webapp up or Bicep/ARM
* **GCP**: gcloud run deploy

**Copilot prompt (inside deploy.yml):**

“Implement a dev deployment using Helm to an existing Kubernetes cluster; use image tag from the previous workflow; use rollout status and fail on timeout.”

**8) Add security & quality gates**

* **Container scan:** Trivy or Grype step in image workflow.

- name: Scan image with Trivy

uses: aquasecurity/trivy-action@0.24.0

with:

image-ref: ghcr.io/${{ github.repository }}:${{ github.sha }}

format: 'table'

exit-code: '1'

vuln-type: 'os,library'

* **IaC scan:** tfsec or checkov in iac-plan.yml.

- name: tfsec

uses: aquasecurity/tfsec-action@v1.0.7

with:

working\_directory: infra

additional\_args: '--soft-fail' # or remove to gate PRs

* **Dependabot:** enable for ecosystem updates (GitHub → Settings → Code security).

**Copilot prompt:**

“Add Trivy scan to fail pipeline on critical vulnerabilities; print summary table.”

**9) Speed & reliability**

* **Caches:** actions/setup-node cache, actions/cache for ~/.cache/pip, .m2, etc.
* **Matrix builds:** test Node {18,20} or OSes.
* **Artifacts:** save test reports (junit.xml), coverage HTML, Terraform plans.
* **Retries/timeouts:** add timeout-minutes: and simple retry wrappers.

**Copilot prompt:**

“Convert CI job to a matrix over Node versions 18 and 20 with caching and junit test report upload.”

**10) Observability & release automation**

* **Release notes:** actions/create-release on tags.
* **SBOM:** anchore/syft-action to generate SBOM and upload.
* **Notifications:** Slack/Teams via webhook on failures.

**Copilot prompt:**

“Create a release workflow that generates changelog from PR titles and attaches SBOM and container digest.”

**11) Common adjustments by platform**

* **AWS:** replace configure-aws-credentials with your role; add CDK/SAM if preferred.
* **Azure:** use azure/login@v2 with federated credentials; azure/CLI@v2 for Web Apps or azure/aks-set-context@v4 for AKS.
* **GCP:** google-github-actions/auth@v2 with WIF; google-github-actions/deploy-cloudrun@v2 for Cloud Run.

**Copilot prompt:**

“Swap AWS steps for Azure federated login, deploy to Azure Web App with health check.”

**Minimal working starter (drop-in)**

If you want to try *now*, create just these two files and commit:

1. .github/workflows/ci.yml (from §4)
2. .github/workflows/cd-image.yml (from §5)

Open a PR → CI runs. Merge to main → image builds & pushes.

**Copilot prompt pack (copy/paste into files or Copilot Chat)**

* “Draft a CI workflow for Node with lint, tests, artifact upload, dependency review, CodeQL.”
* “Create a Docker build-and-push workflow for GHCR with metadata-action for tags.”
* “Add Terraform plan on PR and apply with environment approvals and OIDC.”
* “Add Trivy image scanning and tfsec IaC scanning; fail on critical vulns.”
* “Write Helm-based deploy job to AKS/EKS with rollout status and rollback on failure.”
* “Convert CI to matrix builds with caching.”