GitHub Actions Local Runner Worker Specification

Executive Summary

A ConfigHub worker that enables users to run GitHub Actions workflows locally using ConfigHub-managed configurations and secrets, wrapping the `act` tool for seamless integration between configuration management and CI/CD pipelines.

The Non-Obvious Insight

Traditional Flow:

```
GitHub Repo → GitHub Actions → Pull Secrets from Vault → Deploy
```

ConfigHub Flow:

```
ConfigHub → Local Actions Runner → Test with Real Configs → Validate → Deploy
```

The breakthrough: **Treat GitHub Actions workflows as configuration that can be tested locally with production-like configs before committing**.

Core Value Propositions

1. Pre-Commit Configuration Testing

```yaml
# Test your deployment workflow with actual configs BEFORE pushing
$ cub actions run deploy.yml \
  --space staging \
  --unit webapp \
  --dry-run

✓ Workflow 'deploy' started
✓ Job 'build' completed
✓ Job 'test' completed
⚠ Job 'deploy' would deploy webapp with 5 replicas
✓ Dry run complete - no changes made
```

2. ConfigHub as Secret Provider for Actions

```yaml
```

```yaml
# .github/workflows/deploy.yml
name: Deploy
on: push
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Deploy to Kubernetes
        env:
          # Secrets injected from ConfigHub, not GitHub Secrets
          KUBECONFIG: ${{ confighub.secrets.kubeconfig }}
          DB_PASSWORD: ${{ confighub.secrets.db-password }}
        run: |
          kubectl apply -f manifests/
```

3. Configuration-Driven CI/CD

yaml

```yaml
# ConfigHub unit: ci-cd-config.yaml
apiVersion: v1
kind: WorkflowConfig
metadata:
  name: webapp-deployment
spec:
  workflows:
    - name: "test-and-deploy"
      path: ".github/workflows/deploy.yml"
      when:
        - event: "config.changed"
          spaces: ["staging", "production"]
      parameters:
        environment: "{{ .Space.Name }}"
        replicas: "{{ .Config.spec.replicas }}"
```

Architecture

```
┌─────────────────┐     ┌─────────────────┐
┌─────────────┐
│             │   │             │   │           │
│  ConfigHub  │───────▶│ Actions Worker │───────▶│ act (Docker)│
│             │   │             │   │           │
│ - Configs   │   │ - Wraps act  │   │ - Runs    │
│ - Secrets   │   │ - Injects cfg │   │  workflows │
│ - Workflows │   │ - Reports back │   │           │
└─────────────────┘     └─────────────────┘
└─────────────┘
```

Implementation Design

1. Worker Configuration

```yaml
apiVersion: v1
kind: GitHubActionsWorker
metadata:
  name: actions-runner
spec:
  # Act configuration
  act:
    version: "0.2.65"
    dockerHost: "unix:///var/run/docker.sock"
    cacheDir: "/tmp/act-cache"

  # ConfigHub integration
  configHub:
    secretInjection:
      enabled: true
      method: "environment"  # or "files"
    configInjection:
      enabled: true
      format: "yaml"  # or "json", "env"

  # Execution settings
  execution:
    maxConcurrent: 5
    timeout: "30m"
    allowedImages:
      - "ubuntu-latest"
      - "node:18"
      - "python:3.11"
    resourceLimits:
      cpu: "2"
      memory: "4Gi"

  # Security
  security:
    sandboxing: "strict"
    networkPolicy: "restricted"
    secretLeakPrevention: true
```

2. CLI Integration

```bash
```

```
# Run a workflow with ConfigHub context
$ cub actions run deploy.yml \
   --space production \
   --unit webapp \
   --event push

# List available workflows
$ cub actions list
WORKFLOW                  TRIGGERS           LAST RUN
.github/workflows/ci.yml  push, pull_request 2 hours ago
.github/workflows/deploy.yml push (main)      1 day ago
deploy/special-workflow.yml manual          never

# Test workflow with specific config revision
$ cub actions test deploy.yml \
   --space staging \
   --unit webapp \
   --revision 123 \
   --verbose

# Run workflow from ConfigHub-stored definition
$ cub actions run \
   --from-unit ci-cd-workflows \
   --workflow deploy-prod
```

3. Secret and Config Injection

```go
```

```
type ActionRunner struct {
    act       ActWrapper
    configHub  ConfigHubClient
}

func (ar *ActionRunner) PrepareEnvironment(ctx RunContext) error {
    // 1. Fetch configs from ConfigHub
    configs := ar.configHub.GetConfigs(ctx.Space, ctx.Unit)

    // 2. Fetch secrets (with audit)
    secrets := ar.configHub.GetSecrets(ctx.Space, ctx.SecretRefs)

    // 3. Prepare act environment
    actEnv := map[string]string{
        // Inject as GitHub Actions secrets
        "GITHUB_TOKEN": generateSyntheticToken(),
    }

    // 4. Mount ConfigHub data
    for key, value := range configs {
        actEnv[fmt.Sprintf("INPUT_%s", strings.ToUpper(key))] = value
    }

    // 5. Inject secrets securely
    for key, secret := range secrets {
        actEnv[fmt.Sprintf("SECRET_%s", strings.ToUpper(key))] = secret
    }

    return ar.act.SetEnvironment(actEnv)
}
```

4. Workflow as Configuration

```
yaml
```

```
# Store workflows in ConfigHub as units
apiVersion: v1
kind: ConfigUnit
metadata:
  name: ci-cd-workflows
  labels:
    type: github-actions
spec:
  data: |
    name: ConfigHub-Driven Deploy
    on:
      workflow_dispatch:
        inputs:
          space:
            description: 'ConfigHub space'
            required: true
          unit:
            description: 'ConfigHub unit to deploy'
            required: true

    jobs:
      deploy:
        runs-on: ubuntu-latest
        steps:
          - name: Fetch Config from ConfigHub
            uses: confighub/fetch-config@v1
            with:
              space: ${{ github.event.inputs.space }}
              unit: ${{ github.event.inputs.unit }}

          - name: Validate Configuration
            run: |
              cub function do validate-k8s \
                --space ${{ github.event.inputs.space }} \
                --unit ${{ github.event.inputs.unit }}

          - name: Deploy
            run: |
              kubectl apply -f ${{ steps.fetch.outputs.config-path }}
```

Non-Obvious Benefits

1. Time Travel Testing

```bash
```

```
# Test how workflow would have behaved with last week's config
$ cub actions run deploy.yml \
  --space production \
  --unit webapp \
  --revision "@{1 week ago}" \
  --simulate-event push
```

2. Workflow Drift Detection

```bash
# Compare local workflow behavior vs GitHub
$ cub actions compare deploy.yml \
  --local-vs-github \
  --space production

Differences detected:
- Local: Uses ConfigHub secret 'db-password-v2'
- GitHub: Uses GitHub secret 'DB_PASSWORD' (last rotated 90 days ago)
- Local: Deployment timeout 600s
- GitHub: Deployment timeout 300s
```

3. Config-Triggered Workflows

```yaml
# Run workflows when configs change
apiVersion: v1
kind: WorkflowTrigger
metadata:
 name: auto-test-on-config-change
spec:
 watch:
  - space: staging
   units: ["webapp", "api"]
 on:
  - event: config.changed
   run: ".github/workflows/integration-test.yml"
  - event: config.approved
   run: ".github/workflows/deploy.yml"
```

4. Composite Actions from Multiple Sources

```bash
```

```
# Combine GitHub Actions with ConfigHub functions
$ cub actions compose \
  --github-workflow ci.yml \
  --confighub-function validate-security \
  --confighub-function scan-secrets \
  --output secure-ci.yml
```

Security Considerations

1. Sandboxed Execution

yaml

```yaml
security:
  isolation:
    type: "gvisor"  # or "firecracker"
    networkPolicy:
      ingress: ["github.com", "confighub.io"]
      egress: ["github.com", "registry.docker.com"]
  secrets:
    leakPrevention:
      scanOutput: true
      redactPatterns:
        - "password"
        - "token"
        - "key"
```

2. Audit Trail

json

```json
{
  "event": "workflow.executed",
  "workflow": ".github/workflows/deploy.yml",
  "trigger": "manual",
  "user": "alice@example.com",
  "configs": ["webapp:v123", "api:v456"],
  "secrets_accessed": ["db-password", "api-key"],
  "result": "success",
  "duration": "5m32s"
}
```

Advanced Scenarios

1. Matrix Testing with Config Variants

bash

```
# Test workflow against multiple config variants
$ cub actions matrix-test deploy.yml \
  --space-matrix "dev,staging,prod" \
  --unit webapp \
  --parallel 3

Running matrix tests:
✓ dev: Success (2m31s)
✓ staging: Success (2m45s)
✕ prod: Failed - resource limits exceeded
```

2. Workflow Optimization

```bash
# Analyze workflow performance with different configs
$ cub actions analyze deploy.yml \
  --optimize-for speed

Recommendations:
- Cache Docker layers: 40% speed improvement
- Parallelize test jobs: 25% speed improvement
- Pre-pull images: 15% speed improvement
- Use ConfigHub cache: 10% speed improvement
```

3. GitOps Preview

```bash
# See what GitOps would do without GitHub
$ cub actions preview gitops-sync.yml \
  --source-repo github.com/myorg/configs \
  --target-space production

Would synchronize:
- webapp: 5 config changes
- api: 2 config changes
- database: No changes
- secrets: 3 rotations pending
```

Why This Is Brilliant

1. **Shifts Testing Left**: Test CI/CD with real configs before committing
2. **Unifies Config & CI/CD**: ConfigHub becomes single source of truth for both
3. **Enables Local GitOps**: Full GitOps workflows without GitHub/GitLab
4. **Secrets Stay Secure**: Never commit secrets, always inject from ConfigHub
5. **Time Travel CI/CD**: Test workflows with historical configurations

Implementation Path

Phase 1: Basic Integration

- Wrap `act` with ConfigHub authentication
- Basic secret/config injection
- Simple CLI commands

Phase 2: Advanced Features

- Workflow storage in ConfigHub
- Matrix testing
- Config-triggered workflows

Phase 3: Enterprise Features

- Audit trail
- Compliance scanning
- Resource governance
- Multi-tenancy

This creates a powerful new paradigm: **Configuration-Driven CI/CD** where your deployment pipelines are tested with the same rigor as your configurations.