GitHub Actions CI/CD Pipeline Architecture - 6 Workflows Implementation

CI/CD TRIGGER EVENTS

| ☁️ push (main) Triggers: ci.yml, deploy.yml | ☁️ push (develop) Triggers: ci.yml only | 🔄 pull_request Triggers: ci.yml only | 🖥️ workflow_dispatch Triggers: deploy.yml manual | ⏰ schedule (cron) Triggers: test-automation.yml | 🛡️ schedule (weekly) Triggers: security-monitoring.yml | 🏷️ push (tags) Triggers: release-management.yml |

activates

CI.YML - CONTINUOUS INTEGRATION WORKFLOW

🧪 Test Job
- Checkout code
- Setup Node.js 20
- npm ci
- npm run lint
- npm test
- npm run format:check
- Upload test results

📦 Build Job
- Checkout code
- Setup Docker Buildx
- Build Docker image
- Test container startup
- Health check validation
- Build success verification

🔍 Security Job
- Checkout code
- Setup Node.js 20
- npm ci
- npm audit --audit-level high
- Continue on error
- Report vulnerabilities

✅ Quality Gates
- All tests must pass
- Linting must pass
- Format check must pass
- Build must succeed
- Security scan completion
- Parallel job execution

📋 CI Artifacts
- Test coverage reports
- Build logs
- Security scan results
- Docker build success
- Quality metrics
- Job execution times

🔔 Notifications
- PR status checks
- Build status badges
- Slack/email alerts
- GitHub commit status
- Merge blocking on failure
- Success confirmations

triggers on success

DEPLOY.YML - GOOGLE CLOUD RUN DEPLOYMENT WORKFLOW

🔐 GCP Authentication
- Service account setup
- gcloud CLI configuration
- Docker auth helper
- Project ID validation
- Permissions verification

setup

📦 Artifact Registry
- Create repository
- us-central1 region
- Docker format setup
- Registry authentication
- Image storage preparation

prepare

📦 Image Build & Push
- Docker build with build args
- Multi-environment tagging
- NODE_ENV configuration
- Commit SHA tagging
- Registry push operation

build & push

☁️ Cloud Run Deployment
- Environment selection
- Resource allocation (512Mi, 1CPU)
- Instance scaling (0-10)
- Timeout configuration (300s)
- Traffic allocation

deploy

🔎 Deployment Testing
- Health check validation
- Endpoint testing (/ and /health)
- Response verification
- Performance baseline
- Service availability check

verify & store

📊 Deployment Info Storage
- Custom action usage
- Cloud Storage upload
- Deployment metadata
- Service URLs
- Version information

deploys to environments

MULTI-ENVIRONMENT DEPLOYMENT STRATEGY

🖥️ Development Environment
- Service: simple-webapp-dev
- Trigger: Push to main branch
- Auto-deployment enabled
- Development configurations
- Immediate feedback loop

🖥️ Testing Environment
- Service: simple-webapp-test
- Trigger: Manual dispatch only
- QA testing environment
- Staging configurations
- Pre-production validation

🔴 Production Environment
- Service: simple-webapp-prod
- Trigger: Manual dispatch only
- Production configurations
- Stable release deployments
- Enhanced monitoring

🔒 Environment Isolation
- Separate Cloud Run services
- Environment-specific configs
- Independent scaling
- Isolated networking
- Resource segregation

ADDITIONAL SPECIALIZED WORKFLOWS (4 WORKFLOWS)

🧪 test-automation.yml
- Scheduled testing (daily)
- Performance benchmarks
- Load testing simulation
- Test data management
- Cloud Storage uploads
- Custom action usage

🛡️ security-monitoring.yml
- Weekly vulnerability scans
- npm audit comprehensive
- License compliance check
- Dependency updates
- Security report generation
- Alert notifications

🎉 release-management.yml
- Git tag creation
- Release notes generation
- Changelog automation
- Asset compilation
- Version bump handling
- Distribution preparation

⚙️ advanced-cicd.yml
- Complex deployment logic
- Blue-green deployment prep
- Rollback mechanisms
- Advanced environment mgmt
- Canary deployment support
- Traffic management

🎬 Custom Actions (.github/actions/)
- manage-test-data/action.yml
- store-deployment-info/action.yml
- Reusable workflow components
- Parameter-driven execution
- Cross-workflow functionality

triggers weekly

triggers daily

WORKFLOW ORCHESTRATION & DEPENDENCIES

🪝 Workflow Dependencies
- CI must pass before deployment
- Security scans run independently
- Test automation runs on schedule
- Release management triggered by tags
- Advanced CI/CD for complex scenarios

⚡ Parallel Execution
- CI jobs run concurrently
- Multiple environment deployments
- Independent workflow triggers
- Resource optimization
- Faster feedback loops

⚠️ Failure Handling
- Automatic retry mechanisms
- Graceful degradation
- Error notification systems
- Rollback capabilities
- Debug information collection

📊 Monitoring & Feedback
- Real-time workflow status
- Performance metrics collection
- Success/failure rates
- Deployment frequency tracking
- Quality trend analysis

📋 CI/CD PIPELINE OVERVIEW: GitHub Actions implementation with 6 workflows and 2 custom actions
- Automated testing, building, security scanning • Multi-environment Google Cloud Run deployment • Scheduled automation and release management
- DevOps automation with parallel execution • Quality gates and failure handling • Monitoring and feedback loops