DevPipeline Quick Start Guide

Document Number: PROD-DP-001 **Product:** DevPipeline **Version:** 2.x **Last Updated:** December 1, 2023 **Owner:** DevPipeline Product Team

Introduction

DevPipeline is NovaTech's CI/CD platform that automates building, testing, and deploying your applications. It integrates seamlessly with CloudForge and popular version control systems.

What is DevPipeline?

DevPipeline provides: - **Automated builds:** Build on every push, PR, or schedule - **Parallel testing:** Run tests across multiple environments simultaneously - **Deployment automation:** Deploy to CloudForge, Kubernetes, or custom targets - **Built-in security:** Secrets management, SAST scanning, dependency checking - **Visual pipeline editor:** Create pipelines with code or visual interface

Quick Start (5 Minutes)

Step 1: Connect Your Repository

1. Go to app.devpipeline.io
2. Click **"New Project"**
3. Select your Git provider (GitHub, GitLab, Bitbucket)
4. Authorize access
5. Select your repository

Step 2: Create a Pipeline

DevPipeline auto-detects common project types and suggests a pipeline.

Auto-detected? Review and click "Create Pipeline"

Manual setup? Create `.devpipeline.yaml` in your repository root:

```yaml
name: my-app

triggers:
  - push:
      branches: [main, develop]
```

```yaml
  - pull_request:
      branches: [main]

stages:
  - name: build
    steps:
      - name: Install dependencies
        run: npm install

      - name: Build
        run: npm run build

  - name: test
    steps:
      - name: Run tests
        run: npm test

      - name: Upload coverage
        uses: devpipeline/coverage@v1
        with:
          file: coverage/lcov.info

  - name: deploy
    when: branch == "main"
    steps:
      - name: Deploy to CloudForge
        uses: cloudforge/deploy@v2
        with:
          environment: production
```

Step 3: Push and Watch

Commit your `.devpipeline.yaml` and push:

```
git add .devpipeline.yaml
git commit -m "Add DevPipeline configuration"
git push
```

Your first build starts automatically! View it at: - Dashboard: app.devpipeline.io - Build page: Click notification in PR (if applicable)

Core Concepts

Pipelines

A **pipeline** defines your entire CI/CD workflow: - Triggered by events (push, PR, schedule) - Contains multiple stages - Runs in isolated environments

Stages

Stages are groups of related steps: - Run sequentially by default - Can run in parallel (with `parallel: true`) - Can have conditions (`when:`)

Steps

Steps are individual commands or actions: - Run commands directly (`run:`) - Use pre-built actions (`uses:`) - Can have inputs and outputs

Actions

Actions are reusable pipeline components: - Built by NovaTech (official actions) - Community-contributed - Create your own (private or public)

Pipeline Configuration Reference

Triggers

```
triggers:
  # On push to specific branches
  - push:
      branches: [main, develop]
      paths:
        - "src/**"
        - "package.json"

  # On pull requests
  - pull_request:
      branches: [main]

  # On schedule (cron)
  - schedule:
      cron: "0 2 * * *"  # Daily at 2am UTC

  # Manual trigger
```

```
  - manual:
      inputs:
        environment:
          type: choice
          options: [staging, production]
```

Environment Variables

```
env:
  NODE_ENV: production
  CI: true

stages:
  - name: build
    env:
      BUILD_NUMBER: ${{ pipeline.number }}
    steps:
      - run: echo "Build $BUILD_NUMBER"
```

Secrets

Store secrets in DevPipeline settings, reference in pipeline:

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

Conditions

```
stages:
  - name: deploy-production
    when: |
      branch == "main" &&
      event == "push"
    steps:
      - uses: cloudforge/deploy@v2

  - name: deploy-staging
    when: branch == "develop"
    steps:
      - uses: cloudforge/deploy@v2
```

```
        with:
          environment: staging
```

Matrix Builds

Run the same steps across multiple configurations:

```
stages:
  - name: test
    matrix:
      node: [16, 18, 20]
      os: [ubuntu, macos]
    steps:
      - uses: devpipeline/node@v2
        with:
          version: ${{ matrix.node }}
      - run: npm test
```

Parallel Stages

```
stages:
  - name: lint
    parallel: true
    steps:
      - run: npm run lint

  - name: test
    parallel: true
    steps:
      - run: npm test

  - name: security-scan
    parallel: true
    steps:
      - uses: devpipeline/security-scan@v1
```

Artifacts

Upload and share files between stages or download later:

```
stages:
  - name: build
    steps:
```

```
      - run: npm run build
      - uses: devpipeline/upload-artifact@v1
        with:
          name: dist
          path: ./dist

  - name: deploy
    steps:
      - uses: devpipeline/download-artifact@v1
        with:
          name: dist
      - run: ./deploy.sh dist/
```

Common Pipeline Patterns

Node.js Application

```
name: node-app

stages:
  - name: setup
    steps:
      - uses: devpipeline/checkout@v2
      - uses: devpipeline/node@v2
        with:
          version: "20"
          cache: npm

  - name: test
    steps:
      - run: npm ci
      - run: npm run lint
      - run: npm test

  - name: build
    steps:
      - run: npm run build
      - uses: devpipeline/upload-artifact@v1
        with:
          name: build
          path: ./dist

  - name: deploy
    when: branch == "main"
```

```
    steps:
      - uses: cloudforge/deploy@v2
```

Python Application

```yaml
name: python-app

stages:
  - name: test
    steps:
      - uses: devpipeline/python@v2
        with:
          version: "3.11"
      - run: pip install -r requirements.txt
      - run: pytest --cov=.

  - name: deploy
    when: branch == "main"
    steps:
      - uses: cloudforge/deploy@v2
```

Docker Build and Push

```yaml
name: docker-build

stages:
  - name: build
    steps:
      - uses: devpipeline/docker@v2
        with:
          registry: registry.novatech.io
          username: ${{ secrets.REGISTRY_USER }}
          password: ${{ secrets.REGISTRY_PASSWORD }}

      - run: |
          docker build -t registry.novatech.io/my-app:${{ git.sha }} .
          docker push registry.novatech.io/my-app:${{ git.sha }}
```

Integrations

CloudForge

Native integration for deploying to CloudForge:

```yaml
- uses: cloudforge/deploy@v2
  with:
    project: my-app
    environment: production
    wait: true  # Wait for deployment to complete
```

SecureVault

Fetch secrets from SecureVault:

```yaml
- uses: securevault/fetch@v1
  with:
    secret: production/database
    output: DATABASE_URL
```

Notifications

```yaml
notifications:
  slack:
    channel: "#deploys"
    on: [success, failure]

  email:
    to: team@novatech.com
    on: [failure]
```

CLI

Installation

```
npm install -g @novatech/devpipeline-cli
# or
brew install devpipeline
```

Common Commands

```
# Trigger a build
dp run

# View build status
dp status
```

```
# View logs
dp logs

# Validate pipeline configuration
dp validate

# List recent builds
dp builds

# Cancel a build
dp cancel <build-id>
```

Security Features

Secret Scanning

DevPipeline automatically scans for accidentally committed secrets.

Dependency Scanning

Check for known vulnerabilities:

```
- uses: devpipeline/dependency-scan@v1
  with:
    fail-on: high
```

SAST (Static Analysis)

```
- uses: devpipeline/sast@v1
```

Troubleshooting

Build Fails Immediately

- Check `.devpipeline.yaml` syntax
- Run `dp validate` locally
- Review error message in build logs

Slow Builds

- Enable caching for dependencies
- Use parallel stages where possible
- Consider self-hosted runners for large builds

Deployment Fails

- Check secrets are configured
- Verify CloudForge/target credentials
- Review deployment logs

Getting Help

- **Documentation:** docs.devpipeline.io
- **Community:** community.devpipeline.io
- **Support:** support@devpipeline.io
- **Internal NovaTech:** #devpipeline-help on Slack

Next Steps

1. Pipeline Configuration Reference
2. Actions Marketplace
3. Self-Hosted Runners
4. CloudForge Integration
5. Security Best Practices

Related Documents: Pipeline Configuration Reference (PROD-DP-010), CloudForge Integration (PROD-DP-040), Actions Reference (PROD-DP-025)