DevPipeline Getting Started Guide

Document ID: PRD-DP-001 **Last Updated:** 2024-02-28 **Owner:** DevPipeline Product Team **Classification:** Public

---

Welcome to DevPipeline

DevPipeline is NovaTech's continuous integration and delivery platform. Build, test, and deploy your applications with confidence using powerful automation, built-in security scanning, and seamless integrations.

Quick Start Checklist

- ☐ Create or connect your NovaTech account
- ☐ Connect your Git repository
- ☐ Create your first pipeline
- ☐ Run your first build
- ☐ Set up notifications

Step 1: Account Setup

New Users

1. Visit devpipeline.novatech.com
2. Click **Start Free Trial**
3. Create account or sign in with Google/GitHub
4. Complete profile setup
5. Enable two-factor authentication

Existing NovaTech Users

If you have a NovaTech account (CloudForge, SecureVault, DataLens):

1. Visit devpipeline.novatech.com
2. Click **Sign In**

3. Use existing credentials
4. DevPipeline appears in your product dashboard

Step 2: Connect Your Repository

Supported Providers

- GitHub (Cloud & Enterprise)
- GitLab (Cloud & Self-Hosted)
- Bitbucket (Cloud & Server)
- Azure DevOps

GitHub Connection

1. Go to **Settings → Integrations → Git Providers**
2. Click **Connect GitHub**
3. Authorize DevPipeline app
4. Select repositories to connect (all or specific)
5. Click **Save**

GitLab Connection

1. Go to **Settings → Integrations → Git Providers**
2. Click **Connect GitLab**
3. Enter GitLab URL (for self-hosted)
4. Create Personal Access Token with `api` scope
5. Enter token in DevPipeline
6. Select repositories
7. Click **Save**

Repository Permissions

DevPipeline needs: - Read repository contents - Read/write commit status - Create webhooks - Read pull requests

---

Step 3: Create Your First Pipeline

Option 1: Auto-Detection

1. Click **New Pipeline**
2. Select your repository
3. DevPipeline analyzes your code
4. Accept suggested pipeline configuration
5. Click **Create Pipeline**

DevPipeline auto-detects: - Node.js (package.json) - Python (requirements.txt, pyproject.toml) - Go (go.mod) - Java (pom.xml, build.gradle) - Docker (Dockerfile) - And more…

Option 2: Manual Configuration

1. Click **New Pipeline**
2. Select your repository
3. Click **Manual Configuration**
4. Add a `.devpipeline.yaml` file to your repo:

```yaml
# .devpipeline.yaml
name: my-app-pipeline

trigger:
  branches:
    - main
    - develop
  pull_request: true

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

      - name: Run tests
        run: npm test

      - name: Build
        run: npm run build

  - name: deploy
    when:
```

```
    branch: main
  steps:
    - name: Deploy to production
      run: npm run deploy
```

5. Commit and push the file
6. DevPipeline automatically creates the pipeline

Step 4: Run Your First Build

Manual Trigger

1. Go to your pipeline
2. Click **Run Pipeline**
3. Select branch
4. Click **Start**

Automatic Trigger

Pipelines run automatically on: - Push to configured branches - Pull request creation/update - Scheduled triggers (if configured)

View Build Progress

1. Click on the running build
2. Watch real-time logs
3. View stage progress
4. See test results

Build Status

Status	Description
Running	Build in progress
Passed	All steps completed successfully
Failed	One or more steps failed
Pending	Waiting for approval or resources
Cancelled	Manually cancelled

Step 5: Set Up Notifications

Email Notifications

1. Go to **Settings → Notifications**
2. Enable **Email**
3. Configure triggers:
   - Build failed
   - Build recovered
   - Deployment completed

Slack Integration

1. Go to **Settings → Integrations → Slack**
2. Click **Add to Slack**
3. Select channel
4. Configure notification preferences

Example Slack Message

```
 Pipeline Failed: my-app-pipeline
Branch: main
Commit: abc1234 - "Fix login bug"
Author: john@example.com
Failed stage: test
View: https://devpipeline.novatech.com/builds/12345
```

Pipeline Configuration

Basic Structure

```yaml
name: pipeline-name

# When to run
trigger:
  branches: [main, develop]
  pull_request: true
  schedule:
    - cron: "0 0 * * *"  # Daily at midnight

# Environment variables
```

```yaml
env:
  NODE_ENV: production

# Global settings
settings:
  timeout: 30  # minutes

# Pipeline stages
stages:
  - name: stage-name
    steps:
      - name: step-name
        run: command
```

Common Patterns

Node.js Application:

```yaml
stages:
  - name: build
    image: node:18
    cache:
      paths:
        - node_modules/
    steps:
      - run: npm ci
      - run: npm run lint
      - run: npm test
      - run: npm run build
```

Python Application:

```yaml
stages:
  - name: build
    image: python:3.11
    cache:
      paths:
        - .venv/
    steps:
      - run: pip install -r requirements.txt
      - run: pytest
      - run: python -m build
```

Docker Build:

```yaml
stages:
  - name: build
    steps:
      - name: Build image
        run: docker build -t myapp:${{ pipeline.commit_sha }} .

      - name: Push to registry
        run: docker push myapp:${{ pipeline.commit_sha }}
```

Built-in Features

Security Scanning

DevPipeline includes automatic security scanning:

```yaml
stages:
  - name: security
    steps:
      - name: SAST scan
        uses: security/sast

      - name: Dependency scan
        uses: security/dependencies

      - name: Secret scan
        uses: security/secrets
```

Test Reporting

Upload test results for visualization:

```yaml
- name: Run tests
  run: npm test -- --coverage

- name: Upload results
  uses: test-results
  with:
    format: junit
    path: coverage/junit.xml
```

Artifacts

Save build outputs:

```yaml
- name: Build
  run: npm run build

- name: Save artifacts
  artifacts:
    - path: dist/
      name: build-output
      retention: 30  # days
```

Caching

Speed up builds with caching:

```yaml
cache:
  - key: npm-${{ hash('package-lock.json') }}
    paths:
      - node_modules/
```

Environments and Deployment

Define Environments

1. Go to **Settings → Environments**
2. Click **Add Environment**
3. Configure:
    - Name (e.g., production, staging)
    - Type (Production, Staging, Development)
    - Approvals required
    - Environment variables

Deploy to Environment

```yaml
stages:
  - name: deploy-staging
    environment: staging
    steps:
```

```
      - run: ./deploy.sh staging

  - name: deploy-production
    environment: production
    approval: required
    steps:
      - run: ./deploy.sh production
```

Approval Workflow

For production deployments: 1. Build completes staging deployment 2. Approver receives notification 3. Reviews changes in UI 4. Clicks **Approve** or **Reject** 5. Approved → deployment proceeds

---

Secrets Management

Add Secrets

1. Go to **Settings → Secrets**
2. Click **Add Secret**
3. Enter name and value
4. Select scope (organization, repository, environment)
5. Click **Save**

Use Secrets in Pipeline

```
stages:
  - name: deploy
    steps:
      - name: Deploy
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        run: aws s3 sync dist/ s3://my-bucket/
```

SecureVault Integration

For advanced secret management:

```yaml
stages:
  - name: deploy
    steps:
      - name: Get secrets from SecureVault
        uses: securevault/get-secrets
        with:
          path: secret/data/myapp
```

Runners

Cloud Runners (Default)

DevPipeline provides managed cloud runners: - Pre-configured environments - Auto-scaling - Multiple OS options (Linux, Windows, macOS)

Self-Hosted Runners

For special requirements:

1. Go to **Settings → Runners**
2. Click **Add Self-Hosted Runner**
3. Follow installation instructions
4. Configure runner tags

```yaml
stages:
  - name: build
    runner:
      tags: [self-hosted, gpu]
    steps:
      - run: ./train-model.sh
```

Getting Help

Resources

- **Documentation:** docs.devpipeline.novatech.com
- **Examples:** github.com/novatech/devpipeline-examples
- **Community:** community.novatech.com
- **Status:** status.novatech.com

Support

Plan	Support Level
Free	Community, docs
Team ($15/user/mo)	Email support
Enterprise	Priority support, dedicated CSM

Contact

- **Email:** support@novatech.com
- **In-app:** Help → Contact Support

Next Steps

1. **Add more stages:** Testing, security scans, deployment
2. **Set up environments:** Staging, production
3. **Configure notifications:** Slack, email
4. **Explore integrations:** CloudForge, SecureVault, Jira
5. **Invite your team:** Share access with developers

Related Documents: Pipeline YAML Reference (PRD-DP-020), Runner Configuration (PRD-DP-030), CI/CD Best Practices (PRD-DP-010)