DevPipeline FAQs

Document ID: FAQ-PRD-DP-001 **Last Updated:** March 2024 **Owner:** DevPipeline Product Team **Category:** Product - DevPipeline

---

Getting Started

Q: What is DevPipeline?

A: DevPipeline is NovaTech's CI/CD platform for automating software builds, tests, and deployments. Key features include: - Continuous integration and deployment - Parallel and distributed builds - Security scanning - Mobile app builds (iOS/Android) - Integration with GitHub, GitLab, Bitbucket

---

Q: How do I connect my repository?

A: 1. Go to **Settings → Repositories** 2. Click **Connect Repository** 3. Select your provider (GitHub, GitLab, Bitbucket) 4. Authorize DevPipeline 5. Select repositories to connect

DevPipeline will automatically detect pipeline configurations.

Q: Where do I put my pipeline configuration?

A: Create a `pipeline.yaml` file in your repository root (or `.devpipeline/pipeline.yaml`):

```yaml
# pipeline.yaml
stages:
  - build
  - test
  - deploy

jobs:
  build:
    stage: build
    script:
      - npm install
      - npm run build

  test:
    stage: test
    script:
      - npm test

  deploy:
    stage: deploy
    script:
      - npm run deploy
    only:
      - main
```

Q: What languages and frameworks are supported?

A: DevPipeline supports all major languages:

Language	Built-in Support
JavaScript/TypeScript	npm, yarn, pnpm
Python	pip, pipenv, poetry
Java	Maven, Gradle
Go	Go modules
Ruby	Bundler
.NET	dotnet CLI
Rust	Cargo
PHP	Composer

And many more via custom configurations.

Pipelines

Q: What triggers a pipeline run?

A: Pipelines can be triggered by: - **Push** - Code pushed to repository - **Pull request** - PR opened or updated - **Schedule** - Cron-based schedules - **Manual** - User-initiated - **API** - External trigger via API - **Tag** - Git tag created

Configure triggers in your pipeline.yaml:

```
on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]
  schedule:
    - cron: "0 2 * * *"
```

Q: How do I run jobs in parallel?

A: Jobs in the same stage run in parallel by default:

```
stages:
  - test

jobs:
  test-unit:
    stage: test
    script: npm run test:unit

  test-integration:
    stage: test
    script: npm run test:integration

  test-e2e:
    stage: test
    script: npm run test:e2e
```

All three test jobs run simultaneously.

Q: How do I pass data between jobs?

A: Use artifacts:

```
jobs:
  build:
    script:
      - npm run build
    artifacts:
      paths:
        - dist/

  deploy:
    needs: [build]
    script:
      - deploy dist/
```

The `deploy` job receives the `dist/` folder from `build`.

Q: How long are build logs retained?

A: Log retention by plan:

Plan	Retention
Starter	30 days
Professional	90 days
Enterprise	1 year+

Artifacts have separate retention settings.

Build Minutes

Q: How are build minutes calculated?

A: Build minutes = runtime × parallel jobs

Example: - 3 jobs running for 10 minutes each - If sequential: 30 minutes used - If parallel: 10 minutes used

Parallel execution saves build minutes!

Q: How many build minutes do I have?

A:

Plan	Monthly Minutes
Starter	1,000
Professional	10,000
Enterprise	Custom

Check usage at **Settings** → **Usage**.

Q: What happens if I run out of minutes?

A: Options: - Pipelines queue until next month - Purchase additional minutes - Upgrade plan - Use self-hosted runners (unlimited)

Secrets and Variables

Q: How do I add secrets to my pipeline?

A: 1. Go to **Settings** → **Secrets** 2. Click **Add Secret** 3. Enter name and value 4. Secret is encrypted at rest

Use in pipeline:

```
jobs:
  deploy:
    script:
      - deploy --token ${{ secrets.DEPLOY_TOKEN }}
```

Q: What's the difference between secrets and variables?

A:

Type	Use Case	Visibility
Secrets	Sensitive data (tokens, keys)	Masked in logs
Variables	Non-sensitive config	Visible in logs

Q: Can I use SecureVault for secrets?

A: Yes! Native integration:

```
jobs:
  deploy:
    secrets:
      from: securevault
      path: secret/data/deploy
    script:
      - deploy --token $DB_PASSWORD
```

Caching

Q: How does caching work?

A: Cache dependencies between builds:

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

jobs:
  build:
    script:
      - npm install  # Uses cache if available
      - npm run build
```

Q: How long is cache retained?

A: Cache retention: - Default: 7 days since last use - Maximum: 30 days - Size limit: 5GB per cache key

---

Q: Why isn't my cache being used?

A: Common causes: - Cache key changed (dependency file updated) - Cache expired - Different branch (default: branch-specific) - Cache size exceeded

Check cache hit rate in build logs.

Testing

Q: How do I run tests in parallel?

A: Split tests across parallel runners:

```yaml
jobs:
  test:
    parallel: 4
    script:
      - npm test -- --shard=$CI_NODE_INDEX/$CI_NODE_TOTAL
```

DevPipeline automatically distributes tests.

Q: How do I get test reports?

A: Upload test results:

```yaml
jobs:
  test:
    script:
      - npm test -- --reporter=junit --output=results.xml
    artifacts:
      reports:
        junit: results.xml
```

Results appear in the PR and pipeline UI.

Q: Can DevPipeline run browser tests?

A: Yes! Use built-in browsers:

```yaml
jobs:
  e2e:
    services:
      - chrome
      - firefox
    script:
      - npm run test:e2e
```

Or integrate with BrowserStack/Sauce Labs.

Deployments

Q: How do I deploy to production?

A: Example deployment:

```yaml
jobs:
  deploy-production:
    stage: deploy
    environment:
      name: production
      url: https://app.example.com
    script:
      - npm run deploy:prod
    only:
      - main
    when: manual  # Require manual approval
```

Q: Can I require approval before deployment?

A: Yes, use manual approval:

```
jobs:
  deploy-production:
    when: manual
    allow_failure: false
```

Or use environment protection rules in Settings.

Q: How do I rollback a deployment?

A: 1. Go to **Deployments** 2. Find previous successful deployment 3. Click **Redeploy**

Or trigger from CLI:

```
devpipeline deploy rollback --environment production
```

Security

Q: Does DevPipeline scan for vulnerabilities?

A: Yes! Enable security scanning:

```
security:
  sast: true
  sca: true
  container: true
  secrets: true
```

Results appear in Security tab and PR comments.

Q: How do I block PRs with security issues?

A: Configure security quality gates:

```yaml
security:
  quality_gate:
    enabled: true
    rules:
      - type: sast
        severity: high
        max_allowed: 0
```

PRs with high-severity issues cannot merge.

Q: Are my builds isolated?

A: Yes: - Each build runs in fresh container - No shared state between builds - Secrets are scoped appropriately - Network isolation available

---

Self-Hosted Runners

Q: Can I use my own build machines?

A: Yes, self-hosted runners are supported:

```bash
# Install runner
devpipeline runner install --token $RUNNER_TOKEN

# Start runner
devpipeline runner start
```

Benefits: - Unlimited build minutes - Access to internal resources - Custom hardware (GPU, etc.)

Q: What are the runner requirements?

A: - Linux, macOS, or Windows - 2+ CPU cores, 4GB+ RAM - Docker (for container builds) - Network access to DevPipeline

---

Integrations

Q: Does DevPipeline integrate with Slack?

A: Yes! Get notifications:

1. Go to **Settings** → **Integrations** → **Slack**
2. Connect your workspace
3. Select notification preferences

You'll receive alerts for build failures, deployments, etc.

Q: Can I trigger pipelines from external tools?

A: Yes, via API:

```
curl -X POST https://api.devpipeline.novatech.com/v1/pipelines/trigger \
  -H "Authorization: Bearer $TOKEN" \
  -d '{"ref": "main"}'
```

Or use webhooks for event-driven triggers.

Troubleshooting

Q: My build is failing. How do I debug?

A: 1. Check build logs for error messages 2. Look at the failing step 3. Use SSH debug mode for interactive debugging:

```yaml
jobs:
  build:
    debug:
      ssh: true
```

4. Check recent changes that might have caused failure

Q: My build is slow. How do I speed it up?

A: Optimization tips: 1. **Enable caching** for dependencies 2. **Run jobs in parallel** where possible 3. **Use smaller base images** 4. **Skip unnecessary steps** in feature branches 5. **Use incremental builds** when available

---

Q: Builds are queuing for a long time. Why?

A: Possible causes: - Out of build minutes - Self-hosted runners offline - High concurrent build load - Resource constraints

Check **Settings** → **Usage** for details.

Billing

Q: How is DevPipeline billed?

A: Based on: - Build minutes used - Number of users - Storage for artifacts/cache - Support tier

Build minutes are the primary cost driver.

Q: Can I buy more build minutes?

A: Yes: - Additional minutes available for purchase - Or upgrade to higher plan - Or use self-hosted runners (unlimited)

Contact

- **Documentation:** docs.novatech.com/devpipeline
- **Support:** support@novatech.com
- **Slack:** #devpipeline-help

Last reviewed: March 2024