Engineering FAQs

Document Number: FAQ-ENG-001 **Last Updated:** August 2024 **Owner:** Engineering Operations

---

Development Environment

Q: How do I set up my development environment?

A: See the Development Environment Setup Guide (IT-SW-002). Key steps: 1. Install Homebrew (Mac) or WSL (Windows) 2. Clone the dev setup repo: `git clone git@github.com:novatech/dev-setup.git` 3. Run `./setup.sh` which installs required tools 4. Follow product-specific README for your team

Q: What IDE should I use?

A: Your choice! Most engineers use: - **VS Code** (most popular, free) - **JetBrains IDEs** (IntelliJ, PyCharm, GoLand) - licenses available - **Vim/Neovim** (with our config repo)

Request JetBrains license via IT if needed.

Q: How do I get access to GitHub repos?

A: 1. Link your GitHub account to NovaTech org (invite sent during onboarding) 2. Enable SSO at github.com/orgs/novatech 3. Request team-specific access from your manager 4. Set up SSH keys per the GitHub Access Policy

Q: Where are the coding style guides?

A: Check the engineering wiki on Notion: - Go: `docs/style-guides/go.md` - Python: `docs/style-guides/python.md` - JavaScript/TypeScript: `.eslintrc` + Prettier config - All: We use automated formatters in CI

Code and Git

Q: What's the branching strategy?

A: We use trunk-based development: - `main` is the primary branch (always deployable) - Feature branches are short-lived (1-3 days max) - Branch naming: `feature/JIRA-123-short-description` - PRs required for all changes

Q: How do I create a pull request?

A: 1. Push your branch: `git push -u origin feature/JIRA-123-description` 2. Open PR on GitHub 3. Fill out the PR template (required) 4. Request review from your team 5. At least 1 approval required 6. CI must pass before merge

Q: What if CI fails on my PR?

A: Check the failing job: - **Linting:** Run `make lint` locally and fix issues - **Tests:** Run `make test` locally to debug - **Security scan:** Review findings, fix or add exceptions - **Build:** Check logs for compile errors

Q: How do I handle merge conflicts?

A:

```
git fetch origin
git rebase origin/main
# Resolve conflicts in your editor
git add .
git rebase --continue
git push --force-with-lease
```

Q: What's the commit message format?

A: We follow Conventional Commits:

```
type(scope): short description

Longer description if needed.

JIRA-123
```

Types: `feat`, `fix`, `docs`, `style`, `refactor`, `test`, `chore`

Testing

Q: What's the testing philosophy?

A: We follow the testing pyramid: - **Unit tests:** Fast, isolated, lots of them - **Integration tests:** Test component interactions - **E2E tests:** Critical paths only

Target coverage: 80%+ for new code.

Q: How do I run tests locally?

A:

```
# All tests
make test

# Unit tests only
make test-unit

# Integration tests (requires docker)
make test-integration

# Specific test file
go test ./pkg/api/... -v
```

Q: How do I write tests for my code?

A: Each project has examples. Generally: - Put tests in `*_test.go` (Go) or `test_*.py` (Python) files - Use table-driven tests for multiple cases - Mock external dependencies - See `docs/testing.md` in each repo

Q: What about test data?

A: - Use factories/fixtures for test data - Never use production data in tests - Seed data scripts in `testdata/` directories - Database tests use test containers

---

Deployment

Q: How does deployment work?

A: 1. PR merged to `main` 2. CI builds and tests 3. Automatic deploy to staging 4. Manual approval for production 5. Canary rollout (10% → 50% → 100%)

Q: How do I deploy to staging?

A: Automatic! When your PR merges to `main`, it deploys to staging within 10-15 minutes.

Q: How do I deploy to production?

A: 1. Verify staging looks good 2. Go to DevPipeline dashboard 3. Find your deployment 4. Click "Promote to Production" 5. Requires approval from team lead

Q: What if something goes wrong in production?

A: 1. **Don't panic** 2. Check #incidents Slack channel 3. If you need to rollback: DevPipeline → Deployments → Rollback 4. Page on-call if needed: `/pd trigger` in Slack 5. Document in incident channel

Q: How do feature flags work?

A: We use LaunchDarkly for feature flags:

```
if featureFlags.IsEnabled("new-dashboard", user) {
    showNewDashboard()
}
```

Create flags in LaunchDarkly dashboard, target specific users/segments first.

Infrastructure

Q: How do I access AWS?

A:

```
# Configure AWS SSO
aws configure sso

# SSO start URL: https://novatech.awsapps.com/start
# Region: us-west-2

# Login
aws sso login --profile novatech-dev
```

Q: What Kubernetes clusters are there?

A:

Cluster	Purpose	Access
dev-cluster	Development	All engineers
staging-cluster	Staging	All engineers
prod-us-west	Production US	On-call + leads
prod-eu-west	Production EU	On-call + leads

Q: How do I check logs?

A: - **Datadog:** datadog.novatech.com (primary) - **kubectl:** `kubectl logs -f deployment/my-service` - **Structured search:** Use Datadog log explorer

Q: How do I access a database?

A: - Development: Connection strings in local `.env` - Staging: Via VPN + credentials in SecureVault - Production: JIT access through SecureVault, requires approval

---

On-Call

Q: How does on-call work?

A: Each team has a rotation: - 1-week shifts - Primary + secondary on-call - Scheduled in PagerDuty - Compensated (see On-Call Policy)

Q: What do I do when paged?

A: 1. Acknowledge alert in PagerDuty (within 5 min) 2. Join incident Slack channel (auto-created) 3. Assess severity and impact 4. Follow runbook for the alert 5. Escalate if needed 6. Document in post-mortem

Q: Where are the runbooks?

A: In Notion under Engineering → Runbooks, organized by service and alert type.

Q: Can I swap on-call shifts?

A: Yes! Use PagerDuty's override feature or arrange swap with teammate and notify your manager.

Getting Help

Q: Who do I ask for help?

A: - **Technical questions:** Your team Slack channel - **Cross-team questions:** #engineering-help - **Architecture:** #architecture - **Platform/infra:** #platform-help - **Security:** #security-questions

Q: Where's the documentation?

A: - **Company wiki:** Notion - **API docs:** Auto-generated in each repo's `/docs` - **Architecture:** Architecture Decision Records (ADRs) in repos - **Runbooks:** Notion → Engineering → Runbooks

Q: How do I propose a technical change?

A: 1. Write an RFC (Request for Comments) 2. Template in `novatech/rfcs` repo 3. Post to #engineering-rfcs for feedback 4. Present at architecture review (optional) 5. Get approval from relevant stakeholders

---

Tooling

Q: What monitoring tools do we use?

A:

Tool	Purpose
Datadog	Metrics, logs, APM
PagerDuty	Alerting, on-call
Sentry	Error tracking
Statuspage	Public status

Q: What project management tool do we use?

A: **Linear** for most teams (previously Jira for some). - Access at linear.app with NovaTech email - Your team's workspace auto-assigned

Q: How do I request a new tool?

A: 1. Check if it's already approved (IT-SW-001) 2. If not, submit software request via IT 3. Security review for tools that access data 4. 2-5 days for approval

---

Questions?

- Engineering questions: #engineering-help
- Process questions: Your manager or #eng-ops
- IT/tools: #it-help

Related Documents: Development Environment Setup (IT-SW-002), GitHub Access Policy (IT-SW-003), On-Call Policy (ENG-OPS-005)