

# Cloud Computing Technologies

DAT515 Fall 2025

DevOps: Best Practices and Tools

Prof Hein Meling





# What is DevOps?

An organizational culture that  
integrates development and  
operations for faster, more  
reliable software delivery

## DevOps is an Organizational Culture

- Break down silos between **development and operations teams** for streamlined software delivery
- **Automation** of workflows for faster, more reliable processes
- **Continuous integration and delivery** for rapid, frequent releases
- Improve software quality through **continuous monitoring** and **feedback**
- **Impact:**
  - Faster deployment of new features
  - Reduced time to fix problems

# The DevOps lifecycle

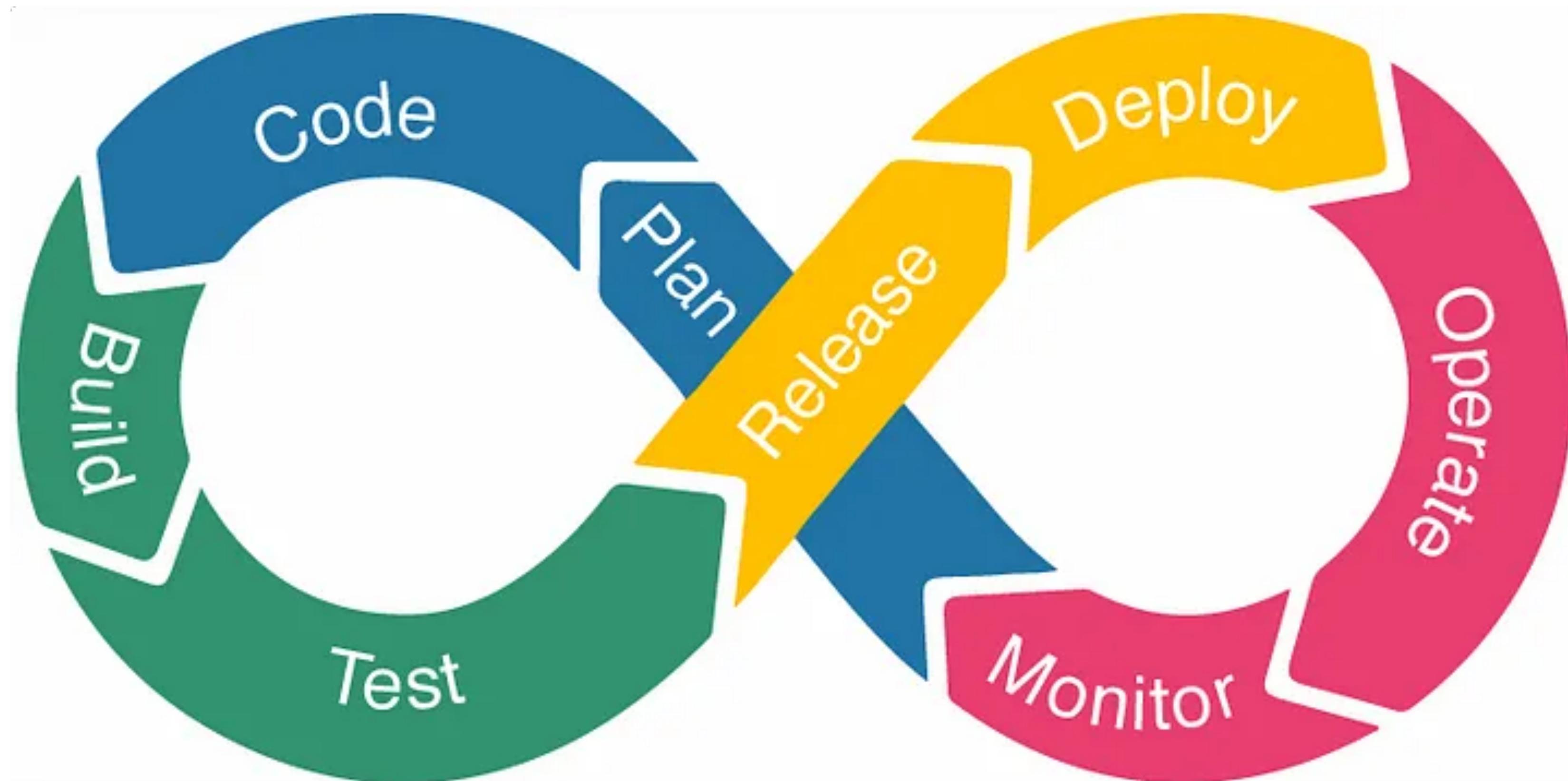


Image: <https://medium.com/t%C3%BCrk-telekom-bulut-teknolojileri/devops-lifecycle-continuous-integration-and-development-e7851a9c059d>

# The DevOps lifecycle

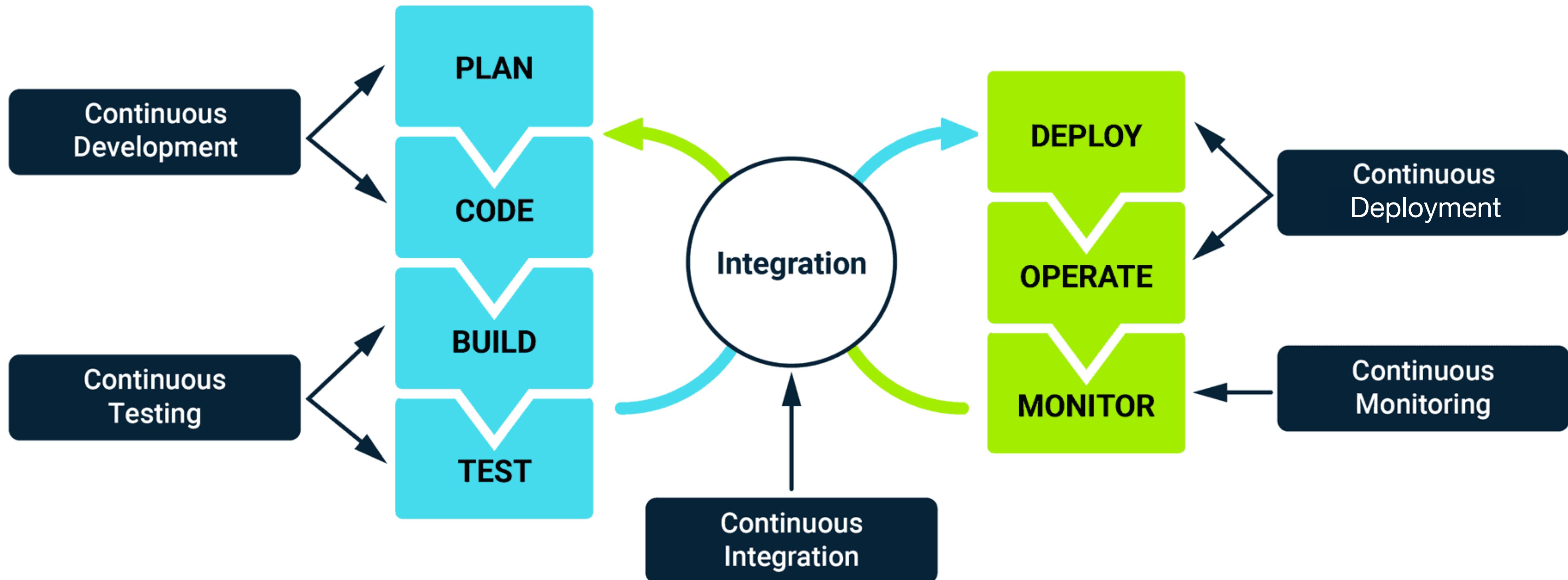


Image: <https://codilime.com/blog/devops-lifecycle/>



# Collaboration

## Communication

- Encourage collaboration between development, operations, and other stakeholders
- Foster a culture where all team members **share responsibility** for the entire process, from development to deployment
- Implement regular **feedback loops** to identify and address issues quickly

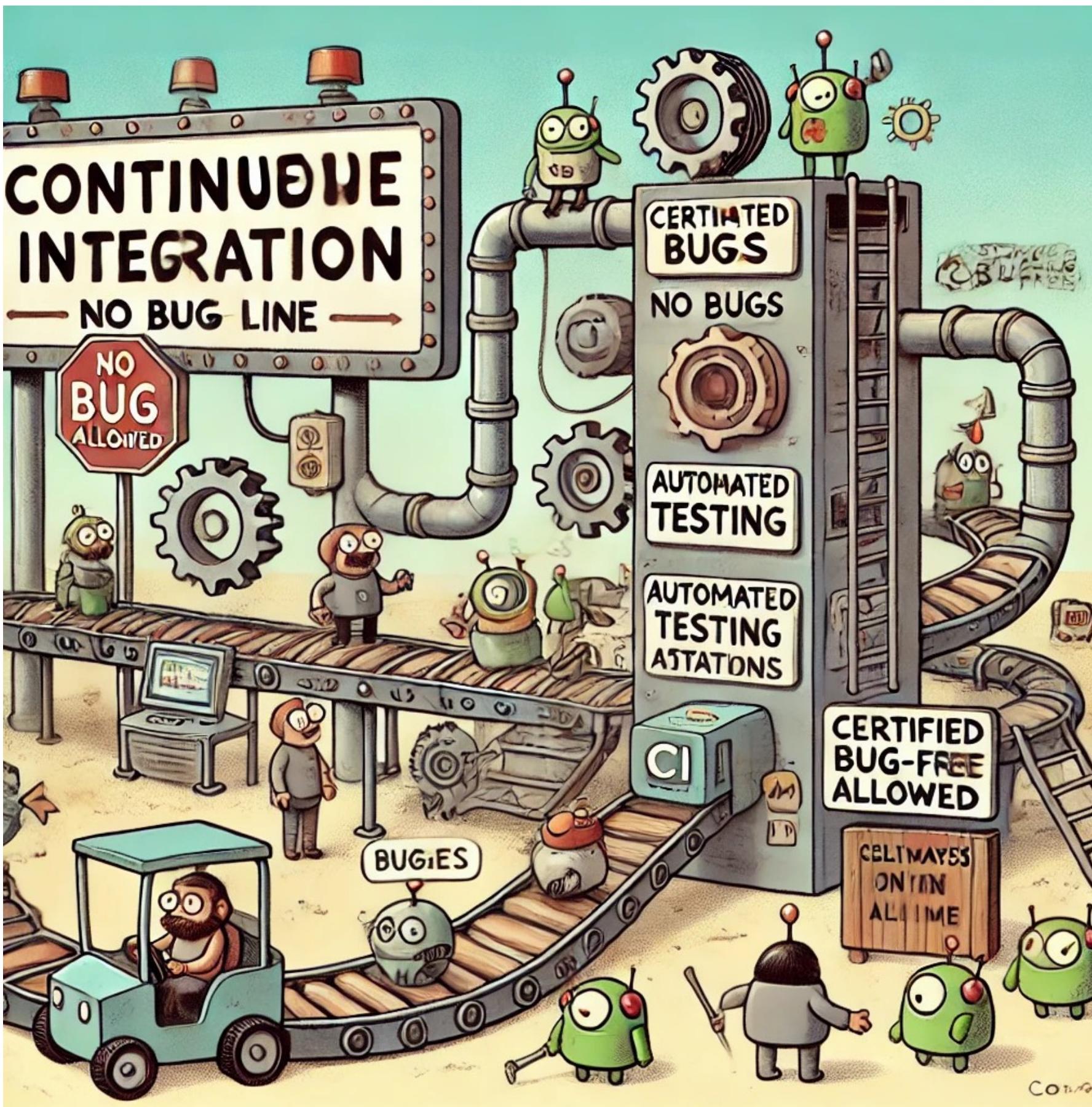
## Communication

- **Blameless Postmortems**
  - **Post-incident reviews** focused on **learning and improvement**
  - Not assigning blame
- Tools and platforms to enhance collaboration and streamline workflows
  - GitHub issue tracker, Jira, Slack, Discord

# Continuous Integration

## CI is the first step in continuous delivery pipelines

- Code changes **frequently merged** into repository
- Each merge triggers an **automated build** process
- **Automated testing** catches bugs early
- Developers get **quick feedback** on their code
- Frequent, smaller integrations **reduce conflicts**



# Continuous Delivery

## Automated Deployment Pipelines

- Always ready for deployment to **production**
- **Automated** Pipeline Stages:
  - build, test, and deploy stages
- Regular, **reliable releases** with minimal manual intervention
- **Rollback** to previous versions if issues arise
- Deploy to **multiple environments** automatically



# Infrastructure as Code

## Manage infrastructure using code, not manual processes

- Version Control: Track infrastructure changes with version control systems
- Consistency: Ensure consistent environments across development, testing, and production
- Automation: Automate provisioning, configuration, and management of infrastructure
- Scalability: Easily scale infrastructure up or down with code changes
- Tooling: Common tools include Terraform, Ansible, and CloudFormation

# Monitoring and Observability

# Monitoring and Observability

- Continuously **track system performance** and **health**
- Collect and analyze key **metrics, logs, and traces**
- Automated alerts for **early detection** of issues
- Continuous Improvement:
  - Use insights to **enhance system reliability** and **performance**



## Metrics

- Quantitative measures of the performance and behavior of a system over time
  - CPU usage, memory consumption, and request rates
  - Often used for monitoring and alerting

## Logs

- Detailed, time-stamped **records of events** and activities within a system
  - Capture what happened at a specific time, such as errors, transactions, or user actions
  - Important for debugging and troubleshooting

## Traces

- Detailed record of the **entire flow of a request** through a system
  - Help identify performance bottlenecks
  - Understand the path taken by a request across distributed systems
  - Understand how different services interact

## Metrics Collection (in ci package)

```
// TestExecutionMetricsCollectors returns a list of Prometheus metrics collectors for test execution.
func TestExecutionMetricsCollectors() []prometheus.Collector {
    return []prometheus.Collector{
        cloneTimeGauge,
        validationTimeGauge,
        testExecutionTimeGauge,
        testsStartedCounter,
        testsFailedCounter,
        testsSucceededCounter,
    }
}
```

## Metrics Collection (in metrics package)

```
var reg = prometheus.NewRegistry()

func init() {
    metricsCollectorsSets := [][]prometheus.Collector{
        interceptor.RPCMetricsCollectors(),
        ci.TestExecutionMetricsCollectors(),
    }
    for _, collectors := range metricsCollectorsSets {
        reg.MustRegister(collectors...)
    }
}

func Handler() http.Handler {
    return promhttp.HandlerFor(reg, promhttp.HandlerOpts{})
}
```

## Metrics Collection (in ci package)

```
cloneTimeGauge = prometheus.NewGaugeVec(prometheus.GaugeOpts{  
    Name: "quickfeed_clone_repositories_time",  
    Help: "The time to clone tests and student repository for test execution.",  
}, []string{"user", "course"})
```

```
validationTimeGauge = prometheus.NewGaugeVec(prometheus.GaugeOpts{  
    Name: "quickfeed_repository_validation_time",  
    Help: "The time to validate student repository for issues.",  
}, []string{"user", "course"})
```

```
testExecutionTimeGauge = prometheus.NewGaugeVec(prometheus.GaugeOpts{  
    Name: "quickfeed_test_execution_time",  
    Help: "The time to run test execution.",  
}, []string{"user", "course"})
```

## Metrics Collection (in ci package)

```
func timer(jobOwner, course string, gauge *prometheus.GaugeVec) func() {
    responseTimer := prometheus.NewTimer(prometheus.ObserverFunc(
        gauge.WithLabelValues(jobOwner, course).Set),
    )
    return func() { responseTimer.ObserveDuration() }
}
```

## Metrics Collection (in ci package)

```
defer timer(r.JobOwner, r.Course.Code, testExecutionTimeGauge)()
logger.Debugf("Running tests for %s", r)
start := time.Now()
out, err := runner.Run(ctx, job)
if err != nil && out == "" {
    testsFailedCounter.WithLabelValues(r.JobOwner, r.Course.Code).Inc()
    if errors.Is(err, ErrConflict) {
        return nil, err
    }
    return nil, fmt.Errorf("test execution failed without output: %w", err)
}
if err != nil {
    // We may reach here with a timeout error and a non-empty output
    testsFailedWithOutputCounter.WithLabelValues(r.JobOwner, r.Course.Code).Inc()
    logger.Errorf("Test execution failed with output: %v\n%v", err, out)
}
```

## Metrics Collection (in ci package)

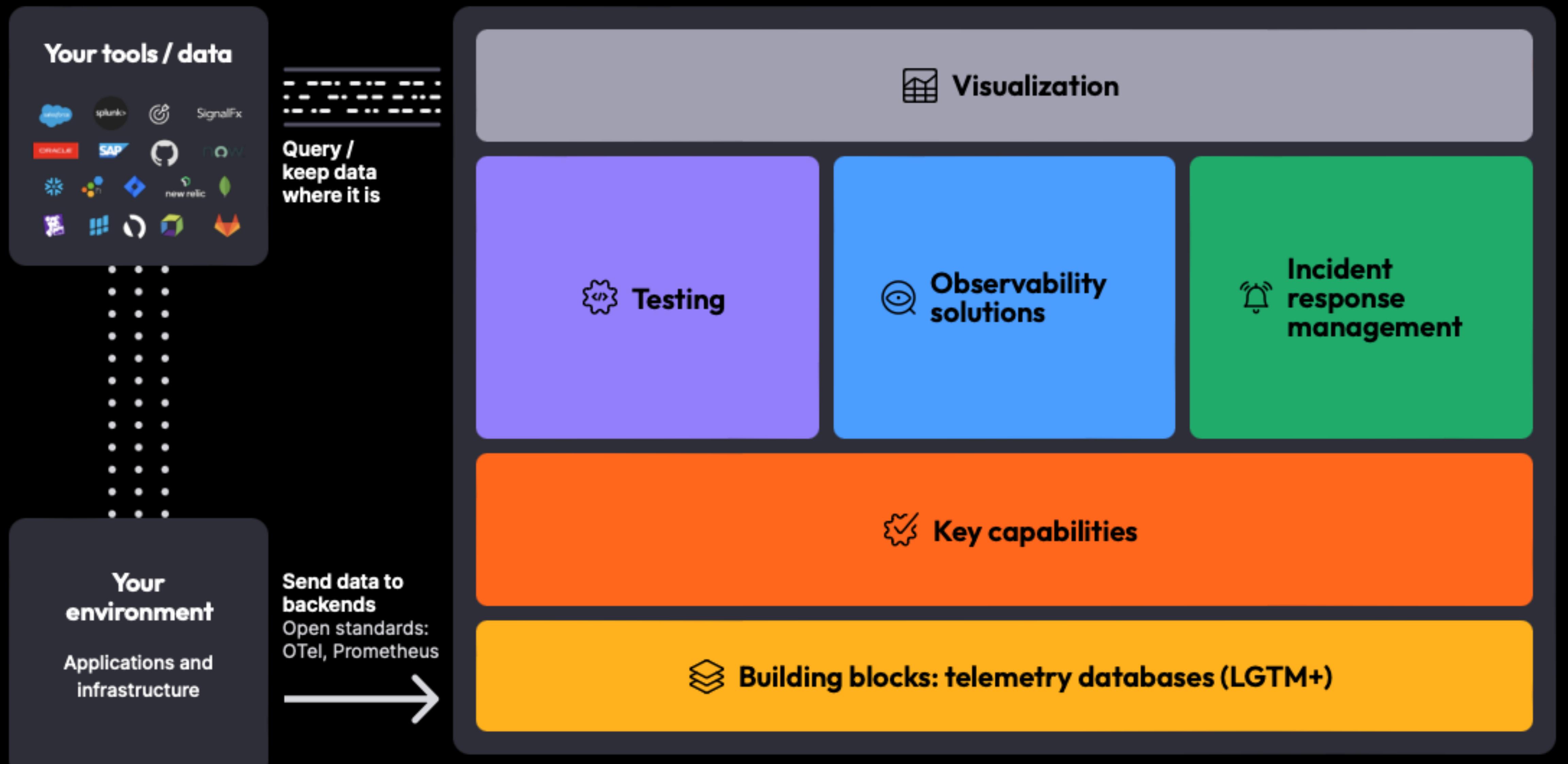
```
results, err := score.ExtractResults(out, randomSecret, time.Since(start))
if err != nil {
    // Log the errors from the extraction process
    testsFailedExtractResultsCounter.WithLabelValues(r.JobOwner, r.Course.Code).Inc()
    logger.Debugf("Session secret: %s", randomSecret)
    logger.Errorf("Failed to extract (some) results for assignment %s for course %s: %v",
        // don't return here; we still want partial results!
    }
}

testsSucceededCounter.WithLabelValues(r.JobOwner, r.Course.Code).Inc()
logger.Debug("ci.RunTests", zap.Any("Results", qlog.IndentJson(results)))
// return the extracted score and filtered log output
return results, nil
```

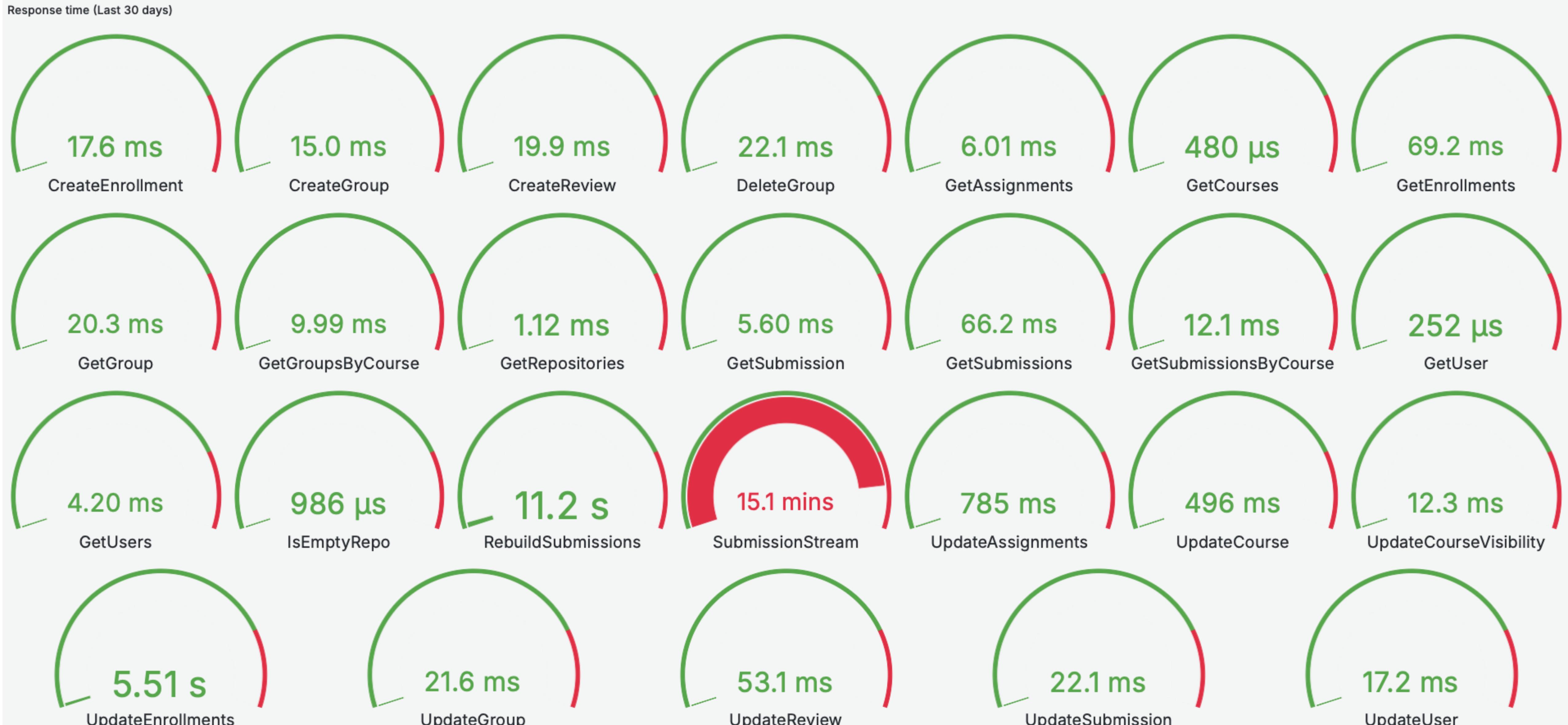
# Query the Prometheus Metrics Endpoint

```
% curl 127.0.0.1:9097
# HELP quickfeed_clone_repositories_time The time to clone tests and student repository for test execution.
# TYPE quickfeed_clone_repositories_time gauge
quickfeed_clone_repositories_time{course="DAT240",user="AnSuChi"} 0.85518224
quickfeed_clone_repositories_time{course="DAT240",user="BaalCat"} 0.889304762
quickfeed_clone_repositories_time{course="DAT240",user="alkval"} 0.851052653
quickfeed_clone_repositories_time{course="DAT240",user="lucyscript"} 0.893811969
quickfeed_clone_repositories_time{course="DAT320",user="IcePingus"} 0.862420794
quickfeed_clone_repositories_time{course="DAT320",user="JosteinLindhom"} 1.012562895
quickfeed_clone_repositories_time{course="DAT320",user="LBerntsen"} 0.729477786
quickfeed_clone_repositories_time{course="DAT320",user="abdu2706"} 0.531776677
quickfeed_clone_repositories_time{course="DAT320",user="aleksanderboe"} 0.934033701
quickfeed_clone_repositories_time{course="DAT320",user="xSARB"} 0.529486225
quickfeed_clone_repositories_time{course="DAT515",user="ForoozanE"} 0.53936162
quickfeed_clone_repositories_time{course="DAT515",user="Sihe12"} 2.061271005
```

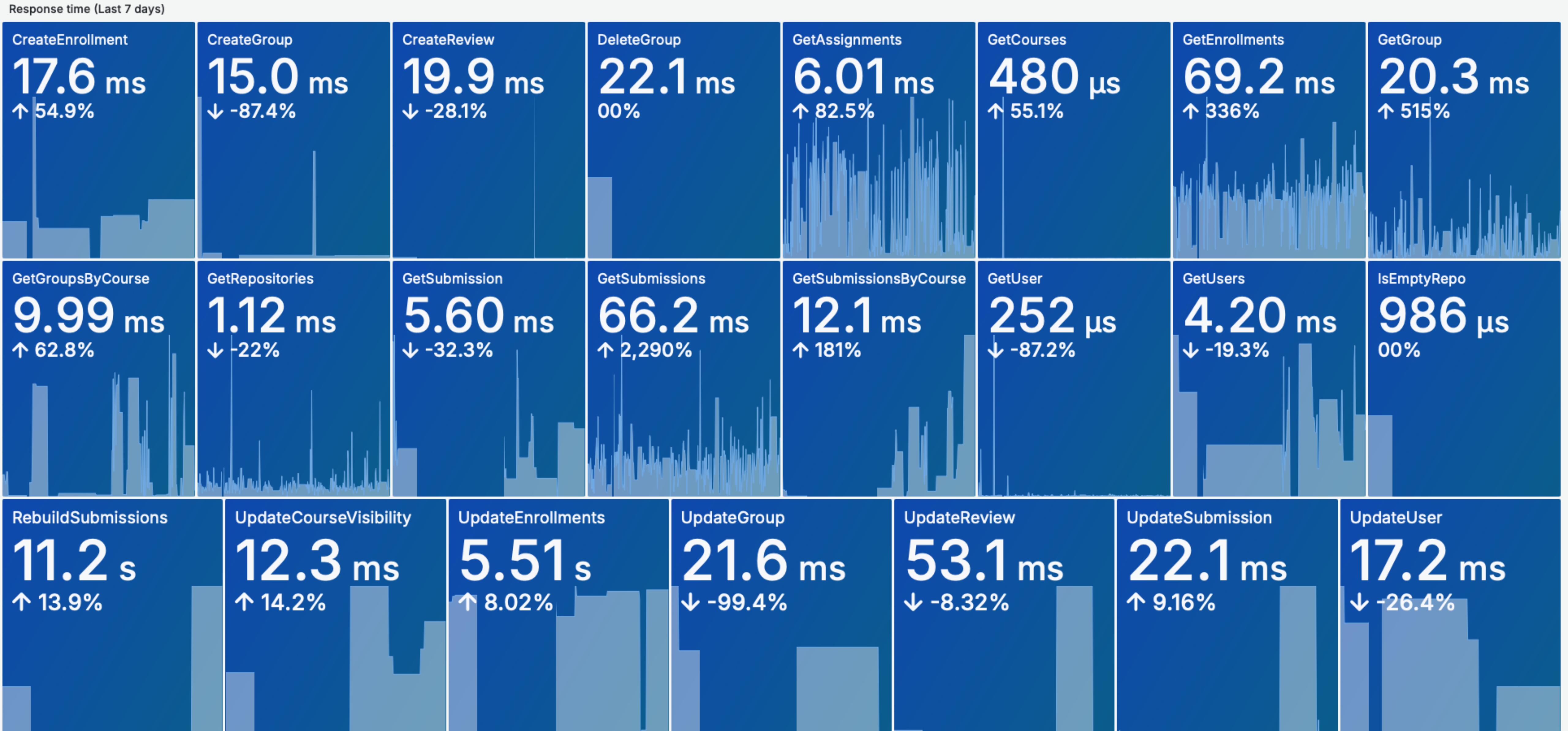
# Grafana - Metrics Dashboard



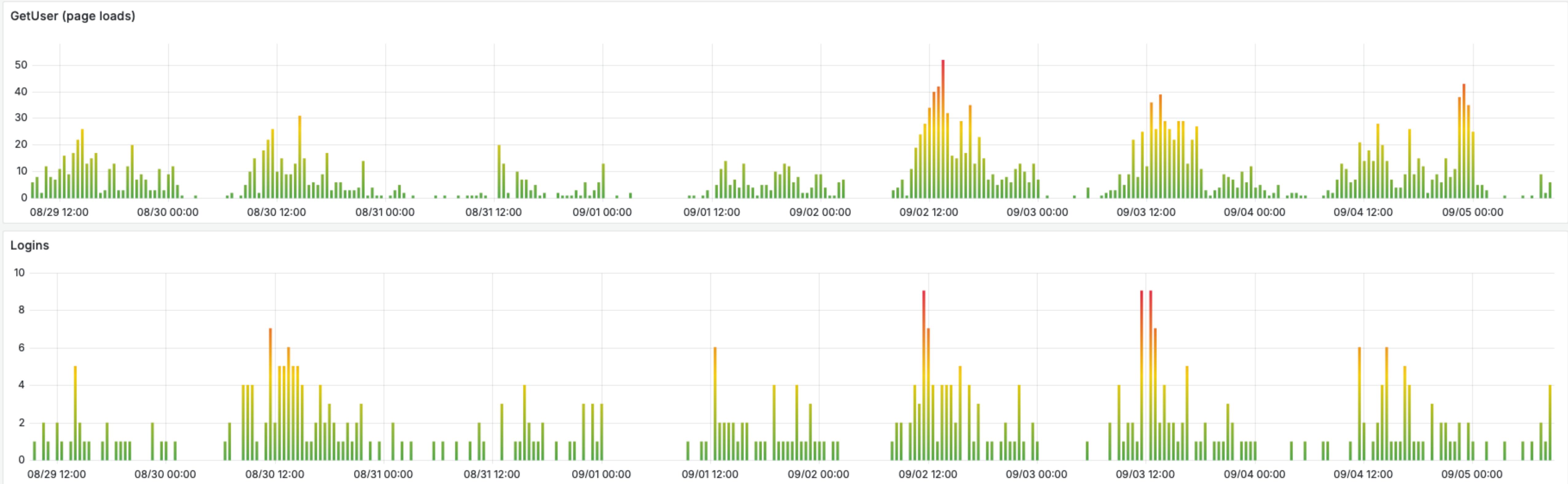
# Grafana - Metrics Dashboard



# Grafana - Metrics Dashboard

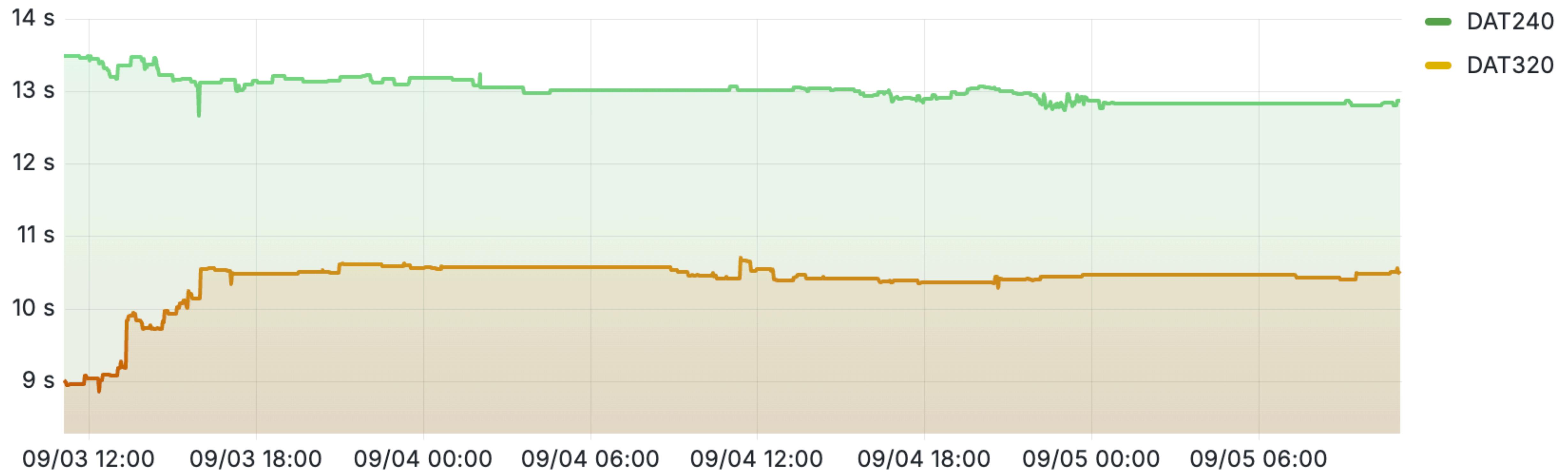


# Grafana - Metrics Dashboard



# Grafana - Metrics Dashboard

Test execution time (Last 2 days)



# Grafana - Metrics Dashboard

Method Accessed Count (Last 90 days)						
CreateEnrollment	CreateGroup	CreateReview	DeleteGroup	GetAssignments	GetCourses	GetEnrollments
<b>131</b>	<b>20</b>	<b>47</b>	<b>0</b>	<b>14.9 k</b>	<b>7.27 k</b>	<b>22.3 k</b>
GetGroup	GetGroupsByCourse	GetRepositories	GetSubmission	GetSubmissions	GetSubmissionsByCourse	GetUser
<b>7.62 k</b>	<b>1.61 k</b>	<b>14.6 k</b>	<b>230</b>	<b>22.4 k</b>	<b>371</b>	<b>8.57 k</b>
GetUsers	IsEmptyRepo	RebuildSubmissions	SubmissionStream	UpdateAssignments	UpdateCourse	UpdateCourseVisibility
<b>185</b>	<b>28</b>	<b>26</b>	<b>14.6 k</b>	<b>0</b>	<b>0</b>	<b>74</b>
UpdateEnrollments	UpdateGroup	UpdateReview	UpdateSubmission	UpdateUser		
<b>250</b>	<b>20</b>	<b>300</b>	<b>134</b>	<b>38</b>		

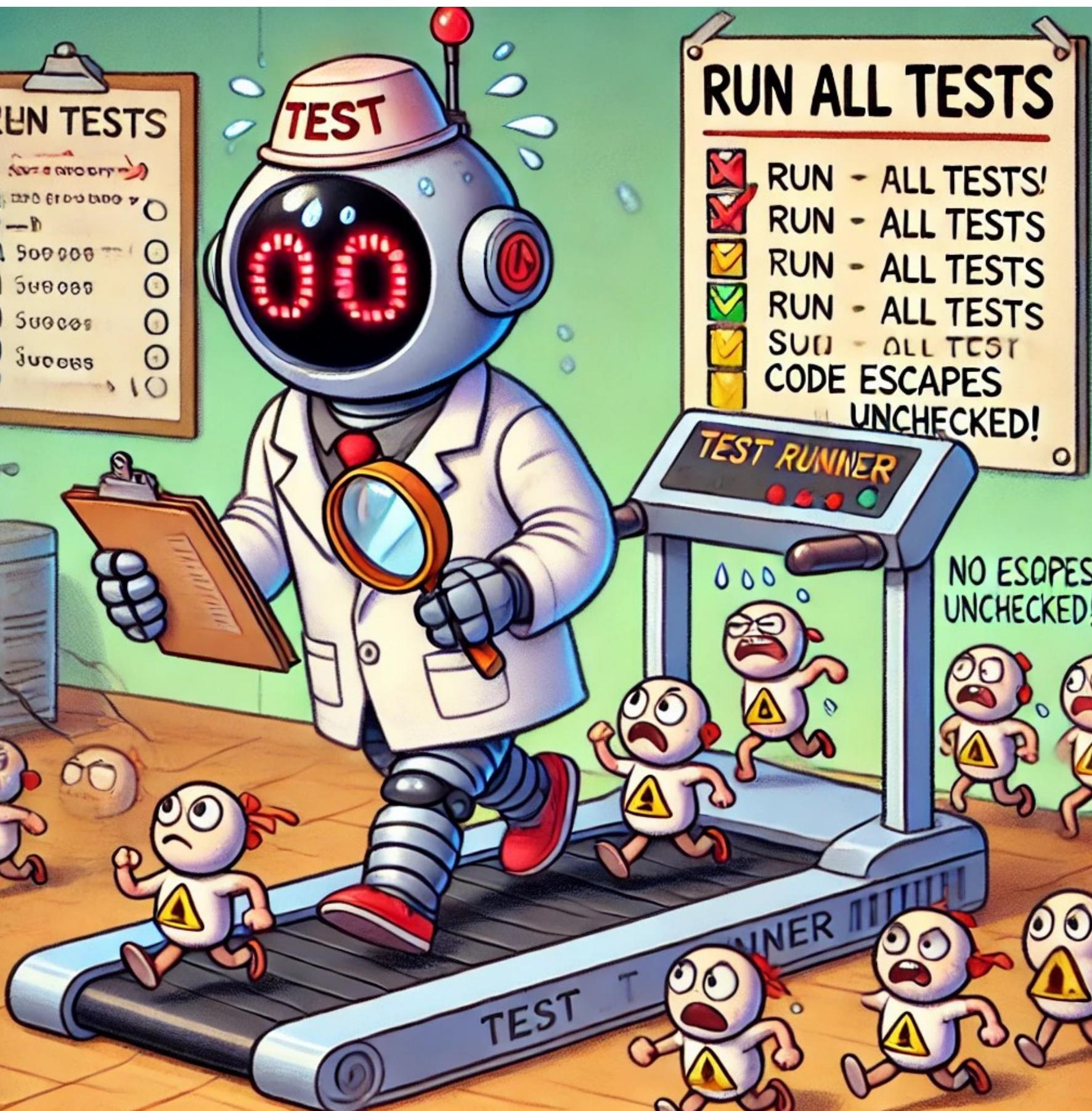
Joke time

# What kind of cloud automation without the DevOps engineer?

# Automated Testing

## Continuous Testing at every stage of the development pipeline

- **Reduce risk** of human errors by automating repetitive test tasks
- Run tests automatically to **accelerate feedback**
- Quickly **catch regressions** with automated tests
- Several types of testing
  - Unit, integration, property-based, fuzz
  - Linters
  - Test coverage



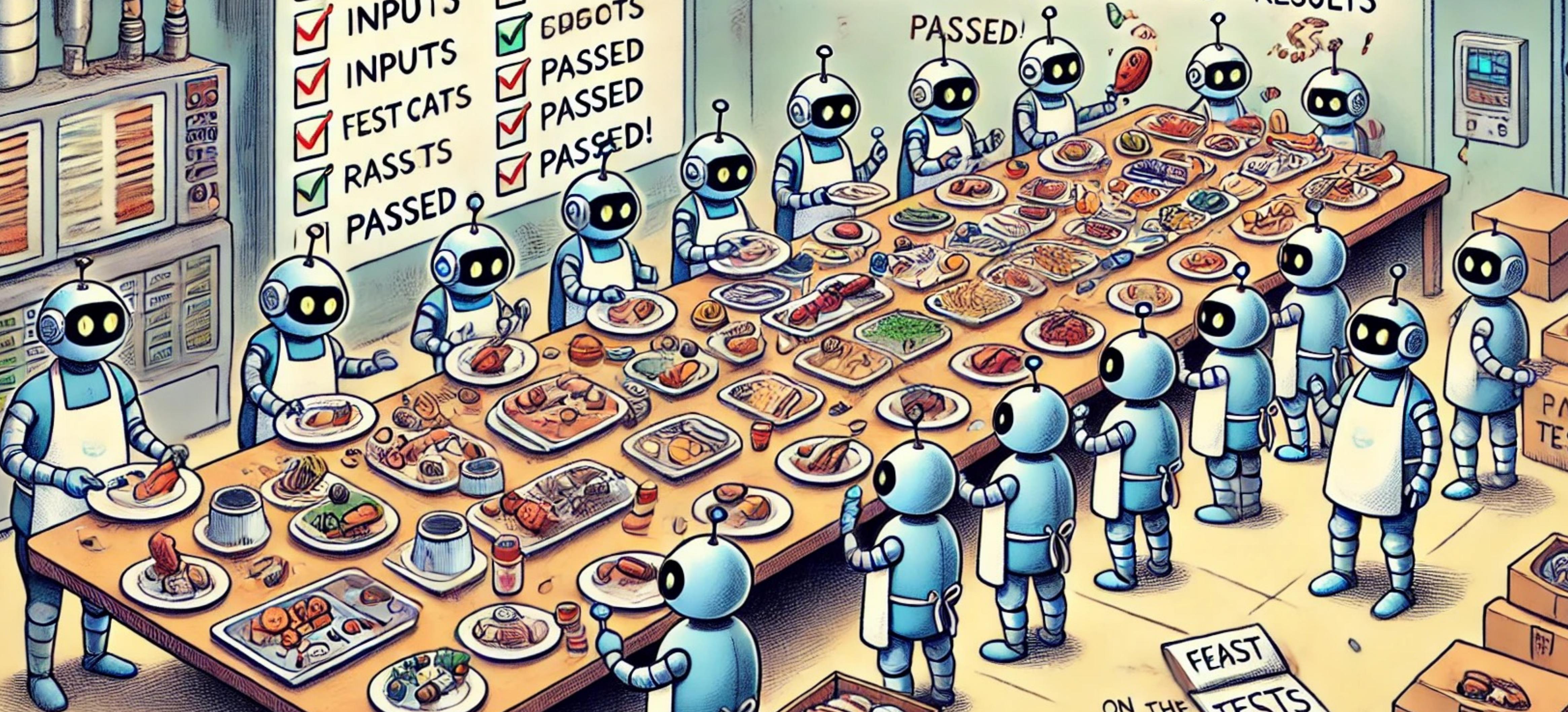
# FABST-DN THE TESTS

## TABLE-DRIVEN TESTS

- |  |  |
|--|--|
| <input checked="" type="checkbox"/> TEST CASES | <input checked="" type="checkbox"/> TEST CASES |
| <input checked="" type="checkbox"/> INPUTS     | <input checked="" type="checkbox"/> INPUTS     |
| <input checked="" type="checkbox"/> INPUTS     | <input checked="" type="checkbox"/> INPUTS     |
| <input checked="" type="checkbox"/> FESTCATS   | <input checked="" type="checkbox"/> FESTCATS   |
| <input checked="" type="checkbox"/> RASSTS     | <input checked="" type="checkbox"/> RASSTS     |
| PASSED   | PASSED!  |

EXPECTED OUTPUTS    EXPECTED OUTPUTS    EXPECTED OUTPUTS    EXPECTED RESULTS

PASSED!



quickfeed > scm > `-go` github\_mock\_test.go > ...

run test | debug test

```
▶ 189 func TestMockGetRepositories(t *testing.T) {
190     tests := []struct {
191         name    string
192         org     string
193         want    []*Repository
194         wantErr bool
195     }{
196         {name: "IncompleteRequest/NilOrg", org: "", want: nil, wantErr: true},
197         {name: "IncompleteRequest/NoOrgName", org: "", want: nil, wantErr: true},
198         {name: "CompleteRequest/NotFound", org: "buz", want: []*Repository{}, wantErr: false},
199         {name: "CompleteRequest/SixRepos", org: "foo", want: []*Repository{
200             {ID: 1, Repo: "info"},
201             {ID: 2, Repo: "assignments"},
202             {ID: 3, Repo: "tests"},
203             {ID: 4, Repo: "meling-labs"},
204             {ID: 5, Repo: "josie-labs"},
205             {ID: 6, Repo: "groupX"}},
206     },
207 }
```

```
208     s := NewMockedGithubSCMClient(qtest.Logger(t), WithOrgs(ghOrgFoo, ghOrgBar, ghOrgBuz), WithRepos(repos...))
209     for _, tt := range tests {
210         t.Run(tt.name, func(t *testing.T) {
211             got, err := s.GetRepositories(context.Background(), tt.org)
212             if (err != nil) != tt.wantErr {
213                 t.Errorf("GetRepositories() error = %v, wantErr %v", err, tt.wantErr)
214                 return
215             }
216             if diff := cmp.Diff(tt.want, got); diff != "" {
217                 t.Errorf("GetRepositories() mismatch (-want +got):\n%s", diff)
218             }
219         })
220     }
221 }
```

## Name function being tested with inputs and got, wanted output

```
s := NewMockedGithubSCMClient(qtest.Logger(t), WithOrgs(ghOrgFoo, ghOrgBar), WithRepos(repos...))
for _, tt := range tests {
    name := qtest.Name(tt.name, []string{"Owner", "Path"}, tt.opt.Owner, tt.opt.Repo)
    t.Run(name, func(t *testing.T) {
        if empty := s.RepositoryIsEmpty(context.Background(), tt.opt); empty != tt.wantEmpty {
            t.Errorf("RepositoryIsEmpty(%+v) = %t, want %t", *tt.opt, empty, tt.wantEmpty)
        }
    })
}
```

```
go test -v -run TestMockRepositoryIsEmpty

    === RUN   TestMockRepositoryIsEmpty
    === RUN   TestMockRepositoryIsEmpty/IncompleteRequest
    === RUN   TestMockRepositoryIsEmpty/IncompleteRequest/Owner=foo
    === RUN   TestMockRepositoryIsEmpty/IncompleteRequest/Path=info
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=info
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=assignments
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=tests
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=meling-labs
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=info
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=assignments
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=tests
    === RUN   TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=meling-labs
--- PASS: TestMockRepositoryIsEmpty (0.01s)
    --- PASS: TestMockRepositoryIsEmpty/IncompleteRequest (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/IncompleteRequest/Owner=foo (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/IncompleteRequest/Path=info (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=info (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=assignments (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=tests (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=meling-labs (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=info (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=assignments (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=tests (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=meling-labs (0.00s)

PASS
ok      github.com/quickfeed/quickfeed/scm      0.359s
```

# Testing that tests fail on bad input

```
go test
--- FAIL: TestMockRepositoryIsEmpty (0.00s)
    --- FAIL: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=baz/Path=tests (0.00s)
        github_mock_test.go:248: RepositoryIsEmpty({ID:0 Repo:tests Owner:baz}) = true, want false
FAIL
exit status 1
FAIL    github.com/quickfeed/quickfeed/scm      8.606s
```

# Testing that tests fail on bad input

```
go test -v -run TestMockRepositoryIsEmpty

==== RUN TestMockRepositoryIsEmpty
==== RUN TestMockRepositoryIsEmpty/IncompleteRequest
==== RUN TestMockRepositoryIsEmpty/IncompleteRequest/Owner=foo
==== RUN TestMockRepositoryIsEmpty/IncompleteRequest/Path=info
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=info
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=assignments
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=tests
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=meling-labs
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=info
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=assignments
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=baz/Path=tests
    github_mock_test.go:248: RepositoryIsEmpty({ID:0 Repo:tests Owner:baz}) = true, want false
==== RUN TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=meling-labs
--- FAIL: TestMockRepositoryIsEmpty (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/IncompleteRequest (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/IncompleteRequest/Owner=foo (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/IncompleteRequest/Path=info (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=info (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=assignments (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=tests (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/Empty/Owner=bar/Path=meling-labs (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=info (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=assignments (0.00s)
    --- FAIL: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=baz/Path=tests (0.00s)
    --- PASS: TestMockRepositoryIsEmpty/CompleteRequest/NonEmpty/Owner=foo/Path=meling-labs (0.00s)

FAIL
exit status 1
FAIL    github.com/quickfeed/quickfeed/scm          0.356s
```

# feat: Add context logging support #1046

 Draft meling wants to merge 1 commit into [master](#) from [logging-with-context](#) 

 Conversation 0  Commits 1  Checks 8  Files changed 3



meling commented on Feb 7 · edited

Member

...

This adds a simple interceptor-based context logging feature that can be used in RPC calls to fetch the context logger to log with session context such as source method, userID, courseID etc. We can add more global context if necessary, but most often we probably want to add logging information via the actual log msg.

This version just uses the current zap logger to avoid introducing another logging framework now. We can decide later whether or not to replace zap with slog.

Fixes [#670](#)

Fixes [#489](#)



  feat(interceptor): added context logging support via interceptor ...  ef5d465

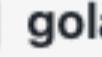


 Some checks were not successful

[Hide all checks](#)

  Go Test / test (1.21, ubuntu-latest) (pull\_request) Failing after 2m [Details](#)

  Test Coverage / test (1.21) (pull\_request) Failing after 2m [Details](#)

  golangci-lint / lint (pull\_request) Failing after 1m [Details](#)

  CodeQL / Analyze (go) (pull\_request) Successful in 2m [Details](#)

# QuickFeed Pull Request

master quickfeed/.github/workflows/go.yml View Runs

meling chore: updated dependencies and github workflows to go 1.22.5

41 lines (35 loc) · 916 Bytes

**Code** Blame

```
1 name: Go Test
2
3 on:
4   push:
5     branches: [master]
6     paths:
7       - '**.go'
8   pull_request:
9     paths:
10    - '**.go'
11
12 jobs:
13   test:
14     strategy:
15       matrix:
16         go-version: ["1.22.5"]
17         platform: [ubuntu-latest, macos-latest]
18       runs-on: ${{ matrix.platform }}
```

- GitHub Actions Workflow

# Testing QuickFeed

quickfeed > .github > workflows >  go.yml

```
1  name: Go Test
2
3  on:
4    push:
5      branches: [master]
6      paths:
7        - '**.go'
8    pull_request:
9      paths:
10     - '**.go'
11
12 jobs:
13   test:
14     strategy:
15       matrix:
16         go-version: ["1.22.5"]
17         platform: [ubuntu-latest, macos-latest]
18       runs-on: ${{ matrix.platform }}
```

```
20   steps:
21     - name: Cache
22       uses: actions/cache@v4
23       with:
24         path: |
25           ~/.cache/go-build      # ubuntu-latest
26           ~/Library/Caches/go-build # macos-latest
27           ~/go/pkg/mod
28         key: ${{ runner.os }}-go-${{ hashFiles('**/go.sum') }}
29         restore-keys: |
30           ${{ runner.os }}-go-
31
32     - name: Checkout code
33       uses: actions/checkout@v4
34
35     - name: Set up Go
36       uses: actions/setup-go@v5
37       with:
38         go-version: ${{ matrix.go-version }}
39
40     - name: Run Go tests
41       run: DOCKER_TESTS=1 go test -v github.com/quickfeed/quickfeed/...
```

**Actions**[New workflow](#)[All workflows](#)[CodeQL](#)[Dependabot Updates](#)**Go Test**[golangci-lint](#)[Jest Test](#)[Test Coverage](#)[Management](#) [Caches](#) [Attestations](#) [Runners](#)**Go Test**[go.yml](#)**321 workflow runs** Filter workflow runs

...

[Event](#) ▾ [Status](#) ▾ [Branch](#) ▾ [Actor](#) ▾**Merge pull request #1124 from quickfee...**[master](#)

last month ...

1m 48s

Go Test #2224: Commit [fd2e4a7](#) pushed by meling**Ignore push on group repo's non-default...**[bugfix/issue1053](#)

last month ...

1m 52s

Go Test #2223: Pull request [#1124](#) opened by meling**Merge pull request #1122 from quickfee...**[master](#)

last month ...

1m 58s

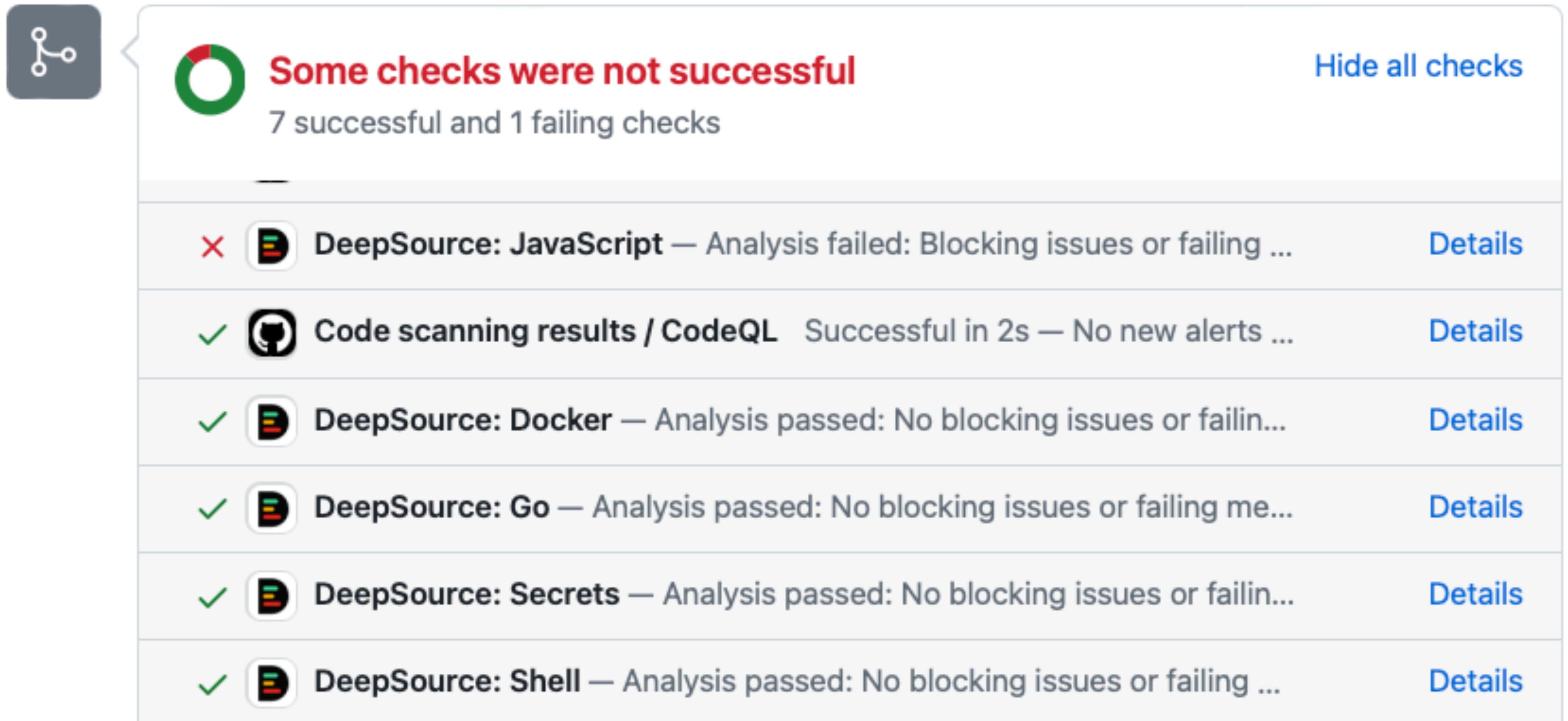
Go Test #2222: Commit [1031154](#) pushed by meling**fix(ci): handle image conflict during bu...**[fix-docker-conflict-error](#)

last month ...

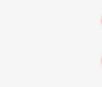
2m 5s

Go Test #2221: Pull request [#1122](#) opened by

JosteinLindhom

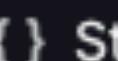


The screenshot shows a list of code analysis checks from DeepSource. At the top, a green circle icon indicates "Some checks were not successful" with a count of "7 successful and 1 failing checks". A "Hide all checks" button is located in the top right corner. Below this, there is a table with six rows, each representing a different analysis type:

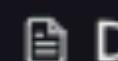
Check Type	Description	Status
DeepSource: JavaScript	Analysis failed: Blocking issues or failing ...	 Details
Code scanning results / CodeQL	Successful in 2s — No new alerts ...	 Details
DeepSource: Docker	Analysis passed: No blocking issues or failin...	 Details
DeepSource: Go	Analysis passed: No blocking issues or failing me...	 Details
DeepSource: Secrets	Analysis passed: No blocking issues or failin...	 Details
DeepSource: Shell	Analysis passed: No blocking issues or failing ...	 Details

 Performance issues

121

 Style issues

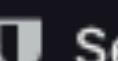
3

 Documentation issues

0

 Anti patterns

158

 Security issues

0

 Coverage issues

0

## ACTIVE ISSUES

305

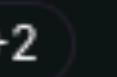
## ISSUES PREVENTED

4.6k

## DEFAULT BRANCH

 master

## ANALYZERS

    +2✓ Last analyzed  6d374c6

3 days ago

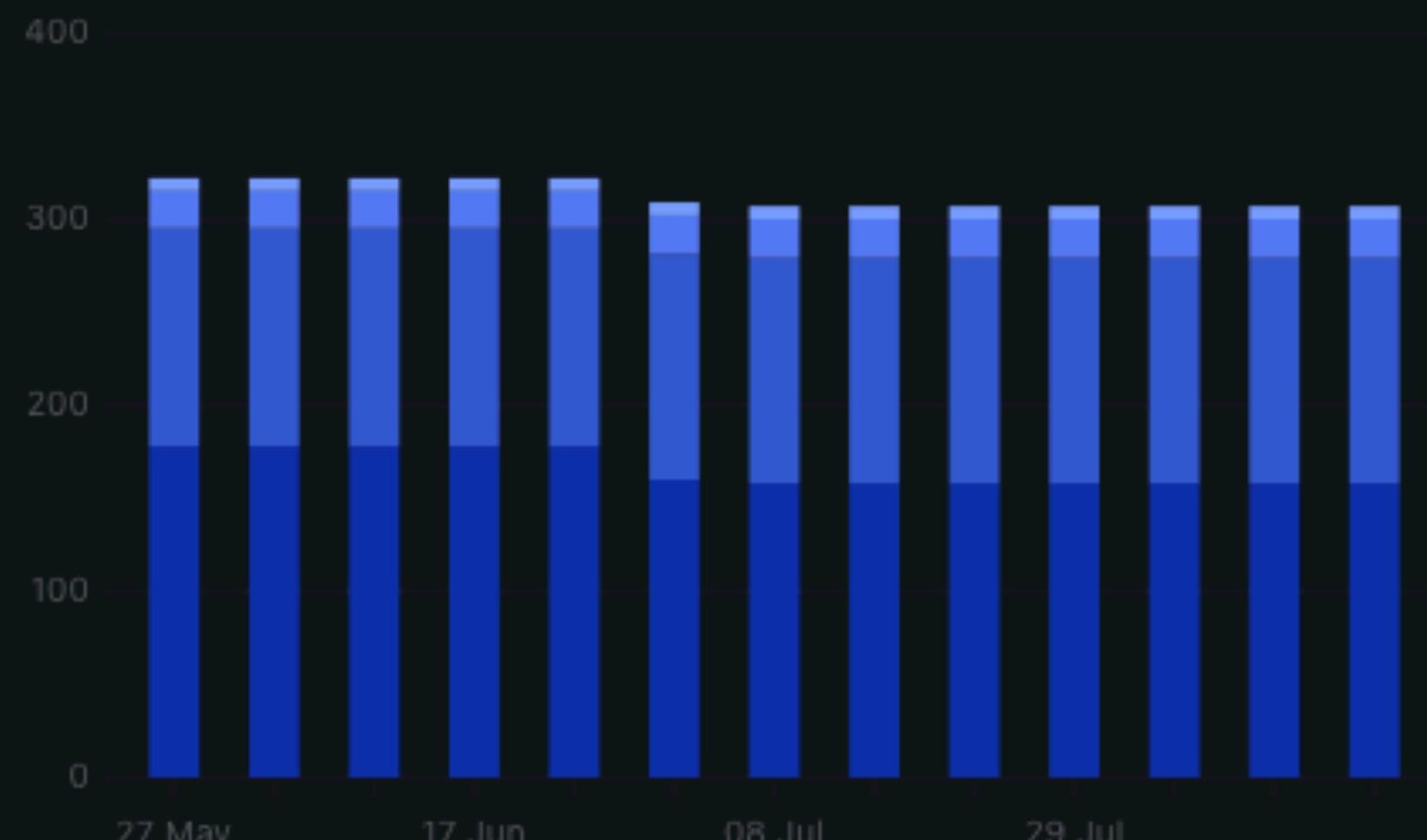
OWASP® Top 10  PASSING

0 active issues

...

Issue Distribution by Category 

305 active issues



5

4

3

2

1

0

27 May

17 Jun

08 Jul

29 Jul

[Overview](#)[Issues](#)[Metrics](#)[Reports](#)[History](#)[Settings](#)[All](#)[∞](#)

Filter by Autofix +

Severity +

Sort

Search for an issue...

All issues

305

Recommended

114

Secrets

2

Bug Risk

21

Anti-pattern

158

Security

0

Performance

121

Typecheck

0

Coverage

0

Style

3

Documentation

0

**Avoid .bind() or local functions in JSX properties** JS-0417

↗ Performance ⚠ Major ⏰ 2 months ago — 3 years old

Seen in 44 files

**103****Non-interactive elements assigned mouse/keyboard event listeners**

JS-0760

↗ Performance ⚠ Minor ⏰ 9 months ago — 3 years old

Seen in 4 files

**4****Prefer not to use words image, photo in image alt content** JS-0750

↗ Performance ⚠ Minor ⏰ 9 months ago — 3 years old

Seen in 1 file

**5****Non-interactive elements should not be assigned interactive roles**

JS-0761

↗ Performance ⚠ Minor ⏰ 9 months ago — 3 years old

Seen in 1 file

**1**

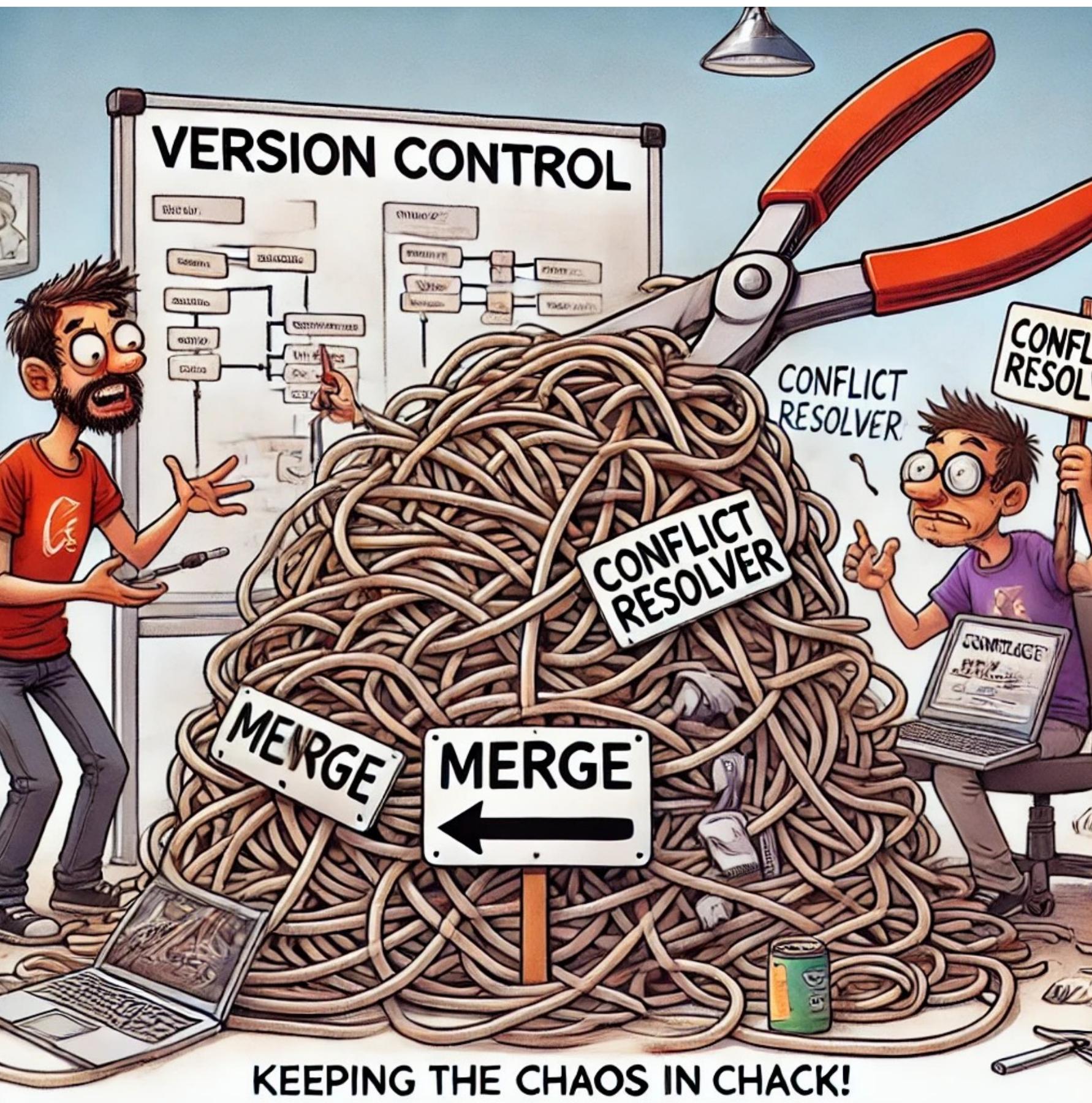
Joke time

# How do I get past engineers stuck in my way.

# Version Control

## Source Code Management

- Use branches for **feature** development, bug **fixes**, and **releases**
- Collaboration with shared repositories and **pull requests**
- Maintain a detailed **history of changes**
  - Who made them and why
  - Conventional commits
- **Revert to previous versions** if issues arise



## Advanced Features

- git bisect
- git blame
- git rebase vs git merge
- git squash



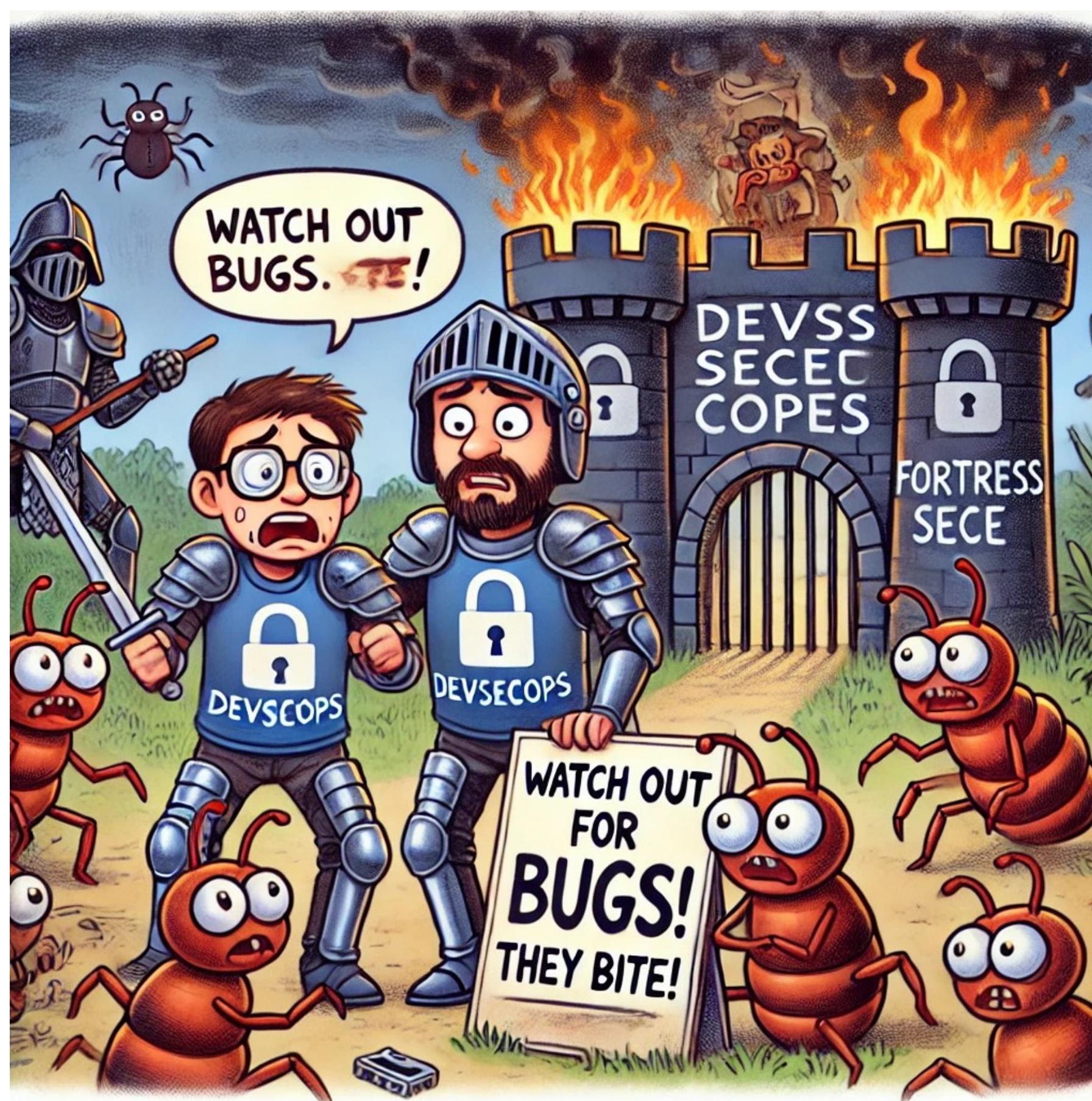
## Jujutsu – headline features

- Works seamlessly with Git repositories and workflows
- Branches themselves are tracked and versioned, not just commits
- Safe, local-first design – experimentation without risk of losing work
- Powerful undo and redo – Easily backtrack or reapply changes
- Flexible and intuitive – Command set designed to be simpler and more consistent than Git

# DevSecOps

## DevSecOps: Embed security practices into every phase

- Shift Left: Move security checks earlier to **catch vulnerabilities sooner**
- Tools to automate **security and vulnerability scans**
  - Govulncheck, Dependabot, Snyk, Aqua Security
- Continuously **monitor** applications and infrastructure for **security threats**
- Define **security policies** and controls as code, ensuring **consistent enforcement**





quickfeed / quickfeed



Code

Issues 76

Pull requests 5

Discussions

Actions

Projects 1

Security

...

Overview

## Security overview

Reporting

Policy

Advisories

Vulnerability alerts

Dependabot

Code scanning

Secret scanning

### Security policy • Disabled

Define how users should report security vulnerabilities for this repository

[Set up a security policy](#)

### Security advisories • Enabled

View or disclose security advisories for this repository

[View security advisories](#)

### Private vulnerability reporting • Enabled

Allow users to privately report potential security vulnerabilities

[See reported vulnerabilities](#)

### Dependabot alerts • Enabled

Get notified when one of your dependencies has a vulnerability

[View Dependabot alerts](#)

### Code scanning alerts • Enabled

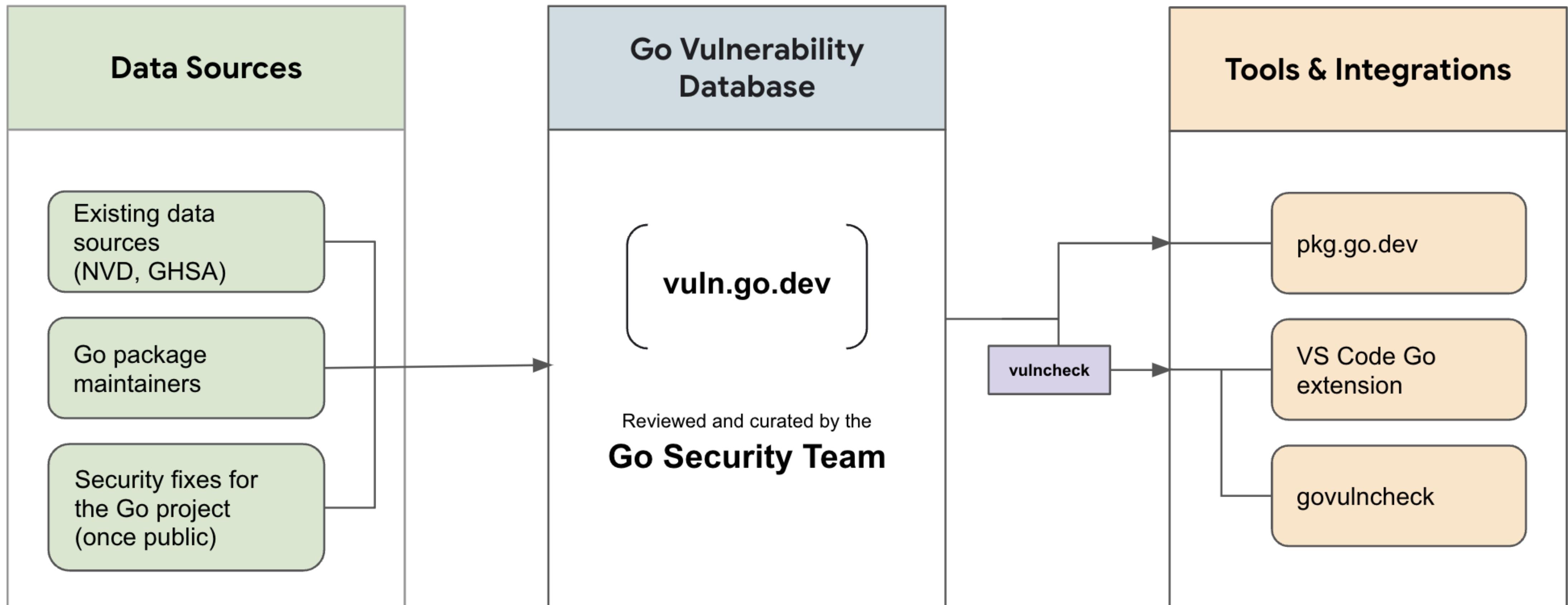
Automatically detect common vulnerability and coding errors

[View alerts](#)

### Secret scanning alerts • Disabled

Get notified when a secret is pushed to this repository

[Enable in settings](#)



- cd work/quickfeed
- go install [golang.org/x/vuln/cmd/govulncheck@latest](https://golang.org/x/vuln/cmd/govulncheck@latest)
- govulncheck ./...
  - Wait a minute or so...
  - No vulnerabilities found.

```
% govulncheck ./...
== Symbol Results ==
```

## Vulnerability #1: GO-2024-2687

HTTP/2 CONTINUATION flood in net/http

More info: <https://pkg.go.dev/vuln/GO-2024-2687>

Module: golang.org/x/net

Found in: golang.org/x/net@v0.22.0

Fixed in: golang.org/x/net@v0.23.0

Example traces found:

#1: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.ConnectionError.Error

#2: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.ErrCode.String

#3: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.FrameHeader.String

#4: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.FrameType.String

#5: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.ReadFrame

#6: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteContinuation

#7: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteData

[snip]

#21: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.duplicatePseudoHeaderError.Error

#22: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.headerFieldNameError.Error

#23: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.headerFieldValueError.Error

#24: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.pseudoHeaderError.Error

#25: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.writeData.String

Your code is affected by 1 vulnerability from 1 module.

This scan found no other vulnerabilities in packages you import or modules you require.

Use '-show verbose' for more details.

```
~/work/gorums git:(master)±2 (13.372s)
govulncheck ./...
==== Symbol Results ===

Vulnerability #1: GO-2024-2687
    HTTP/2 CONTINUATION flood in net/http
    More info: https://pkg.go.dev/vuln/GO-2024-2687
    Module: golang.org/x/net
        Found in: golang.org/x/net@v0.22.0
        Fixed in: golang.org/x/net@v0.23.0
    Example traces found:
        #1: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.ConnectionError.Error
        #2: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.ErrCode.String
        #3: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.FrameHeader.String
        #4: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.FrameType.String
        #5: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.ReadFrame
        #6: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteContinuation
        #7: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteData
        #8: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteGoAway
        #9: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteHeader
        #10: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WritePing
        #11: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteRSTStream
        #12: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteSettings
        #13: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WriteSettingsAck
        #14: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.Framer.WindowUpdate
        #15: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.Settings.String
        #16: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.SettingsID.String
        #17: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.SettingsFrame.ForeachSetting
```

# Vulnerability Report: GO-2024-2687

standard library

Affects: [net/http](#), [golang.org/x/net](#) | Published: Apr 03, 2024 | Modified: May 20, 2024

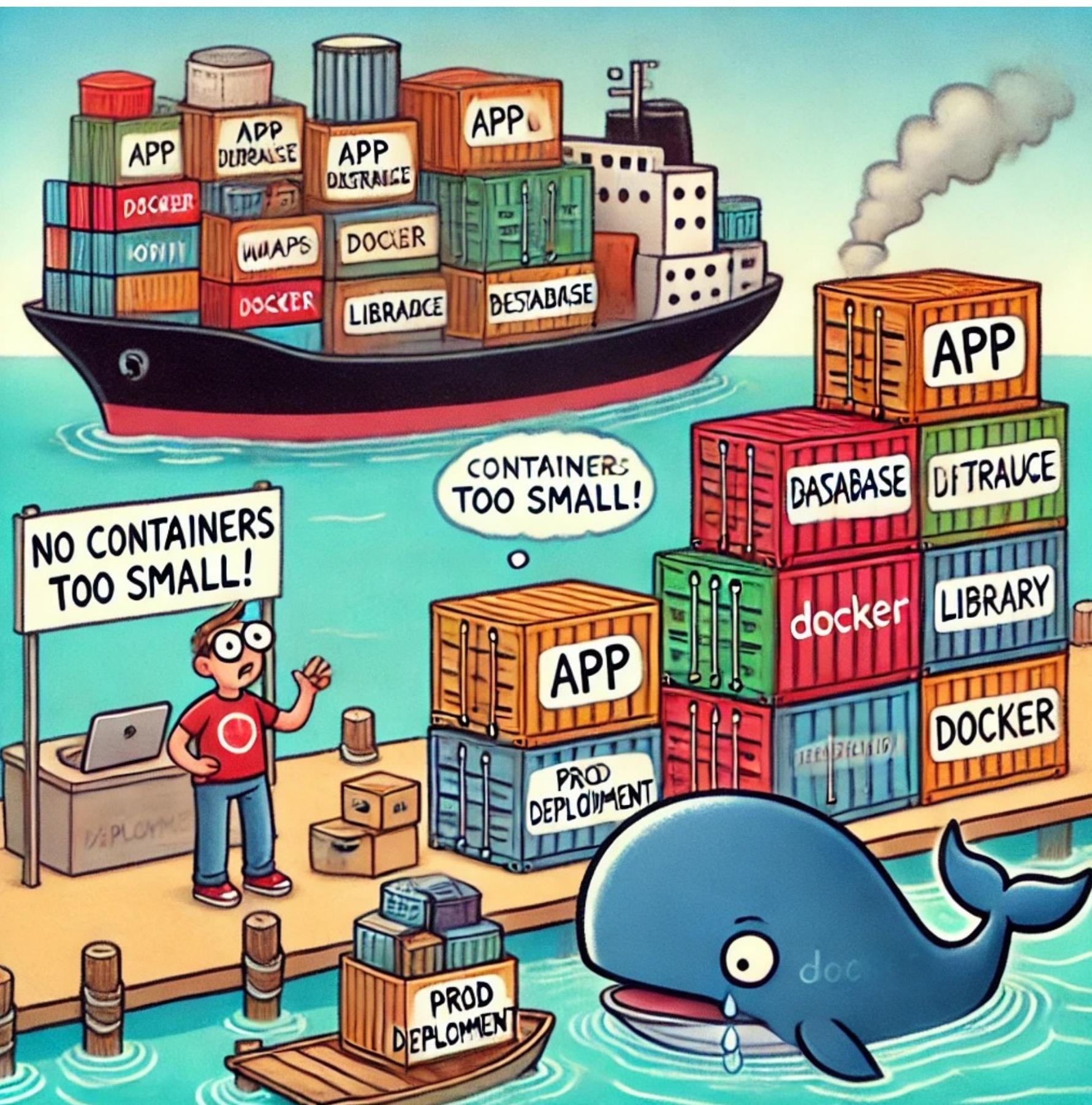
An attacker may cause an HTTP/2 endpoint to read arbitrary amounts of header data by sending an excessive number of CONTINUATION frames. Maintaining HPACK state requires parsing and processing all HEADERS and CONTINUATION frames on a connection. When a request's headers exceed MaxHeaderBytes, no memory is allocated to store the excess headers, but they are still parsed. This permits an attacker to cause an HTTP/2 endpoint to read arbitrary amounts of header data, all associated with a request which is going to be rejected. These headers can include Huffman-encoded data which is significantly more expensive for the receiver to decode than for an attacker to send. The fix sets a limit on the amount of excess header frames we will process before closing a connection.

## Affected Packages

Path	Go Versions	Symbols
<a href="#">net/http</a>	before go1.21.9, from go1.22.0-0 before go1.22.2	▶ 71 affected symbols
<a href="#">golang.org/x/net/http2</a>	before v0.23.0	▶ 37 affected symbols

# Containerization

- Package applications and dependencies together, creating isolated environments
- Run containers consistently across (development, testing, production)
- Start, stop, and scale applications quickly with minimal overhead
- Uniform behavior of applications across multiple environments
- Docker: Widely adopted in DevOps workflows



# Orchestration

## Automates deployment, scaling, and management of containerized applications

- **Load Balancing:** Distributes traffic across containers to ensure optimal performance and availability
- **Self-Healing:** Automatically restarts failed containers
- **Scaling:** Dynamically scale applications up or down based on demand
- **Service Discovery:** Automatically connects services within a cluster
- Kubernetes: Widely adopted in DevOps workflows



# Best Practices

- Encourage small, **frequent commits** to reduce integration challenges
- **Automate Everything:** Build, test, and deployment processes to ensure consistency
- Rapid **feedback** on code changes to identify issues early
- **Comprehensive testing**, including unit, integration, and end-to-end tests, linters and security vulnerability checkers (DevSecOps)
- **Separate environments** for development, testing, and production to avoid conflicts
- Continuously **monitor** pipeline performance and **optimize** for speed and reliability

# Future Trends

- Use of machine learning to **optimize and automate** DevOps processes
- **GitOps**: Manage infrastructure and applications using Git as the source of truth
- Growing adoption of **serverless computing** for cost-effective deployments
- Integration of **security practices**, with emphasis on **proactive** threat detection
- **NoOps**: Towards fully automated operations with minimal human intervention
- Enhanced observability for **deeper insights** into complex systems

# Questions?

- <https://www.ibm.com/topics/devops>
-