Cloud Computing Technologies

DAT515 Fall 2025

DevOps: Best Practices and Tools

Prof Hein Meling

University
of Stavanger


DO
DO NOT
PRESS
CI/CD
DO NOT
PRESS
STOP
DEPLOYE
DEVOPS
DEVOPS
AUTOMATION
AUTOMATION

What is DevOps?

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

What is DevOps?

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

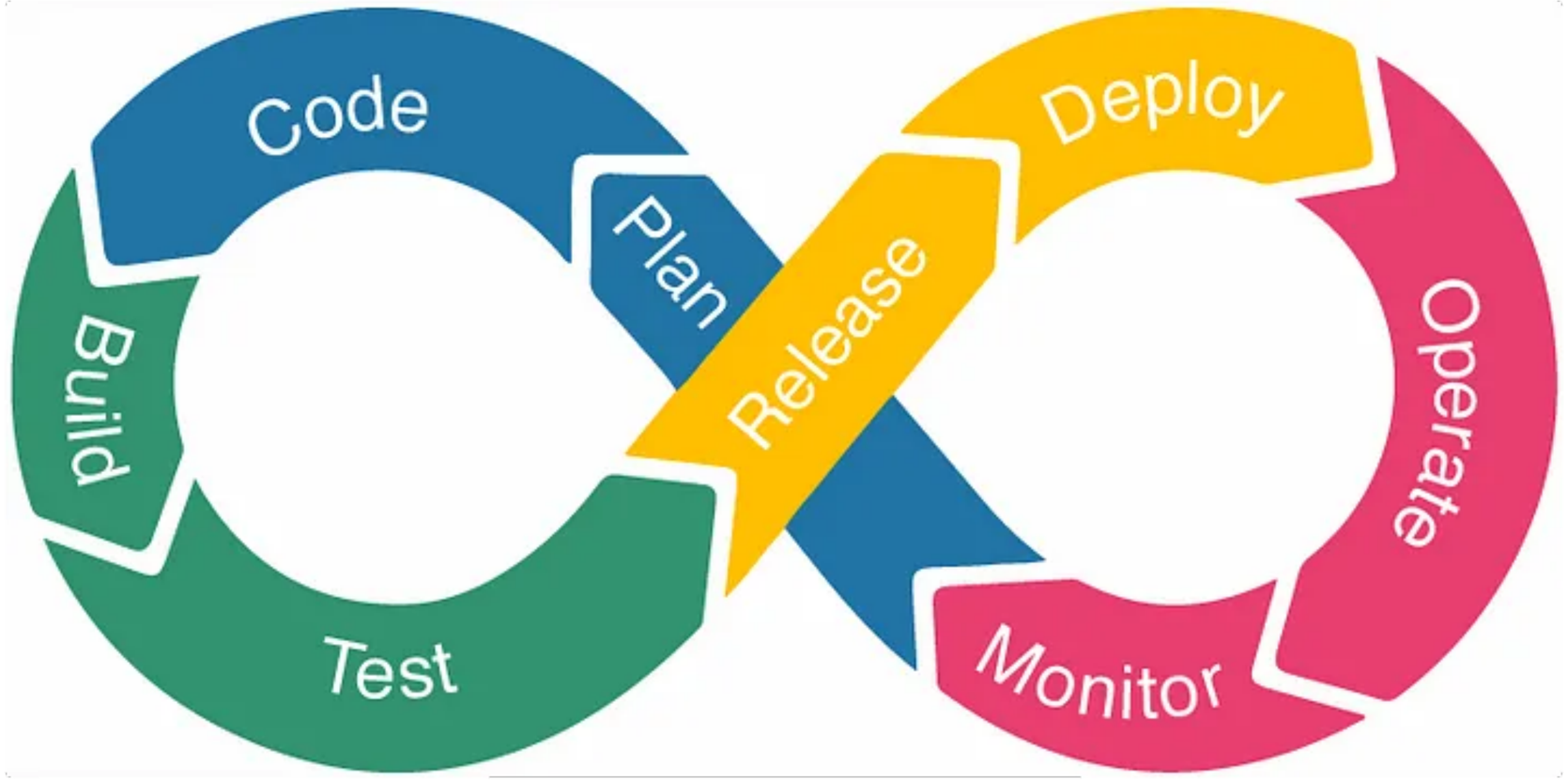


Image: https://medium.com/türk-telekom-bulut-teknolojileri/devops-lifecycle-continuous-integration-and-development-e7851a9c059d

The DevOps lifecycle

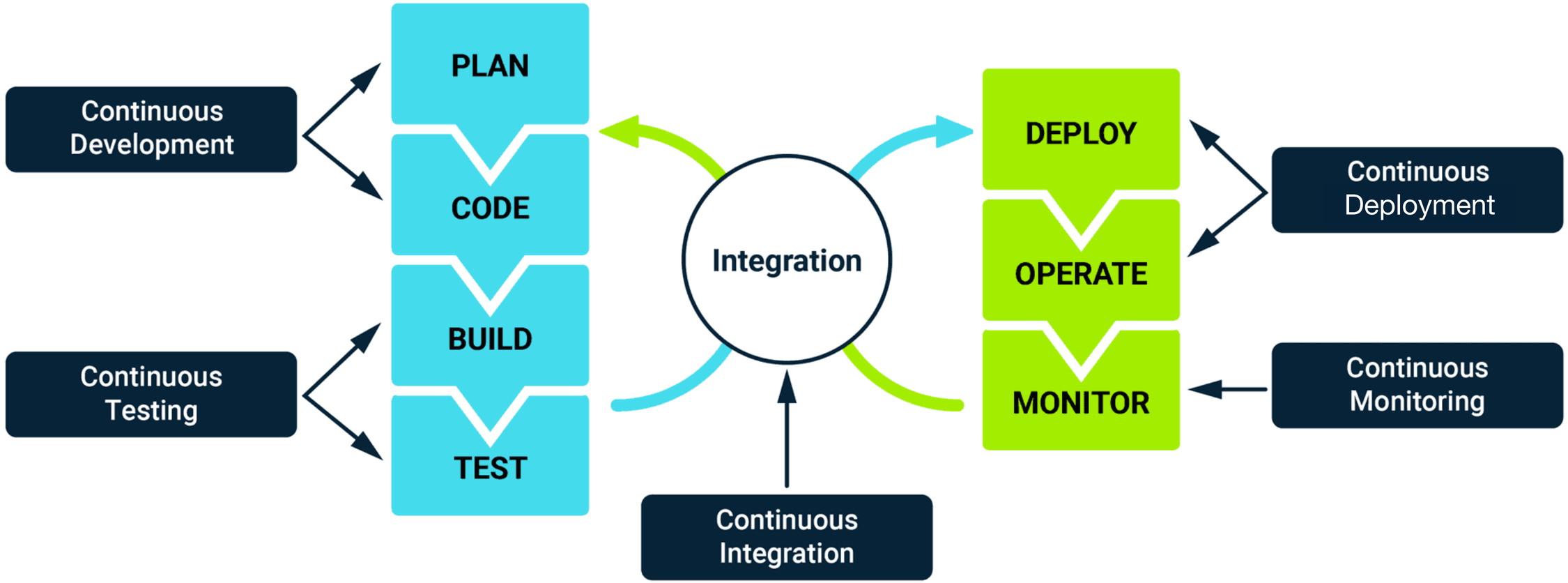
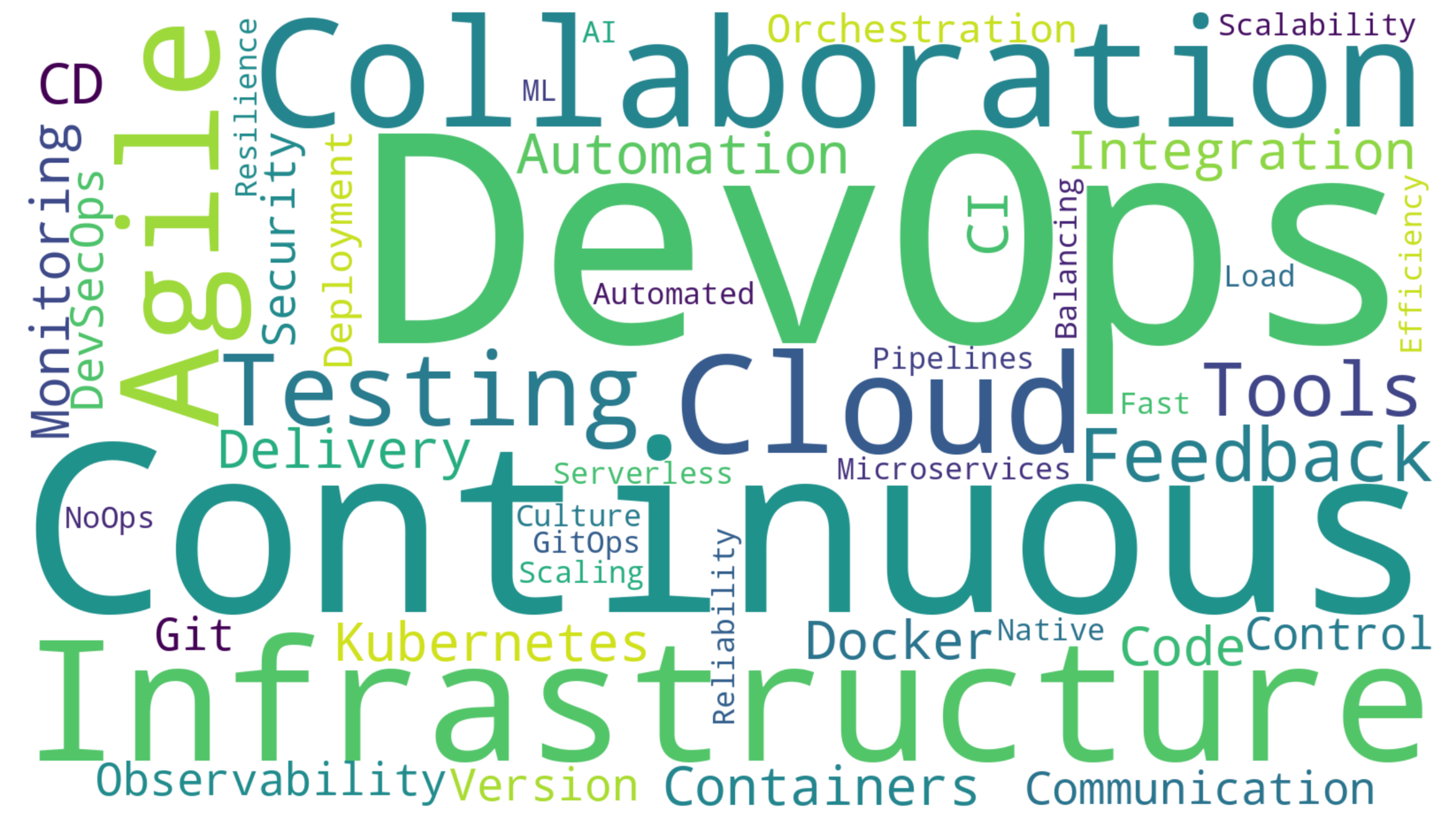


Image: https://codilime.com/blog/devops-lifecycle/


DevOps
Continuous
Infrastructure
Collaboration
Agile
Cloud
Testing
Automation
Integration
Tools
Feedback
Delivery
Monitoring
Security
Deployment
Resilience
DevSecOps
CD
AI
ML
Orchestration
Scalability
CI
Automated
Load
Balancing
Efficiency
Pipelines
Fast
Microservices
Serverless
NoOps
Culture
GitOps
Scaling
Reliability
Git
Kubernetes
Docker
Native
Code
Control
Observability
Version
Containers
Communication

Collaboration

Collaboration in DevOps Teams

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

Collaboration in DevOps Teams

Communication

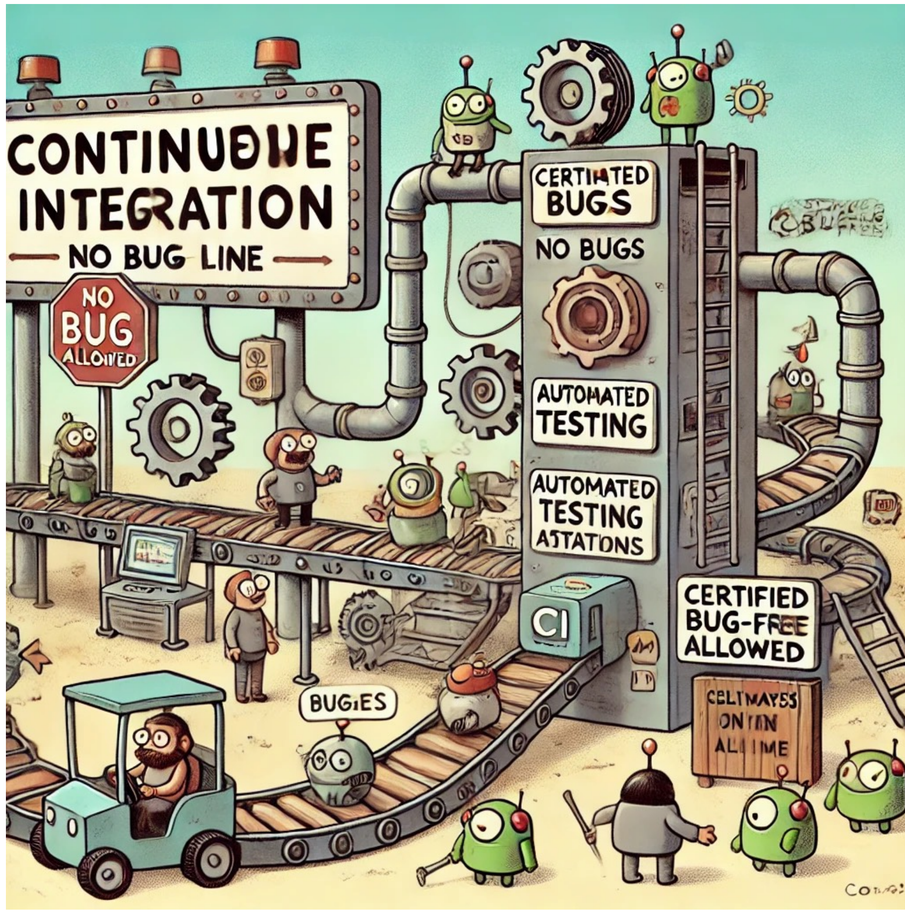
- **Blameless Postmortems**
  - **Post-incident reviews** focused on **learning and improvement**
  - Not assigning blame
- Tools and platforms to enhance collaboration and streamline workflows
  - GitHub issue tracker, Jira, Slack, Discord

Continuous Integration

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

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

Infrastructure as Code (IaC)

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**

Observability Definitions

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

Observability Definitions

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

Observability Definitions

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)
}
```

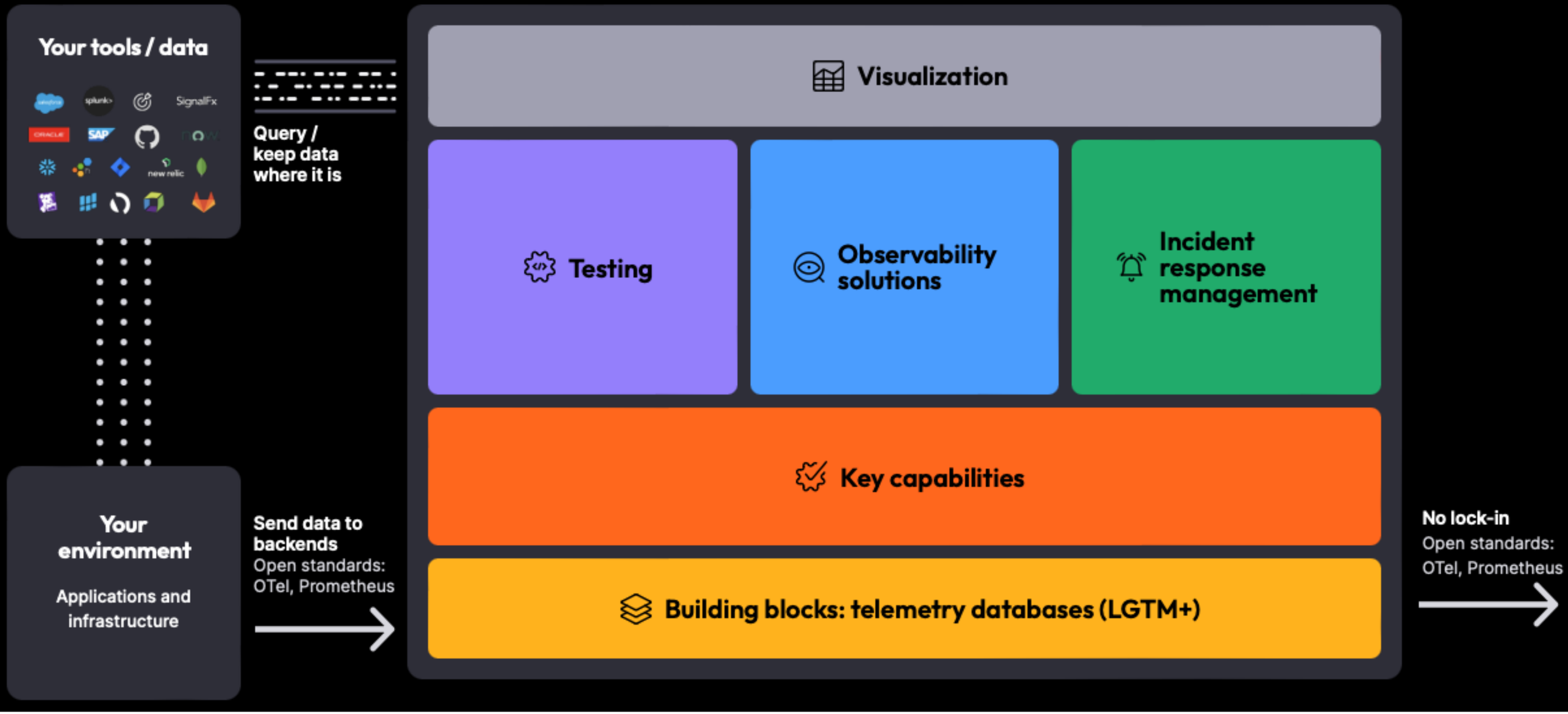
Metrics Collection (in ci package)

```
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
```

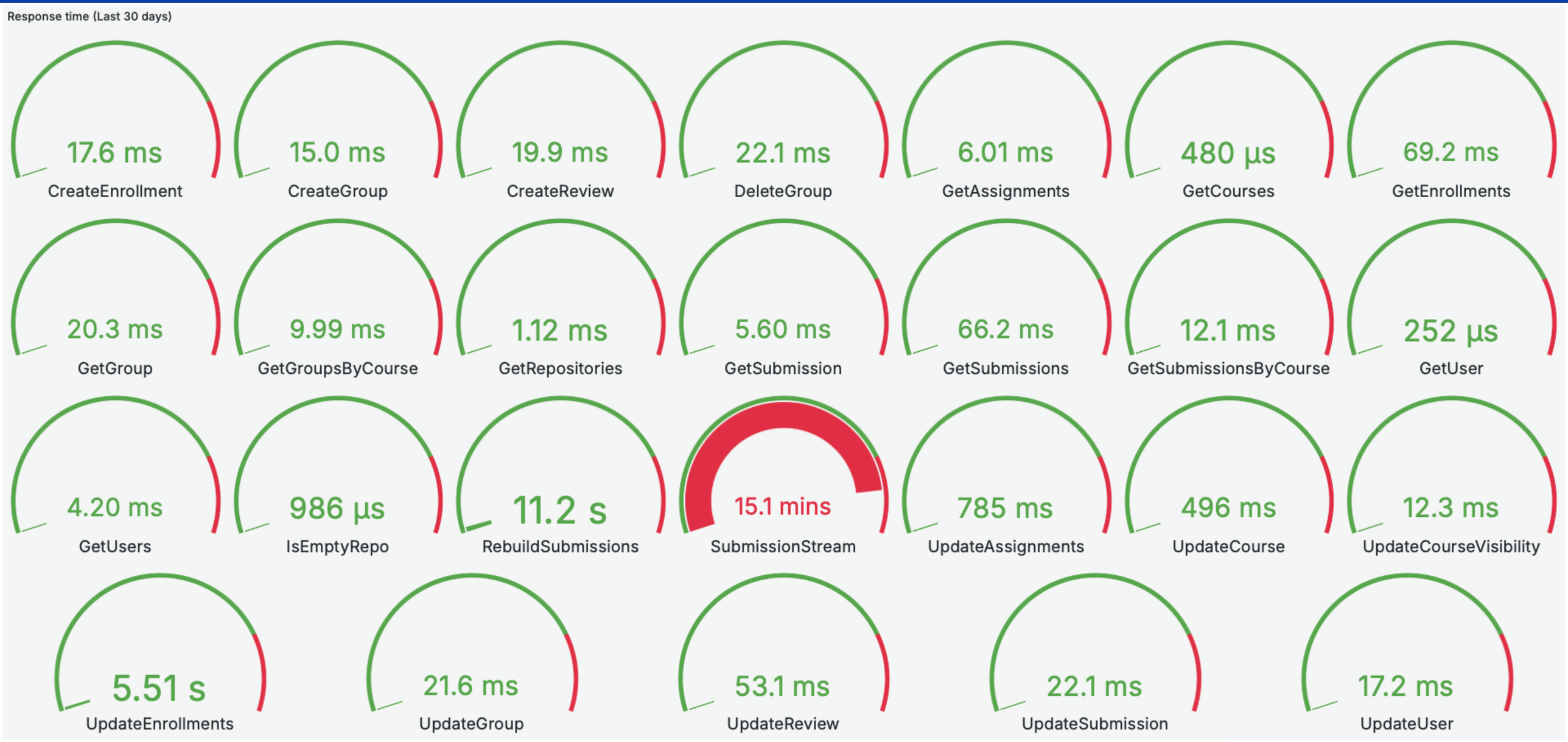
Query the Prometheus Metrics Endpoint

% **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



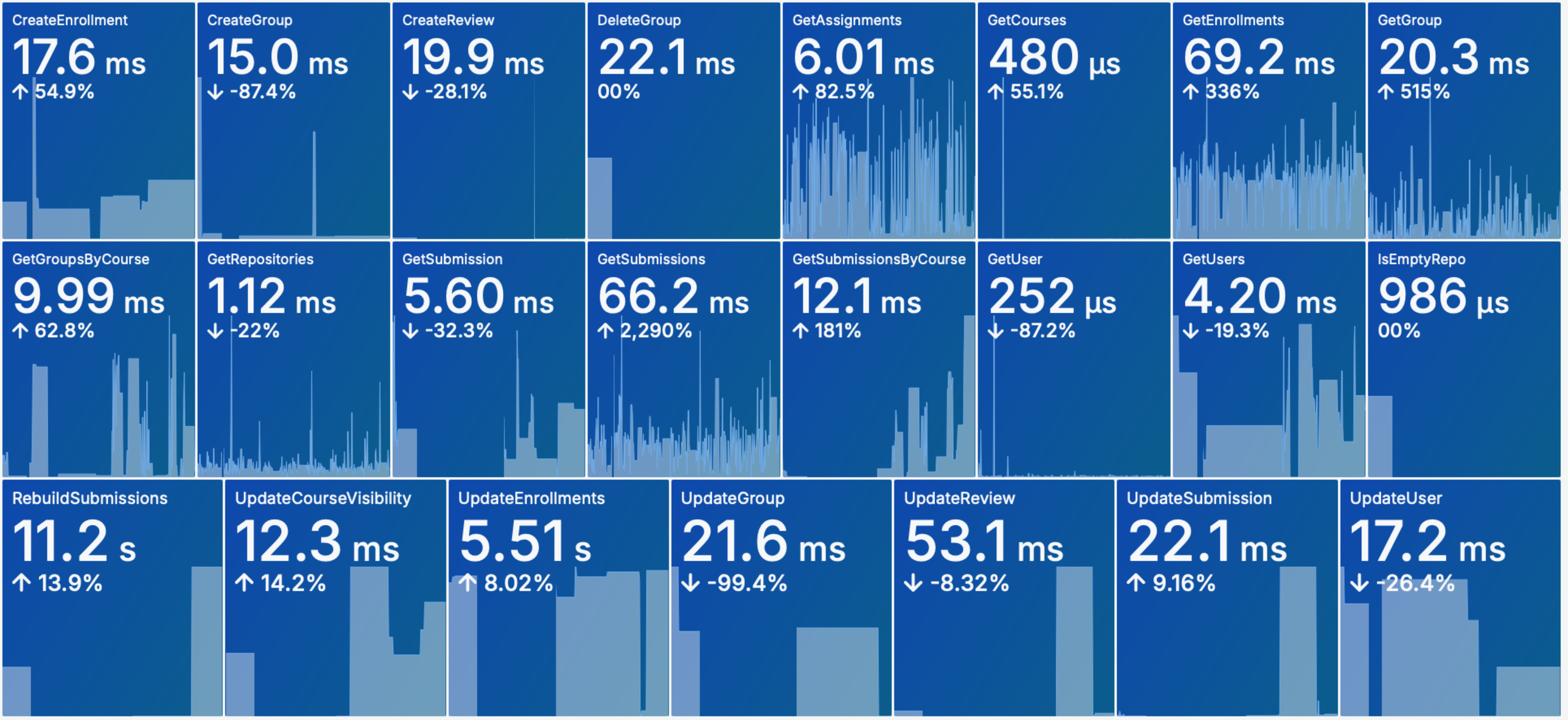
Grafana - Metrics Dashboard



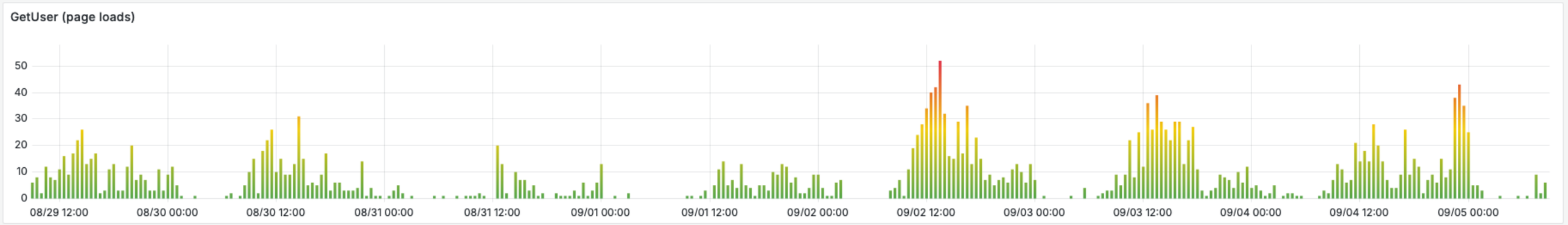
University of Stavanger

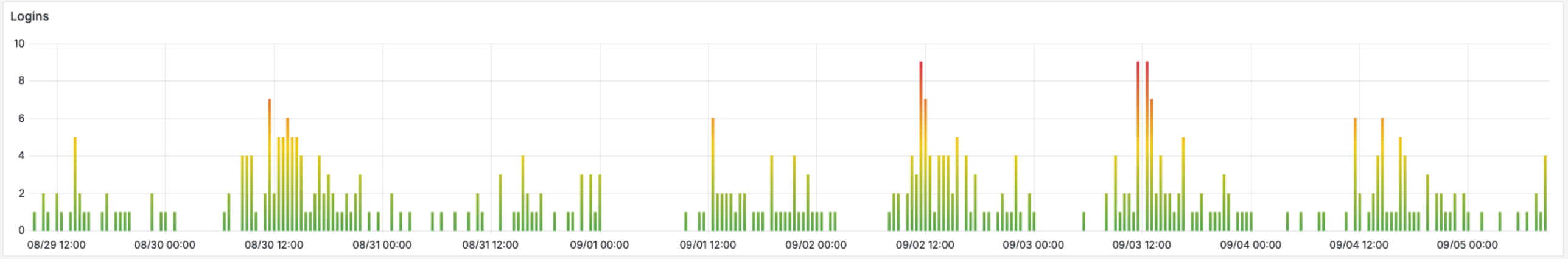
Grafana - Metrics Dashboard

Response time (Last 7 days)

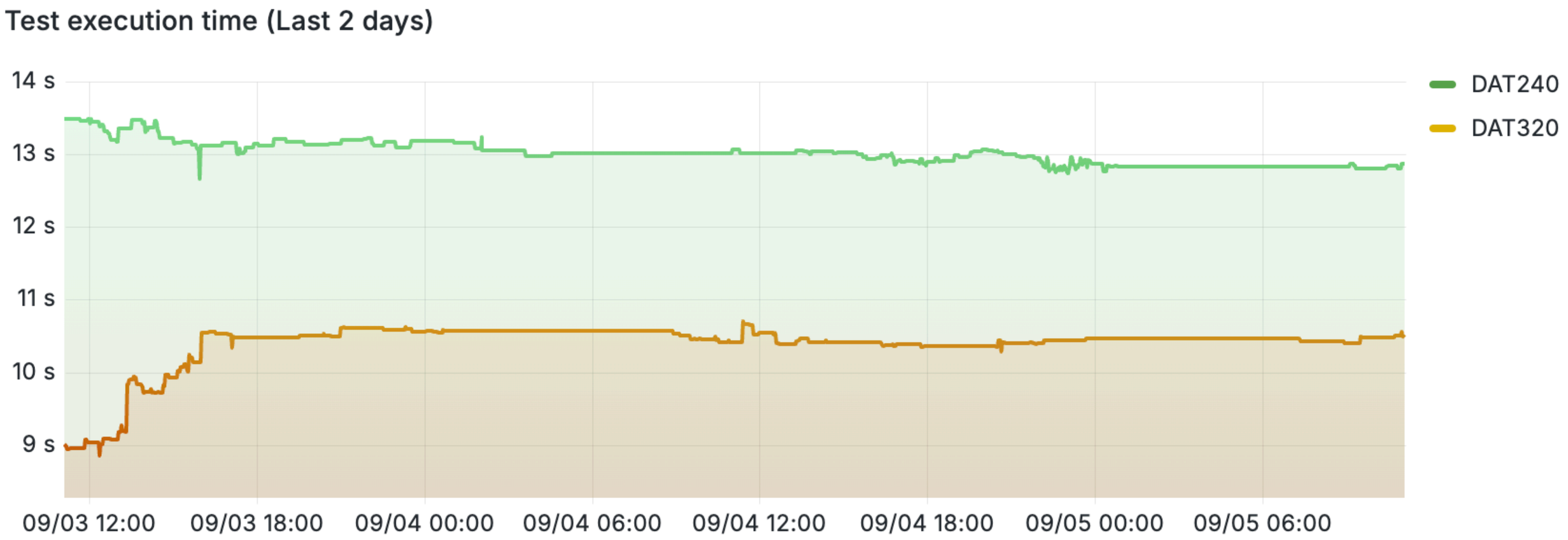


Grafana - Metrics Dashboard





Grafana - Metrics Dashboard



University of Stavanger

Grafana - Metrics Dashboard

Method Accessed Count (Last 90 days)

CreateEnrollment 131	CreateGroup 20	CreateReview 47	DeleteGroup 0	GetAssignments 14.9 K	GetCourses 7.27 K	GetEnrollments 22.3 K
GetGroup 7.62 K	GetGroupsByCourse 1.61 K	GetRepositories 14.6 K	GetSubmission 230	GetSubmissions 22.4 K	GetSubmissionsByCourse 371	GetUser 8.57 K
GetUsers 185	IsEmptyRepo 28	RebuildSubmissions 26	SubmissionStream 14.6 K	UpdateAssignments 0	UpdateCourse 0	UpdateCourseVisibility 74
UpdateEnrollments 250	UpdateGroup 20	UpdateReview 300	UpdateSubmission 134	UpdateUser 38		

Joke time

Why did the developer break up with the DevOps engineer?

Too much automation,

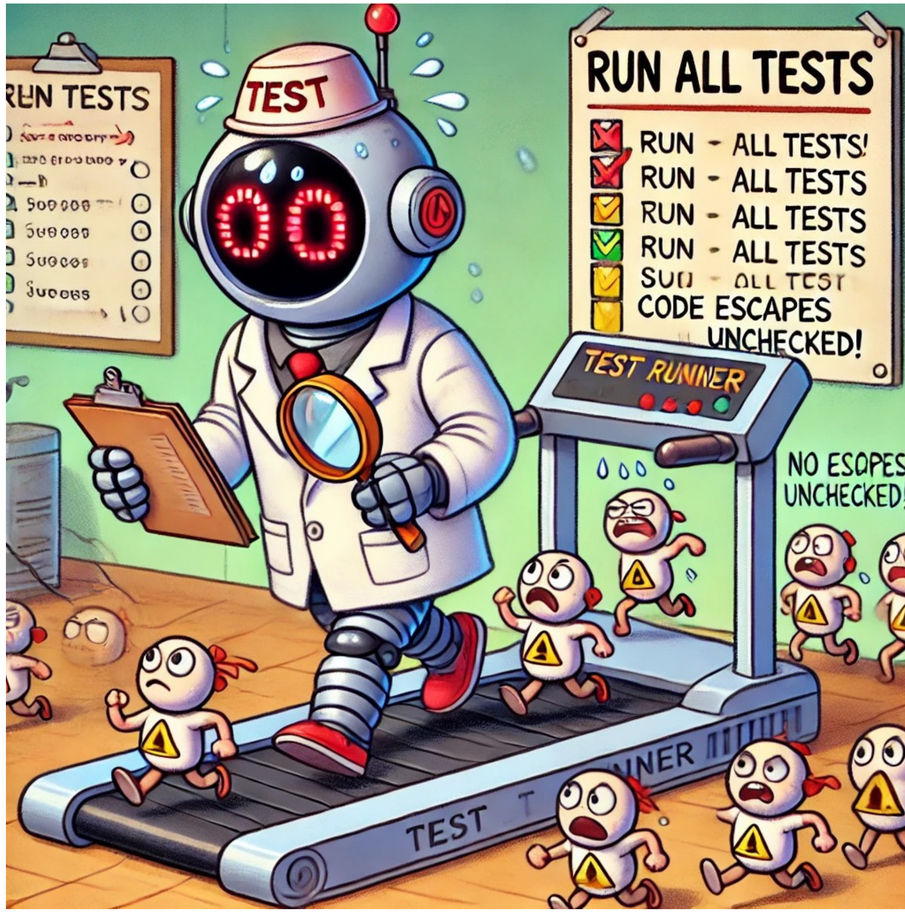
not enough commitment!

Automated Testing

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



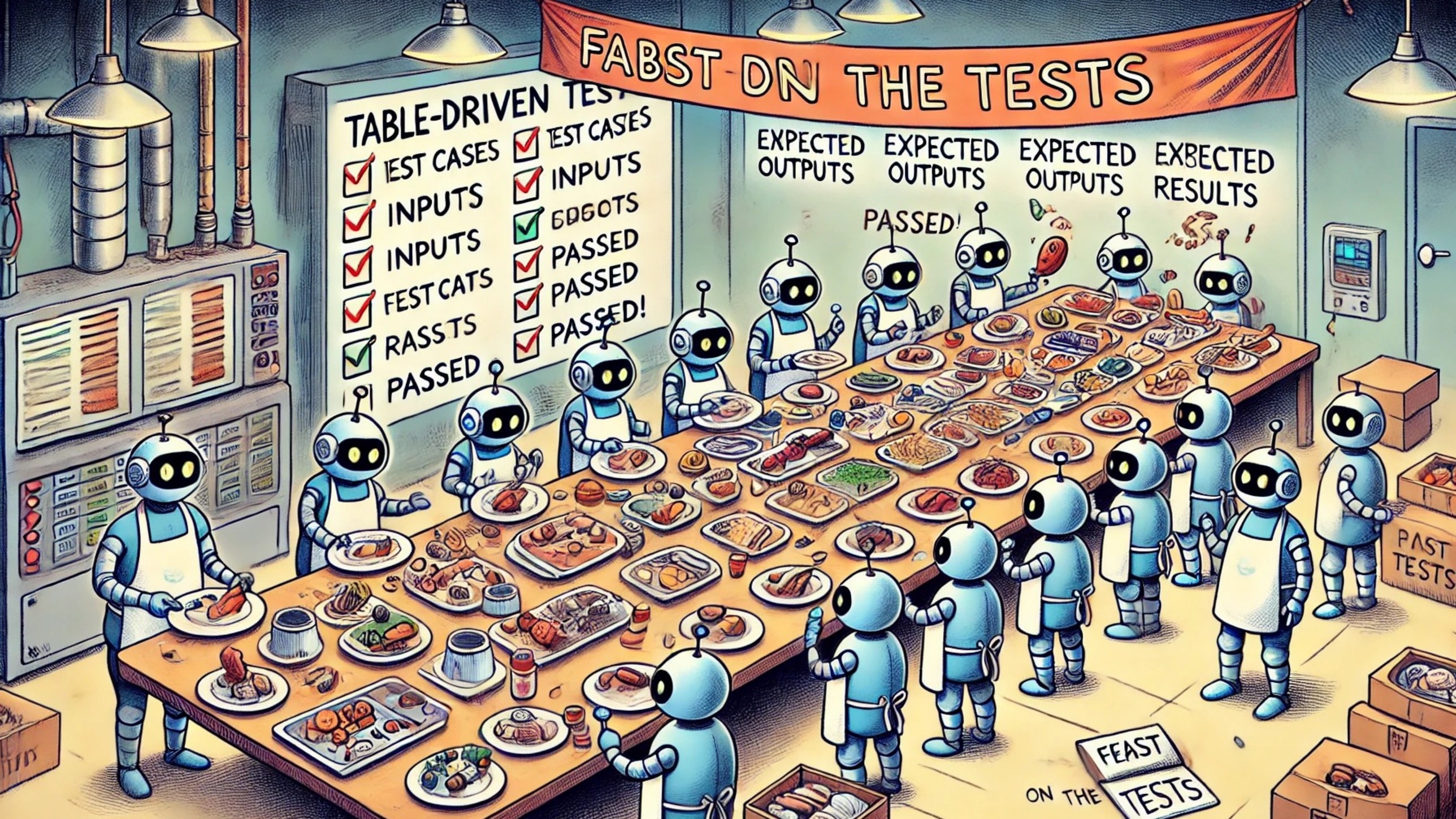

FABST ON THE TESTS
TABLE-DRIVEN TEST
TEST CASES
INPUTS
INPUTS
FEST CATS
RASSTS
PASSED
TEST CASES
INPUTS
GOGOTS
PASSED
PASSED
PASSED!
EXPECTED OUTPUTS
EXPECTED OUTPUTS
EXPECTED OUTPUTS
EXPECTED RESULTS
PASSED!
PAST TESTS
FEAST ON THE TESTS

Table-Driven Tests

quickfeed > scm > github_mock_test.go > ...

run test | debug test

```go
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     }
```

Table-Driven Tests

```
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 }
```

Testing Best Practice

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)
        }
    })
}
```

Running Test Locally

```
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
```

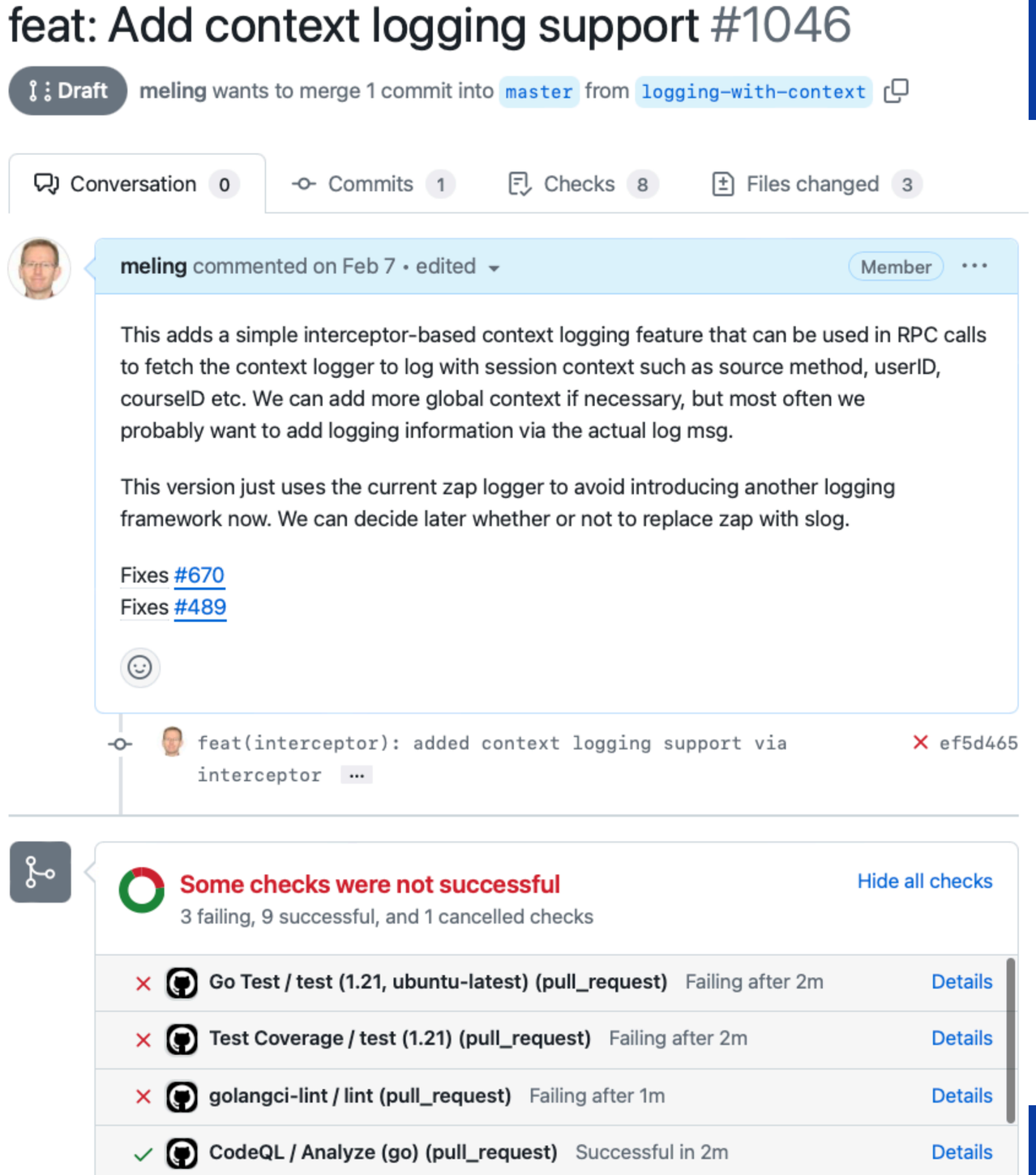
Testing that tests fail on bad input

```
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
```

QuickFeed Pull Request



- Create Pull Request with the gh tool
- gh pr create --title “feat: Add cont…”
- gh pr view --web

Testing QuickFeed

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

```yaml
name: Go Test

on:
  push:
    branches: [master]
    paths:
      - '**.go'
  pull_request:
    paths:
      - '**.go'

jobs:
  test:
    strategy:
      matrix:
        go-version: ["1.22.5"]
        platform: [ubuntu-latest, macos-latest]
    runs-on: ${{ matrix.platform }}

```

- GitHub Actions Workflow

Testing QuickFeed

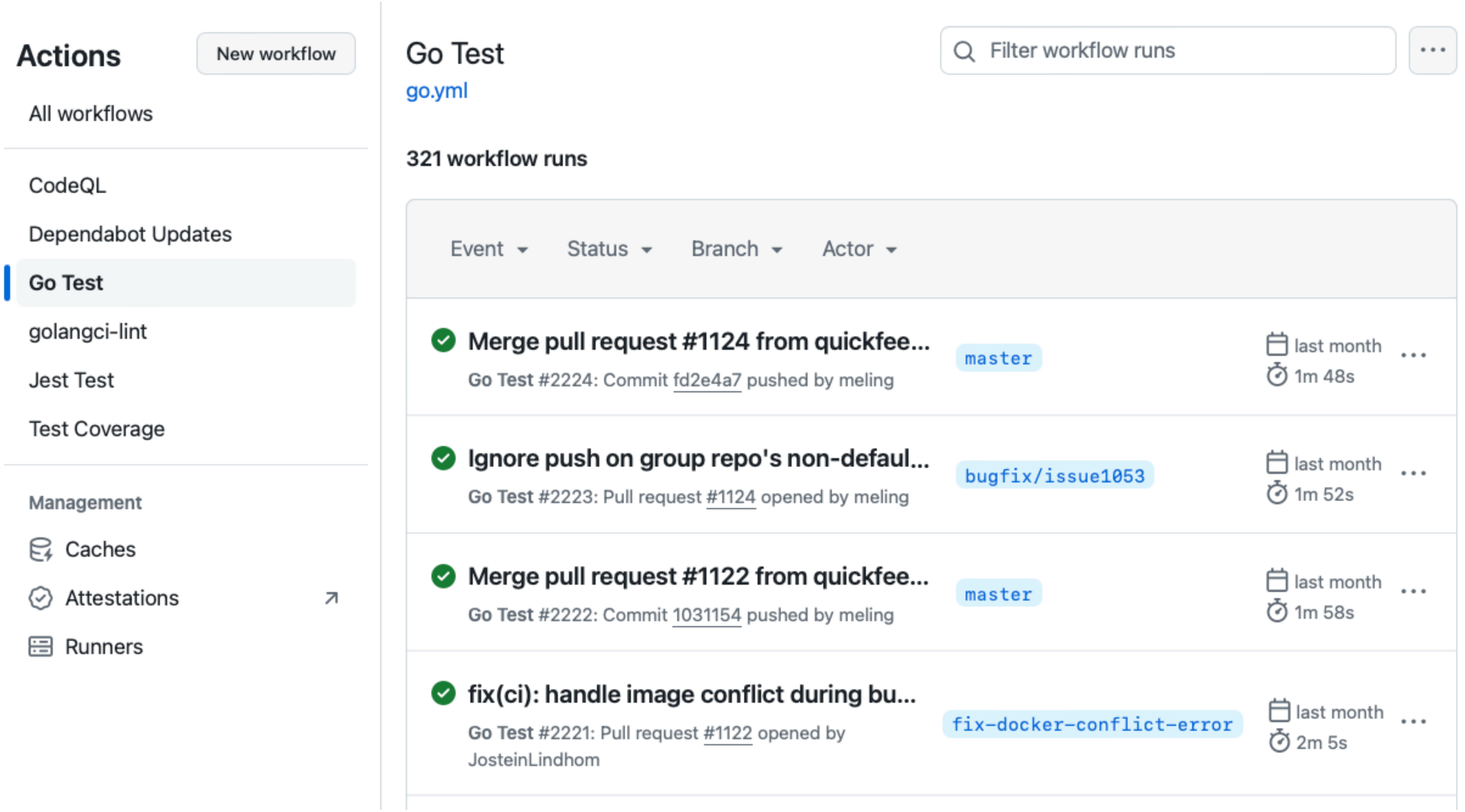
quickfeed > .github > workflows > go.yml

```yaml
name: Go Test

on:
  push:
    branches: [master]
    paths:
      - '**.go'
  pull_request:
    paths:
      - '**.go'

jobs:
  test:
    strategy:
      matrix:
        go-version: ["1.22.5"]
        platform: [ubuntu-latest, macos-latest]
    runs-on: ${{ matrix.platform }}

```

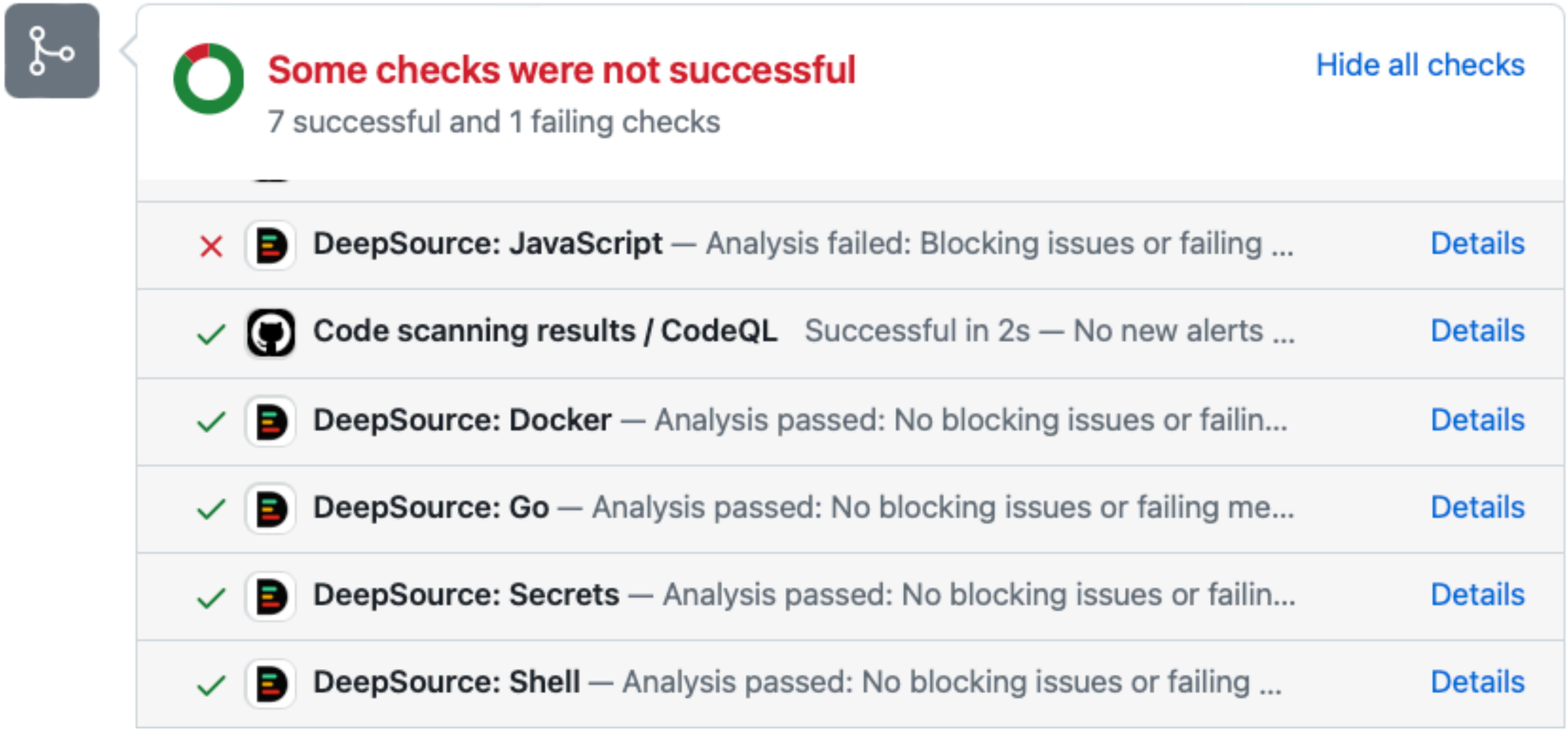
Testing QuickFeed

```yaml
steps:
  - name: Cache
    uses: actions/cache@v4
    with:
      path: |
        ~/.cache/go-build            # ubuntu-latest
        ~/Library/Caches/go-build  # macos-latest
        ~/go/pkg/mod
      key: ${{ runner.os }}-go-${{ hashFiles('**/go.sum') }}
      restore-keys: |
        ${{ runner.os }}-go-

  - name: Checkout code
    uses: actions/checkout@v4

  - name: Set up Go
    uses: actions/setup-go@v5
    with:
      go-version: ${{ matrix.go-version }}

  - name: Run Go tests
    run: DOCKER_TESTS=1 go test -v github.com/quickfeed/quickfeed/...
```

Testing QuickFeed



DeepSource Scanner



quickfeed / quickfeed ACTIVE

Overview | Issues | Metrics | Reports | History | Settings

Performance issues	Style issues	Documentation issues
121	3	0
Anti patterns	**Security issues**	**Coverage issues**
158	0	0

ACTIVE ISSUES	ISSUES PREVENTED
305	4.6k
DEFAULT BRANCH	**ANALYZERS**
master	+2

Last analyzed 6d374c6 3 days ago

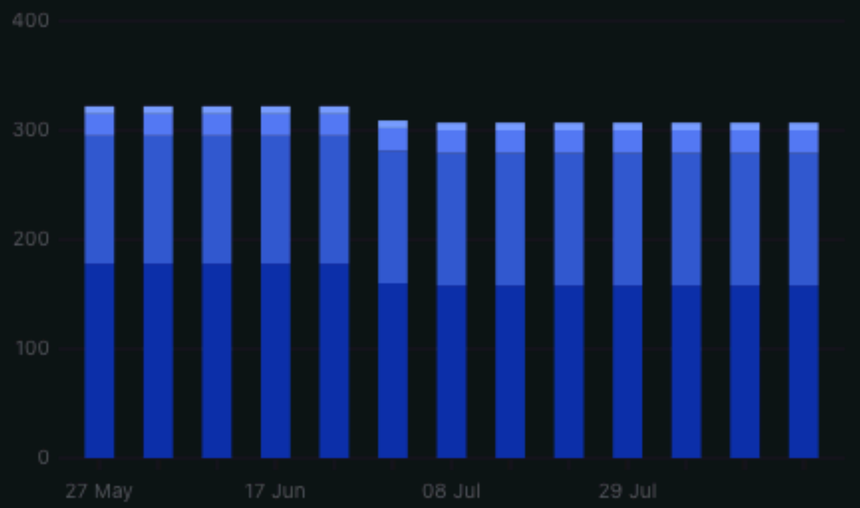
OWASP® Top 10

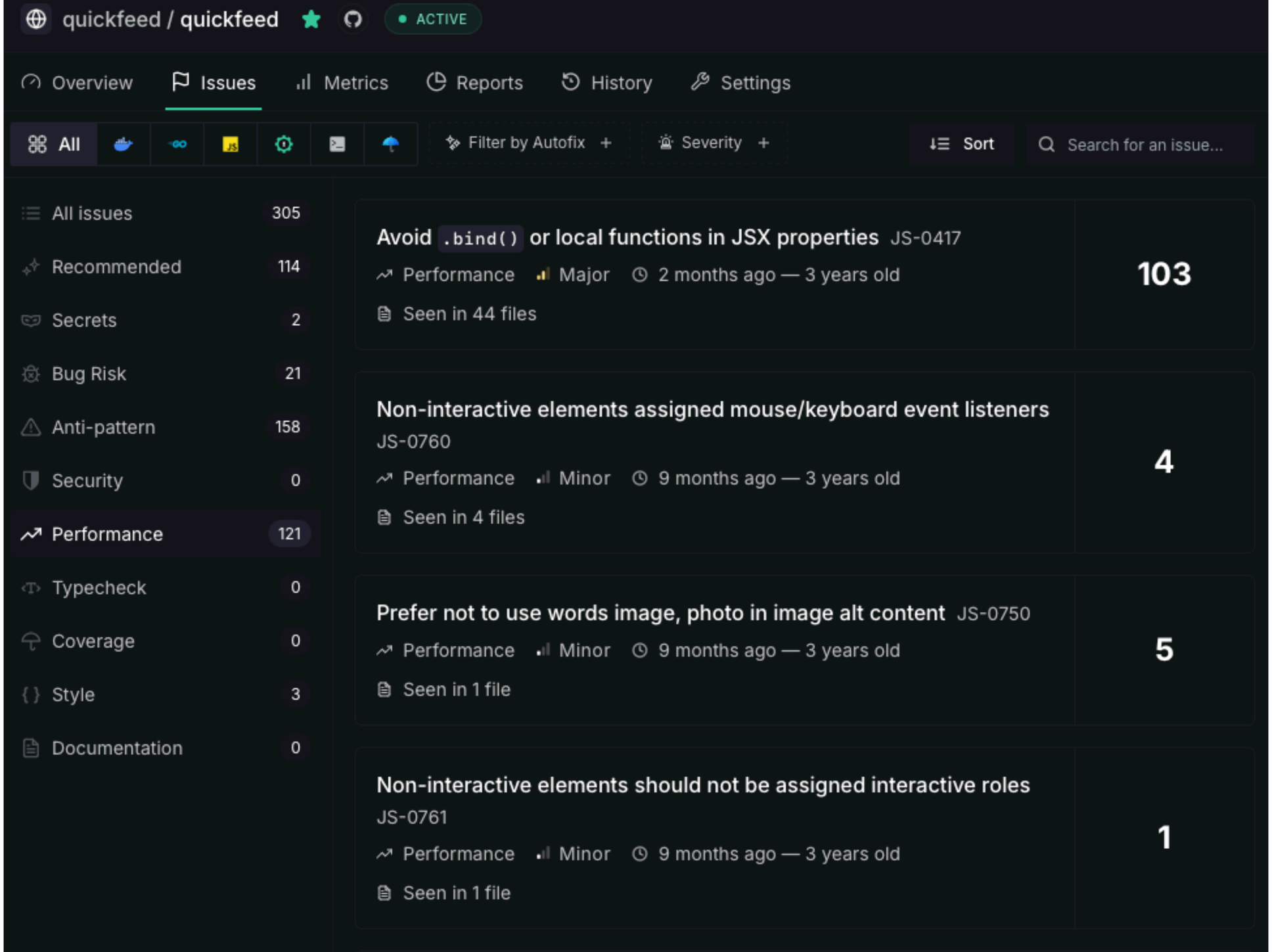
PASSING 0 active issues

Issue Distribution by Category

305 active issues

Anti-pattern Performance Bug risk Others




quickfeed / quickfeed
ACTIVE
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 DevOps engineers stay calm?

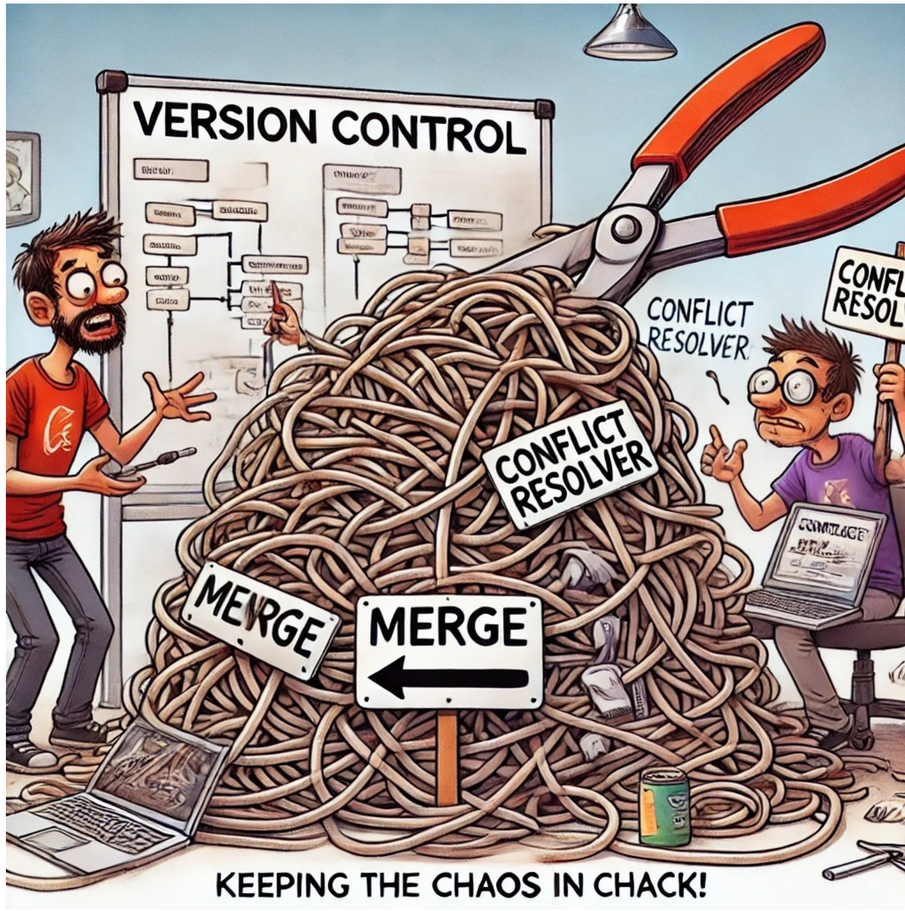
They automate their stress away.

Version Control

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

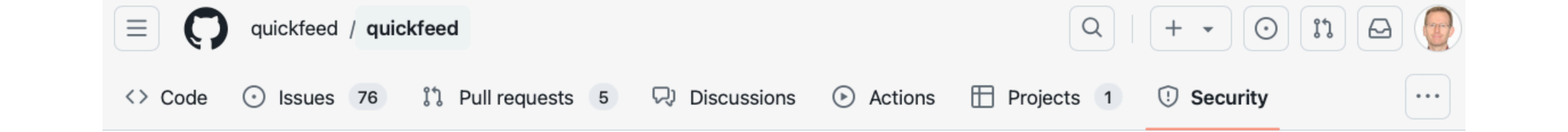
- Works seamlessly with Git repositories and workflows
- Branches themselves are tracked and versioned, not just commits
- Safe, local-first design – experimentation without risk of losing work
- Powerful undo and redo – Easily backtrack or reapply changes
- Flexible and intuitive – Command set designed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

DevSecOps

Security in DevOps

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**



Overview

Reporting

Policy

Advisories

Vulnerability alerts

Dependabot

Code scanning

Secret scanning

Security overview

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

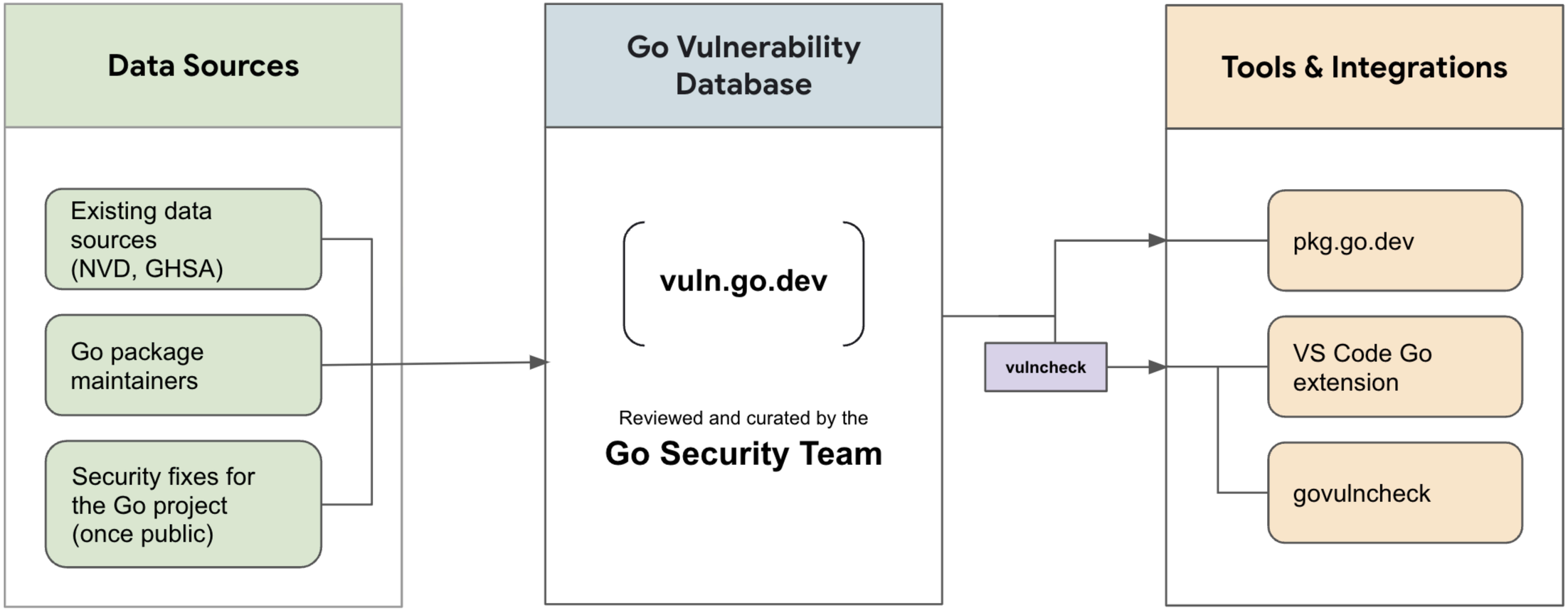
Private vulnerability reporting • **Enabled**
Allow users to privately report potential security vulnerabilities
See reported vulnerabilities

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

Go Vulnerability Tooling



Go's govulncheck Tool

- cd work/quickfeed
- go install golang.org/x/vuln/cmd/govulncheck@latest
- govulncheck ./…
  - Wait a minute or so…
  - No vulnerabilities found.

GovulnCheck on Gorums

```
~/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.WriteHeaders
      #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.WriteWindowUpdate
      #15: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.Setting.String
      #16: node.go:162:21: gorums.RawNode.FullString calls fmt.Sprintf, which eventually calls http2.SettingID.String
      #17: server.go:170:27: gorums.Server.Serve calls grpc.Server.Serve, which eventually calls http2.SettingsFrame.ForeachSetting
```

Search Go's Vulnerability Database

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
net/http	before go1.21.9, from go1.22.0-0 before go1.22.2	▶ 71 affected symbols
golang.org/x/net/http2	before v0.23.0	▶ 37 affected symbols

Containerization

Containerization for DevOps

- 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

Orchestration for DevOps

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

CI/CD Pipeline 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

Future Trends in DevOps

- 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?

References

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