GitOps

Theoretical & Practical Approaches to Git-driven Infrastructure

Guest Lecture
Cloud Computing
Frankfurt University of Applied Sciences

3rd December 2024

claranet®
Make modern happen.

About

Domenico Caruso
Team Lead – Cloud Native Engineering
Claranet GmbH

- Master in Theoretical and Computational Chemistry
- Over 10 years of experience in IT industry
- Leading and working together with a team of cloud-, kubernetes engineers

claranet® Make modern happen.

About

- Bachelor in Computer Science
- Over 5 years of experience in IT industry
- Working together with cloud-native and kubernetes engineers
- Kubernetes, GitOps, Automation

Kai Schäfer
Senior Cloud Native Engineer –
Claranet GmbH

claranet® Make modern happen.

About the Team

- Multiple teams embracing software engineering, cloud based and native workload, linux & windows
- Several platforms: AWS, GCP, Azure and on-premise
- International: based in Germany, Spain, India with over 6 nationalities and languages
- Annual team and family event

claranet® Make modern happen.

About: Claranet group

At a glance

- Founded in 1996
- Owner- managed
- 600 Mio € annualised revenues
- More than 10.000 B2B customers
- Global reach with operations in 11 counties
- More than 3.500 employees

We are experts for modernizing and running critical applications, data and infrastructures 24/7

claranet®
Make modern happen.

Highly accredited with cloud vendors

Compliance

ISO 9001:2015

ISO 22301:2019

ISO 27001:2017 27017:2015 27018:2014

STAR SELF-ASSESSMENT

Trusted Cloud SERVICE 10005

SOC 2 - TYPE II ISAE 3402

CLOUD COMPUTING BSI C5 COMPLIANCE CRITERIA CATALOGUE

CHECK 28

Partnerships

Google Cloud Partner – Managed Service Provider

Gold Microsoft Partner – Microsoft – Azure Expert MSP

aws PARTNER Managed Service Provider

aws PARTNER Premier Tier Services

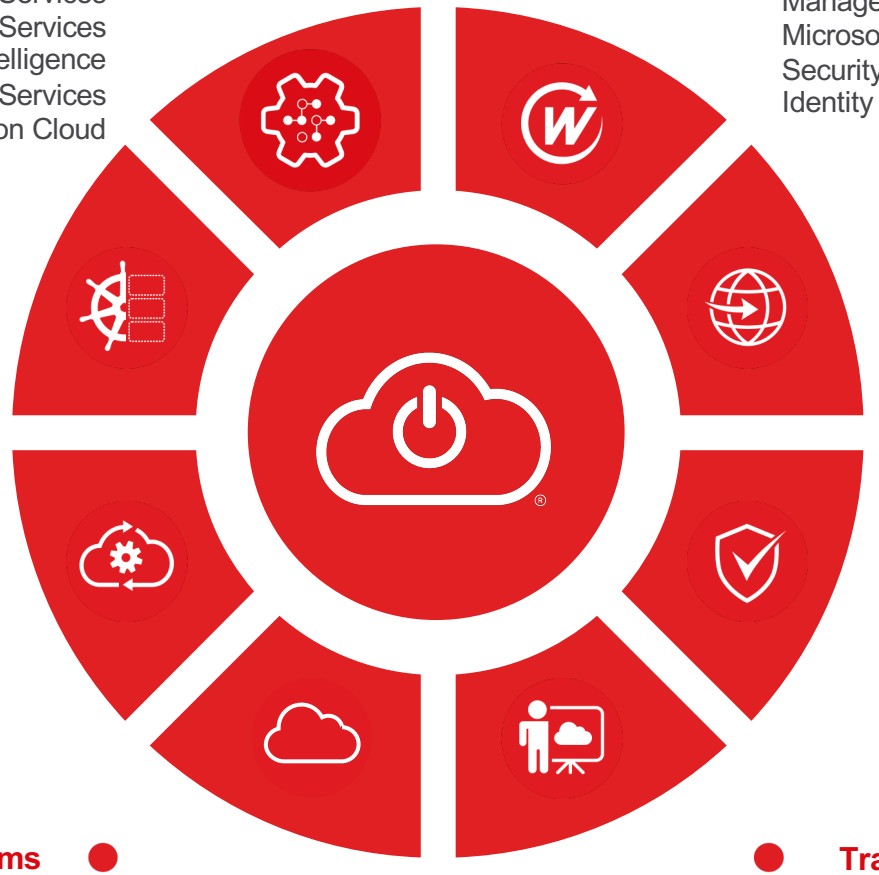
aws PARTNER Select Tier Training

SAP Silver Partner

Trainer Skills & Certified SAP Consultant – SAP Authorized Trainer – 2021

Zertifizierungen und Partnerschaften beziehen sich u.U. nur auf einzelne Unternehmen der Claranet Gruppe

Claranet Service Portfolio



SAP
Managed Services
Consulting Services
Business Intelligence
Security Services
SAP on Cloud

Workplace & Collaboration
Managed Services
Microsoft 365
Security / Endpoint Management
Identity Management

Cloud Native
Managed Container Applications
Managed Kubernetes
DevOps Automation
Cloud Native Strategies

Network Services
Cloud Connect
MPLS
IPsec
SSL

Cloud Services
Cloud Migration
Cloud Management
Managed Applications

Cyber Security
Security Assessments
Penetration Testing & Red Teaming
Security & Compliance Consulting
SOC Services: EDR, MDR
Penetration Testing as a Service

Cloud Platforms
AWS, GCP, Azure
Private Cloud
Hybrid Cloud

Training
Microsoft
SAP
AWS
Linux
Cyber Security

24/7
Business SLA, Monitoring, Reporting, Service Management

What expects you in this lecture?

- GitOps simplifies infrastructure and application deployments by using Git as the single source of truth. This lecture explores the history of infrastructure management, the rise of GitOps, and its unique advantages over traditional DevOps

- **Lecture Goals:**
  - Understand the core concepts of GitOps.
  - Identify the abstractions it introduces.
  - Evaluate if the added complexities are beneficial

Agenda

- Challenges of modern IT
- How did traditional IT worked?
- Demo
- How can git help us?
- gitOps
- Demo

claranet®

Who heard of GitOps outside of this lecture?

Who gained real-world experience with gitOps?

claranet®

Make
modern
happen.

What challenges in modern IT?

Frequent Deployments

In today’s fast-paced environment, teams need to deploy new features, updates, and fixes rapidly

Frequent deployments increase the risk of errors and inconsistencies

Consistency

Ensuring that development, staging, and production environments are identical is essential

Differences in configurations can lead to unexpected issues in production

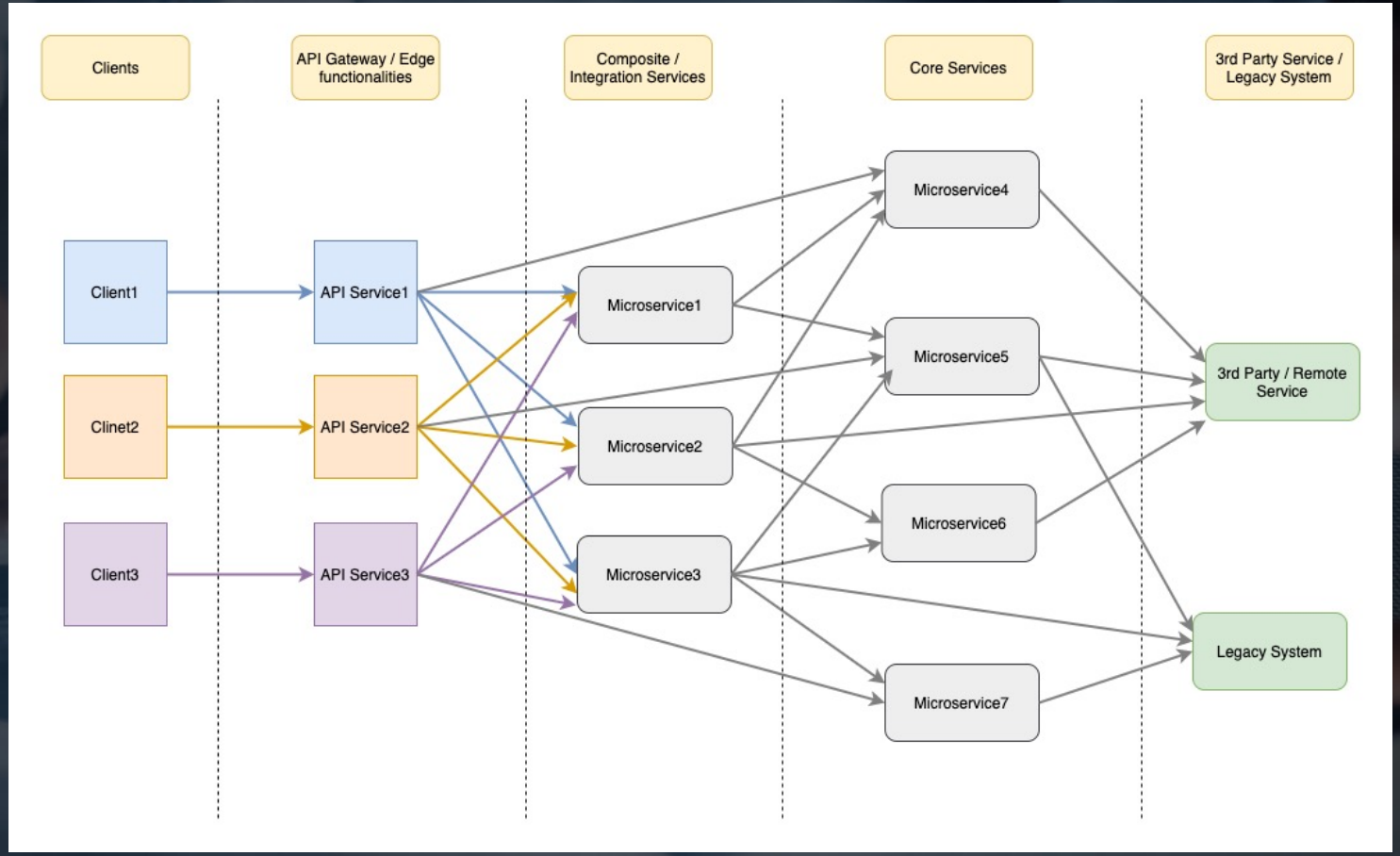
Traceability and Control

With many team members making changes, it’s crucial to know who changed what, when, and why.

Without good traceability, troubleshooting and accountability suffer

What challenges in modern IT?

- **Modern applications can have hundreds of microservices making up the whole application**
- **If you want to keep high level of releases, consistency and traceability you can't work the same way IT used to work until few years ago**

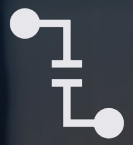


What are then our goals?

Agility

Agility emphasizes rapid, iterative development and delivery to meet evolving user needs and market demands.

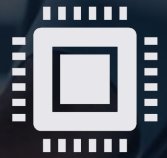
Teams aim to quickly respond to feedback and continuously improve software.



Parallelism

In modern development, teams work on multiple features and services simultaneously.

This parallelism speeds up innovation but also demands strong coordination to avoid conflicts.



Scalability

Microservices are designed to be independently scalable, allowing each service to scale up or down based on demand.

This differs from a monolithic application on a single VM, where scaling is often limited to the entire system.

Managing this dynamic scaling manually is inefficient and impractical, especially at the pace and scale required by modern applications.

Resilience and Fault Isolation

In a microservices architecture, each service operates independently, so failures in one service don’t necessarily bring down the entire system.

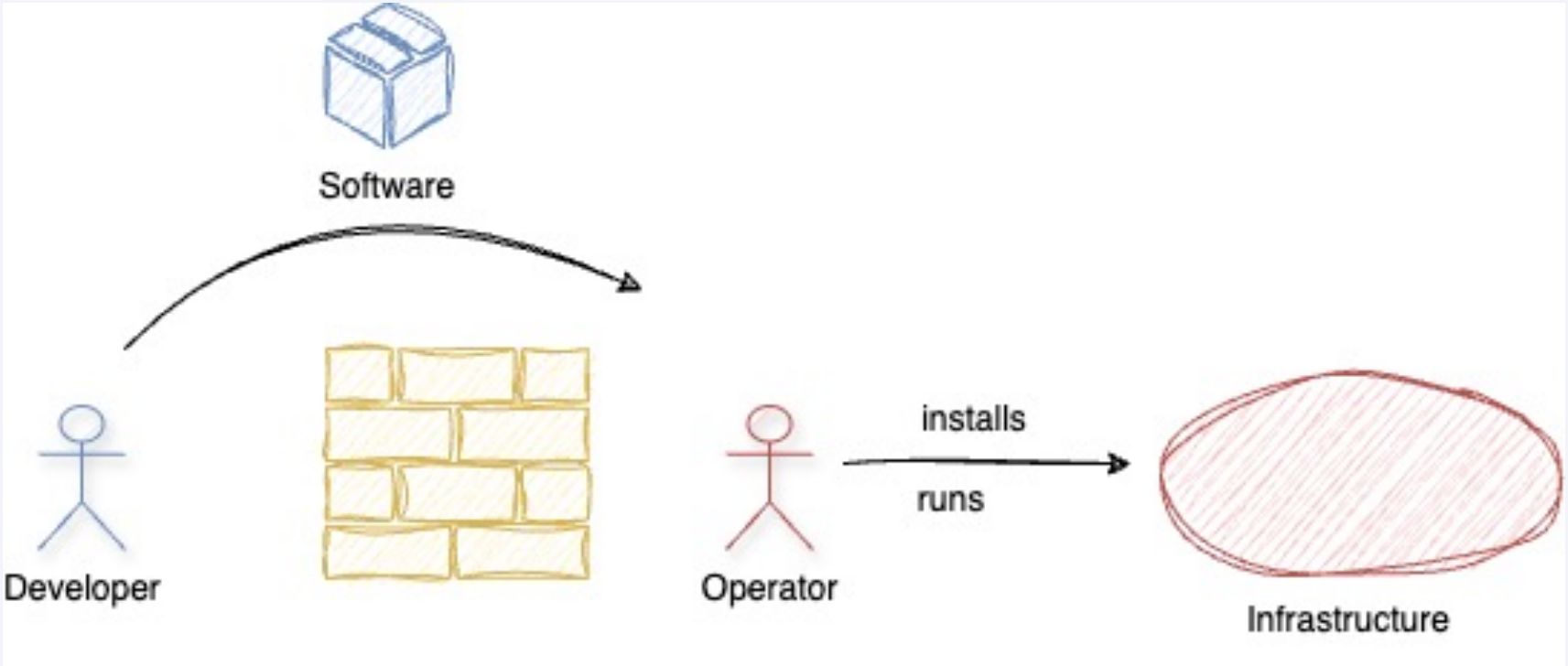
However, this only works well if services can be independently managed, deployed, and recovered.

Traditional approaches that treat the system as a single, unified block are less suited to ensuring this level of resilience, as they lack the flexibility needed to handle failures in isolated parts of the system.


What is Ops and what is Git?
claranet®
Make
modern
happen.

Traditional Ops (Operations)

- **Silos, the old-fashioned way**
  - Developers want to change much and often
  - Operators don’t touch it if it works
  - Limited release cycle



Traditional Ops (Operations)

Manual Configuration and Deployment

Traditionally, "Ops" involved manually setting up servers, networks, and storage. Deploying applications often required hands-on configuration

Reactive Maintenance and Troubleshooting

Ops teams were often seen as those jumping in to fix issues as they arose in production. Monitoring and responding to system alerts was a major focus, with a primary goal of ensuring uptime and minimizing downtime.

Separate from Development

Historically, Operations and Development were often siloed. Developers built features, and Operations deployed and maintained them.

Focus on Stability over Speed

Operations traditionally prioritized stability, with changes happening cautiously to avoid disruptions.

Challenges of traditional Ops

Lack of Traceability

- With manual processes and ad-hoc configurations, it's difficult to track what changes were made, when, and by whom

No Drift Detection

- Configuration drift, where the actual environment deviates from the intended configuration, is a common issue

Inconsistent Environments

- Manually configured environments often lead to inconsistencies between development, staging, and production

Siloed Teams and Communication Barriers

- Traditional operations often keep development and operations teams in separate silos, leading to poor communication and misaligned goals

Difficulty Scaling

- Scaling infrastructure quickly to meet demand is challenging when configurations are handled manually

Limited Rollback and Recovery Options

- When issues arise, rolling back to a previous state can be difficult without a version-controlled approach

DEMO:
traditional Ops

claranet®

Make
modern
happen.

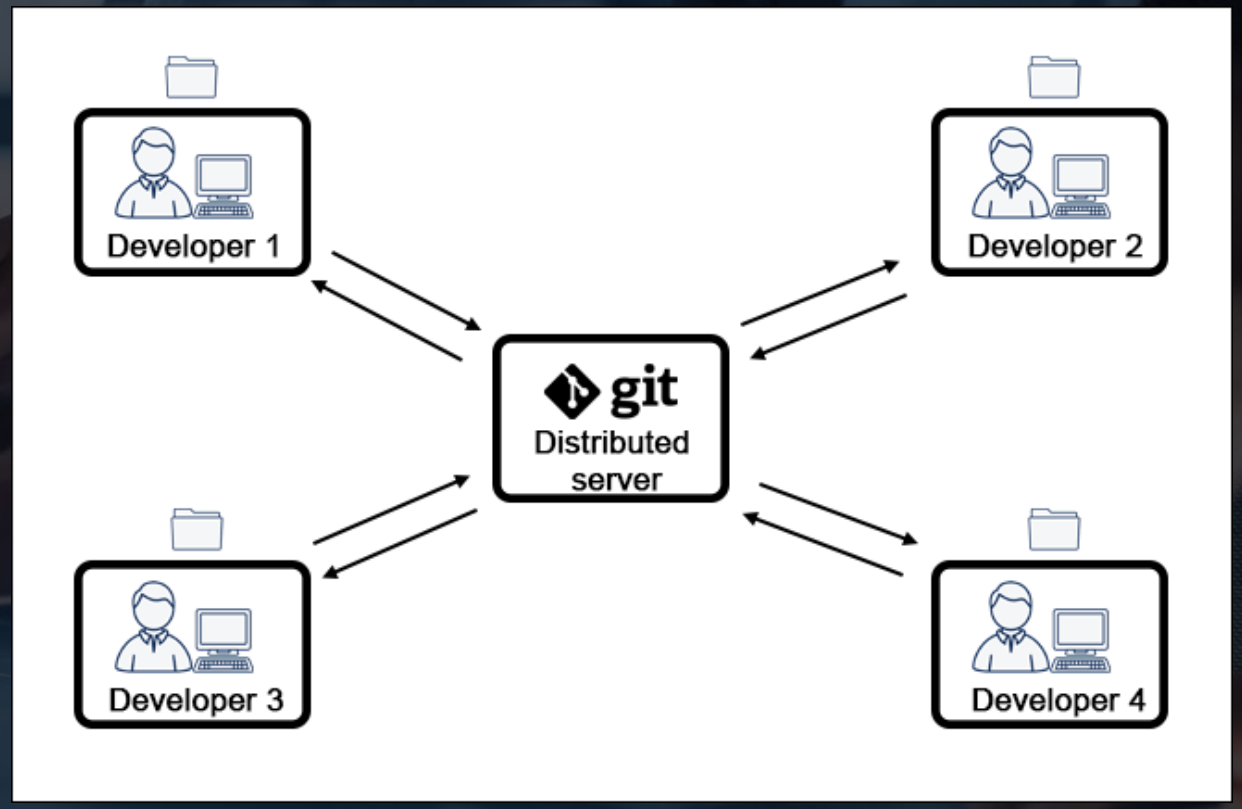
What is Git?

Git – What is it?

Version Control System
Git is a distributed version control system that helps teams manage changes in code

Distributed and Collaborative
Git enables each user to have a local copy of the repository

Commit History and Branching
Git records every change as a “commit” and allows for branching—creating separate lines of development that can later be merged back.



Git – also possible for Infrastructure

Infrastructure as Code (IaC)
As teams began defining infrastructure in code, it became logical to store this code in Git

Immutable Infrastructure
The rise of immutable infrastructure—where servers are replaced rather than modified—means deployments rely on fixed configurations that need to be consistently defined and tracked

Microservices
Microservices architecture, with its many independent, modular components, requires precise management of each service's configuration

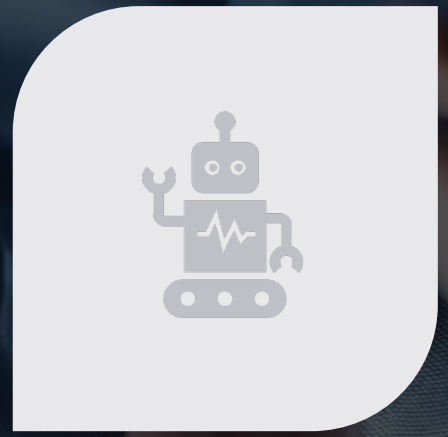
Agility and Speed
Infrastructure teams need to keep up with fast-paced changes, including frequent deployments and iterative improvements.


What is GitOps?
claranet®
Make
modern
happen.

GitOps

Declarative Configuration (IaC)

Versioning (Git)



Automation via CI/CD

GitOps – Declarative Configuration

Declarative Configuration (IaC)

- **Definition of desired end state of infrastructure**
- **Consistent & predictable**
- **Transparent & human readable**
- **Reproducibility and Immutable**

GitOps – Declarative Configuration

```
resource "google_compute_instance" "vm_application" {
  name         = "application-vm"
  machine_type = "n1-standard-1"
  zone         = "europe-west3-a"

  boot_disk {
    initialize_params {
      image = "ubuntu-os-cloud/ubuntu-2004-lts"
    }
  }
}
```

GitOps – Declarative Configuration

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: guest-lecture
spec:
  template:
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        volumeMounts:
          - name: page
            mountPath: /usr/share/nginx/html/index.html
            subPath: index.html
```

GitOps - Versioning

- Single point of view
- History & Auditability
- Collaboration via branches and reviews
- Disaster Recovery & Rollback

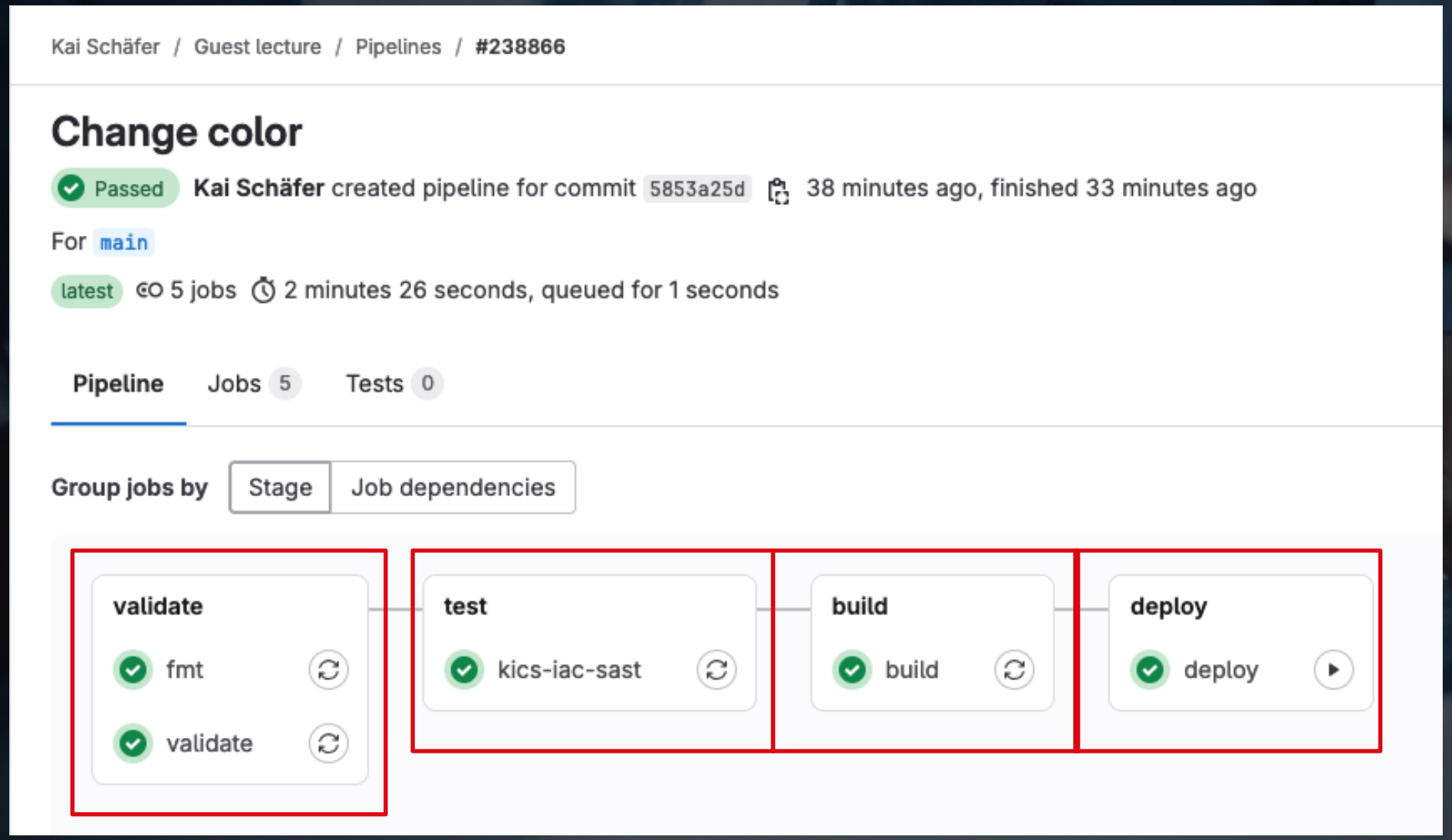
Versioning (Git)

GitOps – Automation

Automation
via CI/CD

- **Compliance and security**
- **Automated testing after deployment**
- **Controlled way & reproducible**
- **Notification about the rollout state**

GitOps – Automation




Demo GitOps
claranet®
Make
modern
happen.

GitOps – Benefits & Challenges

- Benefits
  - Disaster Recovery & Rollback
  - Visibility of state
  - Auditing/Security
  - Reconciliation
- Challenges
  - Slightly more overhead
  - Higher complexity
  - Steep learning curve

GitOps – Conclusion

- Highly effective in containerized and microservices environments
- Ease complexities by centralizing configuration
- Enhanced Security and Compliance
- Consistent and automated deployments regardless of the underlying platform


Questions ?
claranet®
Make
modern
happen.

Wanna join the Cloud Native movement?

Claranet, a place for talented people, partner-like customers, collaborative culture, personal growth, innovative technologies, vibrant international community

- Bachelor- / Master theses
- Cloud Native Engineer
- Cloud Native Consultant
- DevOps Engineer
- Kubernetes Engineer
- Site Reliability Engineer

Check out https://www.claranet.de/jobs


claranet®
Make modern happen.