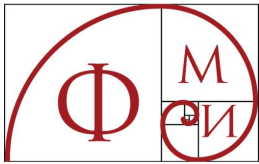
Several Clouds

Modern DevOps Practices

Prepared for

Faculty of Mathematics and Informatics (FMI)



Program overview

Several Clouds

1. Initial meeting
2. Software Development Life Cycle (SDLC)
3. Working with version control systems
4. Microservices and Docker
5. Pipelines
6. Continuous Integration
7. **DevSecOps**
8. **Continuous Delivery**
9. Code Assistants
10. Database versioning
11. Kubernetes
12. Cloud services in AWS
13. Infrastructure as Code with Terraform

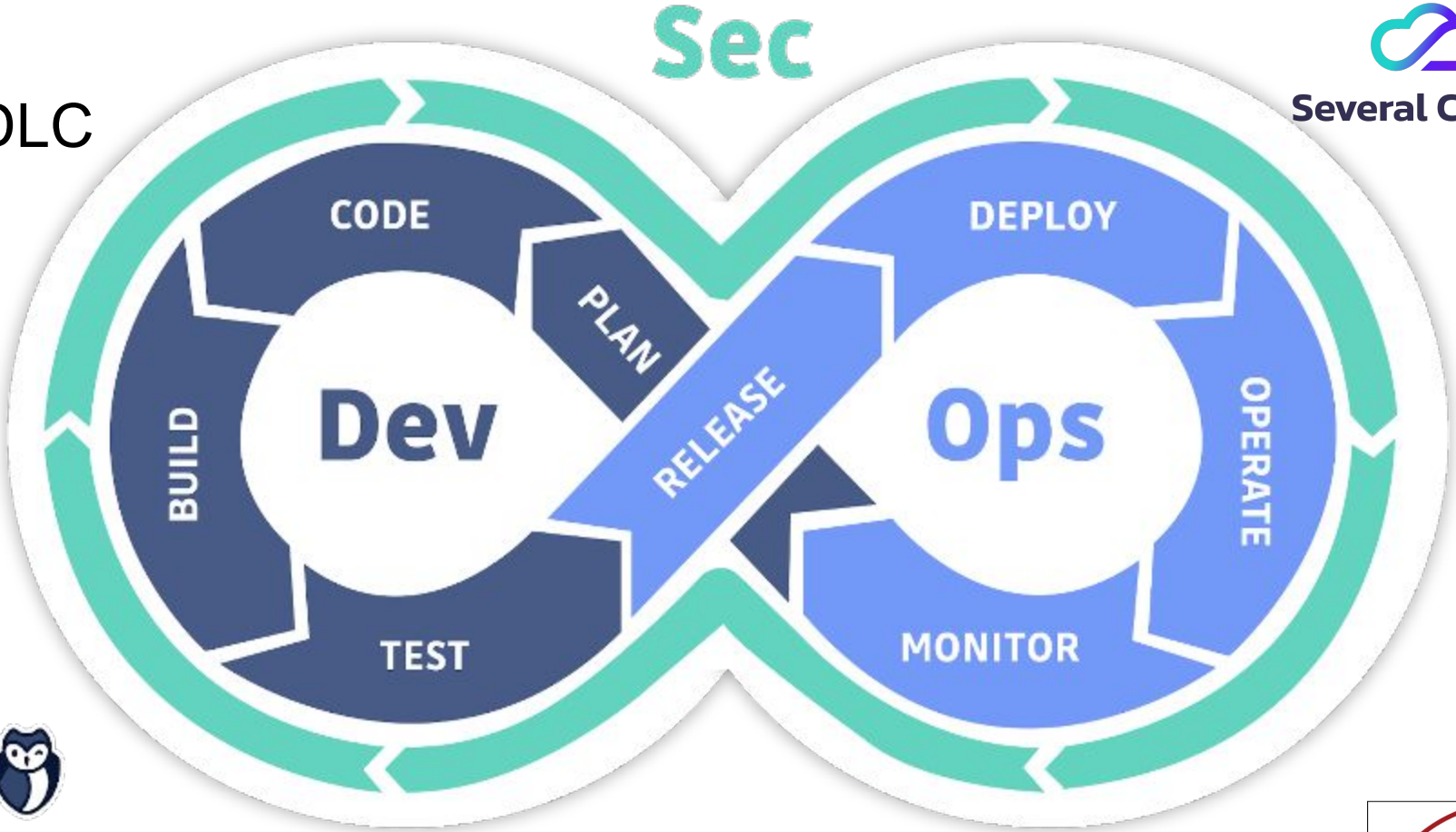
Several Clouds

DevSecOps

Prepared for

Faculty of Mathematics and Informatics (FMI)

SDLC



Several Clouds

What is DevSecOps

Several Clouds

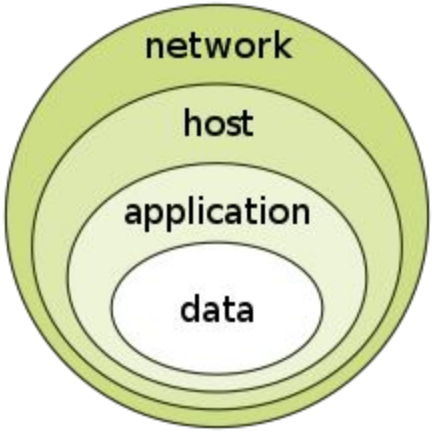
- Collaboration
- In the **past** Security teams were involved **last** in the SDLC

- Create fast feedback loops for Security
- “Shift security left”

What is DevSecOps

- Culture
- Tools

The “onion” information security model



Several Clouds

Multi-layered approach

Several Clouds

What are the components of your software solution?

Software composition analysis

Several Clouds

Configuration and Secrets

Software that you build

Dependencies - OSS and binaries

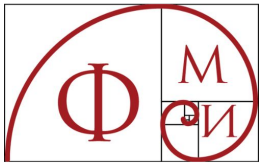
Application runtime

Operating System

Hypervisor Operating System

Hardware Security

Data Centers

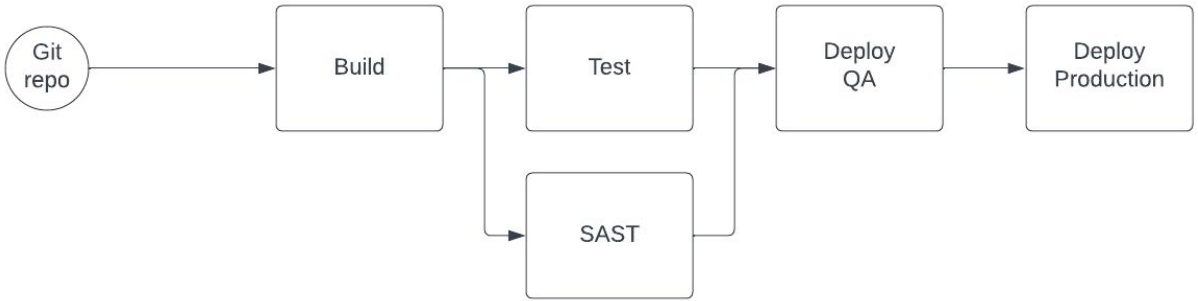


Application Security Testing

- SAST - Static Application Security Testing
- SCA - Software Composition Analysis
- DAST - Dynamic Application Security Testing
- IAST - Interactive application security testing
- RASP - Run-time Application Security Protection

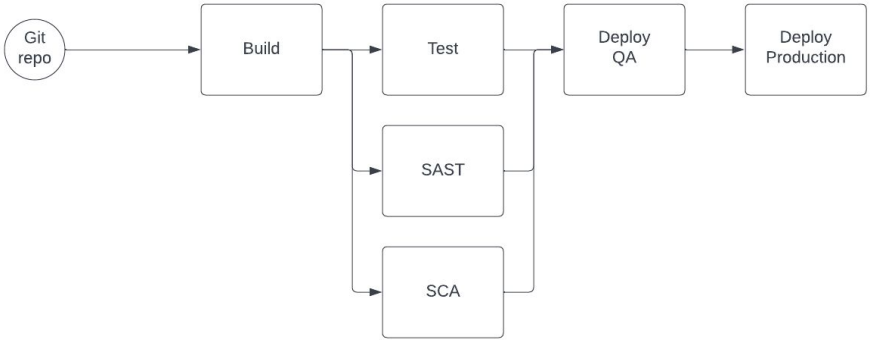
SAST

- White box
- Operates at the same level as the source code in order to detect vulnerabilities
- Static Code Analysis or SAST analysis is conducted before code compilation and without executing it
- Integrates with IDE



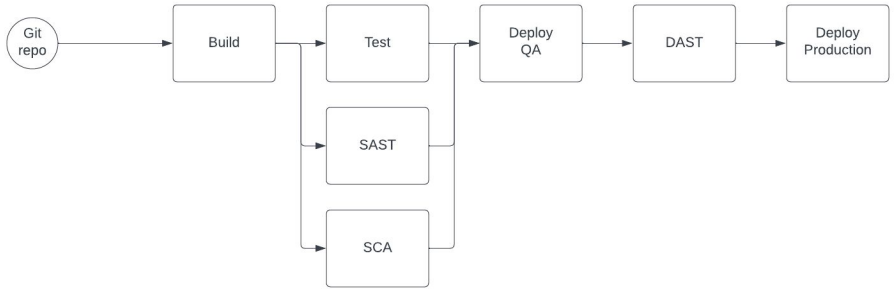
SCA

- Software Composition Analysis
- Quickly track and analyze any open-source and third-party components
- related components and supporting libraries
- direct and indirect dependencies
- detect software licenses, deprecated dependencies, as well as vulnerabilities and potential exploits



DAST

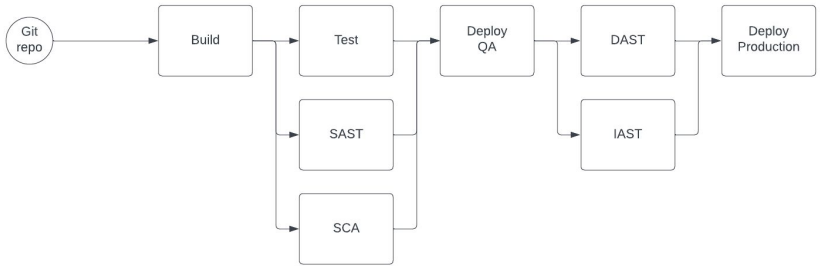
- Dynamic Application Security Testing
- Black box, performed from the outside in
- Security vulnerabilities and weaknesses in a running application
- Penetration testing
- Simulates attacks against an application



IAST

Several Clouds

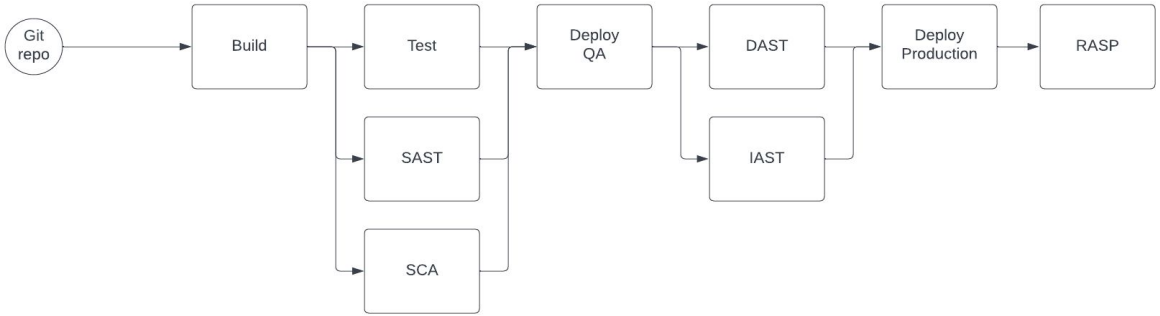
- Interactive application security testing
- Grey Box approach
- Operates inside the application
- IAST agent is working inside the app
- apply its analysis to the entire app
- Observes operation or attacks and identifies vulnerabilities
- Ability to identify third-party and open source components, known vulnerabilities, license types, and other potential risk issues



RASP

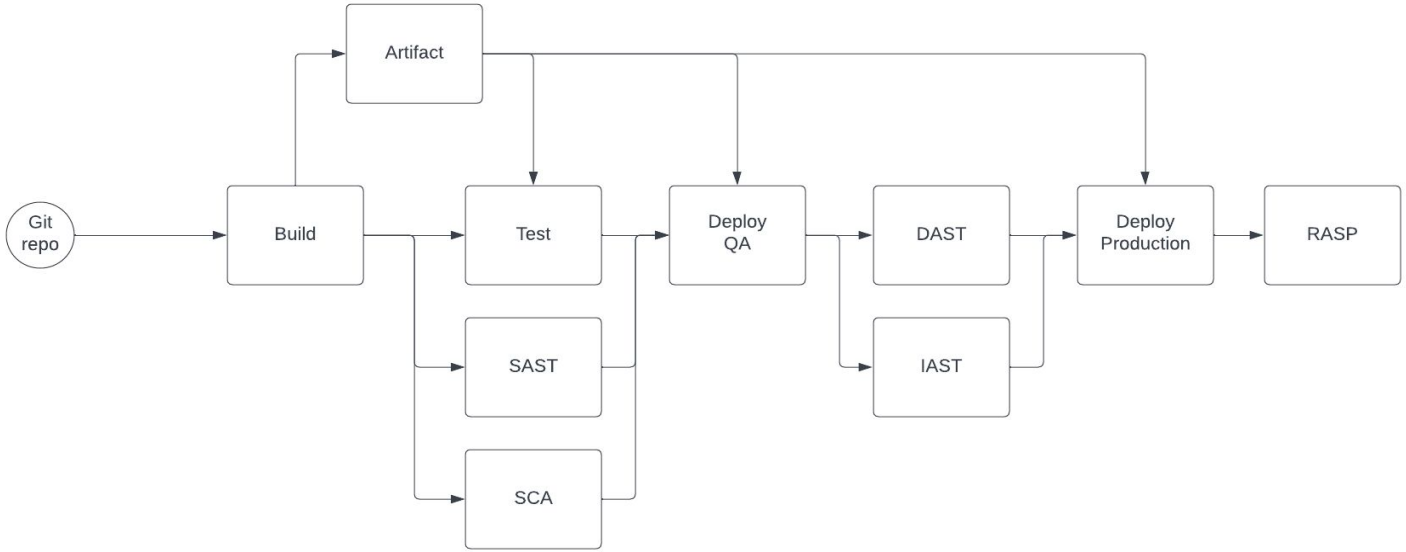
Several Clouds

- Run-time Application Security Protection
- Works inside the application
- Less a testing tool and more a security tool compared to IAST
- Can control application execution
- Run continuous security checks on itself and respond to live attacks by terminating an attacker’s session and alerting defenders to the attack



Scan the artifacts

- Periodic scans on the artifacts



Fast feedback loop

SAST tools integrate with IDE

- Install SonarLint in your IDE

Check for hard coded secrets

- gitleaks; git-secrets; Trivy

Several Clouds

OWASP

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software.

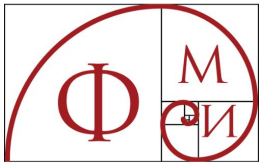
Several Clouds

Several Clouds

Continuous Delivery

Prepared for

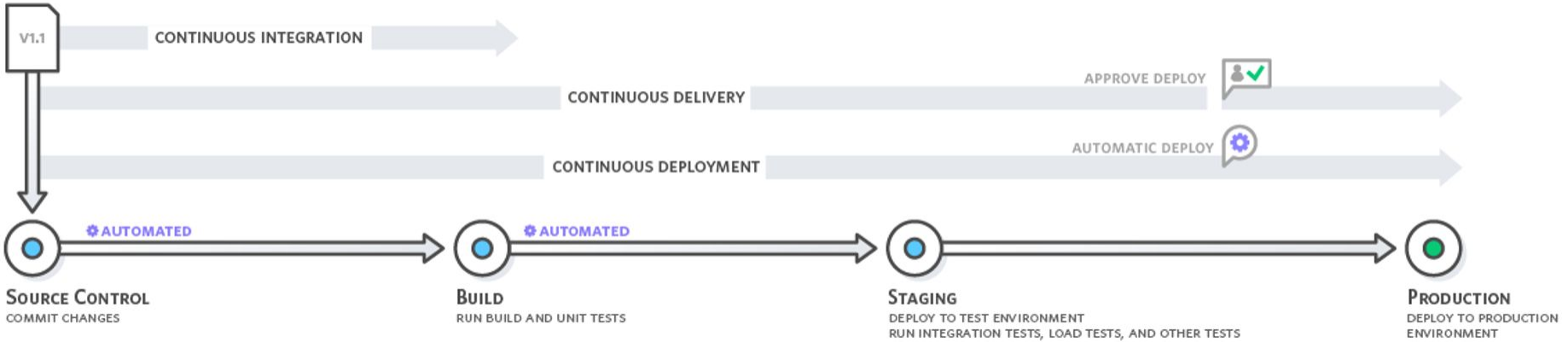
Faculty of Mathematics and Informatics (FMI)



Continuous Deployment

Several Clouds

“Between the time a developer commits code to the repository and the time it runs in production, code is a pure liability.”



Continuous Delivery practices

Several Clouds

- deployment automation
- continuous integration
- use of version control for all production artifacts
- monitoring and observability
- continuous testing
- integrating data and the database into the deployment pipeline
- integrating security into software delivery work

Releasing software should be easy

Several Clouds

- It should be easy because you have tested every single part of the release process hundreds of times before.
- It should be as simple as pressing a button.
- The repeatability and reliability derive from two principles:
  - automate almost everything
  - keep everything you need to build, deploy, test, and release your application in version control

Automate almost everything

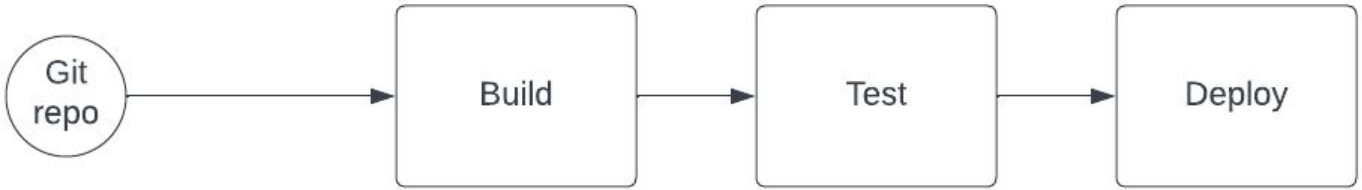
Several Clouds

- Build and Deployment Automation
- Automate Data Migration
- Automate Monitoring and Reporting
- Infrastructure Automation

- Process to roll back to a previous version of production if the deployment goes wrong

The only route to Production

- Run every emergency fix through your standard deployment pipeline
- This is just one more reason to keep your cycle time low



Application deployment and testing strategies

https://cloud.google.com/architecture/application-deployment-and-testing-strategies

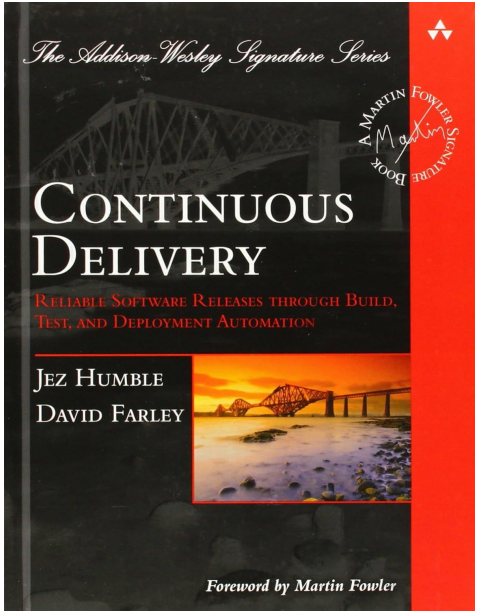
Several Clouds

Resources

Several Clouds

https://www.amazon.com/Continuous-Delivery-Deployment-Automation-Addison-Wesley/dp/0321601912

https://www.youtube.com/@ContinuousDelivery



Several Clouds

Thank you!

daniel@severalclouds.com
https://www.linkedin.com/in/danielrankov/