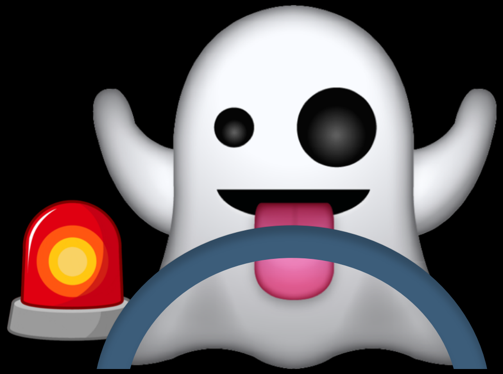


CI/CD @ WDW* @ Bosch

* Wrongway-Driver-Warning.

Introduction

Hai Dang Le

Techlead @ Wrongway-Driver-Warning @ Bosch

haidang.le@bosch.com

Agenda

What is Bosch’ Wrongway-Driver-Warning ?

Introduction: Gitlab

Deep-Dive: WDW Architecture

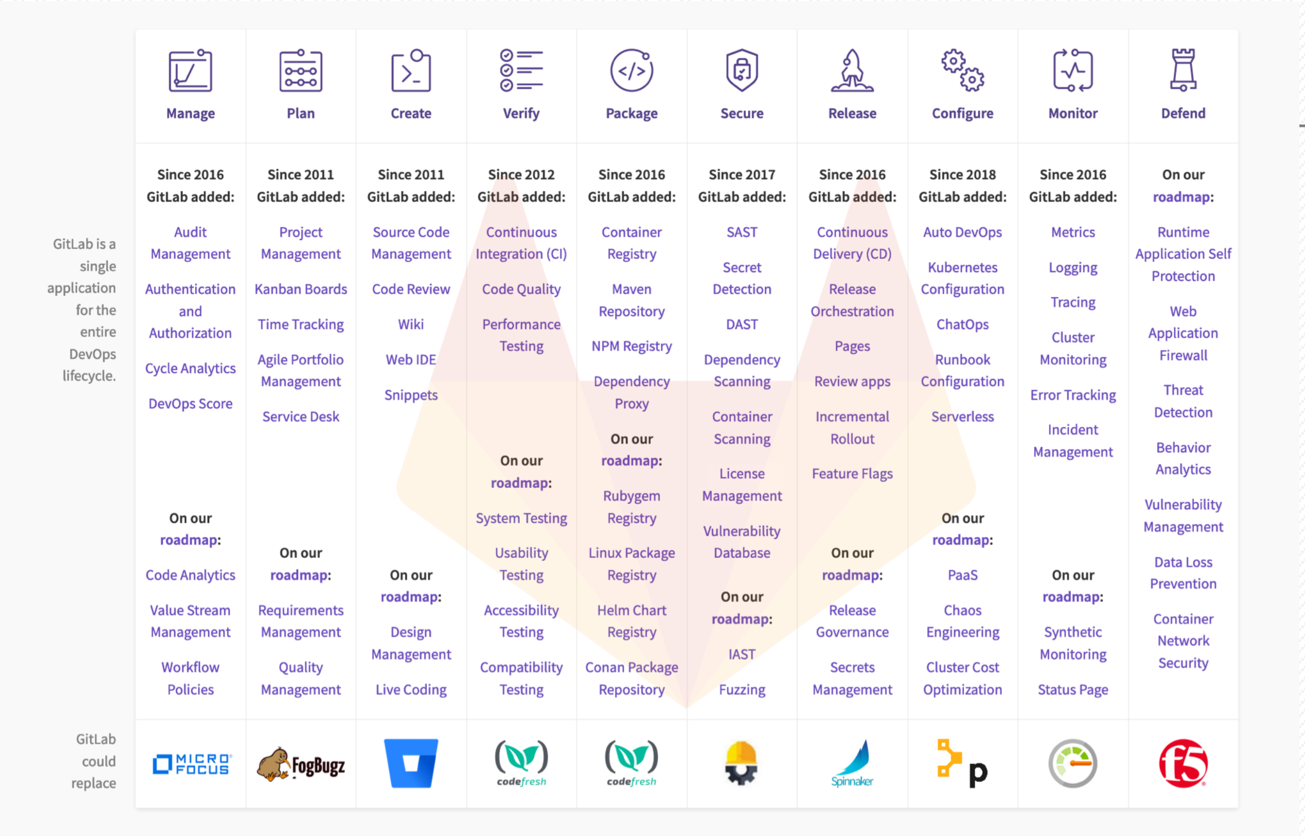
CI/CD: WDW CI/CD workflow

Gitlab | Intro

Single Webapp for Git-Repos, CI/CD, DevOps support
Wiki, Issue Tracker CI/CD Pipelines and more …
Products: CE & EE, On-Prem & SaaS
License: MIT (open-core model), EE with prop features: e.g.
Pricing: Free (2000 CI minutes/month), 4~99$ p.User/p.month

https://about.gitlab.com/

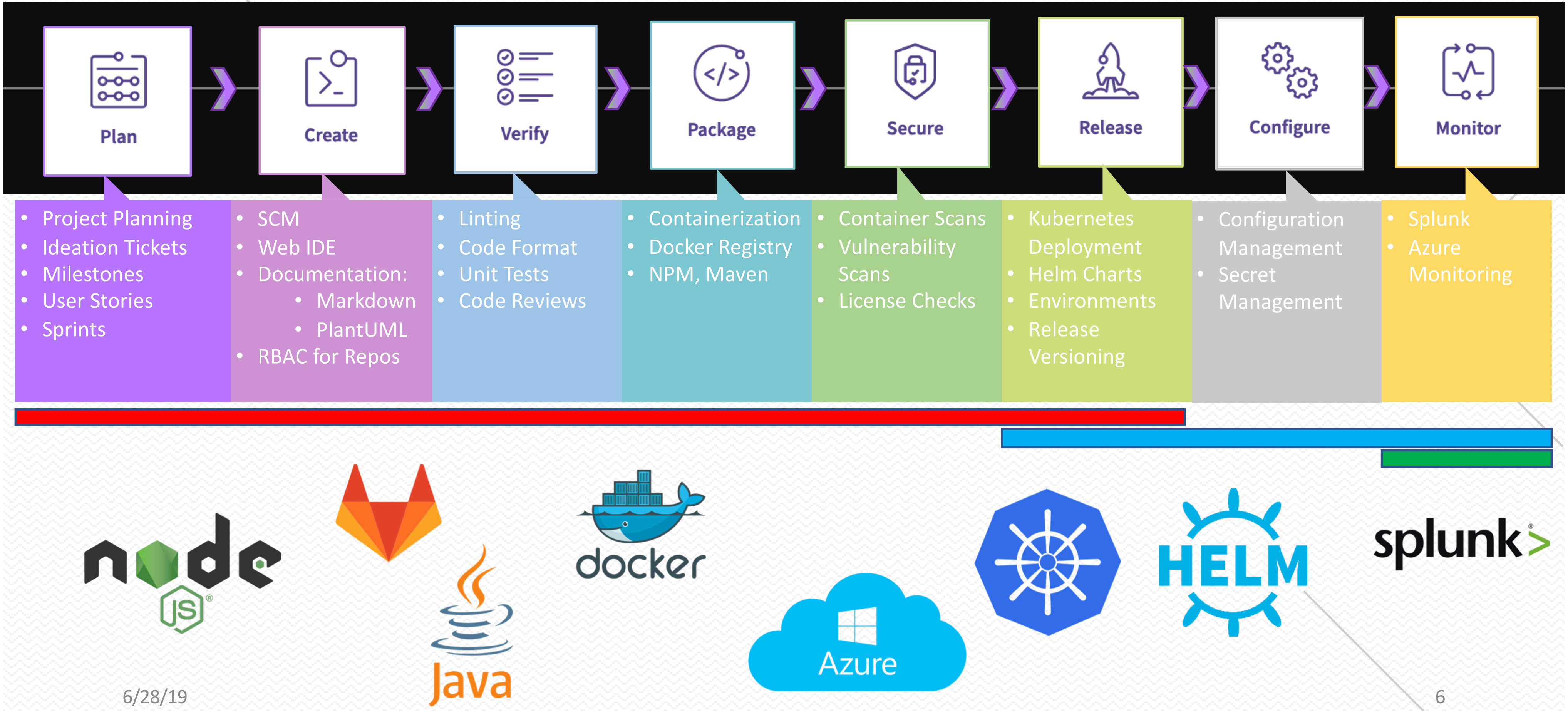
Gitlab Features



	Manage	Plan	Create	Verify	Package	Secure	Release	Configure	Monitor	Defend
	Since 2016 GitLab added:	Since 2011 GitLab added:	Since 2011 GitLab added:	Since 2012 GitLab added:	Since 2016 GitLab added:	Since 2017 GitLab added:	Since 2016 GitLab added:	Since 2018 GitLab added:	Since 2016 GitLab added:	On our roadmap:
GitLab is a single application for the entire DevOps lifecycle.	Audit Management Authentication and Authorization Cycle Analytics DevOps Score	Project Management Kanban Boards Time Tracking Agile Portfolio Management Service Desk	Source Code Management Code Review Wiki Web IDE Snippets	Continuous Integration (CI) Code Quality Performance Testing	Container Registry Maven Repository NPM Registry Dependency Proxy	SAST Secret Detection DAST Dependency Scanning Container Scanning License Management Vulnerability Database	Continuous Delivery (CD) Release Orchestration Pages Review apps Incremental Rollout Feature Flags	Auto DevOps Kubernetes Configuration ChatOps Runbook Configuration Serverless	Metrics Logging Tracing Cluster Monitoring Error Tracking Incident Management	Runtime Application Self Protection Web Application Firewall Threat Detection Behavior Analytics Vulnerability Management Data Loss Prevention Container Network Security
	On our roadmap:	On our roadmap:	On our roadmap:	On our roadmap:	On our roadmap:	On our roadmap:	On our roadmap:	On our roadmap:	On our roadmap:	
	Code Analytics Value Stream Management Workflow Policies	Requirements Management Quality Management	Design Management Live Coding	System Testing Usability Testing Accessibility Testing Compatibility Testing	Rubygem Registry Linux Package Registry Helm Chart Registry Conan Package Repository	IAST Fuzzing	Release Governance Secrets Management	PaaS Chaos Engineering Cluster Cost Optimization	Synthetic Monitoring Status Page	
GitLab could replace	MICRO FOCUS	FogBugz		codefresh	codefresh		Spinnaker	p		

Quelle: https://about.gitlab.com

WDW Tool-Chain



Plan

- Project Planning
- Ideation Tickets
- Milestones
- User Stories
- Sprints

Create

- SCM
- Web IDE
- Documentation:
  - Markdown
  - PlantUML
- RBAC for Repos

Verify

- Linting
- Code Format
- Unit Tests
- Code Reviews

Package

- Containerization
- Docker Registry
- NPM, Maven

Secure

- Container Scans
- Vulnerability Scans
- License Checks

Release

- Kubernetes Deployment
- Helm Charts
- Environments
- Release Versioning

Configure

- Configuration Management
- Secret Management

Monitor

- Splunk
- Azure Monitoring

Gitlab Features: Project Management

- Milestone- & Issue-Boards
- Wiki
- Workspaces
- Permissions, RBAC
- https://docs.gitlab.com/ee/user/project/

Gitlab Features: Collaboration

- Web IDE
  - Commit online
  - Resolve conflicts
  - Mark Down support
- Merge Requests
  - Inline Comments & Discussions
  - Multipe Approvers
  - Squash & Merge
- https://docs.gitlab.com/ee/user/project/web_ide/

Gitlab Features: CI/CD

- Pipelines
  - Pipelines via File
  - Visualization, History
  - Triggering pipelines via commit, Web UI, schedules, Webhooks, HTTP API
  - Docker integration
- Environments
  - Protected ENV variables
  - Stage-scoped ENV variables
- Kubernetes Cluster Integration
  - Deploy to Kubernetes
- https://docs.gitlab.com/ee/user/project/web_ide/

Gitlab Features: License Management

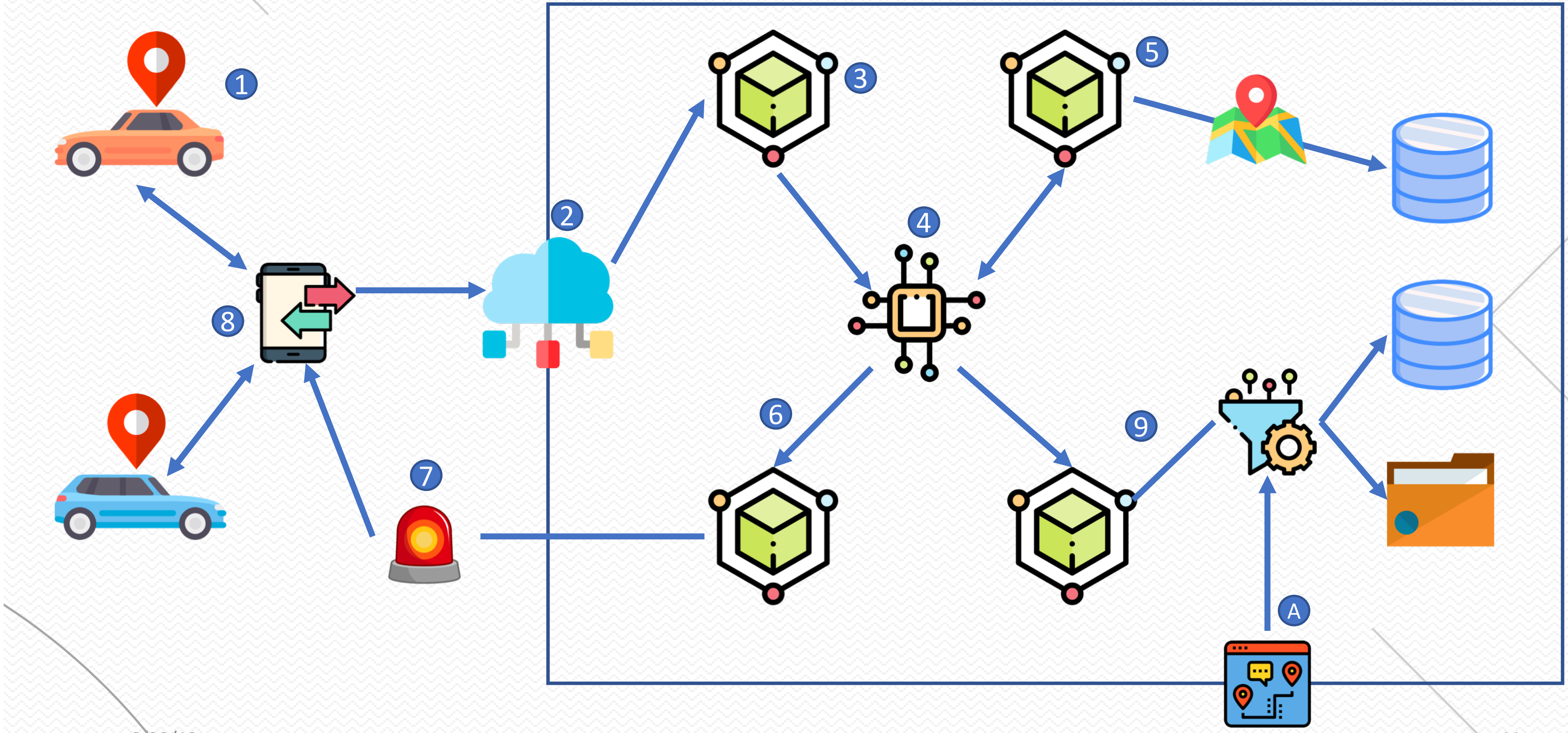
WDW: License Checker for NodeJS

- Checks for License Files/Notes/Headers
- Supports Whitelisting/Blacklisting of Licenses and Modules (+Versions)

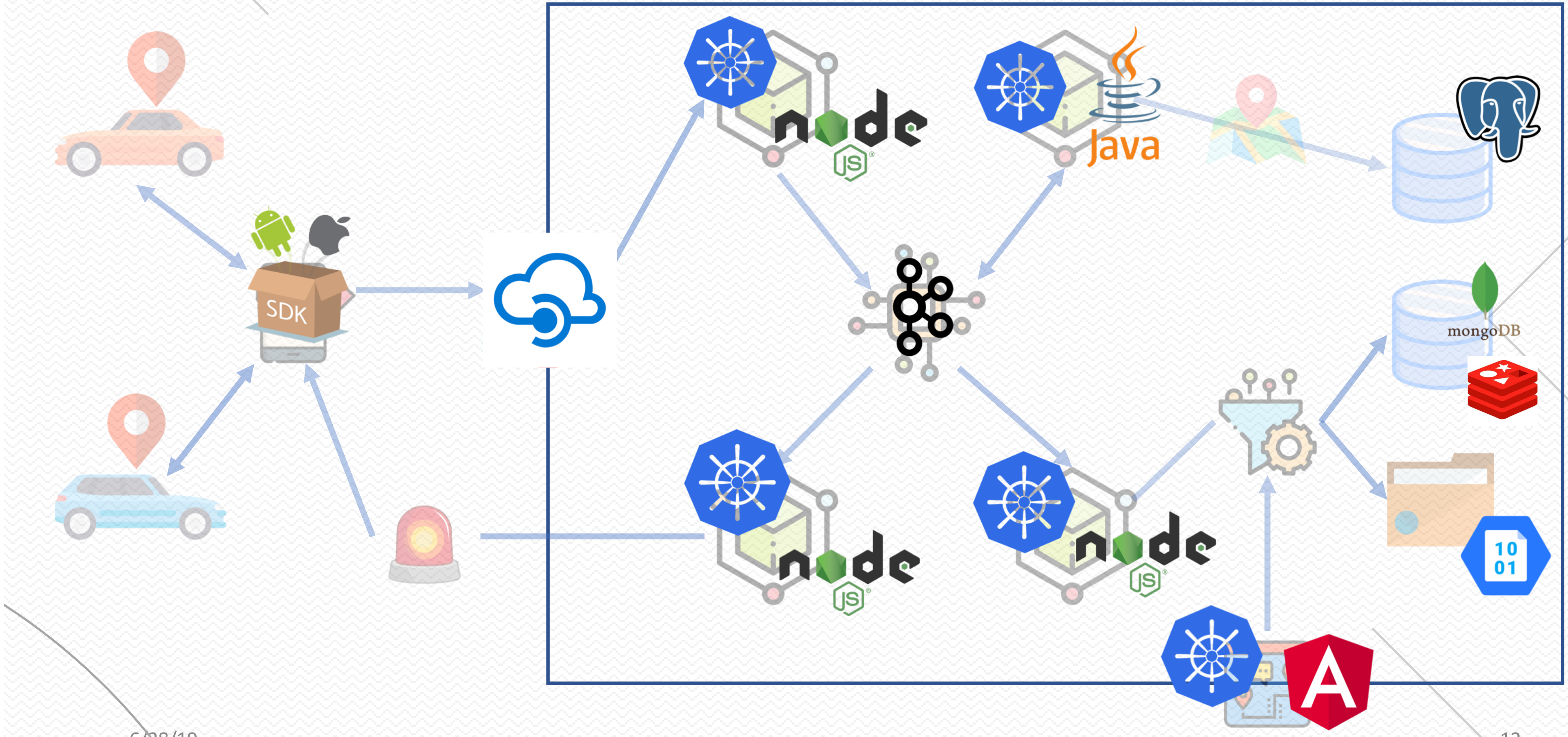
Gitlab License Management

- Java, JS, Go, Ruby, Python, .Net
- Scans Dependencies for Licenses
- Scan Reports in Merge Requests
- Supports Whitelisting/Blacklisting of Licenses
- https://docs.gitlab.com/ee/user/application_security/license_management/

WDW Architecture



WDW Architecture



6/28/19

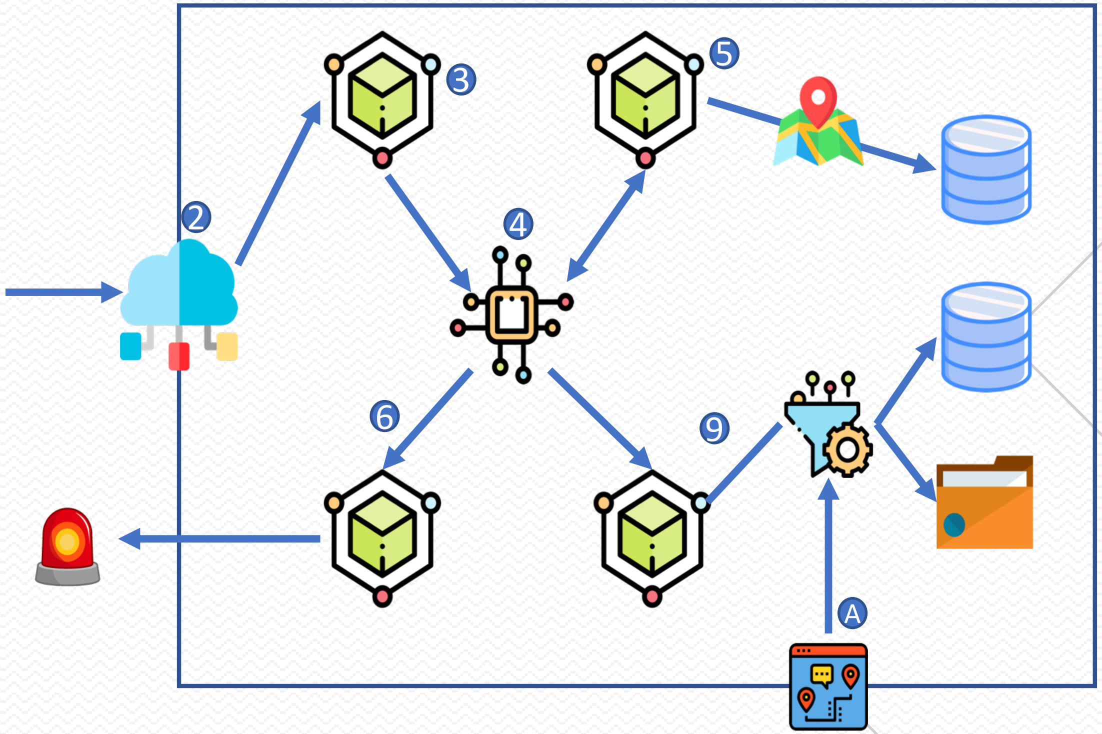
WDW Architecture

Demo
SDK
java
node
mongoDB

WDW CI/CD

CI / CD for Microservices is hard …

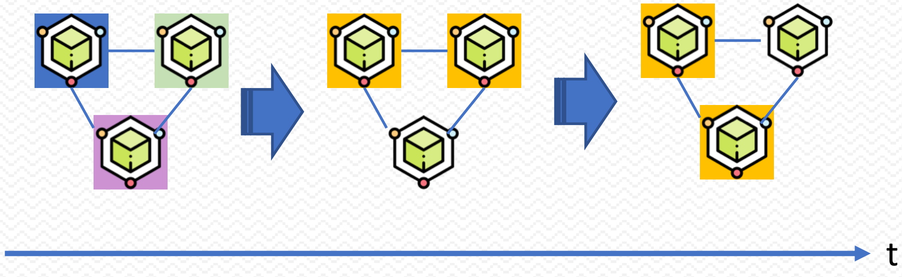
- Different development & release cycles of µServices
- System Release is distributed to µService
- System-Testing is difficult to coordinate
- Change / Version tracking



WDW CI/CD

Requirements:

- Traceability of features:
  - version/build -> commit -> user story -> requirements
  - Changes over time
- Single Point of Truth:
  - all deployed microservices are listed in one location
  - Version/build
- System Tests first:
  - System Tests are more important than unit-, component-, smoke-tests



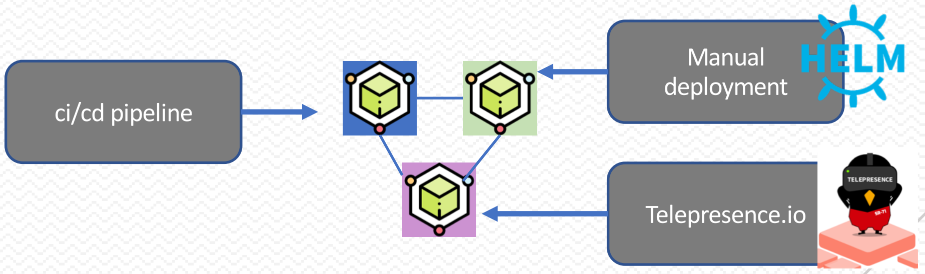
Philosophy:

- Iterative
  - Small iterative changes
- Continous
  - Changes are deployed with every commit to Dev
- Complete System Release
  - frequent system releases (days, < 1 week)
  - continous releases bundles
- Fast-Forward
  - No way back: new features must support backward compability
  - Errors must be fixed asap, must not break system release

WDW Stages

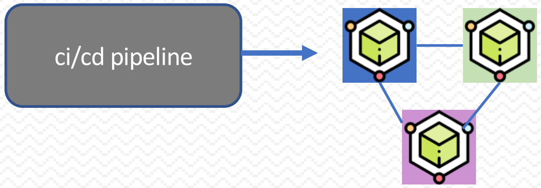
‚D‘ev:

- Current state of development (autom. deployment of master-commits)
- Developer test, smoke test
- Logging & Monitoring



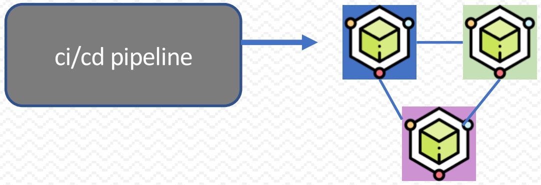
‚S‘taging:

- Integration test / system test
- Performance test
- Customer onboarding
- Logging & Monitoring & Alerting

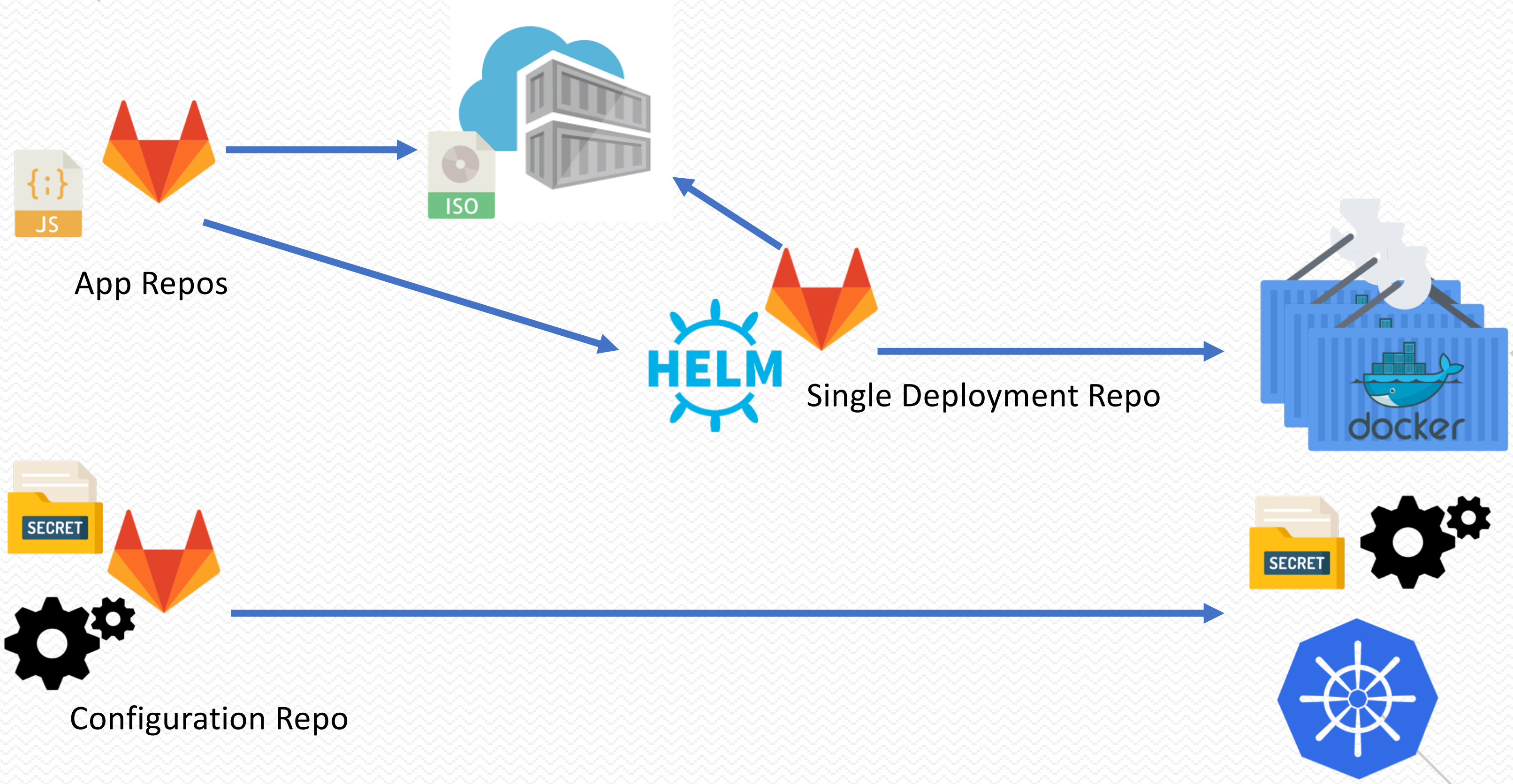


‚P‘rod:

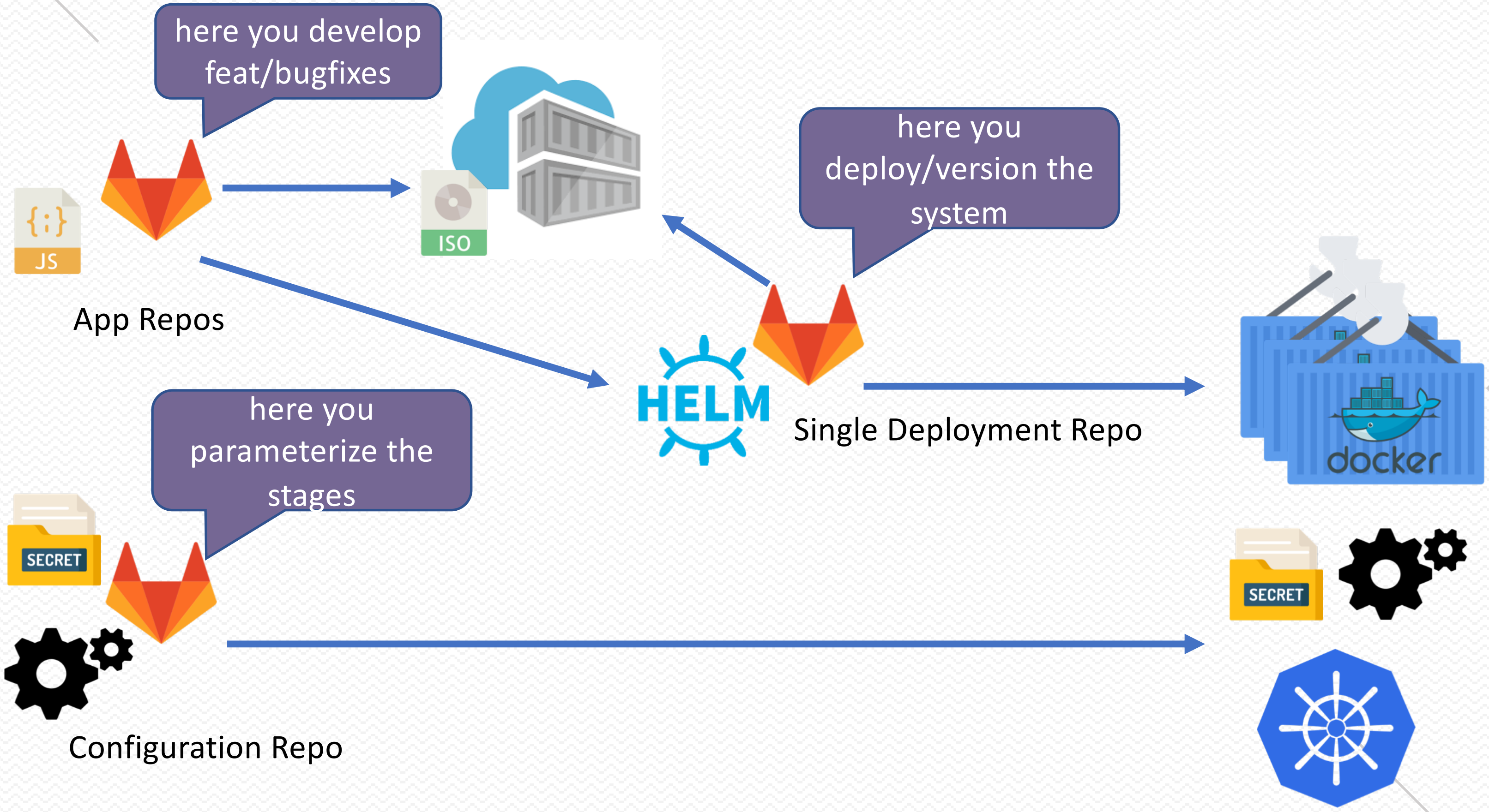
- ‚Live‘-Customer
- Logging & Monitoring & Alerting



WDW Deployment



WDW Deployment

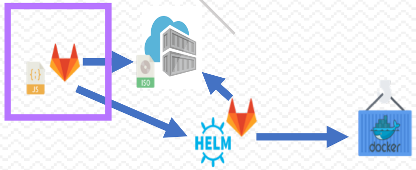


here you develop feat/bugfixes

here you deploy/version the system

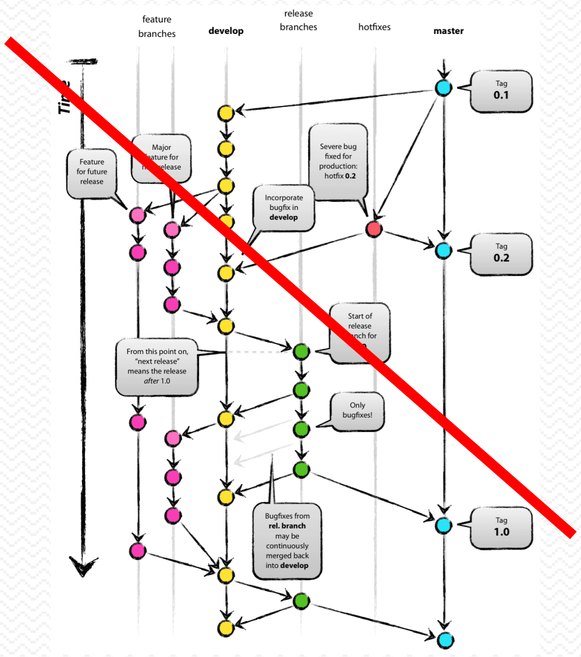
here you parameterize the stages

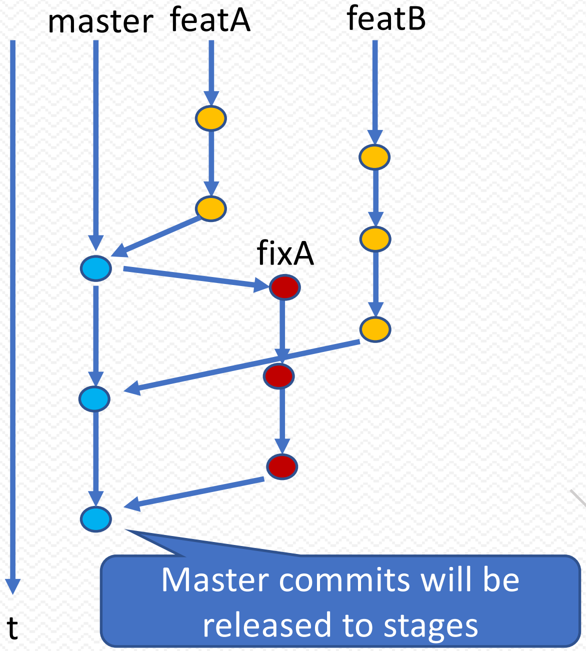
WDW flow



App Repo:

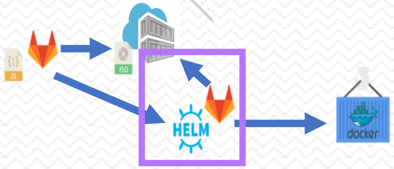
- 1 master branch
- Feat. & Bugfix branches are merged directly to master
- No develop, no release branches


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Major feature for next release
Feature for future release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0



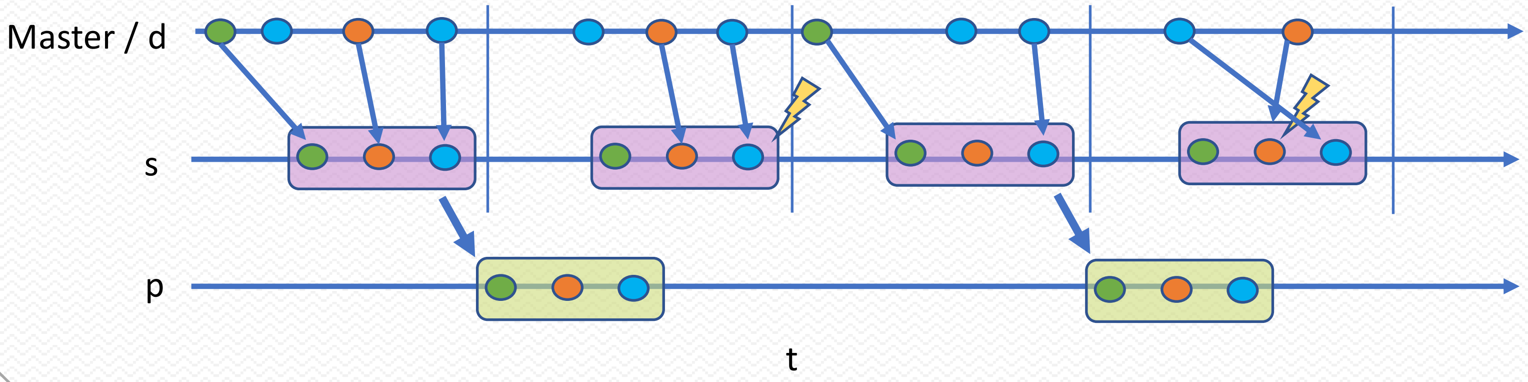
Master commits will be released to stages

WDW flow

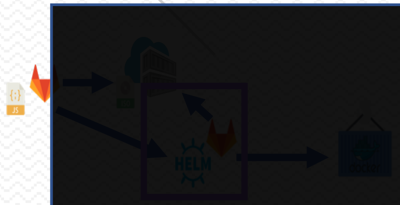


Deployment Repo:

- 1 master branch
- Deployment of apps via Helm
- Master deploys to “development“
- Tags with –“staging“ suffix deploy to „staging“
- Tags with –“prod“ suffix deploy to „prod“



WDW flow



Pros:

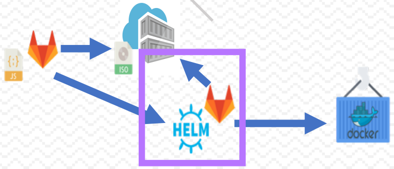
- Lean Git-workflow
- Efficient for testing
- Easy to track changes
- Deployments are versioned
- Easy to create Release Notes
- Manual Editing possible

Cons:

- Errors stop releases of features in respective App
  - Unless rebasing is done
- only support 1 „version“ at a time
  - Not suitable for SDK releases
  with multiple versions in the field

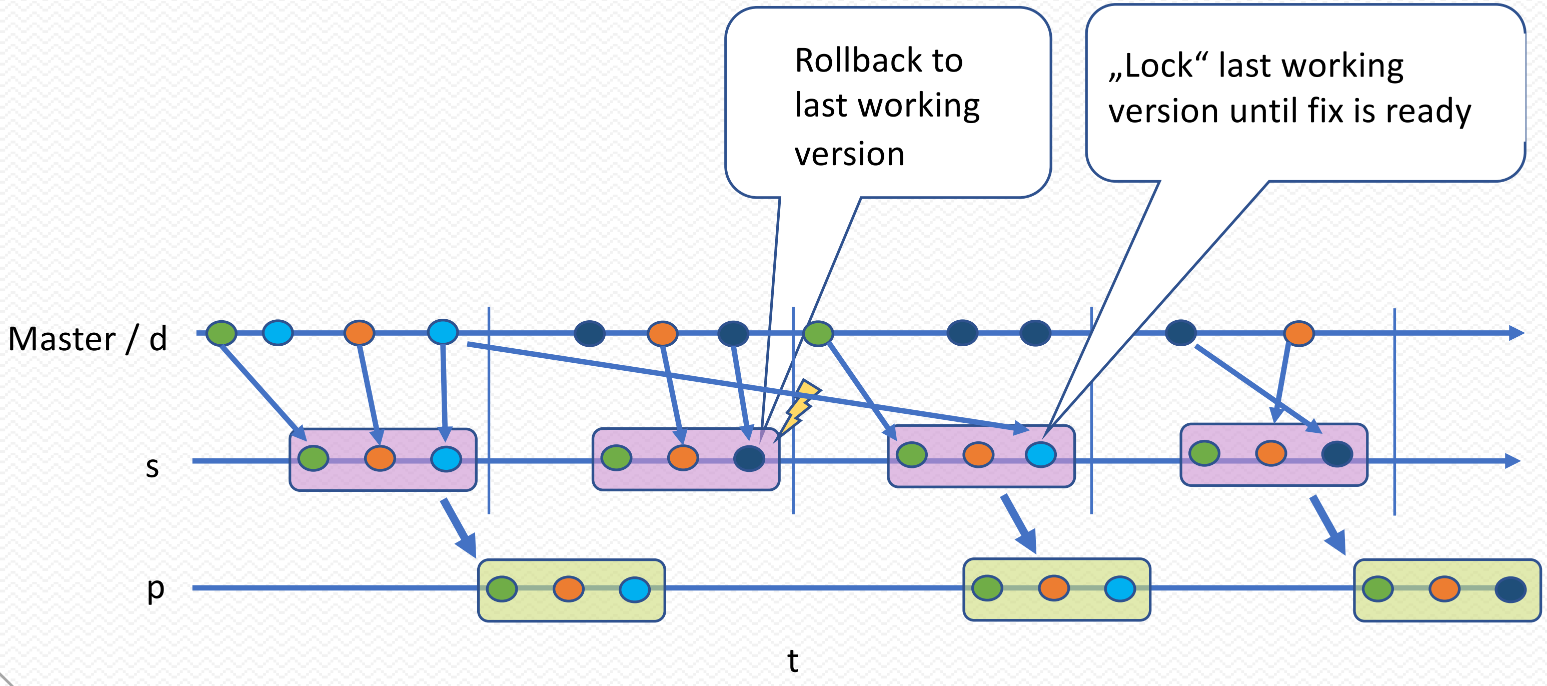
Demo

WDW flow



Premise:

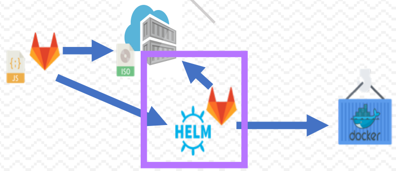
- In case of error, fix Error in App asap
- If no fix is available soon, „lock“ last working version
- If features have to be rolled out, do reset head & rebase



Rollback to last working version

„Lock“ last working version until fix is ready

WDW flow



Pros:

- Lean Git-workflow
- Efficient for testing
- Deployments are versioned
- Easy to compare releases
- Easy to track, what is deployed
- Manual Editing possible

Cons:

- Master only: complicated git-workflow when commits-have to be rolled back
- only support 1 „version“ at a time
  - Not suitable for SDK releases with multiple versions in the field

THANK YOU