



# Making Releases Routine

Maaret Pyhäjärvi



by Maaret Pyhäjärvi is licensed under CC BY 4.0

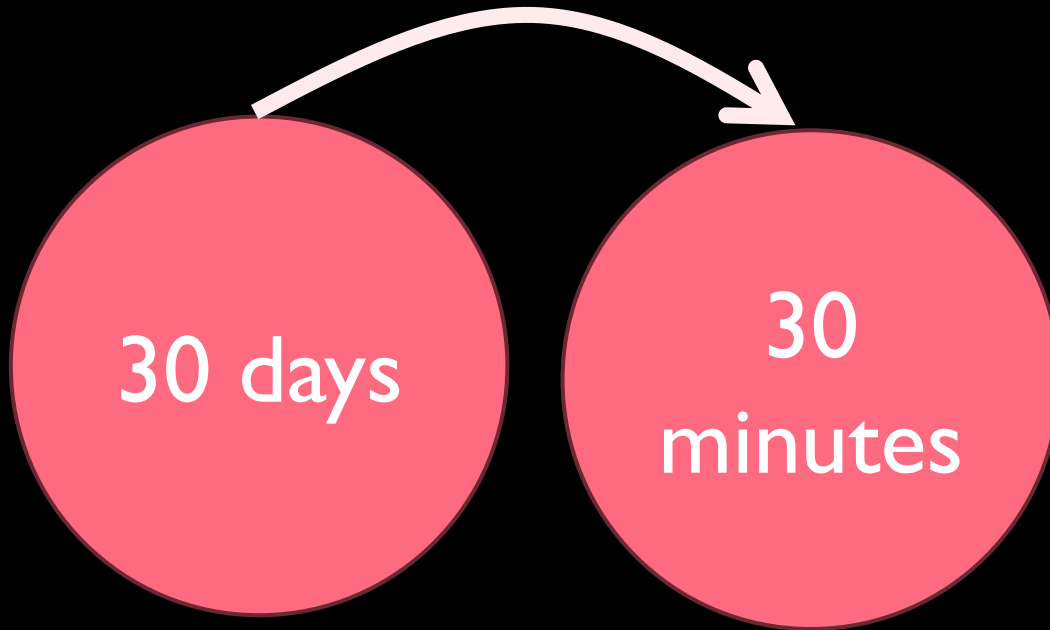


<https://www.linkedin.com/in/maaret/>



[@maaretp@mas.to](https://twitter.com/maaretp)

# 1<sup>st</sup> year: My Signature Move



*Pool is not a  
bigger bathtub.*

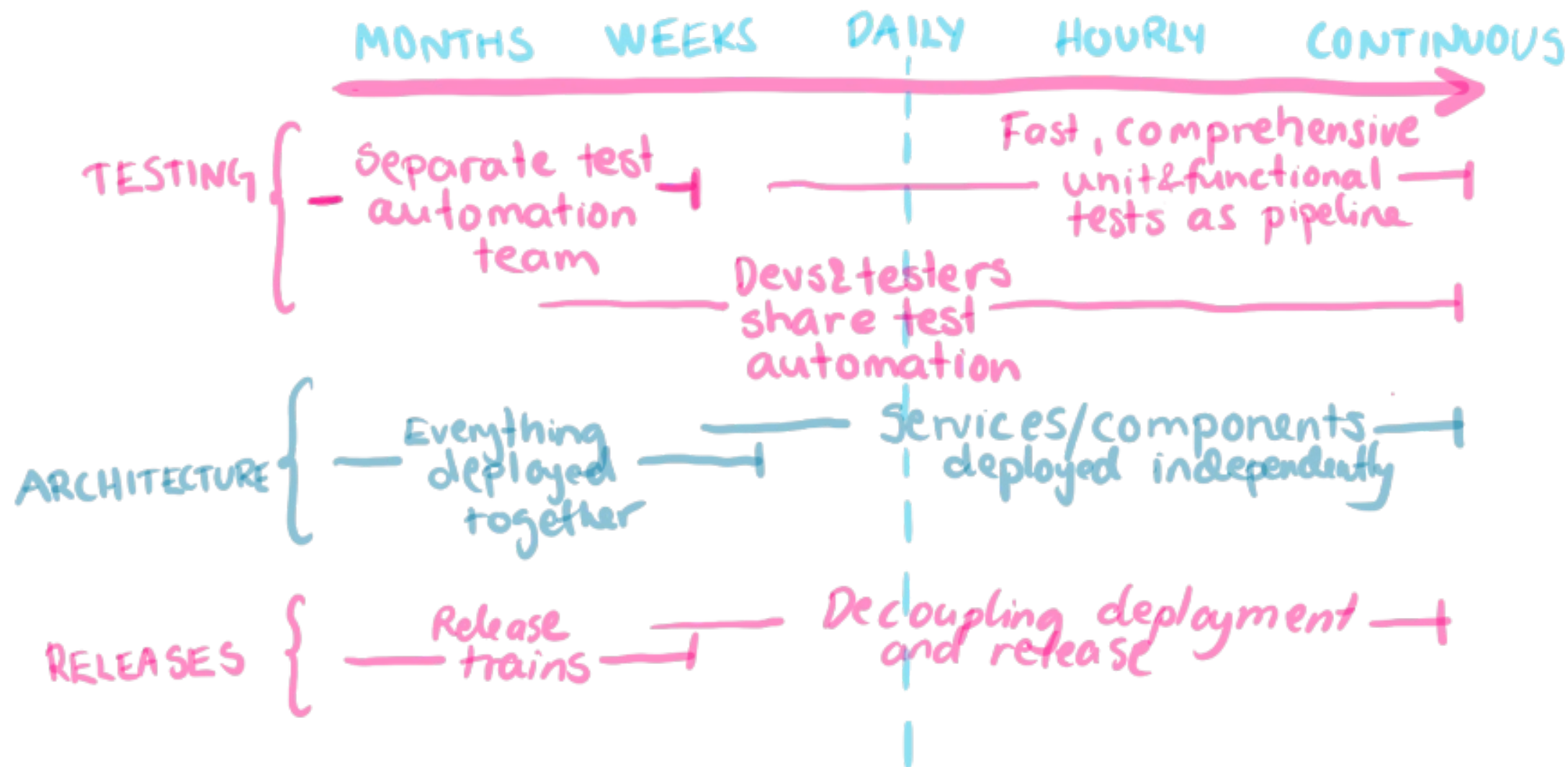


<https://www.linkedin.com/in/maaret/>

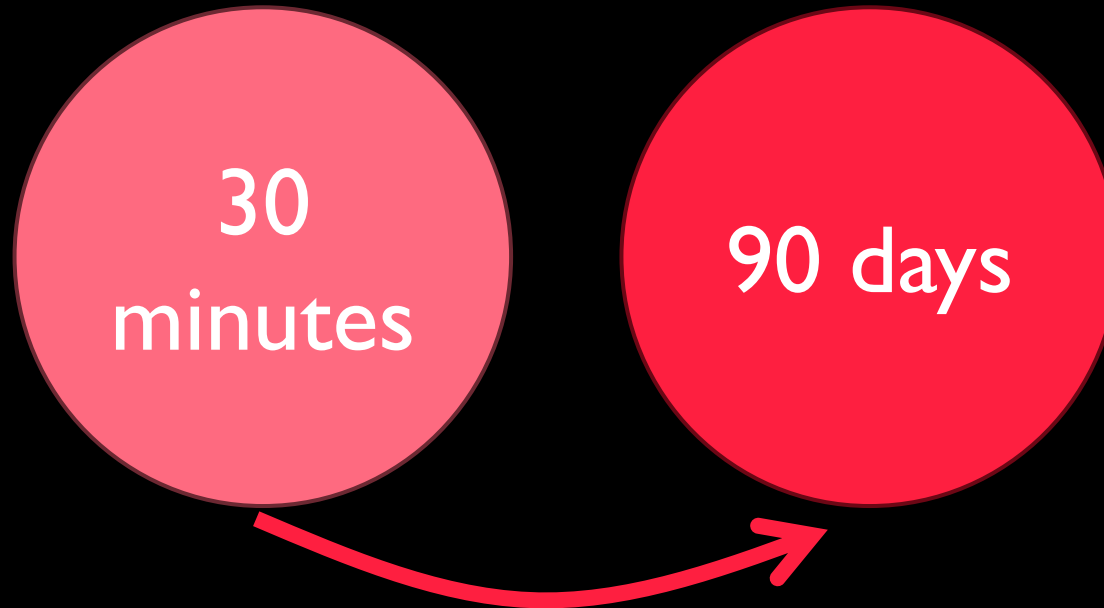


@maaretp@mas.to

# RELEASE FREQUENCY



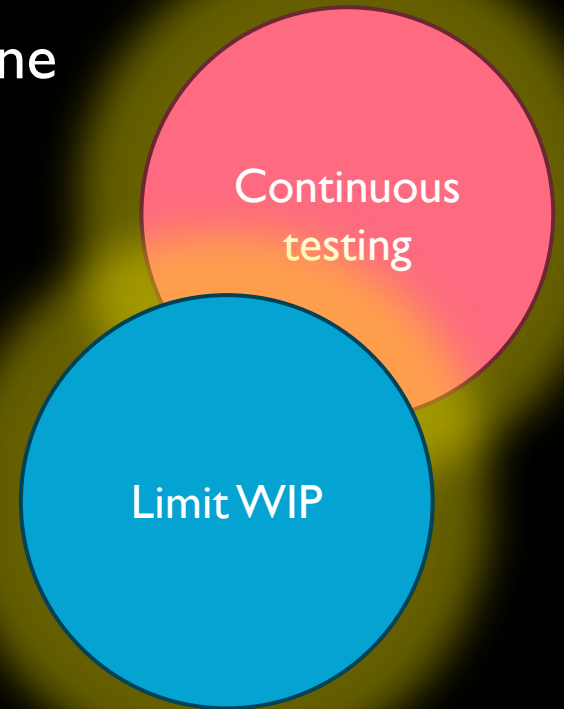
# 2<sup>nd</sup> year: Something Lost



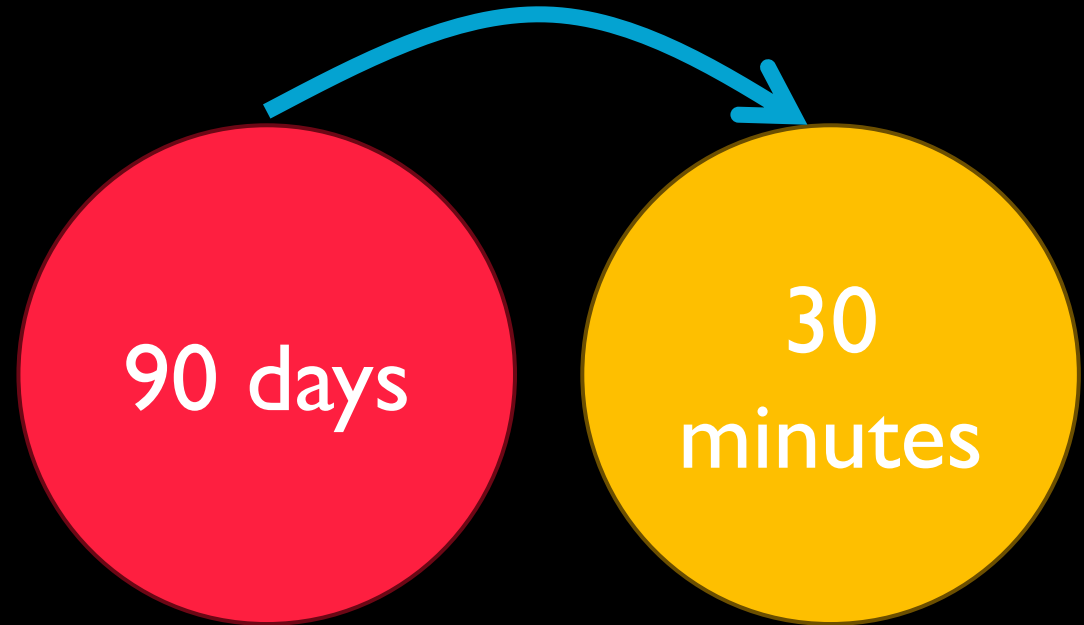
# The Failure

## Mistakes were made. And by me.

1. Making results gap visible is a testing skill. Just because someone tests, does not mean they test well.
  1. Recognizing scope
  2. Nudging scope with conversational approach
  3. Orchestrating fixes
  4. Feeling powerless
2. Premature done starts new work. New work piles up uncertainty.



# 3<sup>rd</sup> year: Make Releases Routine



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Describing Release and Grouping Tasks

## TASKS

**Write release notes** - 26 individual changes to message worth saying  
**Create release checklist** - while I know it by heart, others may find it useful to tick off what needs doing to say its done  
**Select / design title level tests** for test execution (evidence in addition to TA - test automation)  
**Split epics** to this release - other release so that epics reflect completed scope over aspirational scope. and can be closed for the release  
**Document per epic acceptance criteria**, esp. out of scope things - documentation is an output not input, but if I was testing, it was a daily output not something to catch up at release time  
**Add Jira tasks into epics to match changes** - this is totally unnecessary but I do that to keep a manager at bay, close them routinely since you already tested them at pull request stage  
**Link title level tests to epics** - again something normally done daily as testing progresses, but this time was left outside the daily routine  
**Verify traceability matrix** of epics ('requirements') to tests ('evidence') shows right status  
**Execute any tests in test execution** - optimally one we call release testing and would take 15 minutes on the staging environment  
**Open Source license check** - run license tool, compare to accepted OSS licenses and update licenses.txt to be compliant with attribution style licenses  
**Lock release version** - Select release commit hash and lock exact version with a pull request  
**Review Artifactory Xray statistics** for docker image licenses and vulnerabilities  
**Review TA (test automation) statistics** to see it's staying and growing  
**Press Release-button in Jira** so that issues get tagged - or work around reasons why you couldn't do just that  
**Run promotion that makes the release** and confirm the package  
**Install to staging environment** - this is something from 3 minute run a pipeline to 30 minutes do it like a customer does it  
**Announce the release** - letting others know is usually useful  
**Change version for next release** in configs

**GROUP**

## START OF NEW

Create release checklist  
Change version for next release

## CONTINUOUS SYSTEM TESTING

Select / design title level tests  
Split epics  
Document per epic acceptance criteria  
Add Jira tasks into epics to match changes  
Link title level tests to epics  
Verify traceability matrix  
Execute any tests in test execution

## SEPARATE CADENCE

Review Artifactory Xray statistics  
Review TA (test automation) statistics  
Architecture review  
Threat modeling

## DO

Write release notes

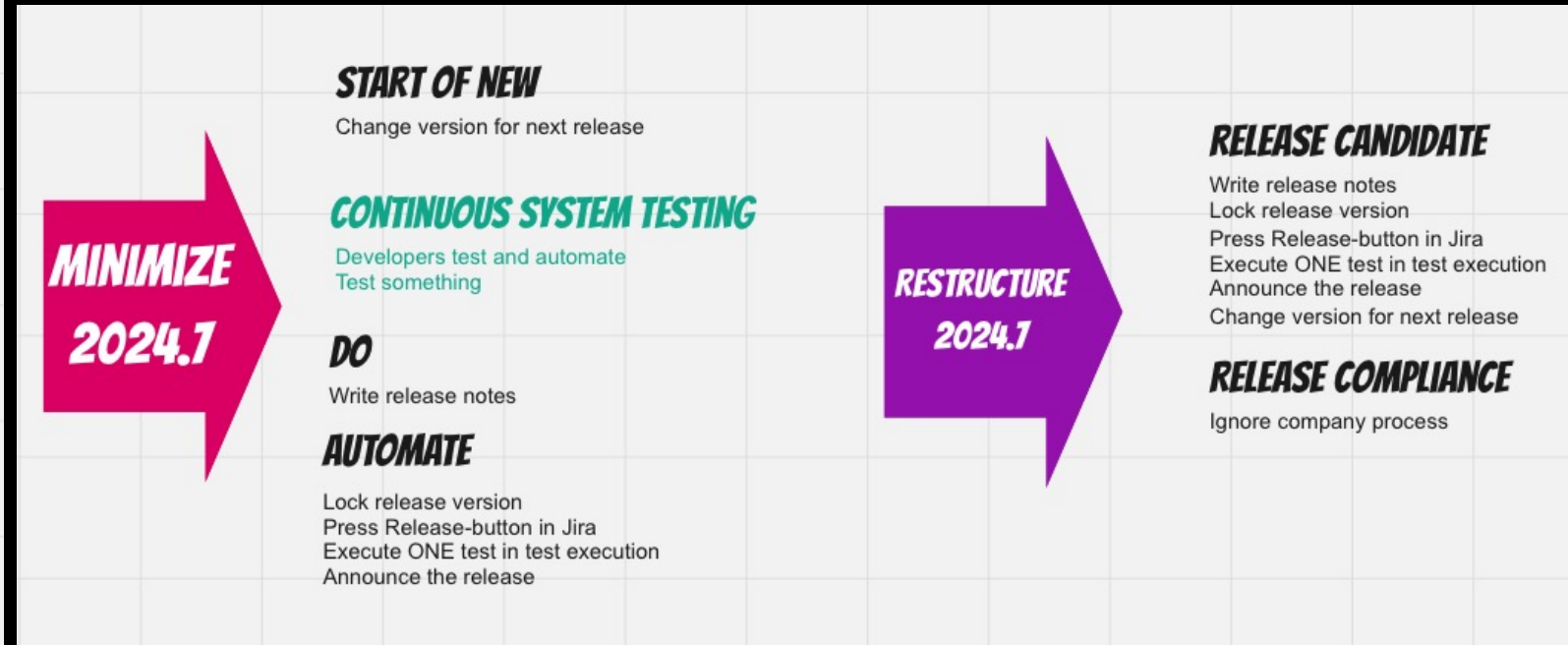
## AUTOMATE

Open Source license check  
Lock release version  
Press Release-button in Jira  
Run promotion that makes the release  
Install to staging environment  
Execute ONE test in test execution  
Announce the release





# Experimenting with Structure



Practice makes better!



<https://www.linkedin.com/in/maaret/>

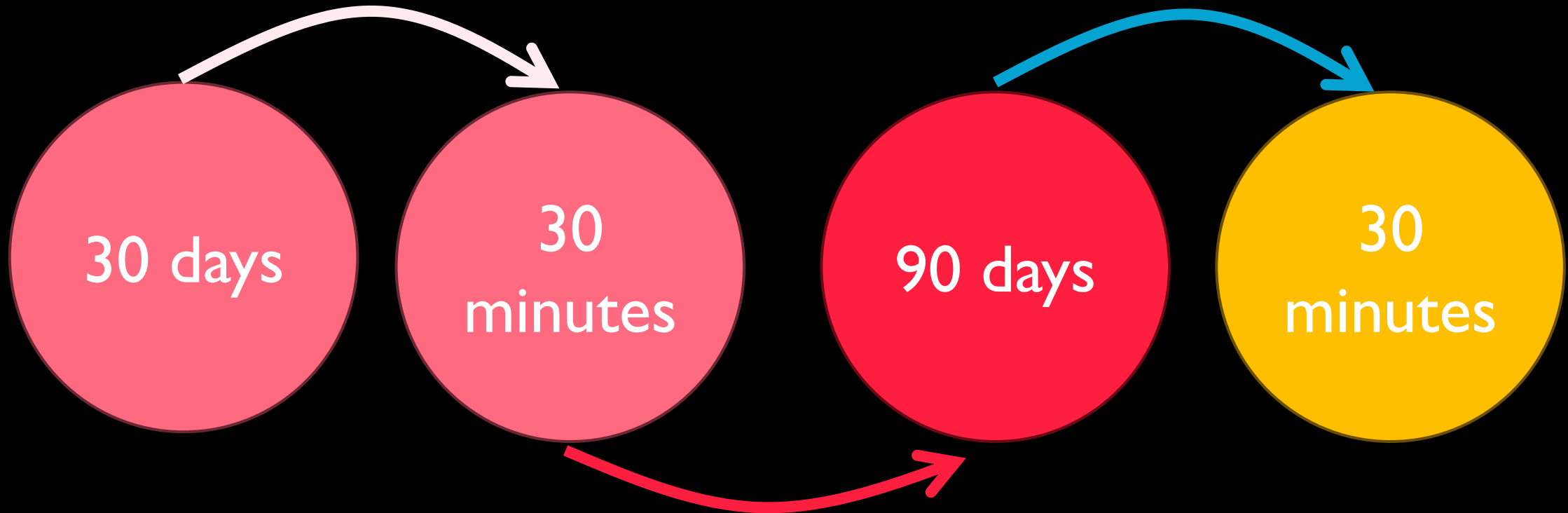


@maaretp@mas.to

# With All Test Green in CI Principle, Release Tagging Preserves Tests for Compliance

When the color is green, no one cares about how many pass and fail.

# Context Changes Over Time



“Releases are  
overhead.  
So much work.”



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# What is the Work?

Build and release pipeline

Evidence for compliance

Testing for all things in the results gap

Approvals and permissions



# Release Review

22 Required High Level Checks

3579  
compliance checks

Action	
Architecture and Designs (D1, D6)	Documentation
Technical Quality (Q2, Q3)	Linters Unit test coverage
System Test Contents (Q1, Q4, Q5, Q6, Q7, D1)	Master test plan, test cases, test summary Sufficient testing
Security (Q9)	Threat analysis Vulnerability management
IPR and Licenses (Q8, E1, E2)	OS, Middleware, Application inventory Forbidden licenses check
Product Perspectives (Q10, Q11, Q12, D3, D4)	Usability Documentation Localization
Stakeholder involvement (Q13)	
Archiving (D2, D5, D7)	Release repeatability Plans for next up Installation media

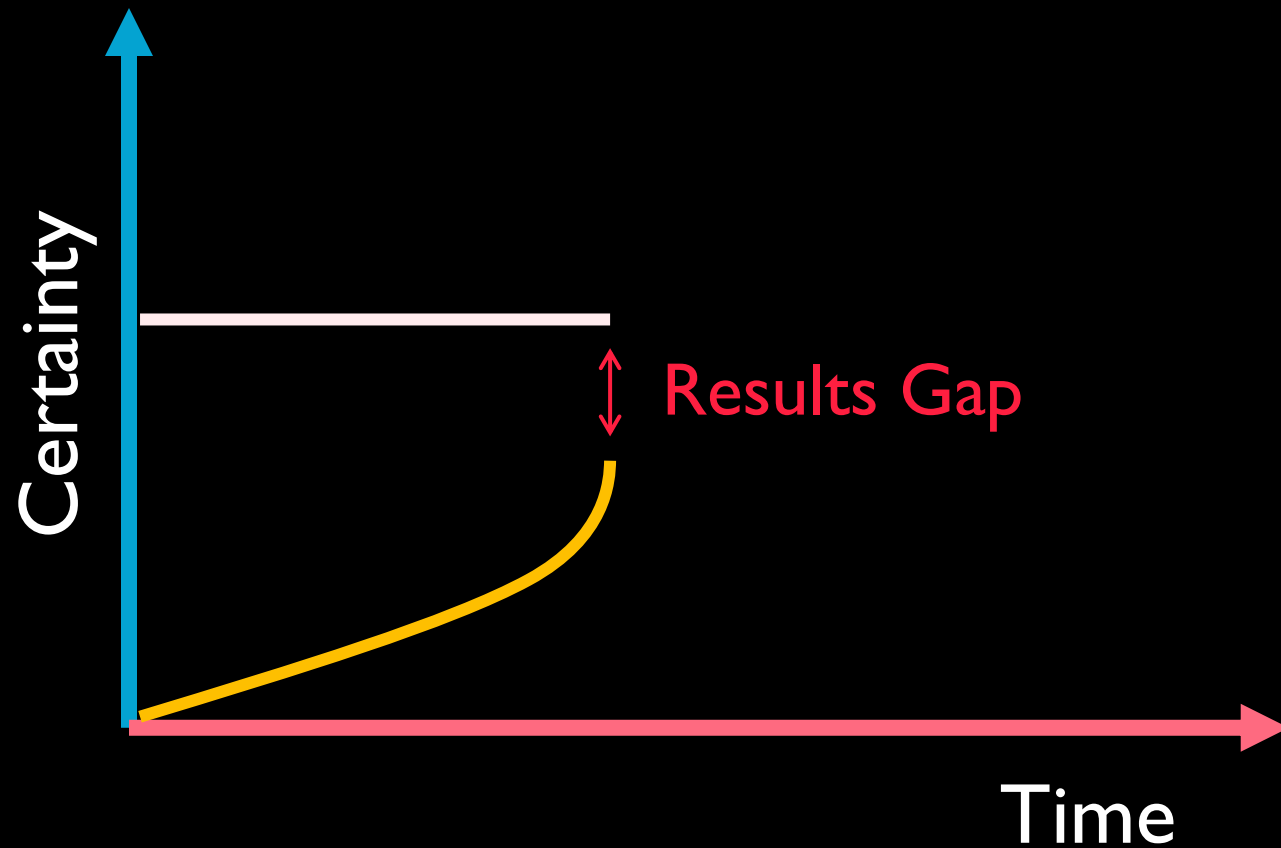


<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Certainty vs. Time



Delivering to  
environment without  
use does not add  
certainty



What must be different  
to shorten release testing  
from 30 days to 30  
minutes?



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to



# Fact Checking

There is no **automation** that gets a month of work done in 30 minutes. Just **no**.



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Fact Checking

We do less in release testing.  
We do more **before** release testing.



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Fact Checking

We **don't have bugs**. **Fixing** delays and causes retesting.

We don't wait for testing. Testing can **continue after**.



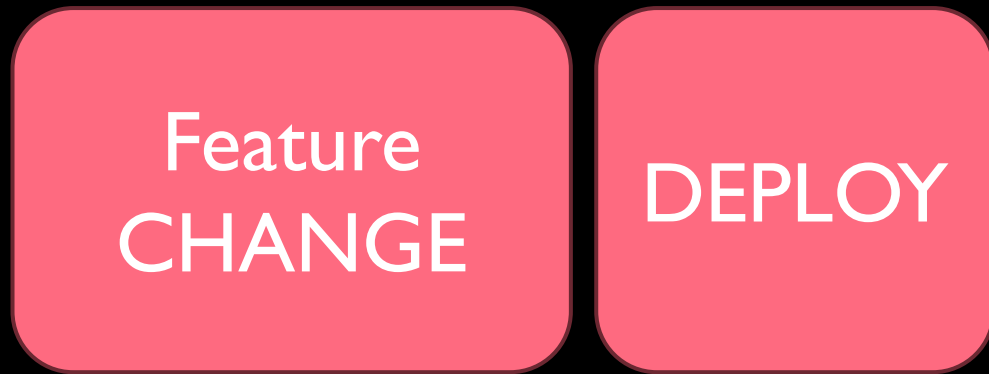
<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Features, Deployments, Releases

Feature flag



System testing = Feature testing +

Release testing



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Guidelines

**EaC.** Nothing changes without changing something. **Follow the changes.**

\* EaC – Everything as Code, particularly Infra



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Guidelines

Continuous testing. Test automation alerts on changes. Change calls to explore.



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

# Guidelines

**Pipelines.** Doing the work all the time, with every change. **Routine comes with repetition.**



<https://www.linkedin.com/in/maaret/>



@maaretp@mas.to

Making releases routine is the heartbeat of a good team, creating a bubble of productive serenity.

—Maaret Pyhäjärvi