DevOops

[R]Evolution of open source CI/CD Tools

by Oleg Nenashev
Dynatrace/CDF/Jenkins

Nov 08, 2021

> whoami

@oleg_nenashev

oleg-nenashev

Agenda

- Why do **WE** need CI/CD?
- Generations of CI/CD tools
- What's now and what's next?
- What can you do after the talk?

Ask me later:

- How do we adjust as a company?
- Jenkins roadmap

These slides:

@oleg_nenashev

bit.ly/devoops2021-cicd

Discussion

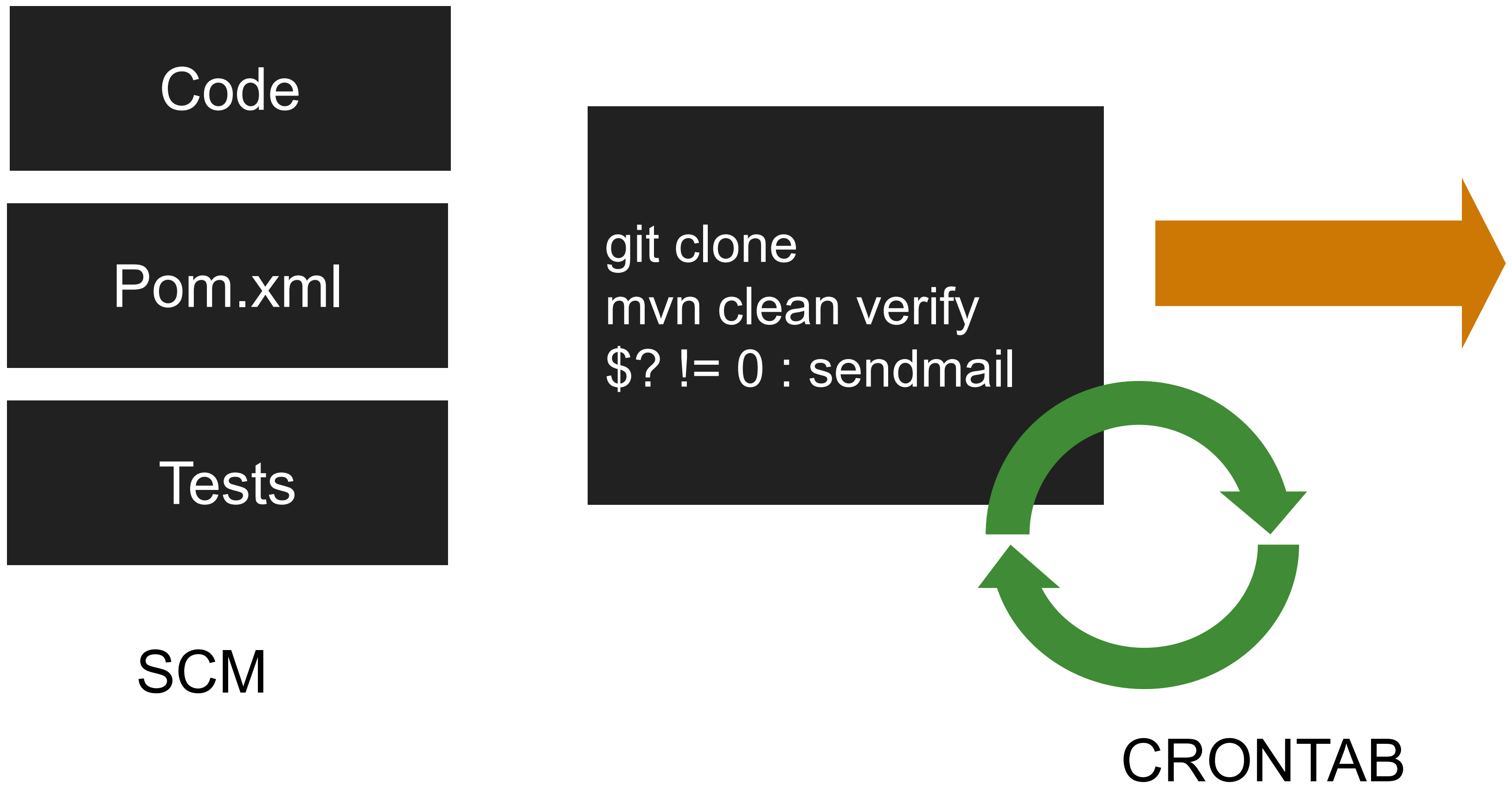
community.jenkins.io/t/new-talk-evolution-of-open-source-ci-cd-tools/207

Disclaimer

- Opinions are my own, happy to discuss
- My talk does not represent opinions of my employer, the Continuous Delivery Foundation, or the Jenkins community
- I present only open source projects today

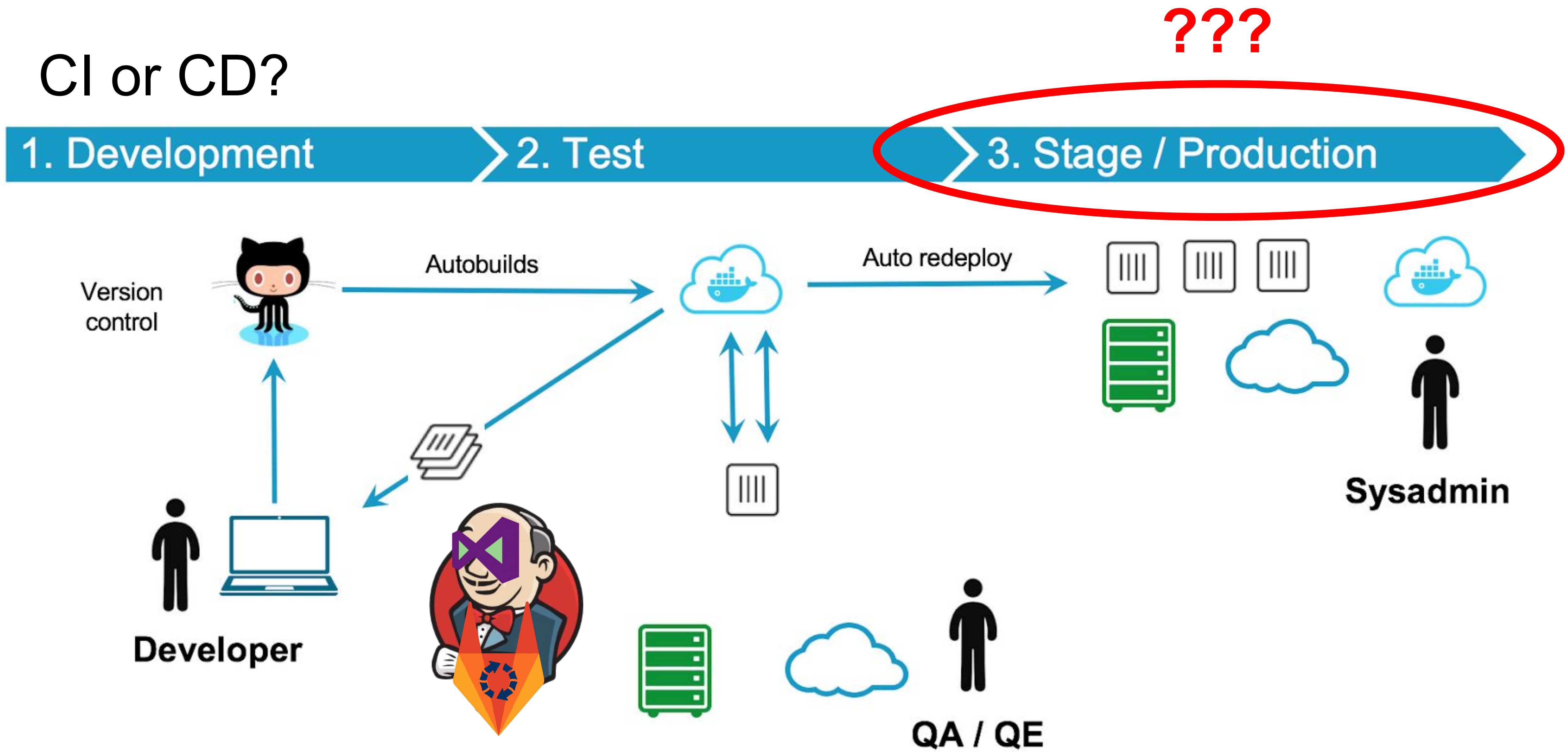
What do Continuous Integration and Continuous Delivery mean?

CI or not CI?

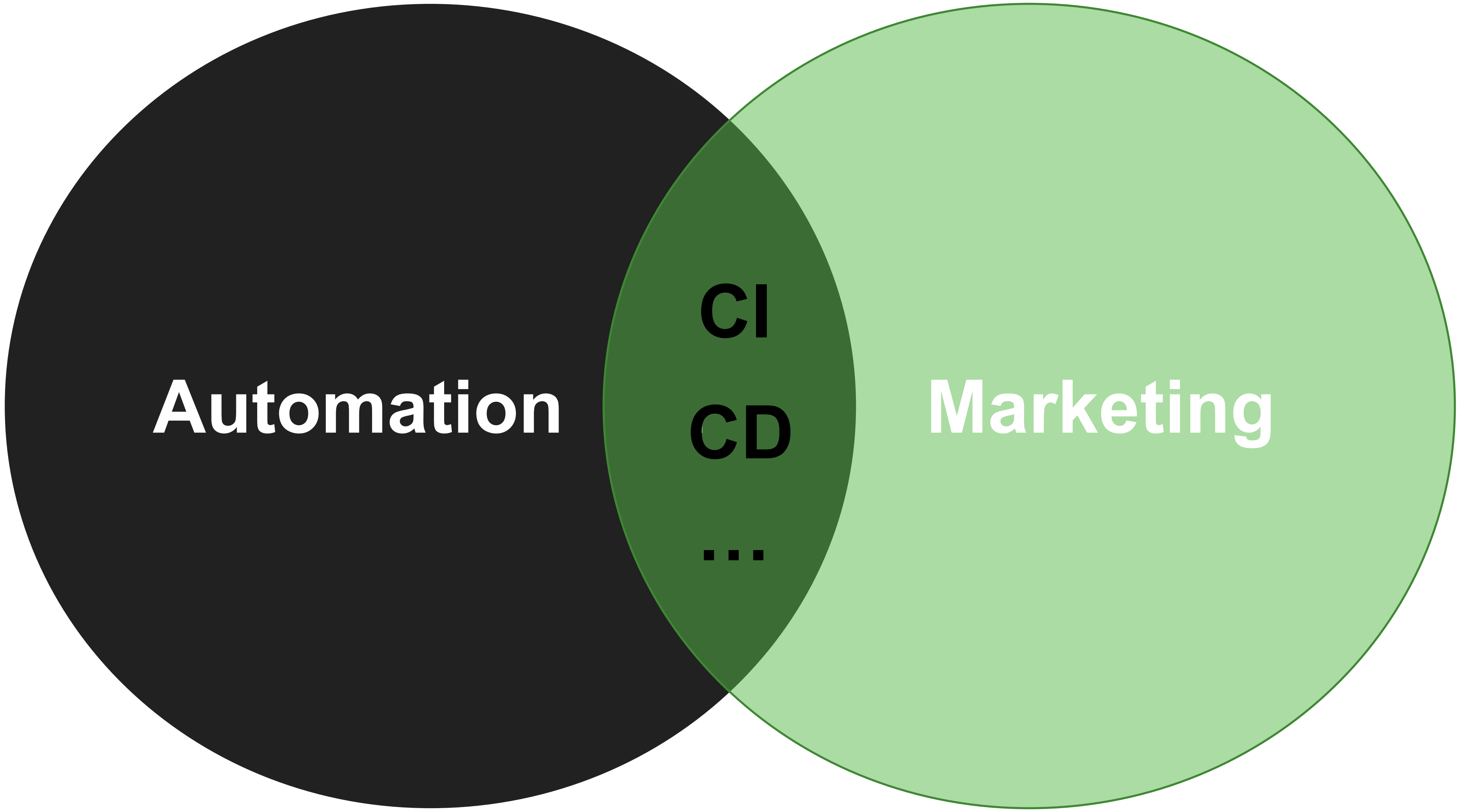


```
git clone
mvn clean verify
$? != 0 : sendmail
```

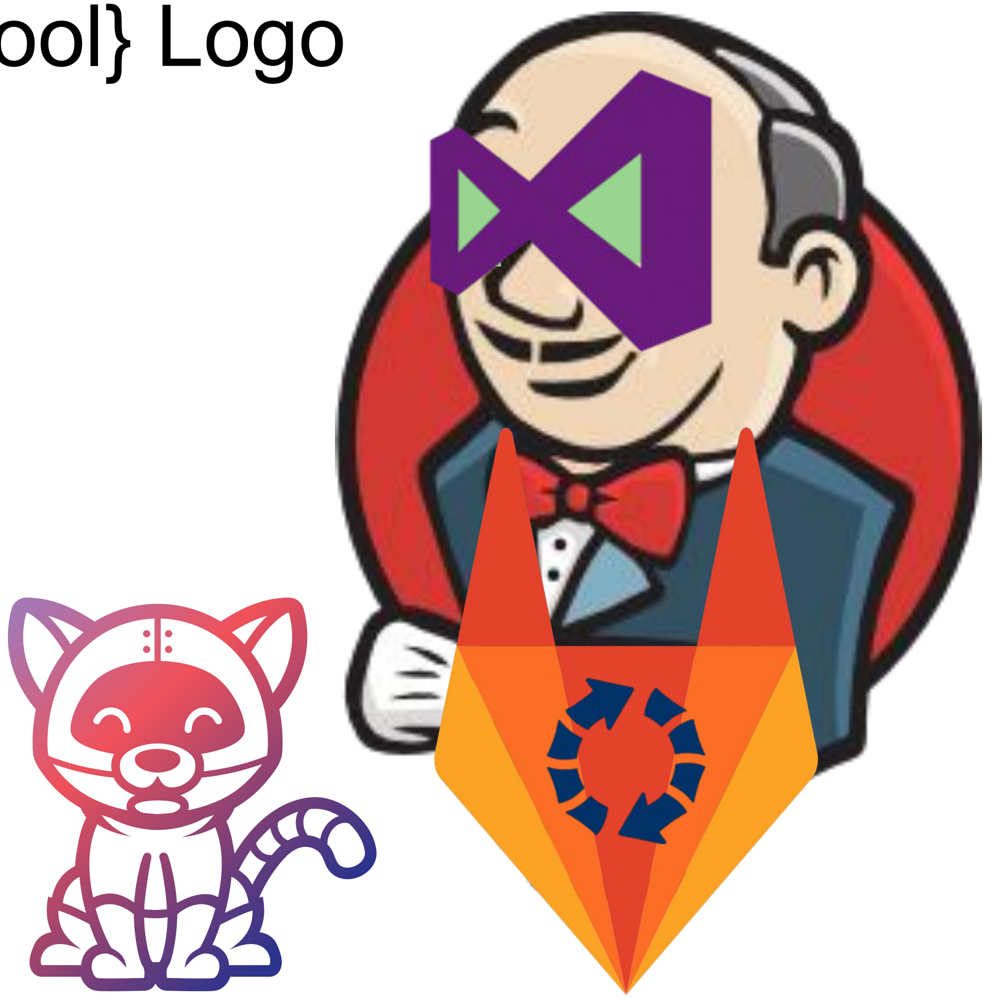
CI or CD?


???
1. Development
2. Test
3. Stage / Production
Version control
Autobuilds
Auto redeploy
Sysadmin
Developer
QA / QE

CI or CD?



Disclaimer - CI/CD Tool
${YourCICDTool} Logo



Why do **we** need CI/CD?

Selfish* view(s) on CI/CD

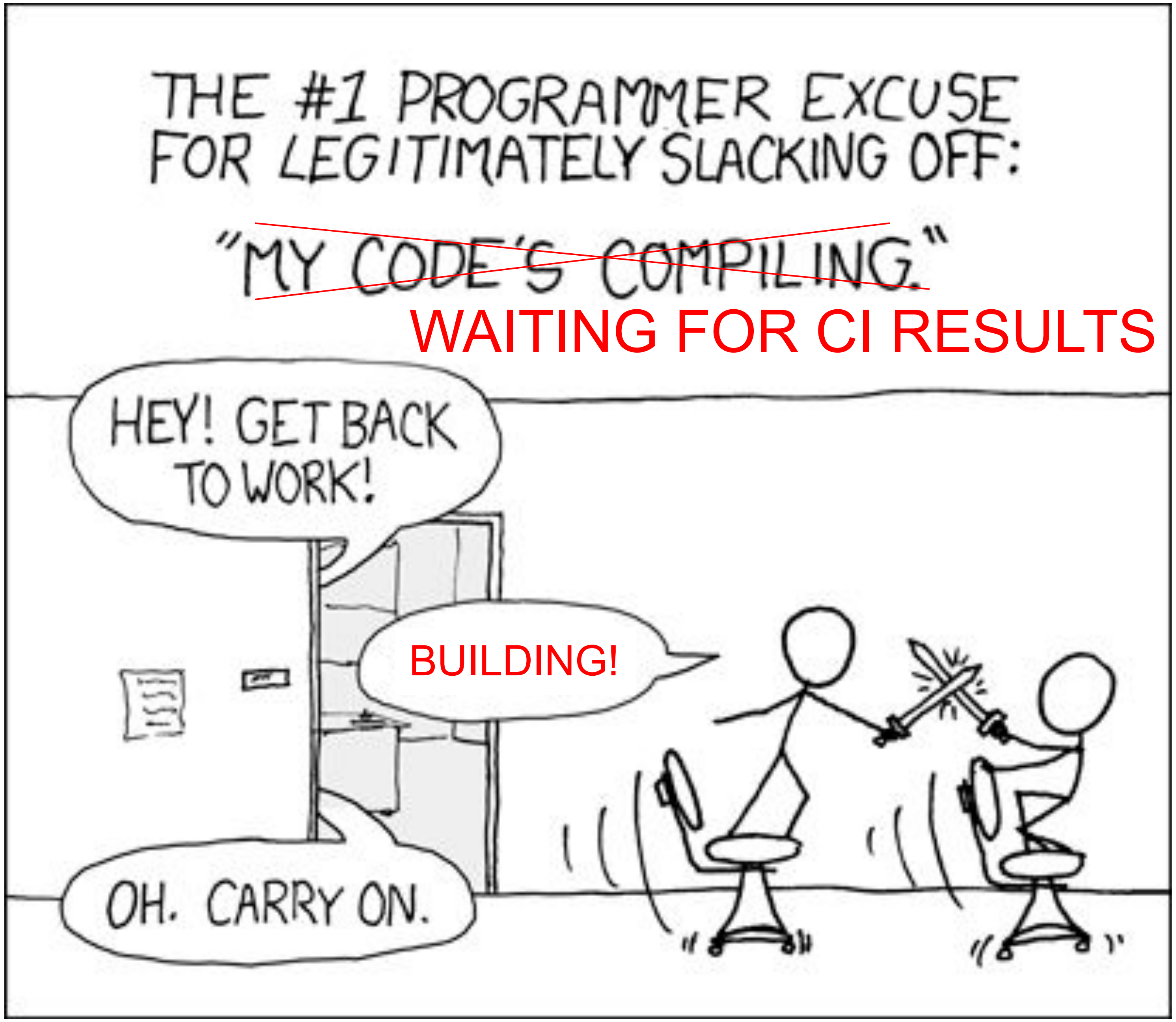
* it’s fine!

Why do **WE** need CI/CD?

- ✔ Fast issue discovery? – Yes…
- ✔ Increased product quality? – Yes…
- ✔ Project transparency? – Yes…

Managers

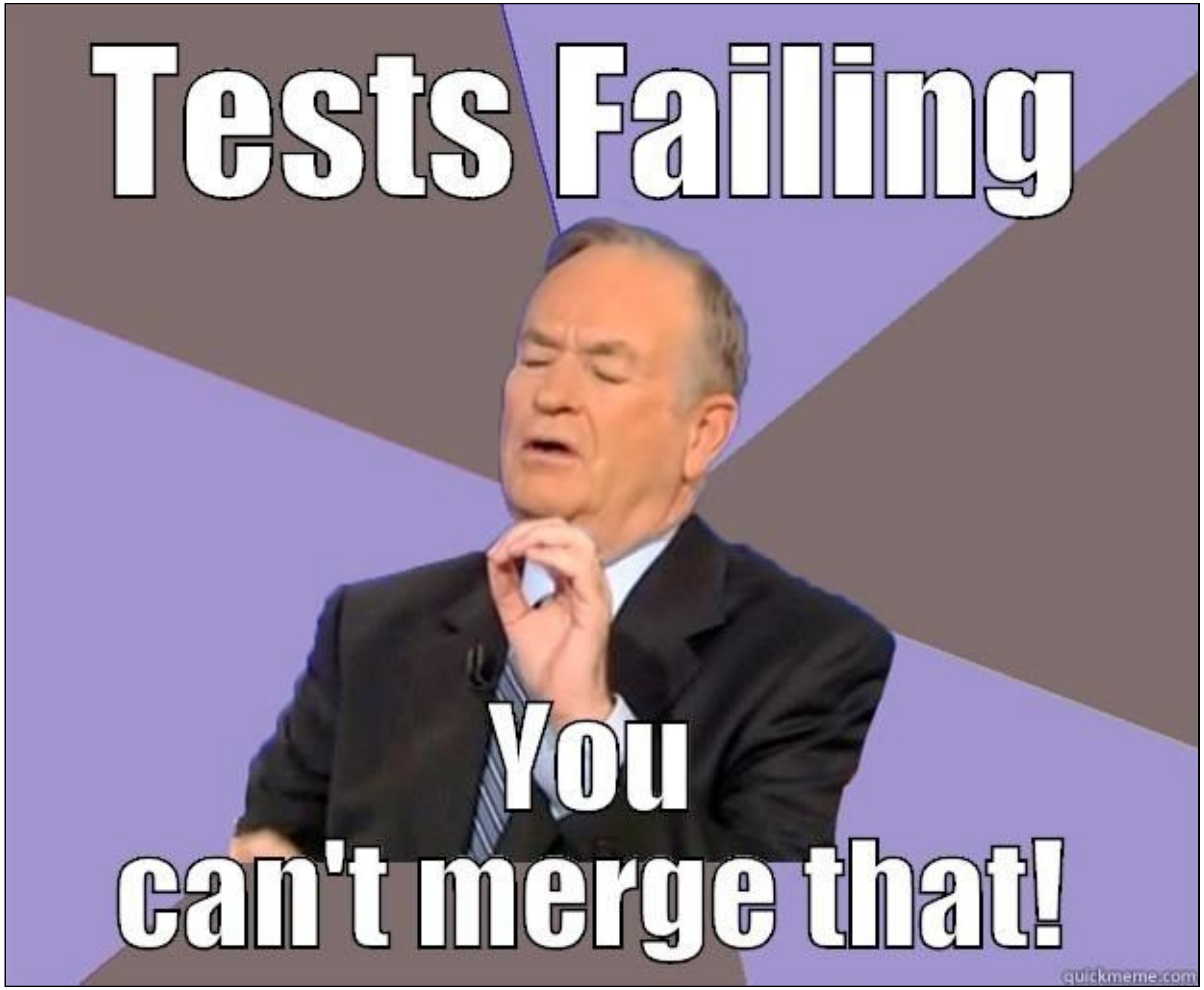
- CI/CD works
- Less resources spent by the team
- Cool data and screenshots for slides


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"~~MY CODE'S COMPILING.~~"
WAITING FOR CI RESULTS
HEY! GET BACK TO WORK!
BUILDING!
OH. CARRY ON.

http://imgs.xkcd.com/comics/compiling.png

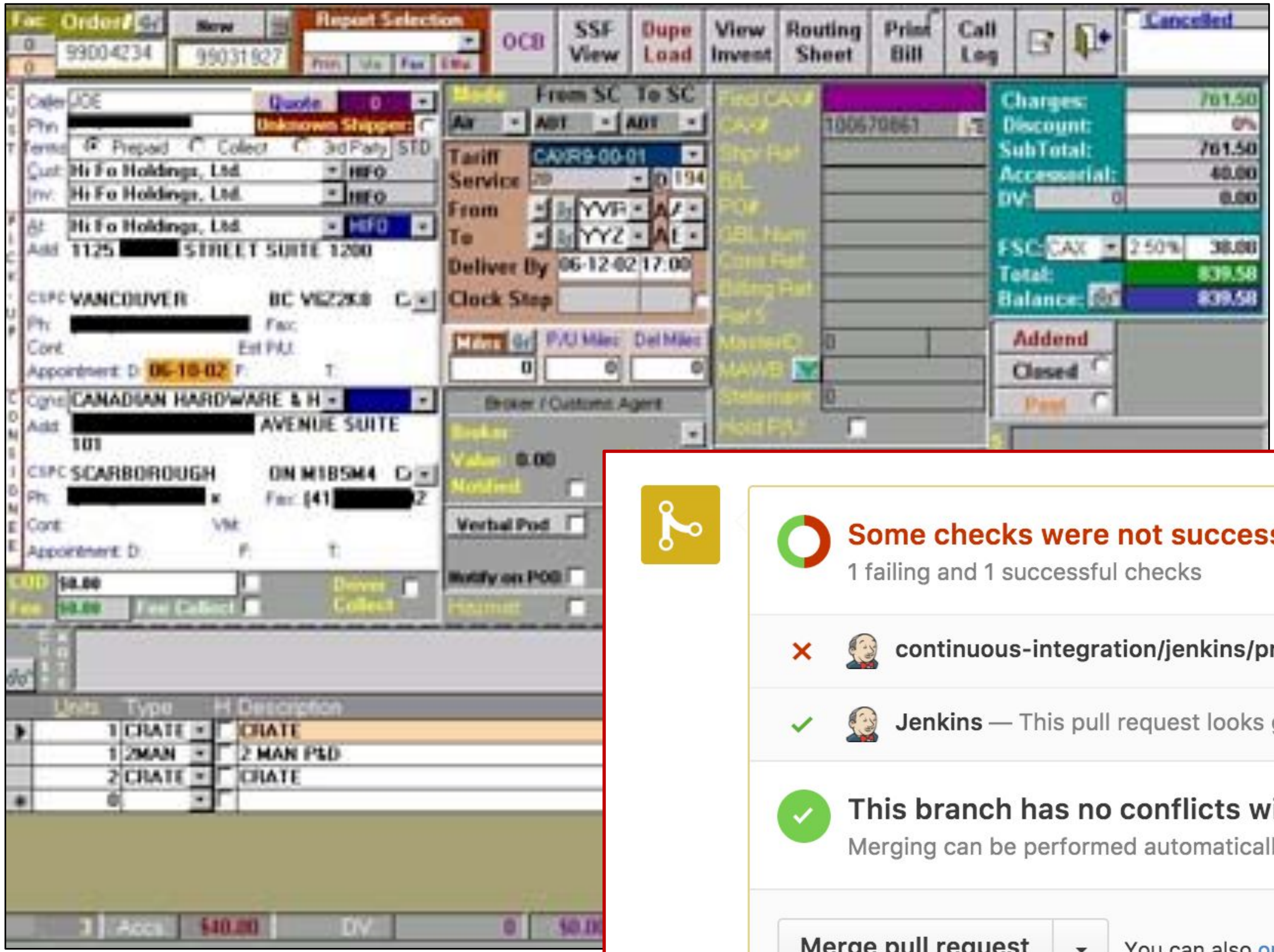
We do NOT want

- Merging changes
- Reviewing broken code
- Reviewing codestyle and code smells
- Manually checking dependencies (licenses, security, etc.)



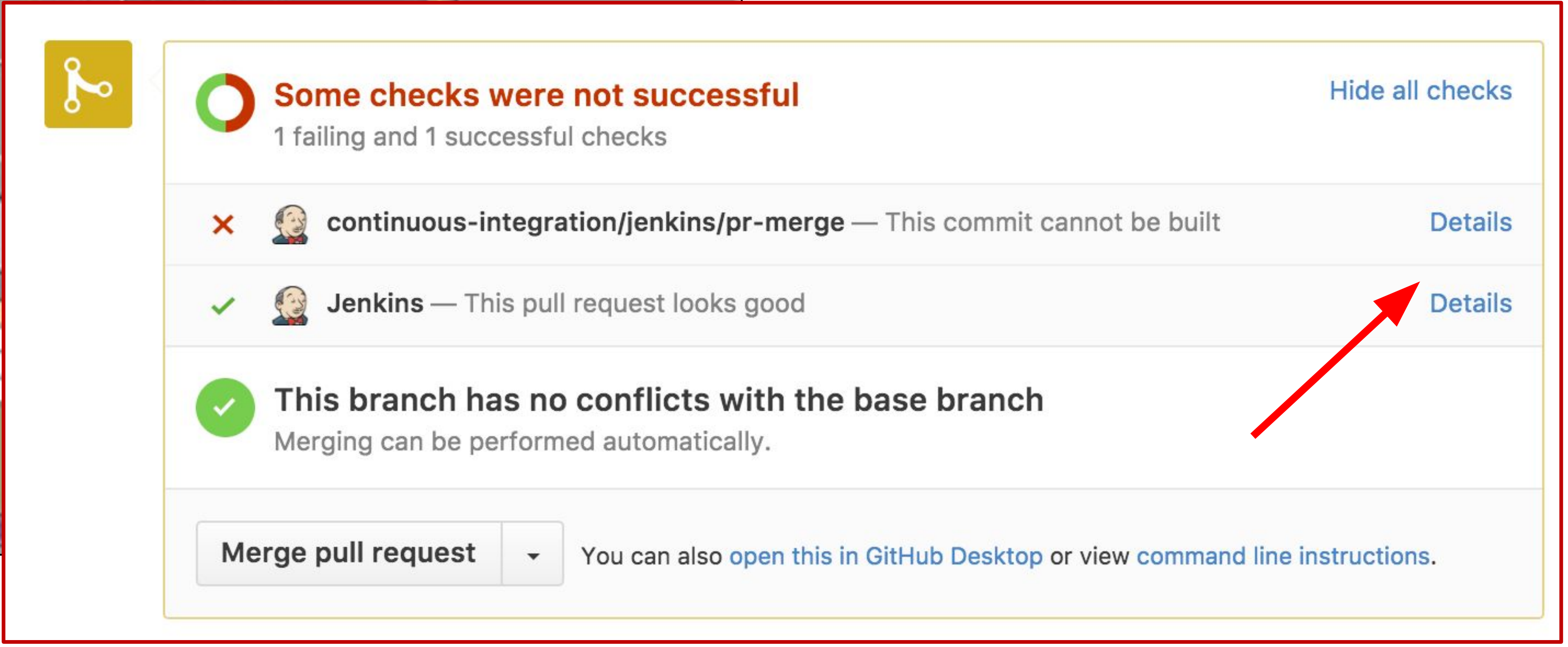
CI/CD Horrors

- Many notifications
- Going through zillions of links
- Digging into reports
- Infrastructure failures
- Flaky test
- UI!


Fac Order# New Report Selection OCB SSF View Dupe Load View Invent Routing Sheet Print Bill Call Log Cancelled
Caller JOE Quote Unknown Shipper
Phn [redacted]
Terms Prepaid Collect 3rd Party STD
Cust Hi Fo Holdings, Ltd. HIFO
Inv Hi Fo Holdings, Ltd. HIFO
At Hi Fo Holdings, Ltd. HIFO
Add 1125 [redacted] STREET SUITE 1200
C/S/PC VANCOUVER BC V6Z2K8
Ph [redacted] Fax
Cont Est P/U
Appointment D 06-10-02 F T
Cgne CANADIAN HARDWARE & H
Add [redacted] AVENUE SUITE 101
C/S/PC SCARBOROUGH ON M1B5M4
Ph [redacted] x Fax (41[redacted]2
Cont VM
Appointment D F T
COD $0.00 Fee $0.00 Fee Collect Driver Collect
Mode From SC To SC Air ABT ABT
Tariff CAXR9-00-01 Service
From YVR To YYZ
Deliver By 06-12-02 17:00 Clock Stop
Miles P/U Miles Del Miles
Broker / Customs Agent Broker Value 0.00 Notified
Verbal Pod Notify on POD Hazmat
Frgt CAX# CAX# 100670861 Shpr Ref B/L PO# GBL Num Cons Ref Billing Ref Ref 5 MasterID MAWB Statement Hold P/U
Charges: 761.50 Discount: 0% SubTotal: 761.50 Accessorial: 40.00 DV: 0 0.00
FSC: CAX 2.50% 38.08 Total: 839.58 Balance: 839.58
Addend Closed Post
Units Type H Description
1 CRATE CRATE
1 2MAN 2 MAN P&D
2 CRATE CRATE
Accs $40.00 DV 0 $0.00

We do **NOT** want

- Complex UI and reports
- Endless forms



Some checks were not successful
1 failing and 1 successful checks

Hide all checks

✗ **continuous-integration/jenkins/pr-merge** — This commit cannot be built Details

✓ **Jenkins** — This pull request looks good Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request ▾ You can also open this in GitHub Desktop or view command line instructions.

What do developers want?

Luigi is an automation engineer

An engineer does not want to study Jenkins.

He wants “build successful”

We do NOT want

- Maintain CI/CD infrastructure
- Troubleshoot infra failures
- Think about the infra at all

CI and CD 2021

- ✔ Fast issue discovery? – Yes…
- ✔ Increased product quality? – Yes…
- ✔ Project transparency? – Yes…

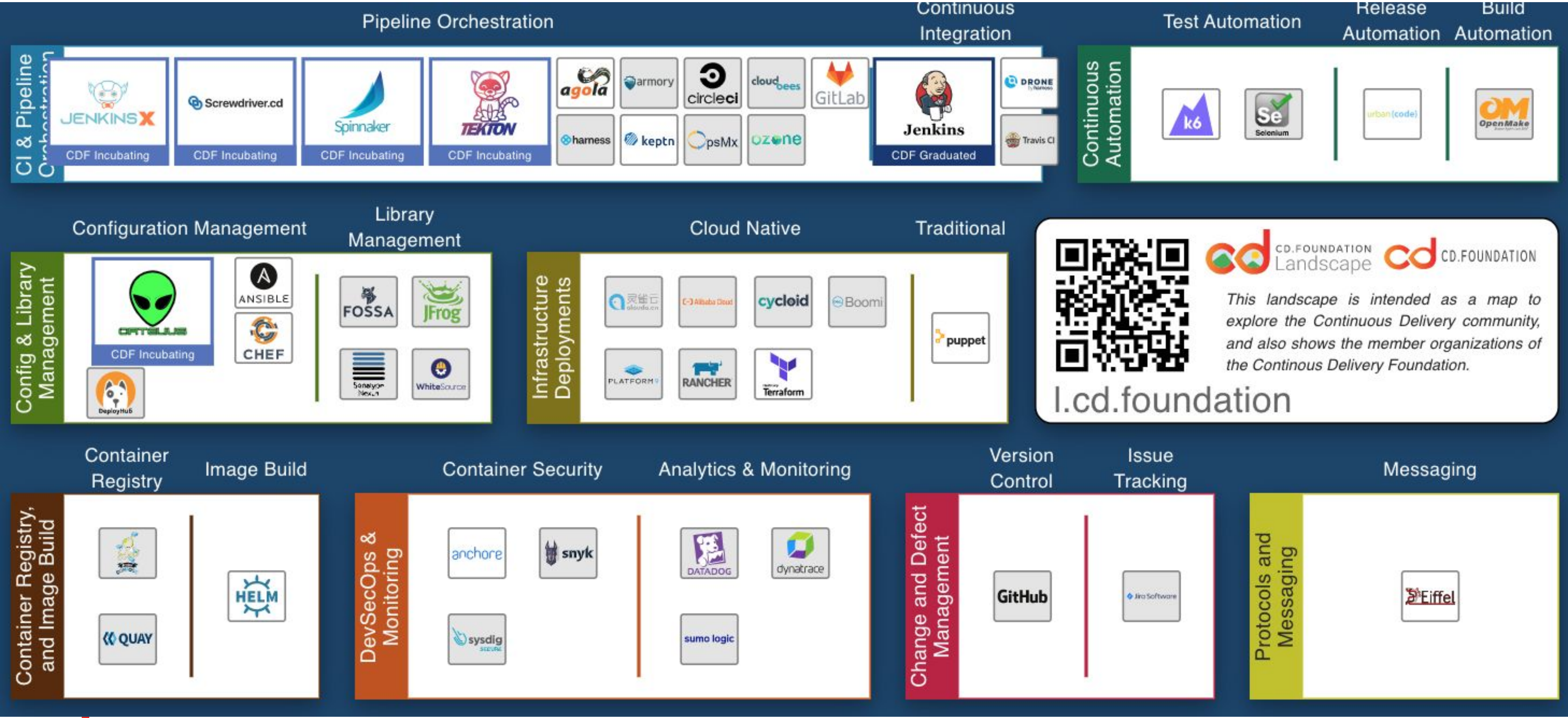
- ✔ **Contributor Experience (aka User Experience aka Developer Experience)**

Takeaways

- Continuous Integration and Continuous Delivery are about processes not tools
- CI and CD are about you => Developer experience
- Tools are here to help with implementing processes
- Tools are about automation

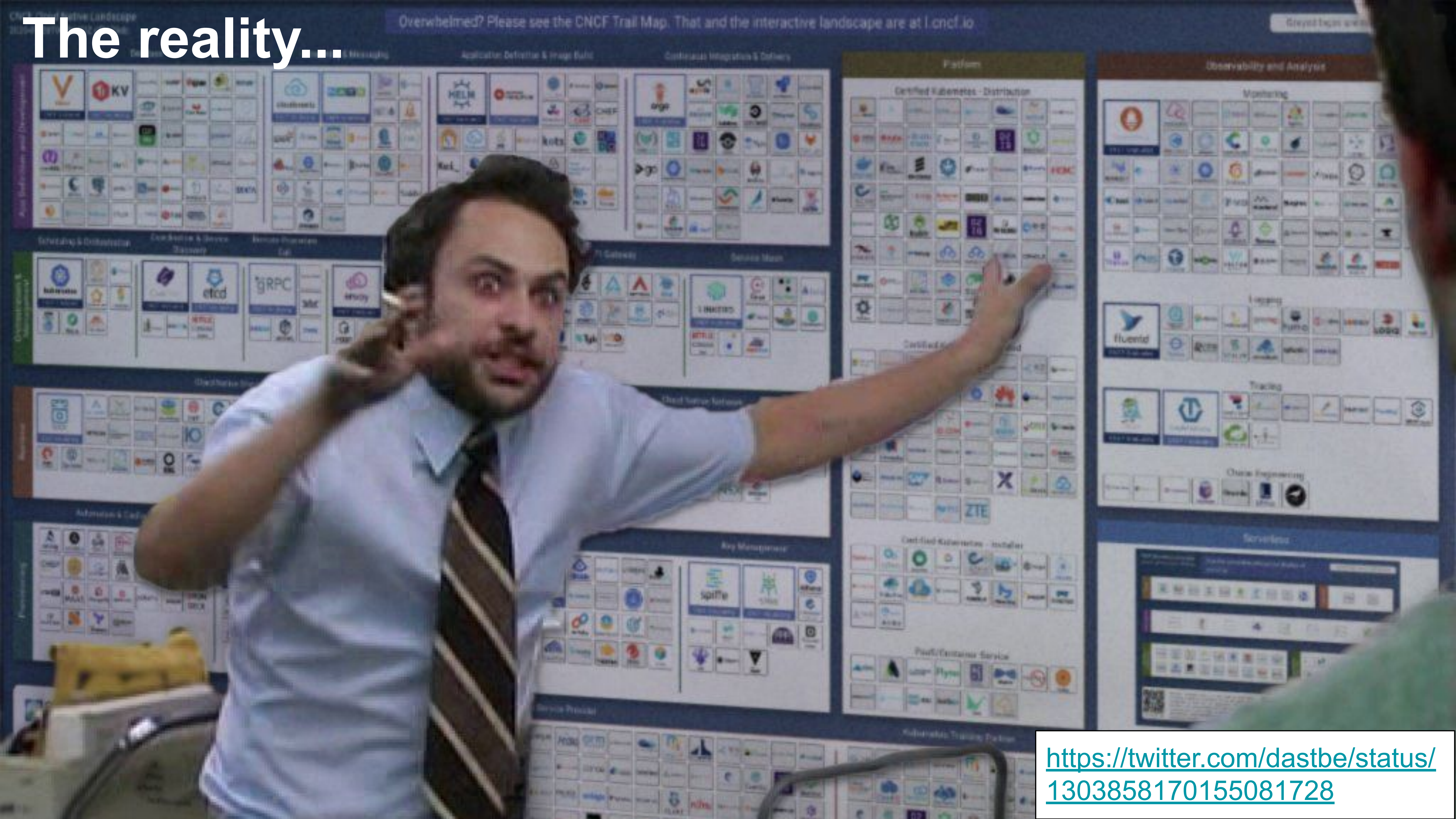
CI/CD Tools and their evolution

Continuous Delivery Landscape (Short)



https://landscape.cd.foundation/

The reality...


Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io
Platform
Observability and Analysis

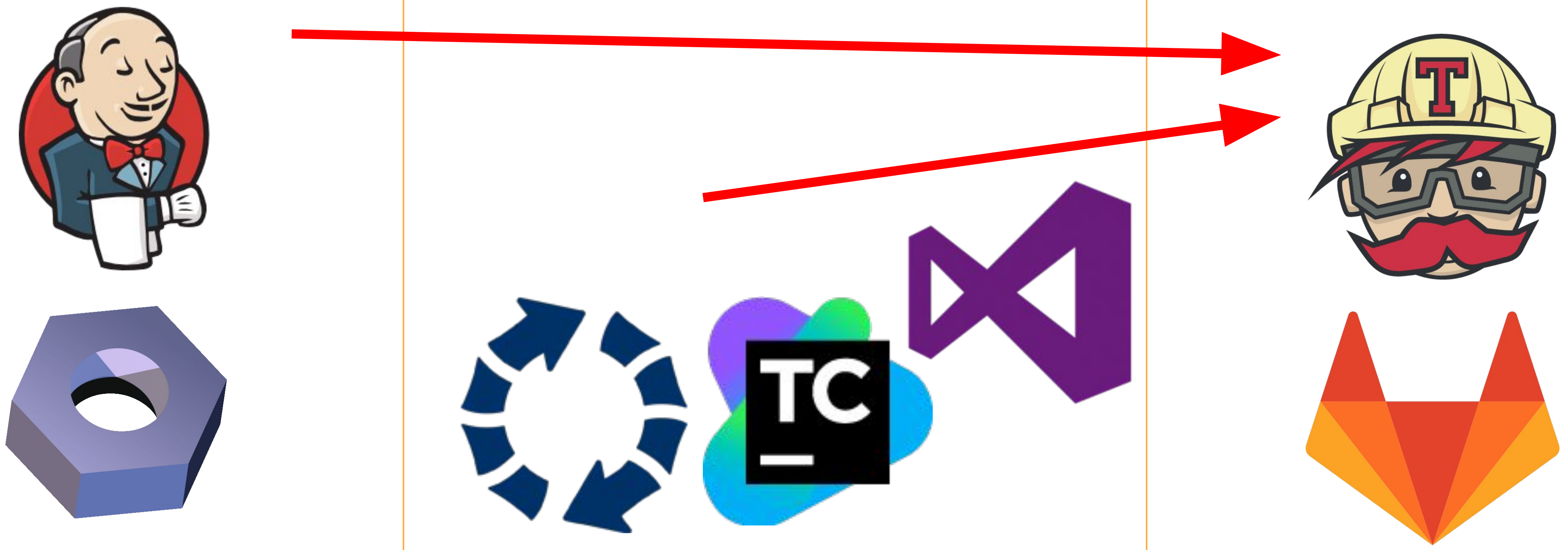
https://twitter.com/dastbe/status/1303858170155081728

Types of CI/CD tools

Frameworks

"Semi-manufactures"

"Ready to fly"



Frameworks

- Maximum flexibility
- Needs configuration

"Semi-manufactures"

- Many features provided out of the box
- Can be extended

"Ready to fly"

- Everything available out of the box
- Opinionated
- Hard to customize

Generations of CI/CD Tools (Oleg's IMHO)

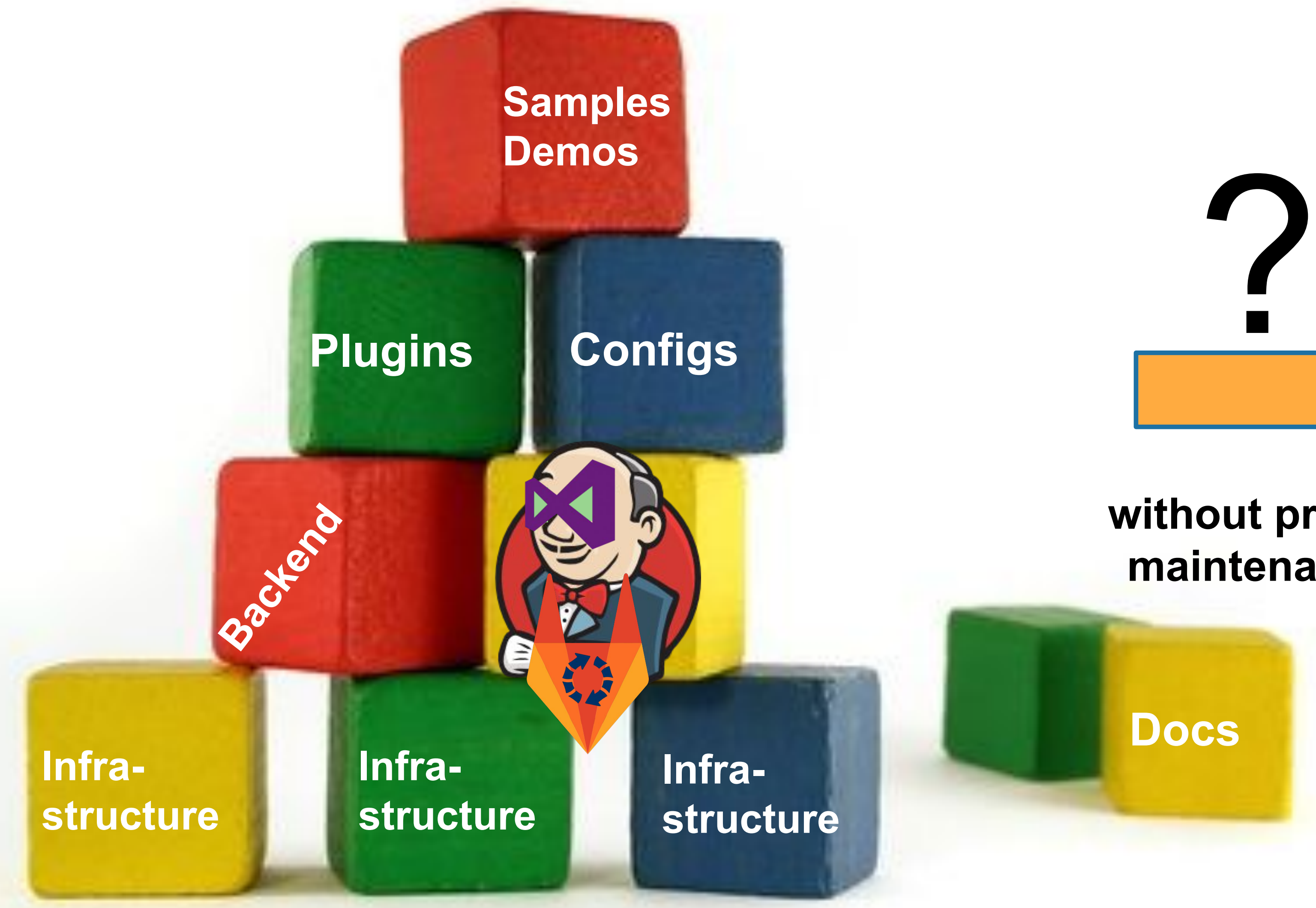
I CRON

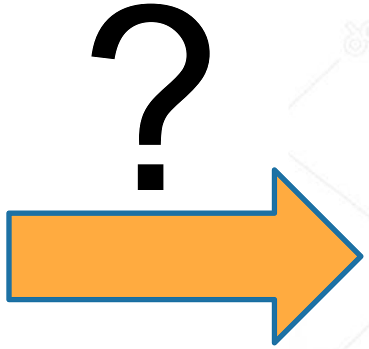
II cruisecontrol

III TC

...

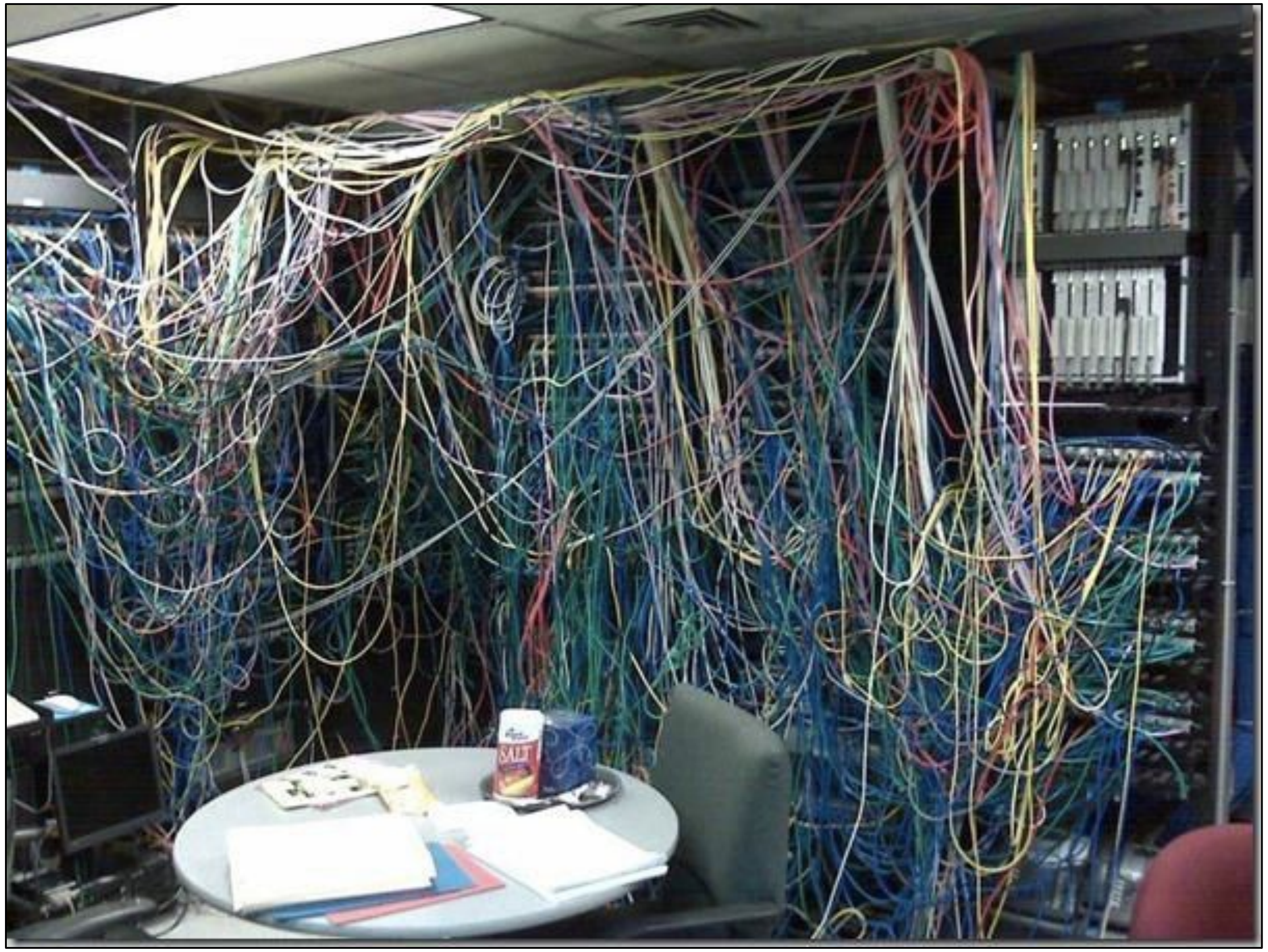
Generation III.
Complexity, mostly frameworks





without proper maintenance

Generation III. Complexity



Spaghetti Automation

- A lot of customization
- Duplication
- Complexity
- Lack of expertise

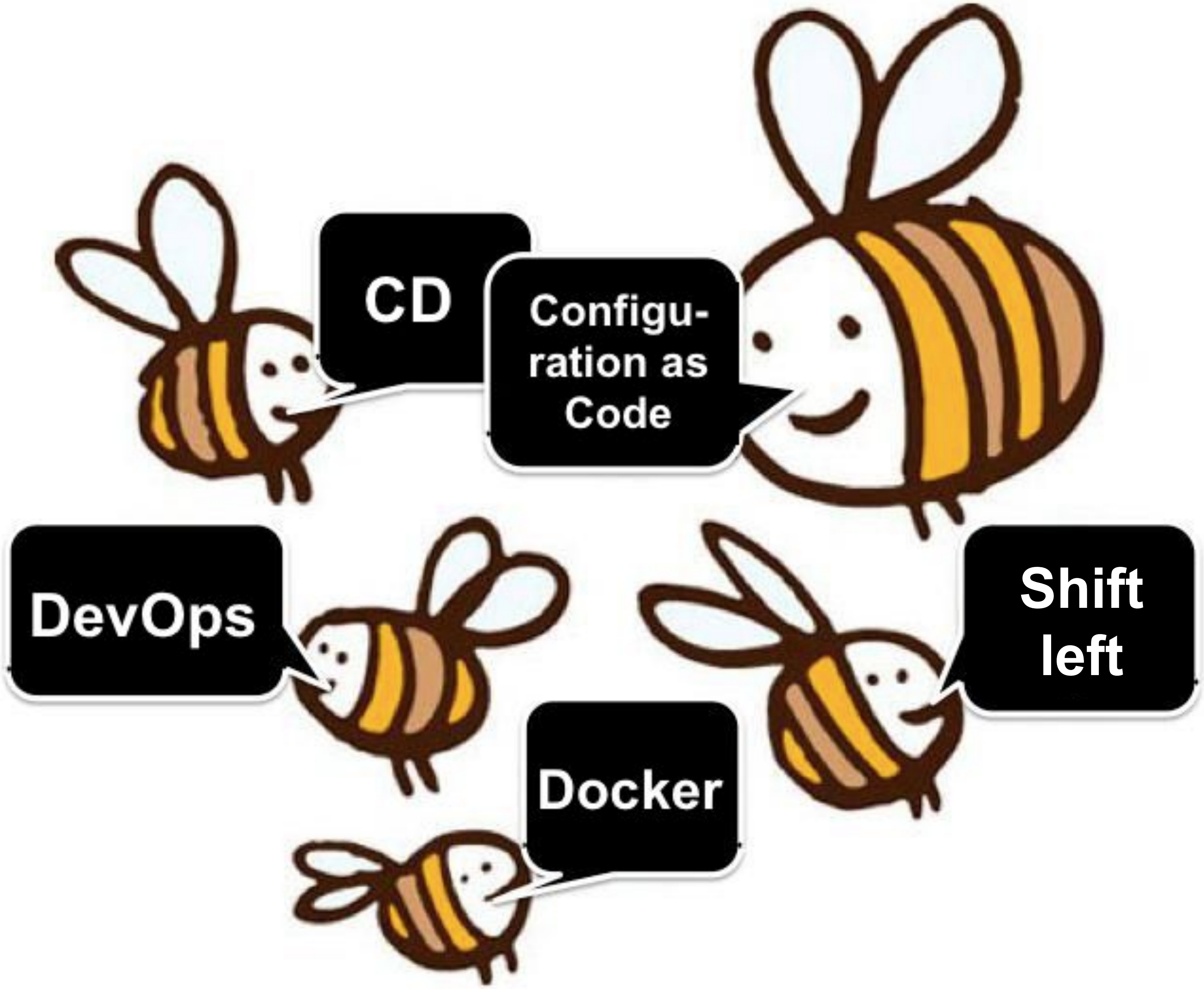
⬇

- Difficult to maintain
- Mistakes

Generation III.
New demands

Generation III evolves

… but not fast enough
for early adopters



Generations of CI/CD Tools (Oleg’s IMHO)

I CRON

II cruisecontrol

III

IV ....

Generation IV

- Takes experiences of Generation III into account
- Targets new environments
- Scalability and public clouds in mind
- Opinionated

IV

BUT:

- Has to catch-up feature wise
- Too opinionated for many

Deployment, originally designed for

Bare-Metal (sic!)

Cloud-Native

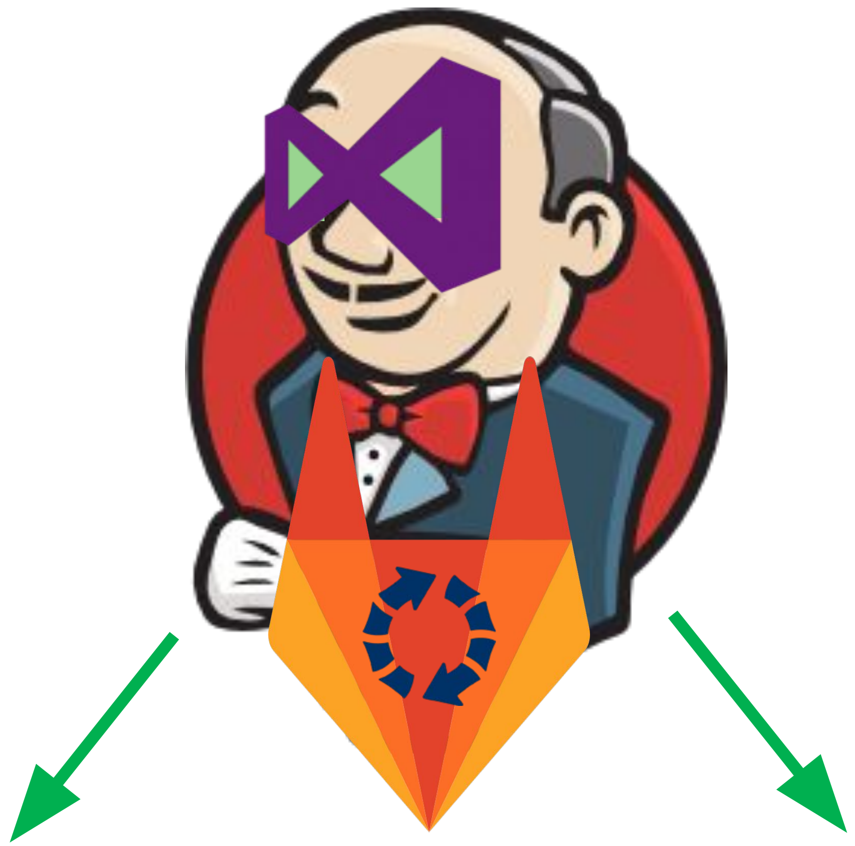
SaaS, Private and Public Clouds

[X] – Configuration,
Documentation,
Pipeline…
Everything!

[X] as code – part of CD and DevOps

- Store [X] along with the project
- Versioning changes
- Testing [X] along with the project

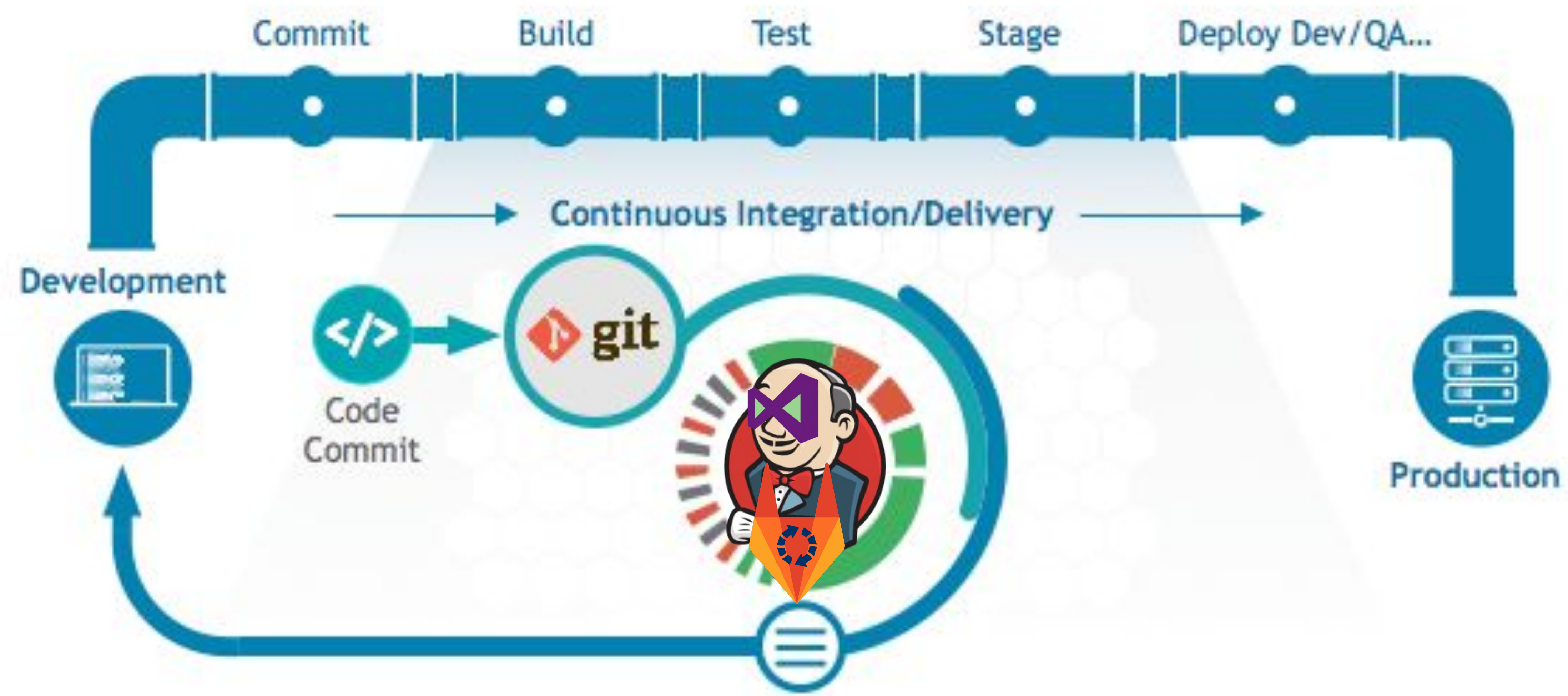
Configuration as Code in CI/CD



Managing the system itself (infrastructure as code)

Managing automation jobs (Pipeline as Code)

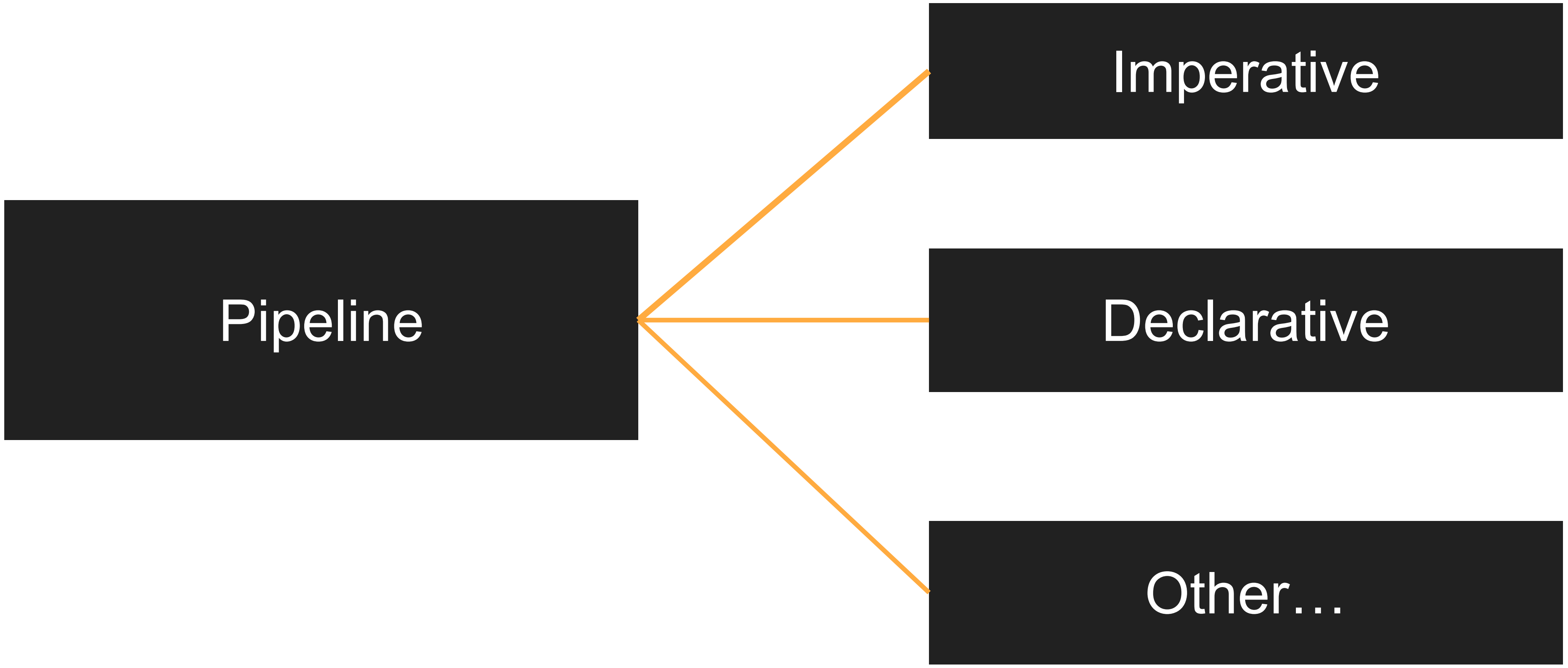
Pipelines


Commit
Build
Test
Stage
Deploy Dev/QA...
Continuous Integration/Delivery
Development
Code Commit
git
Production

Pipelines

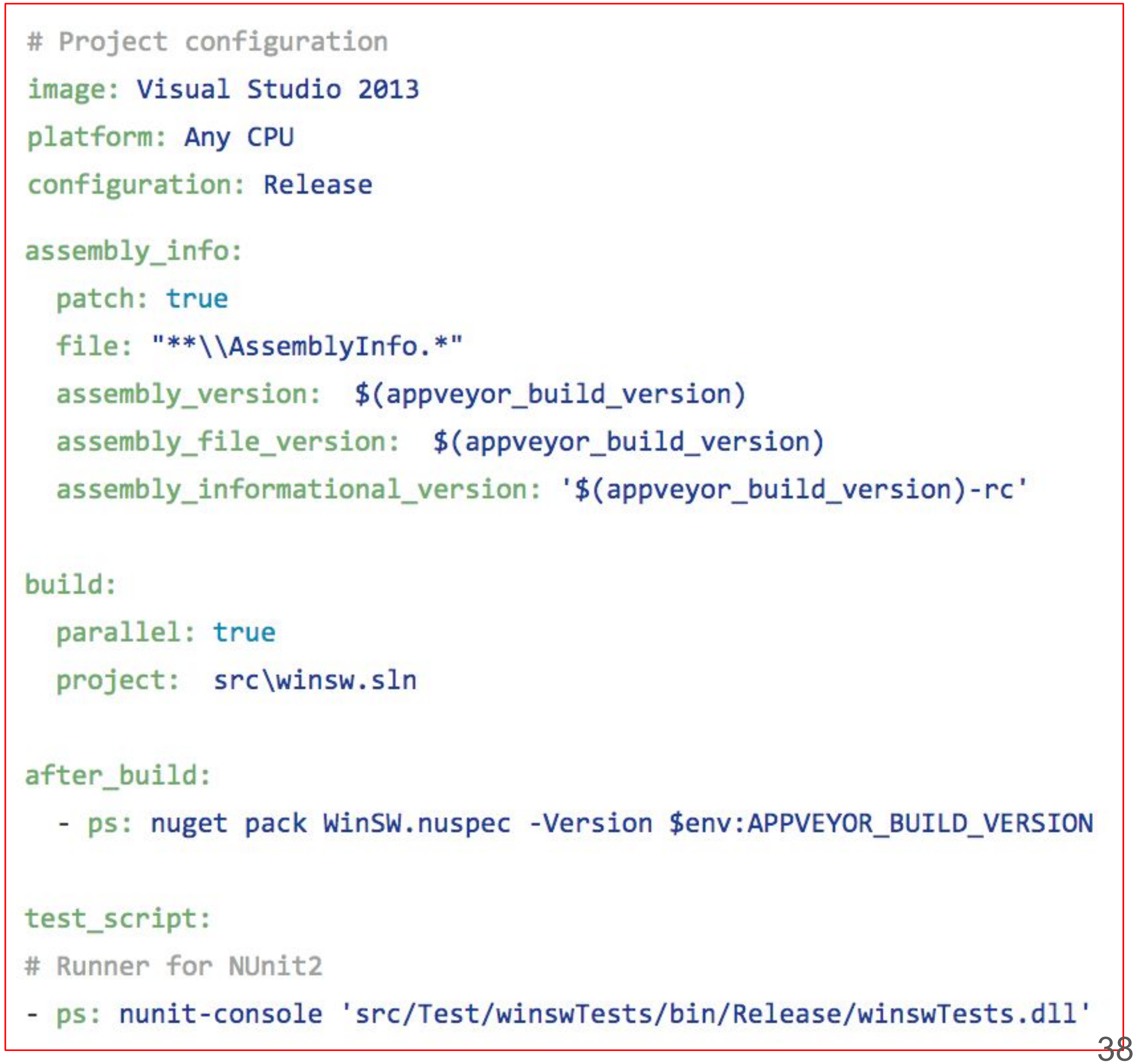
- Just another buzzword?
- All stages as code
- Various formats

Pipeline may be different



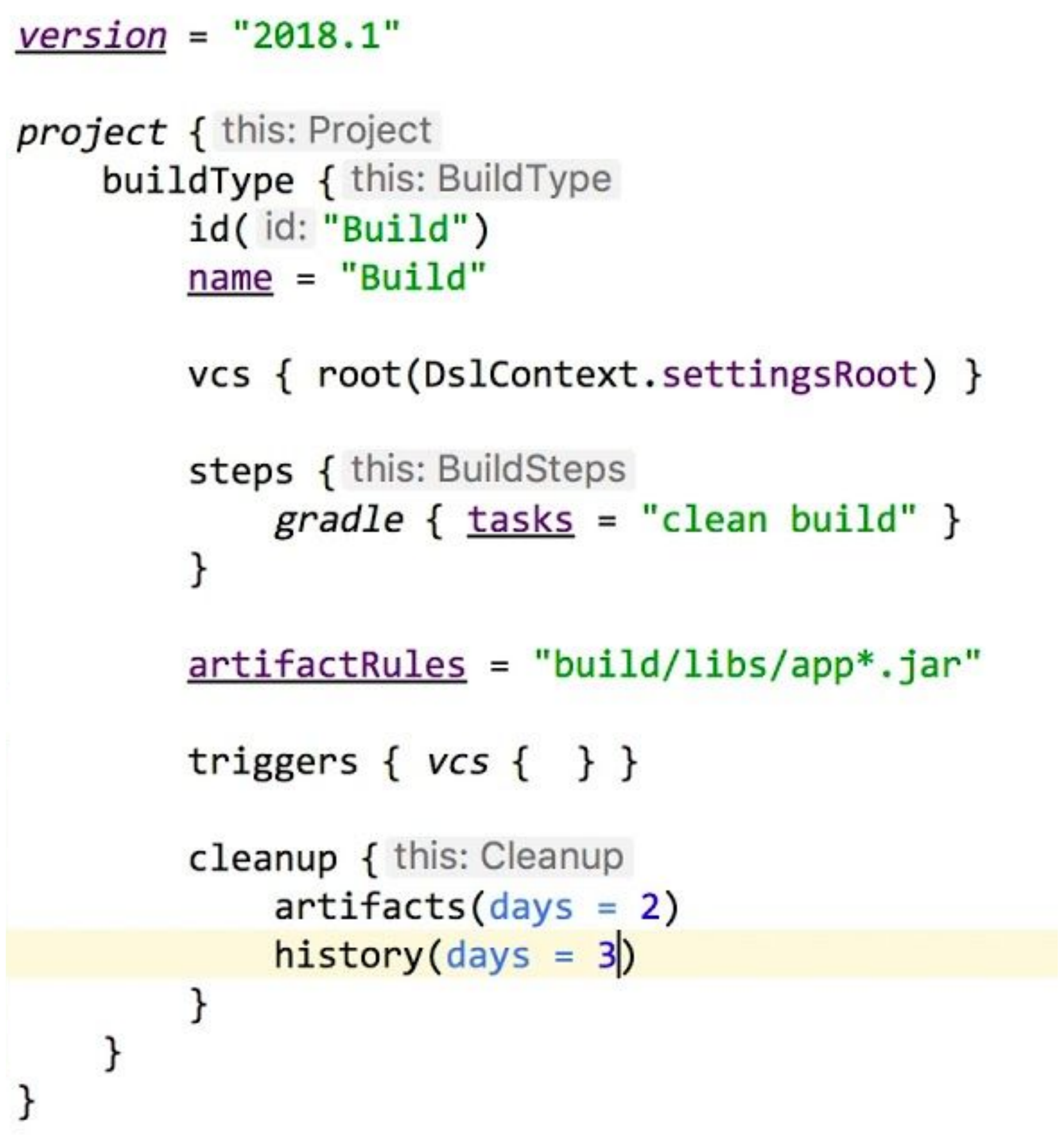
Declarative definitions are easier to use

Example - AppVeyor



```yaml
# Project configuration
image: Visual Studio 2013
platform: Any CPU
configuration: Release

assembly_info:
  patch: true
  file: "**\\AssemblyInfo.*"
  assembly_version:  $(appveyor_build_version)
  assembly_file_version:  $(appveyor_build_version)
  assembly_informational_version: '$(appveyor_build_version)-rc'

build:
  parallel: true
  project:  src\winsw.sln

after_build:
  - ps: nuget pack WinSW.nuspec -Version $env:APPVEYOR_BUILD_VERSION

test_script:
# Runner for NUnit2
- ps: nunit-console 'src/Test/winswTests/bin/Release/winswTests.dll'
```

TeamCity KotlinDSL



```
version = "2018.1"

project { this: Project
    buildType { this: BuildType
        id( id: "Build")
        name = "Build"

        vcs { root(DslContext.settingsRoot) }

        steps { this: BuildSteps
            gradle { tasks = "clean build" }
        }

        artifactRules = "build/libs/app*.jar"

        triggers { vcs {  } }

        cleanup { this: Cleanup
            artifacts(days = 2)
            history(days = 3)
        }
    }
}
```

Imperative definitions are more universal and flexible

Scripted Pipeline

```
#!groovy
def imageName = 'jenkinsciinfra/ircbot’
node('docker') {
  checkout scm
  // Немного магии для получения тэга Docker-образа
  sh 'git rev-parse HEAD > GIT_COMMIT'
  shortCommit = readFile('GIT_COMMIT').take(6)
  def imageTag = "build${shortCommit}»

  stage 'Build ircbot'
  withMavenEnv
      (["BUILD_NUMBER=${env.BUILD_NUMBER}:${shortCommit}"]) {
    sh 'make bot' // Make вызывает Maven
  }

  stage 'Build container'
  def whale = docker.build("${imageName}:${imageTag}”)
  stage 'Deploy container'
  whale.push()
}
```

https://github.com/jenkins-infra/ircbot

WHY NOT BOTH?

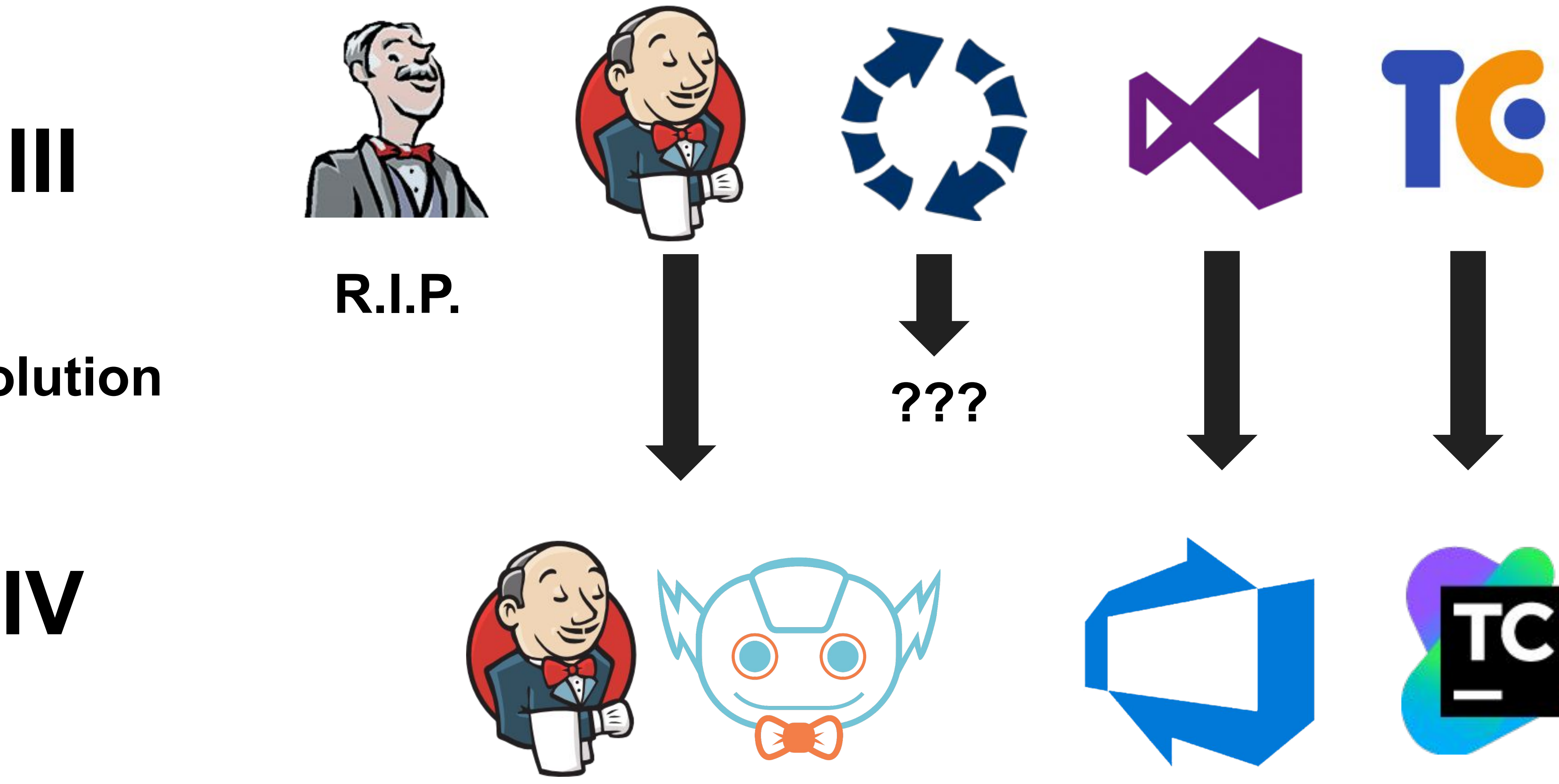
Example - Jenkins

Job DSL	**Groovy**
Jenkins Job Builder	**YAML**
…	
Jenkins Scripted Pipeline	**Groovy**
Jenkins Declarative Pipeline	**Groovy**

Gen III CI/CD tools evolved!

… or became legacy

Evolution of generation III

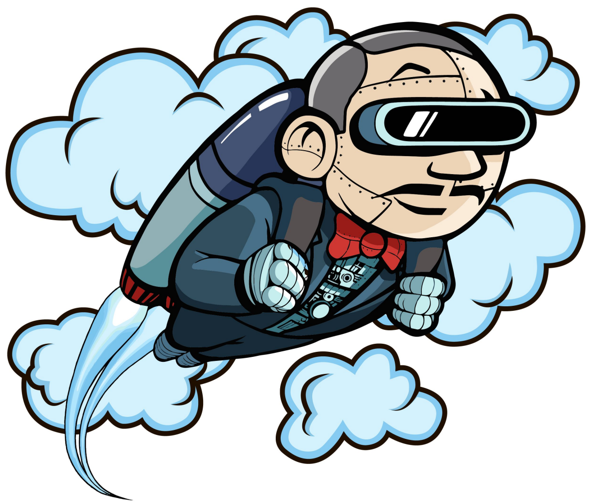
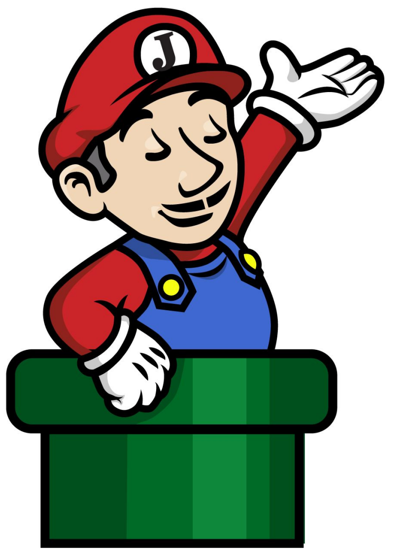


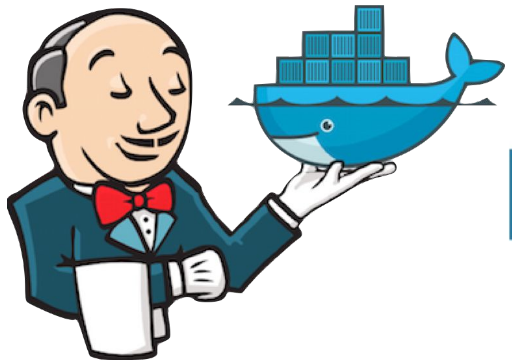
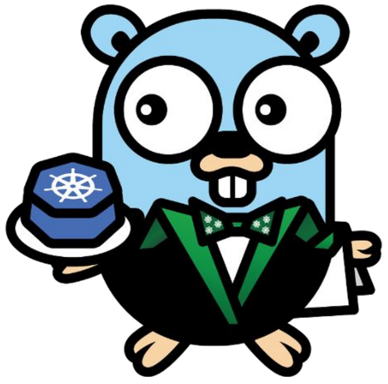
Jenkins Evolution Example

!=

Modern Jenkins

- Pipeline-as-Code
- Configuration-as-Code
- New plugins and integrations
- Modern packaging
- Distributions for public clouds

CI/CD Tools Generation V

CI/CD Tools. Generation V

- Learns from experiences of previous generations
  - Opinionated but no longer general purpose
- High specialization
  - Unix way: “Do One Thing and Do It Well”
- Mostly cloud native
- Mostly open source as a tool or as ecosystem

V

CI/CD Tools. Generation V - Examples

keptn

ORTELIUS

argo

TEKTON

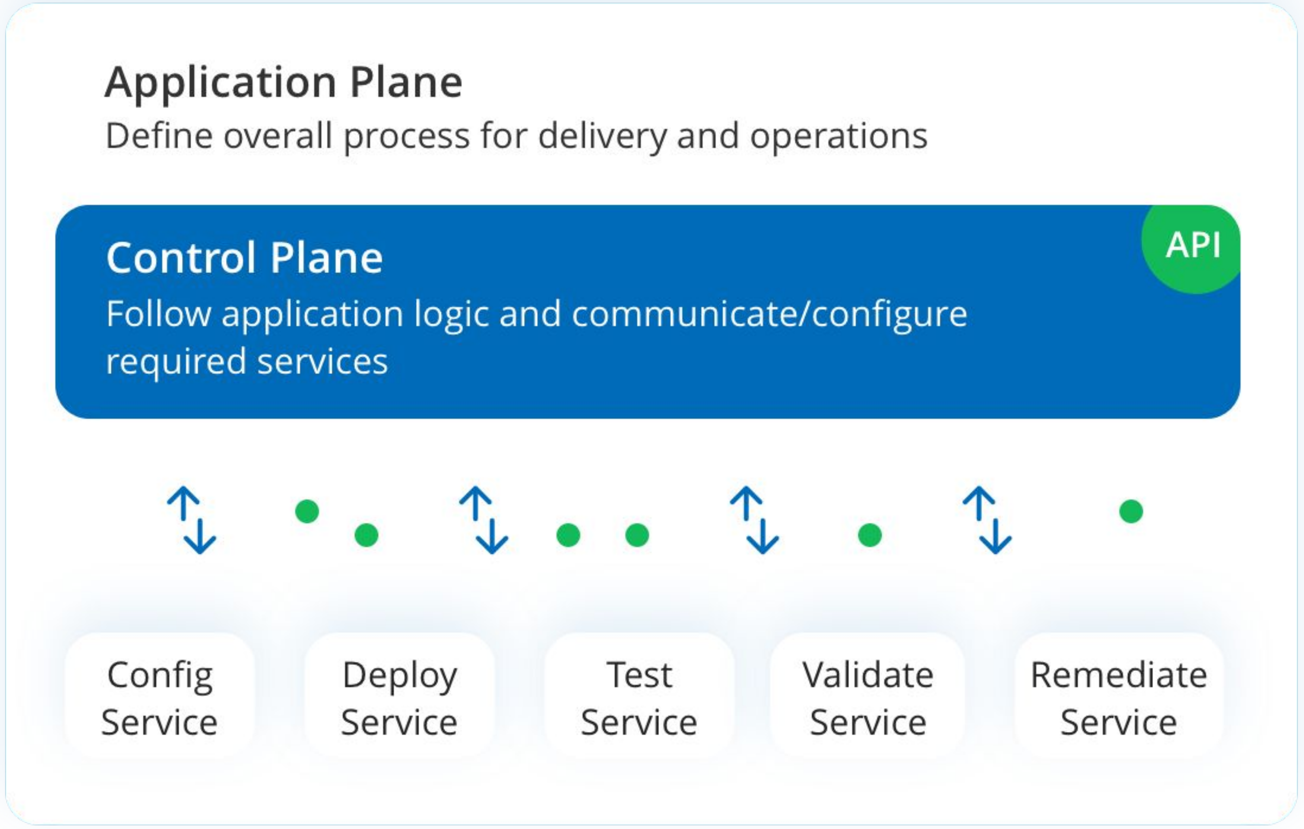
??? Both IV and V

flux

Screwdriver.cd

Example

Keptn - Control-plane for DevOps automation of cloud-native applications


Application Plane
Define overall process for delivery and operations
Control Plane
Follow application logic and communicate/configure required services
API
Config Service
Deploy Service
Test Service
Validate Service
Remediate Service

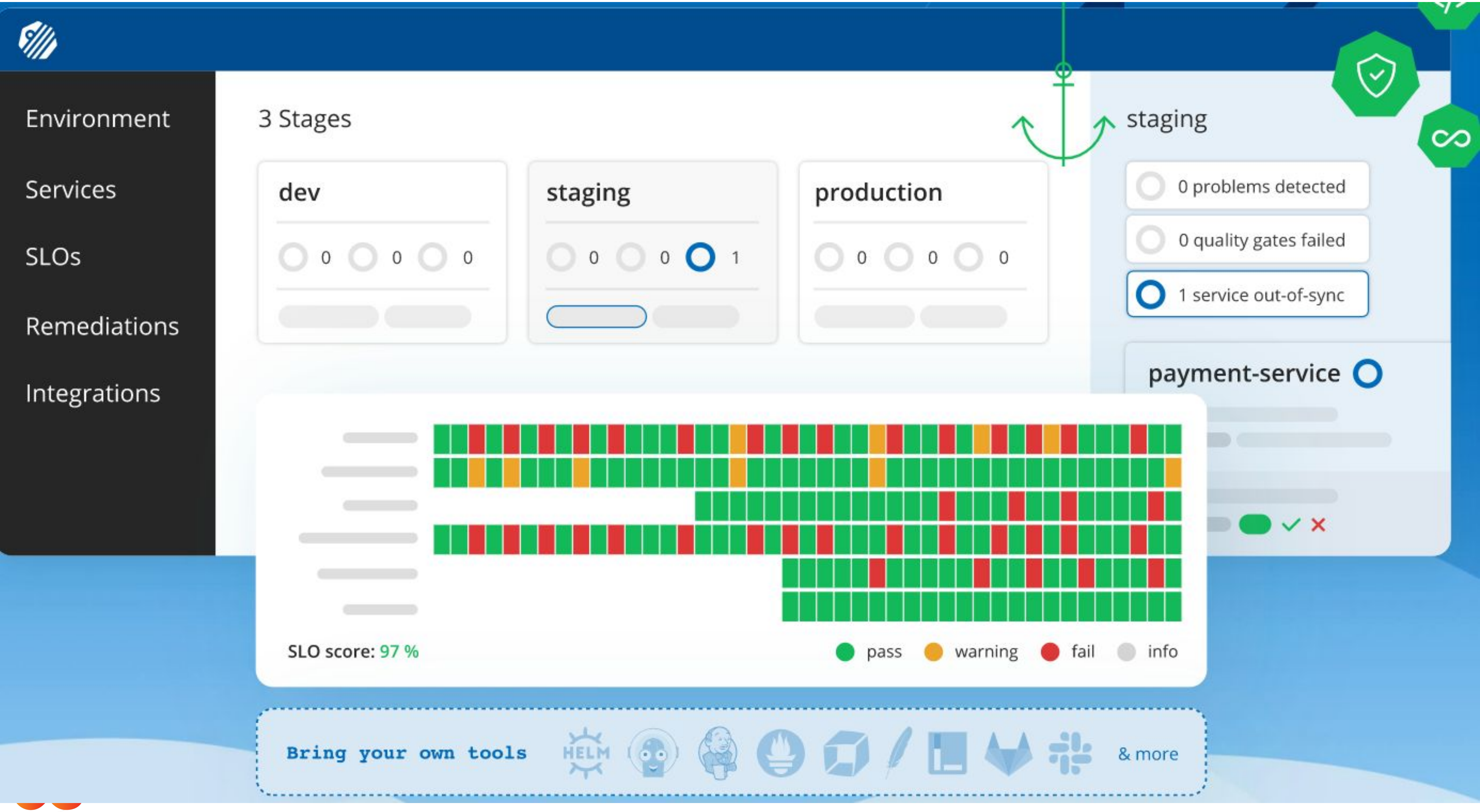
https://keptn.sh

Keptn

// Sandbox project in the CNCF

- Control plane, admin frontend/CLI
- Observability, dashboards & alerting
- SLO-driven multi-stage delivery
- Operations & remediation

https://keptn.sh



Environment
Services
SLOs
Remediations
Integrations
3 Stages
dev
0 0 0
staging
0 0 1
production
0 0 0
staging
0 problems detected
0 quality gates failed
1 service out-of-sync
payment-service
SLO score: 97 %
● pass ● warning ● fail ● info
Bring your own tools
HELM
& more

What’s now?

Oleg’s view

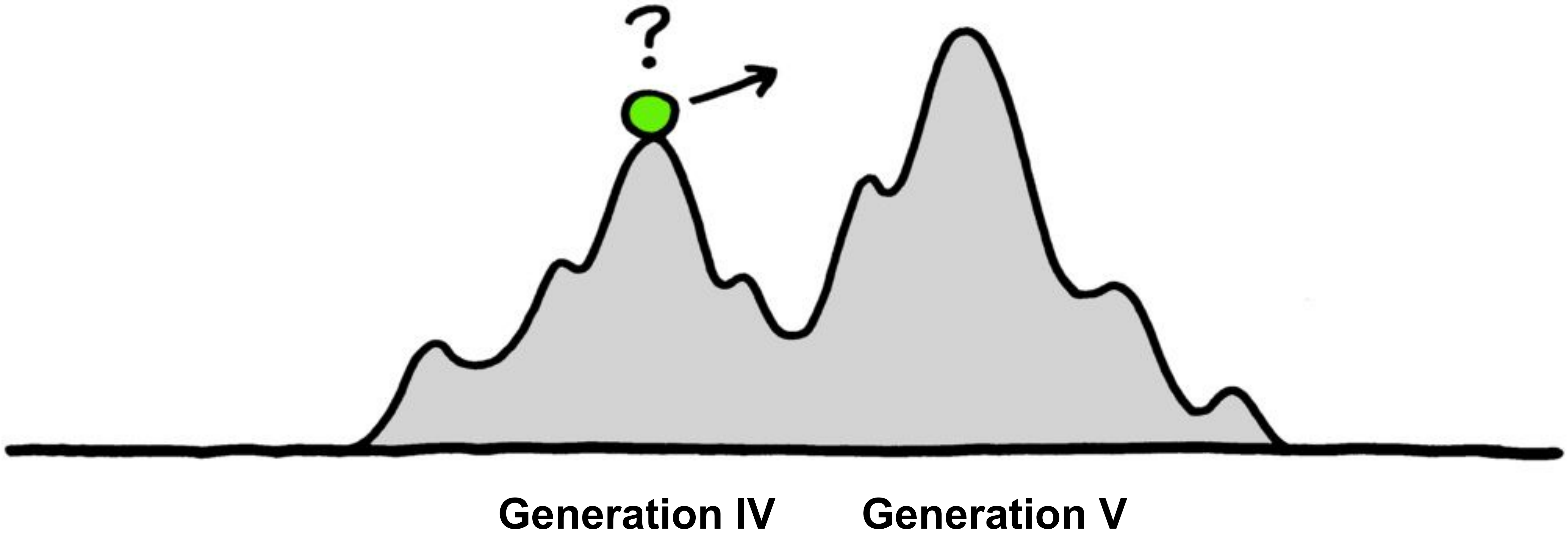
https://cd.foundation/blog/2021/02/01/trends-that-will-define-ci-cd-in-2021/

CI/CD Tools

- **Battle of generations, continued**
  - Generation 3 and 4 tools should keep evolving… or die
  - Generation 5 keeps emerging

Getting out of Local Optimum

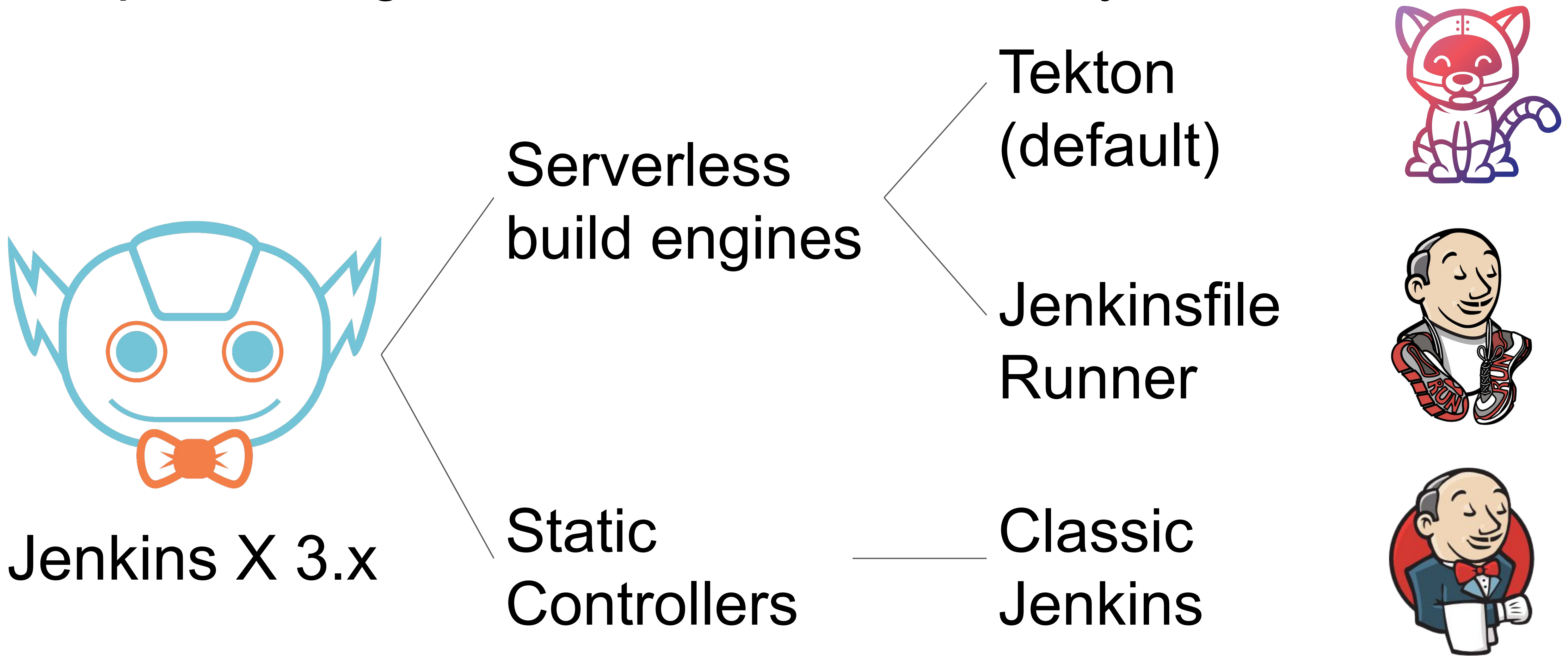
“Jenkins: Shifting gears”, Kohsuke Kawaguchi, 2018
jenkins.io/blog/2018/08/31/shifting-gears



CI/CD Tools

- Battle of generations, continued
- **Automation/Pipeline engines become a commodity**

Pipeline engines become a commodity



https://jenkins-x.io/

CI/CD Tools

- Battle of generations, continued
- Automation/Pipeline engines become a commodity
- **Focus on emerging environments and use-cases**
  - Cloud native environments
  - Container environments
  - Kubernetes, Kubernetes, Kubernetes

Being Cloud Native

- Best service for each need
- Pay per use
- “Infinite” scaling
- Easy to use
- Easy to maintain
- Fast to develop

CLOUD NATIVE
COMPUTING FOUNDATION

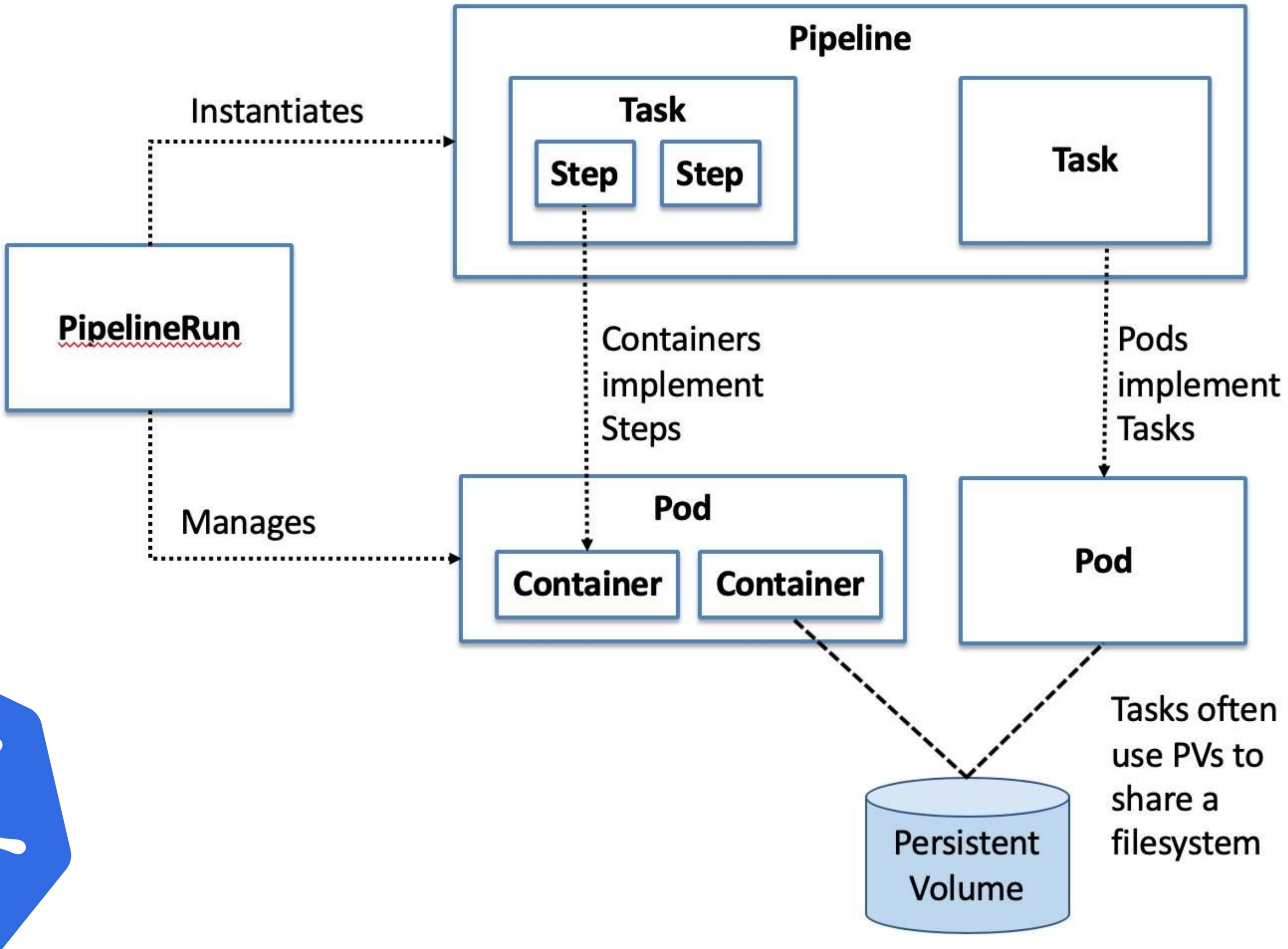
Example: Tekton

Tekton is a powerful and flexible **open-source framework** for creating CI/CD systems, allowing developers to **build, test, and deploy** across cloud providers and on-premise systems.

https://tekton.dev/

Example: Tekton

- Cloud Native
- Kubernetes Native
- Everything is in container
- Steps and pluggability via containers



https://tekton.dev/

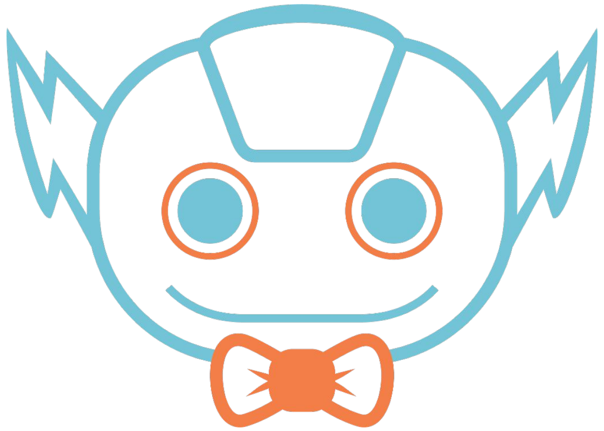
CI/CD Tools

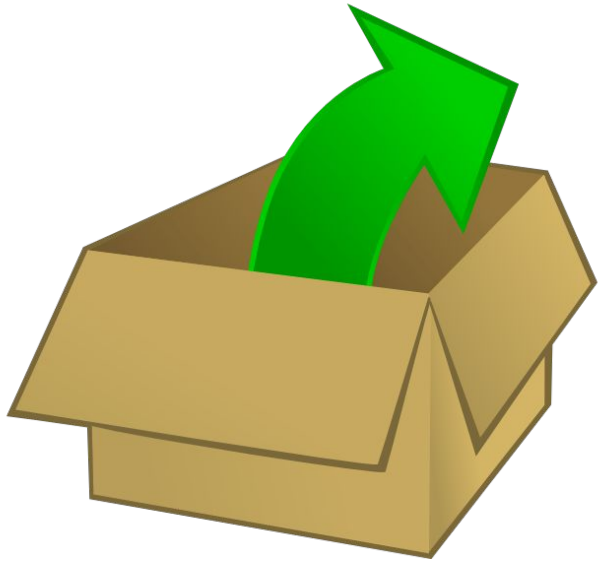
- Battle of generations, continued
- Automation/Pipeline engines become a commodity
- Focus on emerging environments and use-cases
  - Cloud native environments
  - Container environments
  - Kubernetes, Kubernetes, Kubernetes
- **Focus on developer experience**

Focus on developer experience

Jenkins X. Out of the Box experience

- Build packs - Continuous Delivery
- GitOps
- Nexus, chartmuseum, monocular
- Environments: Local, staging, production
- IDE Integrations





CI/CD Tools

- Battle of generations, continued
- Automation/Pipeline engines become a commodity
- Focus on emerging environments and use-cases
  - Cloud native environments
  - Container environments
  - Kubernetes, Kubernetes, Kubernetes
- Focus on developer experience
- **Software as a Service**

Software as a Service (SaaS)

- Ongoing switch to SaaS
- SaaS does will no longer block you from using local resources
  - GitHub Runner
  - Gitlab Runner
  - …
- Internal SaaS Platforms
  - Social coding and Inner Source
  - Same principles as SaaS

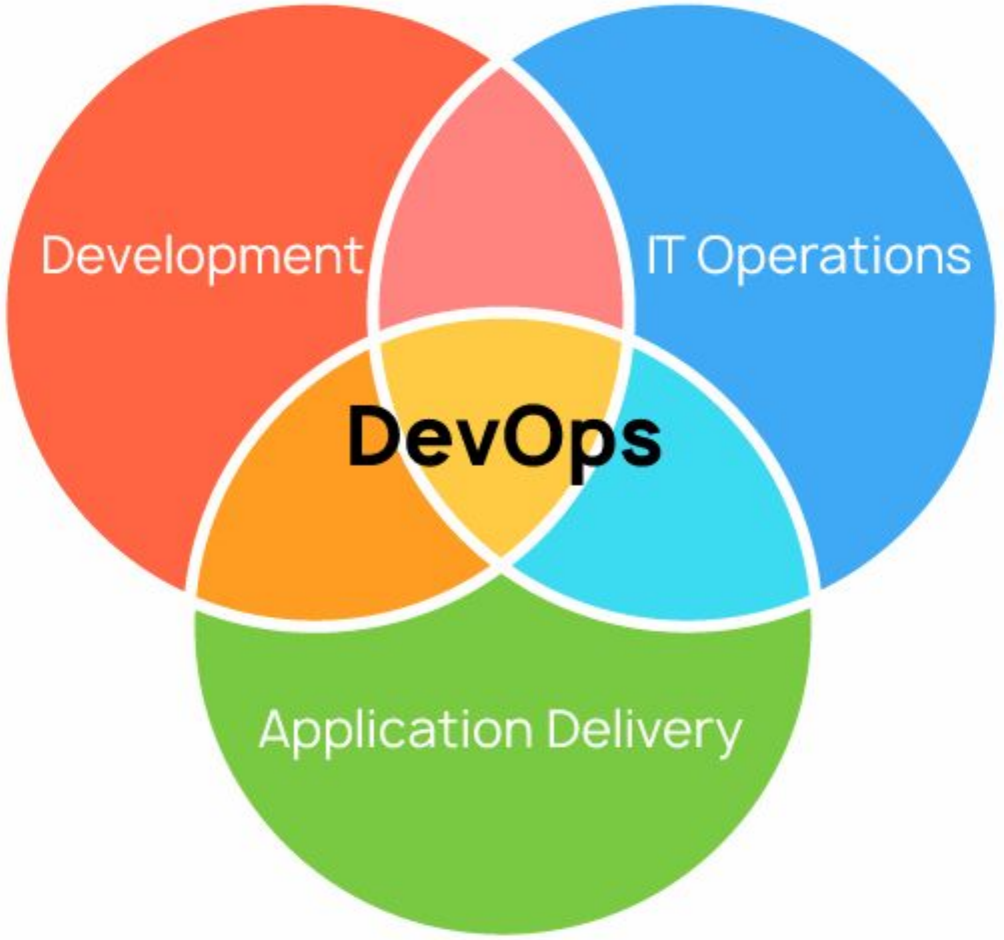
Change is not just about tools. It is about culture

Culture shift

- Not just developers anymore
- All roles are involved
- Cross functional communications
- TeamWork 💪

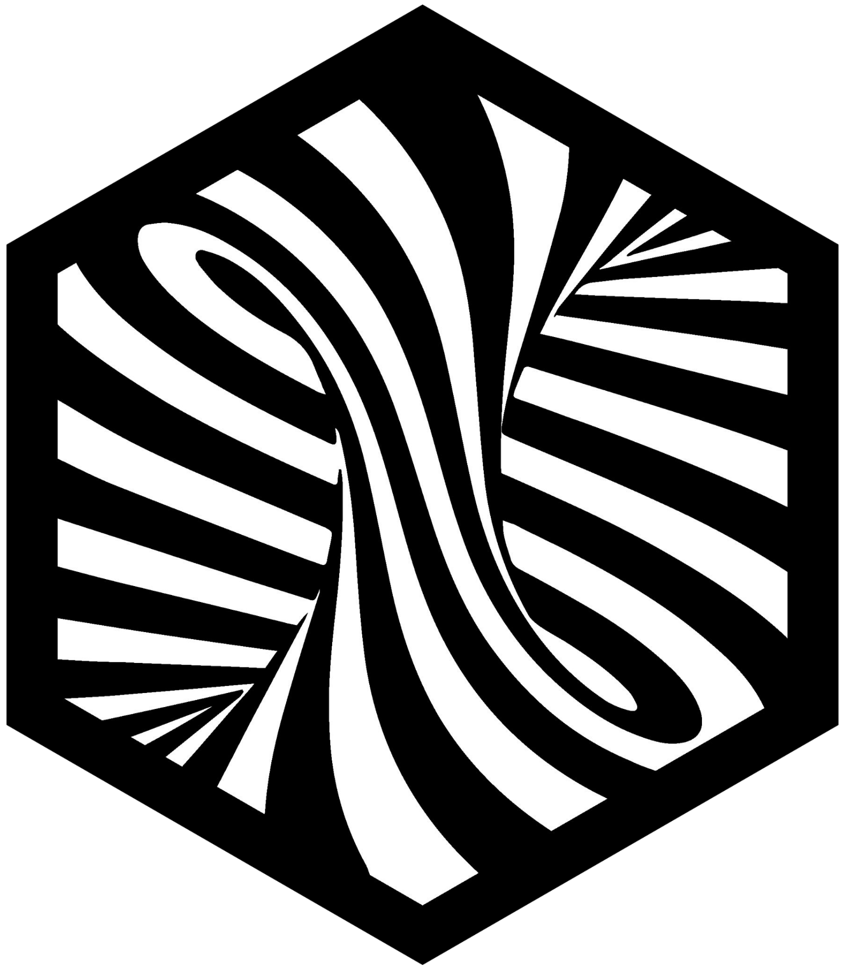
Culture Impact of DevOps

- "Shift left"
  - Developers manage pipelines
  - Developers take more control
- Focus on developer experience
  - Continued evolution of Declarative synta
  - Imperative syntax for advanced users
- Focus on interoperability
- Being open source is an advantage



Culture Impact of Inner Source

- Open Source culture gets adopted internally by the companies
- Collaboration is key
- Internal code collaboration platforms
- CI/CD tools are critical to ensure developer experience and quality



https://innersourcecommons.org/

Open Source Won!

Going alone is no longer an option!

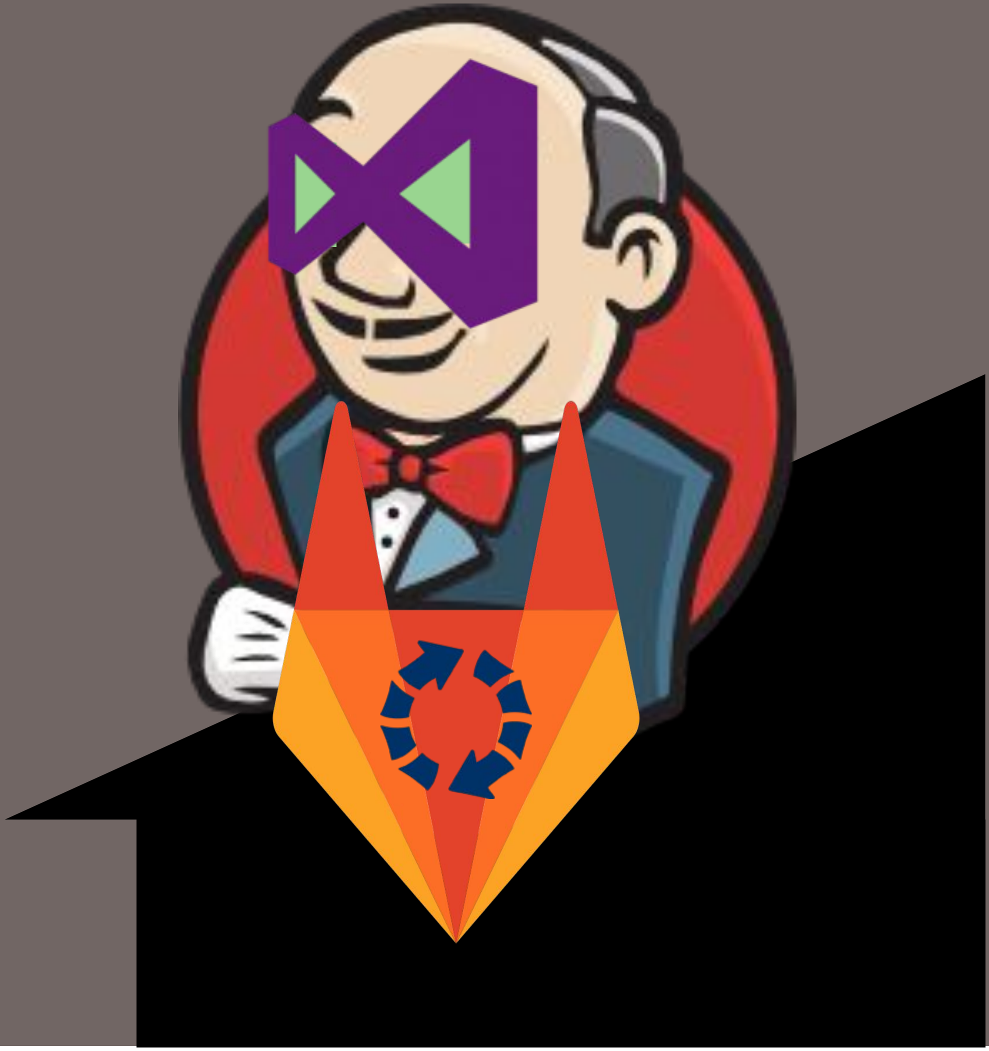
Competing against communities, partnerships and foundations? Good luck!

Source: https://www.freshboxx.in/blog/technology/free-and-open-source-software-foss-a-general-introduction

What’s next?

Oleg’s view

THIS CITY NEEDS A HERO?



~~THIS CITY NEEDS A HERO?~~ BETTER TOGETHER

Continuous
Delivery
Foundation
is here to help!

CD.FOUNDATION

A neutral home for the next generation of continuous delivery collaboration

CD.FOUNDATION

CD Foundation is an open-source community improving the world's ability to deliver software with security and speed

https://cd.foundation/

CD.FOUNDATION

Jenkins

JENKINS X

Spinnaker

TEKTON

Screwdriver.cd

ORTELIUS

SHIPWRIGHT

cdevents

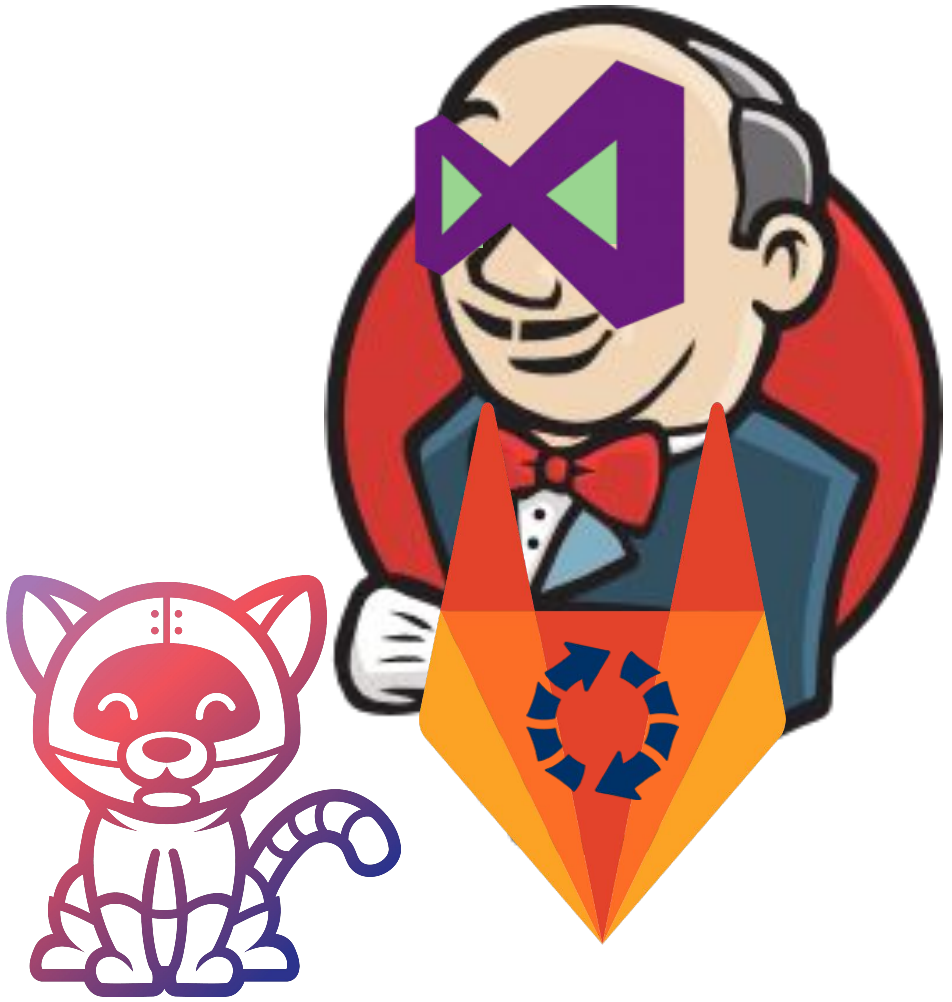
CDF Mission

- CDF believes in the power of Continuous Delivery to empower developers and teams and to produce high quality software more rapidly
- CDF believes in the open-source solutions collectively addressing the whole Software Delivery LifeCycle
- CDF fosters and sustains the ecosystem of open-source, vendor neutral projects through collaborations and interoperability
- CDF advocates this idea and encourages collaborations among practitioners to share and improve their practices

CD.FOUNDATION

CDF Interoperability SIG

- Focus on developer experience and their goals, not particular tools
- How?
  - Clarify what interoperability means for CI/CD
  - Provide a neutral forum for projects
  - Highlight and promote the needs of the users
  - Explore synergies between, and enable collaboration across, the CI/CD projects
  - Pursue solutions which are; loosely coupled, scalable, flexible, and technology agnostic
  - Reduce the need to implement in-house solutions
  - Attract/assist projects focused on interoperability



github.com/cdfoundation/sig-interoperability

CDF Events SIG

- Helping CI/CD systems to talk to each other in a standardized way
- Goal: Provide interoperability through better communication and abstraction
- How?
    - Facilitate adoption of communication standards
    - Promote adoption
    - CDEvents - new standard for CD events over CloudEvents

cloudevents.io

github.com/eiffel-community

https://github.com/cdfoundation/sig-events

The End?

...almost

Takeaways

- Automation “is eating the world”
- Continuous Integration is an industry standard
- Continuous Delivery is an industry standard
- CI and CD are not just about tools
- Tools are there, and they are evolving
- The Continuous Delivery Foundation is here to help

Get ready to the era of open source tools

- Become a good open source citizen!
- Go beyond FOSS side projects, adapt your core business
- Start an Open Source Program Office
- Adopt InnerSource internally
- Participate in foundations

What to do after the talk?

1. Take an open source CI/CD tool
2. Automate one of your projects
3. Try out another tool!
4. ...

It’s a great time to contribute ;)

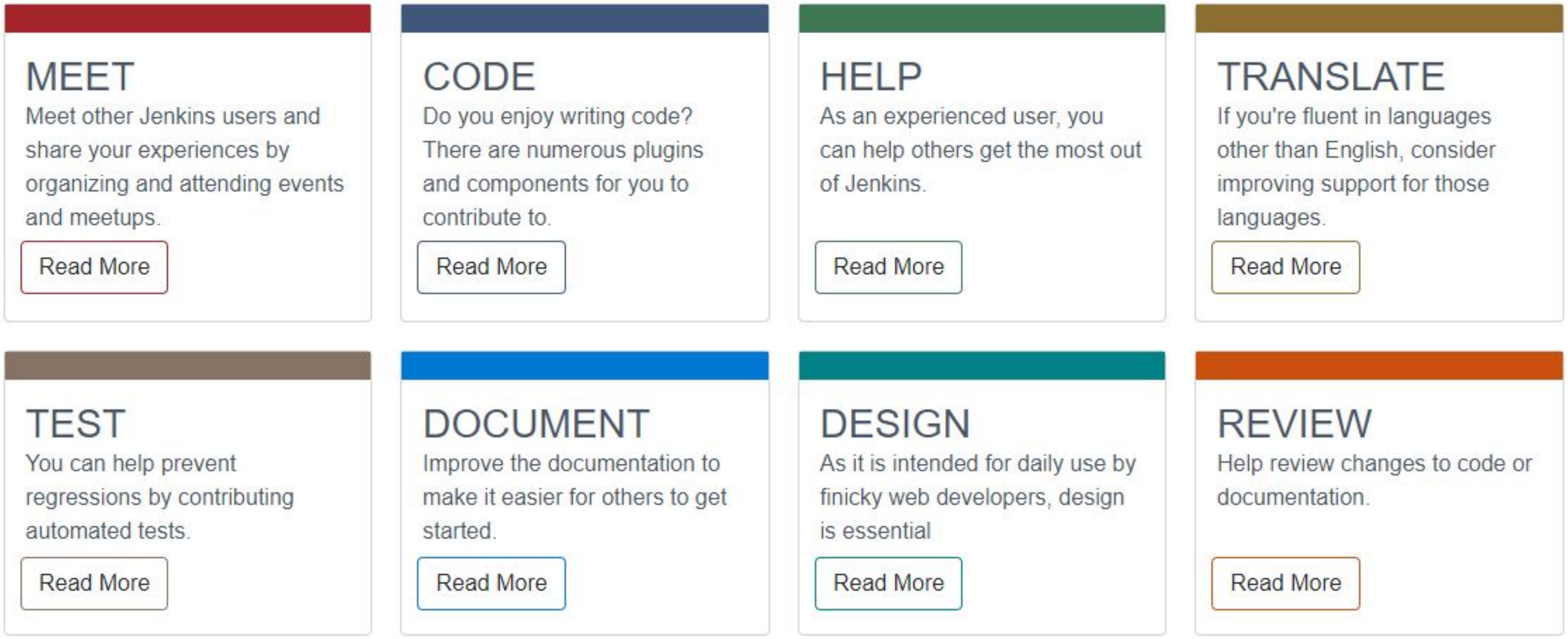
https://jenkins.io/participate

Contributing to the CDF

- Join special interest groups!
  - MLOps, Security,Events
  - Interoperability, Best Practices
- Spread the Word!
- Join Ambassador program
- Extend Landscape
- Contribute to the CI/CD projects, not only member projects

CD.FOUNDATION

Contributing to Jenkins



MEET
Meet other Jenkins users and share your experiences by organizing and attending events and meetups.
Read More

CODE
Do you enjoy writing code? There are numerous plugins and components for you to contribute to.
Read More

HELP
As an experienced user, you can help others get the most out of Jenkins.
Read More

TRANSLATE
If you're fluent in languages other than English, consider improving support for those languages.
Read More

TEST
You can help prevent regressions by contributing automated tests.
Read More

DOCUMENT
Improve the documentation to make it easier for others to get started.
Read More

DESIGN
As it is intended for daily use by finicky web developers, design is essential
Read More

REVIEW
Help review changes to code or documentation.
Read More

jenkins.io/participate

And Keptn too!

https://keptn.sh/community/contributing/

- Code
- Advocacy
- Adoption and Feedback
- Integrations
- CloudEvents Ecosystem

CD.FOUNDATION

QUESTIONS?

Contacts:

E-mail: o.v.nenashev@gmail.com

GitHub: oleg-nenashev

Twitter: @oleg_nenashev