DevSecOps

Part 1

Benjamin Hilaire
Lead Expert, Information Security
AMADEUS

Disclaimer

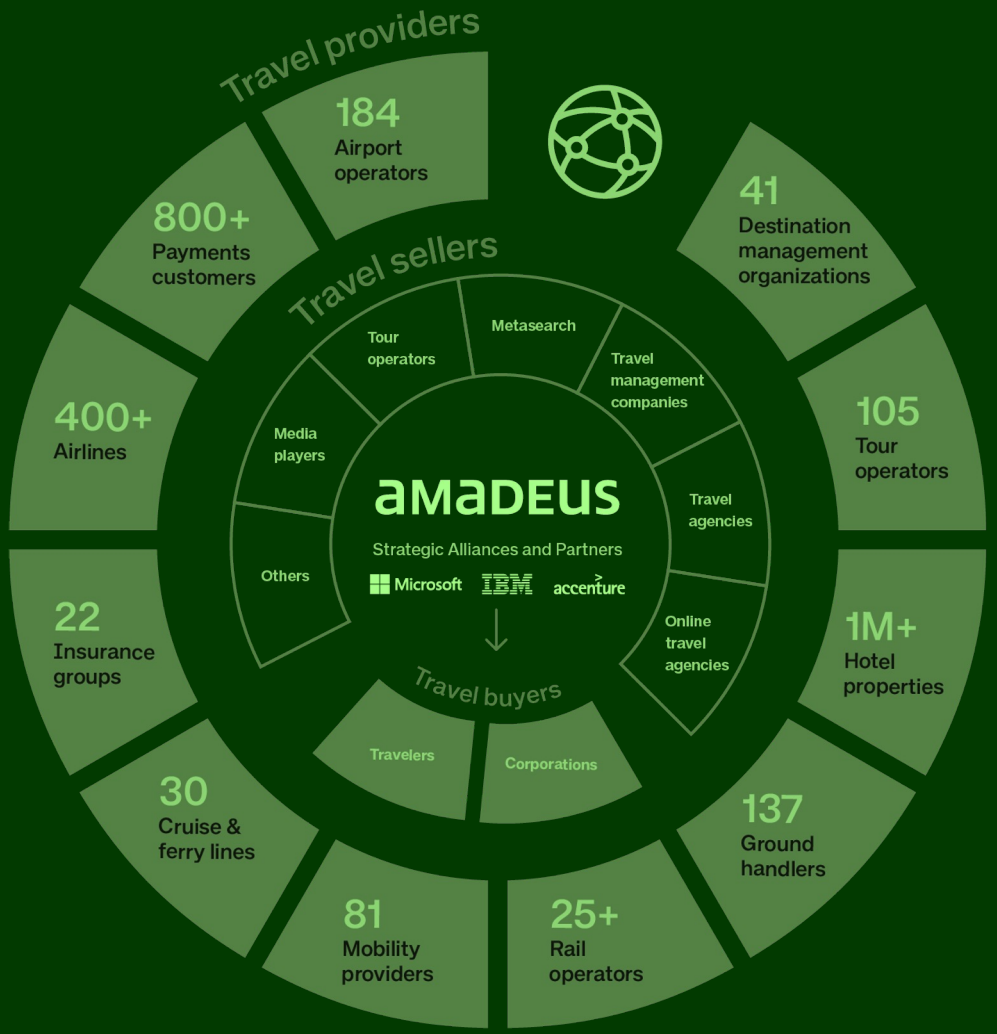
First thing first

- I'm not a teacher, I'm an engineer
  - 1st time teaching 😨
  - I don't know the usual organisation
- I love slides
  - But takes NOTES as not all in in slides
  - And I've got a very BAD handwriting, sorry
- DevSecOps is soooo wide
  - Don't expect an exhaustive tour
- I speak professional English
  - Please don't mind my bio-ti-foule French accent

Amadeus connects the travel ecosystem

4 millions
passengers
boarded per day

100 000 tps
peak inbound
traffic



More than
18,000
people worldwide

190+
countries

150+
nationalities

60+
languages

Who am i?

Benjamin Hilaire

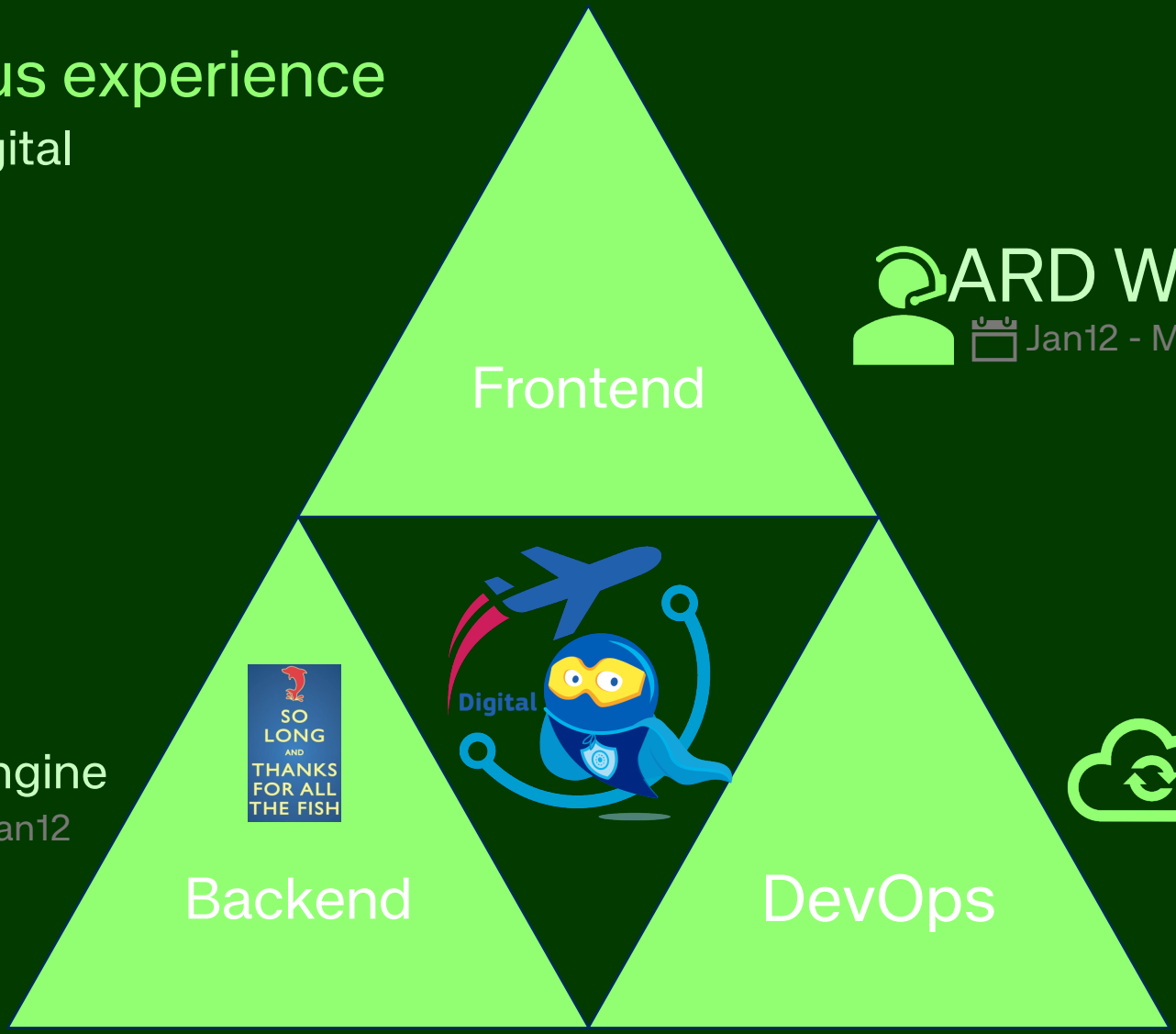
Lead Information Security Expert

Security Manager
Digital for Airlines

SDL Expert
Application Security Office

My Amadeus experience

10+ years in Digital



ARD Web

Jan12 - Mar17

Booking Engine

Aug09 - Jan12

Praxis

Mar17 – Jul19

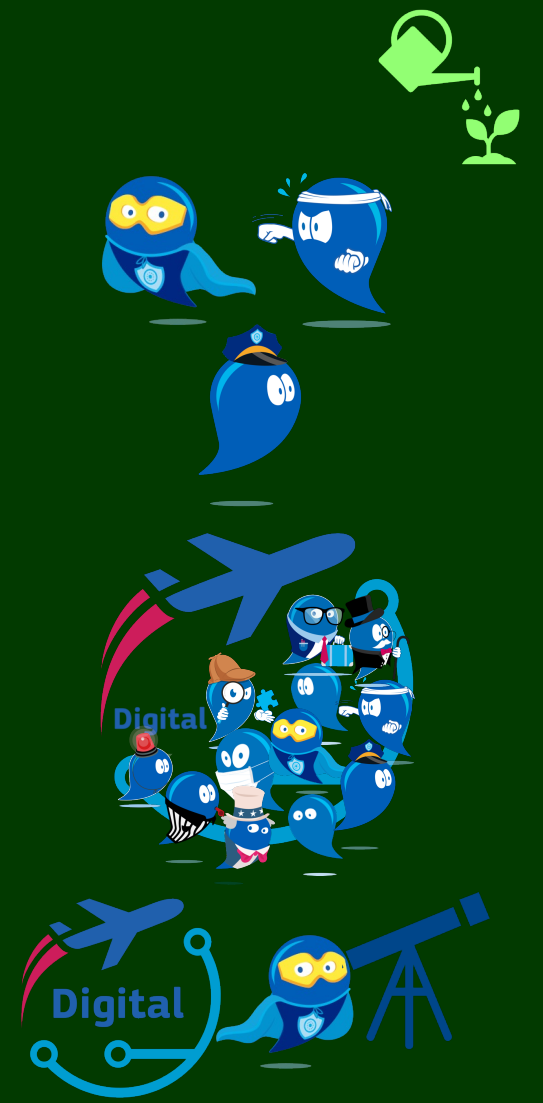
From reactive to proactive

Security maturity

Reacting to incident

Fixing our weaknesses

Security by design



Who are you ?

Quick round table
- Your name
- Your cursus
- Your dev background
- Your DevSecOps knowledge

DevSecOps

- **PART 1 – DevOps foundations**
- PART 2 – DevSecOps to secure software
- PART 3 – Securing DevSecOps
- PART 4 – Lab

amadeus

PART 1 - DevOps

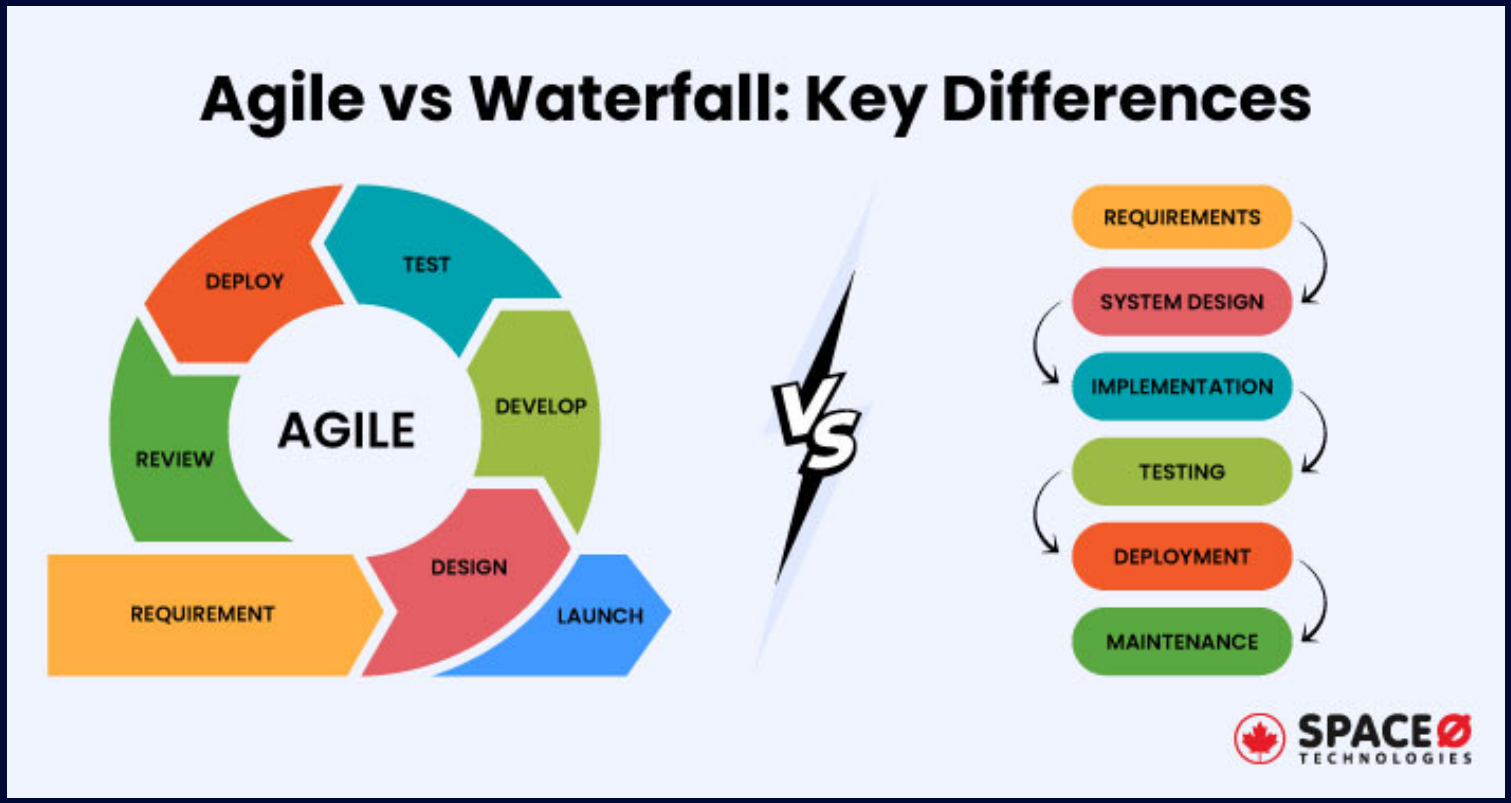
- Software Development methodologies
- Cloud revolution
- CICD technics
- Everything as Code

amadeus

1.1 Software Development methodologies

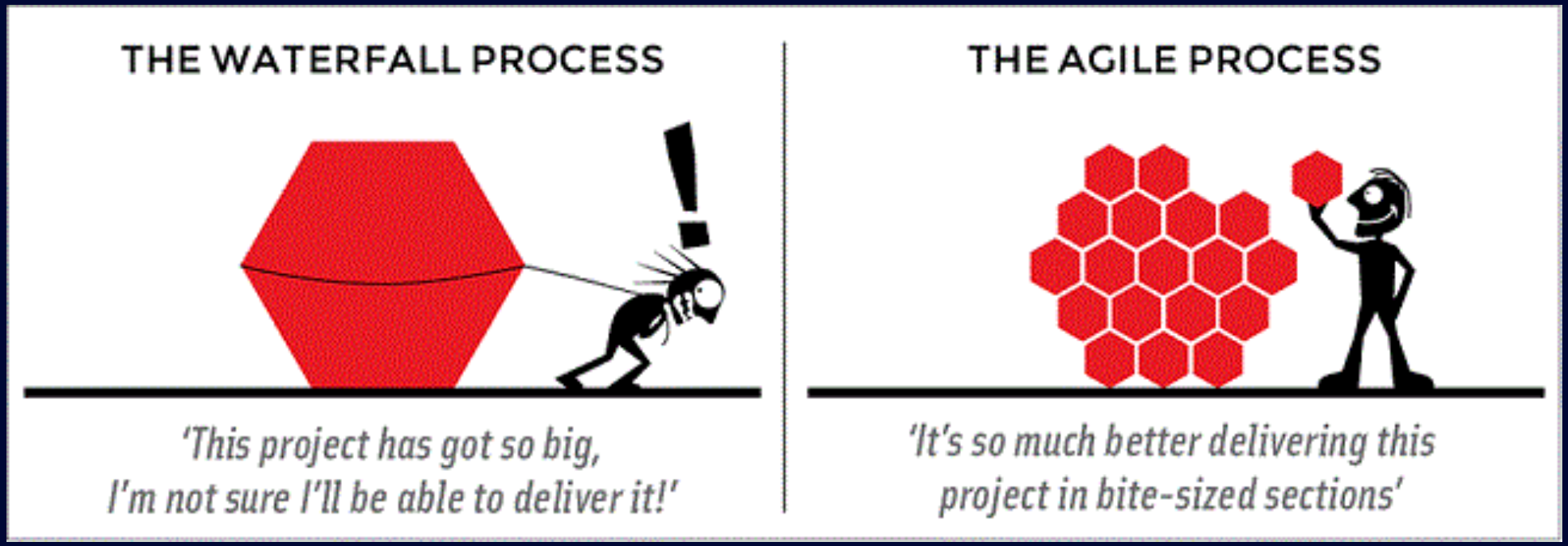
Waterfall vs Agile

Differences



Waterfall vs Agile

Goal



Waterfall vs Agile

Skills

Waterfall vs Agile

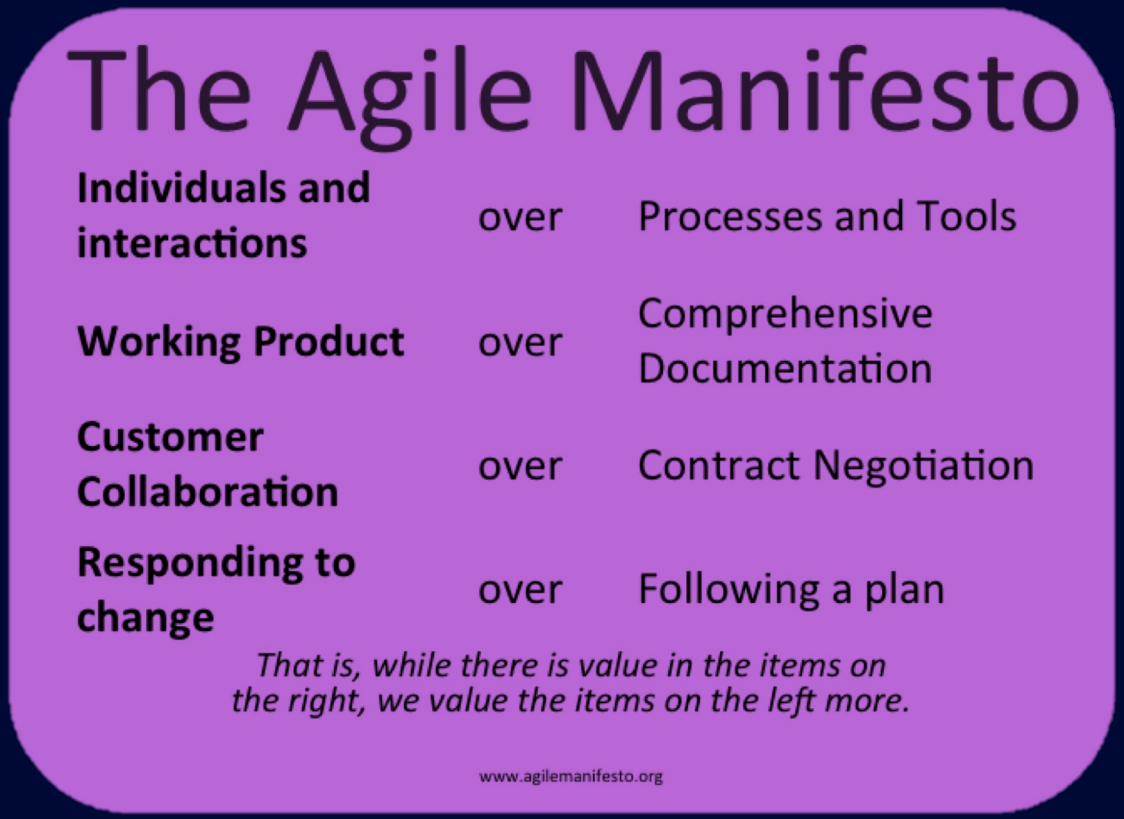
Added value

Challenge

How to release every 3 weeks ?

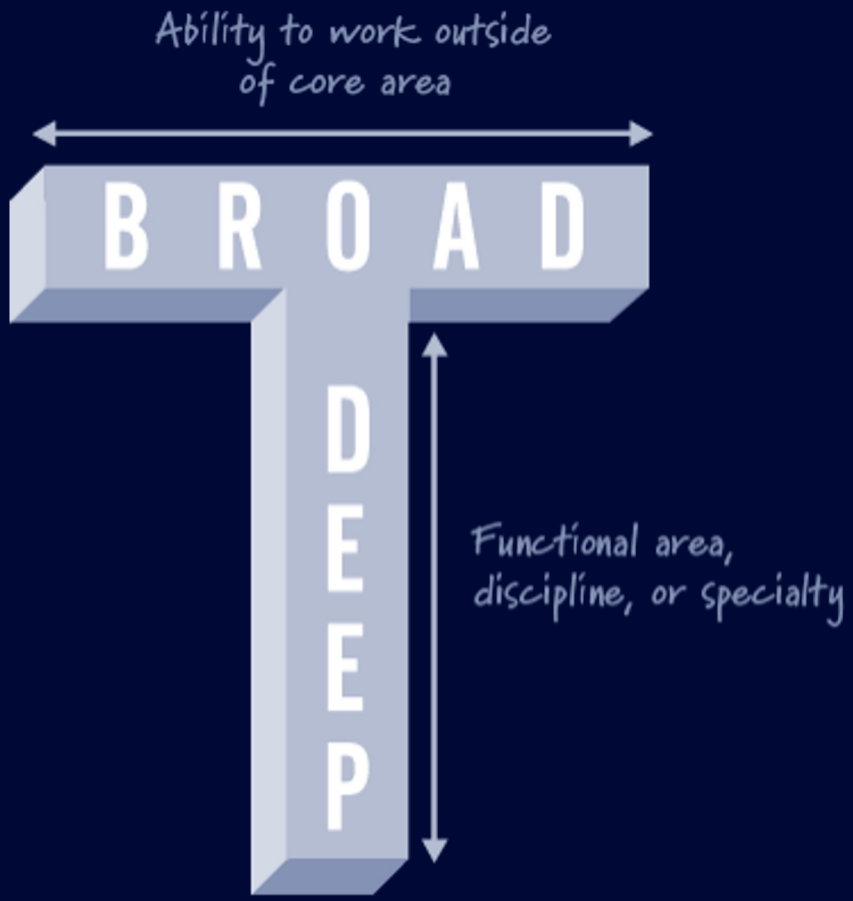
Waterfall vs Agile

Concepts



Waterfall vs Agile

Skills



Waterfall vs Agile

Why DevOps

Imagine

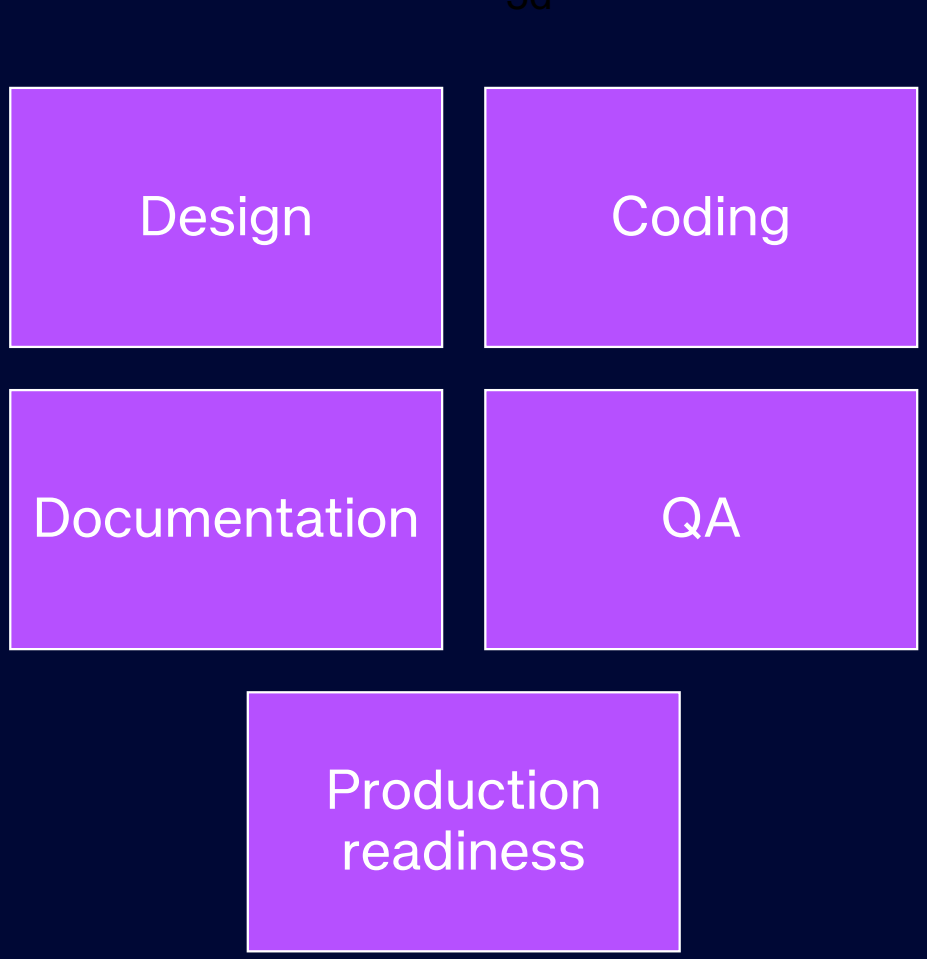
You got a complete skilled team, and you need to deliver a new simple webapp (in 2014)

Question

What are the development phases ?

Question

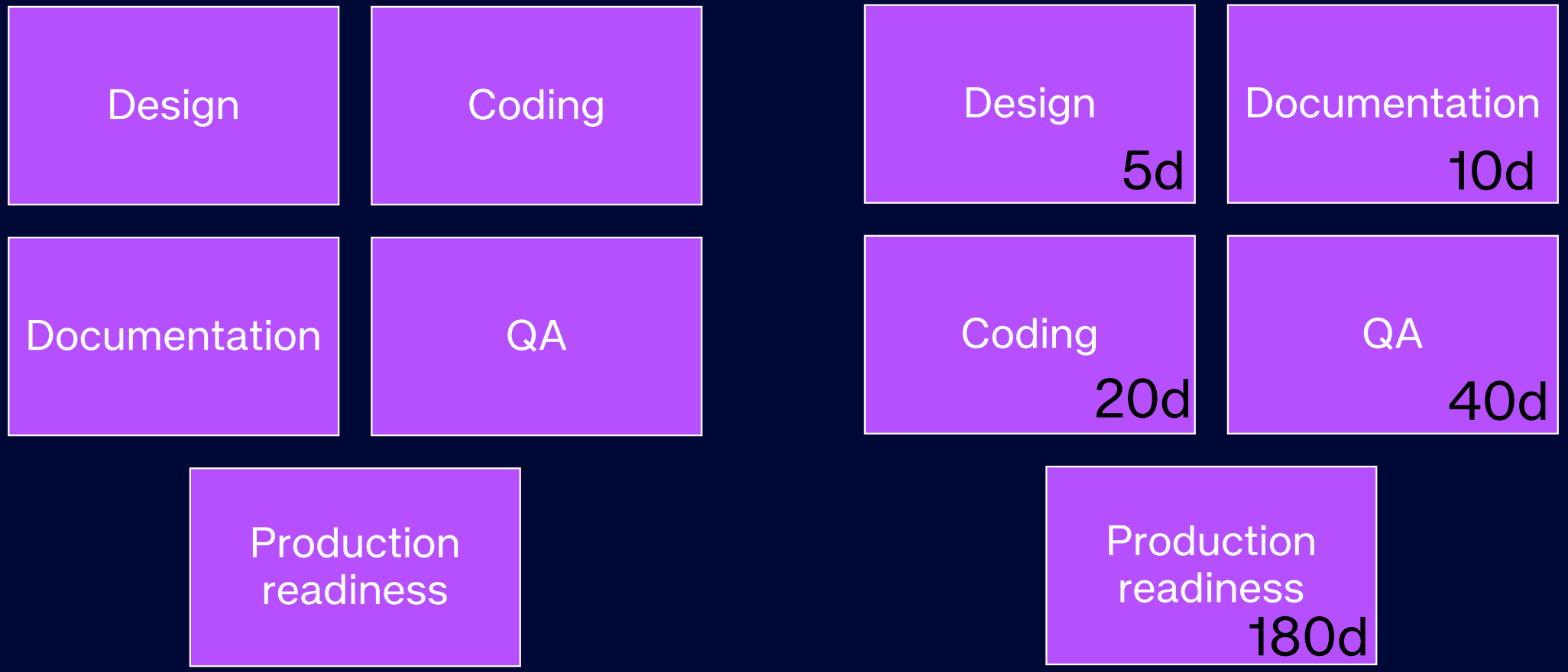
Can we sort the phase from faster to longer ?



Waterfall vs Agile

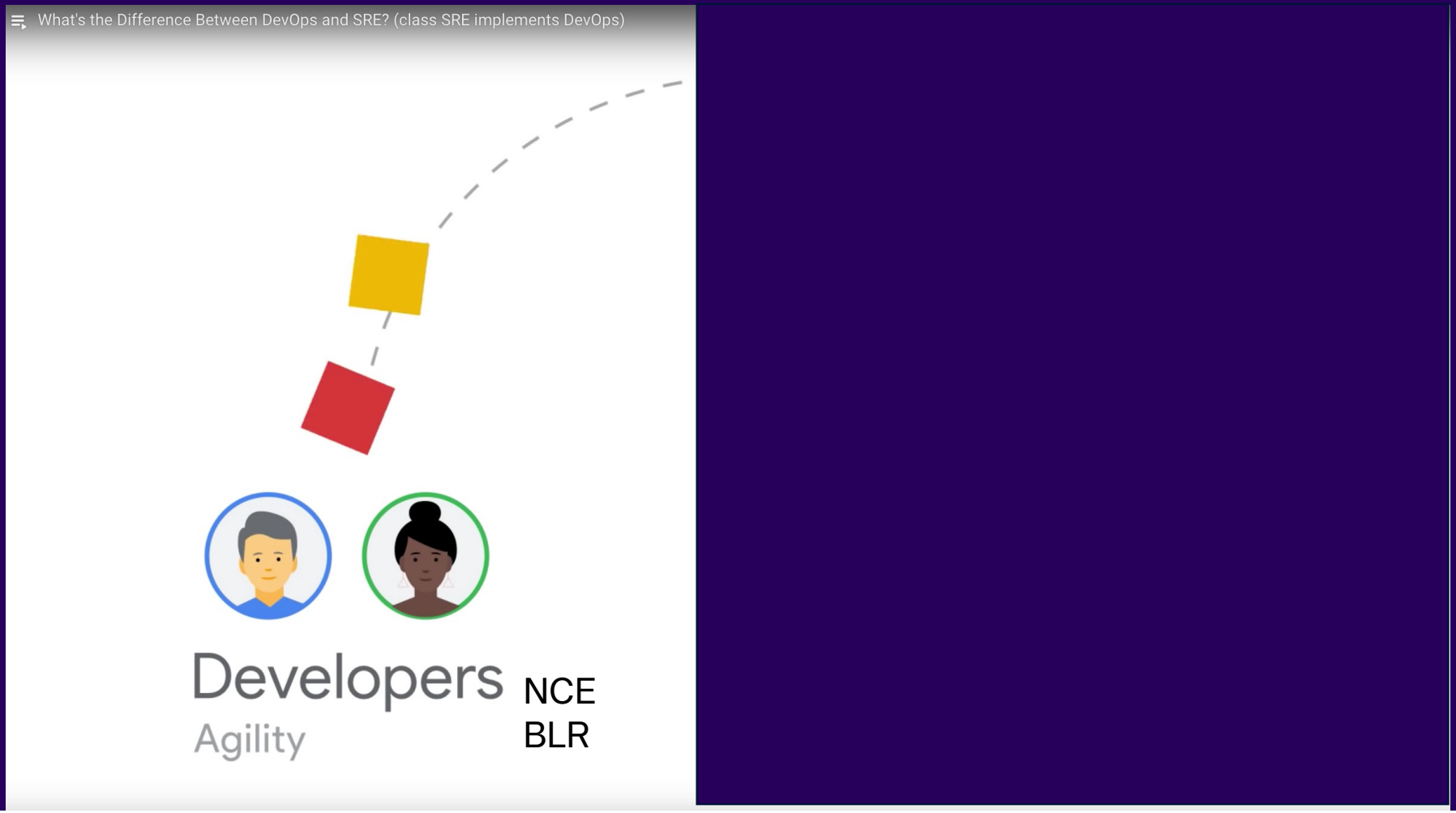
Why DevOps

5d



Why such difference ?

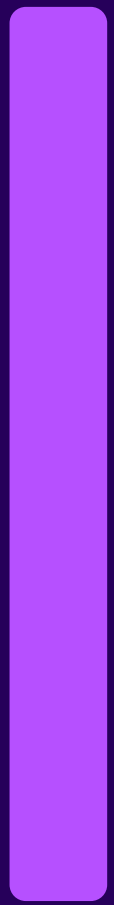
Developers throw code over the wall



Different role with different visions

_Developers

- Throw code
- No knowledge of the operations
- Responsible for feature
- Wants to move faster



•Operators

- Keep code running
- Little knowledge of the code
- Responsible for stability
- Wants to move slower

•Hard to maintain the communication and balance

Take away

Why DevOps

- **New methodology**
  - **Need agility**
  - **Reconcile Dev and Ops**
  - **Operations are too manual and takes too long**

amadeus

1.2 The Cloud revolution

My sample app

Good old times...

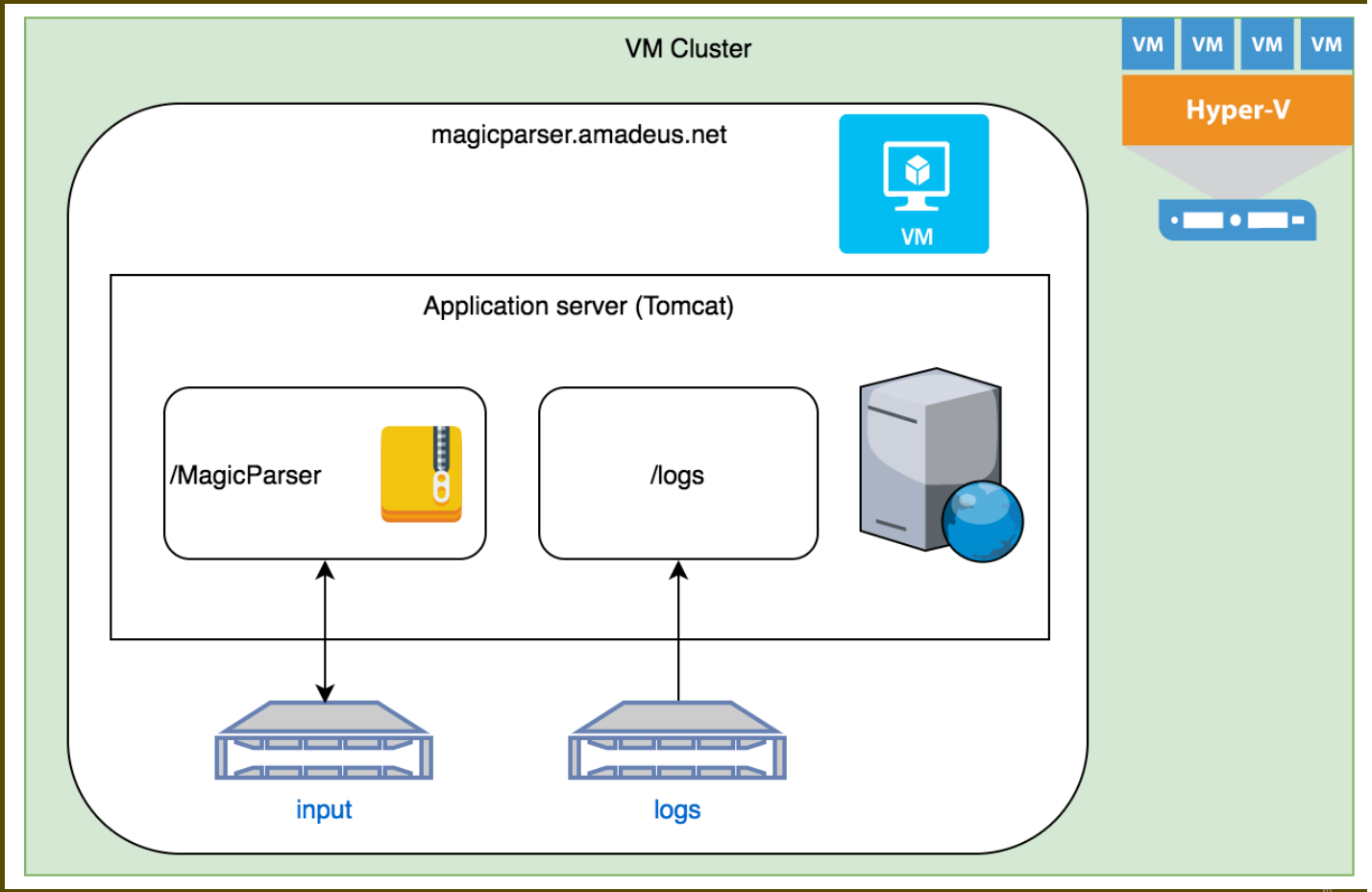
- Let's take an example
  - Tool I developped in 2012 😟
  - Java webapp + javascript (jquery)

⚠️ Presentation and screenshots from 2016

What is Magic Parser ?

Not only a war…

- magicparser.amadeus.net
  - Provided by VM team
  - Admin by us
- Tomcat 8.0
  - Installed/Config by us
- /MagicParser
  - Application war
- /logs
  - Where the HTML/JSON result is stored and served



Operational model

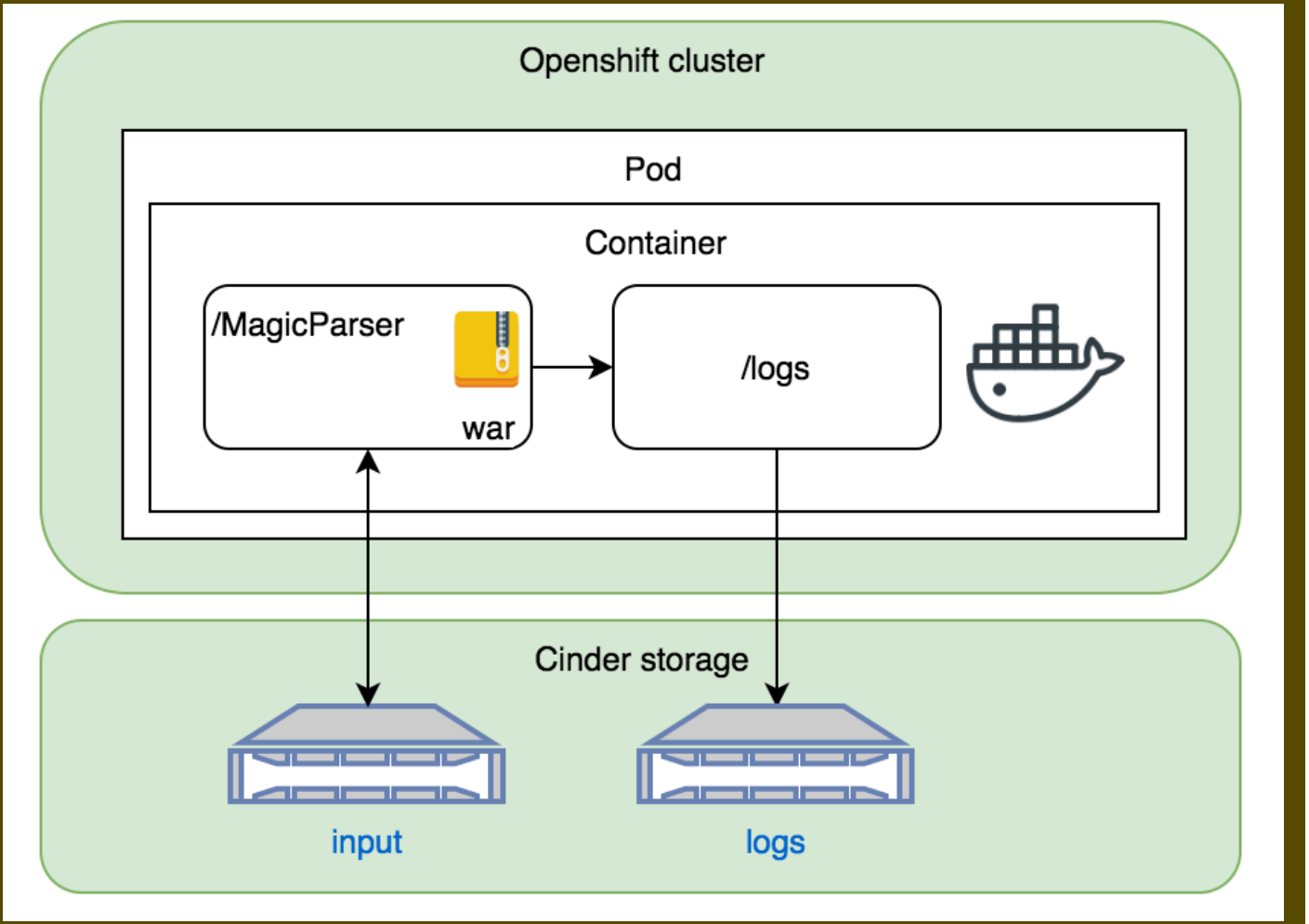
Building is not enough

- Setup
  - Initial installation of Tomcat 8.0 using unzip
  - Manual configuration of tomcat on the machine
  - Manual configuration of the appli (.properties) on the machine
- Test
  - All done in local or after deployment
- Deploy
  - Delete the war / Copy the new one
  - What if the new one is buggy ?
- Admin
  - Manual restart when the customer are not happy of the downtime
  - Apply security patch...once a year (or more)
  - No update of tomcat or java ever done

Magic Parser in Openshift (=kubernetes)

In green we trust the platform

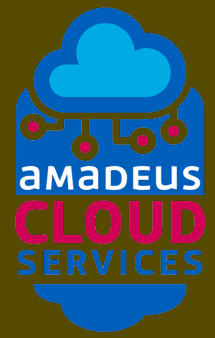
- Container
  - Docker image
- 'Pod' and storage
  - 'Blueprint' of the application deployment (json)
- Cinder storage is persistent volume provided by the platform



Operational model

Building is not enough

- Setup
  - Blueprints (json under source control) - HELM
- Test
  - Done in local and CI
- Deploy
  - One jenkins job to deploy
  - Openshift/Jenkins to fallback
- Admin
  - All is under source control
  - Automatic restart
  - Application server is the only thing to update (1 line of in Dockerfile)



How does a Dockerfile looks like ?

A container is both a VM and a process

JS uploader.js | { } magic-parser-volume-template.json | Dockerfile ● | { } du-volume.json

docker ▸ Dockerfile ▸ ...

```
1  # We inherit from a blank JBoss
2  FROM registry.access.redhat.com/jboss-eap-7/eap72-openjdk11-openshift-rhel8
3  # We play with our container to make /logs writables and target resource folder
4  ENV CONFIG_PATH /resources/
5  USER root
6  RUN mkdir -p /logs/input/temp && \
7      chgrp -R 0 /logs && \
8      chmod -R g=u /logs
9  USER jboss
10 # Copy resources folder into the container
11 COPY resources /resources
12 # Copy the war into /deployments to be deployed at runtime by JBoss
13 ADD MagicParser.war /deployments/
```

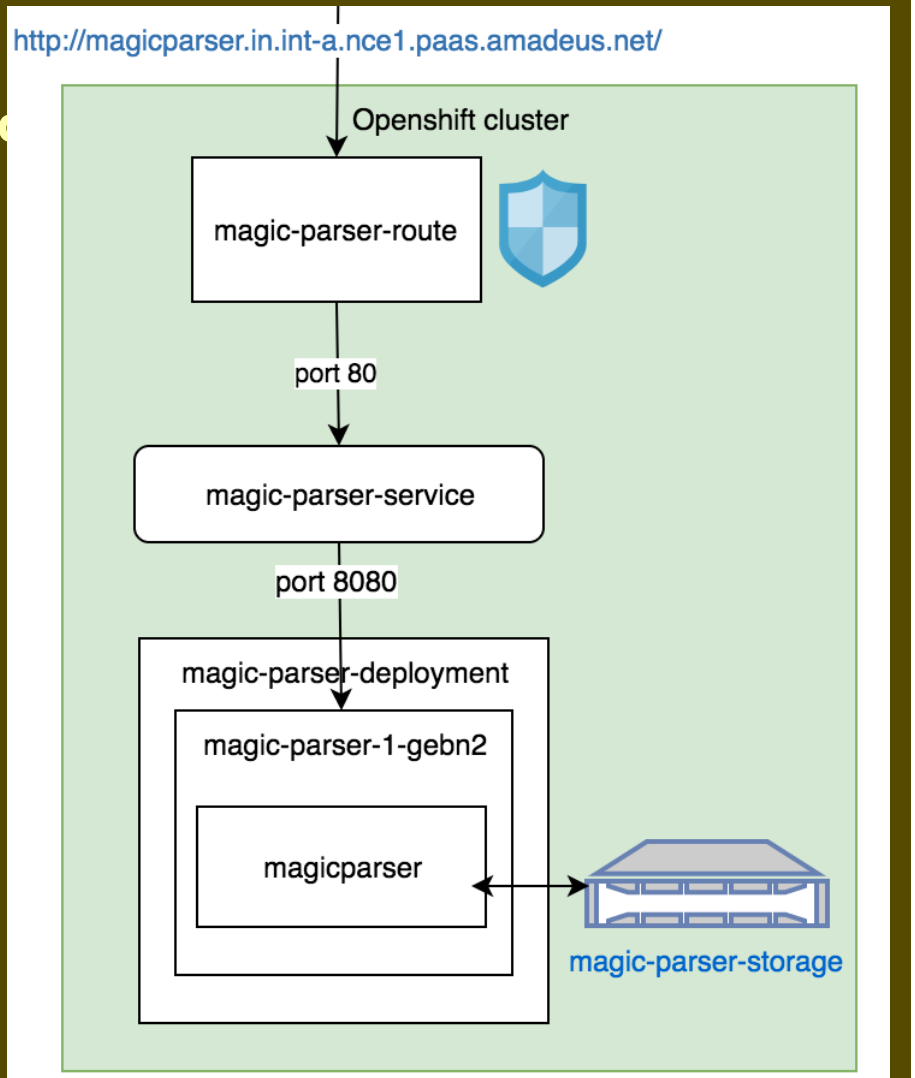
What’s missing ?

Docker != Cloud

How does a blueprint look like

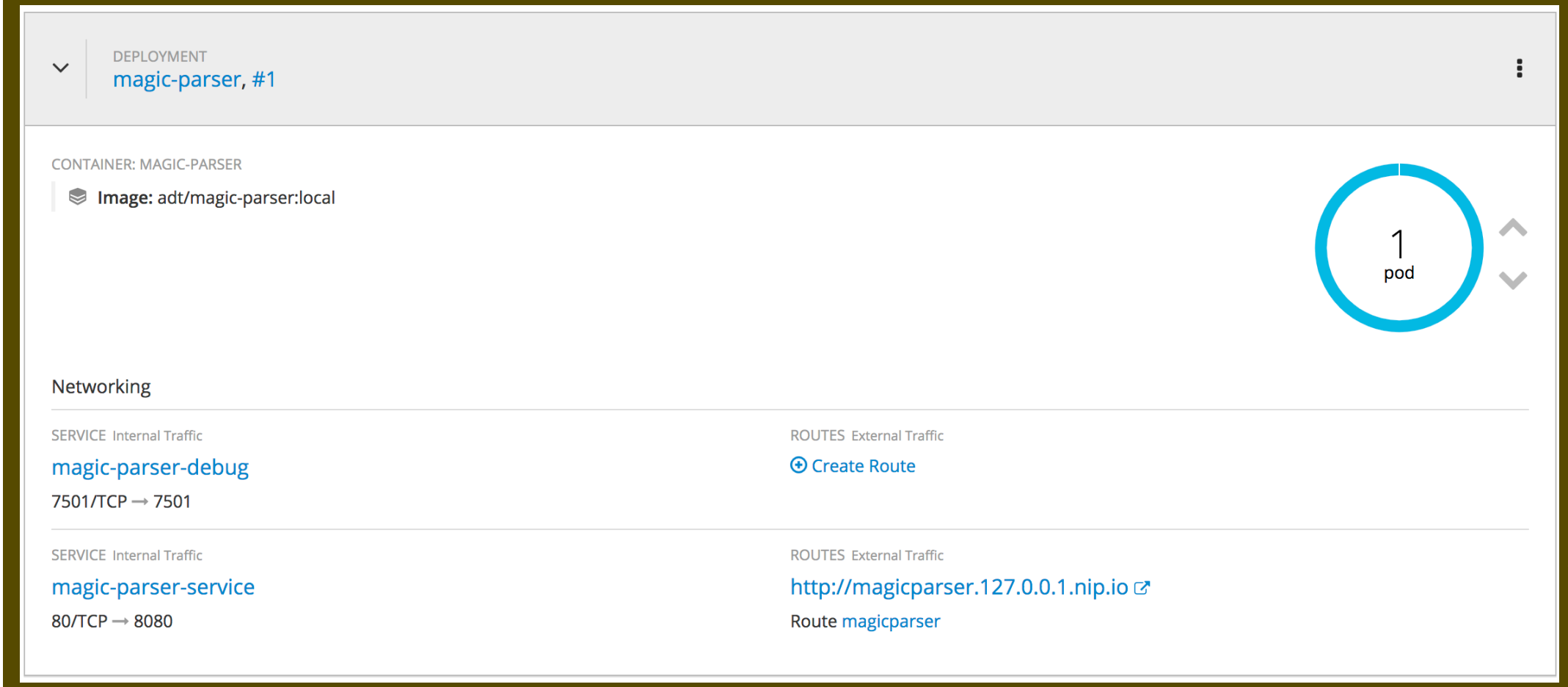
- **Container** magic-parser
  - Part of a multi-container **poc**
  - Mouting **persistent volume** ‘magic-parser’
  - Part of a **deployment**
  - Served by a **service**
  - Routed by a **route**

- All defined in an openshift **template**
  - Inject variable by environment
  - Not amadeus-specific, industry-standard



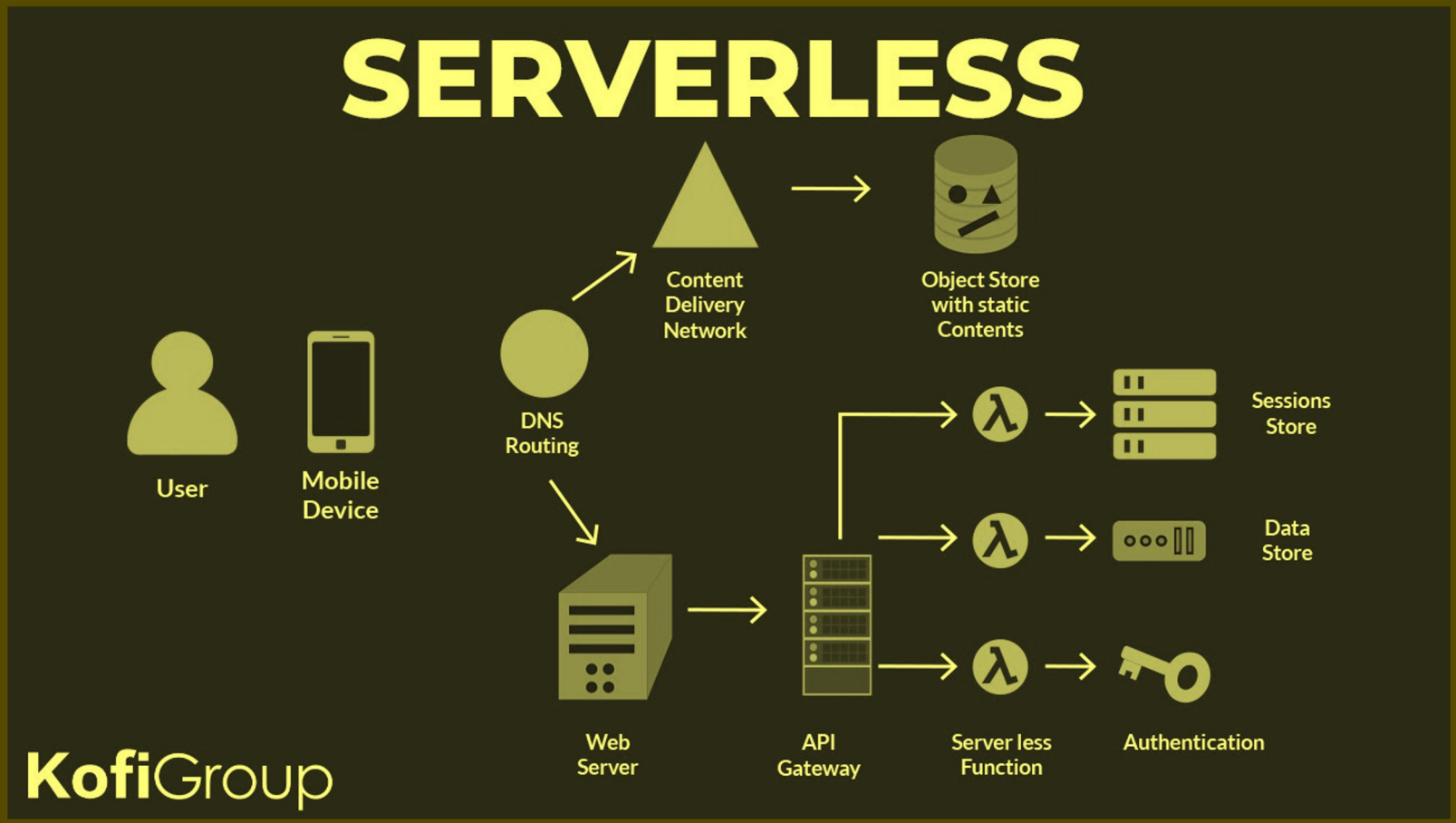
How does it look like in Openshift console

Local example



Cloud Native Applications

Going further



From Pets to Cattle

Usual image

amadeus

1.3
Continuous Integration
Continuous Delivery

amadeus

1.3. CICD

Let’s pause 30s

Copy/Pasting the war is working enough

- Why do I need CI/CD ?
  - Load in prod in 1 click
  - Fallback in 1 click
  - Don’t deploy something broken !
  - Migrate your tool to in 10 minutes
  - Enable HTTPS in 4 lines (3 in yaml)
  - Enable authentication in 10 minutes
  - Don’t spend 1 day creating an outdated operationnal doc but spent it to migrate your app

```
"tls": {
    "insecureEdgeTerminationPolicy": "Allow",
    "termination": "edge"
},
```

Use libraries and workflow

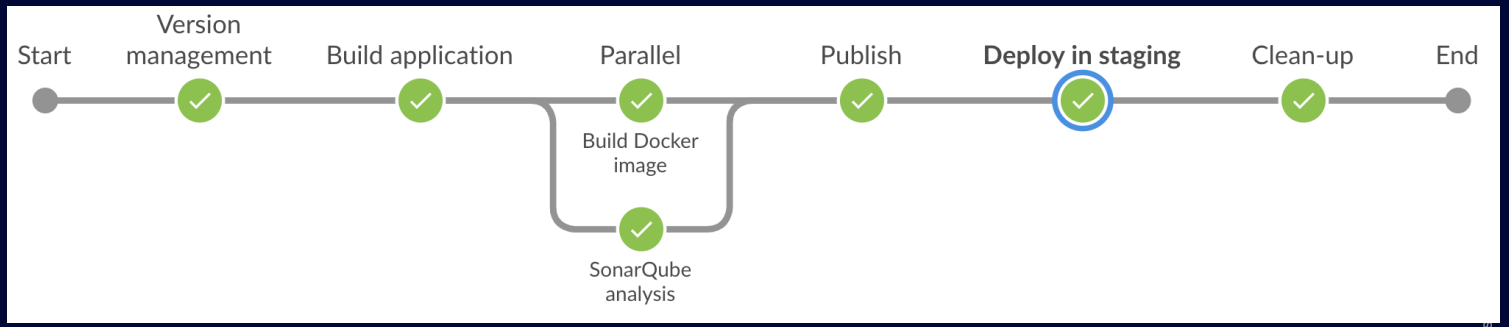
Many DEV did have the same issues as you

- Jenkins pipeline is using groovy scripts
  - Script can be shared (libraries)
  - Jenkinsfile can be shared (workflow)
  - SWB2 is providing to you the usage of both (1 workflow = 1 library)

- OpenShiftComponent library
  - Dedicated for tools CI/CD based on openshift template
  - Based on Deployment (Manager) Unit
  - Allow docker build/publication/promotion/deployment/undeployment

Tool pipeline

- Version management
  - Compute the version based on git tags and rules
- Build application
  - Mvn clean install
  - SonarQube analysis
- Build docker image
  - Docker build
- Publish
  - Publish docker image + deployment units + blueprint + env properties
- Deploy
  - Deploy your application using the blueprint on the target



Let's break it down : BUILD

Already atomic...

_Version management

- Use 'newVersion' to automatically compute the version based on git tags
- Create 'OpenshiftComponent'
- Add token for connection to openshift

```
execStage('Version management'){
  version = newVersion(['versionType': 'sem', 'branchingModel': ['master': '@{VERSION}-master', '.*'
  currentBuild.description = version
  oseComponent = new OpenShiftComponent('adt-magicparser',version,this)
  oseComponent.addTokenParameter('openShiftToken','magicparser-sa')
  oseComponent.addParameter('uid',oseComponent.getContinuousIntegrationName())
}
```

•Build

```
execStage('Build application'){
  sh 'mvn -B clean verify package -DskipStaging=true -DskipTests'
  sh 'cp viewer/target/MagicParser.war docker'
}
```

_Sonarqube

```
'SonarQube analysis': {
    sonarRunner([projectName: 'MagicParser', projectKey: 'com.amadeus.jcp.viewer.log:MagicParser', sonar_nature: 'mvn'])
},
```

•Docker build

```
'Build Docker image': {
    oseComponent.build("adt/magic-parser:${version}", 'docker')
}
```

Let's break it down : PUBLISH and DEPLOY

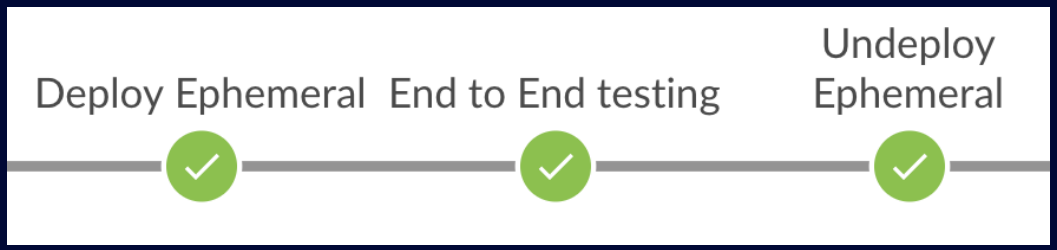
Already atomic...

_Publication reference

- docker
- deployment unit
- openshift template
- environment properties

```
execStage("Publish"){
  oseComponent.setDeploymentUnit('blueprints/du-novolume.json')
  oseComponent.setOpenshiftTemplate('blueprints/magic-parser-no-volume-template.json')
  // Here we put the additional du and template to be published
  oseComponent.setEnvironmentProperties('blueprints/staging.properties','blueprints/prod.properties',
  'blueprints/ephemeral.properties','blueprints/du-volume.json','blueprints/magic-parser-volume-template.json')
  oseComponent.publish()
}
```

- Validation
  - Not done for MP (see watchtower)



- Deployment

```
// Continuous deployment for master branch only
if (env.BRANCH_NAME.equals('master')){
  execStage('Deploy in staging'){
    oseComponent.deployInTarget('staging','magicparser=staging')
  }
}
```

How can I load on prod ?

Power of libraries

deploy-prod.groovy

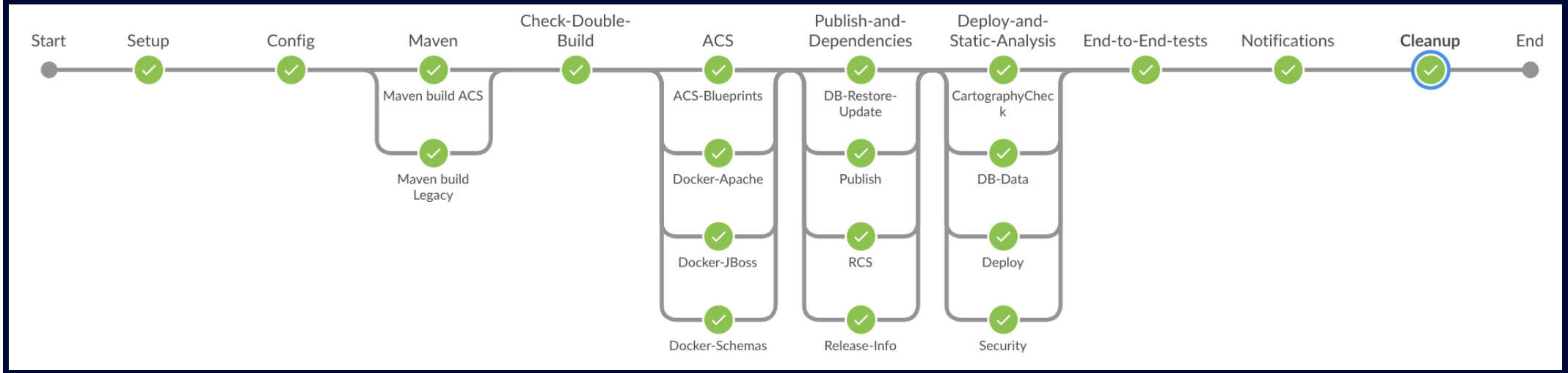
```groovy
#!/usr/bin/groovy
@Library(['pipeline-global-libraries', 'pipelinelibraries@16.0.36']) _

import com.amadeus.air.jenkins.openshift.OpenShiftComponent

execEnv('dockerhub.rnd.amadeus.net:5000/acs_praxis/jenkins-slave:3.3.0', 20) {
  // Prepare openshift component
  execStage("Set-up"){
    oseComponent = new OpenShiftComponent('adt-magicparser',env.VERSION,this)
    currentBuild.description = env.VERSION
    oseComponent.addTokenParameter('openShiftToken','magicparser-sa')
    // Download files needed for deployment
    oseComponent.setPublishedDeploymentUnit('du-volume.json')
    oseComponent.setPublishedOpenshiftTemplate('magic-parser-volume-template.json')
    oseComponent.setPublishedEnvironmentProperties('prod.properties')
  }
  execStage("Promote"){
    // Promote the file in order to be kept for a long time
    oseComponent.promote()
  }
  execStage("Load"){
    // Deploy in prod target using DeploymentManager
    oseComponent.deployInTarget('prod',"magicparser=prod")
  }
}
```

- Reuse the same library in the same spirit
- Every data is published and ready to be used
- Promotion done to make sure the artifact are kept a long time
- Load done using deployment manager and not via oc command

Real (and oudated) example



Key takeaways

- Scripts
  - BUILD (CI)
  - DEPLOY
  - TEST
  - VALIDATE and PROMOTE
  - LOAD in PRD (CD)
- Evolution of
  - MAKE, ANT, MAVEN, BASH,...

Old school CICD

(outdated)

amadeus

1.3.2 Modern CICD

Key takeaways

Stop scripting all CI

How can we improve ?

Modern CI

Stop scripting all CI

- CICD scripts are usually COPY/PASTE
  - Uses library to factorize code
  - Building 1 app or another is mostly the SAME *but* the name of the app (and few more things)
- What if we split the action (what we do, actual code) with the model (parameters we inject) ?

Modern CI

Enhanced copy/paste

- Workflows
  - Framework with variable
  - Actions re-usable
  - Action configurable
- Easy
  - Usage
  - Understanding
  - Maintenance

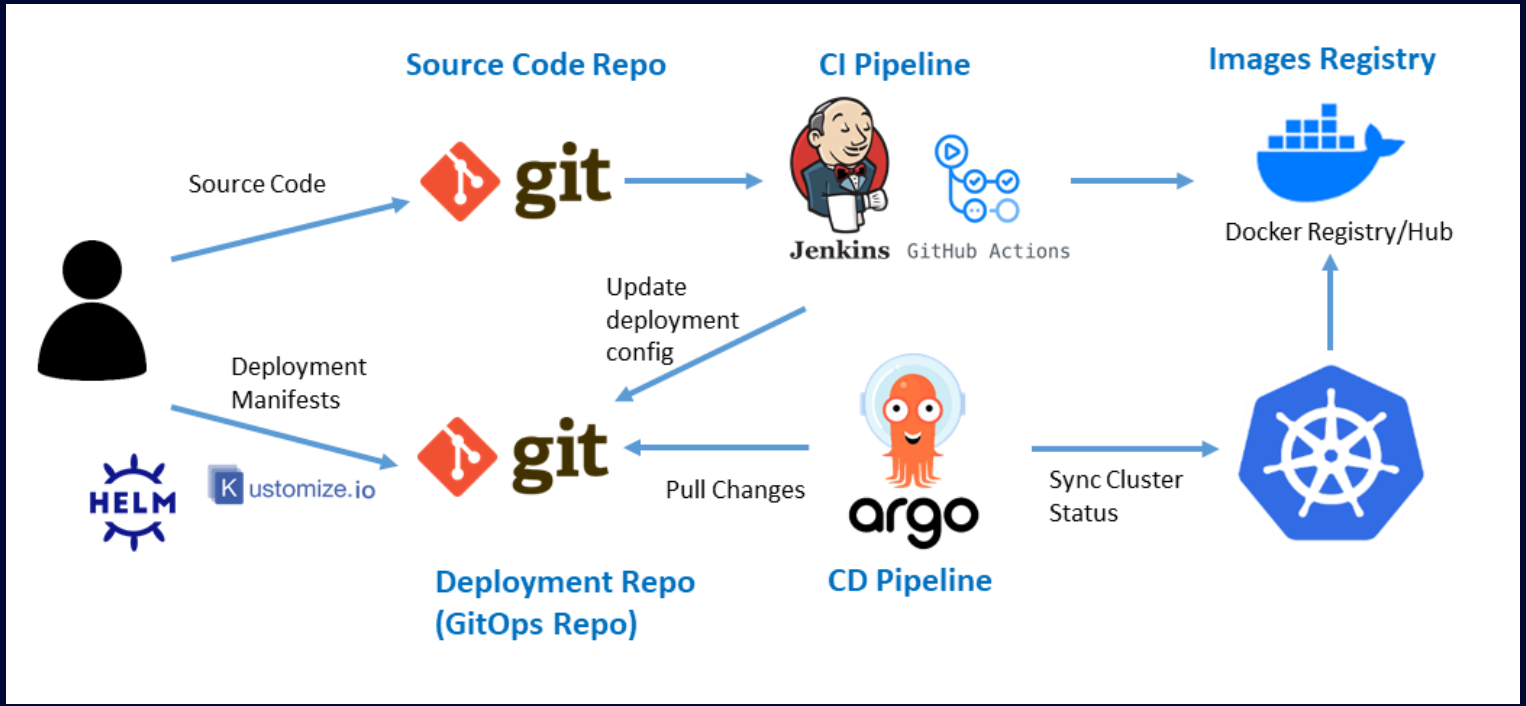
YAML

```yaml
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{
github.event_name }} event."
      - run: echo "🐧 This job is now running on a ${{ runner.os }} server
hosted by GitHub!"
      - run: echo "🔎 The name of your branch is ${{ github.ref }} and your
repository is ${{ github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v4
      - run: echo "💡 The ${{ github.repository }} repository has been cloned to
the runner."
      - run: echo "🖥️ The workflow is now ready to test your code on the
runner."
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🍏 This job's status is ${{ job.status }}."
```

Modern CD

Stop scripting the CD

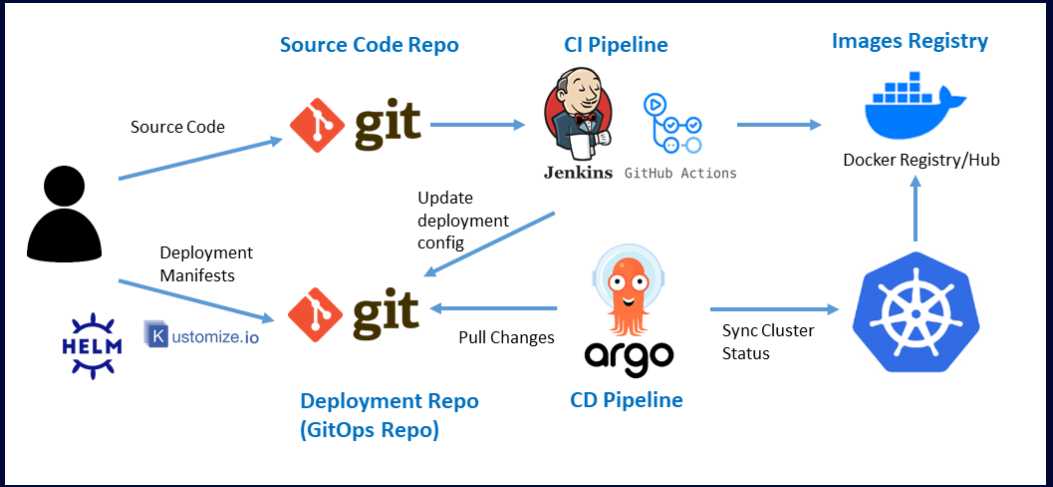
- Deploying is applying a blueprint
- Split SOURCE and BLUEPRINT
- Workflow builds artifact
- ArgoCD watches GIT and applies chart (blueprint)



Modern CD

Stop scripting the CD

Any idea why ArgoCD is much better than before ?



ArgoCD added value

Stop scripting all CI

- What if the containers take 61s to deploy (but timeout is 60s) ?
- What if an operator modify manually the value and forget to put it back ?
- Blueprint GIT must be the reference

Modern CICD

Key takeaway

- Split CI CODE and CI definition (YAML)
- Split CD from CI
- Let ArgoCD watch GIT and provide self-repair

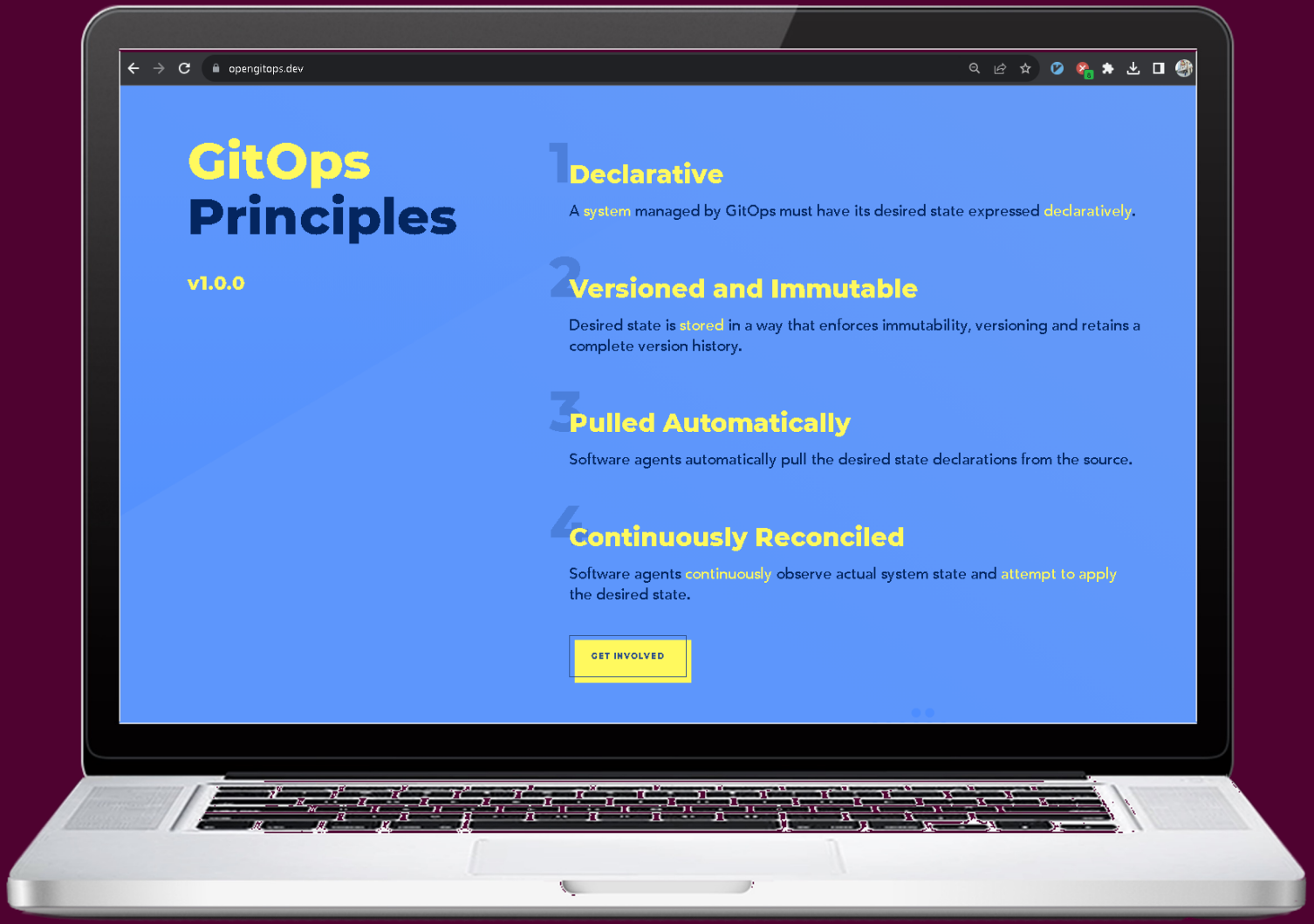
amadeus

1.4 Everything as Code

GitOps

- GitOps
  - Split code and configuration
  - Developers (or 3rd party) are responsible of code
  - DevOps of their configuration
- Git advantages
  - Same tool as developers
    - Review process
    - Merge policies
    - Automated code validation
  - Full history and blame
  - Revert in one click
  - 1 PR for a full migration

GitOps



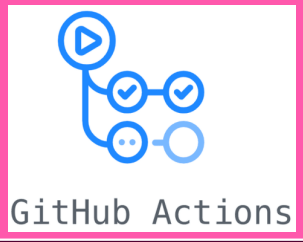
amadeus

GitOps => Everything as Code

Any famous example ?

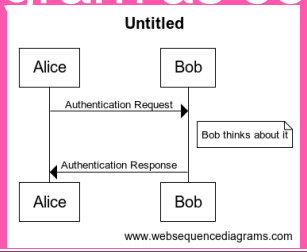
GitOps => Everything as Code

Pipeline as code



Infra as code

Diagram as code



Network as code

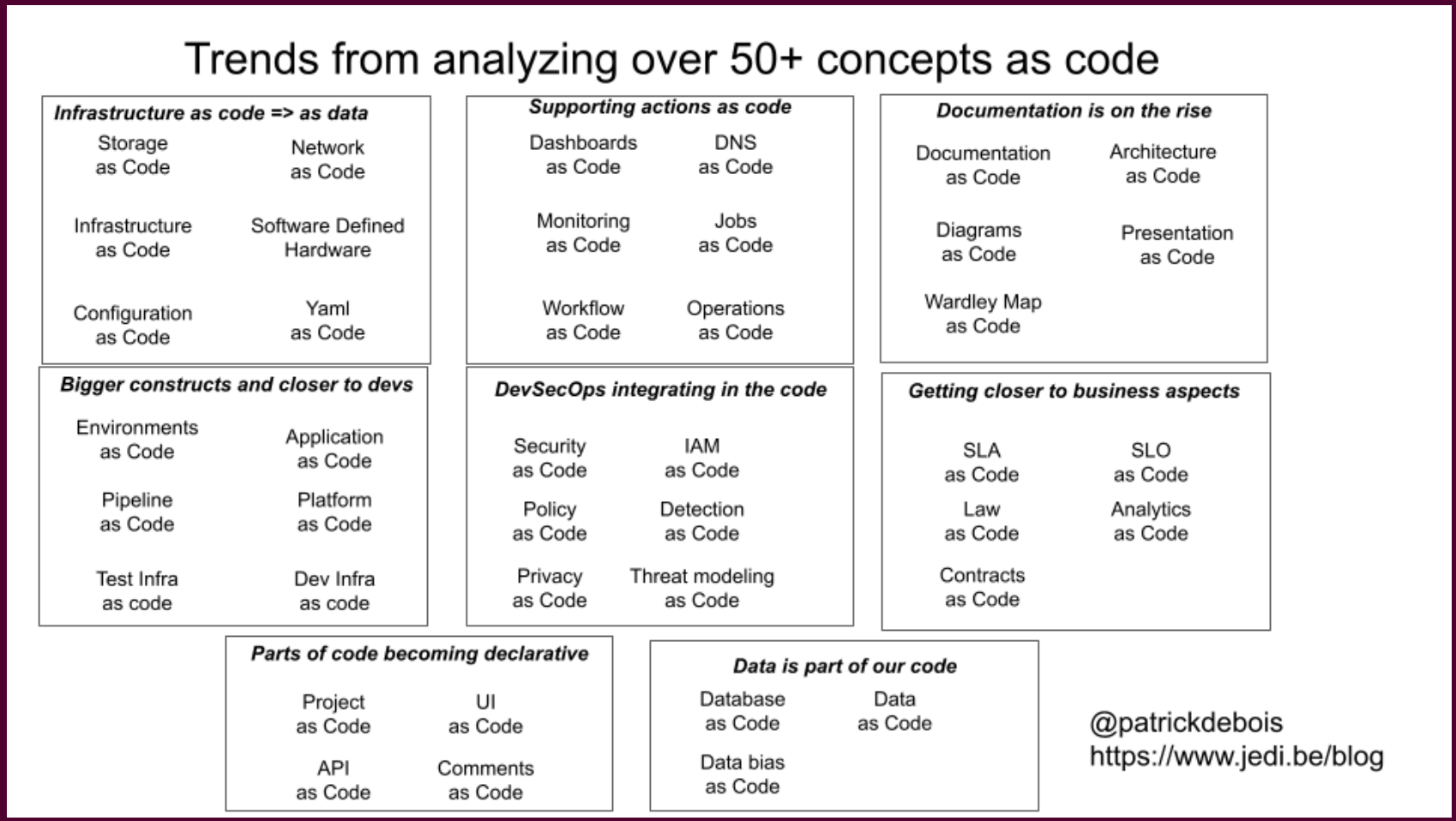
Dashboard as code

Grafana

...

amadeus

A piece of the as-code realm



Key takeaways

- GitOps
  - Backbone of DevOps
  - Allow team collaboration
  - Allow traceability
  - Allow repairing capability
- Now Everything as Code

Take away

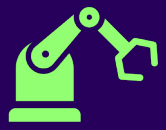
DevOps in 1 slide (don't miss it)

New methodology

- **Need agility**
- **Operations are too manual and takes too long**

Cloud revolution

- **From Pets to Cattle**
- **Binary can run everywhere (container)**
- **Topology is defined and can be cloned**



CICD

- **Continuous Integration**
  - **Build + Validate**
- **Continuous Deployment**
  - **Deploy based on topology**

GitOps

- **Git the is reference for all operation**
- **Config split from code**
- **Evolution to everything as code**

amadeus

Thank you