

GDC CLOUD SCHULUNG PRACTICE DEVOPS

Bernd Rederlechner, 26.10.2017

T - Systems -

AGENDA

01 Intro: Continuous Delivery Pipeline revisited

02 Reproducibility: Everything as Code

02.1 Exercise: Trigger pipeline (Jenkins from GIT)

03 Provisioning time: Infrastructure as Code

03.1 Exercise: Automated install and configuration

04 Dev-Prod-parity: Immutable Infrastructures

04.1 Exercise/Video: Image bakery

05 Conways law and continuous feedback

05.1 Exercise: ELK logging as microservices

06 Learn: Red/black, A/B and canaries

06.1 Exercise: My little A/B test

PATH 1:

Optimize flow

PATH 2:

Include feedback

PATH 3:

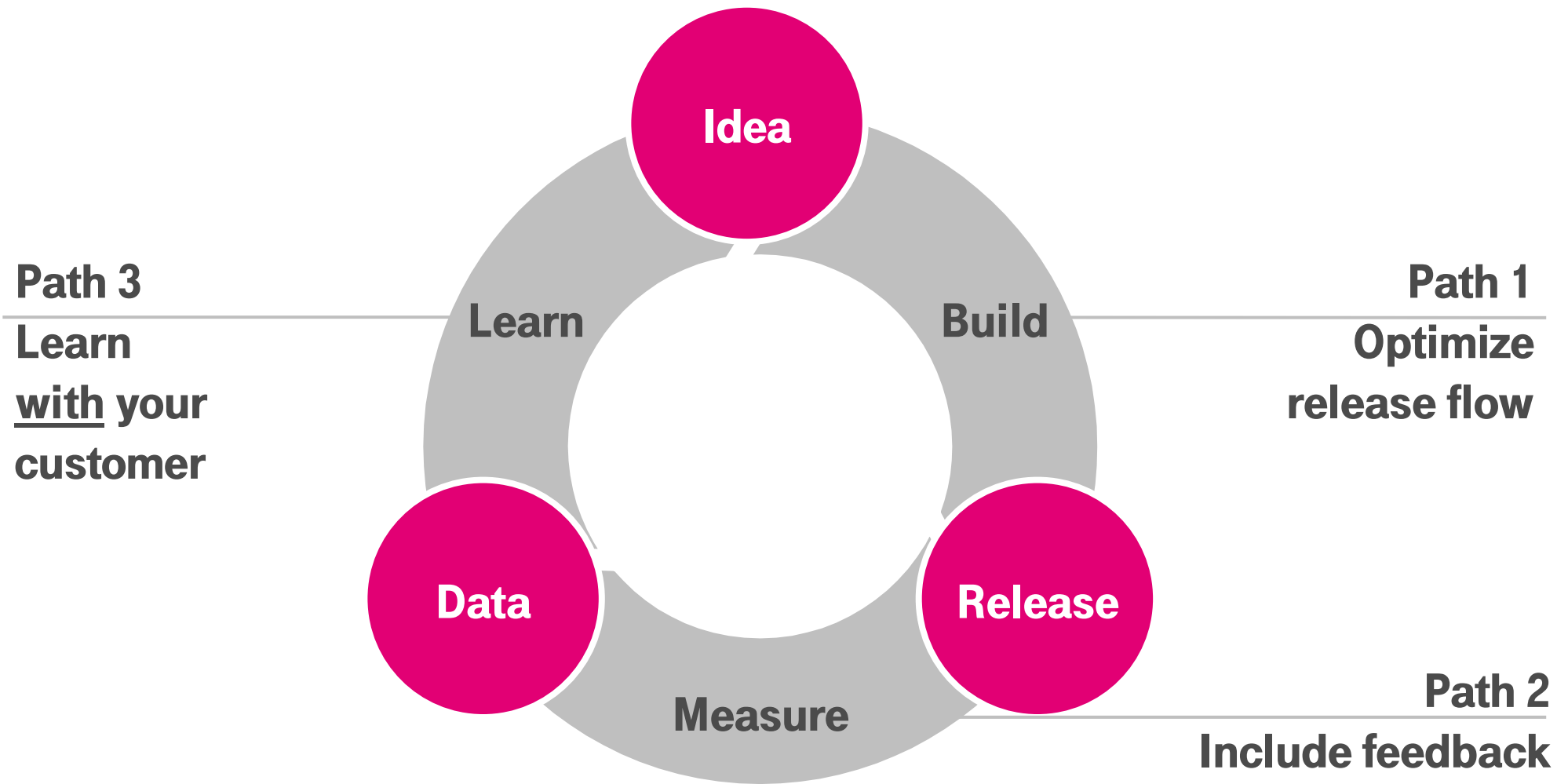
Learn from system

HINWEIS ZUM COPYRIGHT

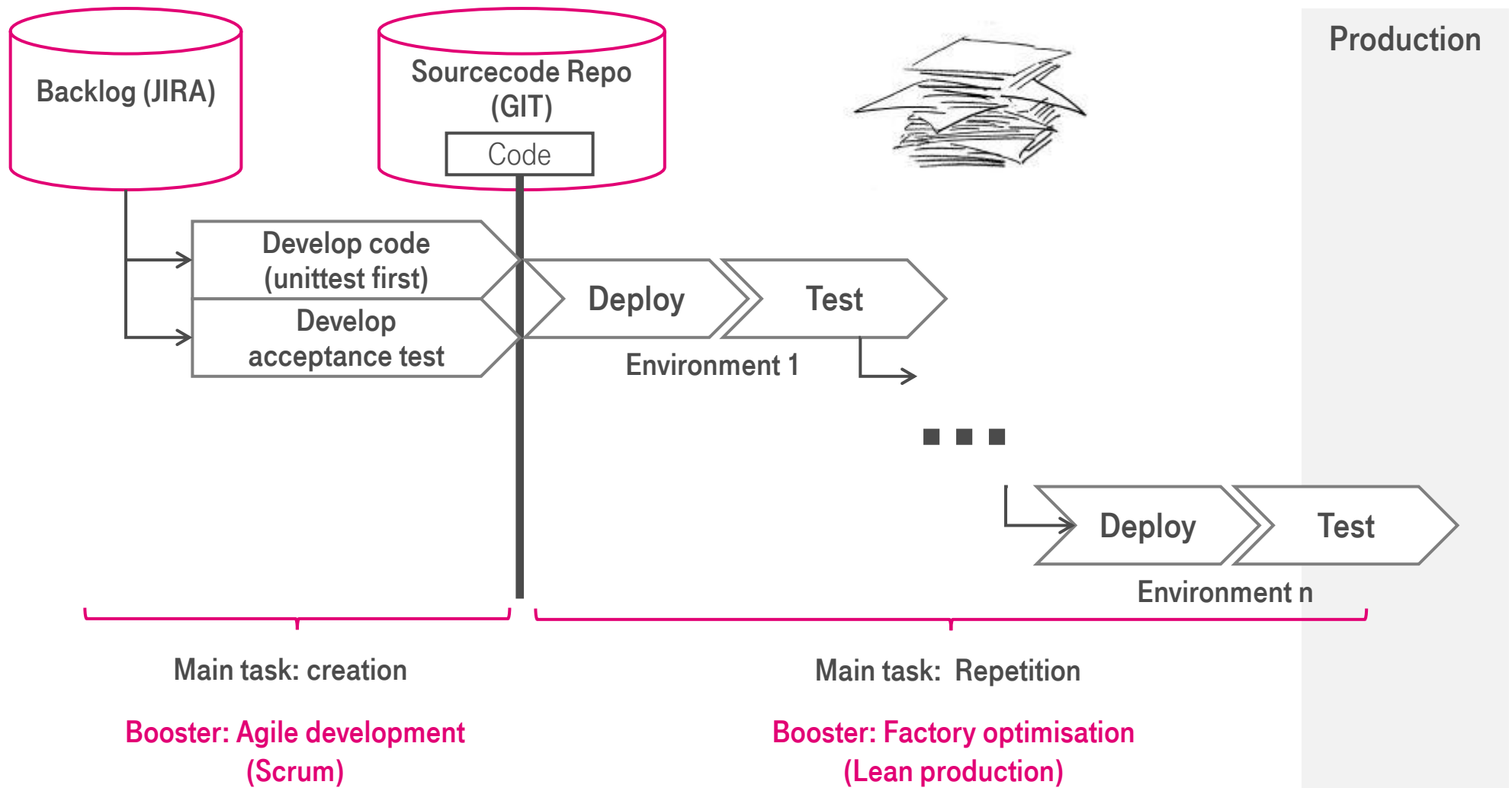
Diese Schulungsunterlage ist ausschließlich für die im Titel genannte Veranstaltung zu verwenden. Eine Weitergabe der gesamten Präsentation oder einzelner Folien sowie eine Verwendung der Inhalte in anderen Vorträgen ist nicht gestattet.

1 CONTINUOUS DELIVERY REVISITED

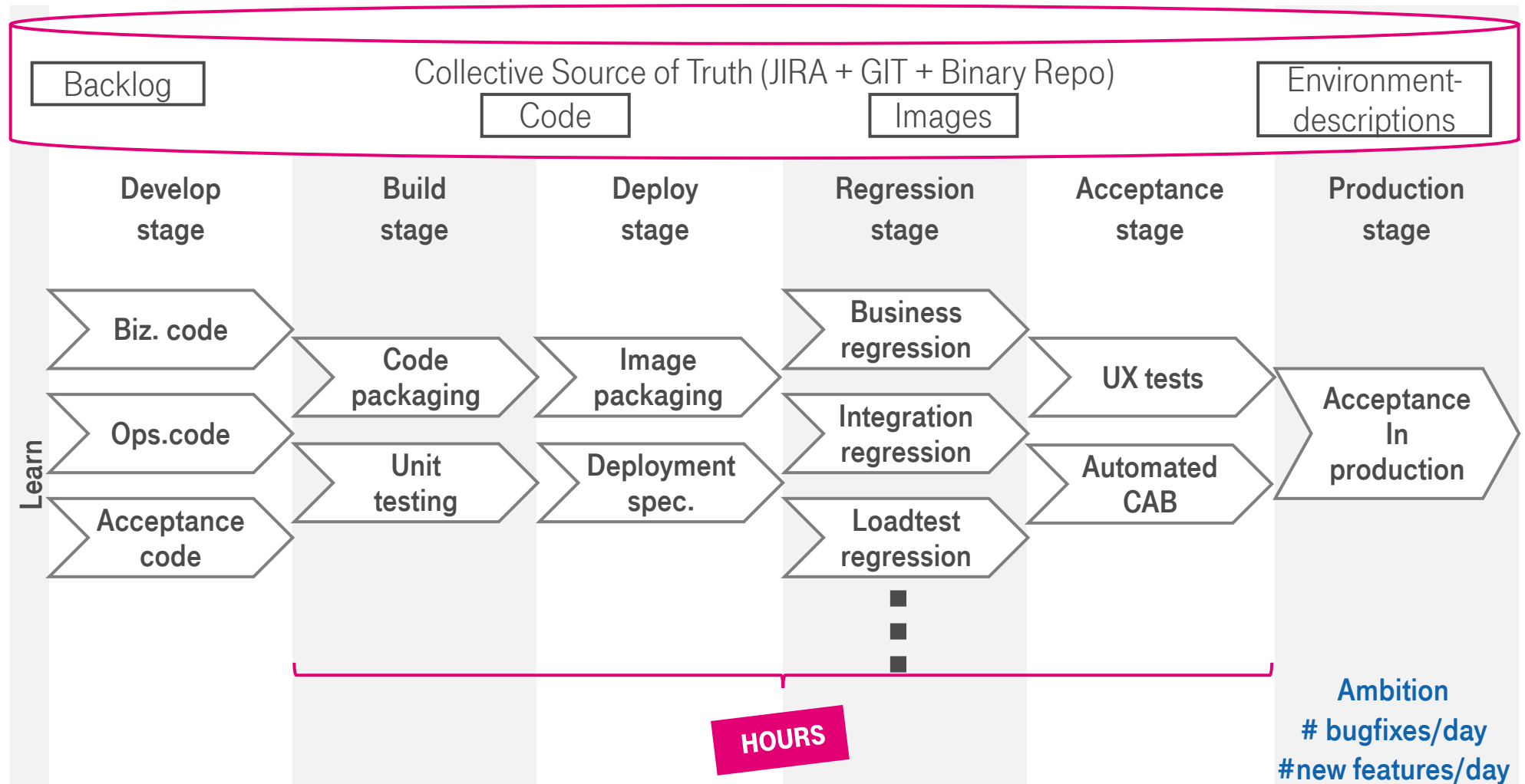
CONTINUOUS IMPROVEMENT - KAIZEN



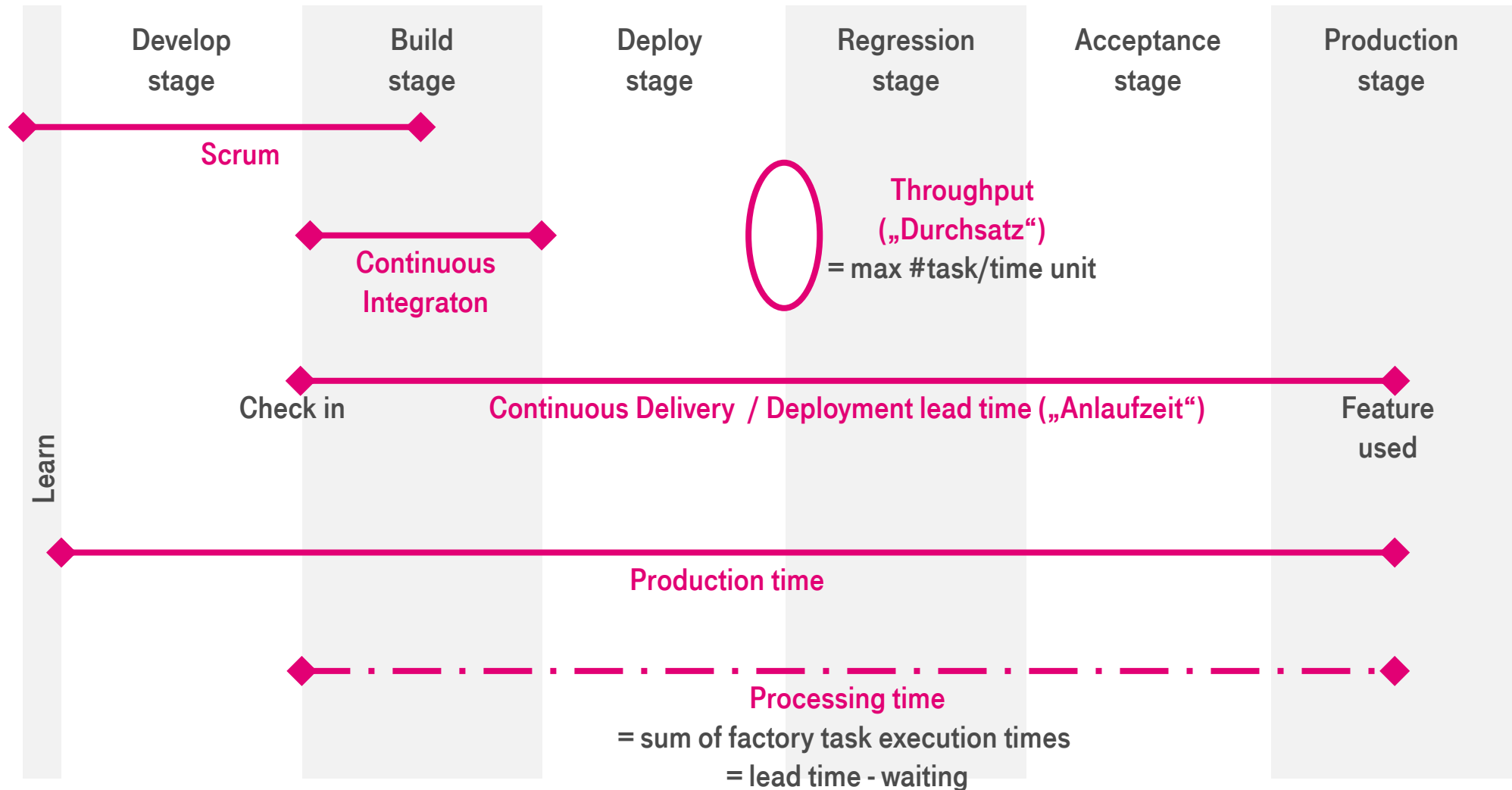
END-2-END AGILE SOFTWARE PRODUCTION AS-IS PROCESS VIEW



END-2-END AGILE SOFTWARE PRODUCTION TO-BE PROCESS VIEW



WORD DEFINITIONS AND TIME INTERVALS



THE SEVEN WASTES OF SOFTWARE DEVELOPMENT

#1 - Partially Done Work (Inventory)

#2 - Extra Features, unused features (Overproduction)

#3 – Relearning (Extra processing)

#4 – Handoffs (Transportation)

#5 – Delays (Waiting)

#6 - Task Switching (Motion, Jumping)

#7 - Defects

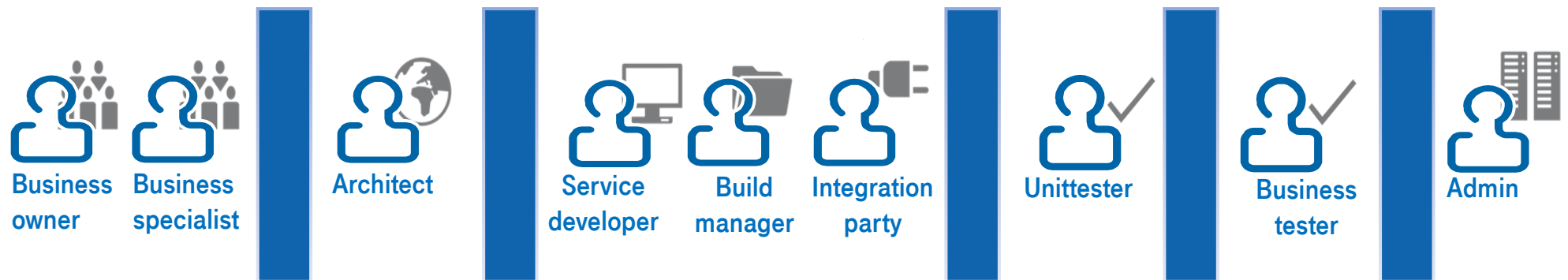
“

You build it, you run it.

Werner Vogel, Amazon

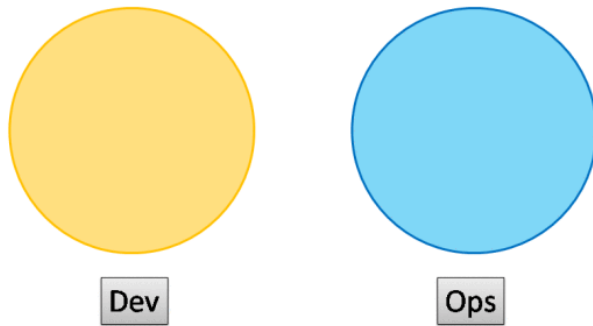
”

HANDOFFS: EXPERT TEAMS REPLACED BY CROSS-FUNCTIONAL TEAMS

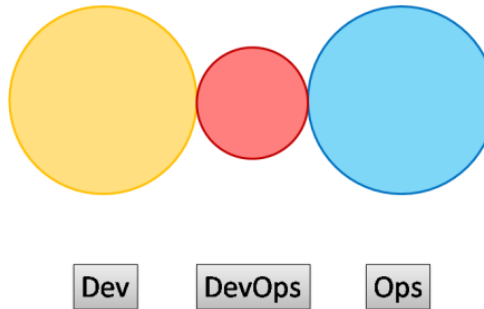


TEAM-STRUCTURES: DEVOPS ANTI-TYPES

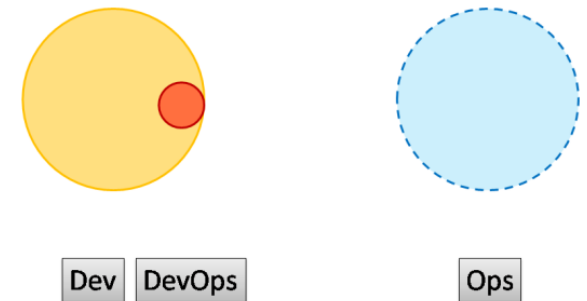
Anti-Type A – Separate Silos



Anti-Type B – Separate DevOps Silo



Anti-Type C – “We Don’t Need Ops”



DevOps Patterns:
Team Topologies

Matthew Skelton

@matthewpskelton

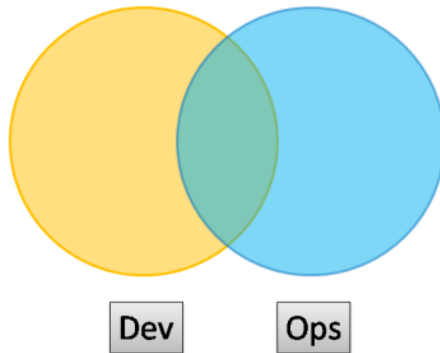
softwareoperability.com + experiencedevops.org

12

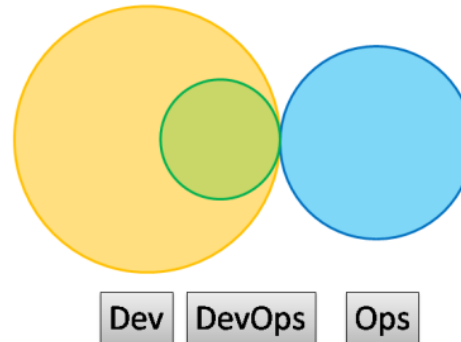
TEAM-STRUCTURES: DEVOPS POSITIVE PATTERNS

REMEMBER: CROSS-FUNCTIONAL IS MANDATORY

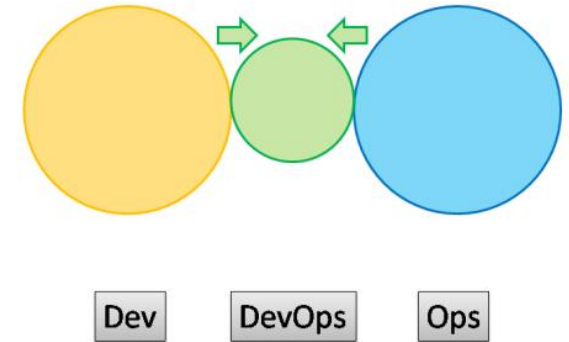
Type 1 – Smooth Collaboration



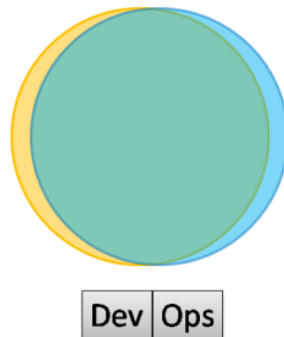
Type 3 – Infrastructure-as-a-Service



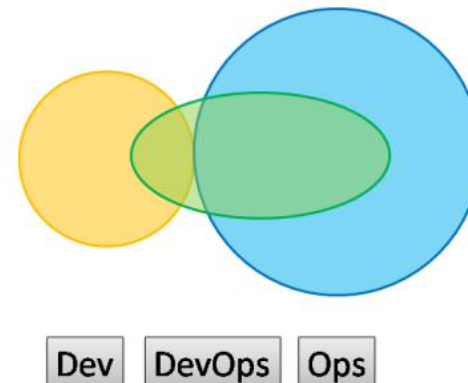
Type 5 – Temporary DevOps Team



Type 2 – Fully Embedded



Type 4 – DevOps-as-a-Service



**KAFFEE-
PAUSE**



AGENDA

01 Intro: Continuous Delivery Pipeline revisited

02 Reproducibility: Everything as Code

02.1 Exercise: Trigger pipeline (Jenkins from GIT)

03 Provisioning time: Infrastructure as Code

03.1 Exercise: Automated install and configuration

04 Dev-Prod-parity: Immutable Infrastructures

04.1 Exercise/Video: Image bakery

05 Conways law and continuous feedback

05.1 Exercise: ELK logging as microservices

06 Learn: Red/black, A/B and canaries

06.1 Exercise: My little A/B test

PATH 1:

Optimize flow

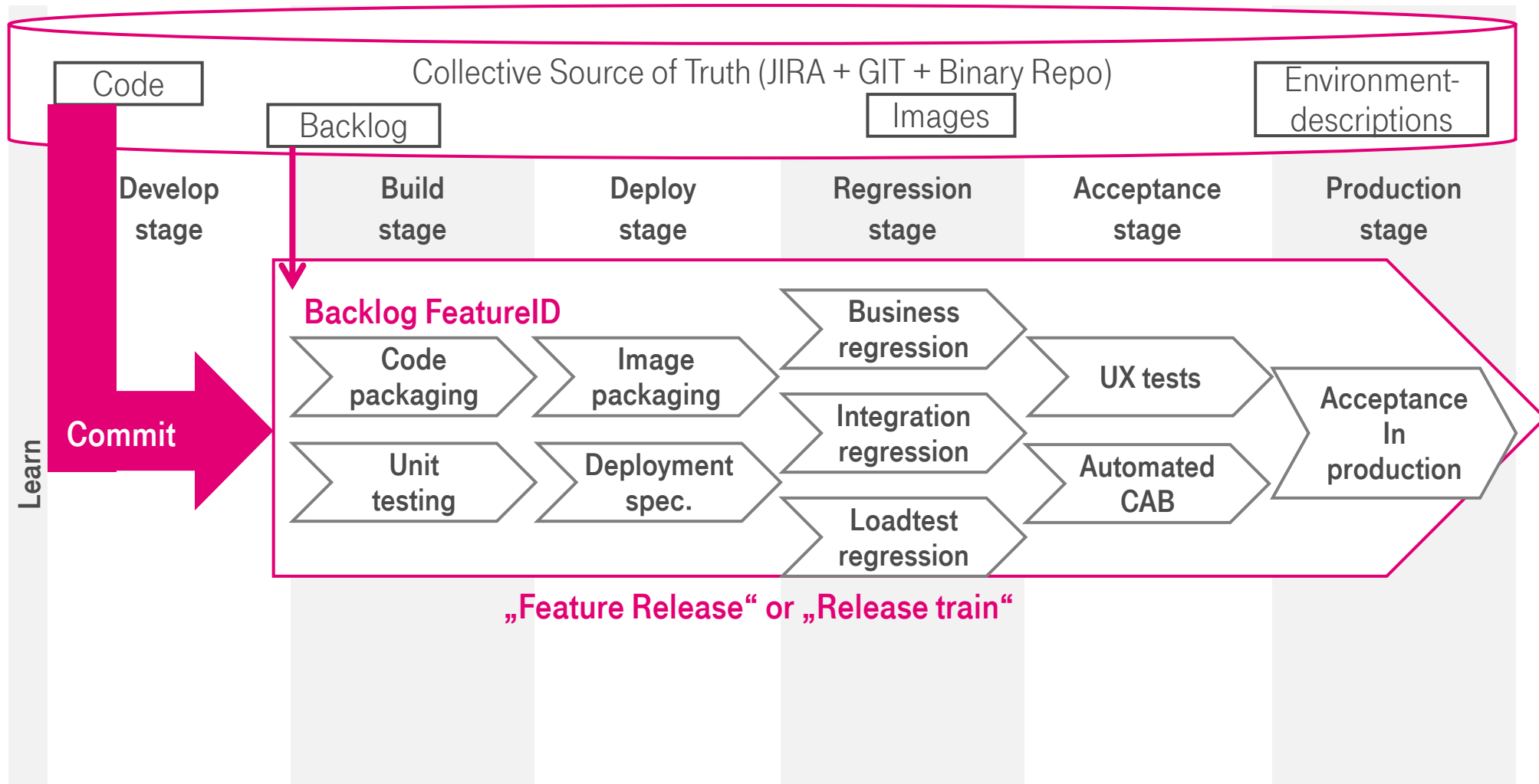
PATH 2:

Include feedback

PATH 3:

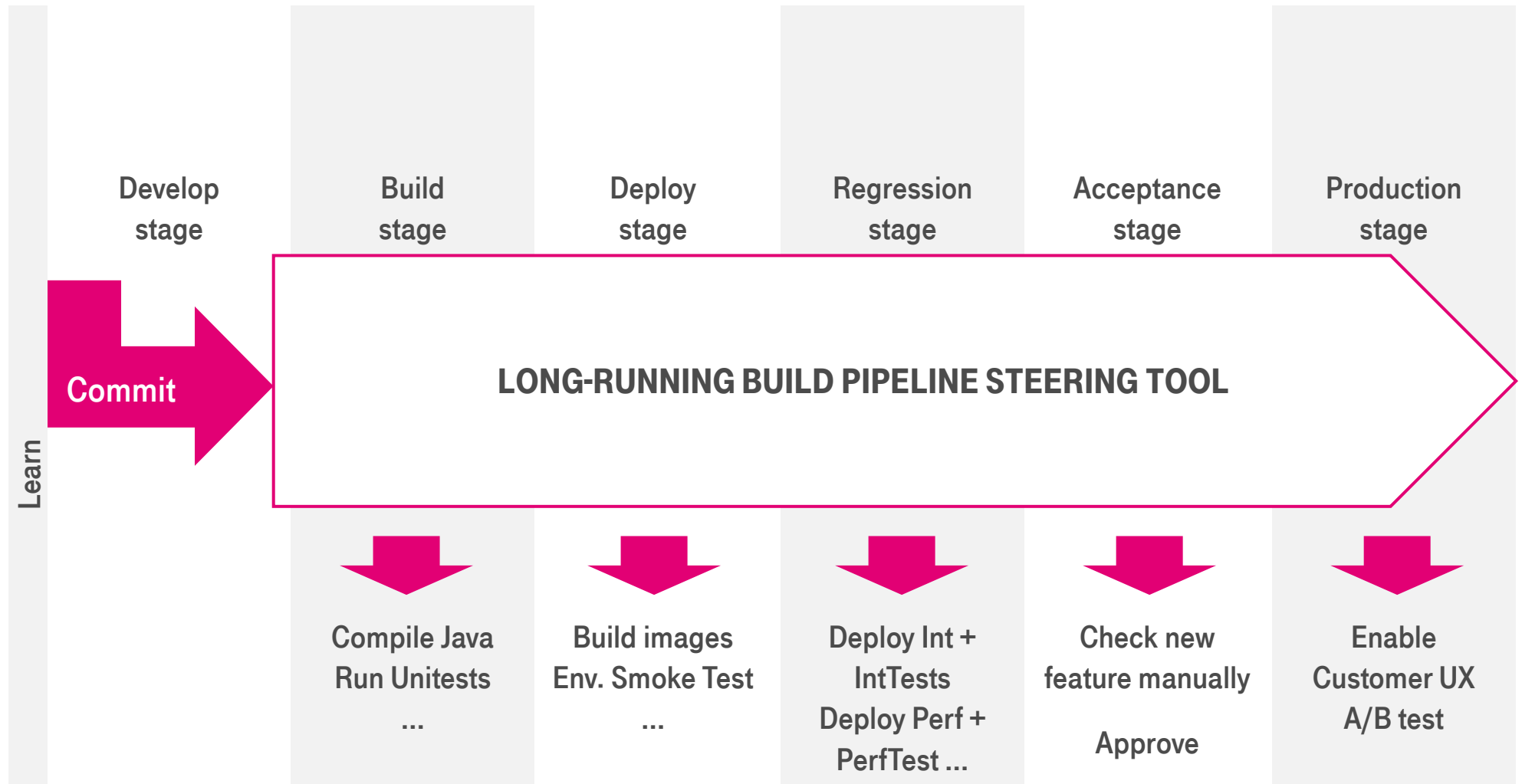
Learn from system

DEVOPS PIPELINE TRIGGER: FEATURE CHECK IN



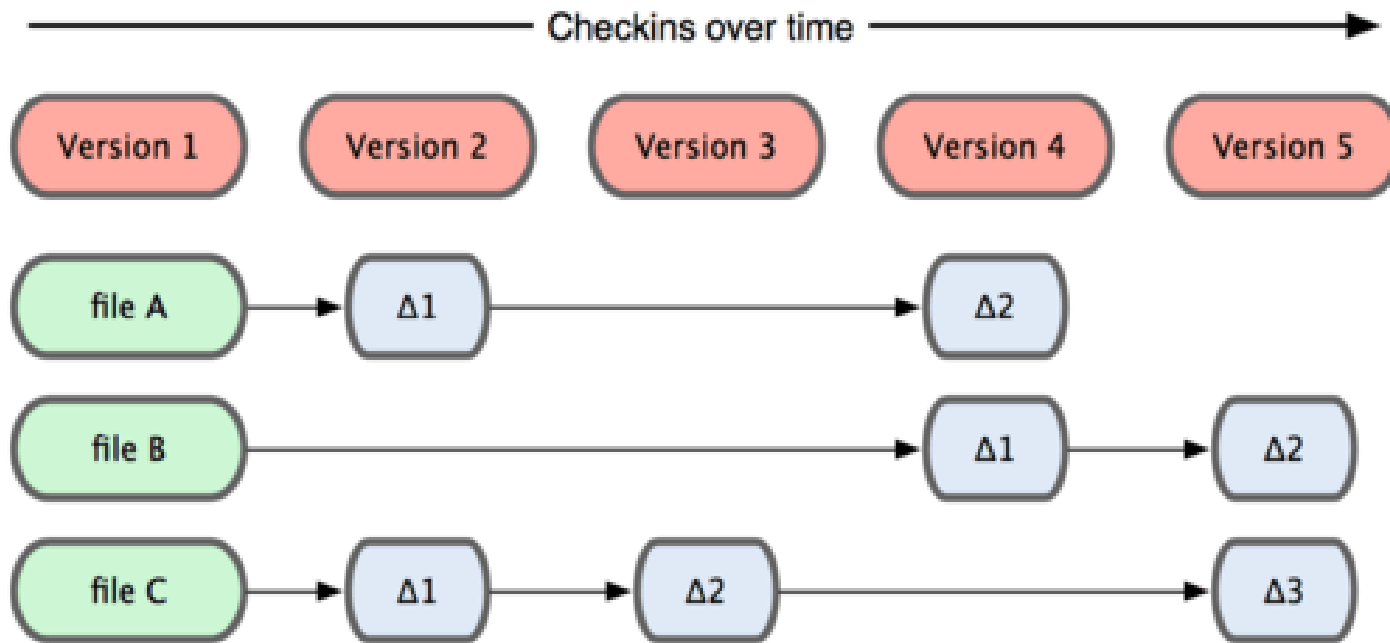
PIPELINE CONTROL

JENKINS, CLOUDBEES, CONCOURSE, OPENSIFT, ...



KONZEPT VON GIT

DATENMODELL BEI CVS UND SVN



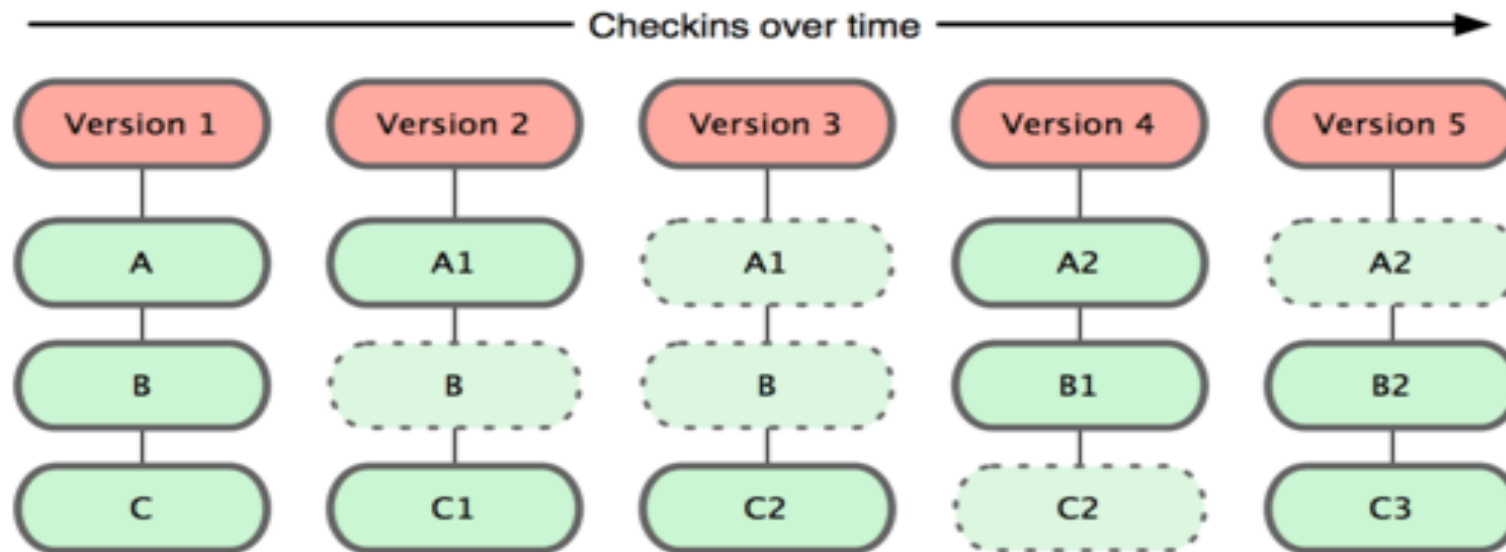
CVS und SVN speichern Informationen als eine fortlaufende von Änderungen an Dateien.

KONZEPT VON GIT

DATENMODELL BEI GIT

Speicherung von Snapshots, nicht Diffs

- Git speichert Informationen als eine Reihe von Snapshots eines Mini-Dateisystems
- Bei jedem Commit wird der Zustand sämtlicher Dateien in diesem Moment gesichert und als Referenz auf diesen Snapshot gespeichert
- Git speichert Daten als eine Historie von Snapshots des Projektes



BEREICHE EINES GIT PROJEKTES

ARBEITSVERZEICHNIS

- Verzeichnis auf Festplatte zum Bearbeiten von Dateien

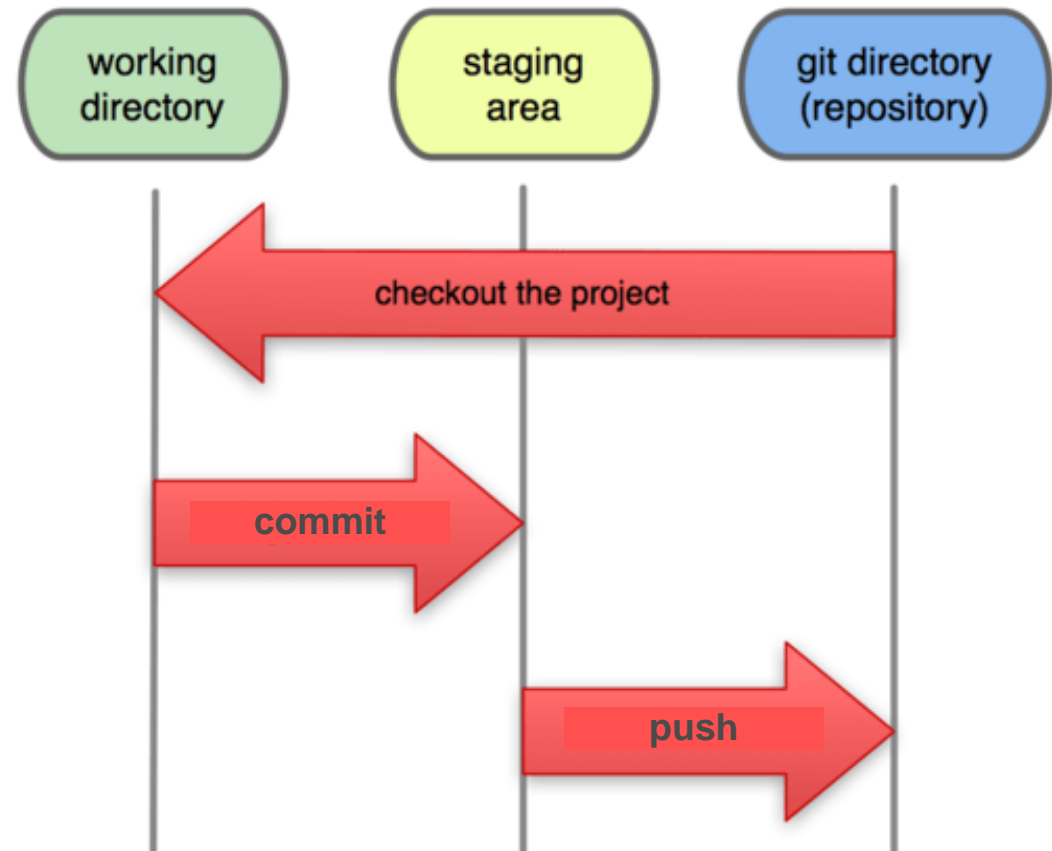
STAGING AREA (INDEX)

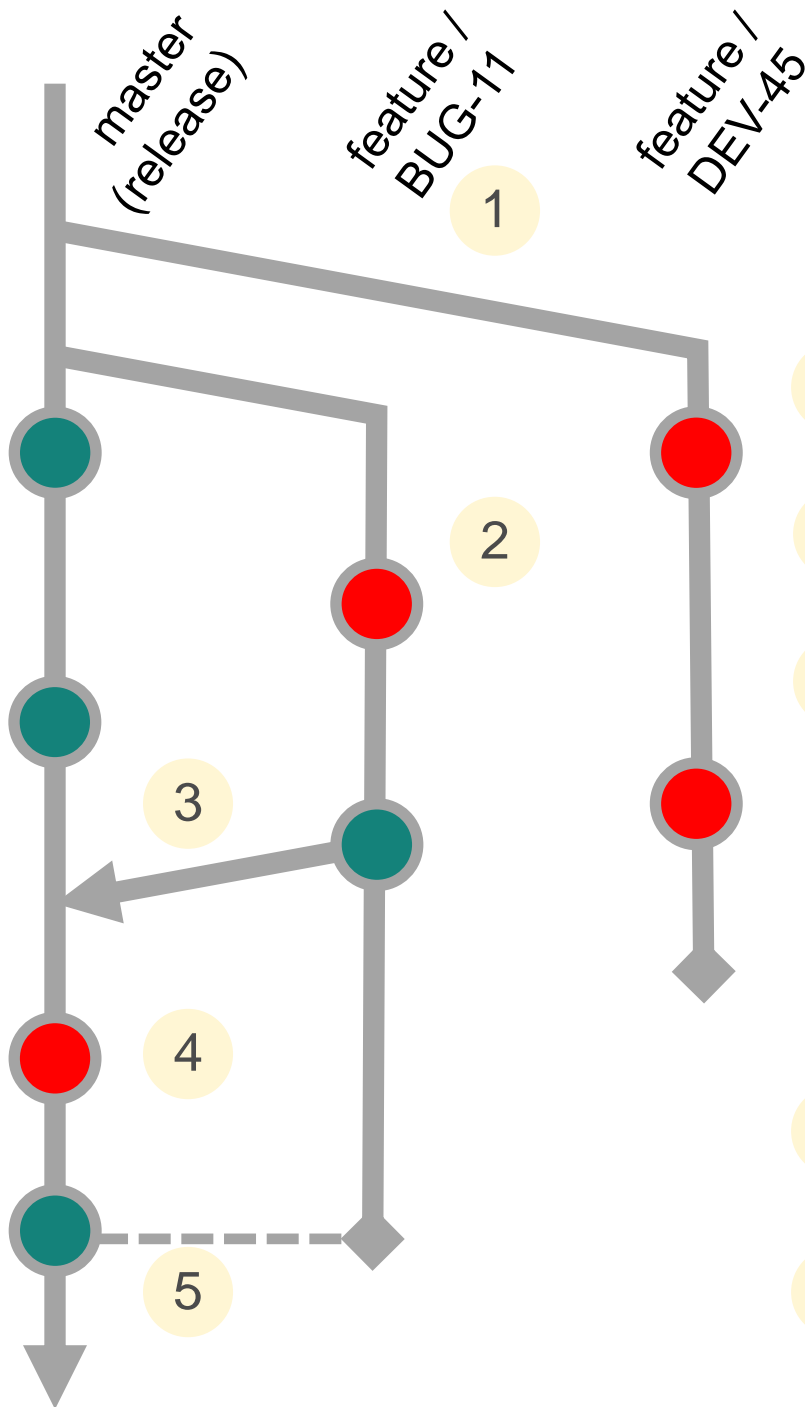
- Datei hält Informationen zu vorgemerkten Dateien

GIT VERZEICHNIS (REPOSITORY)

- Git Metadaten, lokale DB

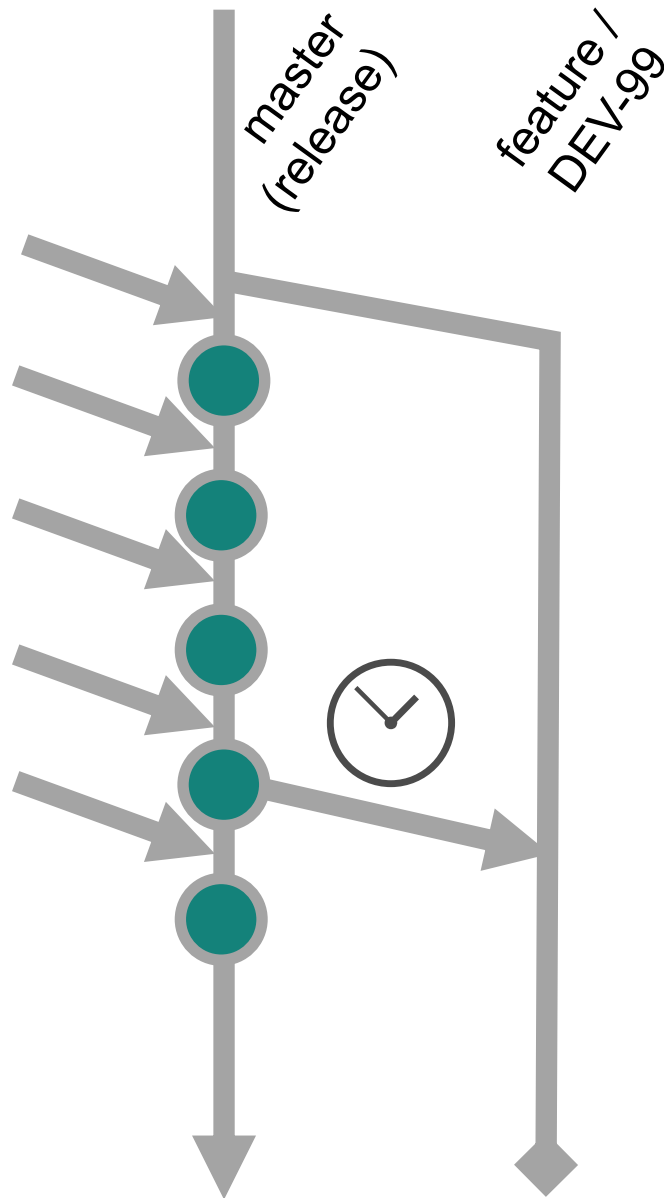
Local Operations





CONTINUOUS DELIVERY BRANCHING STRATEGY

- 1 Every change is a feature branch.**
Branch name contains reference to backlog or issue.
- 2 Break stuff on branches – check with Continuous integration**
Every checkin in branch triggers CI pipeline.
- 3 Ready for release**
Variant a: Pull request
Reviewer pulls requests –
Continuous Delivery pipeline starts on approval
Variant b: Merge up release
Continuous delivery starts on merge to main branch
- 4 Jidoka – „stop the line production“ on broken pipeline**
All people ready for rescue!
- 5 Remove/cleanup branch after release**



ANTI-PATTERN: LONG-RUNNING BRANCHES

Plan/split work
in short-running branches.

Avoid rebase!

Frequently merge master → feature


Useful reporting:


- all non-released branches with age
→ force merges in dailies
- Top-ten of long-running features/issues
→ input for backlog grooming &
work planning (WIP)

<https://www.atlassian.com/continuous-delivery/why-git-and-continuous-delivery-are-super-powered>

EXERCISE 1: PRODUCTION PIPELINE

CONSOLE.OTC.T-SYSTEMS.COM

 **OPEN TELEKOM CLOUD** [_Register](#)

**SIMPLE
SECURE
AFFORDABLE**

Multitenant Login

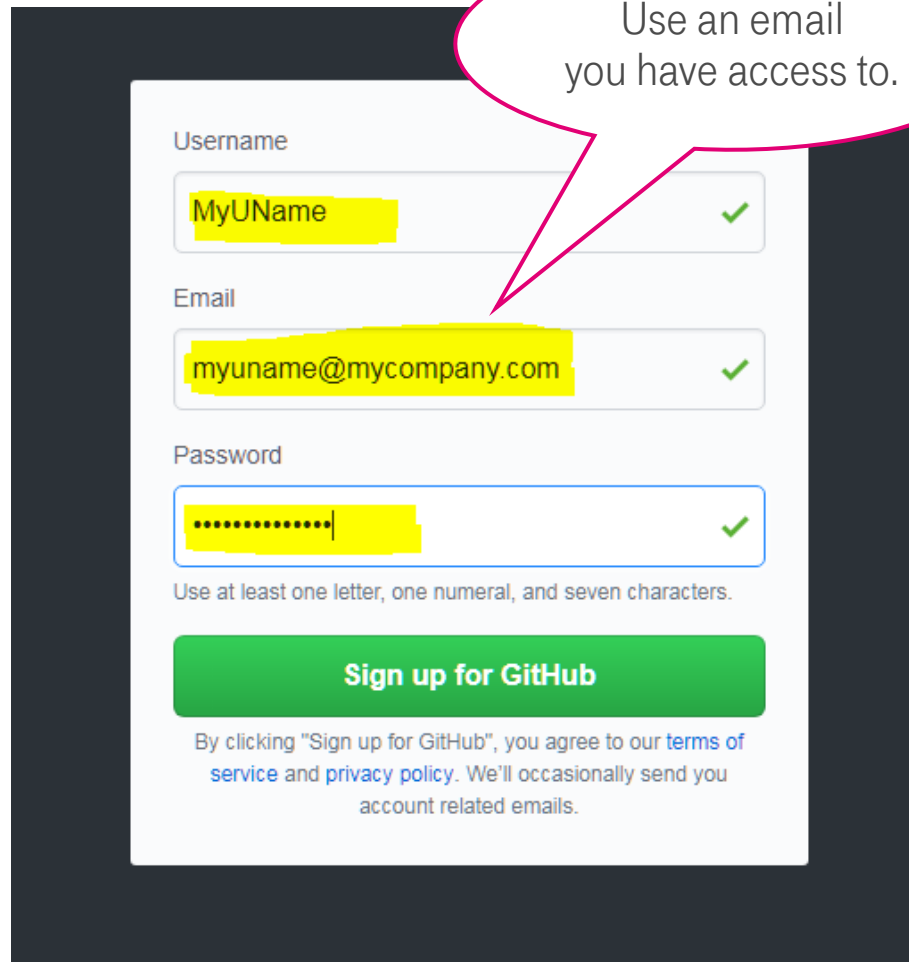
☒ Remember user name

Log In

© 2017 T-Systems International GmbH. All rights reserved. [Legal Statement](#) | [Privacy Protection](#) | [Legal Agreement](#)

REGISTER A PUBLIC GITHUB ACCOUNT

<https://www.github.com>



Use an email you have access to.

Username
MyUName ✓

Email
myuname@mycompany.com ✓

Password
..... ✓
Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

Verify your email address: Klick verification link

Choose your personal plan

- ☒ Unlimited public repositories for free.
- ☐ Unlimited private repositories for \$7/month.

Don't worry, you can cancel or upgrade at any time.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Read the guide

Start a project

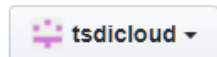
CREATE A (TEMPORARY) GITHUB PROJECT

CHOOSE YOUR OWN ORG AND REPO NAME

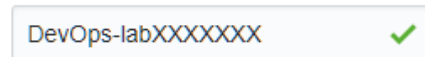
Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name



Great repository names are short and memorable. Need inspiration? How about [miniature-guide](#).

Description (optional)

Tutorial for T-Systems DevOps workshop on Telekom Cloud Platforms



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

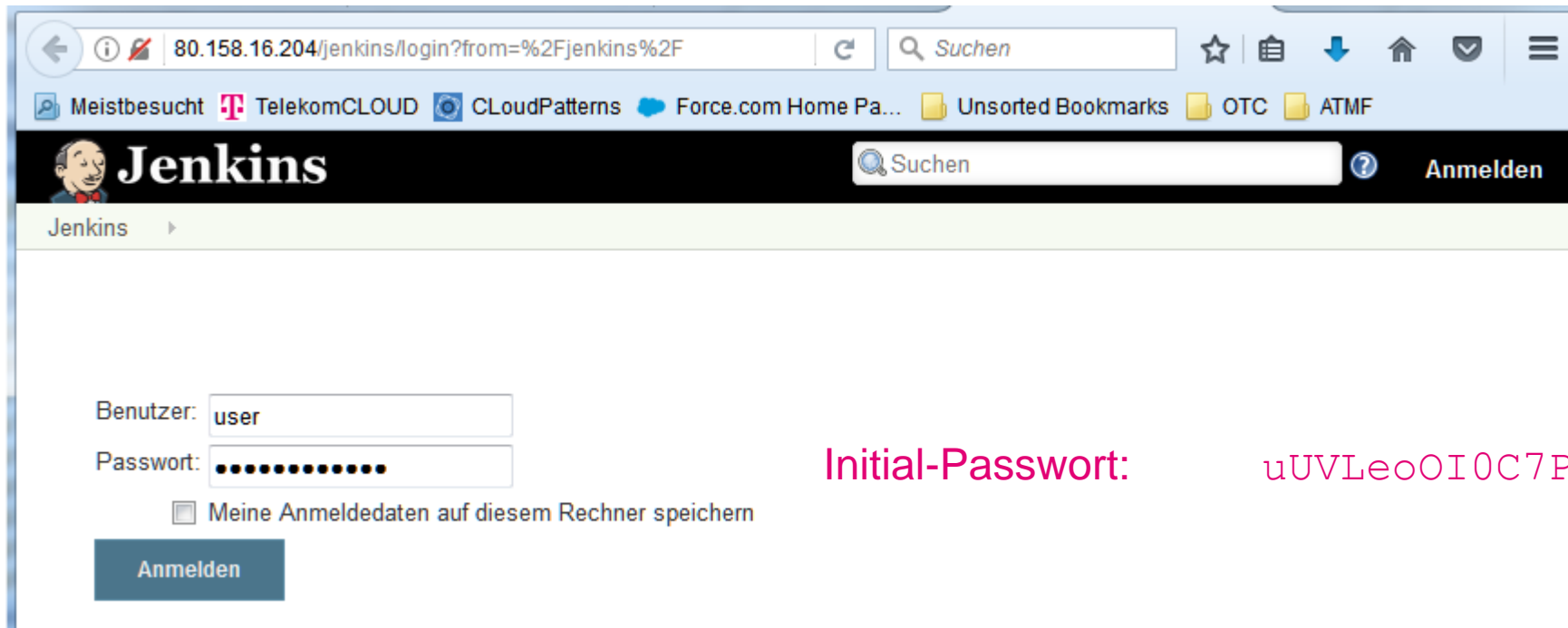
Add .gitignore: **None**

Add a license: **None**



Create repository

JENKINS(1): LOGIN



The screenshot shows a web browser window with the address bar displaying `80.158.16.204/jenkins/login?from=%2Fjenkins%2F`. The browser's bookmark bar includes links like 'Meistbesucht', 'TelekomCLOUD', 'CLOUDPatterns', 'Force.com Home Pa...', 'Unsorted Bookmarks', 'OTC', and 'ATMF'. The Jenkins header features the logo, a search bar with the text 'Suchen', and an 'Anmelden' button. Below the header, the main content area contains a login form with the following elements:

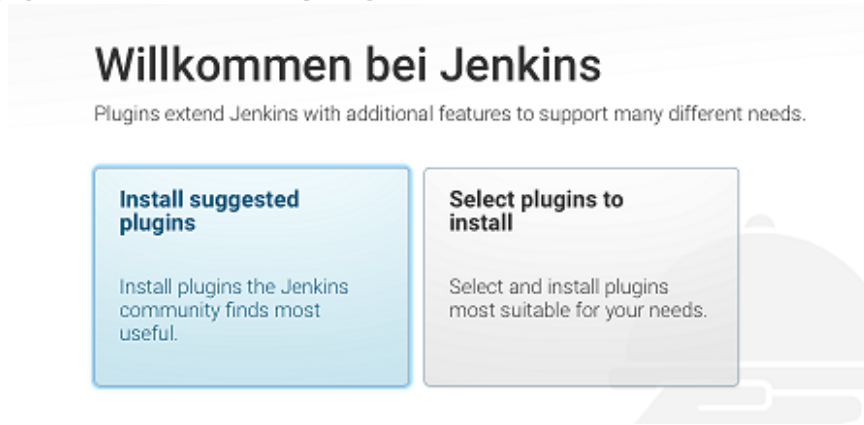
- A label 'Benutzer:' followed by a text input field containing the text 'user'.
- A label 'Passwort:' followed by a password input field filled with dots.
- A checkbox labeled 'Meine Anmeldedaten auf diesem Rechner speichern'.
- A blue 'Anmelden' button.

Initial-Passwort:

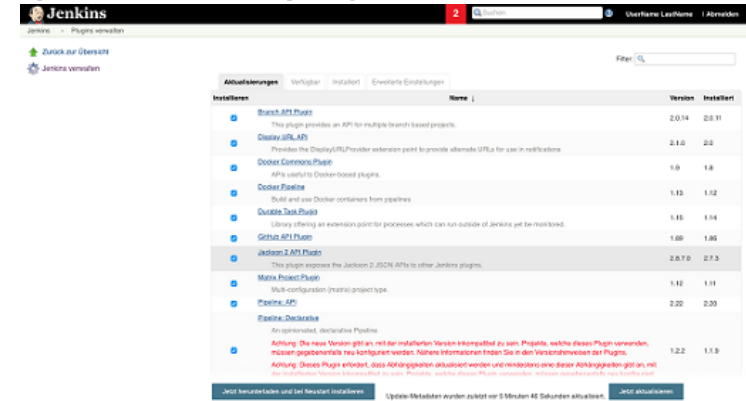
uUVLeoOI0C7P

SETUP JENKINS

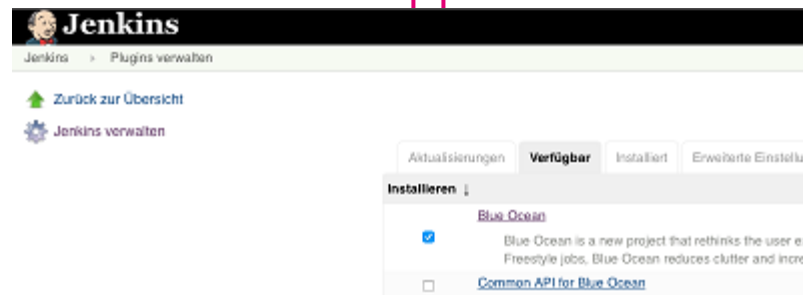
(1) Install default plugins + restart



(2) Update default plugins to current state + restart



(3) Install GitHub integration plugin



(4) Install Blue Ocean plugins + restart

SETUP WEBHOOK IN GITHUB

tsdicloud / devops-tutor

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Options

Collaborators

Branches

Webhooks

Integrations & services

Deploy keys

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

Content type

Secret

Which events would you like to trigger this webhook?

☐ Just the push event.

☐ Send me everything.

☒ Let me select individual events.

☐ **Commit comment**
Commit or diff commented on.

☐ **Delete**
Branch or tag deleted.

☐ **Deployment status**
Deployment status updated from the API.

☐ **Gollum**
Wiki page updated.

☐ **Issues**
Issue opened, edited, closed, reopened, assigned, unassigned, labeled, unlabeled, milestone, or demilestoned.

☐ **Create**
Branch or tag created.

☐ **Deployment**
Repository deployed.

☐ **Public**
Repository changes from private to public.

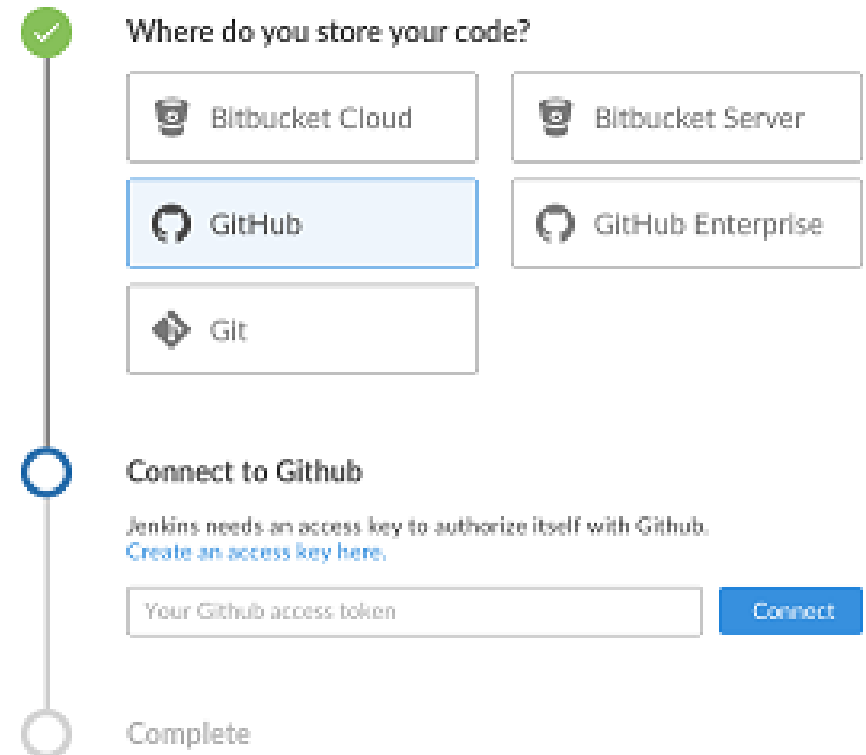
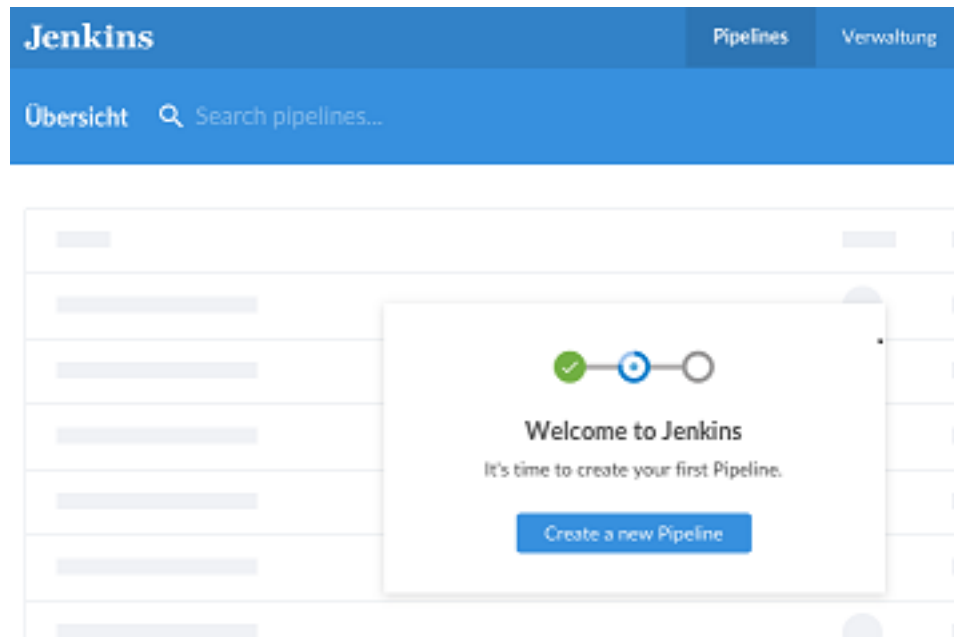
☐ **Pull request review**
Pull request review submitted, edited, or dismissed.

☒ **Push**
Git push to a repository.

☐ **Repository**

http://<jenkins-eip>/jenkins/github-webhook/

CREATE FIRST PIPELINE(1)



CREATE PIPELINE(2)

COPY TOKEN FROM GITHUB

The screenshot shows the GitHub Developer settings page. On the left, the 'Developer settings' menu is open, with 'Personal access tokens' selected. The main content area is titled 'New personal access token'. It includes a 'Token description' field with the text 'Jenkins ContDelivery Interaction', a 'Select scopes' section with checkboxes for 'repo', 'repo:status', 'repo:deployment', 'public_repo', and 'repo:invite', and a 'Generate new token' button. A blue banner message states: 'Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.' Below this, the 'Personal access tokens' section shows a list of generated tokens. The first token is highlighted in green and contains the text 'b47dd5ce0de15aa768f7abdb8c6e17eded8a837'. A blue banner message above the token list says: 'Make sure to copy your new personal access token now. You won't be able to see it again!'. The token is followed by an 'Edit' button. At the bottom, a note explains that personal access tokens function like ordinary OAuth access tokens and can be used instead of a password for Git or to authenticate to the API over Basic Authentication.

Settings / Developer settings

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Token description

Jenkins ContDelivery Interaction

What's this token for?

Select scopes

Scopes define the access for

- ☒ repo
 - ☒ repo:status
 - ☒ repo:deployment
 - ☒ public_repo
 - ☒ repo:invite

Settings / Developer settings

Personal access tokens

Generate new token

Tokens you have generated that can be used to access the [GitHub API](#).


Make sure to copy your new personal access token now. You won't be able to see it again!


✓ b47dd5ce0de15aa768f7abdb8c6e17eded8a837 Edit


Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).


CREATE PIPELINE(3) – USE ORG AND REPO FROM BEFORE


✓ Where do you store your code?

 Bitbucket Cloud

 Bitbucket Server

 GitHub

 GitHub Enterprise

 Git


✓ Connect to Github

Jenkins needs an access key to authorize itself with Github.
[Create an access key here.](#)

.....


✓

○ Which organization does the repository belong to?

 tsdicloud

✓ Choose a repository

Loaded "1" repositories

 Search...

devops-tutor

Create Pipeline

○ There are no Jenkinsfiles in *devops-tutor*

Create Pipeline

DEMO PIPELINE INSTALLATION

Laptop local: Get demo projects

```
$ git clone https://bitbucket.org/gdccloudconsulting/devops\_lab
```

Laptop local: Get GitHub project

```
$ git clone https://github.com/<owner>/<repository\_name>
```

Laptop local: Copy Jenkinsfile

```
$ cp devops_lab/exercises/jenkins/Jenkinsfile <repository_name>
$ cd <repository_name>
$ git commit Jenkinsfile -m „Example pipeling with CI and CD support“
$ git config user.name <github username>
$ git config user.email <email for github notifications>
$ git push origin master
```


GIT BASICS – CONSISTENT CHECKIN COMMENTS

Git cheat sheet:

<https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>

How to Write a Git Commit Message

<https://chris.beams.io/posts/git-commit/>

TIP: Complete the sentence

*“If applying this patch,
the change will...*

<verb> <object of change>”

```
$ git log --oneline -5 --author pwebb --before "Sat Aug 30 2014,,  
5ba3db6 Fix failing CompositePropertySourceTests  
84564a0 Rework @PropertySource early parsing logic  
e142fd1 Add tests for ImportSelector meta-data  
887815f Update docbook dependency and generate epub  
ac8326d Polish mockito usage
```

START DEVELOPMENT AND CONTINUOUS INTEGRATION

Laptop local: Create feature branch

```
$ cd <repository_name>
$ git branch feature/DEV-4711
$ git checkout feature/DEV-4711
```

Add some file „componentA.txt“ to the branch

```
$ git commit componentA.txt -m „Add component A initial implementation“
$ git push origin feature/DEV-4711
```

CI pipeline in Jenkins should start.

START CONTINUOUS DELIVERY – DIRECT MERGE

Laptop local: Switch to master branch, merge feature branch to it

```
$ cd <repository_name>
$ git checkout master
$ git merge feature/DEV-4711
```

Push new master version

```
$ git commit componentA.txt -m „Add component A initial implementation“
$ git push origin master
```

CD pipeline in Jenkins should start.

Accept or cancel pipeline to finish CD job.

PULL REQUEST (REAL WORLD PROJECT)

The screenshot shows the GitHub interface for the `elastic/elasticsearch` repository. The top navigation bar includes links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name 'elastic / elasticsearch' is displayed, along with statistics: 2,173 Watchers, 25,975 Stars, and 9,154 Forks. Below this, a tabbed interface shows 'Pull requests' with 116 items, alongside 'Code', 'Issues' (1,212), 'Projects' (1), and 'Insights'. A search bar with the filter 'is:pr is:open' and buttons for 'Labels' and 'Milestones' are present. A green 'New pull request' button is in the top right. The main section lists 116 open pull requests, with columns for Author, Labels, Projects, Milestones, Reviews, Assignee, and Sort. The list includes pull requests #27090, #27089, #27087, #27086, #27085, #27083, and #27082, each with a title, status (e.g., 'Stats', 'Ingest', 'Allocation'), version tags, and a comment count.

elastic / elasticsearch Watch 2,173 Star 25,975 Fork 9,154

Code Issues 1,212 Pull requests 116 Projects 1 Insights

Filters is:pr is:open Labels Milestones New pull request

116 Open ✓ 12,494 Closed Author Labels Projects Milestones Reviews Assignee Sort

- Expose adaptive replica selection stats in `/_nodes/stats` API** ✗ :Stats enhancement v6.1.0 v7.0.0
#27090 opened 9 hours ago by dakrone
- Introduce templating support to `timezone/locale` in `DateProcessor`** ✓ :Ingest enhancement WIP
#27089 opened 10 hours ago by talevy
- Blacklist Gradle 4.2 and above** ✓ build review v5.6.4 v6.0.0 v6.1.0 v7.0.0 2
#27087 opened 11 hours ago by jasontedor • Approved
- Allocation: add delay between retries for failed allocations** ✓ :Allocation enhancement v7.0.0 2
#27086 opened 12 hours ago by dnhatn
- Deprecate the transport client in favour of the high-level REST client** ✓ :Java API deprecation review v6.0.0 v6.1.0 v7.0.0 1
#27085 opened 14 hours ago by javanna
- Update numbers to reflect 4-byte UTF-8-encoded characters** • 2
#27083 opened 16 hours ago by DaveCTurner • Approved
- Refactor internal engine** ✓ :Engine non-issue review 6
#27082 opened 17 hours ago by jasontedor

**KAFFEE-
PAUSE**



AGENDA

01 Intro: Continuous Delivery Pipeline revisited

02 Reproducibility: Everything as Code

02.1 Exercise: Trigger pipeline (Jenkins from GIT)

03 Provisioning time: Infrastructure as Code

03.1 Exercise: Automated install and configuration

04 Dev-Prod-parity: Immutable Infrastructures

04.1 Exercise/Video: Image bakery

05 Conways law and continuous feedback

05.1 Exercise: ELK logging as microservices

06 Learn: Red/black, A/B and canaries

06.1 Exercise: My little A/B test

PATH 1:

Optimize flow

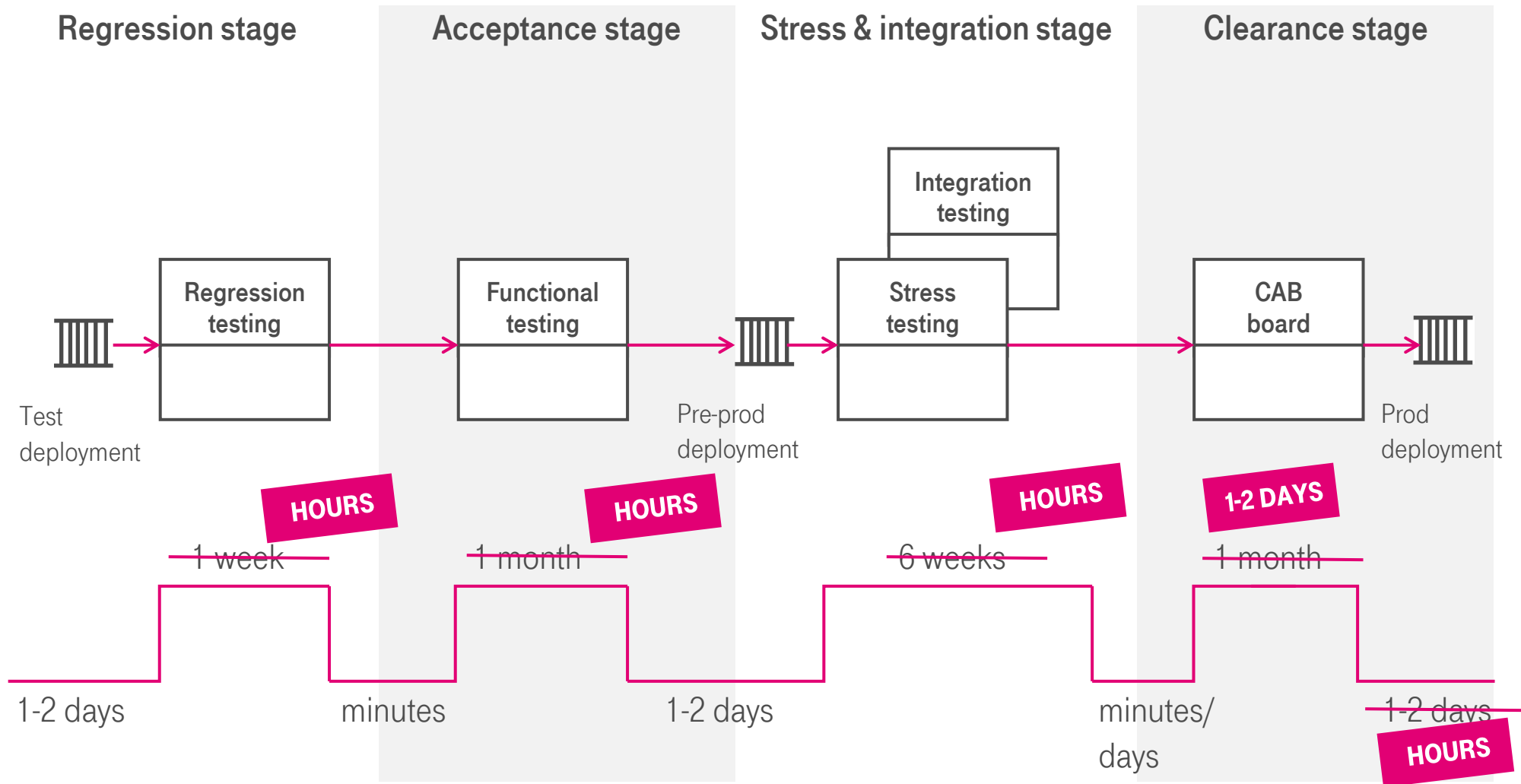
PATH 2:

Include feedback

PATH 3:

Learn from system

VALUE STREAM ANALYSIS (TEST ONLY, SIMPLIFIED)



SMALL BATCH PRODUCTION

THE ENVELOPE EXPERIMENT

JAMES WOMACK AND DANIEL JONES, LEAN THINKING

FATHER

- put address on envelope
- put stamp on envelope
- put letter into envelope
- seal envelope

Next

SON

- all addresses on all envelope
- all stamps on all envelope
- all letter into envelopes
- seal envelopes

→ Forgotten steps: sorting, stacking, handling

→ Learning effect is overestimated

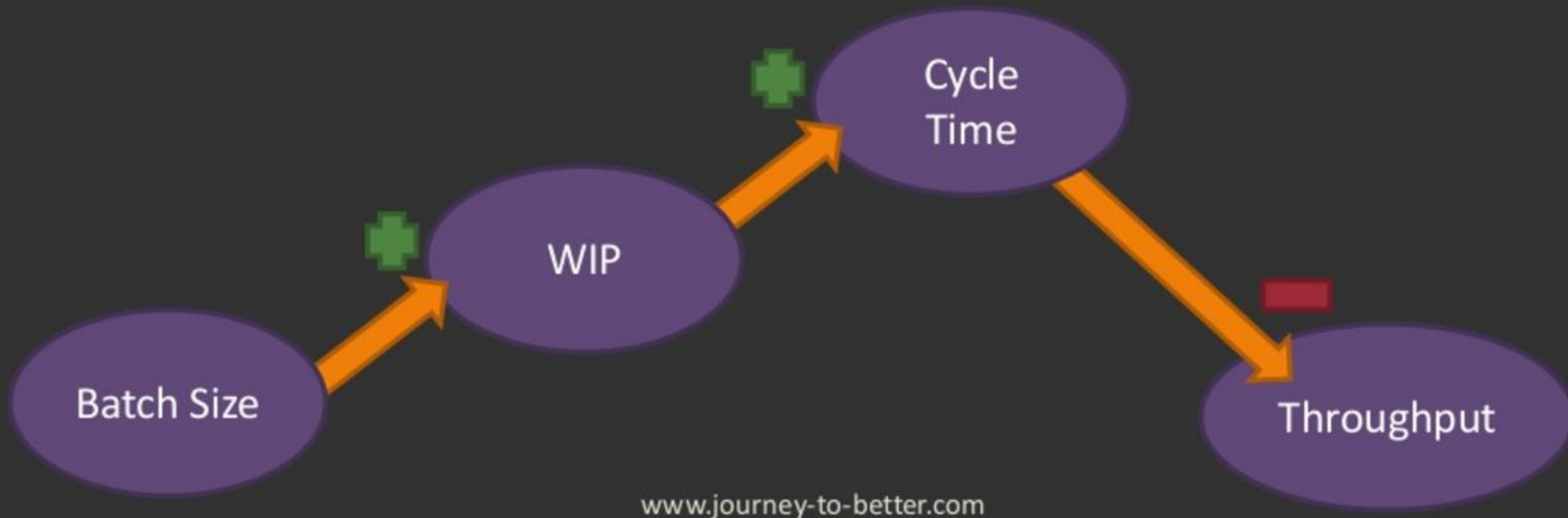
Father's strategy wins (empirically proven)

AND is interruptable

AND has fewer risk for throughput drops.

QUEUING THEORY: LITTLES LAW

$$\text{Avg. Cycle Time} = \frac{\text{Work In Progress (WIP)}}{\text{Avg. Throughput Rate}}$$



<https://de.slideshare.net/andrewrusling/improving-throughput-with-the-theory-of-constraints-and-queuing-theory>

FINDINGS

Work storage = high WIP → high cycle time → low throughput
→ Smaller batch reduces service time.

Utilisation per service station is maximal at the bottleneck.
→ Keep bottleneck at maximal speed.

More resources increase throughput at a work station.
→ More resources only help if they scale the bottleneck.

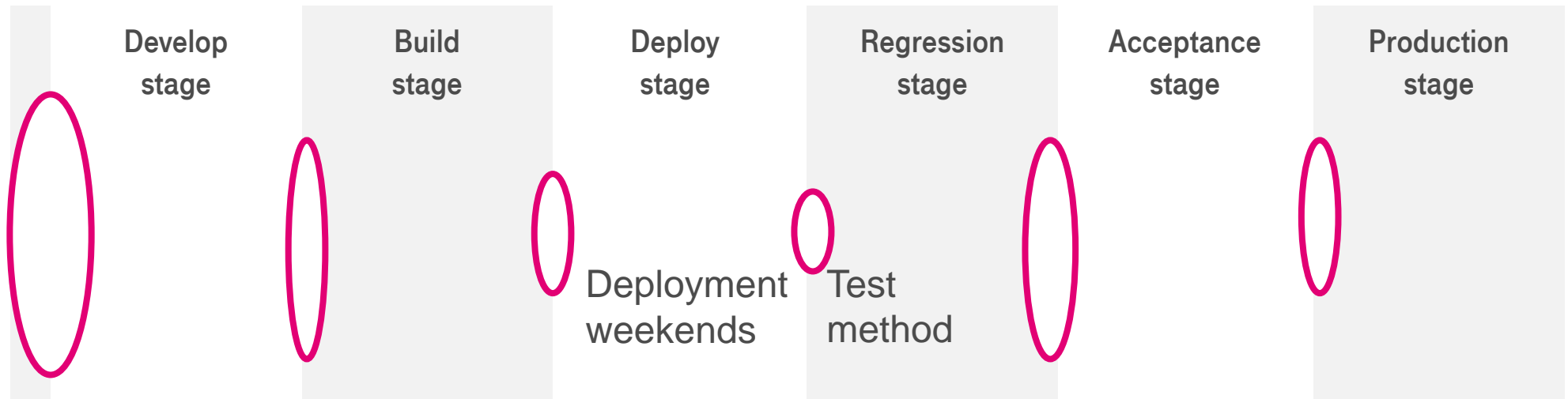
“

A chain is not stronger than
its weakest link.

”

CONSTRAINT OPTIMISATION

HUNT FOR BOTTLENECKS



- Finding: Throughput limitation by MANUAL regression testing
Measure: 90% automation of regression tests
- Finding: Throughput limitation by MANUAL environment deployments
Measure: Automated deployment
- Finding: Throughput limitation by AVAILABILITY of required environments
Measure: ALL ENVIRONMENTS IN CLOUD; add Infrastructure as code to automation

“

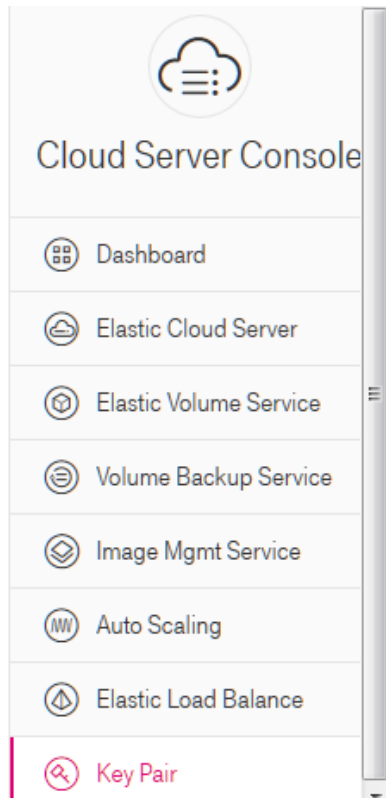
If its hard:
leave it or make it routine.

Kurt Garloff

”

EXERCISE 2: AUTOMATED INSTALL AND CONFIG

02, STEP 1: PREPARE KEY PAIR



If your ECS runs Linux, use an SSH key pair to log in to the ECS.
You can create an SSH key pair and download the private key for login. An SSH private key can be downloaded only once, so keep it secure.
Alternatively, if you already have a key pair, you can import the public key and use the private key for login. [Learn More...](#)

[+ Create SSH Key Pair](#) [Import SSH Public Key](#)

Enter a keyword. 🔍

Name	Fingerprint	Operation
SSHkey-ansible-master	8a:7f:b7:84:19:fd:b9:5d:d3:64:fa:96:7d:06:35:48	Delete

02, STEP 2: CREATE A PRIVATE SERVER

OPEN TELEKOM CLOUD



eu-de ▾ | 14611632... ▾ |

Basic Information

* ECS Name:

lab-group3-w12

If you create more than one ECS at a time, the system automatically adds a suffix to the names of those ECSs, for example, my_ECS-0001, my_ECS-0002...

* AZ:

☐ eu-de-02

☐ eu-de-01



The system automatically allocates the AZ in which to create ECSs using the current image.

* ECS Type:

☐ General-purpose

☐ Computing I

☒ Computing II

☐ Memory-optimized

☐ Disk-intensive

☐ Large-memory

☐ GPU-optimized

☐ High Performance

Provides high computing performance through a 1:2 ratio of vCPUs to memory. It is suitable for scenarios that require scaling out CPU resources, such as distributed analysis and engineering applications.

* vCPU:

1 vCPUs

2 vCPUs

4 vCPUs

8 vCPUs

16 vCPUs

32 vCPUs

* Memory:

2 GB

Selected specifications: c2.medium | 1 vCPUs | 2 GB

* Image Type:

Public Image

Private Image

Shared Image



* Image:

lab-centos7.2-nginx(4GB)



* Disk:

System Disk

Common I/O

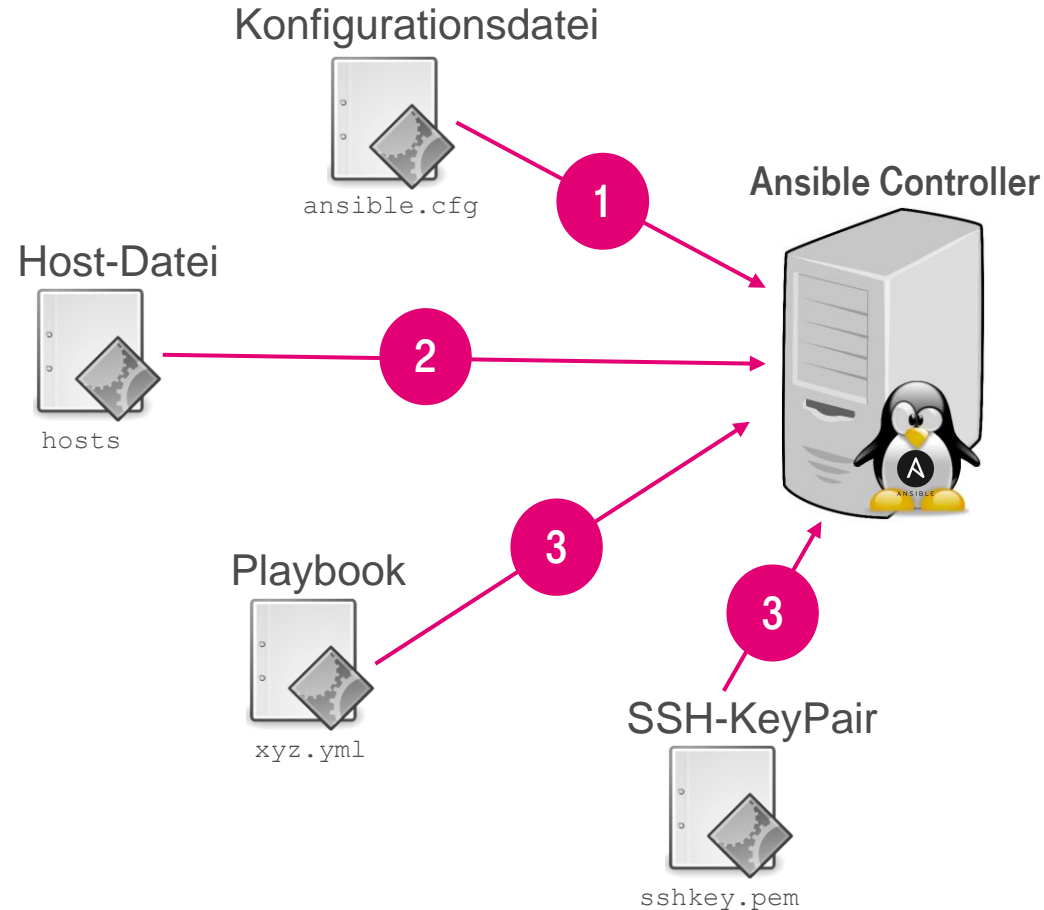
12 GB



SELECT YOUR FRESHLY
GENERATED KEYPAIR !!!

ANSIBLE – AGENTLESS CONFIGURATION MANAGEMENT

THE „SWISS ARMY KNIFE“ FOR OPS AUTOMATION



03, STEP 1: INTIAL LOGIN TO MASTER SERVER (ONLY FOR LAB, USUALLY NOT REQUIRED)

Server IP: 46.29.96.203

User: otclabX (see card)

Pass: <published in classroom>

Note: If using VPN, you may have to use a proxy (http)

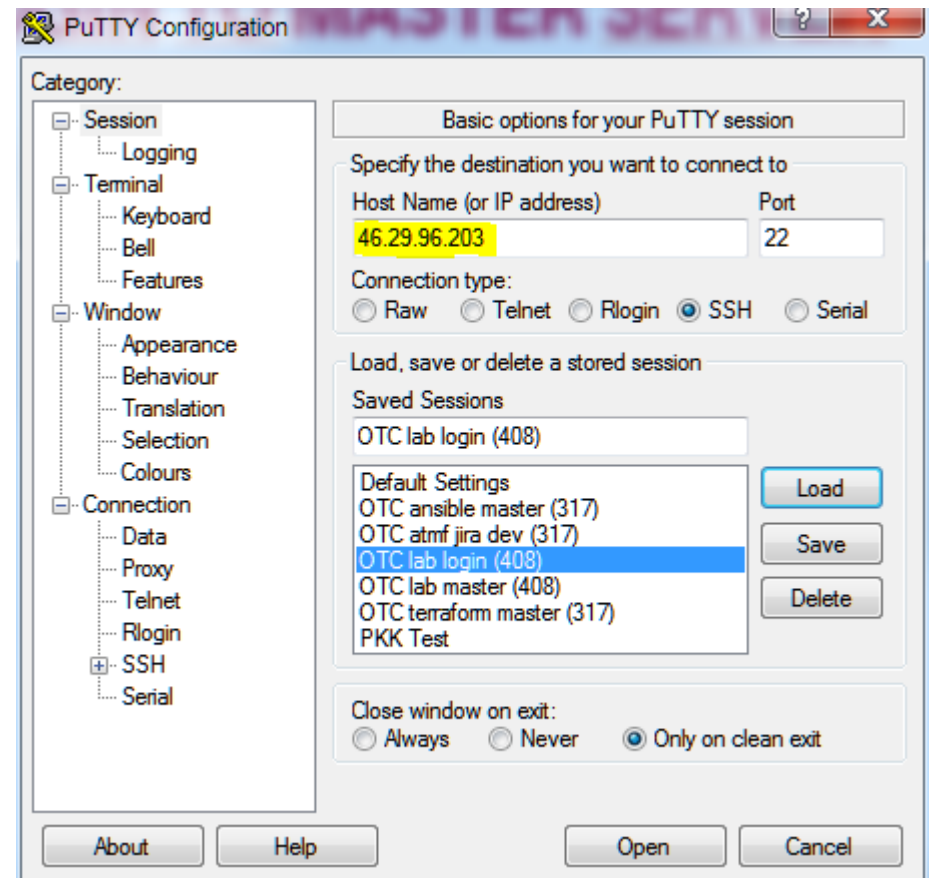
Win/putty: under Connection>Proxy

Mac/Lin: `ssh -l otclabX 46.29.96.203`

Note2: if connection breaks occur :

Win/putty: under Connection
seconds between keepalives

Mac/Lin: `-o ServerAliveInterval=10`



```
.ssh/config:
ProxyCommand nc -X connect -x proxyhost:proxyport %h %p
ServerAliveInterval 10
```

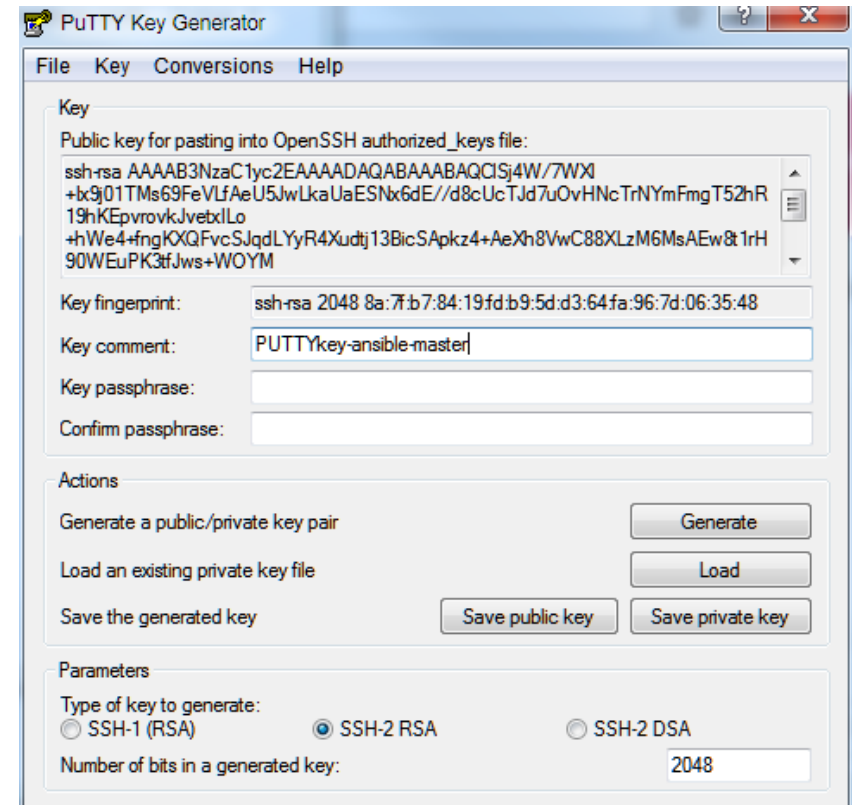
03, STEP 2: EXTRACT PUBLIC/PRIVATE KEYS FOR SSH

MAC OS (login user is „otclabX“):

```
ssh-keygen -y -f SSHkey-otclabX.pem > SSHkey-otclabX.pub  
(use output directly, add comment)
```

Windows / PUTTYgen / Putty: Edit public key

- remove heading/trailing line with ---
- add ssh-rsa at the beginning
- remove linebreaks
- add a comment

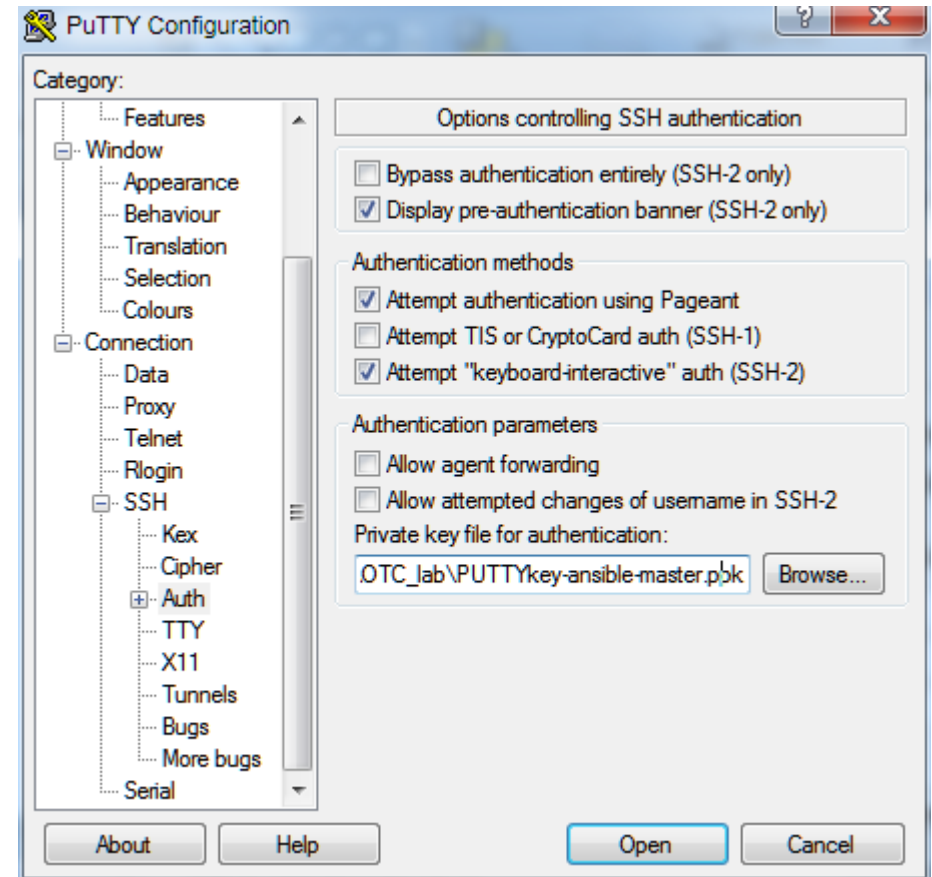


01, STEP 3: PREPARE ACCOUNT ON MASTER (ONLY FOR LAB, USUALLY NOT REQUIRED)

- **Copy** `SSHkey-otclabX.pub` **content to** `.ssh/authorized_keys`
- **Login via putty / SSH with given key** – should work
Do not forget to set login user in Putty to `otclabX`
- **Copy private key** `SSHkey-otclabX.pem` **to** `.ssh/id_rsa` **with sftp** **for private server access**
This is an easy preparation to connect private servers from here

Tests:

1. `ssh -l linux 10.128.0.x`
2. `curl http://10.128.0.x`



01, STEP 4: CLONE SOME EXAMPLE FILES, EDIT BASHRC

```
git clone https://bitbucket.org/gdccloudconsulting/devops\_lab.git
```


TASKS

- Copy `openstack/otc_certs.pem` to home top directory.
- Copy environment variables from `bashrc-example.sh` to `$HOME/.bashrc`
- Adapt variables, use the API key from your card

PLAYBOOKS, YAML, JINJA2

■ Format:

```
---  
key1: value1  
key2: "{{ jinja2_var }}"  
    subkey1: subvalue1
```

 indent 2 spaces

```
-arrayvalue1  
- arrayvalue2
```

■ Simple playbooks:

```
---  
- hosts: <inventory group>  
  vars:  
    variable1: value1  
  tasks:  
    - name:
```

SIMPLE ACTION

EXERCISE: „SERVERNAME“

Add a server-identification string and an individual color to the index.html file in your new server

INVENTORY + EXERCISE „SERVERNAME“

- Inventory file „hosts“:

```
[master]
localhost ansible_connection=local

[worker]
10.128.xx.xx
```

- Start playbook

```
> ansible-playbook -i hosts servername.yml
```

- Module documentation

<http://docs.ansible.com/ansible/index.html>

**MITTAG
ESSEN**



AGENDA

01 Intro: Continuous Delivery Pipeline revisited

02 Reproducibility: Everything as Code

02.1 Exercise: Trigger pipeline (Jenkins from GIT)

03 Provisioning time: Infrastructure as Code

03.1 Exercise: Automated install and configuration

04 Dev-Prod-parity/Immutable Infrastructures

04.1 Exercise: Image bakery

05 Conways law and continuous feedback

05.1 Exercise: ELK logging as microservices

06 Learn: Red/black, A/B and canaries

06.1 Exercise: My little A/B test

PATH 1:

Optimize flow

PATH 2:

Include feedback

PATH 3:

Learn from system

“

In the old way of doing things, we treat our servers like pets, for example Bob the mail server. If Bob goes down, it's all hands on deck. The CEO can't get his email and it's the end of the world. In the new way, servers are numbered, like cattle in a herd. For example, www001 to www100. When one server goes down, it's taken out back, shot, and replaced on the line.

Tim Bell, Cern

”

CLOUD CHARACTERISTICS REVISITED

SO MUCH THEORETICAL WORK IN THE PAST



On-
demand
self-
service

Broad
network
access

Ressource
pooling

Rapid
elasticity

Measured
service

T · · Systems ·

“

Our “holy cow” pet is Dev→PreProd→Prod
expensive, static environments.

With cloud kettles,
you get the environment you need when you need it,
and throw it away when you are done.

B. Rederlechner

”

THROUGHPUT LIMITED BY DEFECTS/ERROR RATE



THE TWELVE-FACTOR APP

“

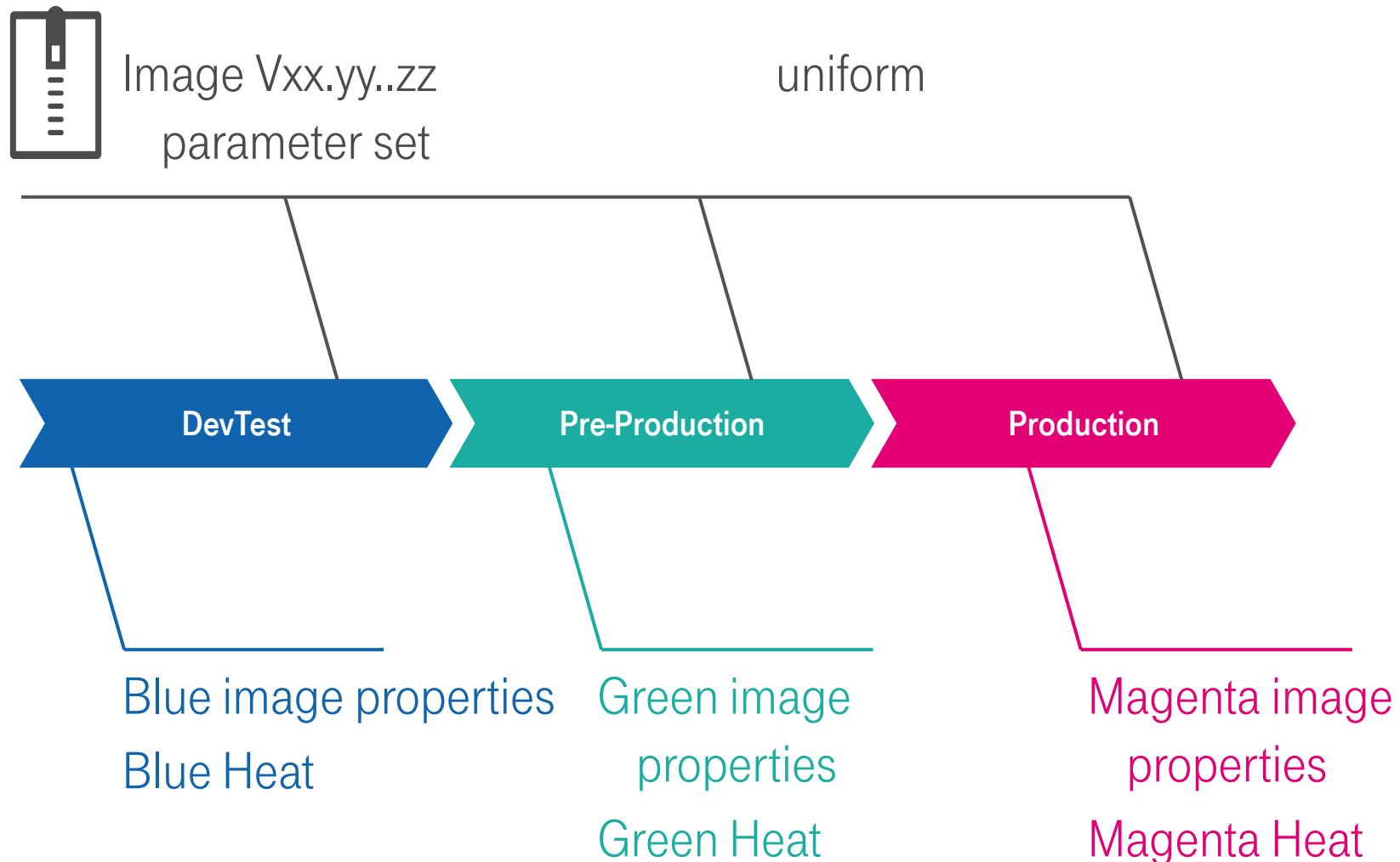
DEV/PROD PARITY:

Keep development, staging, and production as similar as possible.

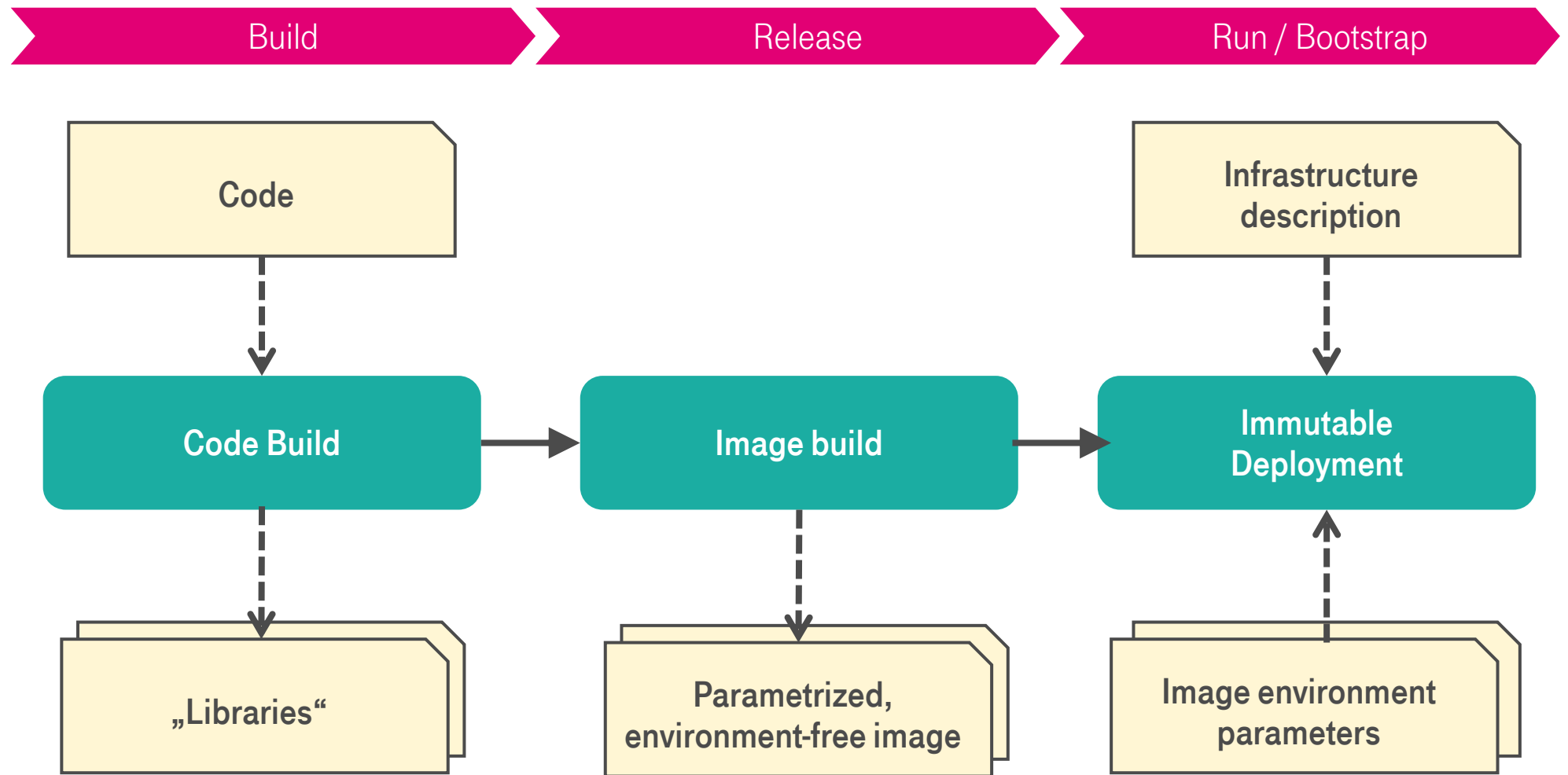
”

BUILD IMAGE ONCE – RUN ANYWHERE

MAXIMIZE DEV/PROD PARITY



INFRASTRUCTURE AS CODE & IMMUTABLE INFRASTRUCTURES



“

Deliver fast, fail fast.

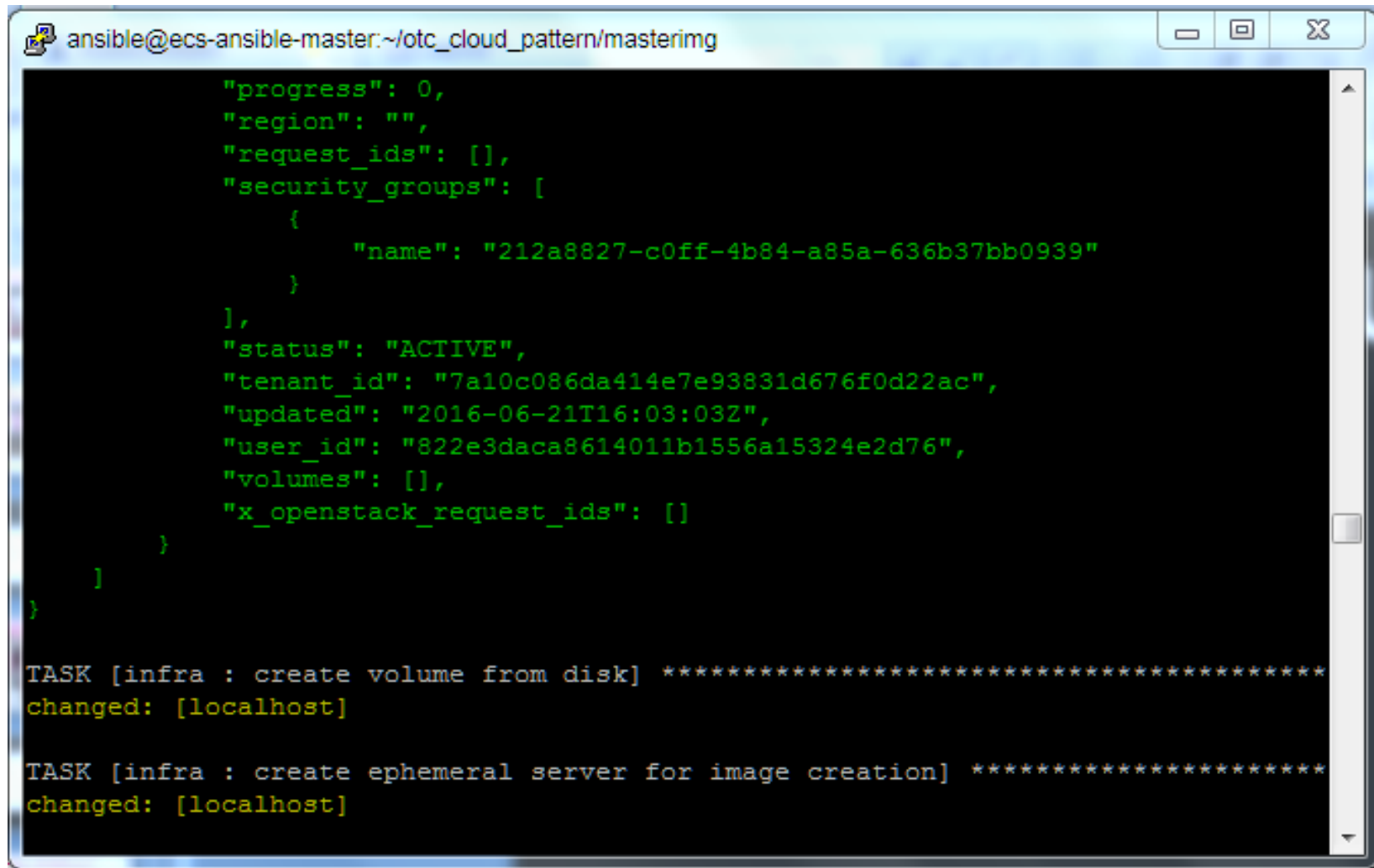
Fowler, Gray, Shore

”

EXERCISE/VIDEO 3: THE IMAGE BAKERY

IMAGE BAKERY

SEE ALSO „03_MASTERIMG“ EXERCISE



```
ansible@ecs-ansible-master:~/otc_cloud_pattern/masterimg

    "progress": 0,
    "region": "",
    "request_ids": [],
    "security_groups": [
      {
        "name": "212a8827-c0ff-4b84-a85a-636b37bb0939"
      }
    ],
    "status": "ACTIVE",
    "tenant_id": "7a10c086da414e7e93831d676f0d22ac",
    "updated": "2016-06-21T16:03:03Z",
    "user_id": "822e3daca8614011b1556a15324e2d76",
    "volumes": [],
    "x_openstack_request_ids": []
  }
]
}

TASK [infra : create volume from disk] *****
changed: [localhost]

TASK [infra : create ephemeral server for image creation] *****
changed: [localhost]
```

ANTI-PATTERN FOR BAKED IMAGES AND EPHEMERAL SERVERS

No environment context within image:

Initialisation script in image:

Wrong:

```
if ( environment == `test` ) // enable some interface
```

Better: Functional switches

```
if ( interface_enabled == true ) // enable some interface
```

Reason: Setup decisions should not be hardcoded.

Dynamic, initial image configuration in a setup repository/DB:

Reason: If setup repository DB FAILS, NOTHING will boot

→ repository /DB becomes role „kernel“

→ must be ultra-available, but with hardcoded configuration

State-of-the-art: config by environment variables

AGENDA

01 Intro: Continuous Delivery Pipeline revisited

02 Reproducibility: Everything as Code

02.1 Exercise: Trigger pipeline (Jenkins from GIT)

03 Provisioning time: Infrastructure as Code

03.1 Exercise: Automated install and configuration

04 Dev-Prod-parity: Immutable Infrastructures

04.1 Exercise/Video: Image bakery

05 Conways law and continuous feedback

05.1 Exercise: ELK logging as microservices

06 Learn: Red/black, A/B and canaries

06.1 Exercise: My little A/B test

PATH 1:

Optimize flow

PATH 2:

Include feedback

PATH 3:

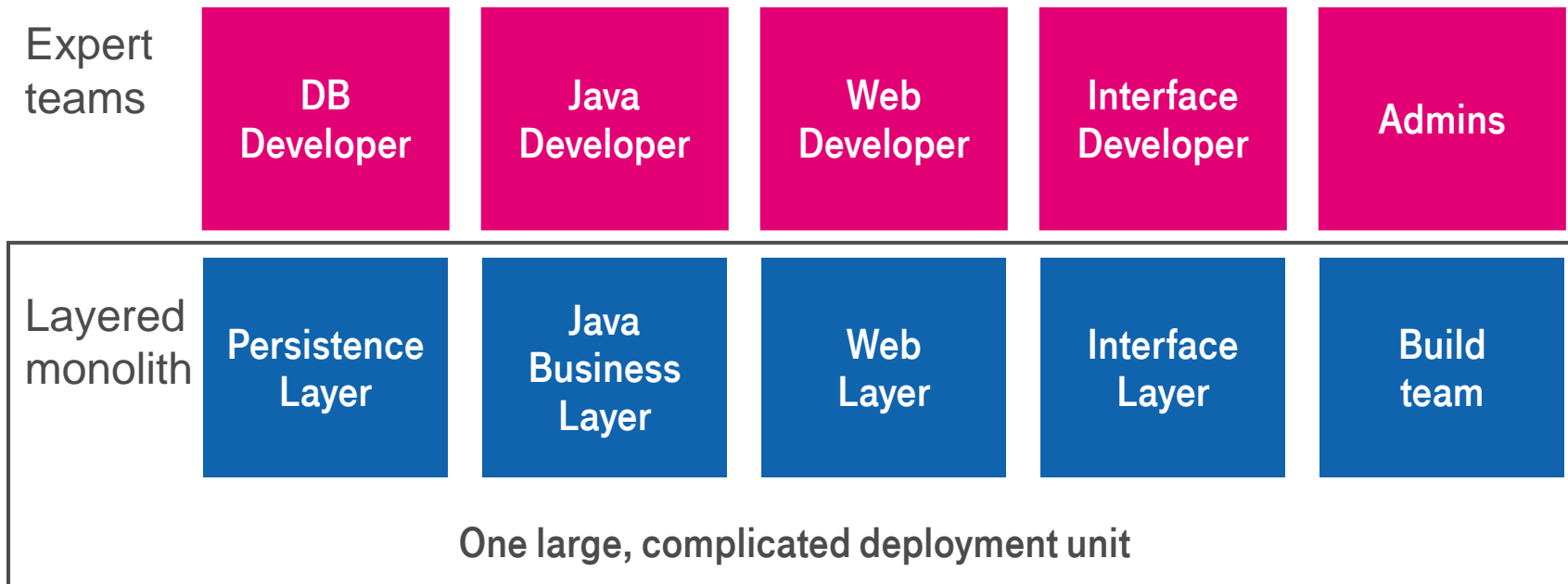
Learn from system

THROUGHPUT LIMITED BY #TEAMS LIMITED BY COST



CONWAYS LAW: EXPERT TEAMS VS. CROSS FUNCTIONAL TEAMS

“ Any organization that designs a system
(defined more broadly here than just information systems)
will inevitably produce a design whose structure is
a copy of the organization's communication structure. ”



CROSS-FUNCTIONAL DEVOPS TEAMS & MICROSERVICES

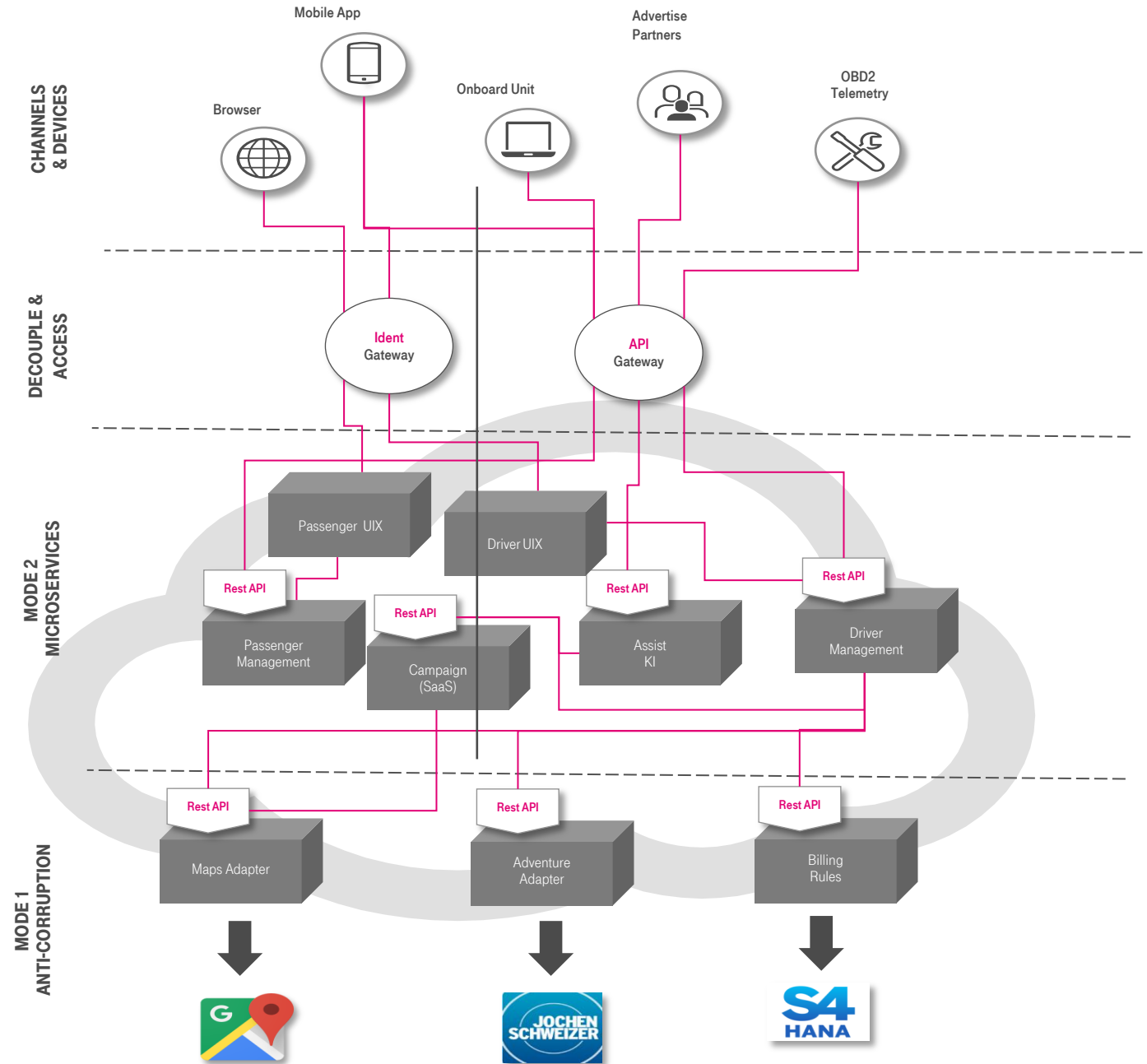
ARCHITECTURE – ORGANISATION COMPATIBILITY

A **Microservice** is a well-defined part of a complex software system with the following properties:

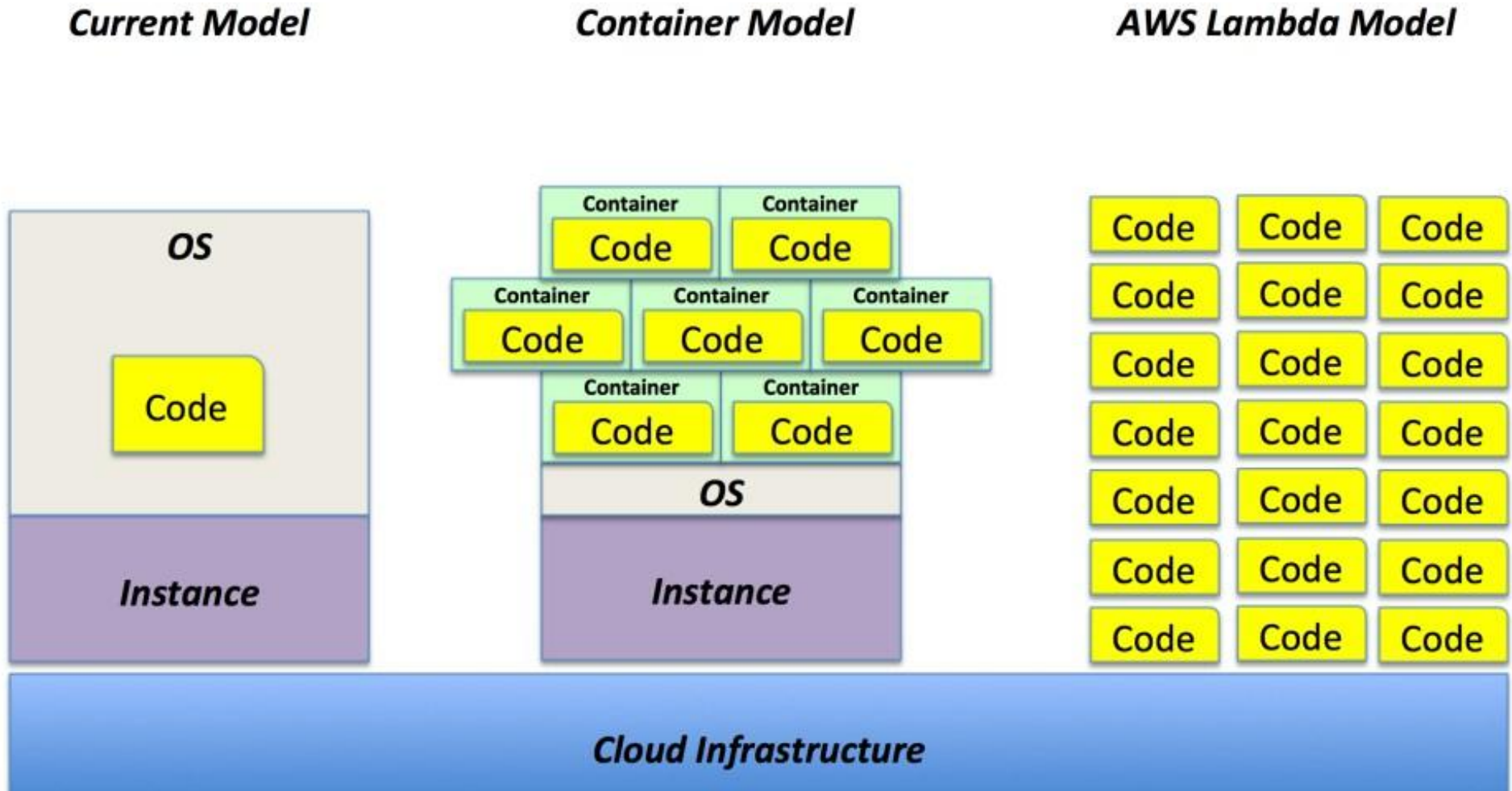
- It represents a dedicated, **bounded-business context** [DDD]
- It is an **independent-deployable package**.
- It delivers **a defined service independent of its consumers** and their bounded-contexts. [SOA]
- It is **consumed only by a well-defined, formal API**. [API]
- It is **delivered and operated by** an independent, **cross-functional team** (= „a team with all necessary skills to deliver and run“) [DevOps]

MICROSERVICE ARCHITECTURE

Complexity
by
distributed system/
call dependencies
is the price to pay.



RESSOURCE USAGE & PACKAGE DENSITY



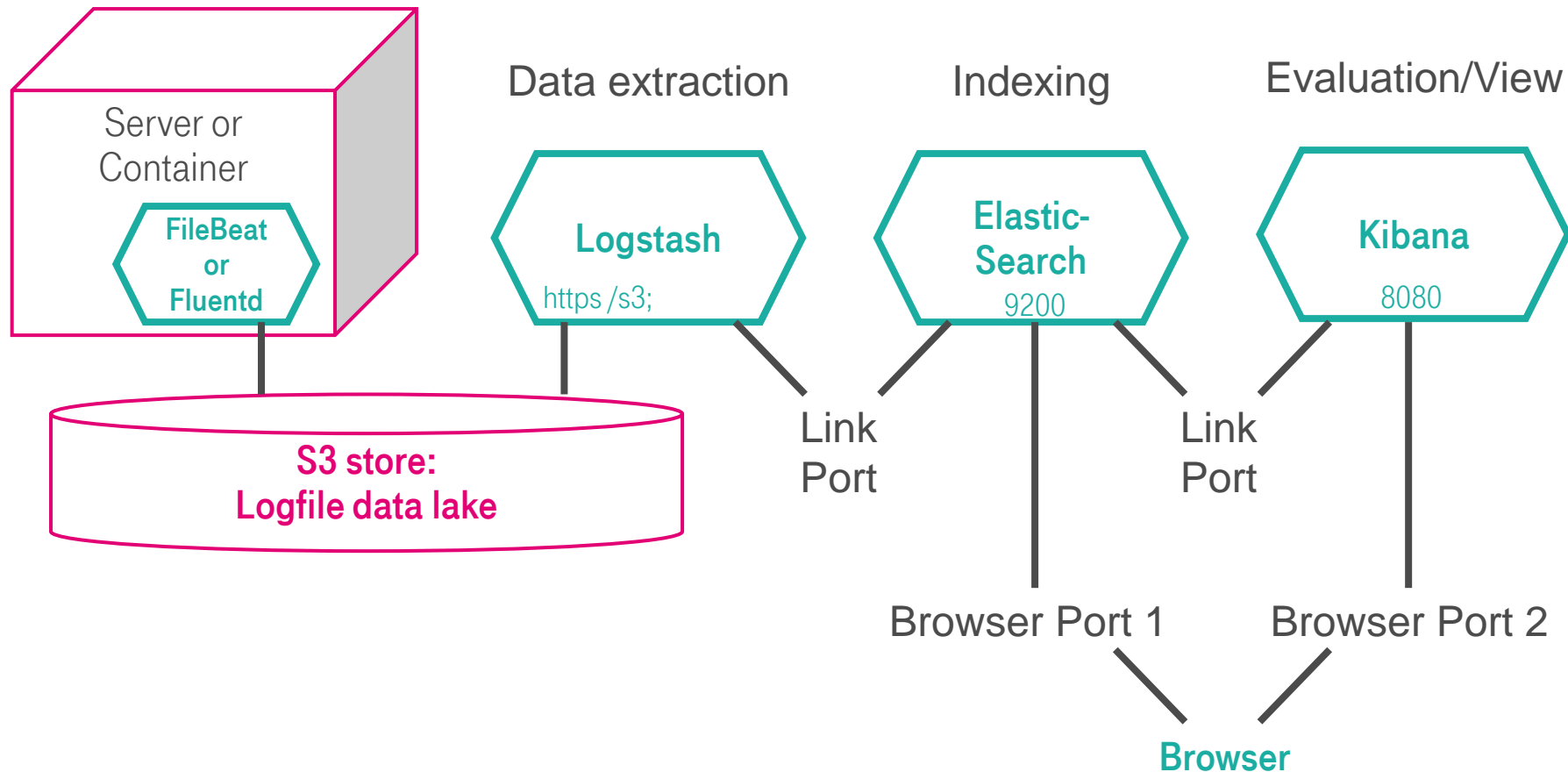
<http://it20.info/2014/12/cloud-native-applications-for-dummies/>

EXERCISE 4: ELK LOGGING AS MICROSERVICES

[HTTPS://BITBUCKET.ORG/GDCLOUDCONSULTING/OTC_CCE_PATTERN](https://bitbucket.org/gdcloudconsulting/otc_cce_pattern)

ELASTICSEARCH – LOGSTASH – KIBANA

MICROSERVICE EXAMPLE AND EVALUATION PLATFORM



STEP 1: BUILD DOCKER IMAGES

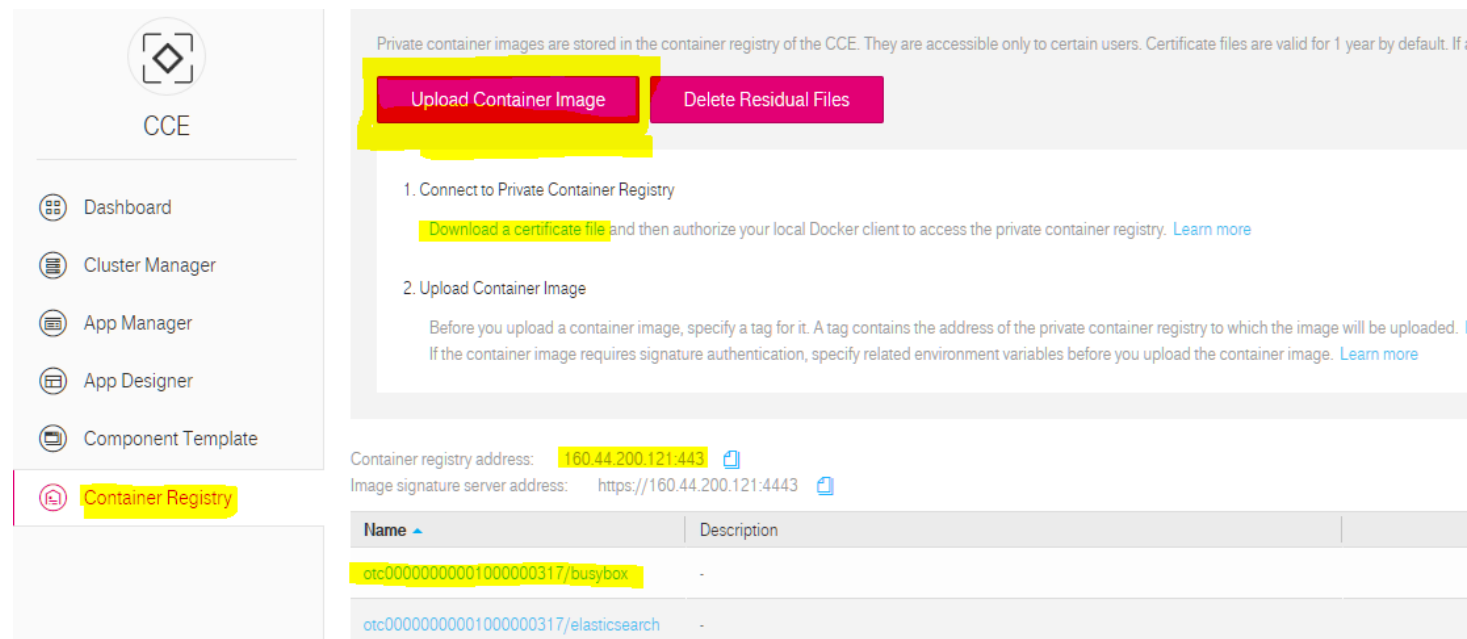
https://bitbucket.org/gdccloudconsulting/otc_cce_pattern/src/13f6d20ff5e9b1f94298c61ebc503aac7ddcc088/cce-master/?at=master

Download dockercfg.txt certificate from OTC CCE user interface, Container Registry. Copy the content to `~/ .docker/config.json`.

Note: Remove any linebreaks in file, everything MUST be in one line!

Add the public IP address of your private registry to `/etc/docker/daemon.json`:

```
{ "insecure-registries": ["160.44.200.121:443"] } -- already perpared for you!
```



Private container images are stored in the container registry of the CCE. They are accessible only to certain users. Certificate files are valid for 1 year by default. If:

Upload Container Image **Delete Residual Files**

1. Connect to Private Container Registry

Download a certificate file and then authorize your local Docker client to access the private container registry. [Learn more](#)

2. Upload Container Image

Before you upload a container image, specify a tag for it. A tag contains the address of the private container registry to which the image will be uploaded. If the container image requires signature authentication, specify related environment variables before you upload the container image. [Learn more](#)

Container registry address: **160.44.200.121:443** [Copy](#)

Image signature server address: <https://160.44.200.121:4443> [Copy](#)

Name	Description
otc0000000001000000317/busybox	-
otc0000000001000000317/elasticsearch	-

BUILDING THE FIRST IMAGE AND UPLOAD TO REGISTRY

Build base image and push

```
$ cd java-centos-base

$ docker build -t 160.44.200.121:443/otc-eu-de-00000000001000023731/otclabXX-java8-centos7-base:1.8 .

$ docker push 160.44.200.121:443/otc-eu-de-00000000001000023731/otclabXX-java8-centos7-base:1.8
```

SETUP KUBERNETES ACCESS, INSTALL SERVICES

```
$ cd exercises/04_elk_obs/bin
```

Adapt otc_env.sh, use your credentials for Open Telekom Cloud

```
$ . ./otc_env.sh
$ ./cce_kubeaddclusters
$ kubectl config set-context dop-cluster01-ctx --namespace=otclabXX
$ kubectl config set current-context dop-cluster01-ctx
```

Remember to adapt Namespace to „otclabXX“ and image name in yaml files.

Install secret:

```
$ echo {"160.xx.xxx.xxx:443":{"auth":"X2xxxxxxxx ....xxx==","email":""}} | base64 -w 0
from $HOME/.docker/config.json, without {"auths":{"160.xx.xxx.xxx:443":{"auth":"X2xxxxxxxx ....xxx==","email":""}}}

$ vi otclabXX-pull-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: otclabXX-secret
data:
  .dockercfg: <paste base64 encoded secret here>
type: kubernetes.io/dockercfg

$ kubectl create -f ./my-pull-secret.yaml
```

KUBERNETES: INSTALL SERVICES

Set your own NodePort in dev/kub-kibana-service.yaml

(300XX for your otclab nr)

```
$ cd elasticsearch
$ kubectl create -f dev/kub-elasticsearch-service.yaml
$ kubectl create -f dev/kub-elastictransport-service.yaml

$ cd kibana
$ kubectl create -f dev/kub-kibana-service.yaml
```

Review in Kubernetes web view:

```
Start your ssh with Tunnel option -L localhost:80XX:localhost:80xx
Do not use „127.0.0.1“ !

$ kubectl proxy --port=80XX &
```

Open your browser with localhost:80XX

INSTALL ELASTICSEARCH AND KIBANA

Remember to adapt imagename to your own images in Dockerfile:

FROM 160.44.200.121:443/otc-eu-de00000000001000023731/otclabXX-...

Elasticsearch:

```
$ cd elasticsearch

$ docker build -t 160.44.200.121:443/otc-eu-de-00000000001000023731/otclabXX-elasticsearch:5.6.1 .
$ docker push 160.44.200.121:443/otc-eu-de-00000000001000023731/otclabXX-elasticsearch:5.6.1

$ kubectl create -f dev/kub-elasticsearch-deploy.yaml
```

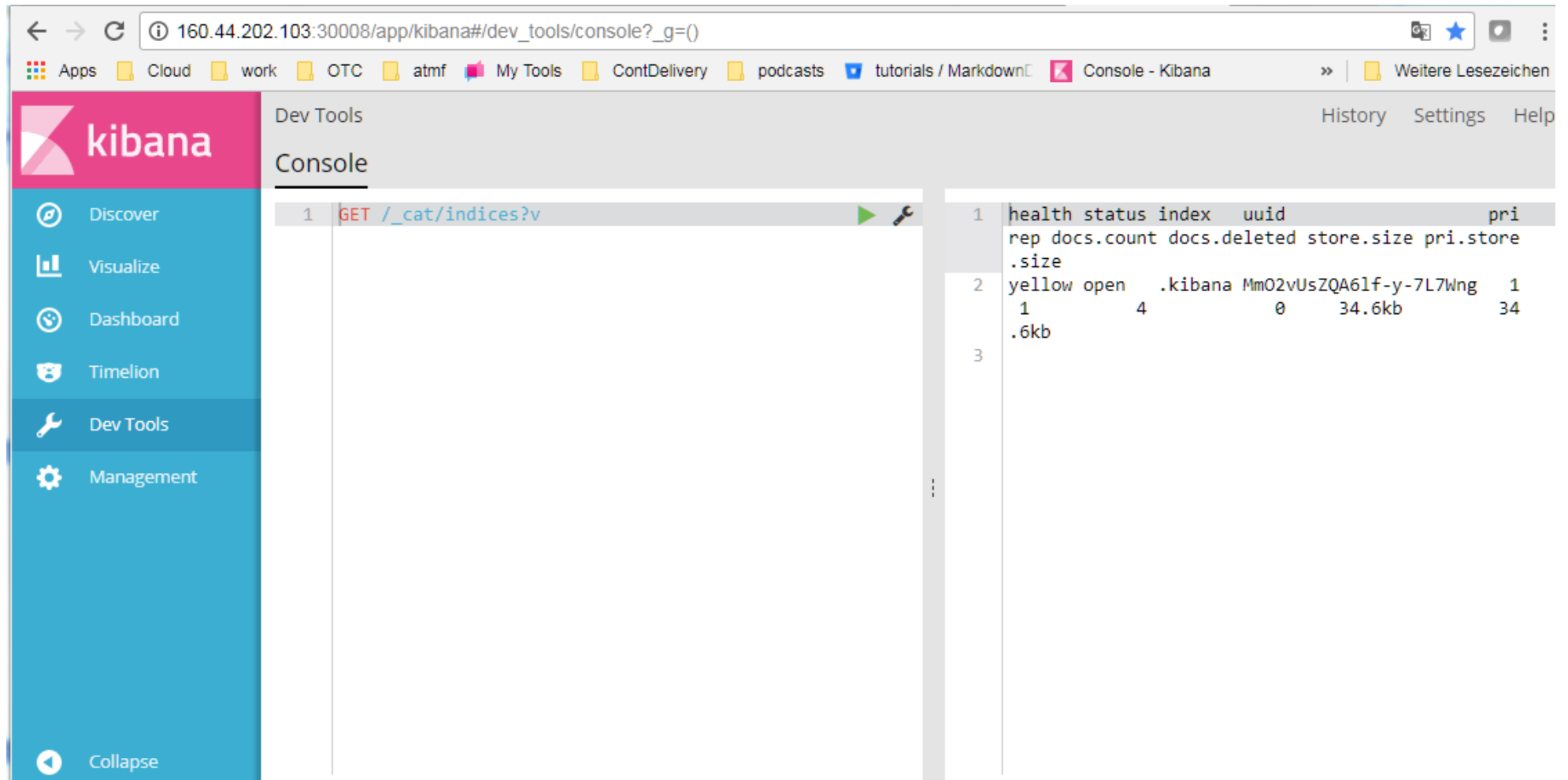
Kibana:

```
$ cd elasticsearch

$ docker build -t 160.44.200.121:443/otc-eu-de-00000000001000023731/otclabXX-kibana:5.6.1 .
$ docker push 160.44.200.121:443/otc-eu-de00000000001000023731/otclabXX-kibana:5.6.1

$ kubectl create -f dev/kibana/kub-kibana-deploy.yaml
```



CHECK FOR RUNNING KIBANA







The screenshot shows the Kibana web interface with the Dev Tools console open. The console displays a successful GET request to the Elasticsearch `/_cat/indices?v` endpoint. The response is a table of index information.

1	health	status	index	uuid	pri
1	rep	docs.count	docs.deleted	store.size	pri.store
2	yellow	open	.kibana	Mm02vUsZQA61f-y-7L7Wng	1
3	1	4	0	34.6kb	34
			.6kb		


GET KEY FOR OBJECTSTORE

 ERLEBEN, WAS VERBINDET.

OPEN TELEKOM CLOUD  

dop-brederle |  Message | My Quota |  Help Center

My Credential [Learn more](#)


[Change](#)

User Name: dop-brederle

User ID: 55b2dc0fdad7419da2fa61f1b1ae63fd

Domain Name: OTC-EU-DE-00000000001000023731

Domain ID: e779432fe7834d5d937e791bcc0900e1


Verified Email Address: d***e@t-online.de [Modify](#)

Mobile Number: -- [Modify](#)


Password: SecurityStrong ■■■■ [Modify](#) ("API Key" is now "Password". [Learn more](#))

Verify Login by SMS Message: This function cannot be enabled. Bind a mobile number and email address.

Project List | **Access Keys**



Access Key ID ▾	Created ▾	Status ▾	Operation
X2VEDEDKJ02DXBUESBXA	10/21/2017 13:30:20 GMT+02:00	Active	Delete

 [Add Access Key](#) (You can add 1 sets of access keys.)

OPTIONAL: UPLOAD FILES TO OBJECTSTORE

```
$ s3cmd --configure  
  
$ vi ~/.s3cfg  
host_base = „https://obs.eu-de.otc.t-systems.com“  
  
$ s3cmd --configure  
  
$ s3cmd ls  
  
$ s3cmd put -ssl logs.jsonl s3://my-bucket-name
```

https://docs.otc.t-systems.com/en-us/doc/pdf/20170829/20170829170403_50525.pdf

INSTALL LOGSTASH

Edit conf/*.conf:

```
input {  
  s3 {  
    access_key_id => „<my AK from OTC>”  
    secret_access_key => „<my SK from OTC>”  
    endpoint => "https://obs.eu-de.otc.t-systems.com"  
    bucket => "85bblog"  
  }  
}
```

Install logstash:

```
$ cd logstash  
  
$ docker build -t 160.44.200.121:443/otc-eu-de-00000000001000023731/otclabXX-  
logstash:5.6.1 .  
$ docker push 160.44.200.121:443/otc-eu-de00000000001000023731/otclabXX-logstash:5.6.1  
  
$ kubectl create -f dev/kibana/kub-logstash-deploy.yaml
```

TIP: KUBERNETES DEBUG QUICKIES

Quick start of own Docker image in registry:

```
kubect1 run -ti myjavabox --image=160.44.200.121:443/otc00000000001000000317/java-centos7-base:0.1 -- bash
```

Quick start of a bare client linux box within cluster (e.g. to test IP connections)

```
kubect1 run -ti mybox --image=busybox -- sh
```

Connect to a running pod:

```
kubect1 exec -ti <pod-id> -- bash
```

Kubernetes cheat sheet:

<https://kubernetes.io/docs/user-guide/kubect1-cheatsheet/>

**KAFFEE-
PAUSE**



AGENDA

01 Intro: Continuous Delivery Pipeline revisited

02 Reproducibility: Everything as Code

02.1 Exercise: Trigger pipeline (Jenkins from GIT)

03 Provisioning time: Infrastructure as Code

03.1 Exercise: Automated install and configuration

04 Dev-Prod-parity: Immutable Infrastructures

04.1 Exercise/Video: Image bakery

05 Conways law and continuous feedback

05.1 Exercise: ELK logging as microservices

06 Learn: Red/black, A/B and canaries

06.1 Exercise: My little A/B test

PATH 1:

Optimize flow

PATH 2:

Include feedback

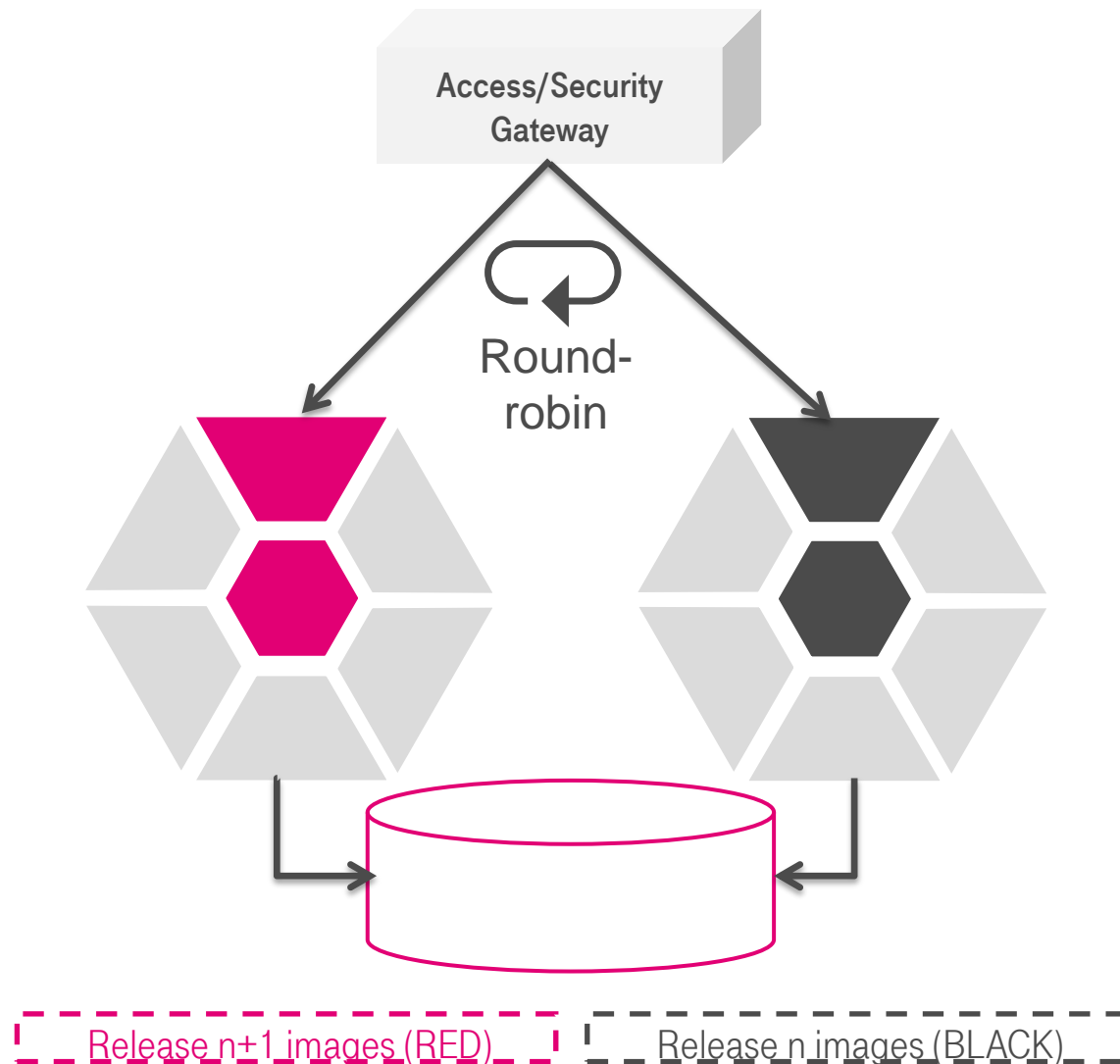
PATH 3:

Learn from system

LOGGING REVISITED

LOG LEVEL	CONTENT	MAIN PURPOSE
Fatal, Error, Warning	Exceptions Unexpected behaviour	Alarming [prod enabled]
INFO	Business events Business correlation ID	Business Learning [prod enabled]
DEBUG, TRACE	Developer details	Debugging [prod disabled]

ROLLING UPDATE, RED/BLACK DEPLOYMENT



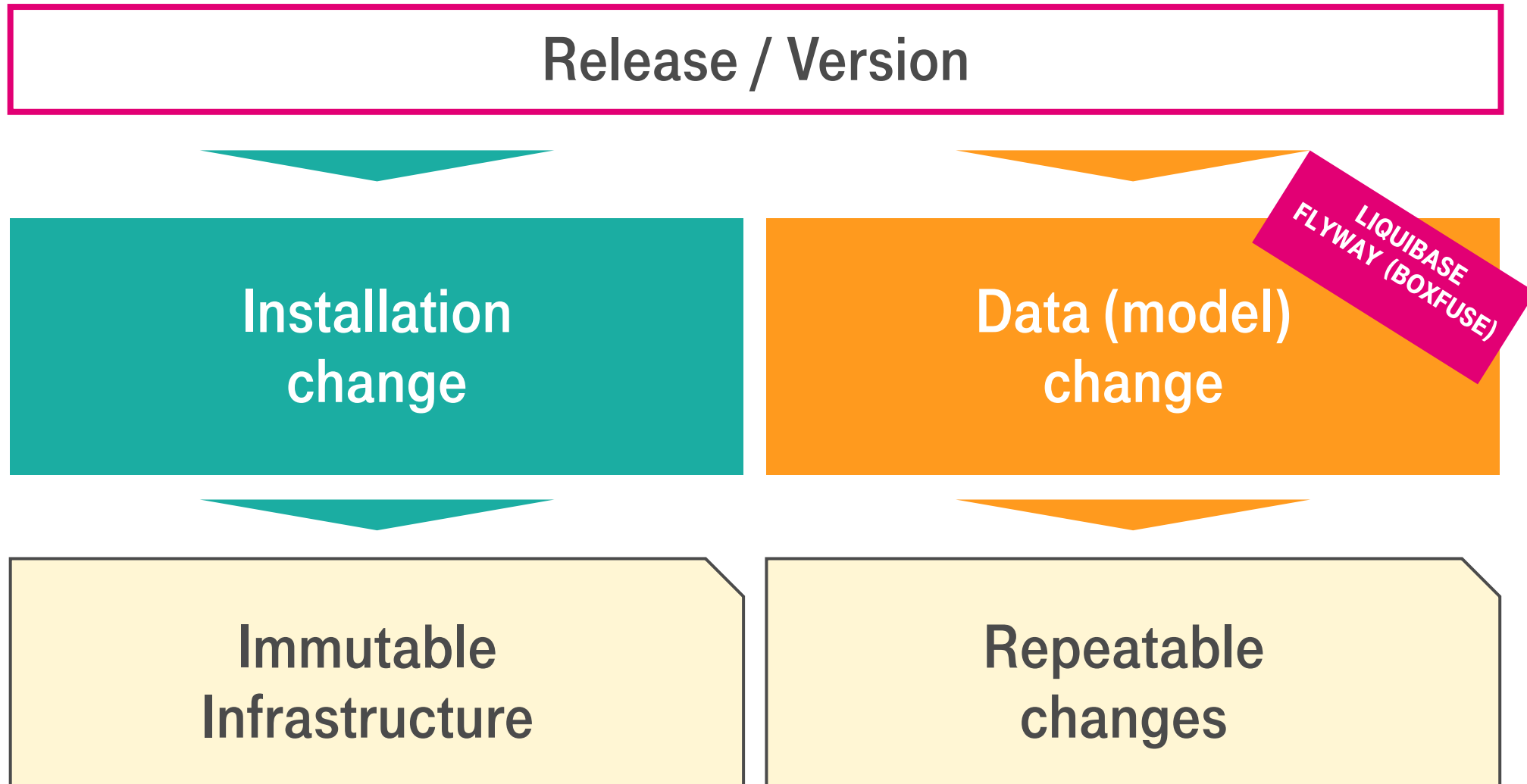
TIP: VERSIONING AND ROLLING UPDATES

[Product](#)[Pricing](#)[Customers](#)[Resources](#)[Request Demo](#)[Free Trial](#)

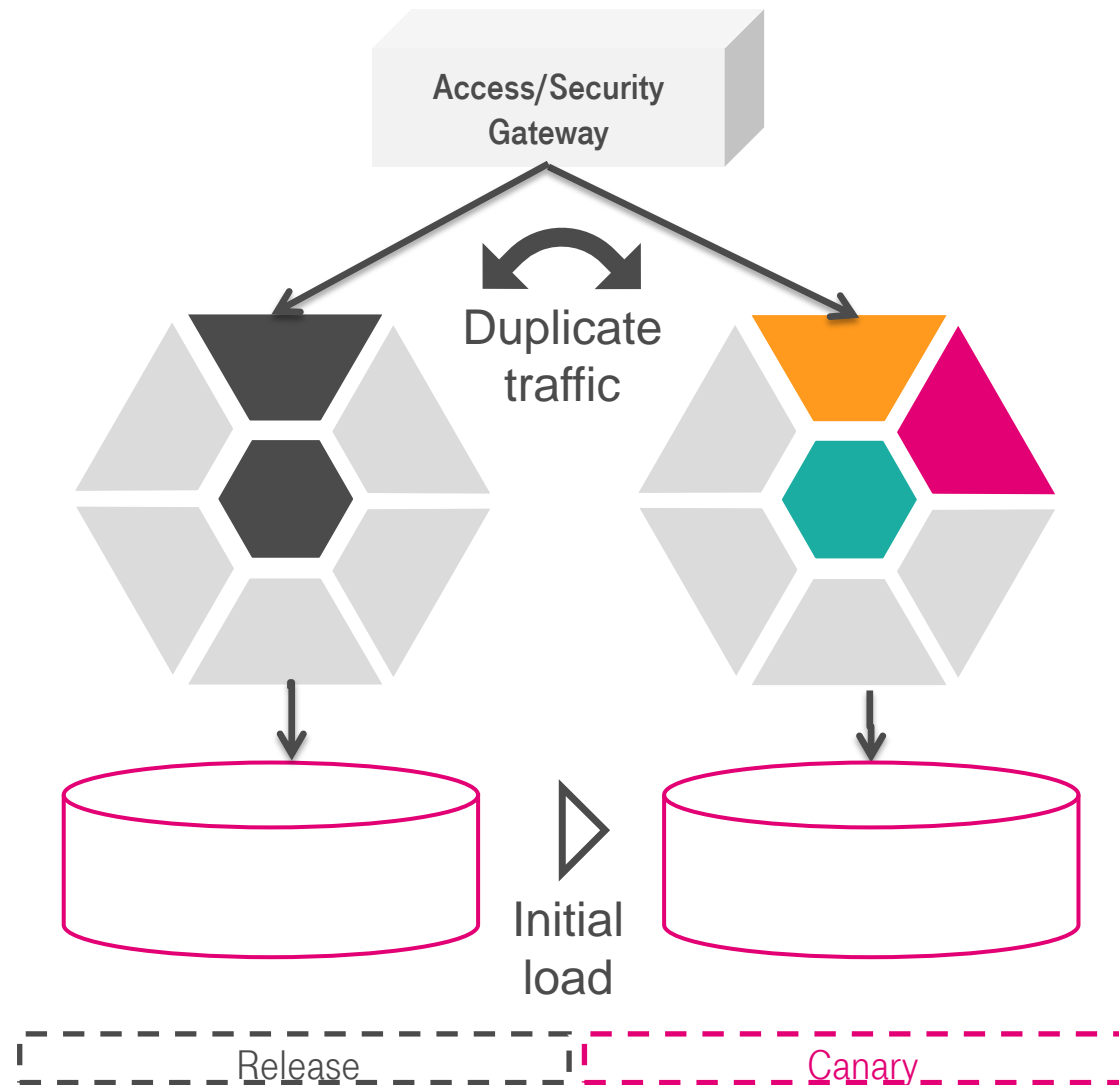
Reading Before Your Update

It's crucial to read before you start to plan. First, look at the Elasticsearch documentation [relevant for upgrades](#) — it's pretty straightforward. But remember this rule-of-thumb: Minor version changes (from 2.X to 2.Y) support **rolling upgrades** (one node at a time), but major version updates (from 1.X to 2.X) **require full cluster restarts**.

RELEASES: DON'T FORGET YOUR DATA CHANGES!



CANARY RELEASES



FEATURE TOGGLES (IN PRODUCTION)

```
export FEAT1_ENABLED=true
export FEAT2_ENABLED=false
export FEAT3_ENABLED=true
export FEAT4_ENABLED=true
export FEAT5_ENABLED=false
export FEAT6_ENABLED=true
export FEAT7_ENABLED=false
export FEAT8_ENABLED=false
...
```

Inject

ECS or Container



```
if ( ${FEAT4_ENABLED} == true ) {
    // execute feature
}
```

A/B TESTS IN PRODUCTION

```
export FEAT1_RATIO=0.6
export FEAT2_RATIO=0.3
export FEAT3_RATIO=0.0
export FEAT4_RATIO=0.4
export FEAT5_RATIO=0.0
export FEAT6_RATIO=0.5
export FEAT7_RATIO=1.0
export FEAT8_RATIO=0.5
...
```

Inject

ECS or Container



```
if (random() > ${FEAT4_RATIO}) {
  log(`FEAT4,variant A applied`)
  // execute old feature
} else {
  log(`FEAT4,variant B applied`)
  // execute new feature
}
```


6.1 LEARN: MY LITTLE A/B TEST (A FREE EXERCISE)

THANK YOU!

BACKUP

INVENTORY

Partially done work!

T · · Systems ·

OVERPRODUCTION

**Extra features,
Unused features!**

T · · Systems ·

EXTRA PROCESSING

Relearning!

T · · Systems ·

TRANSPORTATION

Handoffs!

T · · Systems ·

WAITING

Delays!

T · · Systems ·

MOTION, JUMPING

Task switching!

T · · Systems ·

DEFECTS

Defects!

T · · Systems ·

THE SEVEN WASTES OF SOFTWARE DEVELOPMENT

#1 - Partially Done Work (Inventory)

#2 - Extra Features, unused features (Overproduction)

#3 – Relearning (Extra processing)

#4 – Handoffs (Transportation)

#5 – Delays (Waiting)

#6 - Task Switching (Motion, Jumping)

#7 - Defects