

Speaker: Marcel Bernet

# Aufbau einer Continuous Integration / Delivery Pipeline mit Docker/Kubernetes



# Vorstellung Marcel Bernet



## ★ Marcel Bernet

- IT-Consultant, IT-Architekt (Beratung, Schulung)
- Seit über 20 Jahren Selbständig.
- Themenkreise: Digitalisierung, Internet of Things, Big Data, Machine Learning, Microservices, DevOps, Containerisierung
- Ehemaliges Mitglied Expertenkommissionen swissICT, eCH
- Ehemaliger CH-open Präsident
- <https://github.com/mc-b/>

# Agenda



- ★ Docker und Kubernetes
- ★ Continuous Integration (CI) / Continuous Delivery/Deploy (CD) / DevOps
- ★ Pipeline
- ★ Rolling Updates (CD manuell)
- ★ Continuous Delivery/Deploy (CD) automatisiert mittels Branches

# Docker und Kubernetes

Platform / Infrastructure

# Kubernetes (K8s) im Überblick



- ★ Das im Juli 2014 gestartete (griechisch: Steuermann) stellt die derzeit populärste Container-Cluster-/Orchestrierungs-Lösung dar.
- ★ Kubernetes ist mittlerweile bei der Cloud Native Computing Foundation (CNCF) gehostet.
- ★ **Kubernetes' Hauptaufgabe ist die Verwaltung und Orchestrierung der Container innerhalb eines Clusters, der üblicherweise aus mindestens einem Kubernetes Master und multiplen Worker Nodes besteht.**
- ★ Kubernetes wurde von Google ursprünglich als Orchestrierungs-Framework unter der Code-Bezeichnung Borg initiiert. Ziel war es, Docker- oder Rocket-(CoreOS)-Container im grossem Umfang zu deployen, zu administrieren und transparent zu orchestrieren.
- ★ Da Google Kubernetes bzw. seine Google-interne Implementierung davon selbst einsetzt, ist relativ sicher, dass K8s kein kurzlebiges Projekt sein wird.
- ★ **K8s arbeitet primär mit Docker-Containern.**

Quelle: Buch Skalierbare Container-Infrastrukturen, Kapitel 13

# Kubernetes (K8s) Merkmale



- ★ Immutable (Unveränderlich) statt Mutable.
- ★ **Deklarative** statt Imperative (Ausführen von Anweisungen) **Konfiguration**.
- ★ **Selbstheilende Systeme - Neustart bei Absturz**.
- ★ Entkoppelte APIs – LoadBalancer / Ingress ([Reverse Proxy](#)).
- ★ **Skalieren (Vertikal/Horizontal) der Services** durch Änderung der Deklaration.
- ★ Anwendungsorientiertes statt Technik (z.B. Route 53 bei AWS) Denken.
- ★ **Abstraktion der Infrastruktur** statt in Rechnern Denken.

# Deklaration: YAML (Dateien)

```
apiVersion: v1
kind: Service
metadata:
  name: kafka
  labels:
    app: kafka
    name: kafka
spec:
  ports:
    - port: 9092
  selector:
    app: kafka
    tier: middleware
clusterIP: None
```

```
apiVersion: apps/v1beta2 # for versions before
kind: Deployment
metadata:
  name: kafka
  labels:
    app: kafka
spec:
  selector:
    matchLabels:
      app: kafka
      tier: middleware
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: kafka
        tier: middleware
    spec:
      containers:
        - name: kafka
          image: confluentinc/cp-kafka:latest
          imagePullPolicy: IfNotPresent
```

- ★ YAML ist eine vereinfachte Auszeichnungssprache (englisch markup language) zur Datenserialisierung, angelehnt an XML (ursprünglich) und an die Datenstrukturen in den Sprachen Perl, Python und C sowie dem in RFC2822 vorgestellten E-Mail-Format.
- ★ Die grundsätzliche Annahme von YAML ist, dass sich jede beliebige Datenstruktur nur mit assoziativen Listen, Listen (Arrays) und Einzelwerten (Skalaren) darstellen lässt. Durch dieses einfache Konzept ist YAML wesentlich leichter von Menschen zu lesen und zu schreiben als beispielsweise XML, ausserdem vereinfacht es die Weiterverarbeitung der Daten, da die meisten Sprachen solche Konstrukte bereits integriert haben.

# Befehl um YAML Dateien Auszuwerten



- ★ Das **kubectl**-Kommando stellt, eine der Schaltzentralen des K8s Clusters zur Administration der Ressourcen dar.
- ★ Die wichtigsten kubectl-Subkommandos in der Übersicht
  - `kubectl create -f YAML-Datei` – Erstellt eine Ressource laut YAML Datei
  - `kubectl run` – startet ein Container Image
  - `kubectl get [all]` – zeigt Ressourcen an
  - `kubectl explain [pods]` – zeigt die Dokumentation zu einer Ressource an
  - `kubectl delete -f YAML-Datei` – Löscht eine Ressource laut YAML Datei
  - `kubectl exec` – startet einen Befehl im Container
  - `kubectl apply -f YAML-Datei` – führt die Änderungen in der YAML im Cluster nach
  - `kubectl cluster-info` – Liefert Informationen zum K8s Cluster



# Live Session: Docker und Kubernetes

DEV {</>  
Day

by digicomp

- ★ Alle Beispiele laufen auf einer Linux Umgebung (in einer Virtuellen Maschine) mit installierter «Internet of Things», «Machine Learning», «Data Processing», Docker und Kubernetes Infrastruktur.

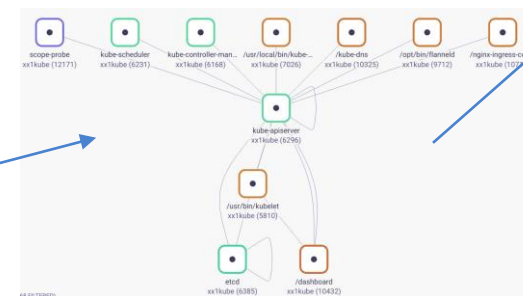
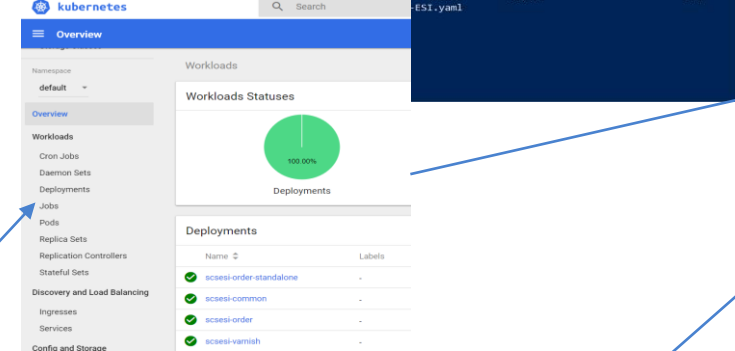
- ★ **Kommandozeile (CLI)\*:** Starten mittels **kubeps.bat (PowerShell)** oder **kubesh.bat (Bash)**, im Verzeichnis lernkube, und Hilfsprogramm **kubectl** zum Erstellen, Start UI, Löschen von Services, z.B.:

- `kubectl apply -f duk/osticket`
- `startsrv osticket`
- `kubectl delete -f duk/osticket`

- ★ **Dashboard:** Starten mittels **dashboard.bat**. Details siehe <https://github.com/kubernetes/dashboard>

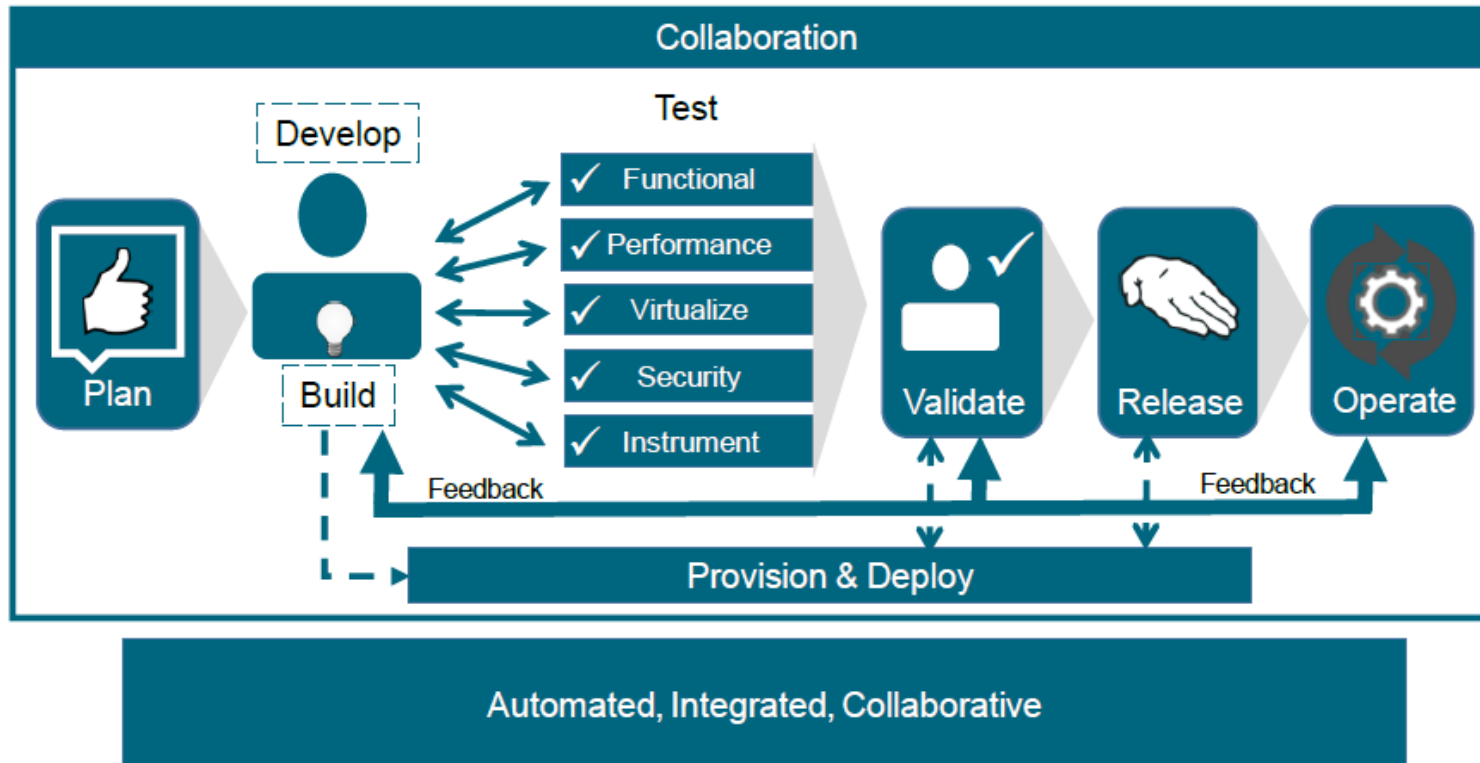
- ★ **Weave Scope:** Starten in CLI mittels **weave.bat**. Details siehe: <https://github.com/weaveworks/scope>

```
PS C:\Users\Public\Desktop\misegr> kubectl create -f .\ewolf\SCS-ESI.yaml
service "common" created
deployment "common" created
service "order" created
deployment "order" created
service "varnish" created
deployment "scsesi-varnish" created
PS C:\Users\Public\Desktop\misegr> kubectl get pods --all-namespaces
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE
ingress-nginx   default-http-backend-55c6c69b88-zrmm5   1/1     Running   1           30m
ingress-nginx   nginx-ingress-controller-5984cb9f7-rqjkg 1/1     Running   1           30m
kube-system   etcd-misegr                             1/1     Running   1           29m
kube-system   kube-apiserver-misegr                   1/1     Running   1           30m
kube-system   kube-controller-manager-misegr          1/1     Running   1           30m
kube-system   kube-dns-6r4fd4bdf-cbjx2               3/3     Running   3           30m
kube-system   kube-flannel-ds-25m5w                  1/1     Running   2           30m
kube-system   kube-proxy-8sf59                       1/1     Running   1           30m
kube-system   kube-scheduler-misegr                   1/1     Running   1           30m
kubernetes-dashboard   kubernetes-dashboard-5bd6f67c7-f2v45 1/1     Running   1           30m
scsesi   common-f997f4fc9-tkzn9                 1/1     Running   0           15s
scsesi   order-549cf54966-zfsgj                 1/1     Running   0           15s
scsesi   scsesi-varnish-5079c754d7-5swm7         1/1     Running   0           15s
weave     weave-scope-agent-96tgy                 1/1     Running   1           30m
weave     weave-scope-app-57566f676-zh2bz        1/1     Running   1           30m
PS C:\Users\Public\Desktop\misegr> kubectl get services --all-namespaces
NAMESPACE   NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes   kubernetes ClusterIP   10.96.0.1     <none>         443/TCP          31m
default     default-http-backend ClusterIP   10.110.117.150 <none>         80/TCP          30m
ingress-nginx   ingress-nginx NodePort    10.97.6.160    <none>         80:30080/TCP, 443:30443/TCP 30m
kube-system   kube-dns ClusterIP   10.96.0.10    <none>         53/UDP, 53/TCP 30m
kube-system   kubernetes-dashboard ClusterIP   10.106.151.115 <none>         443/TCP          30m
scsesi   common ClusterIP   None          <none>         <none>           27s
scsesi   order ClusterIP   None          <none>         <none>           27s
scsesi   varnish NodePort    10.111.124.112 <none>         8080/TCP, 8080:32080/TCP 27s
scsesi   scsesi-varnish ClusterIP   10.103.315.114 <none>         80/TCP          30m
```



# Continuous Integration (CI) / Continuous Delivery (CD)

# IT Soll (DevOps)



- ★ Automatisierter Release Zyklus, Tests, ...
- ★ Das gleiche gilt für die IT Operations!

Quelle: WhitePaper IT4IT von OpenGroup

# DevOps - Gartner



- ★ **DevOps steht** für eine Veränderung der IT-Kultur und konzentriert sich auf schnelle IT-Servicebereitstellung durch die Einführung von agilen, schlanken Praktiken im Rahmen eines systemorientierten Ansatzes.
- ★ DevOps betont Menschen (und Kultur) und versucht, die Zusammenarbeit zwischen Unternehmen und Entwicklungsteams zu verbessern.
- ★ DevOps-Implementierungen nutzen Technologie - insbesondere Automatisierungstools, die eine zunehmend **programmierbare** und **dynamische** Infrastruktur aus einer **Lebenszyklusperspektive** nutzen können.
- ★ Quelle: <https://www.gartner.com/it-glossary/devops>

# Pipeline

Continuous Integration / Continuous Delivery

# CI/CD Pipeline: Tools

The screenshot displays a CI/CD pipeline tool interface. At the top, there's a 'KB Demo Project' header with navigation tabs: Menu, Overview, Board, Calendar, List, and Gantt. The 'Board' tab is active, showing a Kanban board with columns: Backlog (1), Ready (3), and Work in progress (1). The 'Ready' column contains three tasks: #52 (Update screenshots, documentation), #53 (Fix API bug (edge case), api), and #50 (Improve Markdown editor, markdown, user interface). Below the Kanban board, there's a user profile for '无闻' (unknown) and a repository 'sample-hook' with a commit '0df42db' from 5 days ago. A pipeline execution flow is shown, starting with 'Build', followed by 'Browser Tests' (Chrome, Firefox, Internet Explorer, Safari), 'Static Analysis', and 'Deploy'. The 'Steps - Firefox' section shows a 'Shell Script' step running successfully in less than 1 second.

Continuous Integration / Delivery Pipeline

★ Für die Umsetzung einer DevOps CI/CD Pipeline verwenden wir folgende Tools:

- Planung – [Kanboard](#)
- Sourceverwaltung – Git, [Gogs](#)
- CI/CD – [Jenkins Blue Ocean](#)

★ Die Tools laufen in der Docker/Kubernetes Umgebung.

★ Jenkins startet für den Buildprozess jeweils eigene Docker Container.

# CI/CD Pipeline: Artefakte



```
pipeline {
  agent none
  stages {
    stage('Build') {
      agent {
        docker {
          image 'maven:3-alpine'
          args '-v /root/.m2:/root/.m2'
        }
      }
    }
  }
  steps {
    sh 'cd scs-demo-esi-order/ && mvn -B -DskipTests clean package'
    archiveArtifacts 'scs-demo-esi-order/target/*.jar'
  }
}
```

- ★ Alle Sourcen zu den Beispielen werden in (Git-)Repository gespeichert und mit einem [Jenkinsfile](#) ergänzt.
- ★ Beispiele aus dem Kurs Microservices:
  - Frontend Integration - <https://github.com/mc-b/SCS-ESI>
  - Asynchrone Microservices - <https://github.com/mc-b/microservice-kafka>
  - Synchrone Microservices - <https://github.com/mc-b/microservice-kubernetes>
  - BPMN - <https://github.com/mc-b/bpmn-tutorial>
- ★ Die Jenkinsfile übernehmen den
  - **Build** - der Applikation
  - **Test** – Durchführung der Unit-Tests, Kontrolle Sourcecode (z.B. [Checkstyle](#))
  - **Build Images** – Builden der Docker Images
  - **Deploy** – Verteilung der Docker Images

# Live Session: CI/CD

## ★ Tools starten

- `kubectl apply -f devday/devops/`

## ★ SW Entwicklung

- Planen ([Kanboard](#)).
- Sourcen in Versionsverwaltung (Repository) stellen ([Gogs](#)).
- Sourcen um CI/CD Konfiguration ergänzen (Jenkinsfile)
- Neue Pipeline in [Jenkins](#) anlegen
- Änderungen in Versionsverwaltung in KanBoard übernehmen
- SW automatisch Verteilen
- Resultat in Dashboard und als neues BPMN [Frontend](#)



Create a new Pipeline

### Where do you store your code?



Bitbucket Cloud



Bitbucket Server



GitHub



GitHub Enterprise



Git

### Connect to a Git repository

Any repository containing a Jenkinsfile will be built automatically. Not sure what we are talking about? [Learn more about Jenkinsfile's](#).

Repository URL

`https://github.com/mc-b/SCS-ESI.git`

**Warning**

Saving Pipelines is unsupported using http/https repositories. Please use SSH instead.

Credentials

System Default

Add

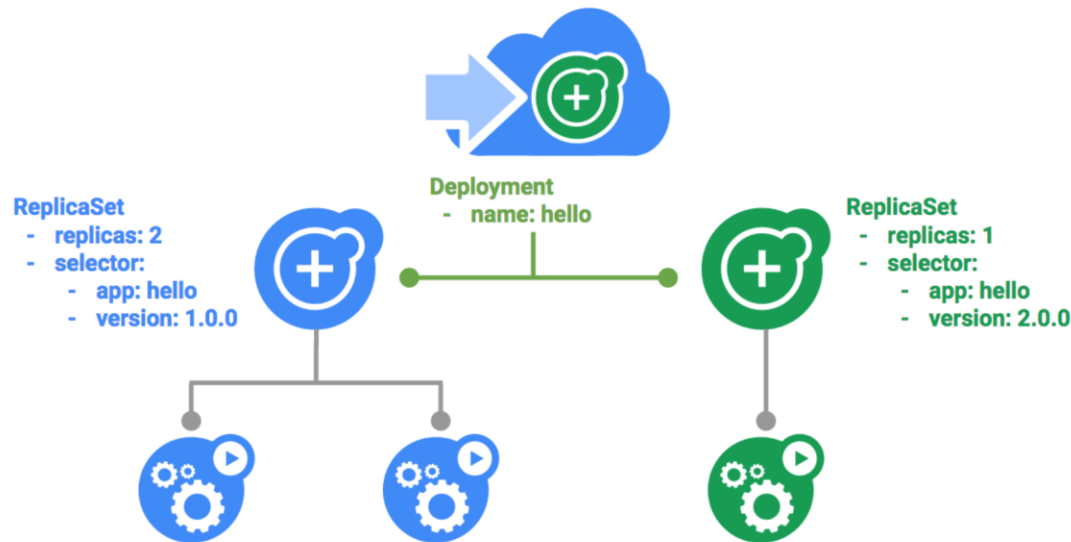
Create Pipeline



# Rolling Update

Weitere Möglichkeiten mit Kubernetes

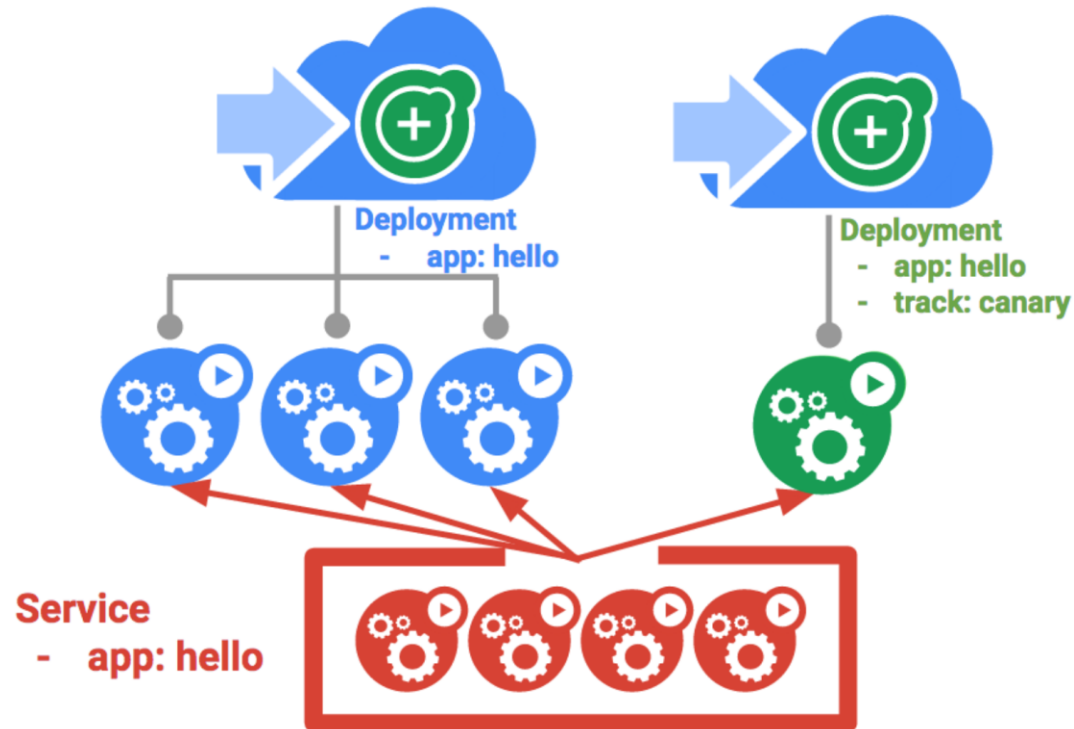
# Live Session: Rolling Update



- ★ **Deployments** unterstützen das **Aktualisieren** von **Image** auf eine neue Version mithilfe eines fortlaufenden Aktualisierungsmechanismus.
  - ★ Wenn ein Deployment mit einer neuen Version aktualisiert wird, erstellt es ein neues ReplicaSet und erhöht langsam die Anzahl der Replikate im neuen ReplicaSet, da die Replikate im alten ReplicaSet verringert werden.
- 
- ★ `kubectl set image deployment/bpmn-frontend \ bpmn-frontend=misegr/bpmn-frontend:V1.0`
  - ★ `kubectl describe deployments bpmn-frontend`

Quelle: <https://codelabs.developers.google.com/codelabs/k8s-kickstart/#10>

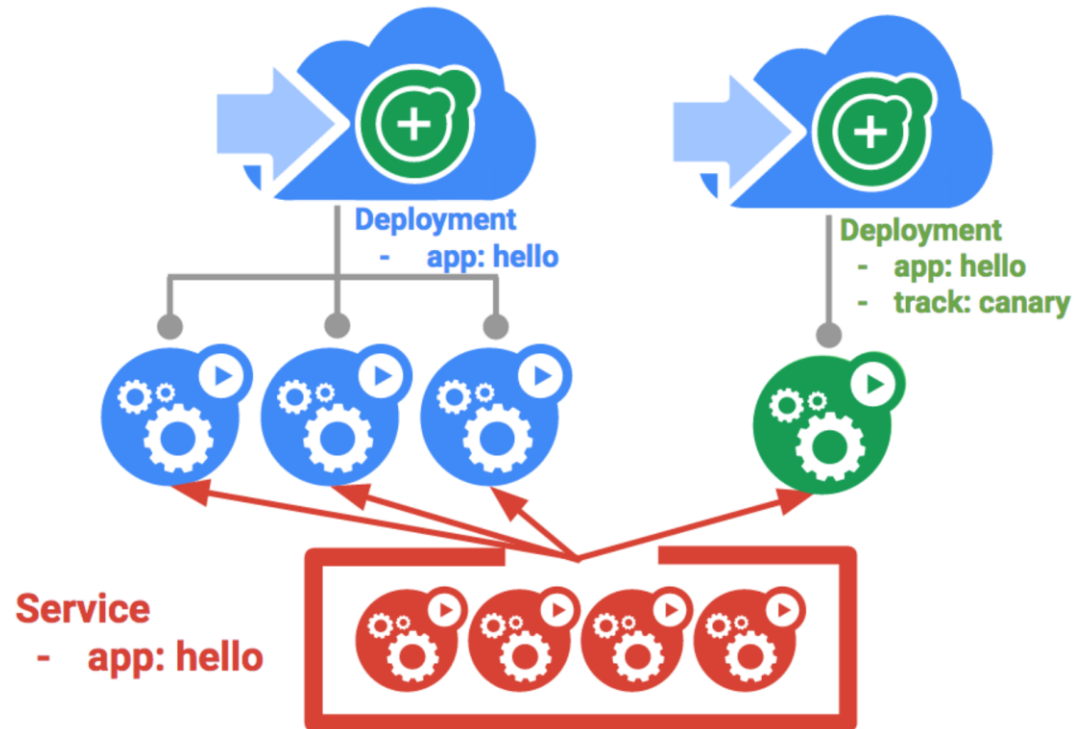
# Canary Deployments



- ★ Wenn Sie eine neue SW Version in der Produktion mit einer **kleinen Gruppe** Ihrer Benutzer testen möchten, können Sie ein «Canary» Deployment durchführen.
- ★ Bei «**Canary**» **Deployment** können Sie eine Änderung nur einer **kleinen Gruppe** Ihrer Benutzer **freigeben**, um das Risiko von neuen Versionen zu verringern.

Quelle: <https://codelabs.developers.google.com/codelabs/k8s-kickstart/#11>

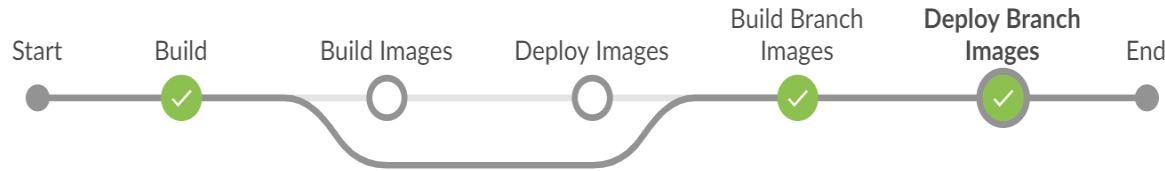
# Blue/Green Deployments



- ★ Rolling-Updates sind ideal, da sie Ihnen ermöglichen, eine Anwendung langsam mit minimalem Overhead, minimalen Auswirkungen auf die Leistung und minimalen Ausfallzeiten bereitzustellen.
- ★ Es gibt jedoch Fälle, in denen es sinnvoll ist, die Load Balancer so zu ändern, dass sie **erst** nach der **vollständigen Verteilung** auf die **neue Version** verweisen.
- ★ In diesem Fall sind so genannte Blue-Green-Bereitstellungen der richtige Weg.

Quelle: <https://codelabs.developers.google.com/codelabs/k8s-kickstart/#12>

# CI/CD Pipeline Erweiterungen: Jenkinsfile



```
stage('Build Branch Images') {  
  agent any  
  when {  
    not { branch 'master' }  
    not { branch 'canary' }  
  }  
  steps {  
    unstash 'jar'  
    sh("cd bpmn-frontend/ && /usr/bin/docker build -t misegr/bpmn-frontend:${env.BRANCH_NAME}.${env.BUILD_NUMBER} .")  
    sh("cd bpmn-backend/ && /usr/bin/docker build -t misegr/bpmn-backend:${env.BRANCH_NAME}.${env.BUILD_NUMBER} .")  
  }  
}  
stage('Deploy Branch Images') {  
  agent any  
  when {  
    not { branch 'master' }  
    not { branch 'canary' }  
  }  
  steps {  
    sh("git clone https://github.com/mc-b/misegr.git")  
    sh("sed -i 's#latest#${env.BRANCH_NAME}.${env.BUILD_NUMBER}#' misegr/bpmn/*.yaml")  
    sh("kubectl apply -f misegr/bpmn")  
  }  
}
```

- ★ Alle Sourcen zu den Microservices werden in (Git-)Repository gespeichert und mit einem Jenkinsfile ergänzt.
- ★ Beispiel:
  - BPMN - <https://github.com/mc-b/bpmn-tutorial>
- ★ Die Jenkinsfile übernehmen den
  - **Build** - der Applikation
  - **Test** – Durchführung der Unit-Tests, Kontrolle Sourcecode (z.B. Checkstyle)
  - **Build Images** – Builden der Docker Images
  - **Deploy** – Verteilung der Docker Images, aufgrund eines neuen Branches.
- ★ **Kontrollieren**
  - `kubectl describe deployments bpmn-frontend`
  - BPMN Frontend

# Live Session: CI/CD Rolling Updates



## ★ SW Entwicklung

- Neuer Branch Erstellen
- Änderungen im Source Code vornehmen und einchecken, z.B.:
  - ★ Index.html Titel ändern
- Jenkins Pipeline anstossen
- Neue Version wird automatisch mittels Rolling Update nachgeführt.



Where do you store your code?

Bitbucket Cloud	Bitbucket Server
GitHub	GitHub Enterprise
Git	



Connect to a Git repository

Any repository containing a Jenkinsfile will be built automatically. Not sure what we are talking about? [Learn more about Jenkinsfile's.](#)

Repository URL

**Warning**

Saving Pipelines is unsupported using http/https repositories. Please use SSH instead.

Credentials

[Add](#)

[Create Pipeline](#)

# Live Session: Kanboard Updaten



KB Aufbau CI/CD Pipeline

Übersicht Issues Pull-Requests Erkunden

status:open

Überblick Pinnwand Liste

(4) Ideen (4) (1) Bereit (1) In Arbeit (3) Erledigt

#5 Sources um CI/CD Konfiguration ergänzen ~4Std ~3Std P0

#6 Neue Pipeline in Jenkins anlegen ~4Std ~4Std P0

#7 Rolling Update durchführen <15min <15min P0

#8 Notizen für Daily Scrum <15min <15min P0

#4 Versionsverwaltung einrichten ~4Std ~4Std P0

#1 Docker / Kubernetes aufsetzen

#2 CD / CD Tools

#3 Kanboard auf Projekt "Aufbau CI/CD Pipeline" erstellen

root / bpmn-tutorial

Beobachten beenden 1 Favorit hinzufügen 0 Fork

Dateien Issues 0 Pull-Requests 0 Wiki Einstellungen

Keine Beschreibung

34 Commits 1 Branches 0 Releases

Branch: master bpmn-tutorial

Neue Datei Datei hochladen HTTP SSH http://localhost:32300/root/

mc-b 6dda896

bpmn-backend bpmn-frontend images

Änderungen einchecken

Neuer Punkt Daily Scrum #8

Eine ausführlichere Beschreibung kann hinzugefügt werden...

- ★ Gogs kann Änderungen in Repository, bzw. dessen Kommentar an Kanboard weitergeben.
- ★ Details siehe: <https://github.com/kanboard/plugin-gogs-webhook>

# Kurse



- ★ [Microservices-Grundlagen \(«MISEGR»\)](#)
- ★ [Docker und Kubernetes – Übersicht und Einsatz \(«DUK»\)](#)
- ★ [Internet of Things \(IoT\) im Einsatz \(«IOTEIN»\)](#)
- ★ [Machine Learning Grundlagen \(«MLG»\)](#)



# Abschluss



★ Viel Erfolg mit Continuous Integration /  
Delivery und Docker/Kubernetes

★ Marcel Bernet