Machine Learning auf Kubernetes: Kubeflow

ContainerConf 2020 Kubernetes Edition

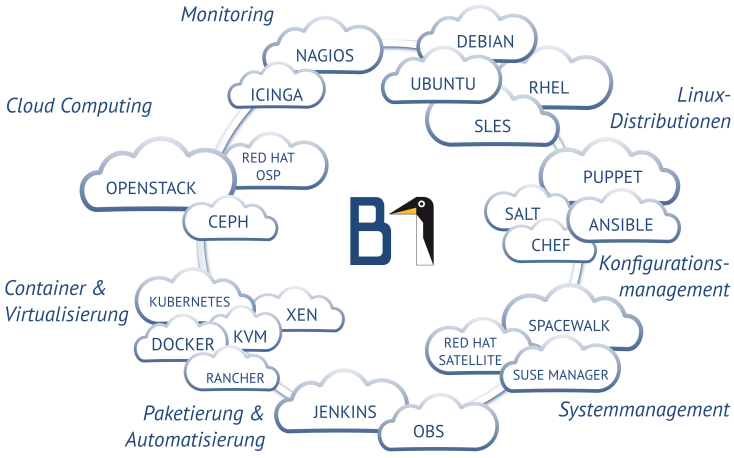
21. Juli 2020

Christian Rost
Linux Consultant & Trainer
B1 Systems GmbH
info@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- Linux/Open Source-Themen
- national & international tätig
- über 125 Mitarbeiter
- unabhängig von Soft- & Hardware-Herstellern
- Leistungsangebot:
  - Beratung & Consulting
  - Support
  - Training
  - Managed Service & Betrieb
  - Lösungen & Entwicklung
- Standorte in Rockolding, Köln, Berlin & Dresden

Schwerpunkte

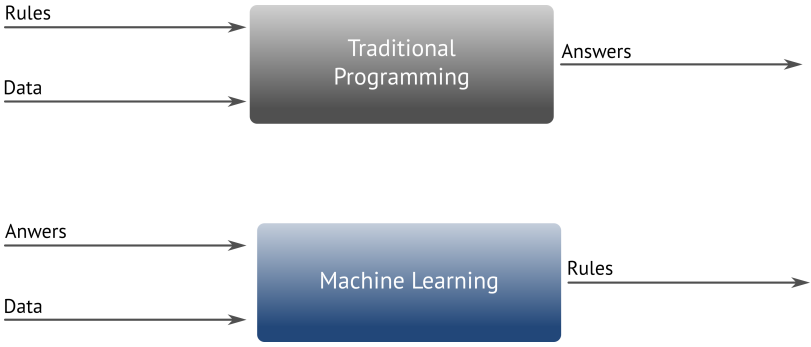


MachineLearning

MachineLearning

- Teilbereich der Künstlichen Intelligenz
- aus Datenbestände werden Muster und Regelwerke erkannt
- Lösungen (Prognose) werden errechnet
- durch Training werden Muster und Regelwerke besser bzw. Prognosen genauer
- Frameworks
  - Tenserflow
  - PyTorch

“Traditionelle” Programmierung vs MachineLearning



Beispieldatensatz MNIST

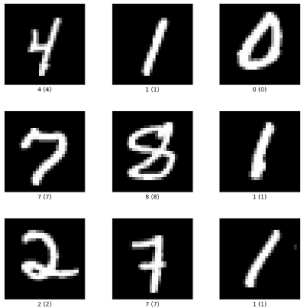


Abbildung: Datensatz mit Handschriften – Quelle: https://github.com/kubeflow/examples)

MachineLearning auf Kubernetes?

Warum auf Kubernetes?

- Container sind toll ;P
- viele Schritte bis Produktion
- jeder Schritt ein Container
- einfachere Anpassung der Parameter und Datenbestände
- reproduzierbare Infrastruktur
- Notebook/onPremise/Cloud

Kubeflow

Kubeflow

- seit 2017
- März 2020 v1.0
- Toolset
  - Kubeflow UI
  - kfctl
  - Jupyter Notebooks (Develop)
  - Fairing (Build)
  - TFJob/PyTorch (Train)
  - KFServing (Deploy)
  - ...
- Istio
- Knative

Kubeflow UI

- zentrale Schnittstelle
- Namespaces
- Zugriff auf Kubeflow ToolChain

Kubeflow UI

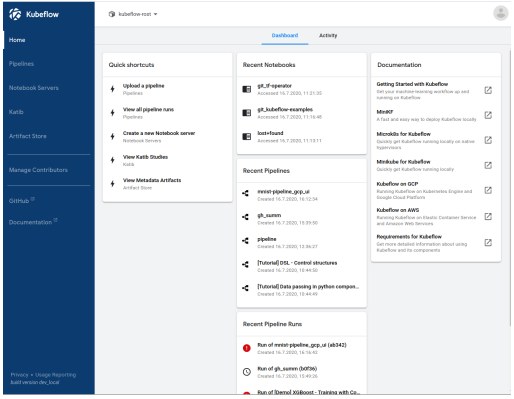


Abbildung: Kubeflow Dashboard

Jupyter Notebooks (Develop)

- Open Source Web Application
- Dokumente bestehend aus Zellen
- Zellen können Code oder Markdown enthalten
- Code kann direkt ausgeführt werden
- “IDE” für Kubeflow

Jupyter Notebooks (Develop)

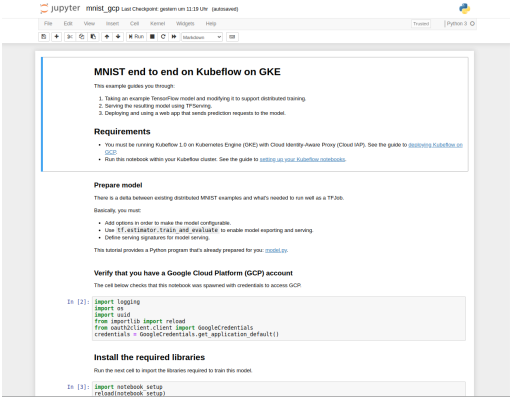


Abbildung: Jupyter Notebook

Fairing (Build)

- baut Container Image in Kubernetes
- nutzt Kaniko zum Bauen
- aus Jupyter heraus gestartet
- Trainingscode & Framework im Image

TFJob/PyTorch (Train)

- Kubernetes CRDs & Operator
- Tenserflow/PyTorch (stable)
- Auf GPU/TPU Nodes
- mehrere Container je Job

Chief	Orchestriert Training
Ps	Distributed Model Parameter Server
Worker	Hier wird Trainiert
Evaluator	Verarbeitet Metriken des Trainings

- Modell nach Training auf Persistent Storage

KFServing (Deploy)

- Software in Produktion
- Greift auf trainiertes Modell zu
- besteht meist aus
  - Kubernetes Deployment
  - Kubernetes Service
  - Istio VirtualService

App in Produktion

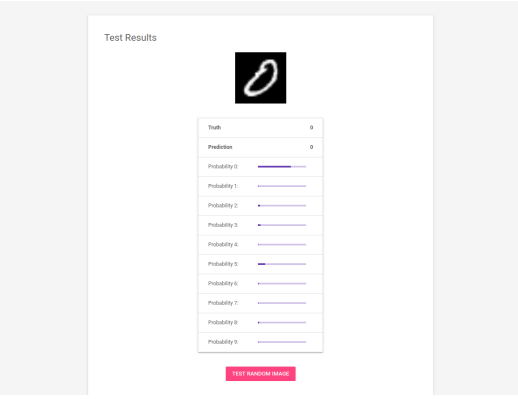


Abbildung: MNIST-Beispielapplikation

App in Produktion

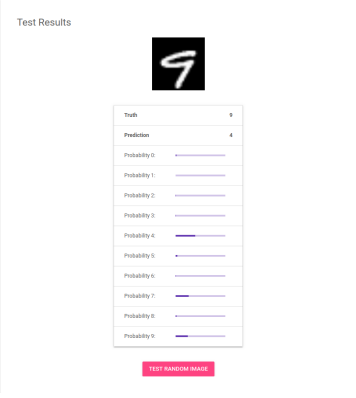


Abbildung: MNIST-Beispielapplikation

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de oder +49 (0)8457 - 931096