

Machine Learning auf Kubernetes: Kubeflow

ContainerConf 2020 Kubernetes Edition

21. Juli 2020

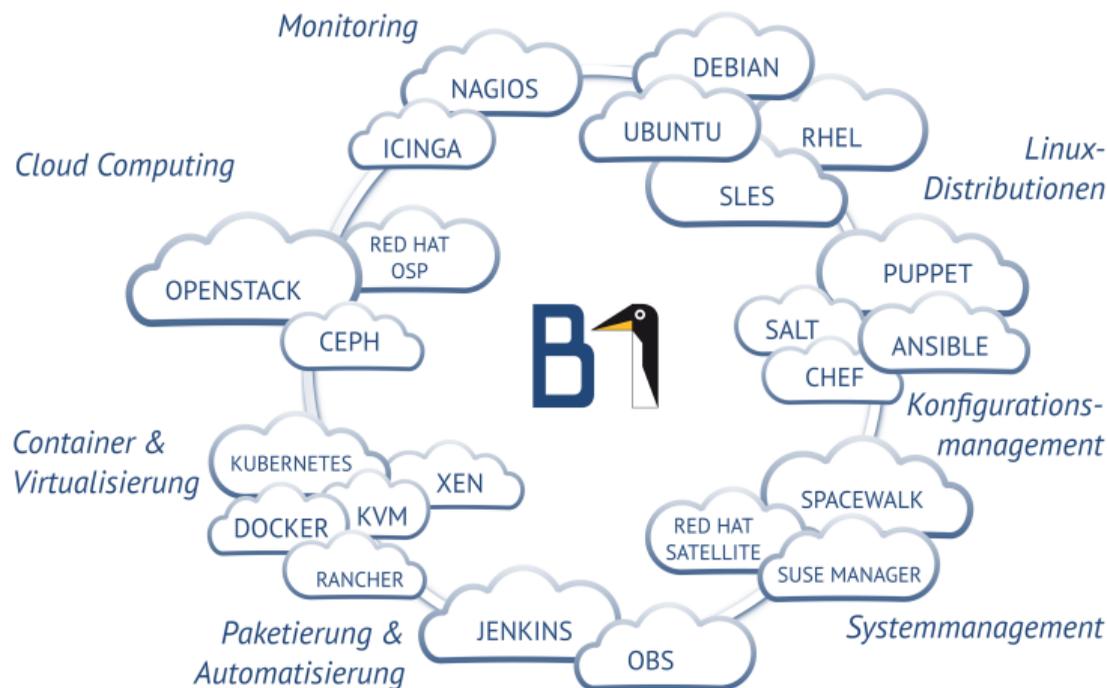


Christian Rost
Linux Consultant & Trainer
B1 Systems GmbH
info@b1-systems.de

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

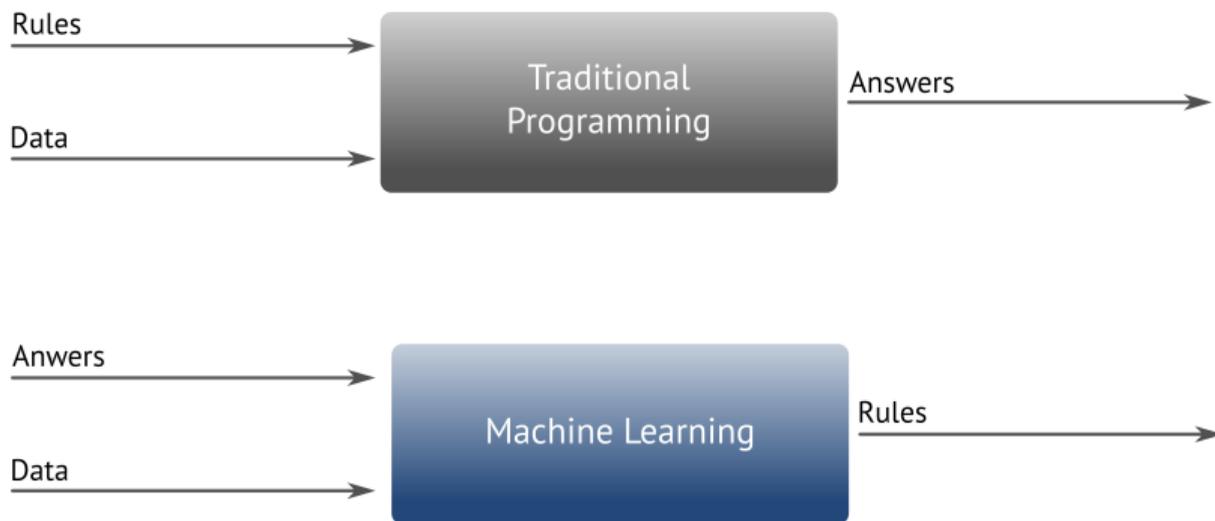


MachineLearning

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

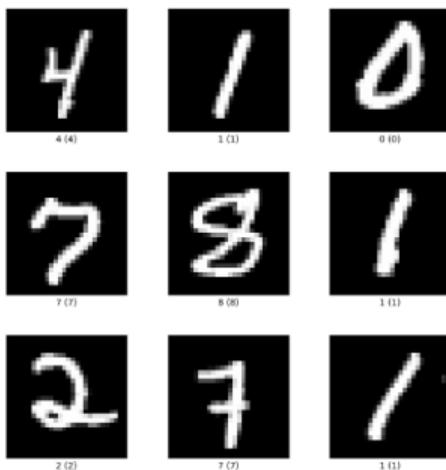


Abbildung: Datensatz mit Handschriften – Quelle: <https://github.com/kubeflow/examples>)

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

The screenshot displays the Kubeflow Dashboard for a user named 'kubeflow-root'. The interface is divided into several sections:

- Left Sidebar:** Contains navigation links for Home, Pipelines, Notebook Servers, Katsub, Artifact Store, Manage Contributors, GitHub, and Documentation.
- Quick shortcuts:** A list of actions including 'Upload a pipeline', 'View all pipeline runs', 'Create a new Notebook server', 'View Katsub Studies', and 'View Metadata Artifacts'.
- Recent Notebooks:** A list of notebooks such as 'git-operator', 'git.kubeflow-examples', and 'lost-found', each with an 'Accessed' timestamp.
- Recent Pipelines:** A list of pipeline runs including 'main-pipeline_gcp_us', 'gh_sumen', 'pipeline', and 'Tutorial[1] DSL - Central structures'.
- Recent Pipeline Runs:** A list of specific pipeline run instances like 'Run of main-pipeline_gcp_us (4b342)' and 'Run of gh_sumen (30756)'.
- Documentation:** A collection of links to guides and tutorials, such as 'Getting Started with Kubeflow', 'Minikube', 'Microk8s for Kubeflow', 'Minikube for Kubeflow', 'Kubeflow on GCP', 'Kubeflow on AWS', and 'Requirements for Kubeflow'.

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)

Jupyter mnist_gcp Last Checkpoint: gcpkm-um 12.19 Uhr (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.0

Run MacBook

MNIST end to end on Kubeflow on GKE

This example guides you through:

1. Taking an example TensorFlow model and modifying it to support distributed training.
2. Serving the resulting model using TF Serving.
3. Deploying and using a web app that sends prediction requests to the model.

Requirements

- You must be running Kubeflow 1.0 on Kubernetes Engine (GKE) with Cloud Identity-Aware Proxy (Cloud IAP). See the guide to [deploying Kubeflow on GCP](#).
- Run this notebook within your Kubeflow cluster. See the guide to [setting up your Kubeflow notebooks](#).

Prepare model

There is a delta between existing distributed MNIST examples and what's needed to run well as a TPJOB. Basically, you must:

- Add options in order to make the model configurable.
- Use `tf.estimator.train_and_evaluate` to enable model exporting and serving.
- Define serving signatures for model serving.

This tutorial provides a Python program that's already prepared for you: [code.py](#).

Verify that you have a Google Cloud Platform (GCP) account

The cell below checks that this notebook was spawned with credentials to access GCP

```
In [2]: import logging
import os
import uuid
from importlib import reload
from oauthlib_client import Import GoogleCredentials
credentials = GoogleCredentials.get_application_default()
```

Install the required libraries

Run the next cell to import the libraries required to train this model.

```
In [3]: import notebook_setup
reload(notebook_setup)
```

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
- Tensorflow/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

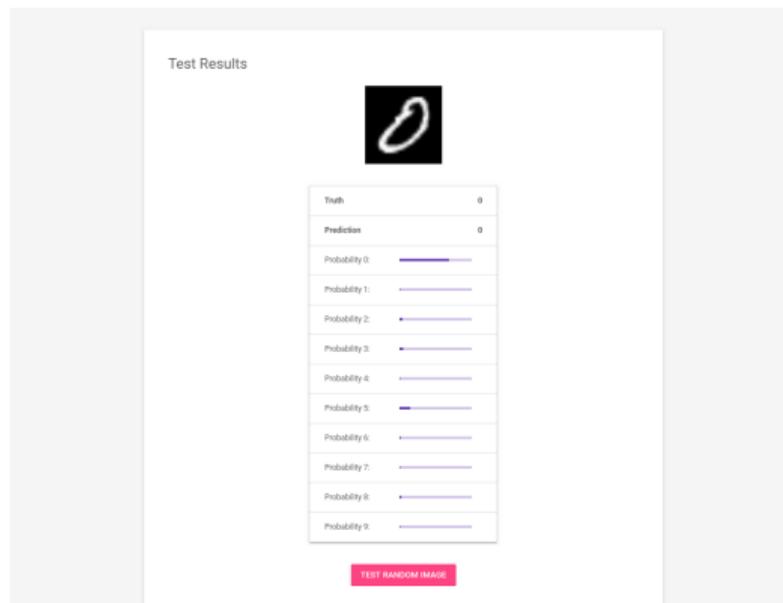


Abbildung: MNIST-Beispielapplikation

App in Produktion

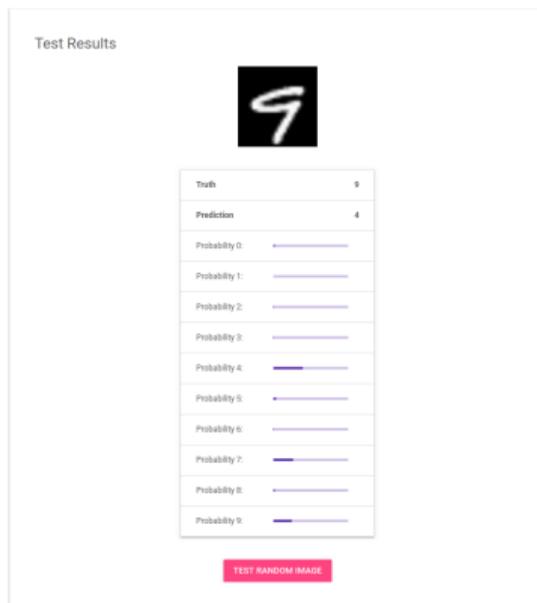


Abbildung: MNIST-Beispielapplikation

Vielen Dank für Ihre Aufmerksamkeit!

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