Kubeflow: ML on OpenShift

Pete MacKinnon
Principal Software Engineer
Red Hat Inc., AI Center of Excellence
pmackinn@redhat.com

January 27, 2019
DevConf.cz Brno
Agenda

- The science of Data Science
- The challenge of ML DevOps
- Project history, stakeholders, and community
- Kubeflow core* components
- OpenShift considerations
- Demo
- Q&A
Old school ML
Modern ML techniques and their applications

Techniques
- Convolutional neural networks
- Gradient Boosting
- Restricted Boltzmann machines (RBM)
- Autoencoder
- Supervised vs. unsupervised training

Applications
- image, voice, and video recognition
- OCR
- network intrusion
- fraud detection
- ranking
- email filtering
- etc.
Enablement: modern hardware for ML

- **CPU**
  - Intel Core i9-7980XE: 1955 GFLOPS (single)

- **GPU**
  - Once a rig enhancement for video gamers
  - The same features designed for boosting FPS at higher resolutions turn out to be pretty good at other complex matrix computations
  - API and libraries to make use of the increased parallel processing power (e.g., CUDA)
  - NVIDIA V100 (Mezzanine): 14899 GFLOPS (single)
Cucumber classification with TensorFlow
A simplified ML pipeline (courtesy of M.Hild)

**Simplified ML Workflow/Pipeline**

- Keeping track of datasets is hard
- How to do automatic retraining when data changes?
- Storage and network bottlenecks

Classic app development issues:
- Reproducibility (“it works on my machine”)
- Scalability
- Getting feedback from production

- Slow sequential training
- Hard to explore hyper-parameters space
- Distributed training is hard to set up
(De)composition of a ML pipeline
ML for everyone

- A lot of pieces needed to enable a comprehensive ML platform
- Difficult for one project or institution to solve all these challenges
- Start with a high-level mission statement
  - Portable: bare metal to cloud
  - Scalable: from 1 machine to 100s
  - Composable: microservice architecture
- Kubernetes (and OpenShift) are a good starting point
- Origin story
  - Google engineers had developed ML pipelines for the YouTube recommendation platform
  - Patterns and learnings developed there informed the idea for an open-source project
The Kubeflow mission

- Dedicated to making deployments of machine learning (ML) workflows on Kubernetes simple, portable, and scalable
- Goal is not to recreate other services, but to provide a straightforward way to deploy best-of-breed open source systems for ML to diverse infrastructures
  - For example, use TensorFlow or PyTorch or XGBoost, but don’t re-invent them
- Anywhere you run Kubernetes, you should be able to run Kubeflow
  - Generally true for OpenShift
The community today(-ish)

- GitHub
- Apache 2.0 license
- 23 repositories
- 143 contributors
- 1017 commits in core repo
- 0.4.0 release
- New integrations happening continuously
- Institutional participation
  - Not all are depicted
Kubeflow components overview

- **Core components:**
  - Ambassador for ingress control
  - JupyterHub* to create and manage interactive Jupyter notebooks
  - Katib for hyperparameter tuning
  - Kubeflow Pipelines
  - TensorFlow training controller for launching TFJobs
  - TensorFlow Serving to export trained TensorFlow models to Kubernetes

- **Other components:**
  - Argo for workflow management
  - PyTorch operator
  - Seldon for model inference serving
  - Pachyderm for data governance
  - scikit-learn, MPI operator, Arena, Fairing, and more to come
Ksonnet

- JSON-based CLI tool to generate Kubernetes deployment objects
- Based on the Jsonnet language
- More expressive than managing and patching YAML descriptors directly
  - Conceptually similar to OpenShift templates
- Generates resources that are entirely compatible with Kubernetes
  - You can still edit/patch the generated resources yourself as YAML/JSON
- Enables re-use of components for different envs
  - dev, test, staging, prod, etc.
- Parameter settings for different components
  - Continuous deployment
  - Just apply your changes and resources will be immediately updated deletion of resources
Notebooks

● JupyterHub project
  ○ currently provides the pod spawner for Kubeflow
  ○ ability to customize

● Kubeflow curates TensorFlow CPU and GPU notebooks
  ○ based on the default builds from the TF project
  ○ users can add pypi packages to suit their needs

● Kaggle project
  ○ adaption of the Kaggle Python notebook for Kubeflow

● RAPIDS AI project at NVIDIA
  ○ Kubeflow adaptation of their notebook
  ○ RAPIDS provides Python libraries for GPU-accelerated
dataframe manipulation, ML algorithms,
and graph processing
Ambassador

- Cloud-native ingress controller (based on Envoy)
- Reverse proxy
  - enhancement over Node.js proxy in JupyterHub
- Annotation-based URL mapping to services
  - hub/
  - tfjobs/ui/
  - pipeline/
- IAP auth and can integrate with cert-manager
Argo

- Kubernetes native workflow manager (CRD and operator)
- Used by project in pre/post-submit testing in GCP, Kubebench, and Kubeflow Pipelines sub-project
- Define workflows where each step in the workflow is a pod
- Model multi-step workflows as a sequence of tasks or capture the dependencies between tasks using a graph (DAG)
- Well suited to compute intensive jobs for machine learning or data processing on Kubernetes
- Argo is cloud agnostic and can run on any Kubernetes cluster
Pipelines

● Pachyderm was an early integration for data pipelines and governance (and still a deployable component)
  ○ “Git for data science”
● “Pipelines” has become a formal sub-project within Kubeflow
  ○ A UI for managing and tracking experiments, jobs, and runs
  ○ An operator-based engine for scheduling multi-step ML workflows
  ○ A Python SDK for defining and manipulating pipelines and components
  ○ Uses Argo as the workflow orchestrator
Seldon

- Deploy and manage runtime inference graphs at scale
  - often different from training graphs
  - the graphs are deployed as microservices in Kubernetes
    - Model, Transformer, Router, Combiner, Output Transformer
    - A/B testing, multi-armed bandit
- TensorFlow, scikit-learn, Spark, H2O, R models
- Mix and match components for advanced graph deployments
- gRPC and REST interfaces
- Model wrappers for Python, R, Java and Node.js
  - uses OpenShift s2i for generating the wrappers
Kubeflow on OpenShift: The lost pages

- **OpenShift: the enterprise Kubernetes**
  - Developer focused
  - Extended resources and concepts
  - RBAC, and then some…
    - Namespaces, projects, and users
    - UID permissions and constraints

- **Kubeflow**
  - Open source development by contributors is generally done using upstream Kubernetes (but with vanilla RBAC)
    - Bare metal, laptop, minikube, etc.
    - Google dominates the project so GKE is go-to for them
  - Project and its ecosystem makes significant use of...

- **Operators**
  - Custom Resource Definitions (CRD)
  - ClusterRoles and ClusterRoleBindings
  - Need appropriate user roles to use some of the operators, (e.g., TFJob)
Kubeflow on OpenShift: 0.4.0 & 3.11

- Even as cluster-admin, there are adjustments to get things going

```shell
# UI pods can’t bind to port 80
oc adm policy add-scc-to-user anyuid -z ambassador
oc adm policy add-scc-to-user anyuid -z jupyter
oc adm policy add-scc-to-user anyuid -z katib-ui

# minio UID issue (mkdir /.minio: permission denied)
# however there is currently no SA for minio (yet)
oc adm policy add-scc-to-group anyuid system:authenticated

# mysql and vizier-db require a PV to run at a location that
# selinux has correct attributes for
# solution: create a default StorageClass that mysql is
# permitted to write its /var/lib/mysql to, then chcon ...
```
Live demo and Q&A
THANK YOU

https://www.kubeflow.org
https://github.com/kubeflow/kubeflow
kubeflow-discuss@googlegroups.com

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat