O'REILLY TensorFlow World Conference, Santa Clara 2019

Distributed training with TensorFlow on AWS

Shashank Prasanna,
Amazon Web Services
@shshnkp

29th October 2019

aws

Agenda

Introduction (15 mins)

- Deep learning and need for scale
- Challenges with scaling training – algorithms and infrastructure
- Approaches to distributed training
- What you need to go from a training script to distributed training

Hands-on (2 hours 30 mins)

- Distributed training with TensorFlow
    - Setup
    - Amazon SageMaker
    - Amazon Elastic Kubernetes Service (Amazon EKS)

aws

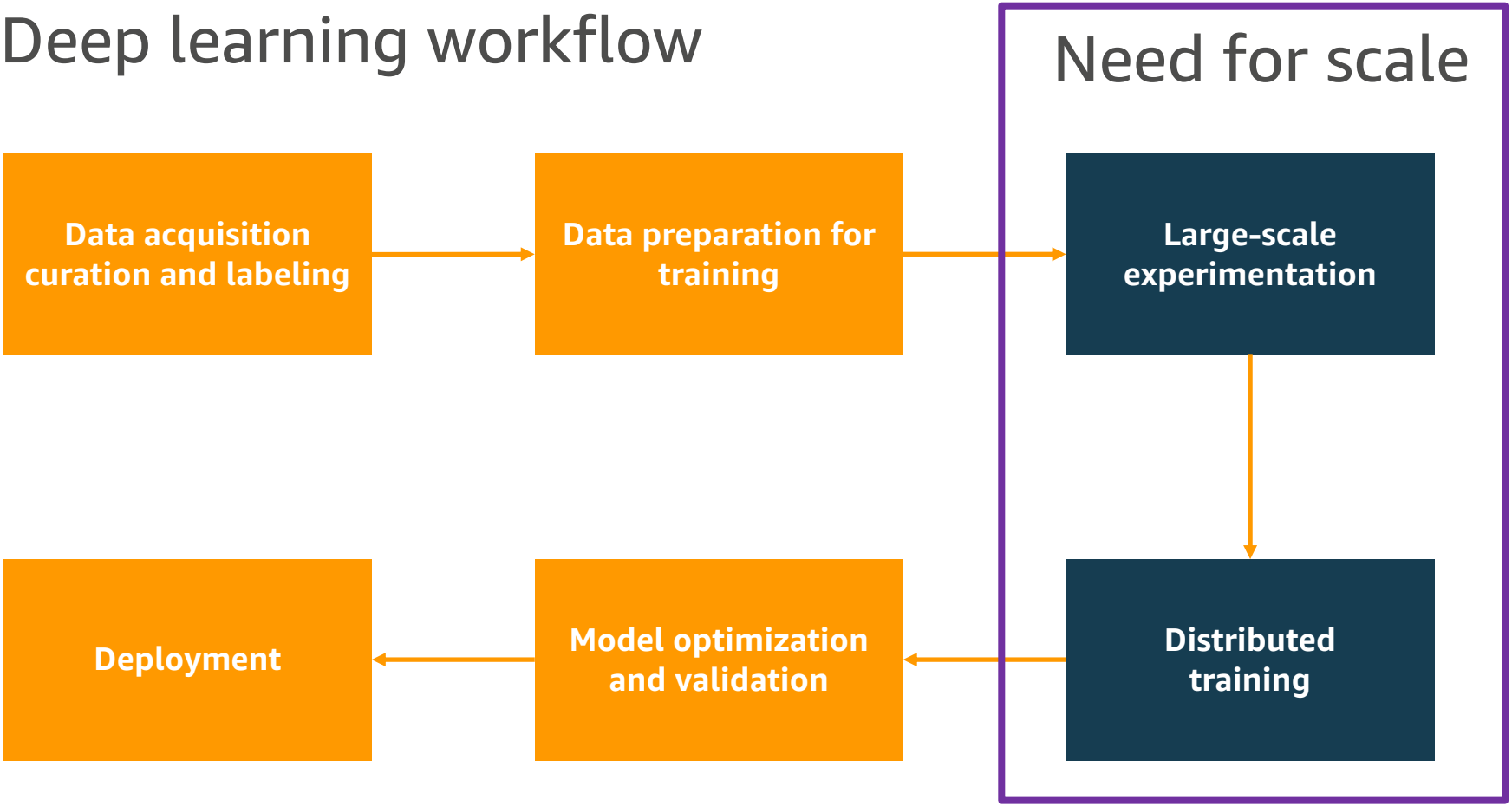
Workshop learning objectives

You will be able to:

- Identify when to consider distributed training
- Describe different approaches to distributed training using TensorFlow
- Outline libraries and tools needed for distributing TensorFlow training workloads on AWS
- Demonstrate TF script changes required to go from single-GPU to multi-GPU distributed training
- Demonstrate using Amazon SageMaker and Amazon EKS to run distributed training jobs
- Apply these skills to your own deep learning problem

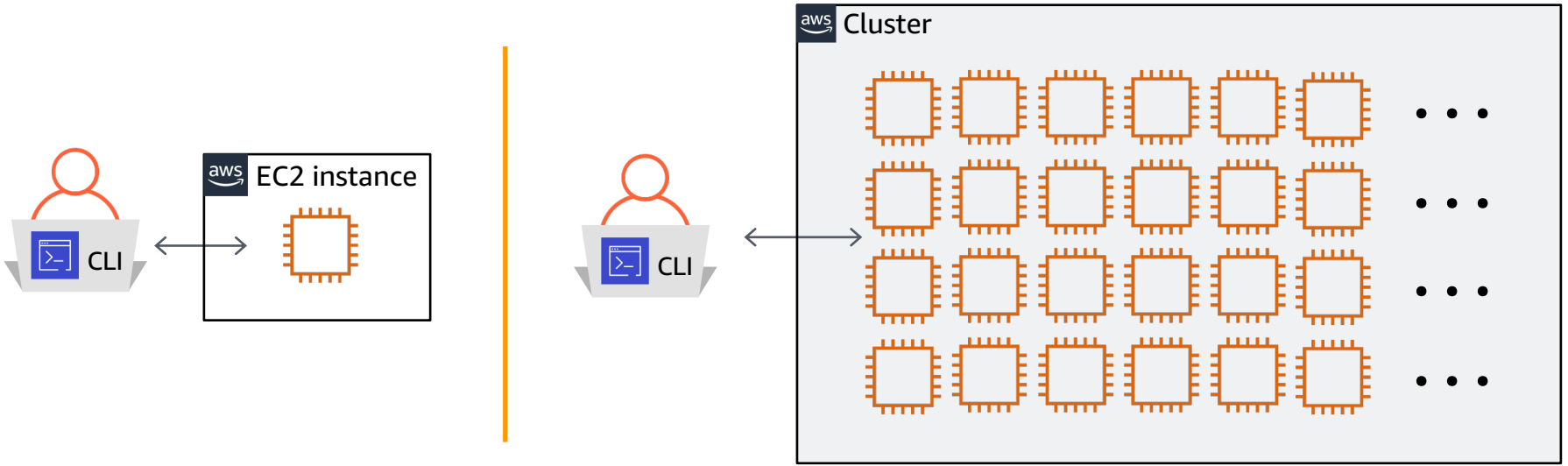
aws

Deep learning workflow



aws

Deep learning is computationally expensive, but can be scaled-out

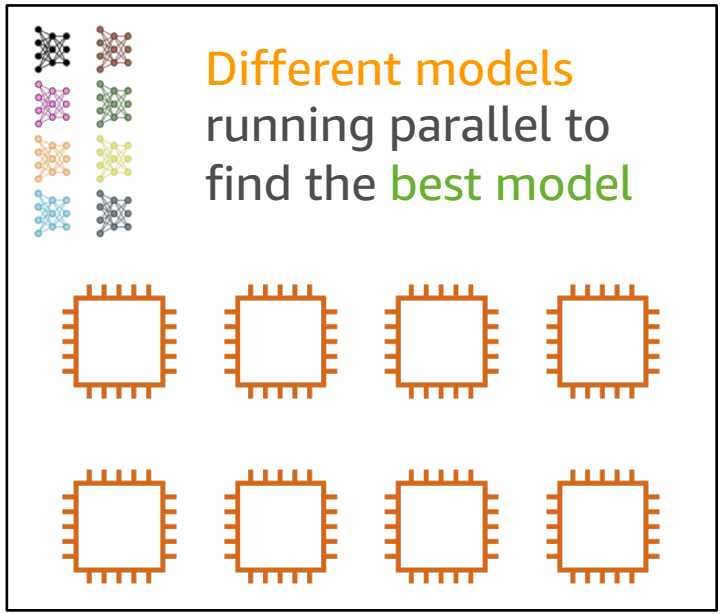


this…

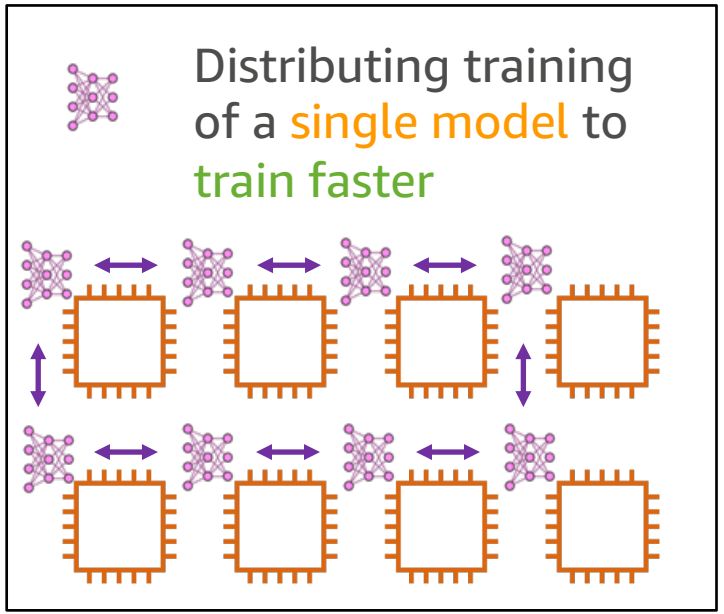
…to this

Scaling-out deep learning training

Parallel experiments

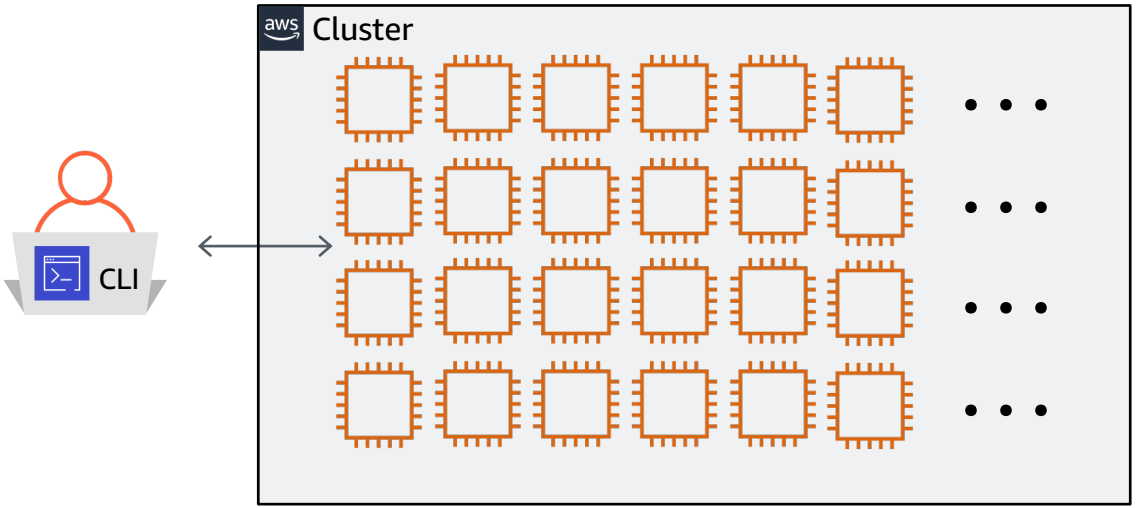


Distributed training



aws

Challenges with scaling deep learning training



1. **Software dependencies**
2. **Infrastructure management**

Machine learning stack is complex

- “My code requires building several dependencies from source”
- “My code isn’t taking advantage the GPU/GPUs”
    - “is cudnn, nccl installed, is it the right version?”
- “My code is running slow on CPUs”
    - “oh wait, is it taking advantage of AVX instruction set ?!?”
- “I updated my drivers and training is now slower/errors out”
- “My cluster runs a different version of framework/linux distro”

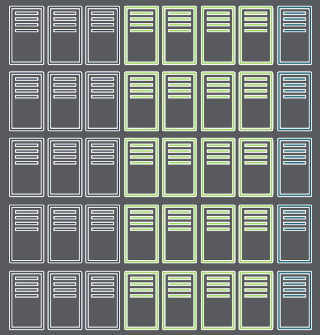
Makes portability, collaboration, scaling training really really hard!

aws

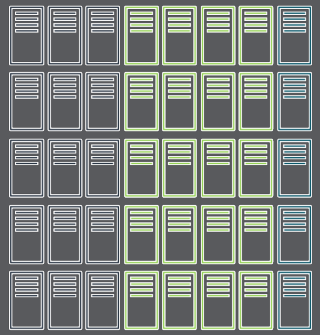
My code

Development system

Multiple points of failure

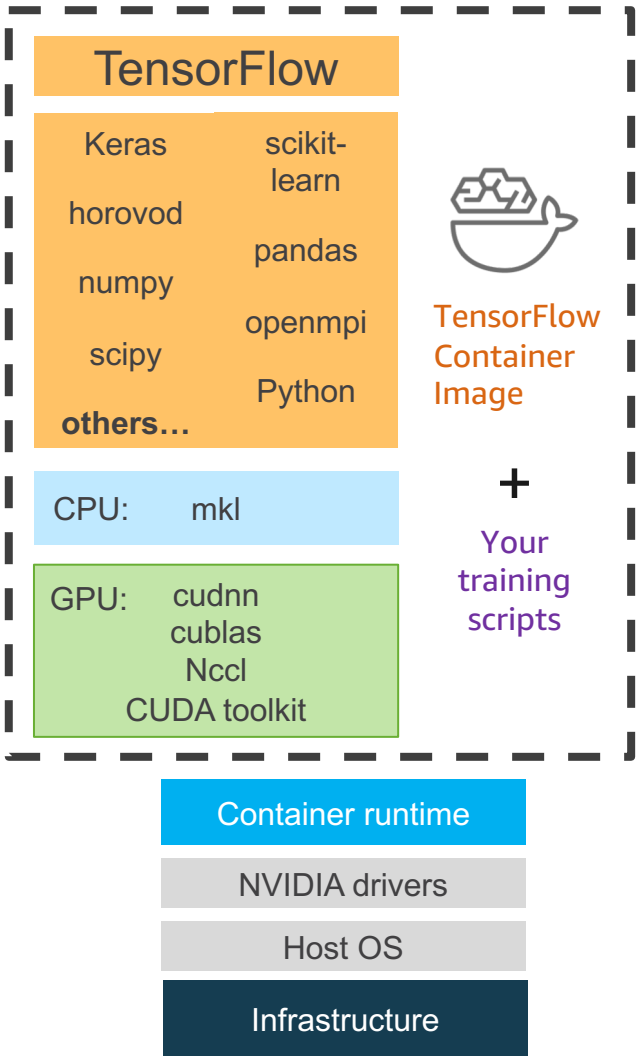


Training cluster



aws

Containers for Machine Learning



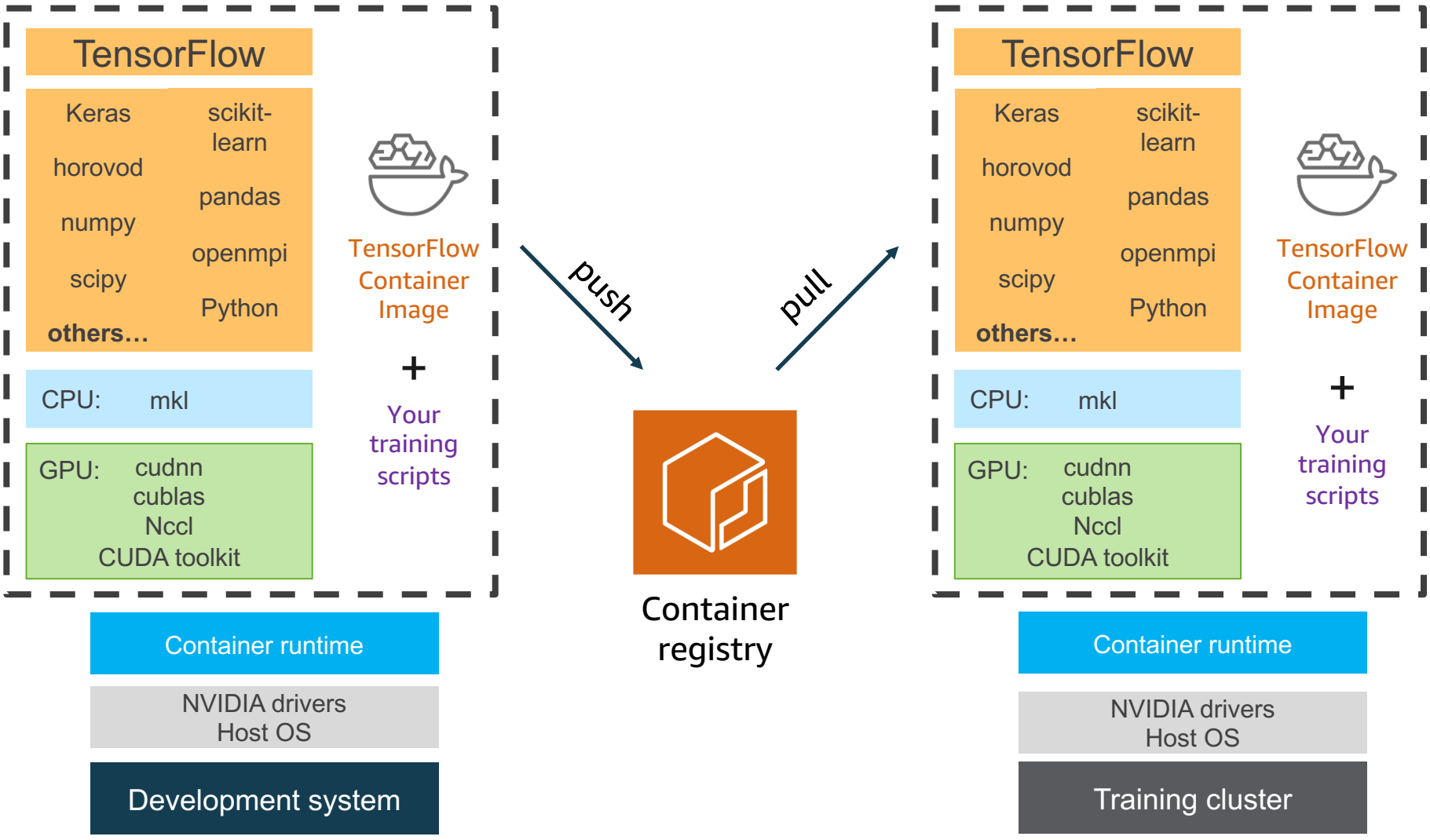
Packages:

- Training code
- Dependencies
- Configurations

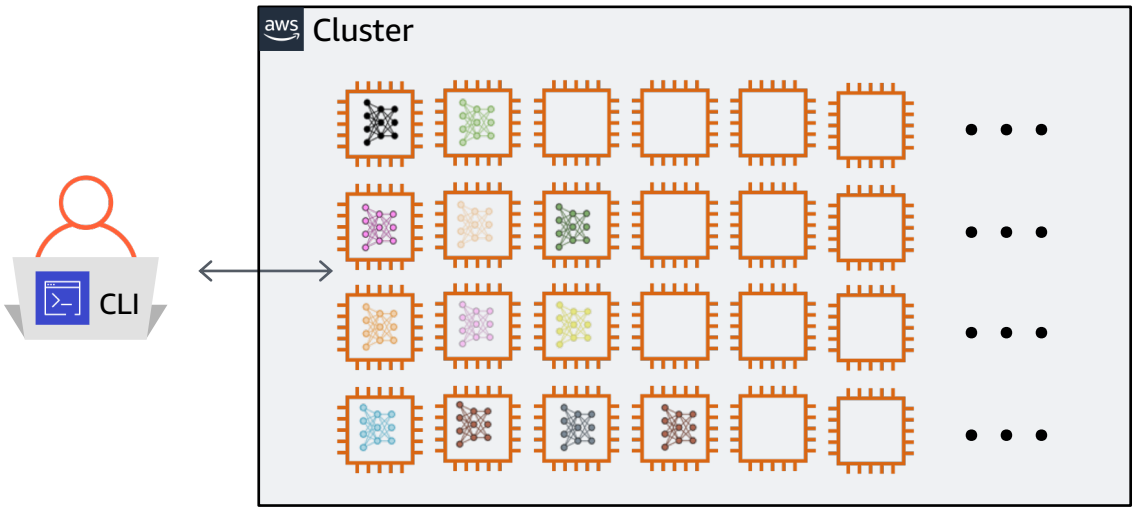
ML environments that are:

- Lightweight
- Portable
- Scalable
- Consistent

aws


TensorFlow
Keras
scikit-learn
horovod
pandas
numpy
openmpi
scipy
Python
others…
CPU: mkl
GPU: cudnn
cublas
Nccl
CUDA toolkit
TensorFlow Container Image
+
Your training scripts
push
pull
Container registry
Container runtime
NVIDIA drivers
Host OS
Development system
Training cluster

Challenges with scaling deep learning training



1. **Software dependencies**

2. **Infrastructure management**

aws

ML infrastructure and cluster management

ML services

Fully-managed service that covers the entire machine learning workflow

Amazon SageMaker


Jupyter notebook instances


high performance algorithms


Large-scale training


Optimization


One-click deployment


Fully managed with auto-scaling

Management

Deployment, scheduling, scaling, and management of containerized applications


Amazon Elastic Container Service (Amazon ECS)


Amazon Elastic Kubernetes Service (Amazon EKS)

Compute

Where the containers run


Amazon EC2

Image registry

Container image repository


Amazon Elastic Container Registry (Amazon ECR)

aws

Amazon SageMaker

Use built-in algorithms

- Bring your own data

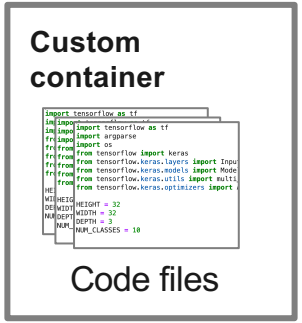
- K-Means Clustering
- Principal Component Analysis
- Neural Topic Modelling
- Factorization Machines
- Linear Learner (Regression)
- BlazingText
- Reinforcement learning
- XGBoost
- Topic Modeling (LDA)
- Image Classification
- Seq2Seq
- Linear Learner (Classification)
- DeepAR Forecasting

Use deep learning frameworks

- Bring your own data
- Bring your own script

Use custom containers

- Bring your own data
- Bring your own container



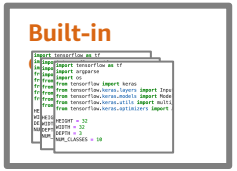
aws

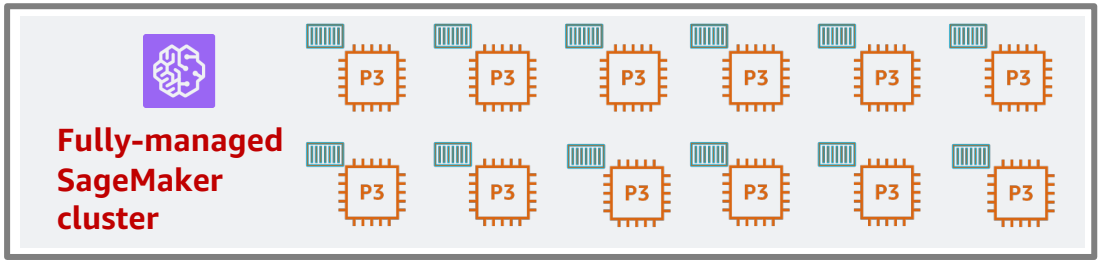
Workflow: Amazon SageMaker built-in

Built-in algorithm

Container registry

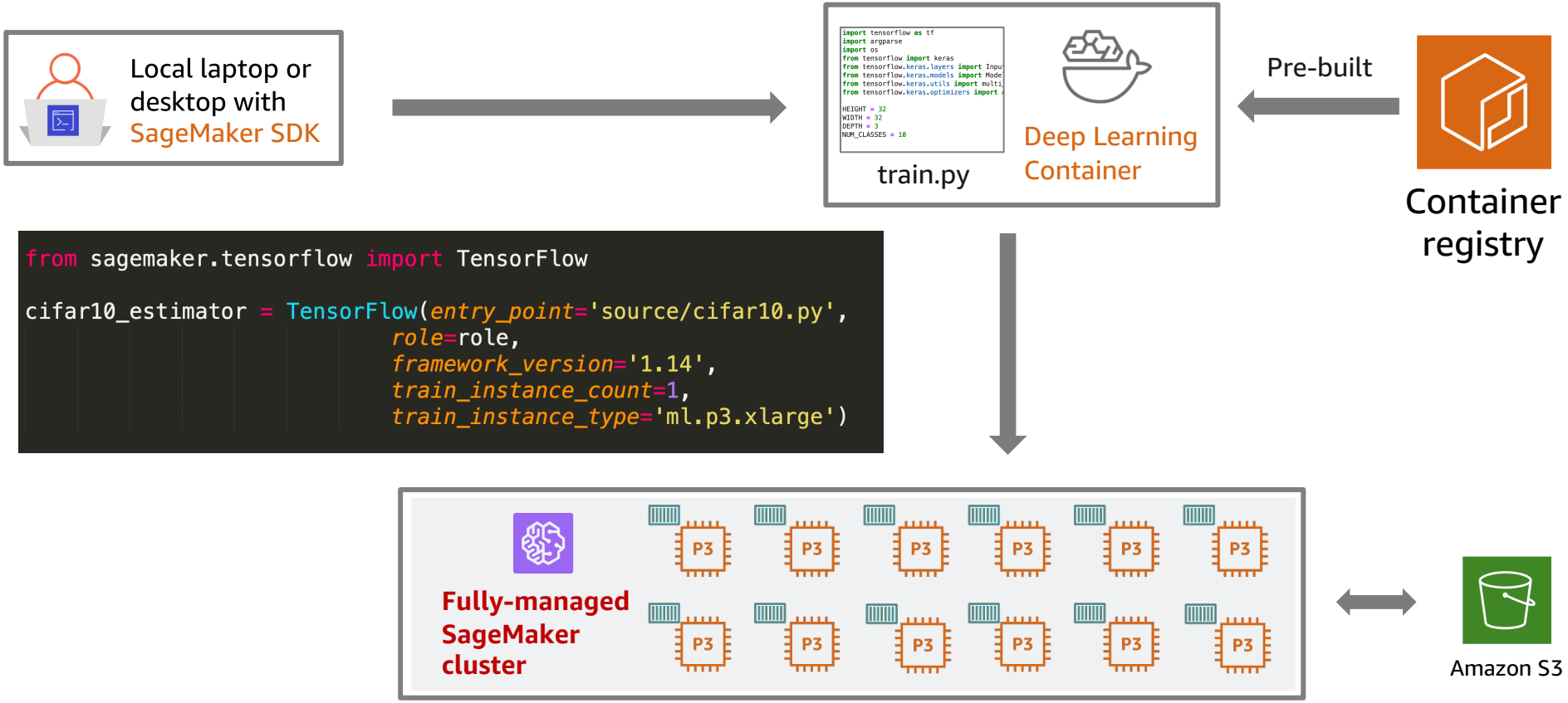
```
from sagemaker.amazon.amazon_estimator import get_image_uri
training_image = get_image_uri(sess.boto_region_name,
                               'image-classification', repo_version="latest")

ic = sagemaker.estimator.Estimator(training_image,
                                   role,
                                   train_instance_count=1,
                                   train_instance_type='ml.p3.xlarge',
                                   train_volume_size = 50,
                                   train_max_run = 360000,
                                   input_mode= 'File',
                                   output_path=s3_output_location,
                                   sagemaker_session=sess)
```



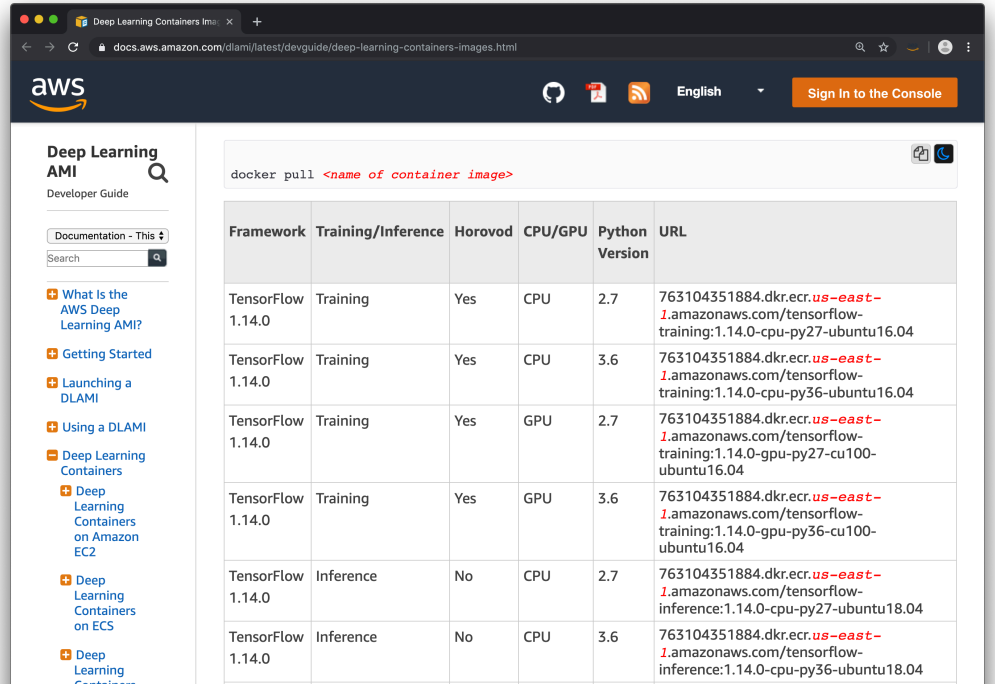


Amazon S3

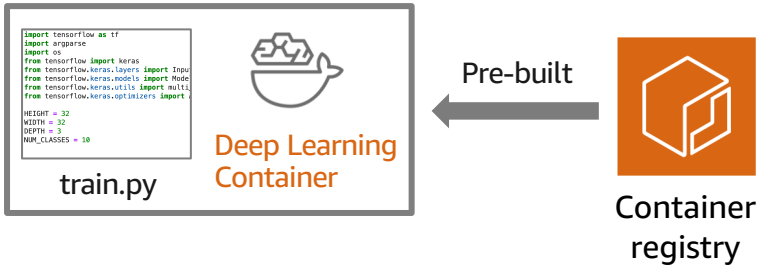
Workflow: Amazon SageMaker deep learning



AWS Deep Learning Containers



Framework	Training/Inference	Horovod	CPU/GPU	Python Version	URL
TensorFlow 1.14.0	Training	Yes	CPU	2.7	763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training:1.14.0-cpu-py27-ubuntu16.04
TensorFlow 1.14.0	Training	Yes	CPU	3.6	763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training:1.14.0-cpu-py36-ubuntu16.04
TensorFlow 1.14.0	Training	Yes	GPU	2.7	763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training:1.14.0-gpu-py27-cu100-ubuntu16.04
TensorFlow 1.14.0	Training	Yes	GPU	3.6	763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training:1.14.0-gpu-py36-cu100-ubuntu16.04
TensorFlow 1.14.0	Inference	No	CPU	2.7	763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference:1.14.0-cpu-py27-ubuntu18.04
TensorFlow 1.14.0	Inference	No	CPU	3.6	763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference:1.14.0-cpu-py36-ubuntu18.04



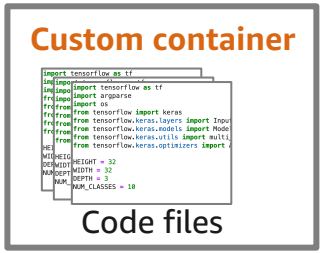
Prepackaged machine learning container images fully configured and validated

Optimized for performance with latest NVIDIA driver, CUDA libraries, and Intel libraries

https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-images.html

Amazon SageMaker custom container

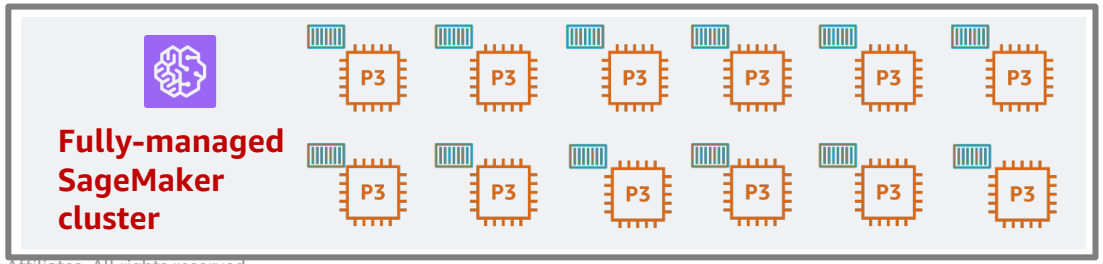
Docker build



push

Container registry

```
ic = sagemaker.estimator.Estimator(training_image,
                                   role,
                                   train_instance_count=1,
                                   train_instance_type='ml.p3.xlarge',
                                   train_volume_size = 50,
                                   train_max_run = 360000,
                                   input_mode= 'File',
                                   output_path=s3_output_location,
                                   sagemaker_session=sess)
```



Amazon S3

Amazon EKS, Kubeflow and Katib

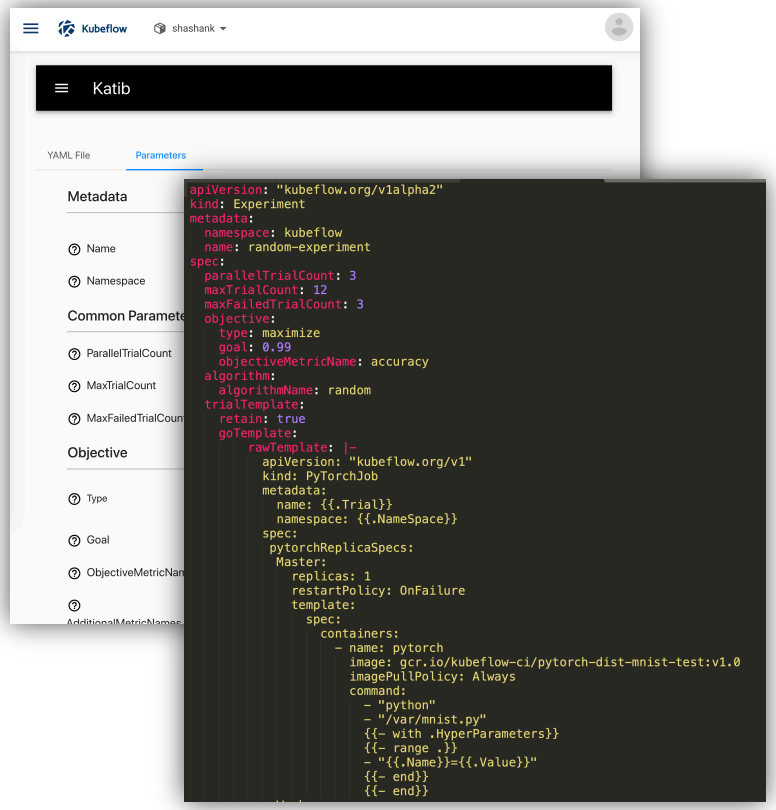
Katib

Hyperparameter Tuning and Neural Architecture Search

Kubeflow

Machine learning workflows on Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS)

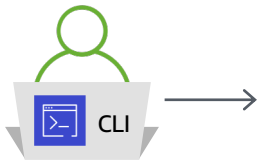


aws

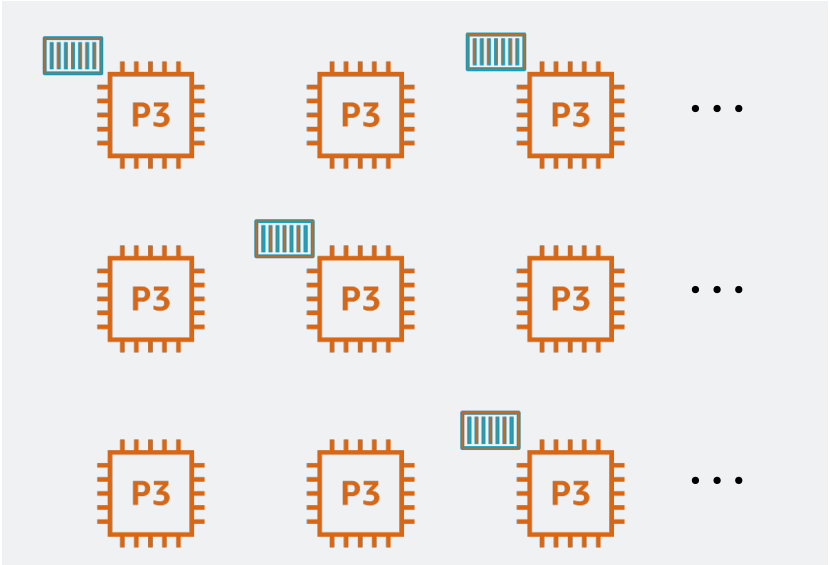
Create a Kubernetes cluster

Create cluster

```
eksctl create cluster \
--name eks-gpu \
--version 1.12 \
--region us-west-2 \
--nodegroup-name gpu-nodes \
--node-type p3.8xlarge \
--nodes 8 \
--timeout=40m \
--ssh-access \
--ssh-public-key=<public-key> \
--auto-kubeconfig
```

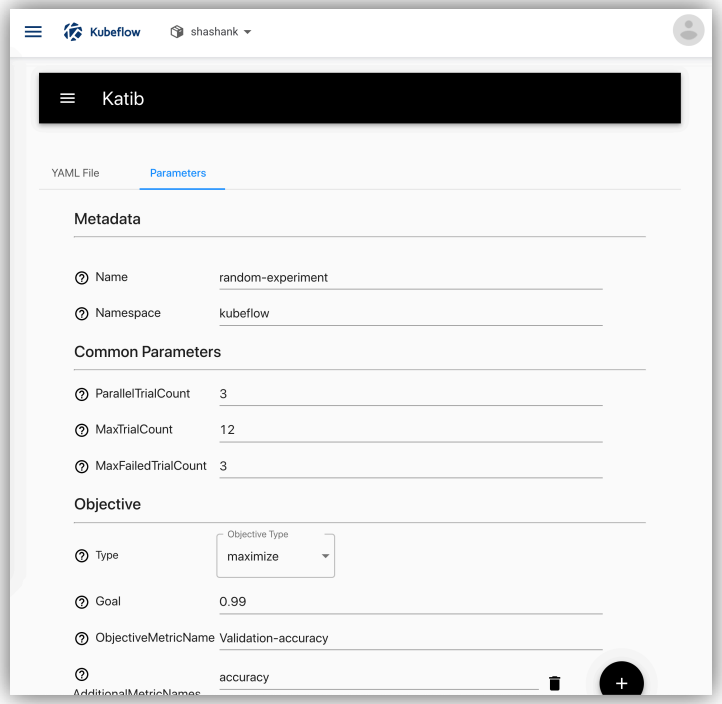


Submit a training jobs

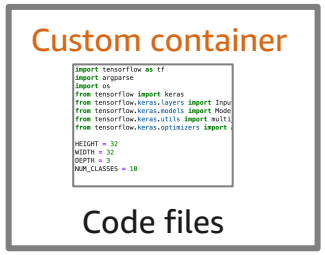


Setting up kubeflow in kubernetes

```
curl --silent --location
https://github.com/kubeflow/kubeflow/releases/download/
v0.7.0-rc.5/kfctl_v0.7.0-rc.5-0-g18fba77a_linux.tar.gz
| tar xz


export
CONFIG="https://raw.githubusercontent.com/kubeflow/mani
fests/v0.7-branch/kfdef/kfctl_aws.yaml"


export KF_NAME=${CLUSTER_NAME}
export KF_DIR=${KF_NAME}
mkdir -p ${KF_DIR}
cd ${KF_DIR}
kfctl build -V -f ${CONFIG}
# update kfctl_aws.yaml with node and cluster name
kfctl apply -V -f kfctl_aws.yaml
```



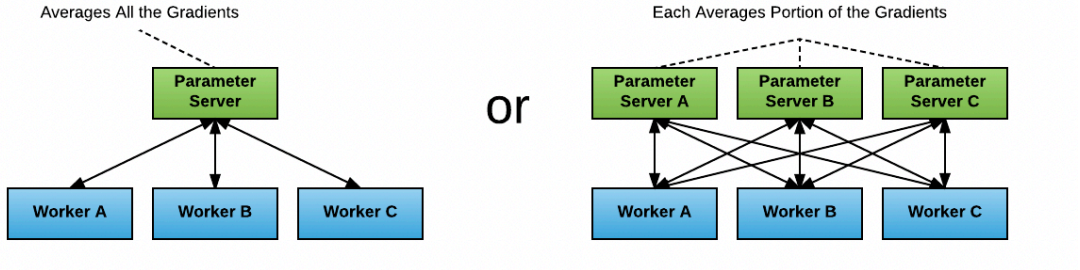
Amazon EKS + Kubeflow



MPIJob YAML specification file

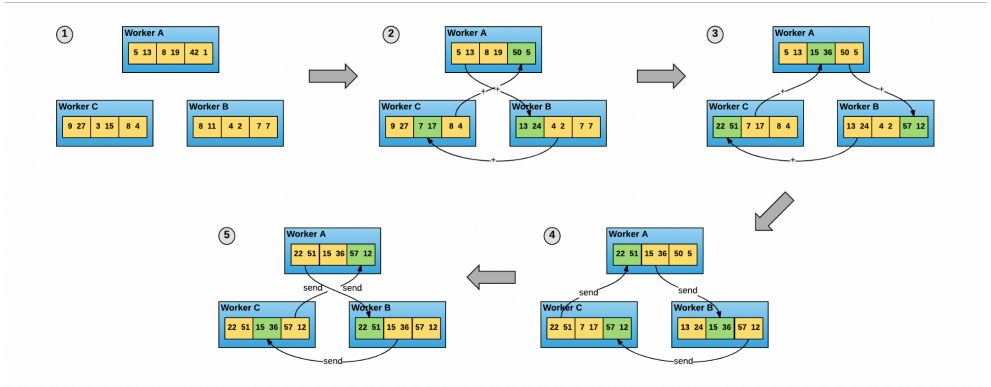
```yaml
apiVersion: kubeflow.org/v1alpha1
kind: MPIJob
metadata:
  name: eks-tf-distributed-training
spec:
  replicas: 2
  template:
    metadata:
      annotations:
        sidecar.istio.io/inject: "false"
    spec:
      restartPolicy: Never
      containers:
      - name: eks-tf-dist-job
```

Distributed training approaches



Parameter Server

"asynchronous"



Ring All-reduce

"Synchronous"

aws

Horovod: fast and easy distributed deep learning in TensorFlow, Alexander Sergeev and Mike Del Balso (https://arxiv.org/abs/1802.05799)

Horovod and the Ring All-reduce approach

- Library for distributed deep learning with support for multiple frameworks including TensorFlow
- Separates infrastructure from ML engineers
- Uses ring-allreduce and uses Message Passing Interface (MPI) popular in the HPC community
- Infrastructure services such as Amazon SageMaker and Amazon EKS provides container and MPI environment

horovod.ai

aws

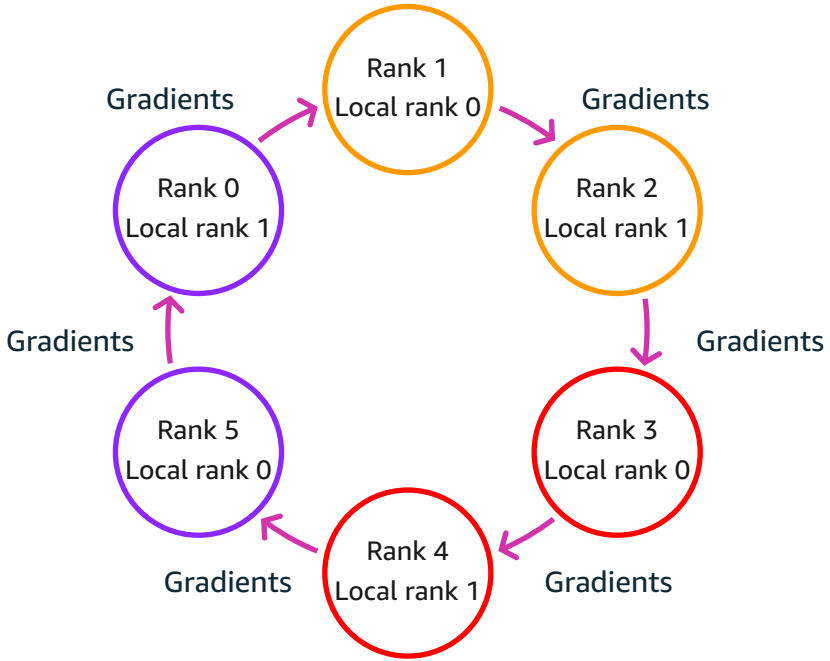
Horovod and the Ring All-reduce approach

Horovod is based on the MPI concepts:
size, rank, local rank, allreduce, allgather, and broadcast.

Size: number of processes.
“nodes”, “instances”, “server”
times number of GPUs

Rank: Unique process ID (size - 1).

Local rank: Unique process ID within the “node”, “instance”, ”server”



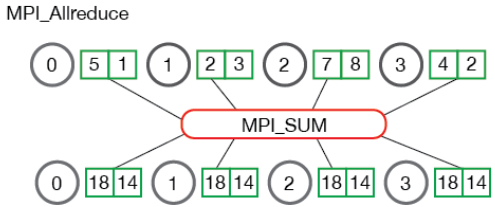
Horovod and the Ring All-reduce approach

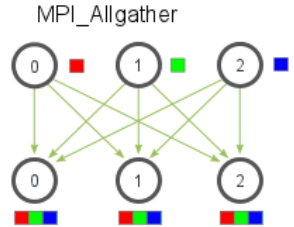
Horovod is based on the MPI concepts:
size, rank, local rank, allreduce, allgather, and broadcast.

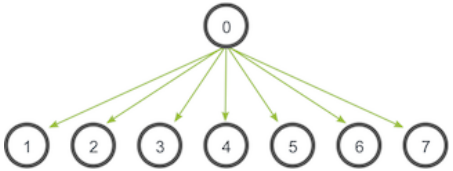
All reduce: aggregates data among multiple processes and distributes results back to them

All gather: gathers data from all processes on every process

Broadcast: broadcasts data from one process, identified by root rank, onto every other process

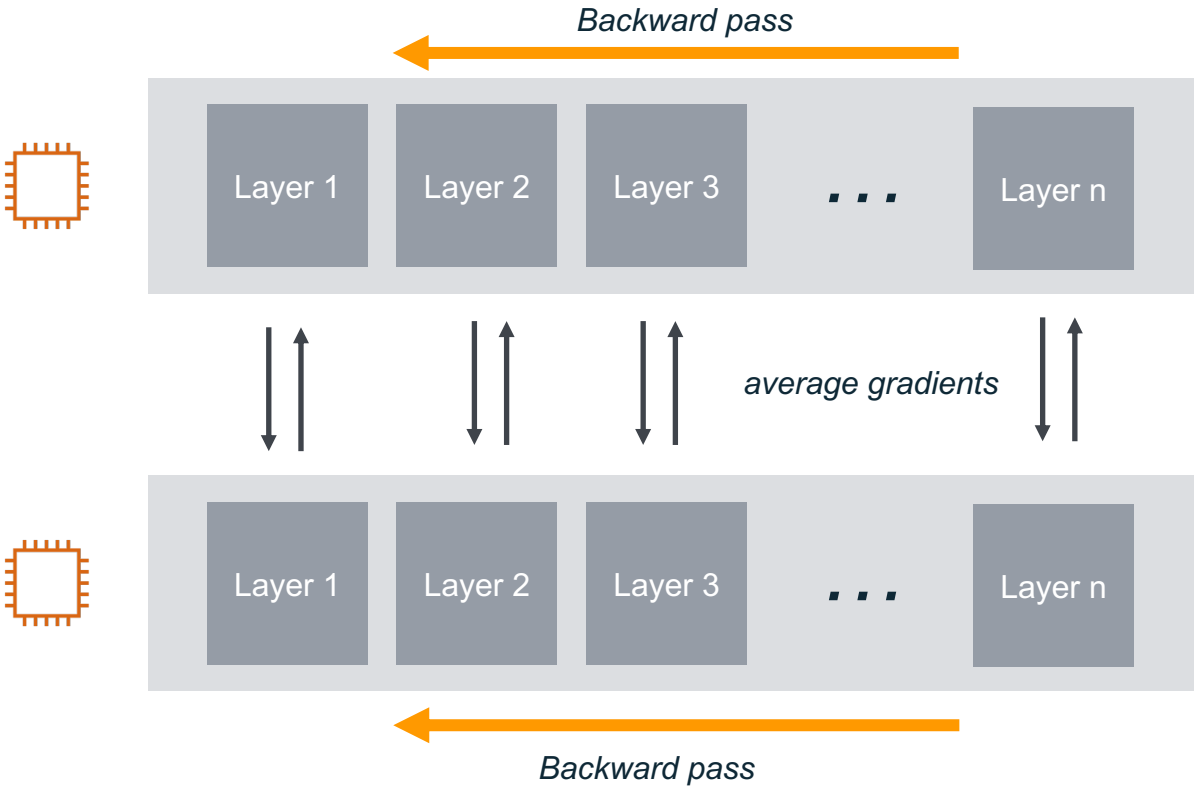






https://mpitutorial.com/tutorials/mpi-scatter-gather-and-allgather/

Add-reduce algorithm



1. Forward pass on each device
2. Backward pass compute gradients
3. ”All reduce” (average and broadcast) gradients across devices
4. Update local variables with “all reduced” gradients

aws

Updating training scripts with Horovod

1. Run `hvd.init()`.

2. Pin a server GPU to be used by this process using `config.gpu_options.visible_device_list`.

3. Scale the learning rate by the number of workers.

4. Wrap the optimizer in `hvd.DistributedOptimizer`.

5. Add `hvd.callbacks.BroadcastGlobalVariablesCallback(0)` to broadcast initial variable states from rank 0 to all other processes.

6. Modify your code to save checkpoints only on worker 0 to prevent other workers from corrupting them.

You'll be performing these steps in Part 1 of the workshop

https://horovod.readthedocs.io/en/latest/keras.html

Using horovod updated training scripts

Horovod will run the same copy of the script on all hosts/servers/nodes/instances


```
horovodrun -np 16 -H server1:4,server2:4,server3:4,server4:4 python training_script.py
```

Workshop outline

distributed-training-workshop.go-aws.com

Prerequsites:

Setup your development environment on Amazon SageMaker notebook instance

Part 1:

Preparing your TensorFlow training script for distributed training with horovod

Part 2:

Running TensorFlow distributed training on Amazon SageMaker

Part 3:

Setting up Amazon EKS and Kubeflow for distributed training on Kubernetes and KubeFlow

aws

distributed-training-workshop.go-aws.com

Part 0 (15 mins)

Setup your development environment on Amazon SageMaker notebook instance

aws

distributed-training-workshop.go-aws.com

Part 1 (30 mins)

Preparing your TensorFlow training script for distributed training with horovod

aws

distributed-training-workshop.go-aws.com

Part 2 (1 hour)

Running TensorFlow distributed training on Amazon SageMaker

distributed-training-workshop.go-aws.com

Part 3 (1 hour)

Setting up Amazon EKS and Kubeflow for distributed training on Kubernetes and KubeFlow

distributed-training-workshop.go-aws.com

Thank you!