Implementing Distributed Training on Multiple Machines

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Explore options available to train PyTorch on the cloud

Parallelize training across machines

Distributed training with the AWS Sagemaker API

Distributed Training in PyTorch

Multiprocessing

Data Parallel

Model Parallel

Distributed Data Parallel

Distributed Training in PyTorch

Multiprocessing

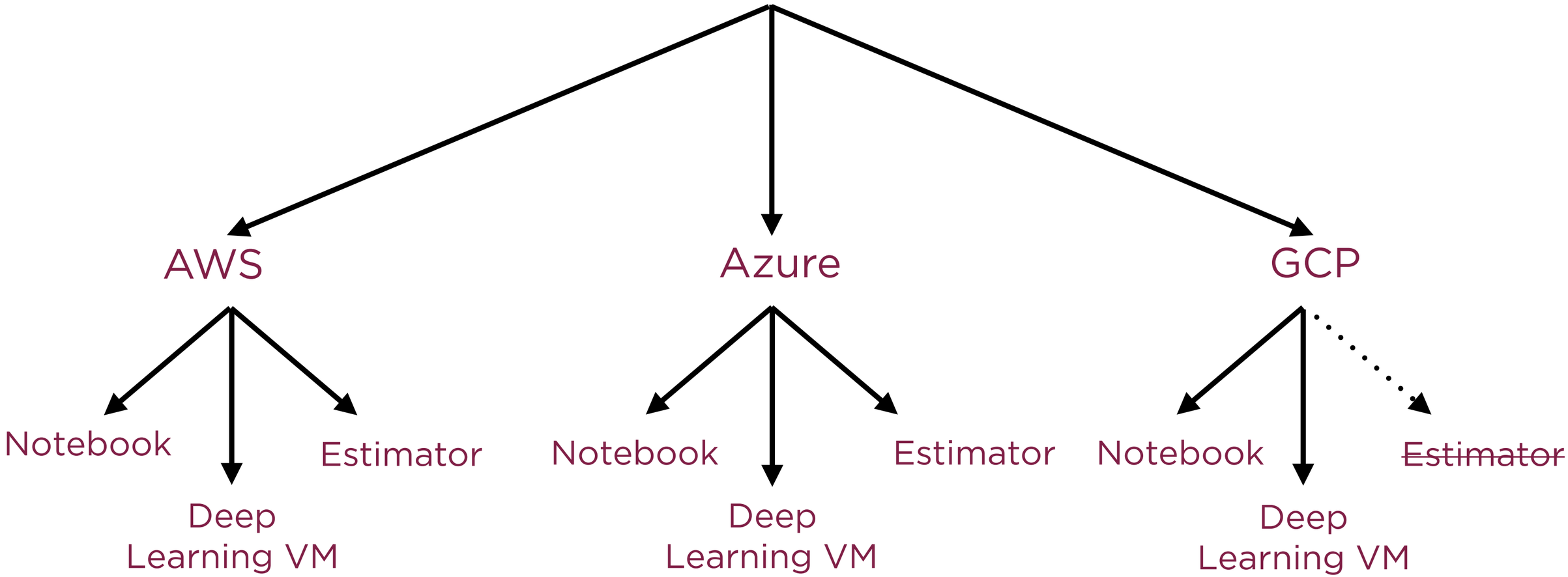
Data Parallel

Model Parallel

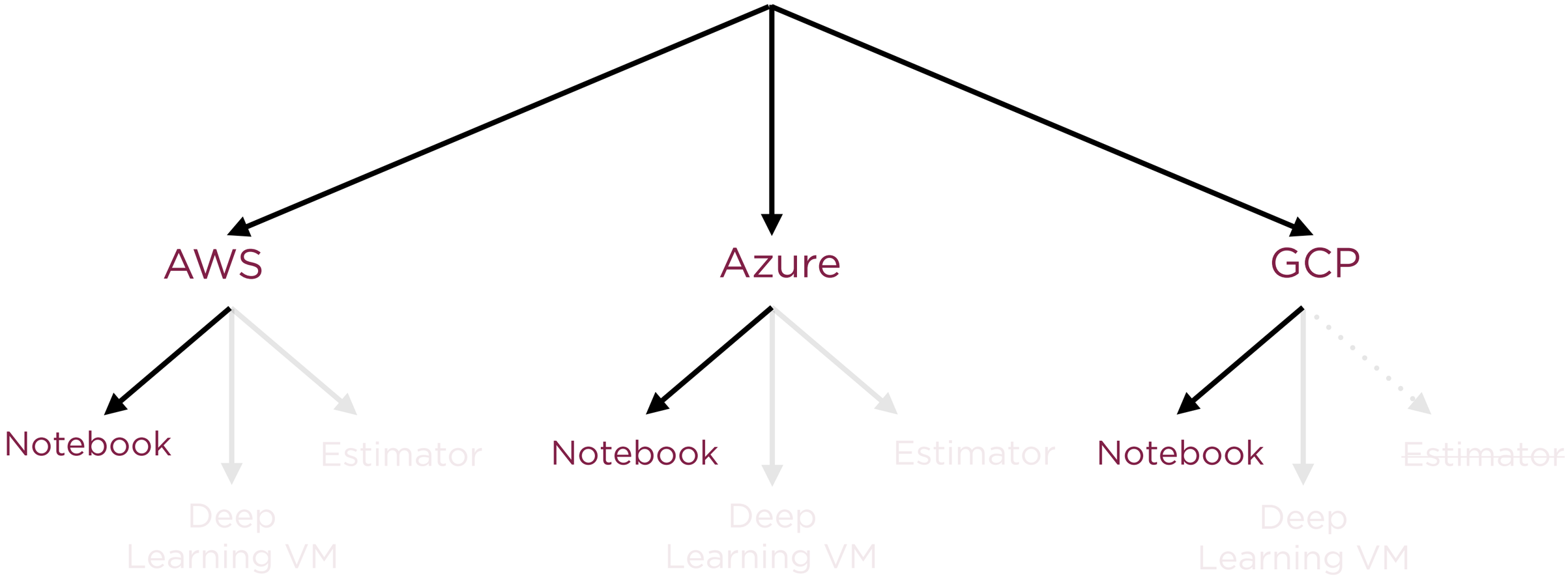
Distributed Data Parallel

PyTorch on the Cloud: Taxonomy of Solutions

PyTorch on the Cloud



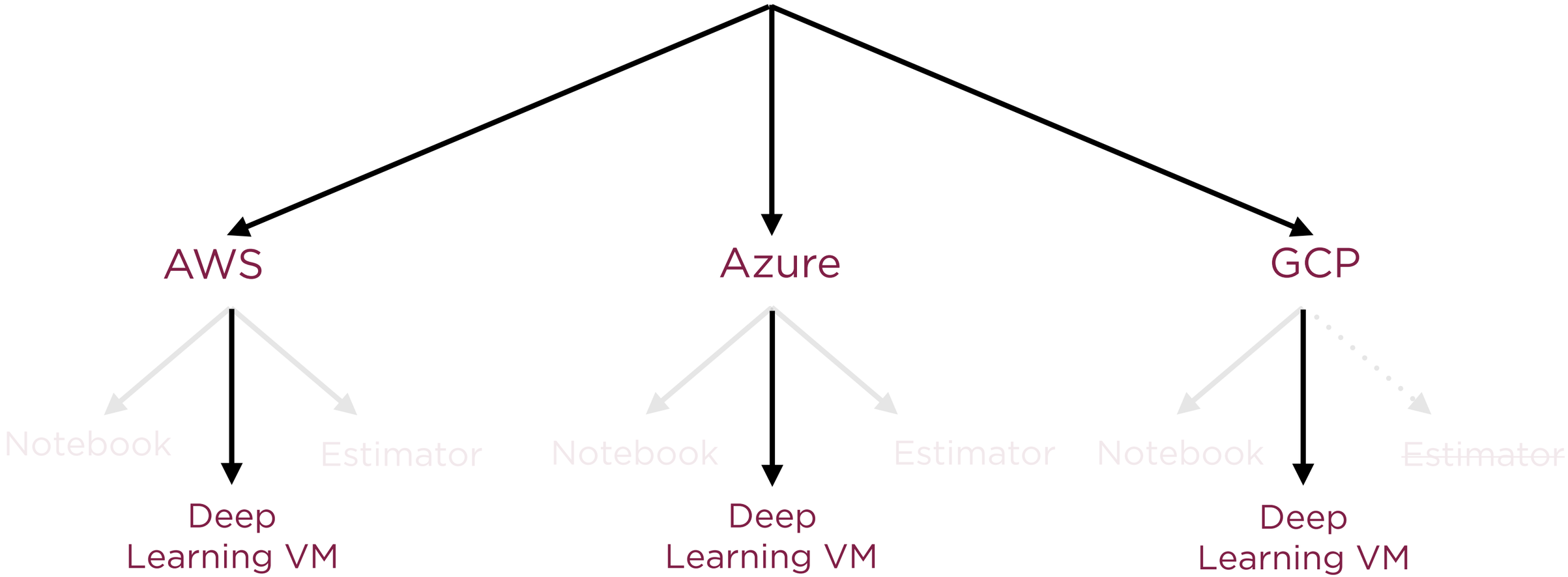
PyTorch on the Cloud



Notebook

Cloud-hosted Python notebook. Could be platform-agnostic (Jupyter) or platform-specific (e.g. Datalab on GCP)

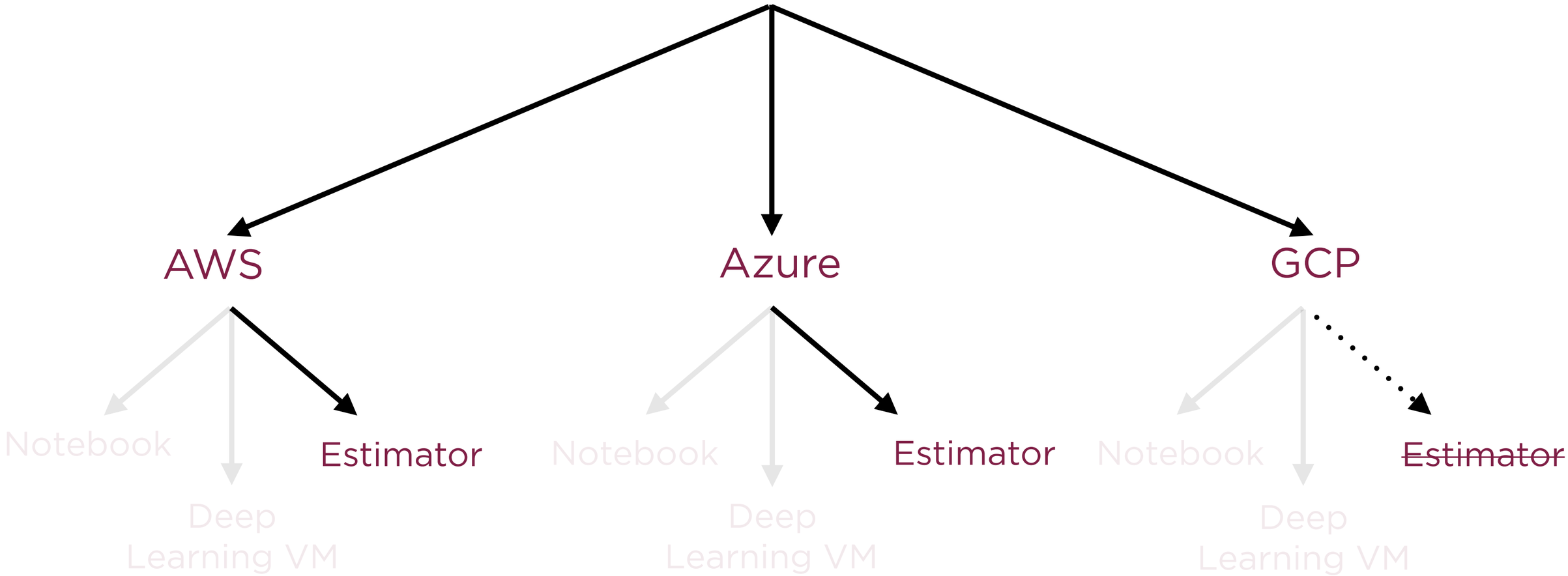
PyTorch on the Cloud



Deep Learning VM

Cloud-specific virtual machine instance (e.g. EC2 on AWS, GCE on GCP) equipped with GPUs for optimized PyTorch performance

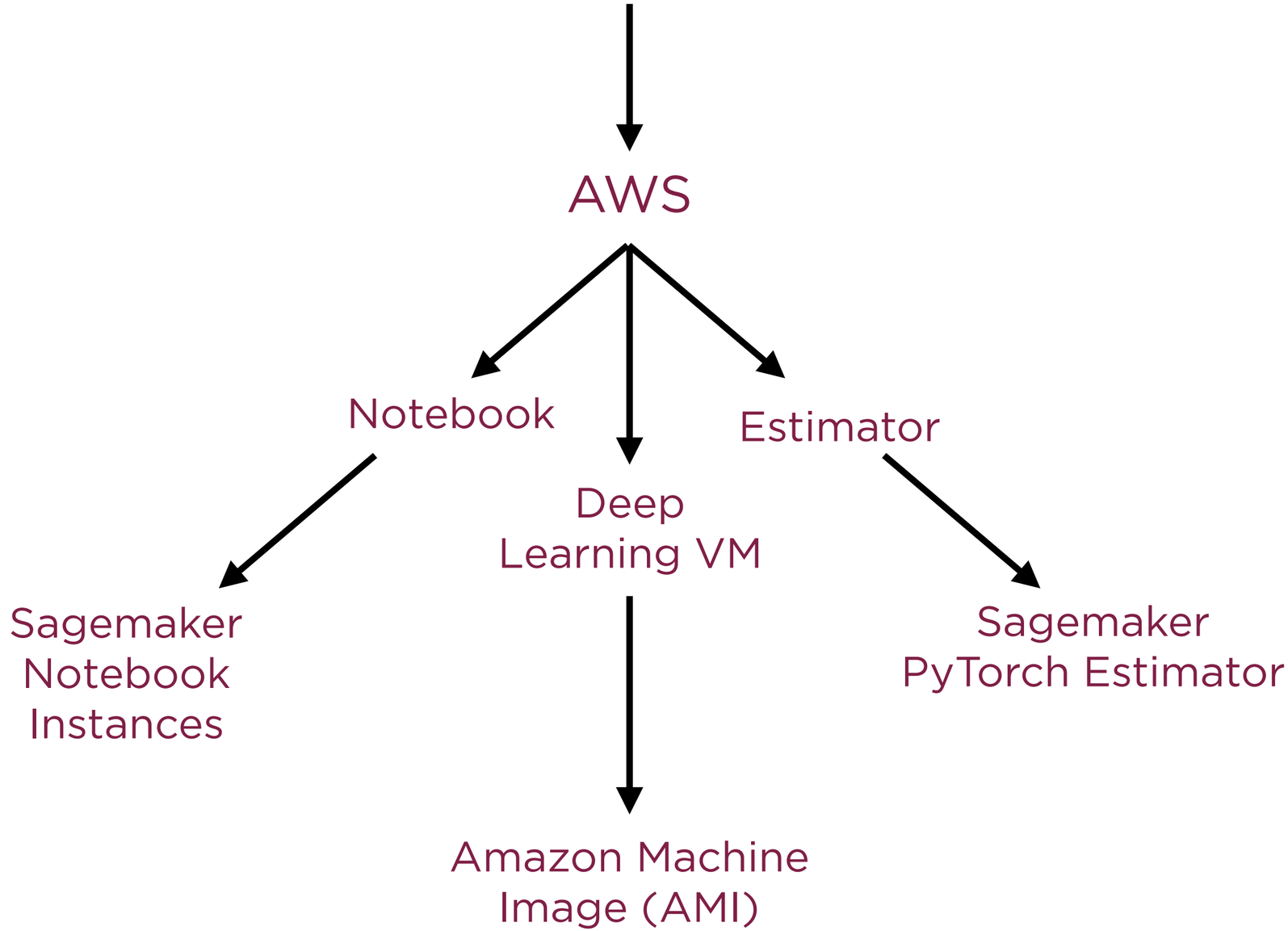
PyTorch on the Cloud



Estimator

High-level API, specific to a cloud platform, that helps build, train, and deploy PyTorch models

PyTorch on the Cloud



PyTorch on SageMaker

SageMaker Python SDK

PyTorch estimators and models

PyTorch open-source container

Distributed Training Backends

Gloo

NCCL

MPI

Choosing Backends

Rule of thumb

- Gloo for distributed CPU training
- NCCL for distributed GPU training
- More fine print in the docs

Choosing Backends

Backends that come with PyTorch

PyTorch distributed currently only supports Linux. By default, the Gloo and NCCL backends are built and included in PyTorch distributed (NCCL only when building with CUDA). MPI is an optional backend that can only be included if you build PyTorch from source. (e.g. building PyTorch on a host that has MPI installed.)

Which backend to use?

In the past, we were often asked: "which backend should I use?".

- Rule of thumb
  - Use the NCCL backend for distributed **GPU** training
  - Use the Gloo backend for distributed **CPU** training.
- GPU hosts with InfiniBand interconnect
  - Use NCCL, since it's the only backend that currently supports InfiniBand and GPUDirect.
- GPU hosts with Ethernet interconnect
  - Use NCCL, since it currently provides the best distributed GPU training performance, especially for multiprocess single-node or multi-node distributed training. If you encounter any problem with NCCL, use Gloo as the fallback option. (Note that Gloo currently runs slower than NCCL for GPUs.)
- CPU hosts with InfiniBand interconnect
  - If your InfiniBand has enabled IP over IB, use Gloo, otherwise, use MPI instead. We are planning on adding InfiniBand support for Gloo in the upcoming releases.
- CPU hosts with Ethernet interconnect
  - Use Gloo, unless you have specific reasons to use MPI.

Demo

Running distributed training on multiple machines in a cluster using torch.nn.parallel.DistributedDataParallel

Summary

Explore options available to train PyTorch on the cloud

Parallelize training across machines

Distributed training with the AWS Sagemaker API