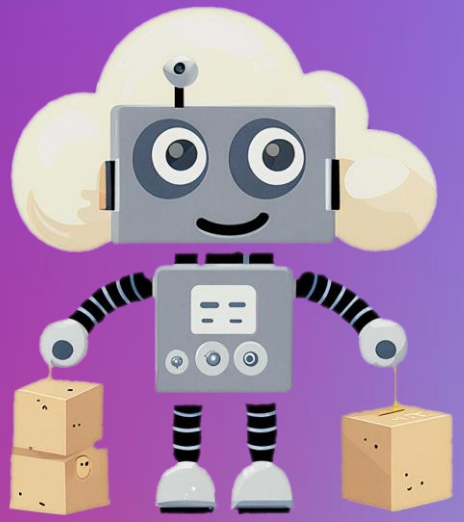
WOR101

Efficiently train and deploy LLMs on AWS Trainium and Inferentia2 accelerators

Luca Perrozzi
Solutions Architect
AWS

Dmitri Laptev
Solutions Architect
AWS

Roy Allela
AIML Specialist Solutions Architect
AWS



Agenda

Trends fueling generative AI innovation

Common challenges and trade-offs

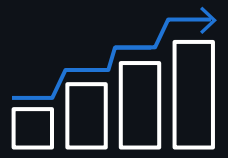
AWS Trainium

AWS Inferentia2

Hands-on Lab

Q&A

Trends fueling artificial intelligence (AI) innovation



Large language models

Explosive growth of large language models based on transformer architectures

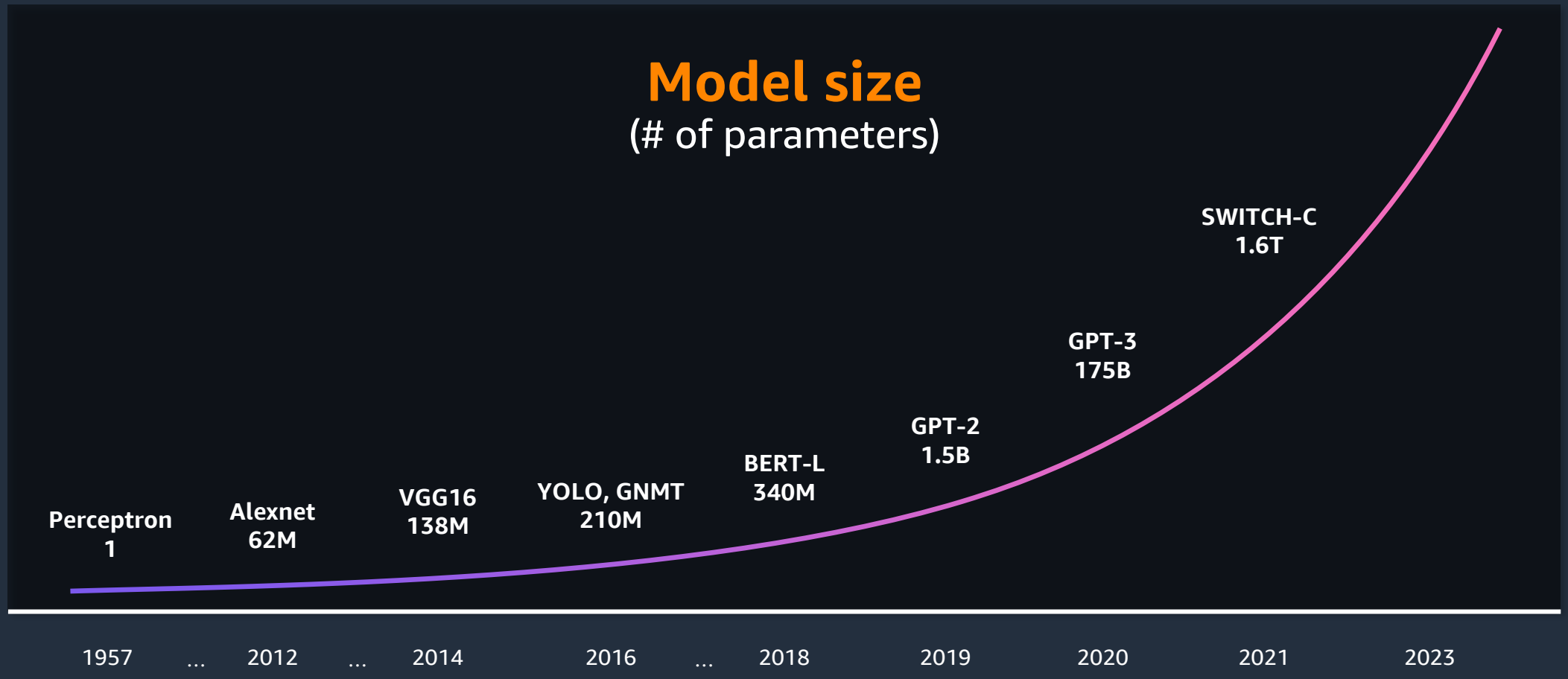
Self-supervised learning

Data scientists are no longer limited to human-labeled data for training models

Open-source momentum

Increased momentum from industry offering open-source, pretrained models

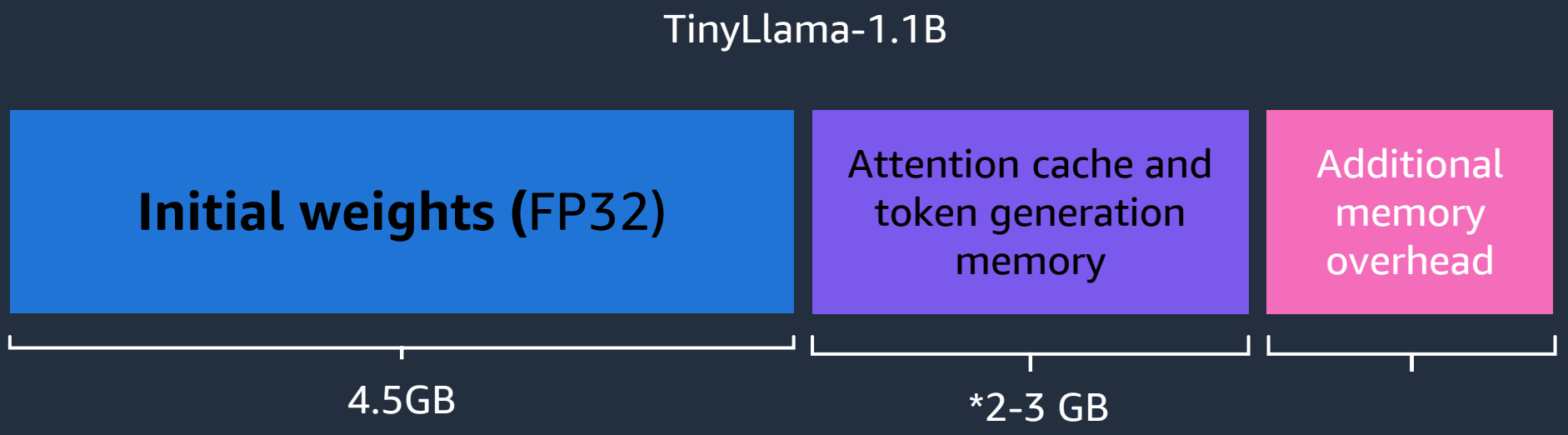
AI models are getting bigger...



Challenges of LLM

EXAMPLE: TINYLLAMA-1.1B MODEL FOR INFERENCE

Model size = Parameters (1-billion) x 4 bytes (FP32) = 4.5 GB



* The required memory depends on input/output sequence length, decoding strategy, model architecture, and batch size

Size comparison:
FP 16 = ½ FP 32
BF 16 = ½ FP 32
INT 8 = ¼ FP 32

Common challenges and trade-offs

Performance

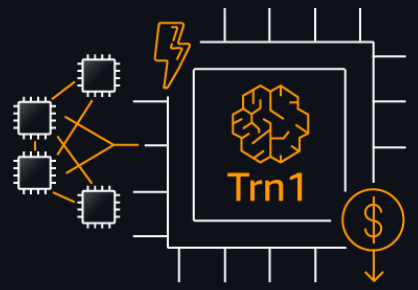
Cost

Ease of use

Energy efficiency

AWS purpose-built accelerators for generative AI

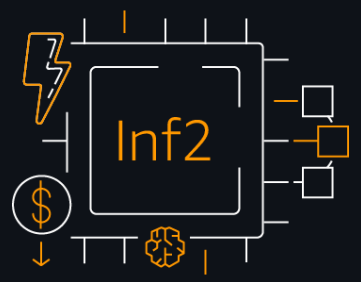
AWS Trainium



The most cost efficient for high-performance training of LLMs and diffusion models

Up to 50% cost-to-train savings over comparable GPU-based instances

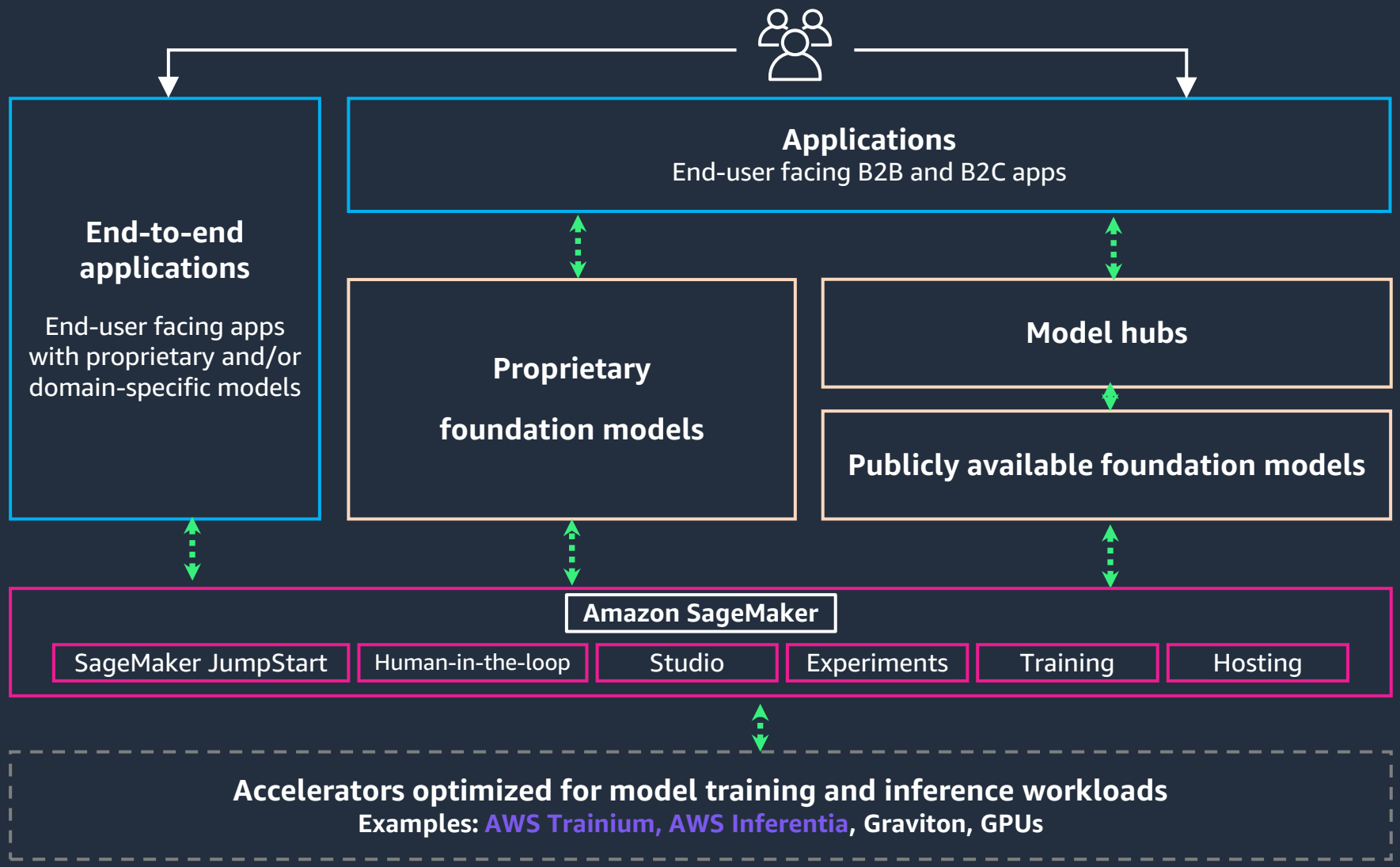
AWS Inferentia2



High performance at the lowest cost per inference for LLMs and diffusion models

Up to 40% better price performance than comparable GPU-based instances

Generative AI: Lay of the land using SageMaker



Customer momentum for AWS Trainium and AWS Inferentia

airbnb
AUTODESK
Finch COMPUTING
sprinklr
SCREENING EAGLE
amazon robotics
amazon advertising
amazon alexa
The Asahi Shimbun
NTTPC
CONDÉ NAST
Anthem
DISCO
SKYWATCH
Money Forward
Amazon Rekognition
Talroo™
DMP DIGITAL MEDIA PROFESSIONALS
prime video
Helixon 华深智药
CACTUS

Better together:
Support for AWS Trainium and AWS Inferentia

PyTorch

TensorFlow

OpenXLA

AWS Neuron SDK

RAY

OctoML

Weights & Biases

Getting started with AWS Trainium and AWS Inferentia

Launch instances
(Trn1, Inf2, Inf1)

AWS Deep Learning AMIs

AWS Deep Learning Containers

Bring your own model

PyTorch

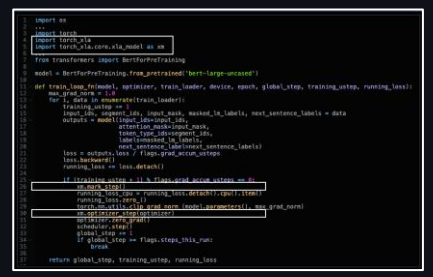
TensorFlow

Hugging Face

OpenXLA

Modify a few lines of code → run

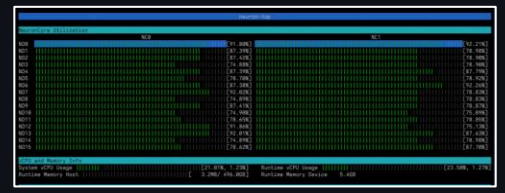
Neuron SDK



https://awsdocs-neuron.readthedocs-hosted.com
https://github.com/aws/aws-neuron-sdk

Monitor, tune, scale

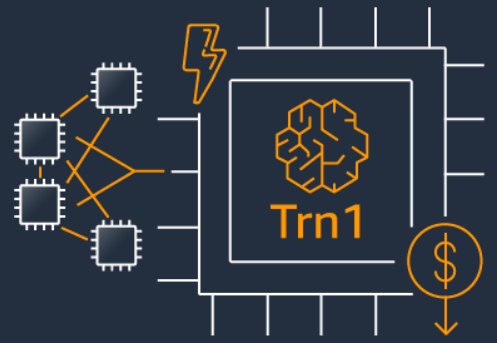
Neuron SDK



AWS Trainium

Amazon EC2 Trn1 instances powered by AWS Trainium

COST-EFFICIENT, HIGH-PERFORMANCE DL TRAINING INSTANCES



High performance on training of popular NLP models on AWS

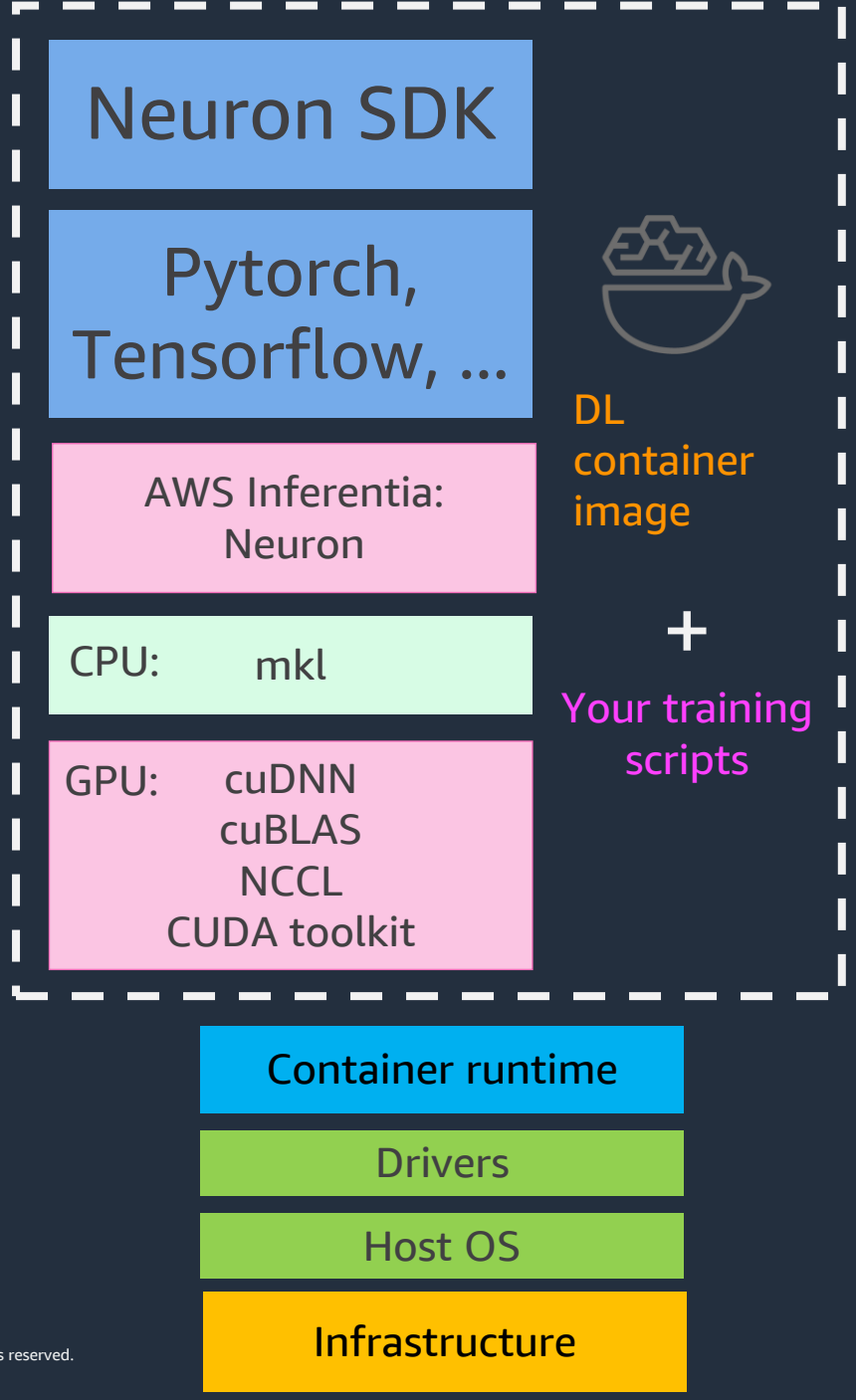
Up to 50% cost-to-train savings

Up to 4x network bandwidth

Instance size	vCPUs	Instance memory	Trainium chips	Accelerator memory	NeuronLink	Instance networking	On-demand price
Trn1.2xlarge	8	32 GB	1	32 GB	N/A	Up to 10 Gbps	$1.34/hr
Trn1.32xlarge	128	512 GB	16	512 GB	Yes	800 Gbps	$21.5/hr
Trn1n.32xlarge	128	512 GB	16	512 GB	Yes	1600 Gbps	$24.78/hr

Trn1 available now in US-East-1 (N. Virginia) and US-West-2 (Oregon)

AWS Deep Learning Containers



ML environments that are:

- Lightweight
- Portable
- Scalable
- Consistent

Packages:

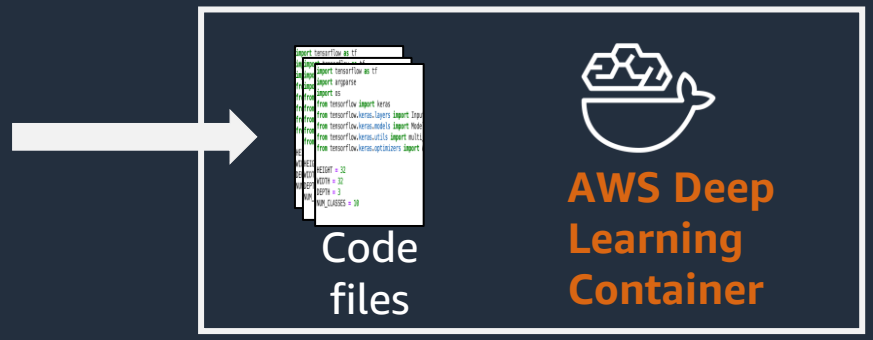
- Training code
- Dependencies
- Configurations

High Level Workflow

```python
import argparse
import os

if __name__ =='__main__':

    parser = argparse.ArgumentParser()

    # hyperparameters sent by the client are passed as command-line arguments to the script.
    parser.add_argument('--epochs', type=int, default=50)
    parser.add_argument('--batch-size', type=int, default=64)
    parser.add_argument('--learning-rate', type=float, default=0.05)
    parser.add_argument('--use-cuda', type=bool, default=False)

    # Data, model, and output directories
    parser.add_argument('--output-data-dir', type=str, default=os.environ['SM_OUTPUT_DATA_DIR'])
    parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])
    parser.add_argument('--train', type=str, default=os.environ['SM_CHANNEL_TRAIN'])
    parser.add_argument('--test', type=str, default=os.environ['SM_CHANNEL_TEST'])

    args, _ = parser.parse_known_args()

    # ... load from args.train and args.test, train a model, write model to args.model_dir.
```

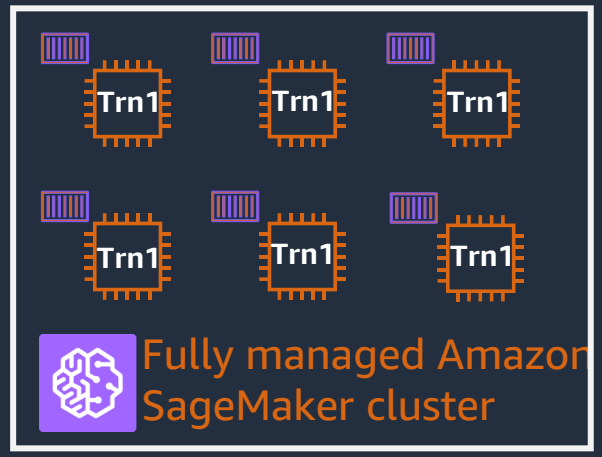
```python
from sagemaker.pytorch import PyTorch

pt_estimator = PyTorch(entry_point="pytorch-train.py",
                       role=role,
                       instance_type= "ml.trn1.2xlarge",
                       instance_count=1,
                       framework_version="1.6.0",
                       py_version="py36",
                       hyperparameters=hyperparameters)

pt_estimator.fit({'training': inputs})
```

Local laptop or desktop with Amazon SageMaker SDK



Amazon ECR

Container registry



Amazon S3

Ease of use

Import model from Hugging Face

Use DL Neuron containers

```python
from optimum.neuron import NeuronTrainer as Trainer
from optimum.neuron.distributed import lazy_load_for_parallelism

# [...]

    if model_args.model_name_or_path:
        torch_dtype = (
            model_args.torch_dtype
            if model_args.torch_dtype in ["auto", None]
            else getattr(torch, model_args.torch_dtype)
        )
        with lazy_load_for_parallelism(tensor_parallel_size=training_args.tensor_parallel_size):
            model = AutoModelForCausalLM.from_pretrained(
                model_args.model_name_or_path,
                from_tf=bool(".ckpt" in model_args.model_name_or_path),
                config=config,
                cache_dir=model_args.cache_dir,
                revision=model_args.model_revision,
                use_auth_token=True if model_args.use_auth_token else None,
                torch_dtype=torch_dtype,
                low_cpu_mem_usage=model_args.low_cpu_mem_usage,
            )

# [...]

    # Initialize our Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset if training_args.do_train else None,
        eval_dataset=eval_dataset if training_args.do_eval else None,
        tokenizer=tokenizer,
        # Data collator will default to DataCollatorWithPadding, so we change it.
        data_collator=default_data_collator,
        compute_metrics=compute_metrics if training_args.do_eval and not is_torch_tpu_available() else None,
        preprocess_logits_for_metrics=preprocess_logits_for_metrics if training_args.do_eval and not is_torch_tpu_available() else None,
    )

    # Training
    if training_args.do_train:
        checkpoint = None
        if training_args.resume_from_checkpoint is not None:
            checkpoint = training_args.resume_from_checkpoint
        elif last_checkpoint is not None:
            checkpoint = last_checkpoint
        train_result = trainer.train(resume_from_checkpoint=checkpoint)
        trainer.save_model()  # Saves the tokenizer too for easy upload

        metrics = train_result.metrics

        max_train_samples = (
            data_args.max_train_samples if data_args.max_train_samples is not None else len(train_dataset)
        )
        metrics["train_samples"] = min(max_train_samples, len(train_dataset))

        trainer.log_metrics("train", metrics)
        trainer.save_metrics("train", metrics)
        trainer.save_state()
```

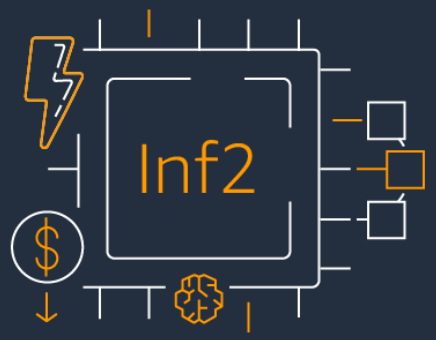
https://github.com/huggingface/optimum-neuron

AWS Inferentia2

NEW

Amazon EC2 Inf2 instances powered by AWS Inferentia2

HIGH PERFORMANCE AT THE LOWEST COST FOR GENERATIVE AI MODELS



Up to 4x higher throughput and 10x lower latency

9.8 TB/s aggregated accelerator memory bandwidth

Support for ultra-large generative AI models

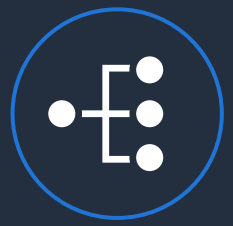
Instance size	vCPUs	Instance memory	Inferentia2 chips	Accelerator memory	NeuronLink	Instance networking	On-demand price
Inf2.xlarge	4	16 GB	1	32 GB	N/A	Up to 15 Gbps	$0.76/hr
Inf2.8xlarge	32	128 GB	1	32 GB	N/A	Up to 25 Gbps	$1.97/hr
Inf2.24xlarge	96	384 GB	6	192 GB	Yes	50 Gbps	$6.49/hr
Inf2.48xlarge	192	768 GB	12	384 GB	Yes	100 Gbps	$12.98/hr

Inf2 available now in US-East-1 (N. Virginia) and US-East-2 (Ohio)

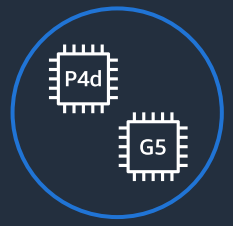
Large Model Inference (LMI) container

Large ML models

with 100 billion + parameters



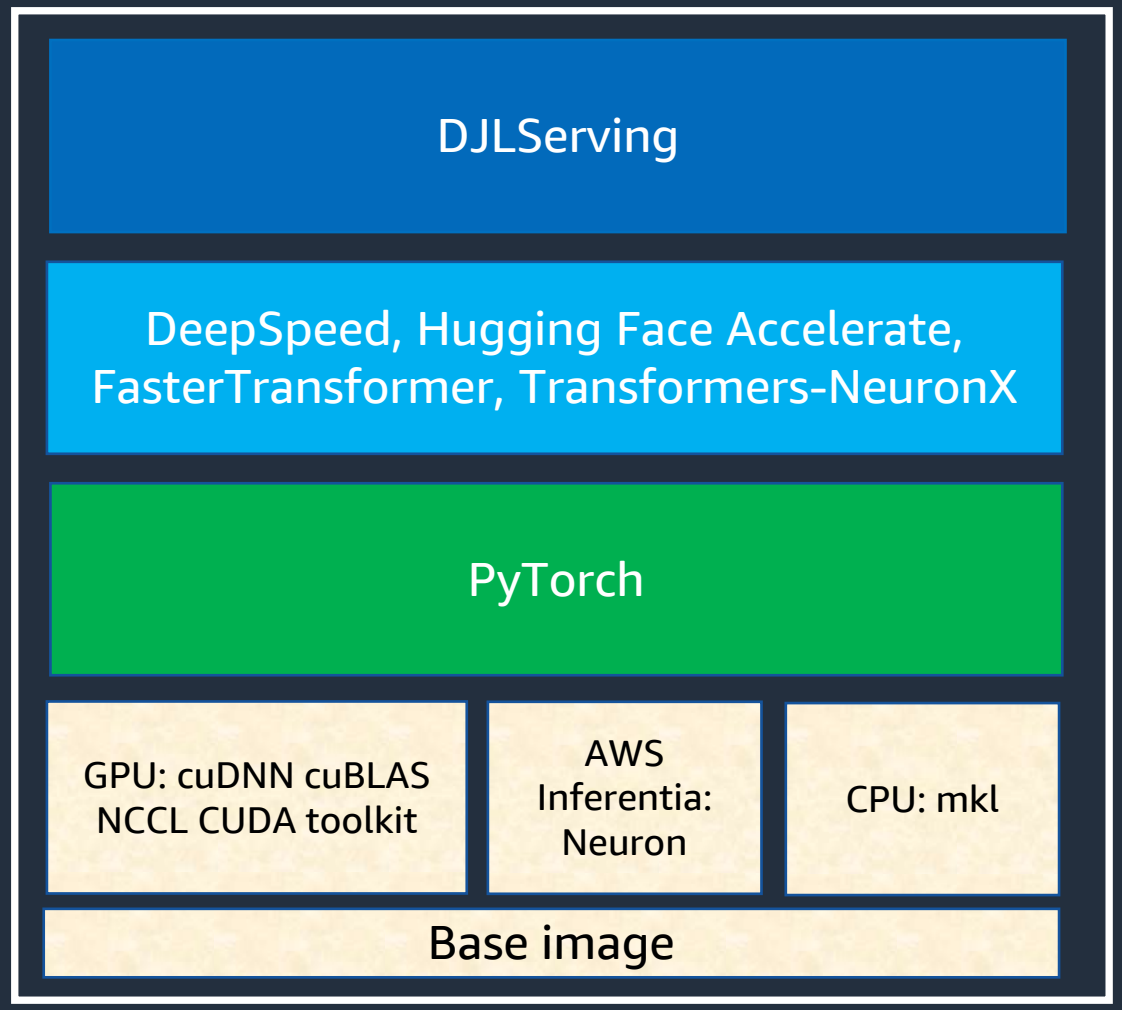
Easily parallelize models across multiple GPUs to fit models into the instance and achieve low latency



Deploy models on the most performant and cost-effective GPU-based instances or on AWS Inferentia

Pre-built foundation software stack including PyTorch, NCCL, and MPI, Transformers-NeuronX, DeepSpeed, Hugging Face Accelerate, and FasterTransformer

Large Model Inference (LMI) container on SageMaker



- Zero code setup:
  DeepSpeed, StableDiffusion and HuggingFace, FasterTransformer, Transformers-NeuronX Handlers
- Optimized environment with minimal setup (less than 8GB)
- Frameworks Support:
  HuggingFace Accelerate and DeepSpeed, FasterTransformer, Transformers-NeuronX
- Model Server:
  DJLServing - Multi-process execution with auto-scaling and UI

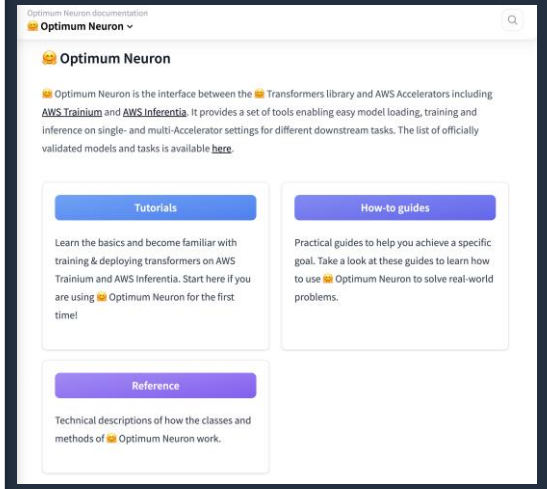
Getting started with Hugging Face Optimum Neuron

Hugging Face

+

Neuron SDK

Documentation & tutorials



Hugging Face documentation

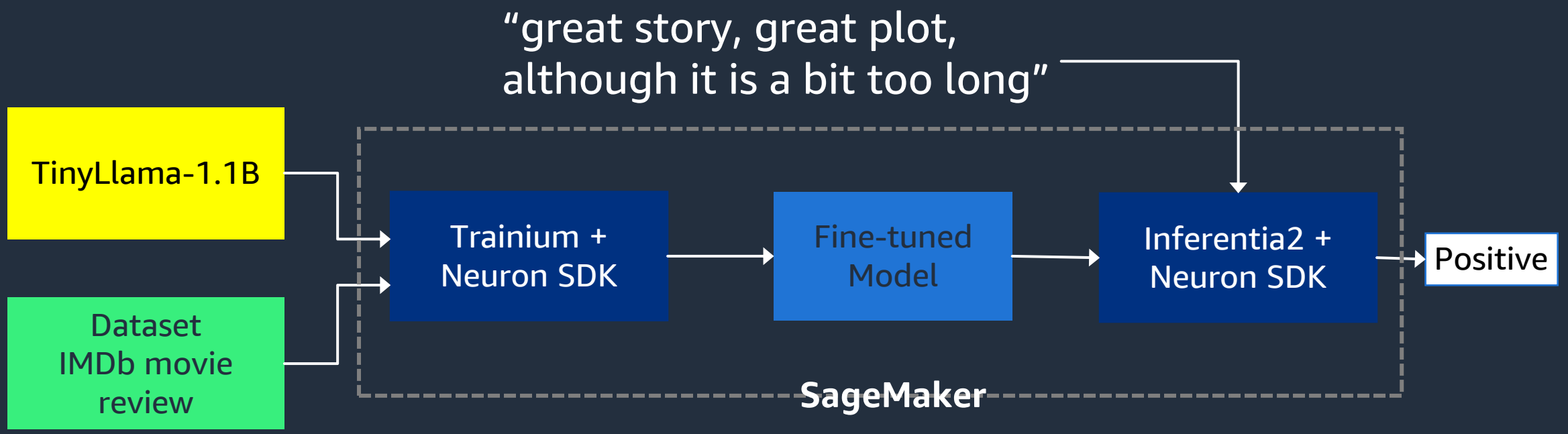
Code examples

Access GitHub

Hands-on Labs

Getting started with this workshop

Fine-tune TinyLlama-1.1B model on ml.trn1.2xlarge and then deploy the fine-tuned model on ml.inf2.xlarge

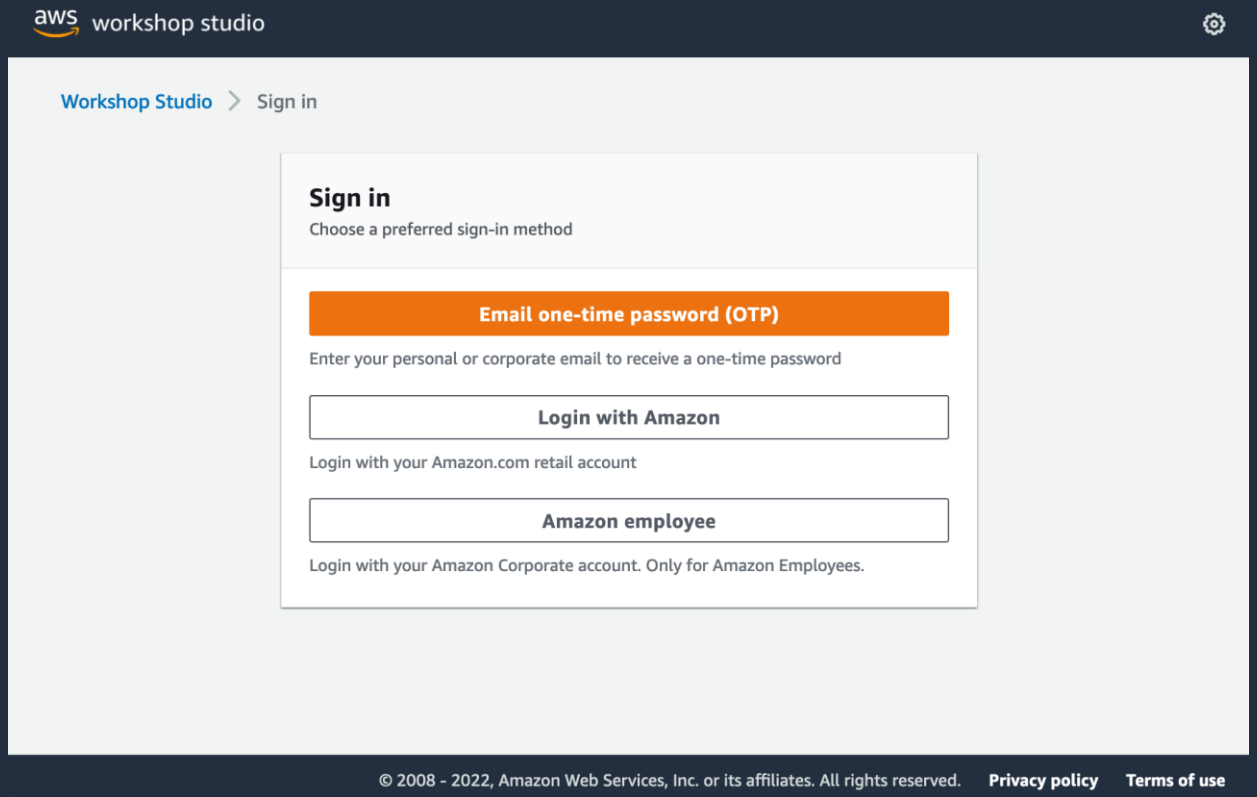


Getting started with this workshop

- As a participant, you will have access to an AWS account with any optional pre-provisioned infrastructure and AWS Identity and Access Management (IAM) policies needed to complete this workshop
- The AWS account will only be available for the duration of this workshop – You will lose access to the account thereafter
- The optional pre-provisioned infrastructure will be deployed to a specific AWS Region; check your workshop content to determine whether other Regions will be used
- Be sure to review the terms and conditions of the event. Do not upload any personal or confidential information in the account

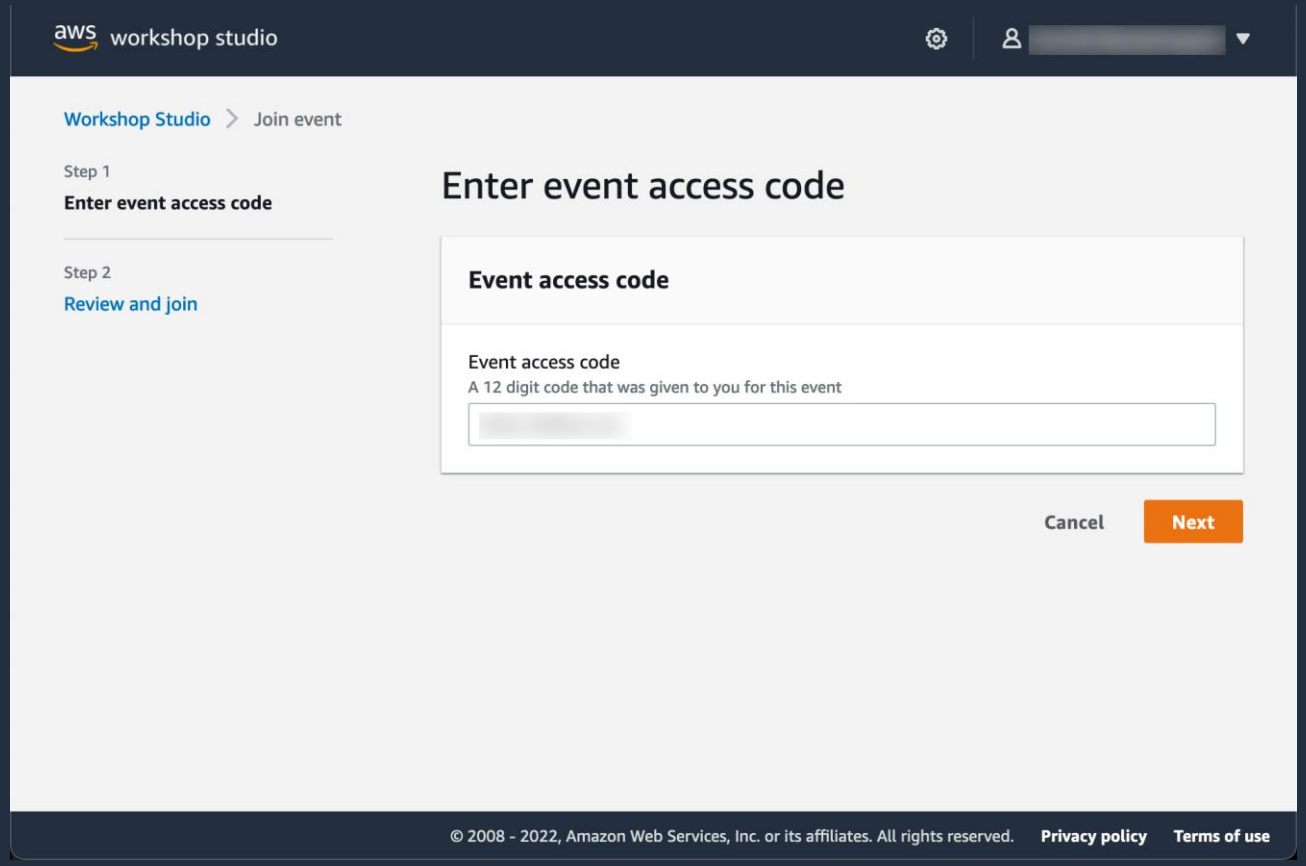
Step 1: Sign-in via your preferred method

http://tinyurl.com/bdd9kf5d

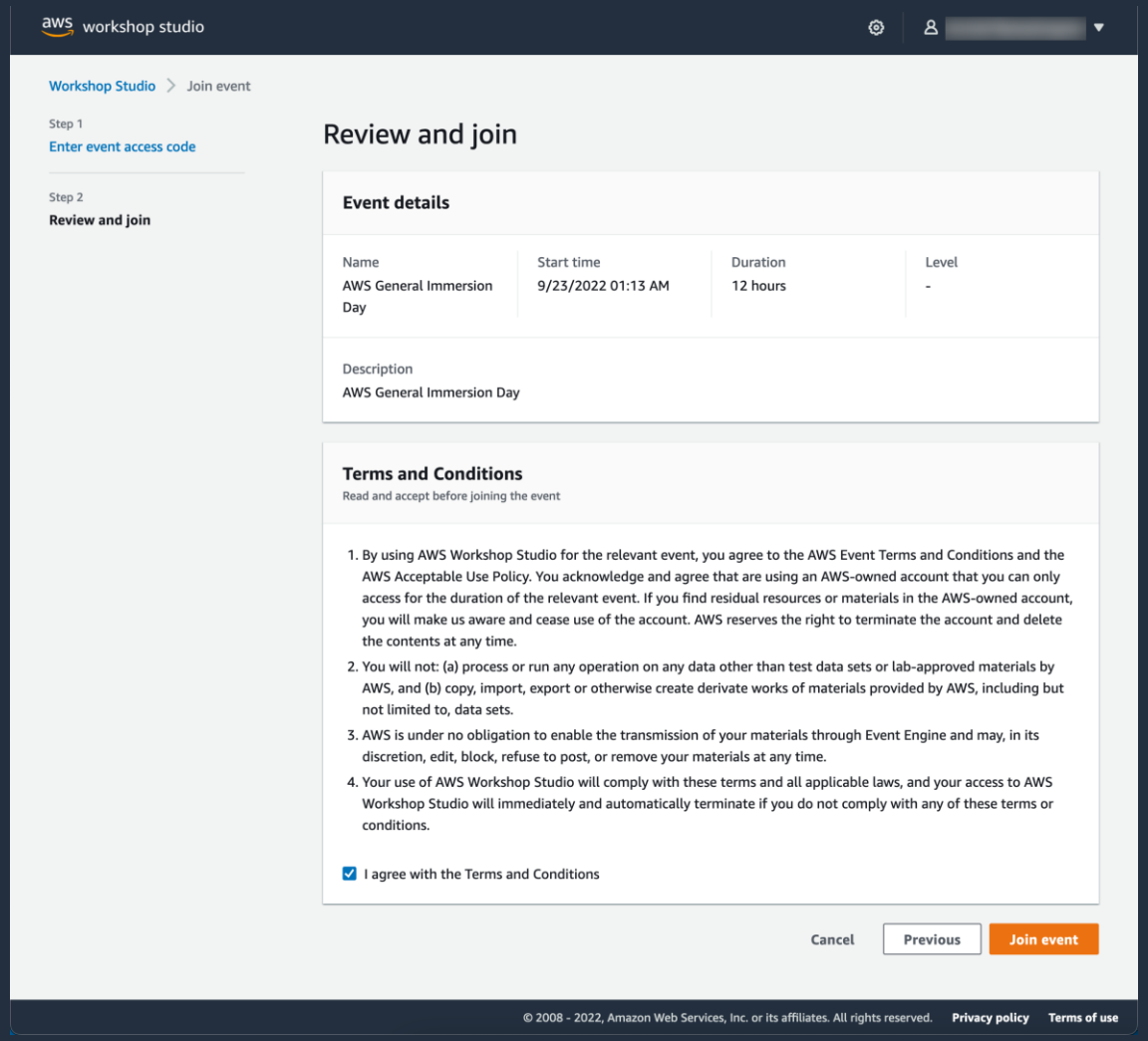


Step 2: Enter event access code

- Enter 12 digit event access code - **55fe-0d0eb7-c0**

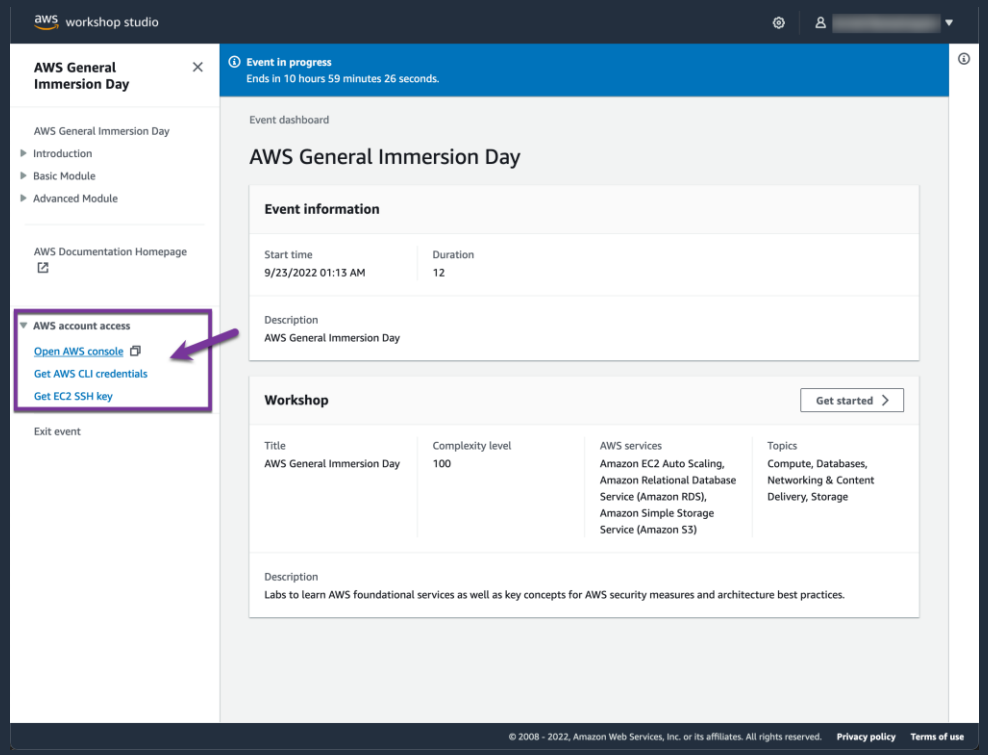


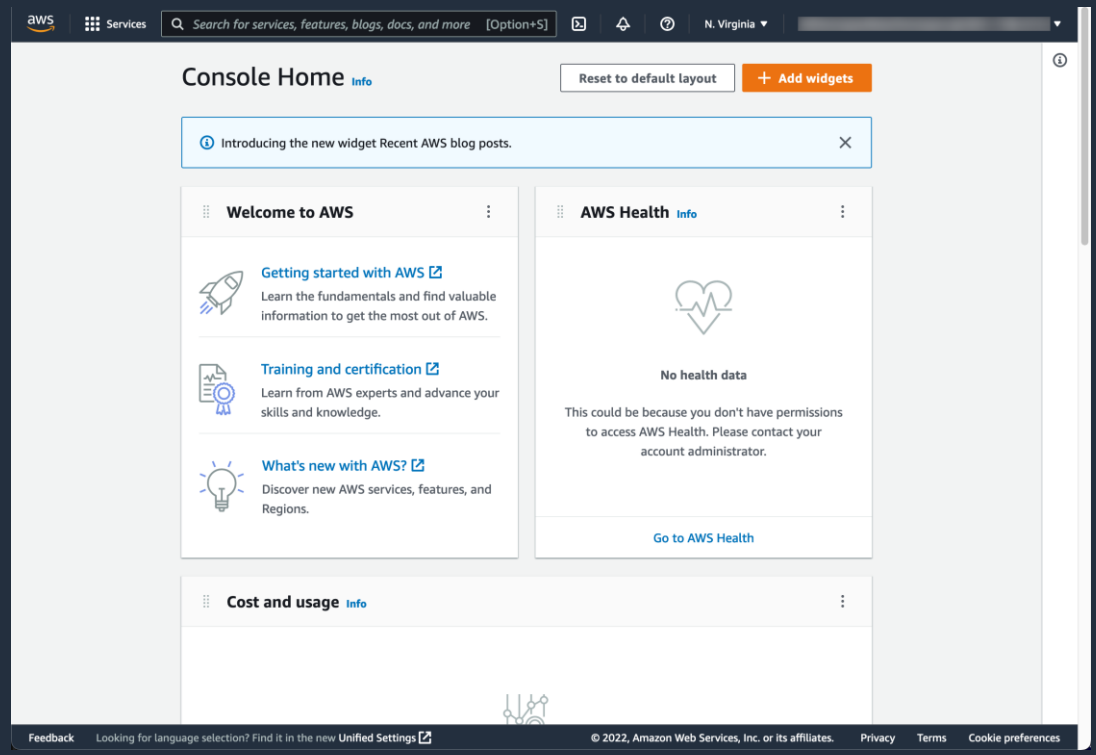
Step 3: Review terms and join event



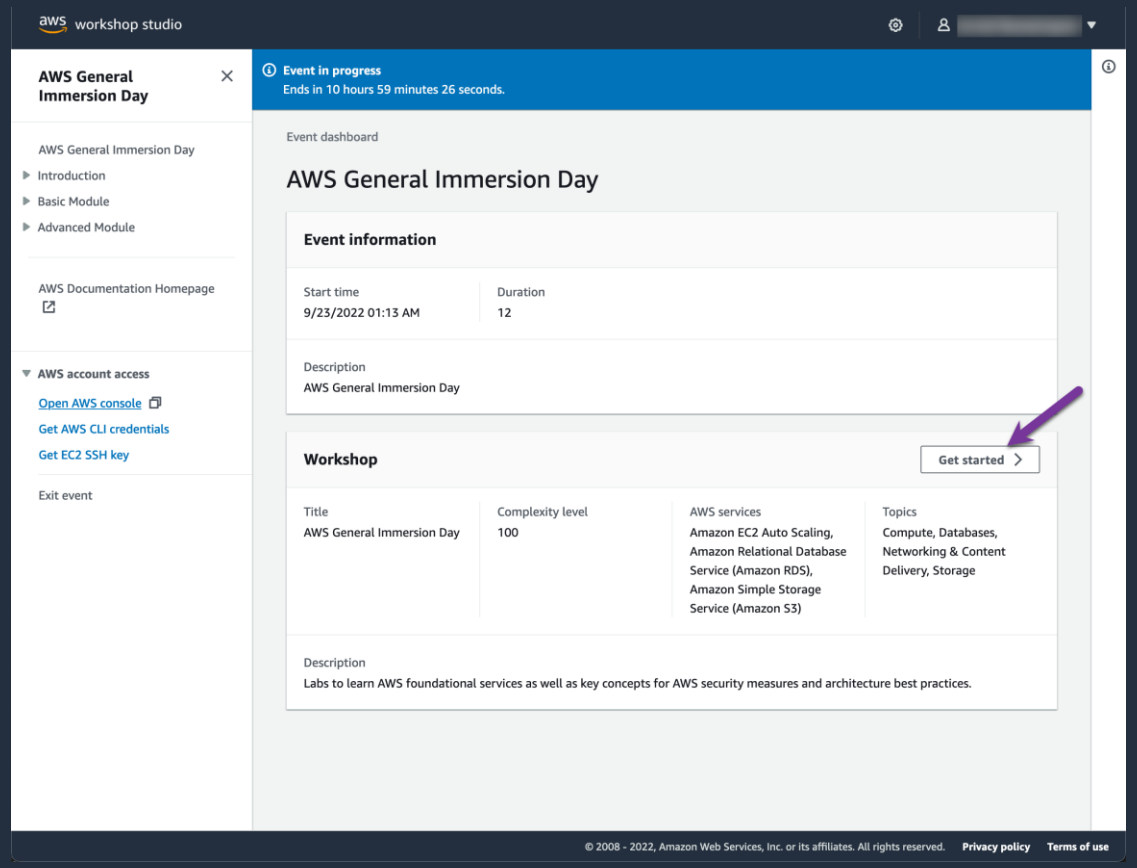
Step 4: Access AWS account

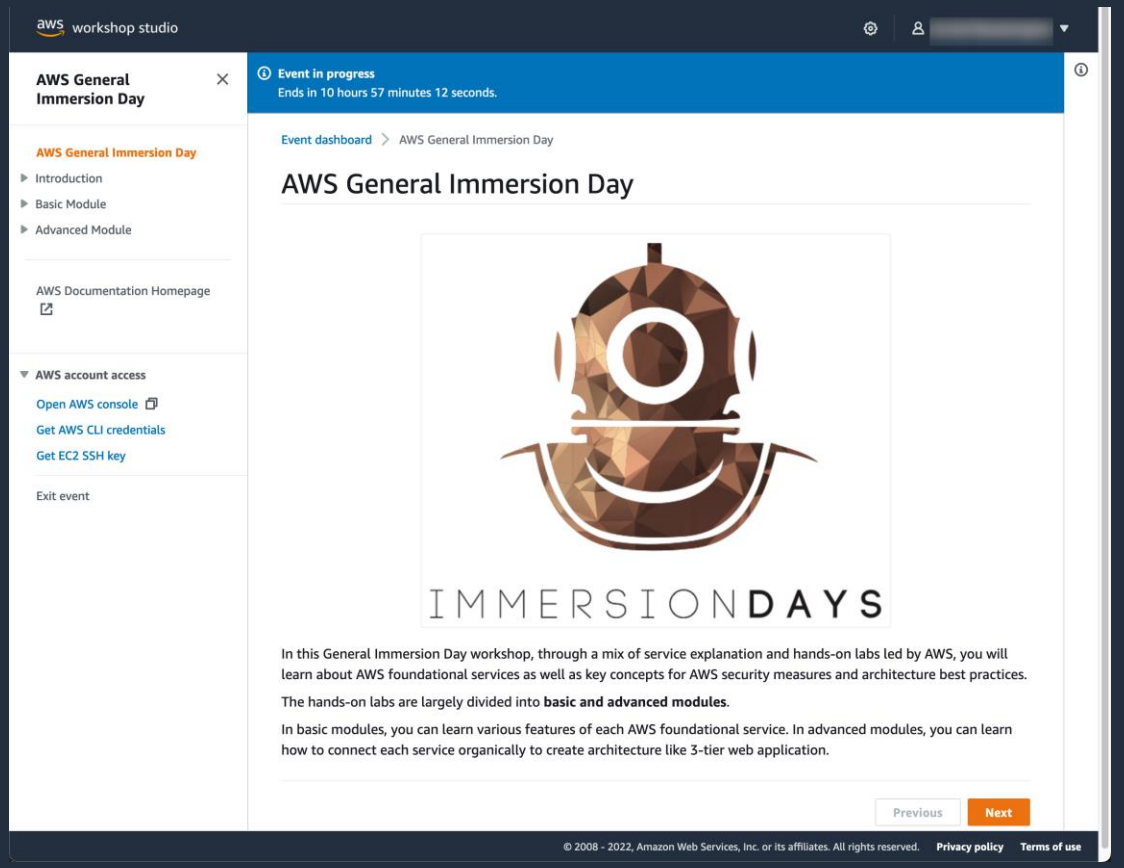
Access the AWS Console, or generate AWS CLI credentials as needed



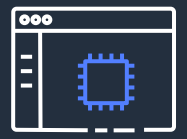


Step 5: Get started with the workshop





Resources



AWS Neuron SDK Documentation

Neuron Containers Documentation

GitHub tutorials/project

PyTorch

Getting Started with PyTorch Neuron

AWS Neuron forum and blogs

TensorFlow

Getting Started with TensorFlow Neuron

Q&A

Please complete the session survey.

Thank you!

Luca Perrozzi

Solutions Architect

AWS

Dmitri Laptev

Solutions Architect

AWS

Roy Allela

AIML Specialist Solutions Architect

AWS

Backup

Tensor parallel model

ALSO CALLED INTRA-LAYER MODEL PARALLELISM