Building Scalable End-to-End Deep Learning Pipeline in the Cloud

Rustem Feyzkhanov
Machine Learning Engineer @ Instrumental
AWS Machine Learning Hero

DEVOPS WORLD by CloudBees

Data science process

Business understanding → **Data acquisition** → **Modeling** → **Deployment** → **Customer acceptance**

Business understanding
- Define objectives
- Identify data sources

Data acquisition
- Ingest data
- Explore data
- Update data

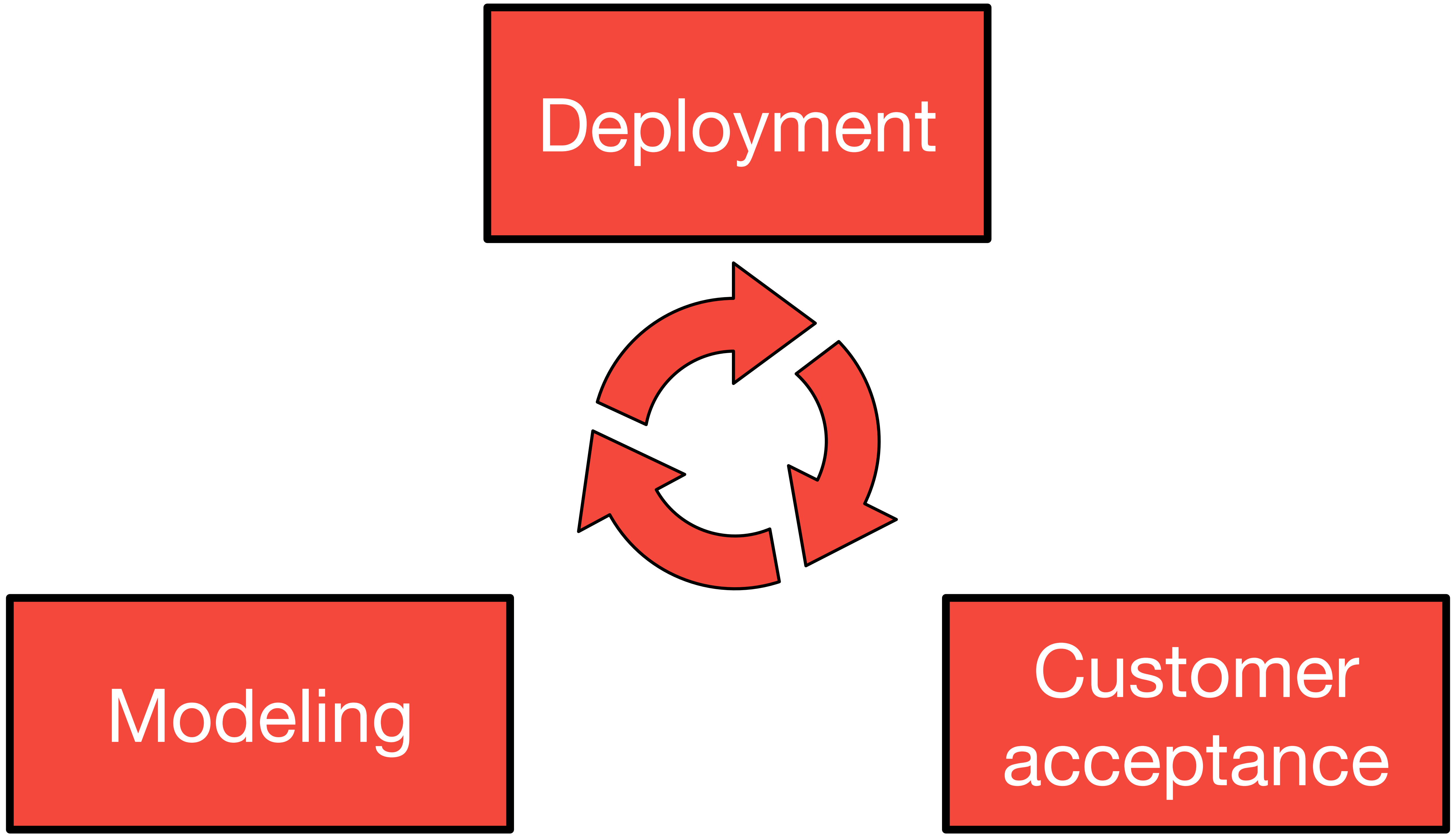
Modeling
- Feature selection
- Create model
- Train model

Deployment
- Operationalize

Customer acceptance
- Testing and validation
- Handoff
- Re-train and re-score

from https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview

Data science process



Challenges:

- starting fast
- being flexible
- integrating in current infrastructure

Production pipeline steps

- Data preprocessing
- DL/ML training
- DL/ML inference

Data preprocessing

- Challenges
  - Getting and transforming data from multiple sources
  - Combination of multiple frameworks and libraries
  - Scaling based on load
  - Combination of heavy processing, long running processing and parallel one

ML/DL training

- Challenges
  - High cost of GPU instances
  - Checking multiple sets of hyperparameters
  - Handling semi-automatic logic

ML/DL inference

- Challenges
  - Handling multiple frameworks
  - Handling model versioning
  - Scaling based on load
  - Implementing custom logic for choosing the result

Serverless approach

- Use scalable processing nodes AWS Lambda for short/parallel processing
- Use scalable container service AWS Batch for heavy and parallel processing and GPU training jobs
- Use Amazon SageMaker for GPU training jobs and distributed training
- Use scalable container service AWS Fargate for long running processing
- Use orchestrator AWS Step Functions to organize workflows

What is serverless

On premise	Iaas	CaaS	PaaS	FaaS	SaaS
Functions	Functions	Functions	Functions	Functions	Functions
Application	Application	Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime	Runtime	Runtime
Container	Container	Container	Container	Container	Container
Operating system	Operating system	Operating system	Operating system	Operating system	Operating system
Vizualization	Vizualization	Vizualization	Vizualization	Vizualization	Vizualization
Networking	Networking	Networking	Networking	Networking	Networking
Storage	Storage	Storage	Storage	Storage	Storage
Hardware	Hardware	Hardware	Hardware	Hardware	Hardware

What is serverless

On premise	Iaas	CaaS	PaaS	FaaS	SaaS
Functions	Functions	Functions	Functions	Functions	Functions
Application	Application	Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime	Runtime	Runtime
Container	Container	Container	Container	Container	Container
Operating system	Operating system	Operating system	Operating system	Operating system	Operating system
Vizualization	Vizualization	Vizualization	Vizualization	Vizualization	Vizualization
Networking	Networking	Networking	Networking	Networking	Networking
Storage	Storage	Storage	Storage	Storage	Storage
Hardware	Hardware	Hardware	Hardware	Hardware	Hardware

Container/Function-as-a-Service

On premise	Iaas	CaaS	PaaS	FaaS	SaaS
Functions	Functions	Functions	Functions	Functions	Functions
Application	Application	Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime	Runtime	Runtime
Container	Container	Container	Container	Container	Container
Operating system	Operating system	Operating system	Operating system	Operating system	Operating system
Vizualization	Vizualization	Vizualization	Vizualization	Vizualization	Vizualization
Networking	Networking	Networking	Networking	Networking	Networking
Storage	Storage	Storage	Storage	Storage	Storage
Hardware	Hardware	Hardware	Hardware	Hardware	Hardware

'Serverless' cluster

- On-demand cluster/worker to scale with your consumption
- Requires to define just code and launching configuration
- Scaling technique:
  - scales based on job queue (AWS Batch)
  - starts VM per job (AWS Fargate, Amazon SageMaker)
  - starts worker per job (AWS Lambda)

'Serverless' cluster comparison

	Lambda	SageMaker	Fargate	Batch
Type	FaaS	Pure container(s) as a service	Pure container as a service	Service which starts cluster and executes jobs
Pros	Fast startup time (~100ms) Price per 100ms Very scalable	Most instance types available Build-in dashboard Spot instances available	Customizable instances Medium startup time (~10-20s) Spot instances available	Full control VM Spot instances available
Cons	Higher price per CPU/second Timeout limit Only CPU	Medium startup time (~30-1min) Price per 1s (min 1 min)	Price per 1s (min 1 minute) Only CPU	Slow startup time (~1-4min) Price per 1s (min 1 minute)
Use cases	Short term processes	GPU long running processes	CPU long running processes	CPU/GPU medium running multiple tasks processes

CPU vs GPU for ML

- Speed of single inference/training
- Speed of batch inference
- Cost per inference/training
- Scalability

Inference cost - Inception V3

Service	Type	Inference time (s)	Cost per hour	Cost per prediction	Cost of 1M predictions	Cost per month	Lambda predictions
Lambda	3GB RAM 2vCPU	0.338	$0.18	$0.0000179	$17.9		
AWS EC2	c5a.large on demand	0.177	$0.077	$0.000003786	$3.79	$55.44	3.1M
AWS EC2	c5a.large spot	0.177	$0.032	$0.000001573	$1.57	$23.04	1.29M
AWS EC2	p2.xlarge on demand	0.057	$0.9	$0.00001425	$14.25	$648.00	36.2M
AWS EC2	p2.xlarge spot	0.057	$0.27	$0.000004275	$4.28	$194.40	10.86M
AWS EC2	inf1.large on demand	0.0095	$0.368	$0.000000971	$0.97	$264.96	14.8M
AWS EC2	inf1.large spot	0.0095	$0.1104	$0.000000291	$0.29	$79.49	4.44M

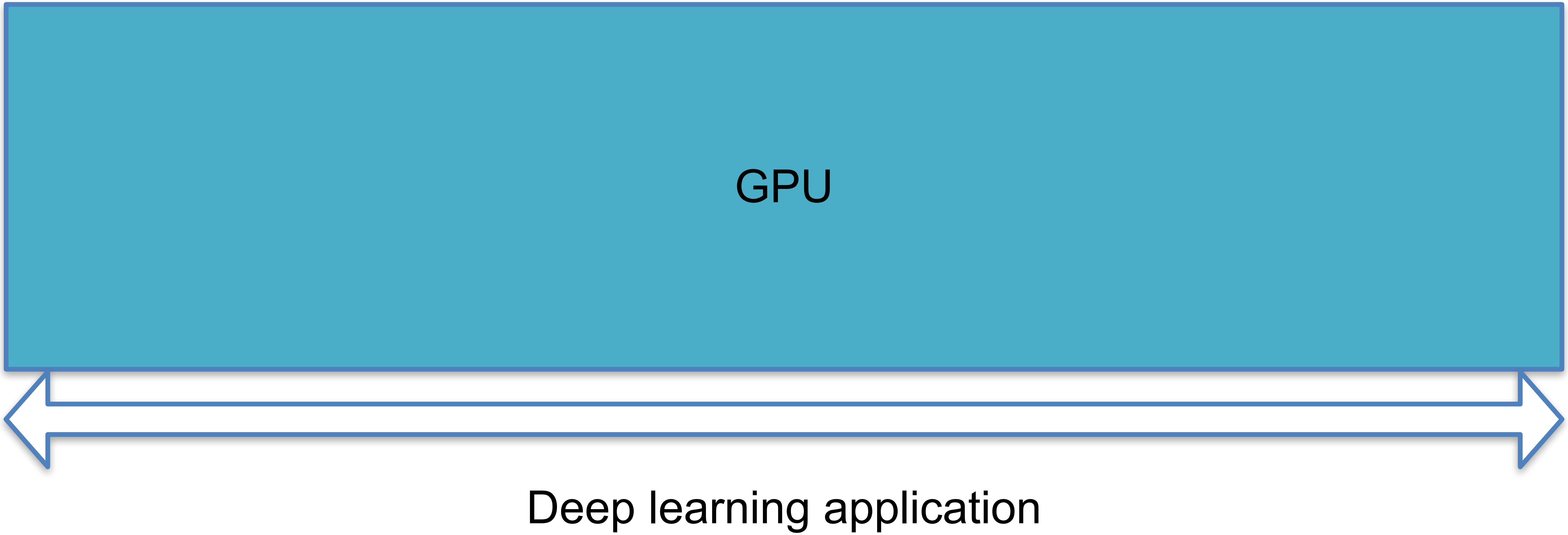
Price comparison - CPU

- C5 Large Instance - 2 vCPU 4GB RAM
  - AWS Lambda
    - 3GB RAM x 0.00001667 x 3600 **= 0.18$ per hour**
  - AWS Fargate
    - 4GB RAM x 0.0044 + 2 vCPU x 0.0404 **= 0.098$ per hour**
  - AWS Batch
    - C5 Large On Demand **= 0.085$ per hour**
    - C5 Large Spot **= 0.033$ per hour**

Price comparison - GPU

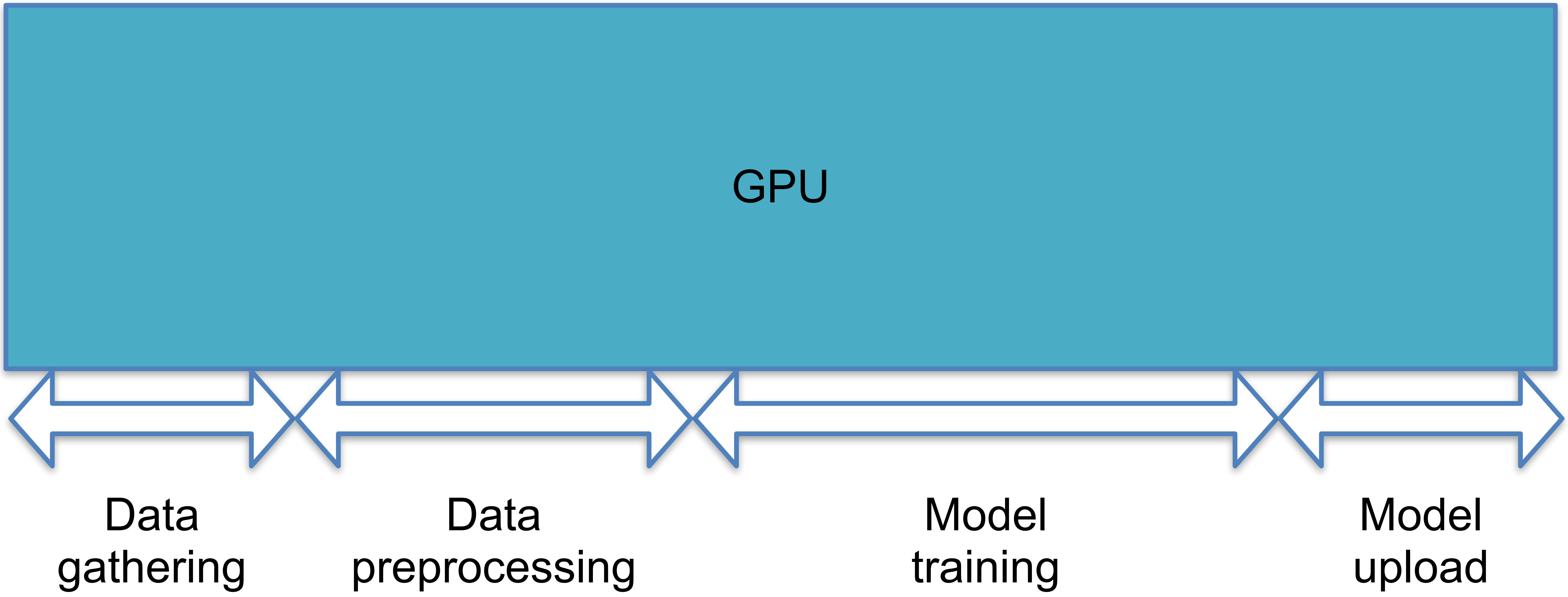
- P2 Xlarge Instance - 1 NVIDIA K80 GPU, 4 vCPU
  - Amazon SageMaker
    - P2 Xlarge ML instance = **1.26$ per hour**
    - P2 Xlarge ML instance Spot = **0.37$ per hour**
  - AWS Batch
    - P2 Xlarge On Demand = **0.90$ per hour**
    - P2 Xlarge Reserved = **0.42$ per hour**
    - P2 Xlarge Spot = **0.27$ per hour**

Modular approach

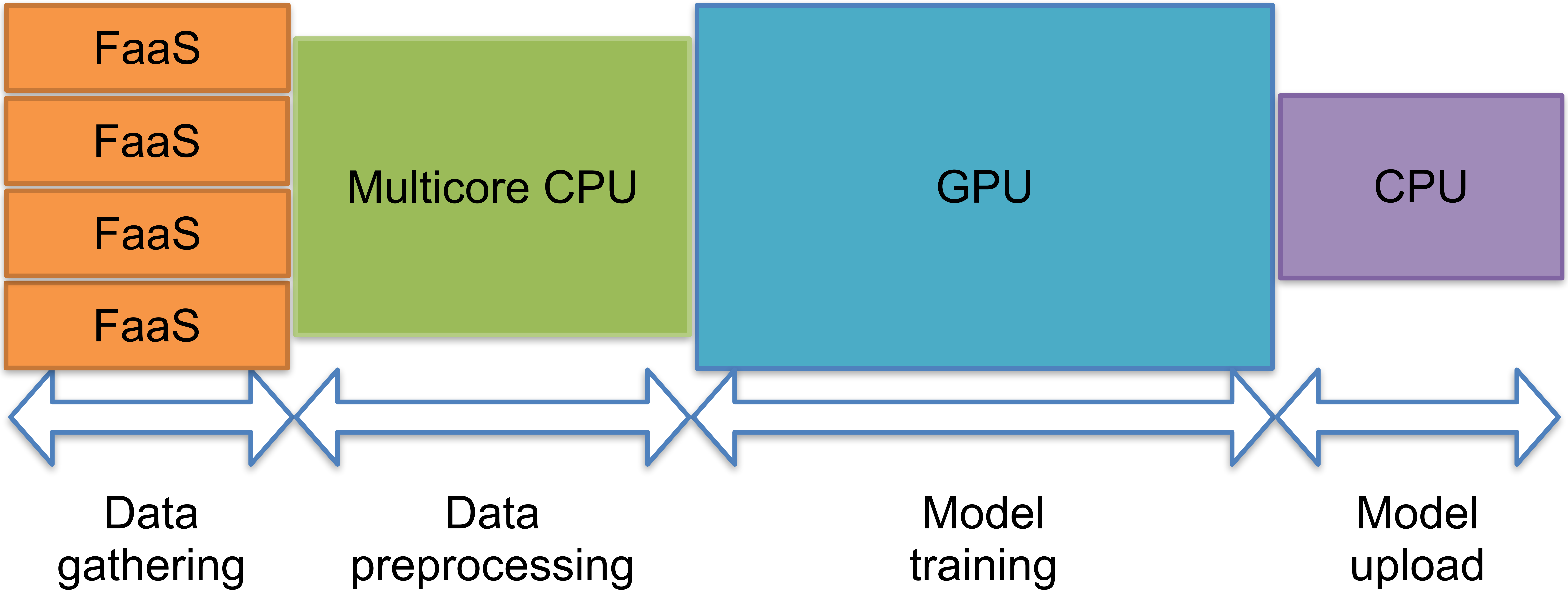


DEVOPSWORLD by CloudBees

Modular approach



Modular approach



Platform-as-a-Service

On premise	Iaas	CaaS	PaaS	FaaS	SaaS
Functions	Functions	Functions	Functions	Functions	Functions
Application	Application	Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime	Runtime	Runtime
Container	Container	Container	Container	Container	Container
Operating system	Operating system	Operating system	Operating system	Operating system	Operating system
Vizualization	Vizualization	Vizualization	Vizualization	Vizualization	Vizualization
Networking	Networking	Networking	Networking	Networking	Networking
Storage	Storage	Storage	Storage	Storage	Storage
Hardware	Hardware	Hardware	Hardware	Hardware	Hardware

Microservice connectors

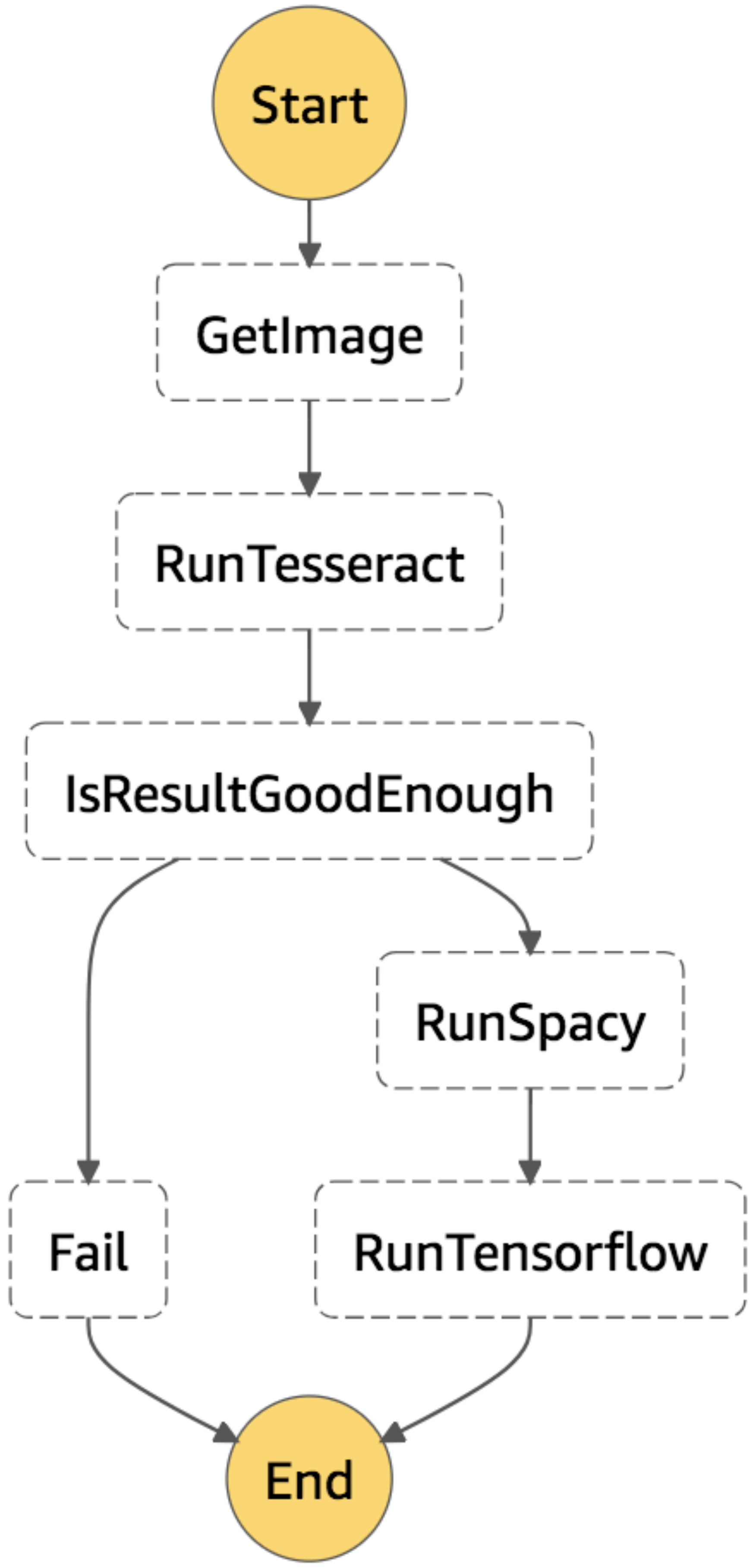
Rest API	Event queue	Orchestrator
Synchronous process	Asynchronous process	Asynchronous process
Short-term process	Long-term process	Long-term process
Simple intermediate logic	Simple intermediate logic	Complex intermediate logic
Doesn’t trace the whole process	Doesn’t trace the whole process	Traces the process
Cheap	Cheap	Expensive

Cloud native orchestrators

- Native support for FaaS and CaaS
- Central monitoring
- Central logging and tracing
- On-demand scaling

Orchestrators for hybrid architecture

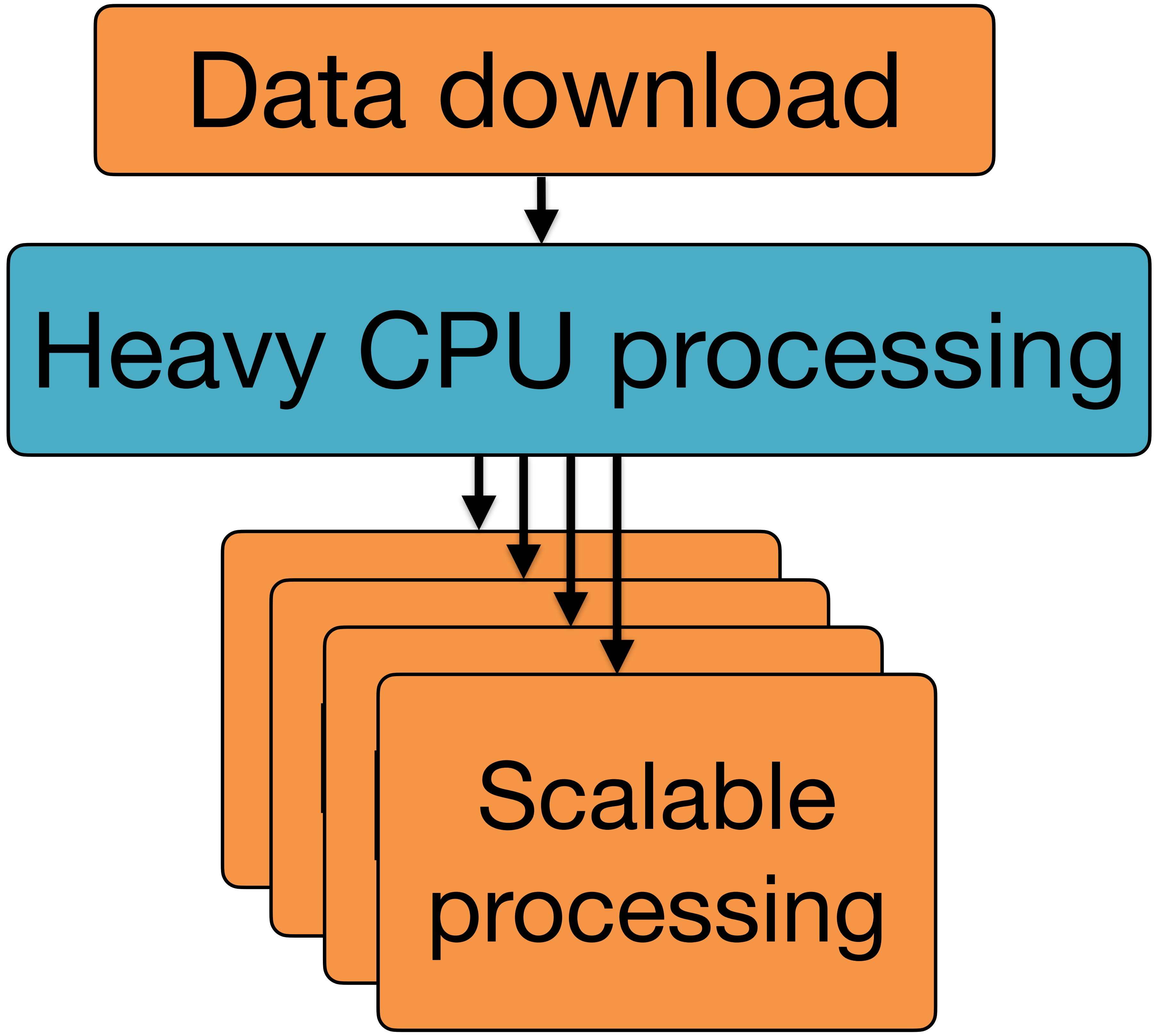
- Graph-based description
- Processing nodes: FaaS or Clusters
  - Task state and waiting for the node
  - Invocation of processing node
- Logic for error handling
- Parallel execution
- Branching and loops
- Scheduler



Data pipeline

- Challenges
  - Getting and transforming data from multiple sources
  - Combination of multiple frameworks and libraries
  - Scaling based on load
  - Combination of heavy processing, long running processing and parallel one

Data pipeline



- Modular approach
- Parallel data download and parsing
- FaaS for parallel processing
- Cluster for heavy processing

How do you know if this is for you

- You have peak loads and want to scale automatically
- You have custom logic (scheduler, error handling, etc) in business logic
- You want to make customizable pipeline with multiple frameworks

Repositories to check

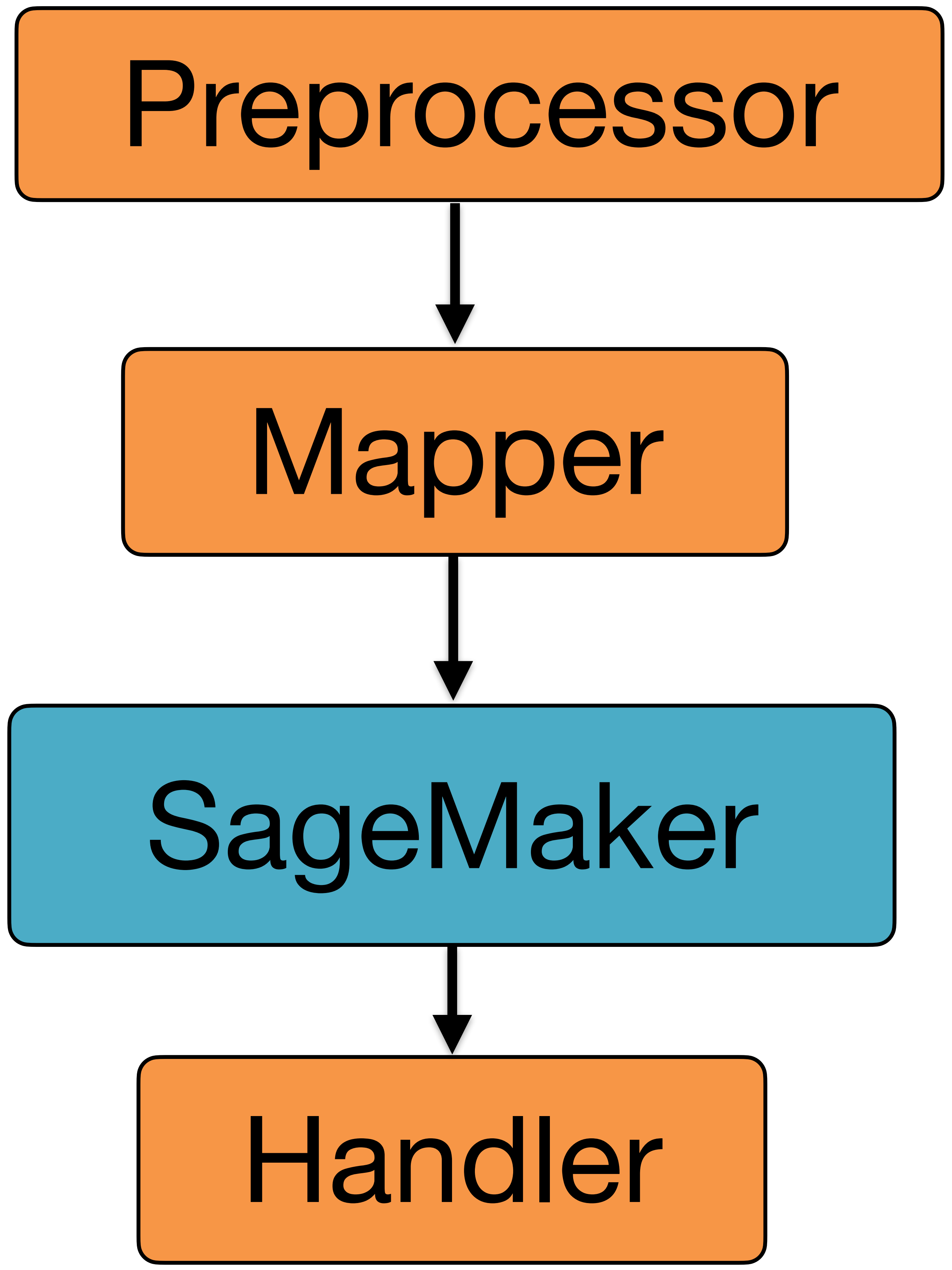
https://github.com/ryfeus/stepfunctions2processing

- Serverless configuration files which allows to deploy:
  - AWS Step Functions
  - AWS Lambda
  - AWS Batch + AWS Fargate

ML/DL training pipeline

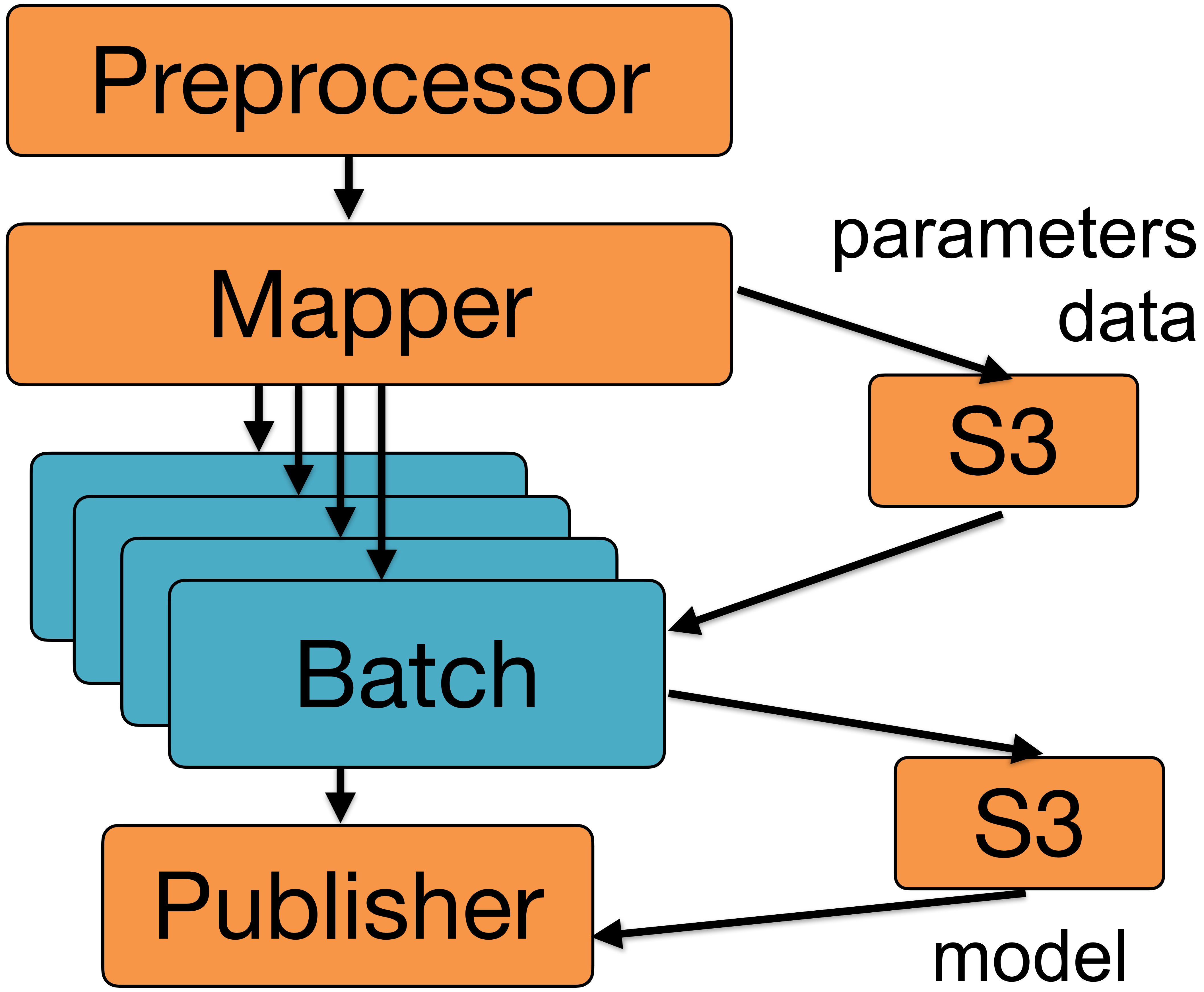
- Challenges
  - High cost of GPU instances
  - Checking multiple sets of hyperparameters
  - Handling semi-automatic logic

Amazon SageMaker



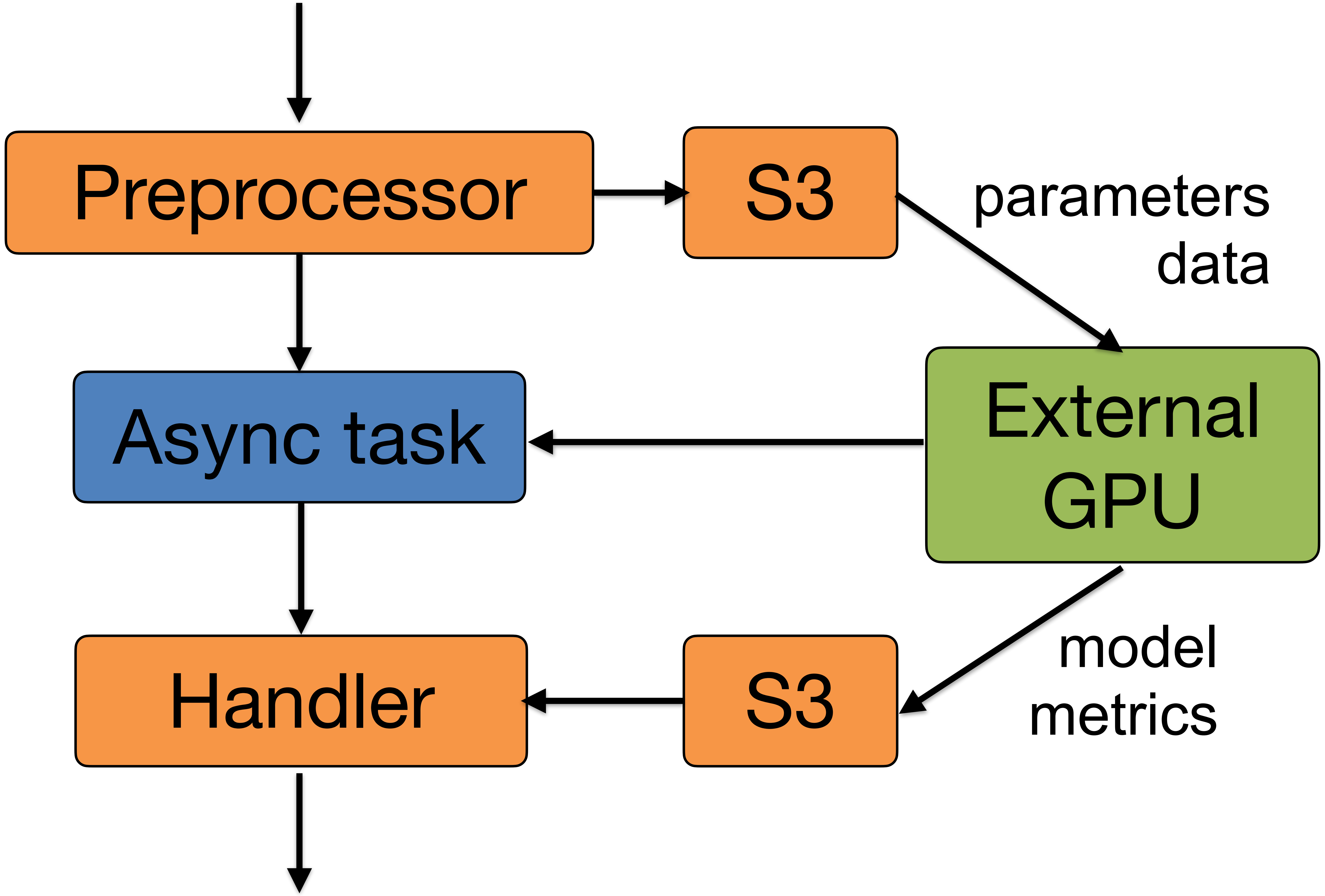
- Automatic handling of hyper parameters and metrics
- Automatic handling of model and input data
- Automatic hyperparameters optimization
- Handling error on each branch
- Distributed training

AWS Batch



- Parallel training on multiple sets of hyper parameters
- Central gathering of the results
- Handling error on each branch
- Capability for feedback loop
- Test after training

External infrastructure



- Integrating production cloud environment with on-premise infrastructure
- Preparing data and providing access
- Handle publishing completed model

Repositories to check

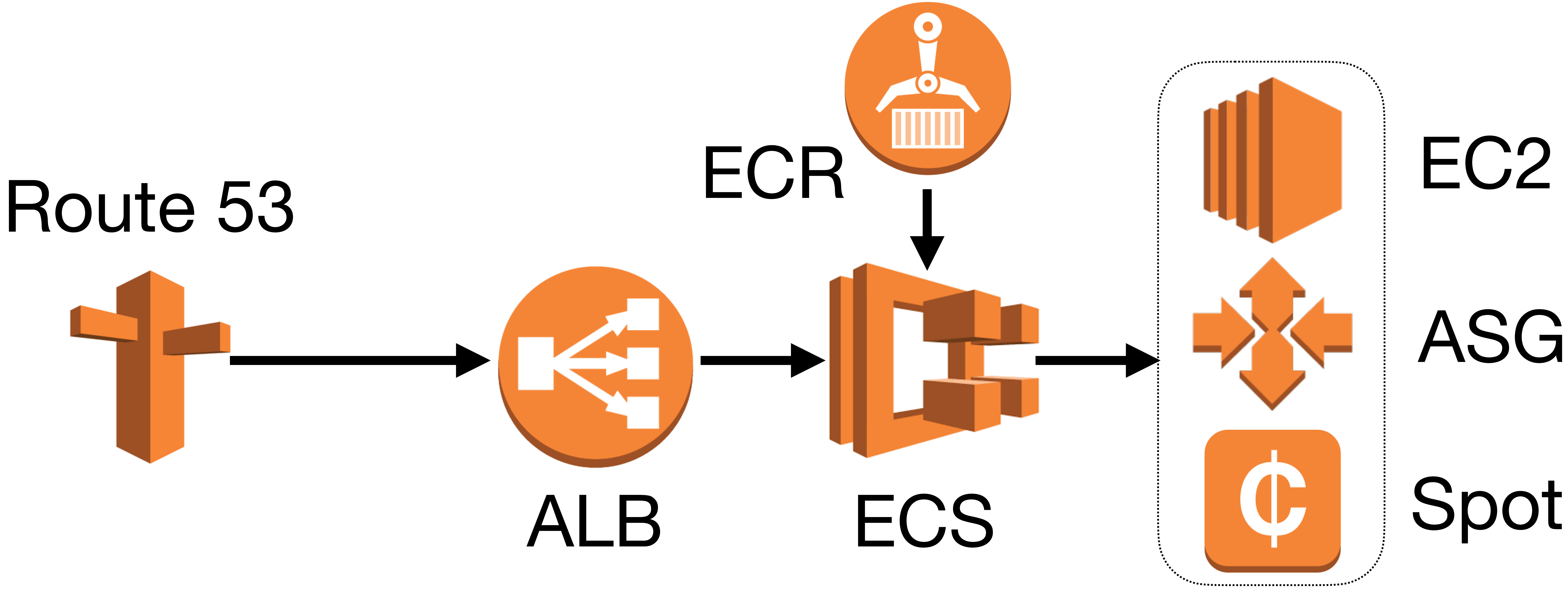
https://github.com/ryfeus/stepfunctions2processing

- Serverless configuration files which allows to deploy:
  - AWS Step Functions
  - AWS Lambda
  - AWS Batch, Amazon SageMaker

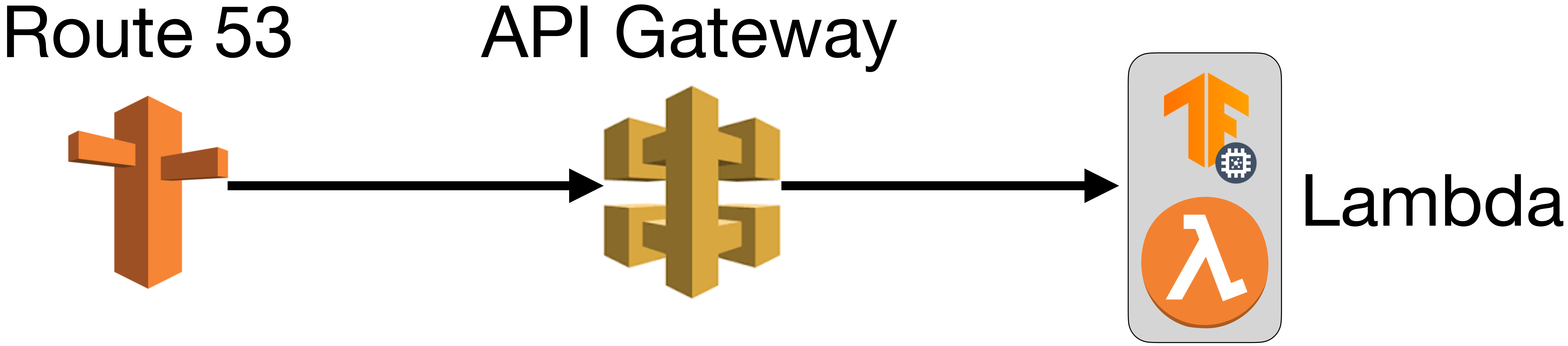
ML/DL inference pipeline

- Challenges
  - Handling multiple frameworks
  - Handling model versioning
  - Scaling based on load
  - Implementing custom logic for choosing the result

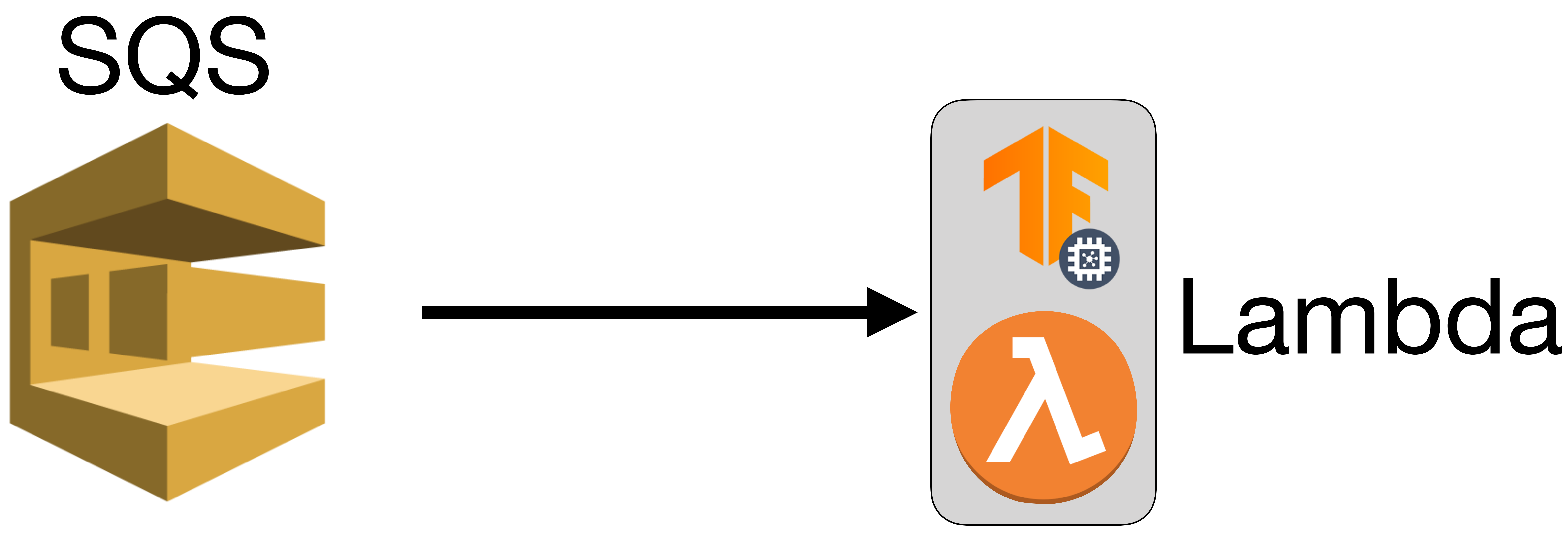
Usual AWS architecture for inference



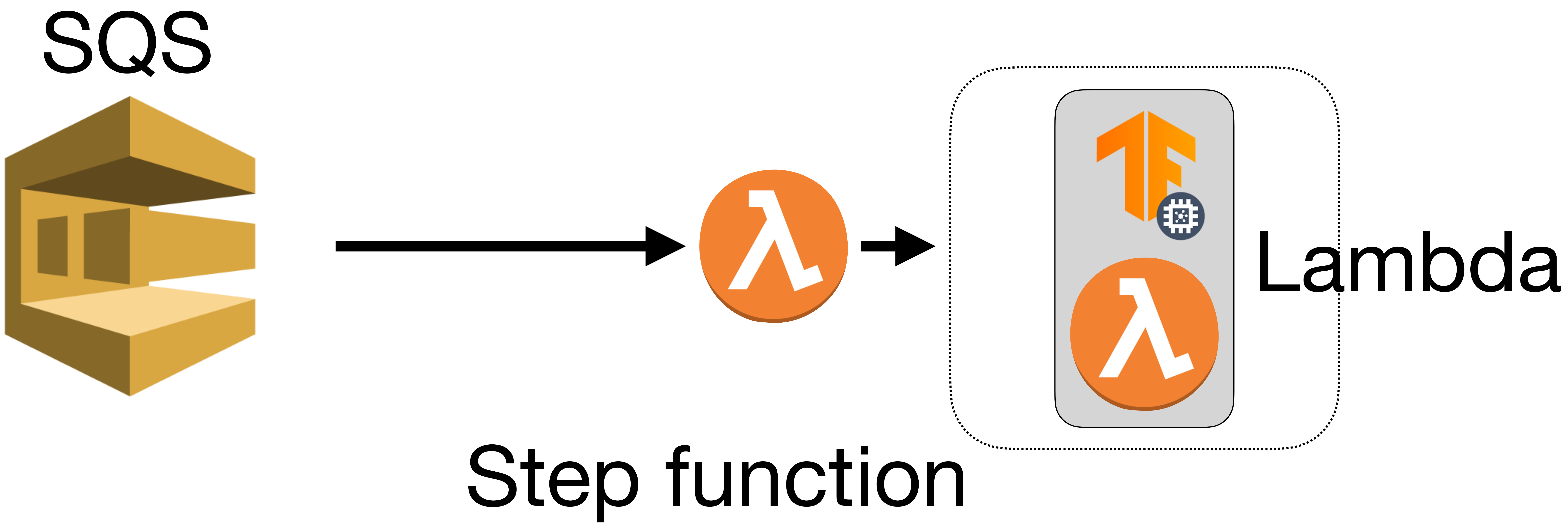
Architecture using Lambda



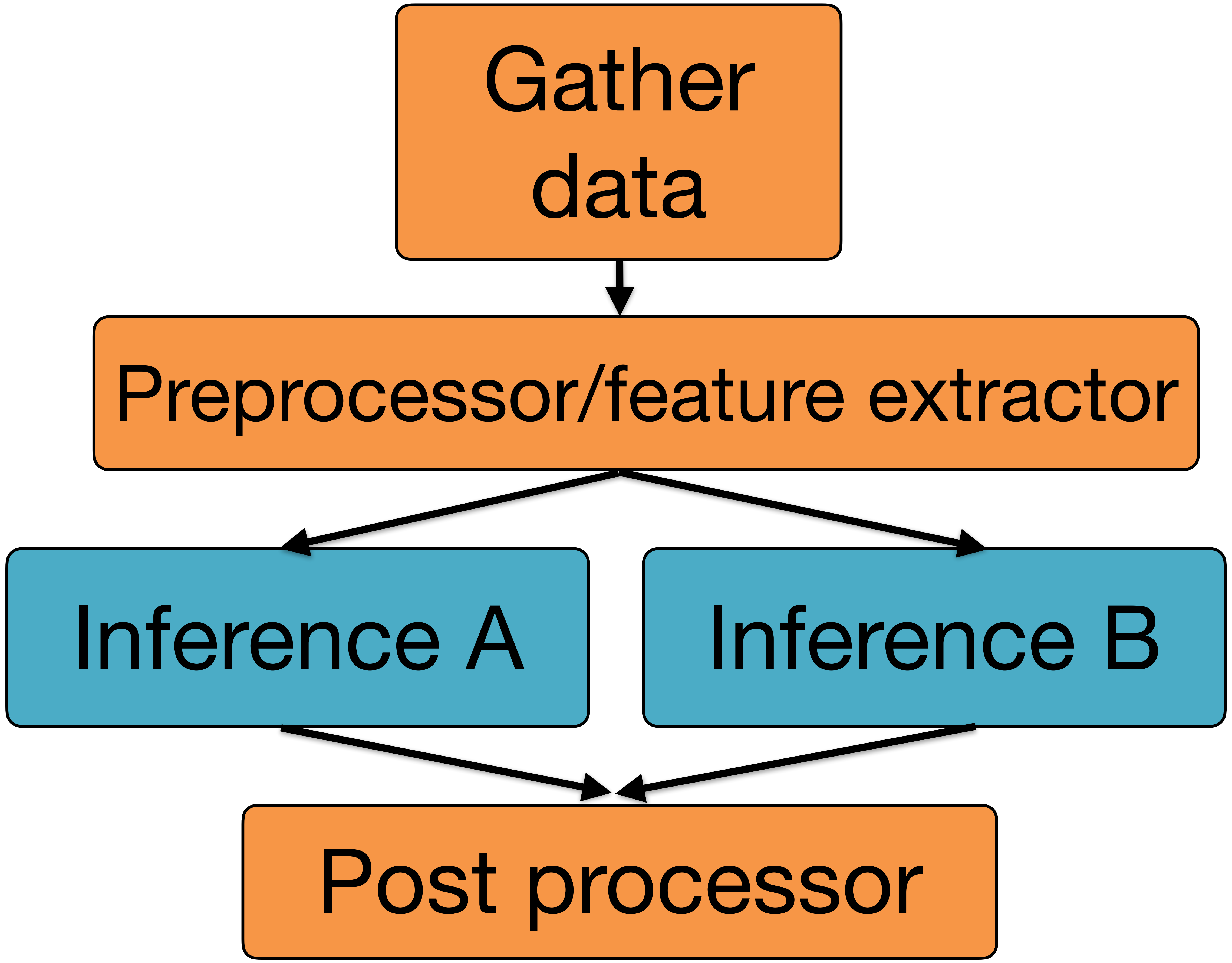
Architecture using Lambda



Architecture using Lambda



ML/DL inference pipeline



- A/B testing to test performance of multiple models - either in parallel or separately
- Scalable inference which allows to run batches in parallel
- Allows modular approach (multiple frameworks)

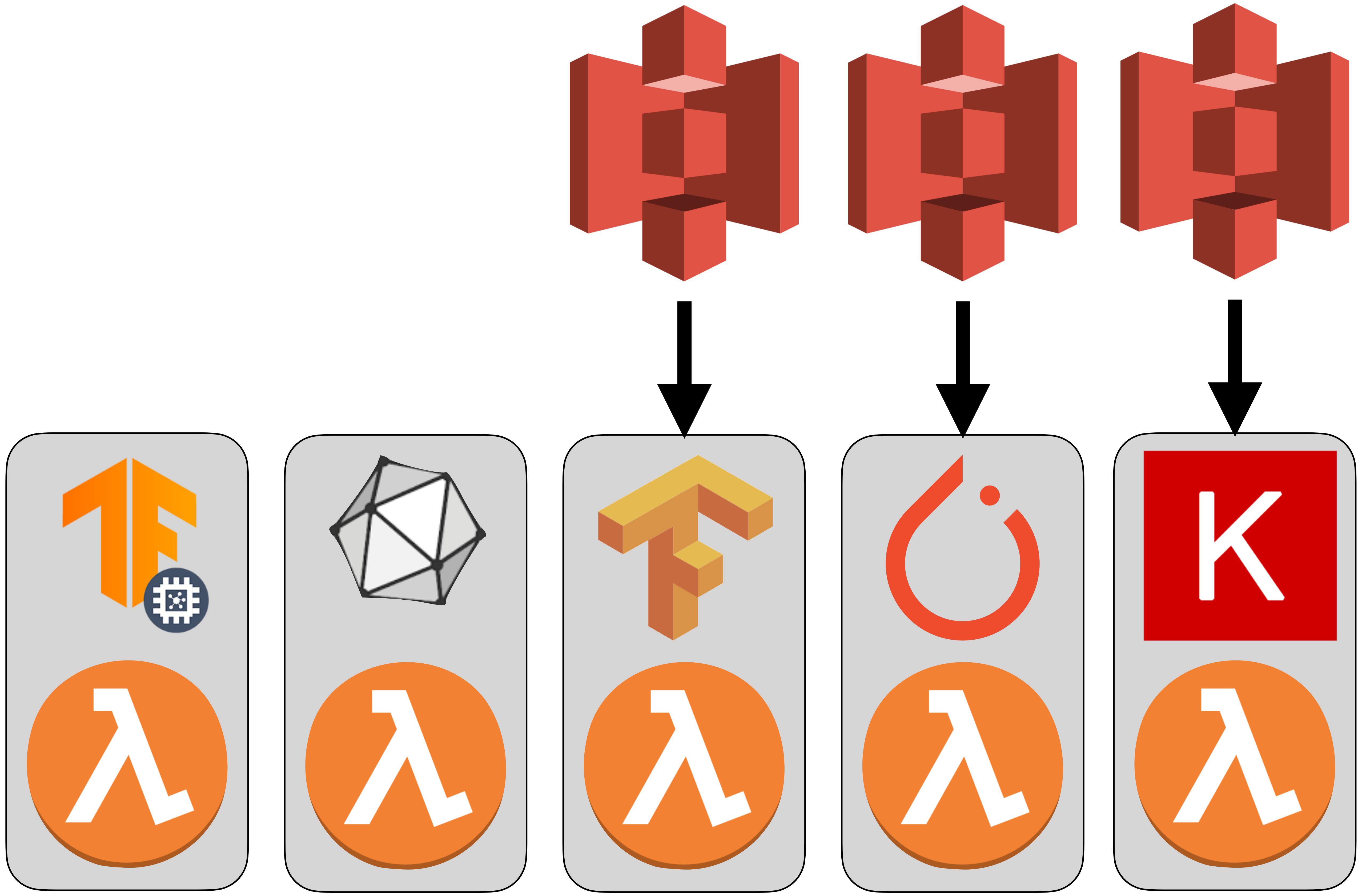
How to import models

Import from S3:

- Keras - h5 files
- TensorFlow - pb/ckpt files
- PyTorch - path files

Models in package:

- TensorFlow - TFlite export
- PyTorch - ONNX export



Inference cost - Inception V3

Framework	RAM	Cold invocation	Warm invocation	Cold inv per 1$	Warm inv per 1$
Tensorflow	3 GB	2.9s	0.6s	6.8K	32K
Tensorflow	1.5 GB	3.6s	1.1s	10.1K	35K
TFLite	3 GB	8.5s	0.4s	2.3K	47K
TFLite	1.5 GB	8.8s	0.7s	4.5K	54K

Lifehacks for serverless inference

- Store model in memory for warm invocations
- Use AWS EFS for storing the model
- Store part of the model with the libraries
- Download model in parallel from storage
- Separate layers on multiple lambdas and chain them
- Batch the workload
- Balance RAM/Timeout to optimize your costs

How do you know if this is for you

- You want to deploy your model for pet project
- You want to make s simple MVP for your startup/project
- You have simple model and this architecture will reduce cost
- You have peak loads and it is hard to manage clusters

How do you know if this is NOT for you

- You want to have real time response
- Your model requires a lot of data
- Your model requires a lot of processing power
- You want to handle large number of requests (>10M per month)

Repositories to check

https://github.com/ryfeus/lambda-packs https://github.com/ryfeus/gcf-packs

- Packages for AWS Lambda and Google Cloud Functions including:
  - Tensorflow (including 2.0), PyTorch - Deep Learning
  - Scikit Learn, LightGBM, H2O - Machine Learning
  - Scikit Image, Scipy, OpenCV, Tesseract - Image processing
  - Spacy - Natural Language Processing

Summary

- Cloud native orchestrators are convenient for constructing scalable end-to-end deep learning pipelines
- There are multiple services at your disposal for constructing deep learning workflow and it depends on your context
- You can deploy this kind of workflows pretty easily even for research projects

Thank you!

Packages for AWS Lambda and Google Cloud Functions

https://github.com/ryfeus/lambda-packs

https://github.com/ryfeus/gcf-packs

Infrastructure configuration files for AWS Step Functions, AWS Batch, AWS Fargate, Amazon Sagemaker

https://github.com/ryfeus/stepfunctions2processing

Link to my website: https://ryfeus.io