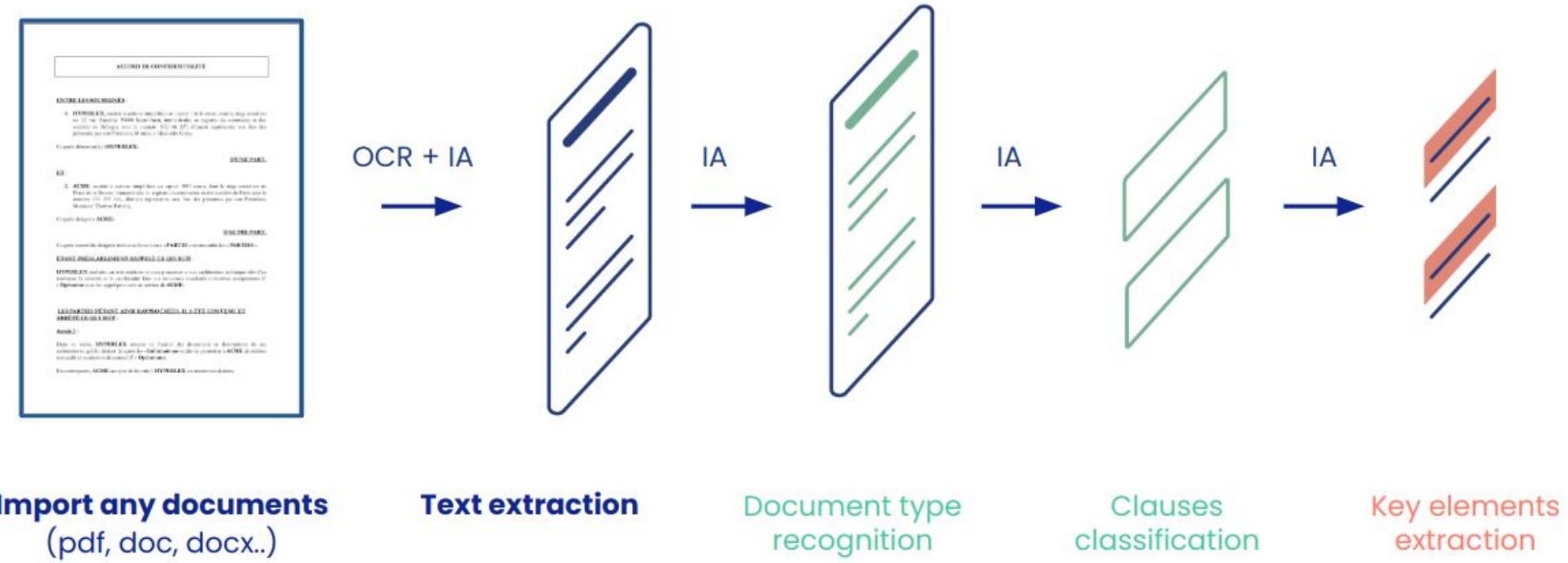
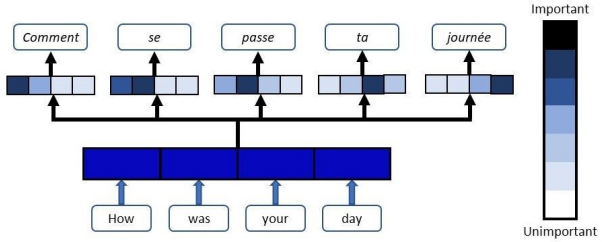
Accélérer le déploiement de modèles de machine learning

DiliTrust

ML chez DiliTrust



Evolution des architectures de modèles de NLP



Bard

MistralAI

AI2

UltraChat

BigScience

Meta AI

2013

- Approches n-gram
- Word2vec
- RNNs

2017

- LSTM
- Encoder / Decoder

2018 …

- GPT-1
- BERT

2020

- GPT-3
- Hugging Face ecosystem
- Llama
- Bloom
- Vicuna
- Mistral
- Olmo

Mais aussi

- RLHF
- LoRA
- DPO
- ML in Rust

2024

Evolution des architectures de modèles de NLP

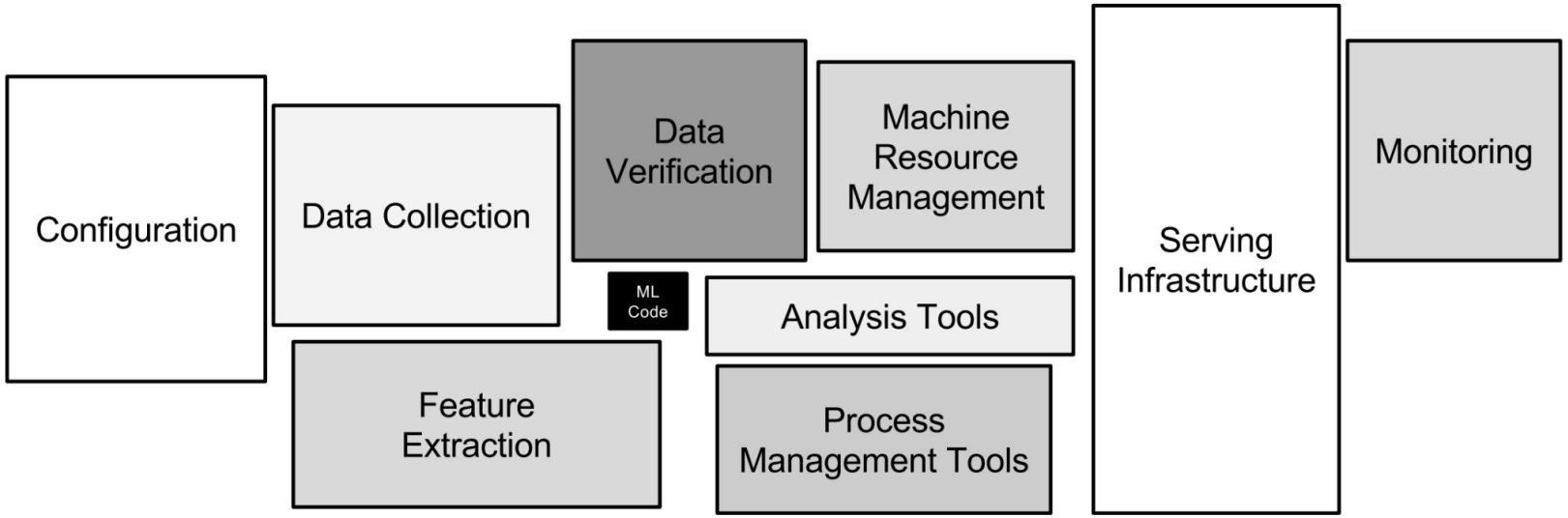
Mixture of Expert

OpenChat update

MLX,
efficient inference on iPhones

Efficient inference on
distributed serving

Maintenir une modèle de NLP embarqué dans une application



Recherche — Production

- Code volatile
- Scripts, jobs automatisés
- Latences fortes

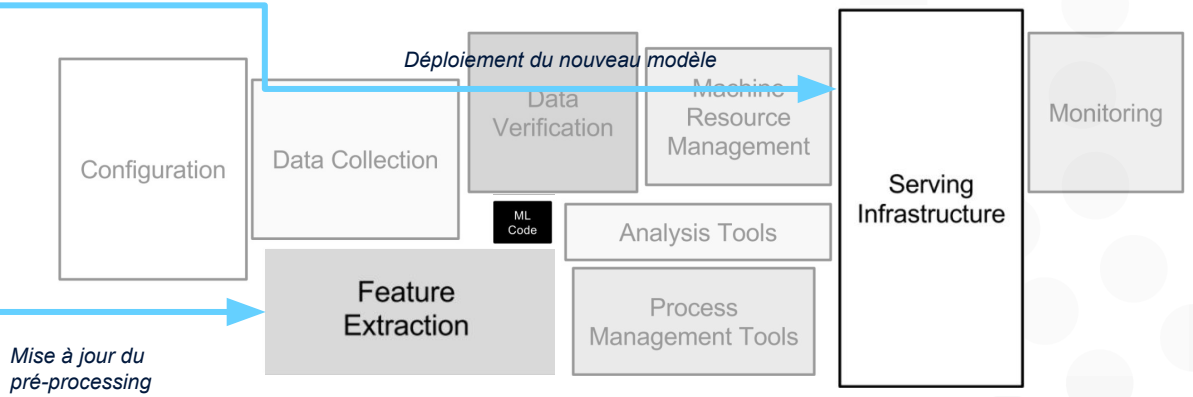
- Stabilité du code
- Applications scalables, relativement urbanisées dans le SI
- Latences faibles

En pratique, qu’est ce que cela implique ?

Faire évoluer le périmètre fonctionnel

Ajouter une langue au modèle nécessite une nouvelle méthode de pré-processing : détecter l’alphabet avant de passer à l’inférence :

- Déployer la détection de l’alphabet
- Mettre à jour le modèle



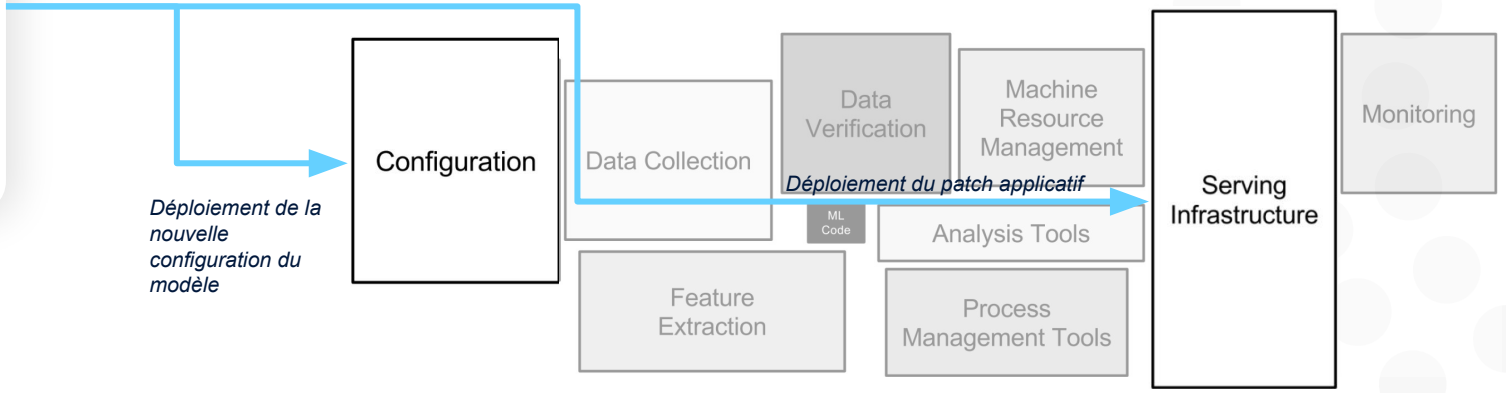
DILITRUST

En pratique, qu’est ce que cela implique ?

Patcher un bug

Bug détecté en production, nécessitant la mise à jour de la configuration du modèle ainsi que du code servant à préparer le document pour l’inférence

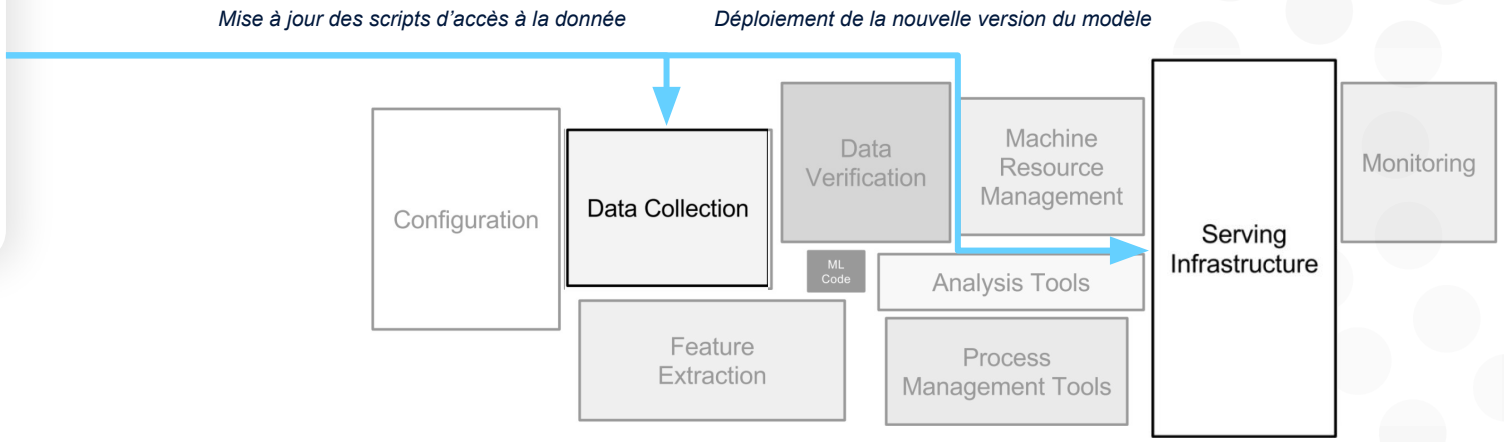
- Déploiement de la configuration du modèle
- Livraison du patch applicatif



En pratique, qu’est ce que cela implique ?

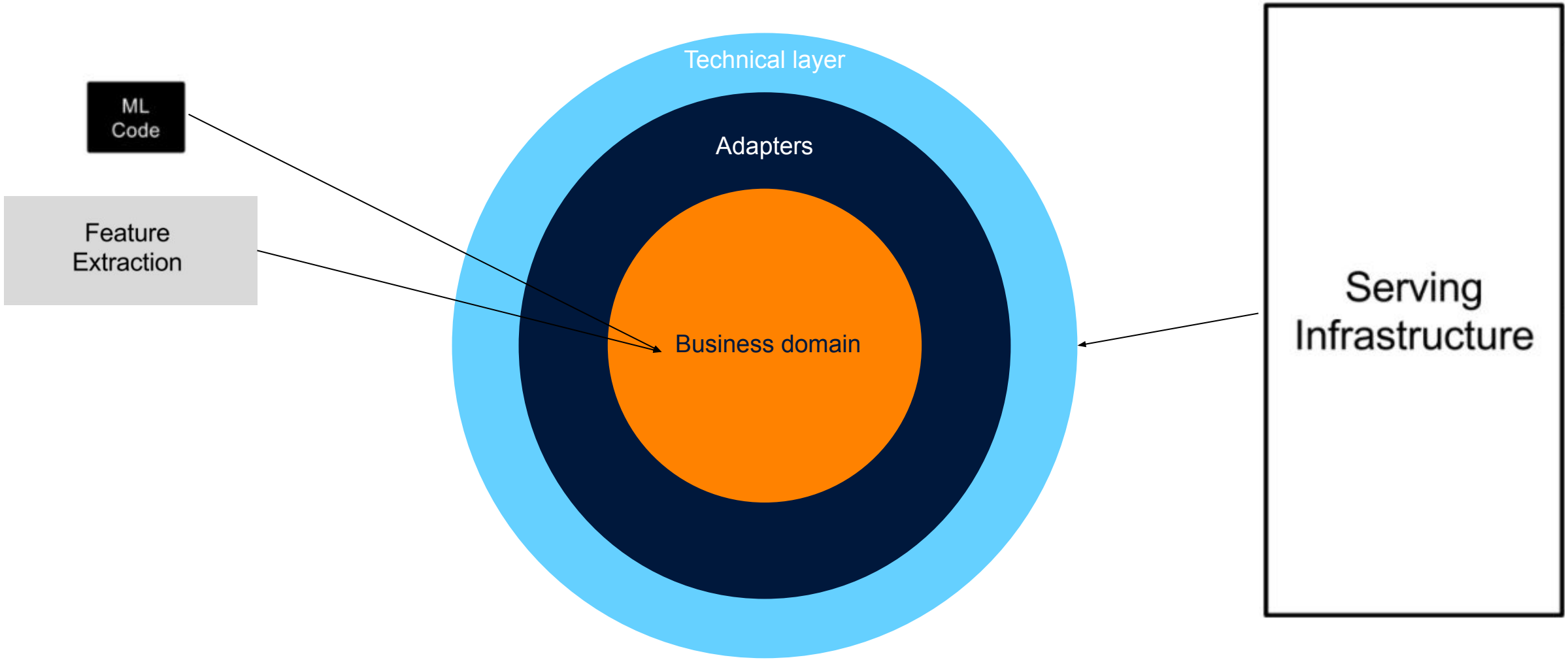
Ré-entraîner un modèle

Les annotateurs ont fini une grosse campagne d’analyse de documents, il est temps de réentraîner et déployer une nouvelle version du modèle

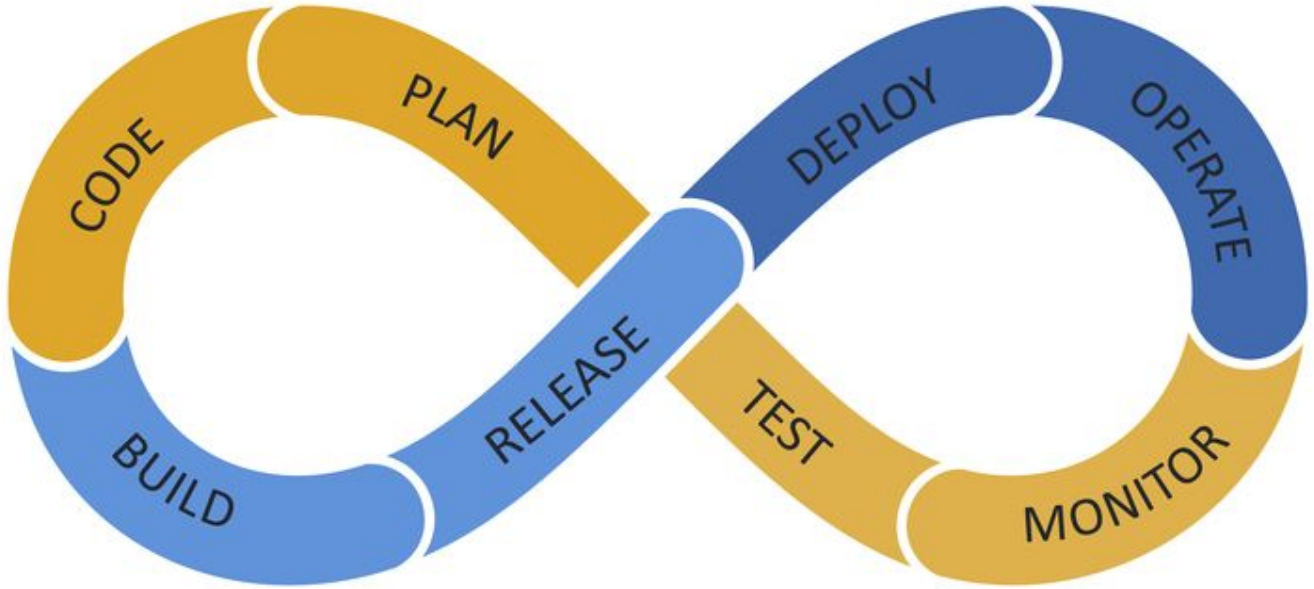


DILITRUST

Comment l'architecture hexagonale sert la maintenabilité d'un service basé sur du ML ?



Comment s’inspirer des pratiques de déploiement du DevOps ?



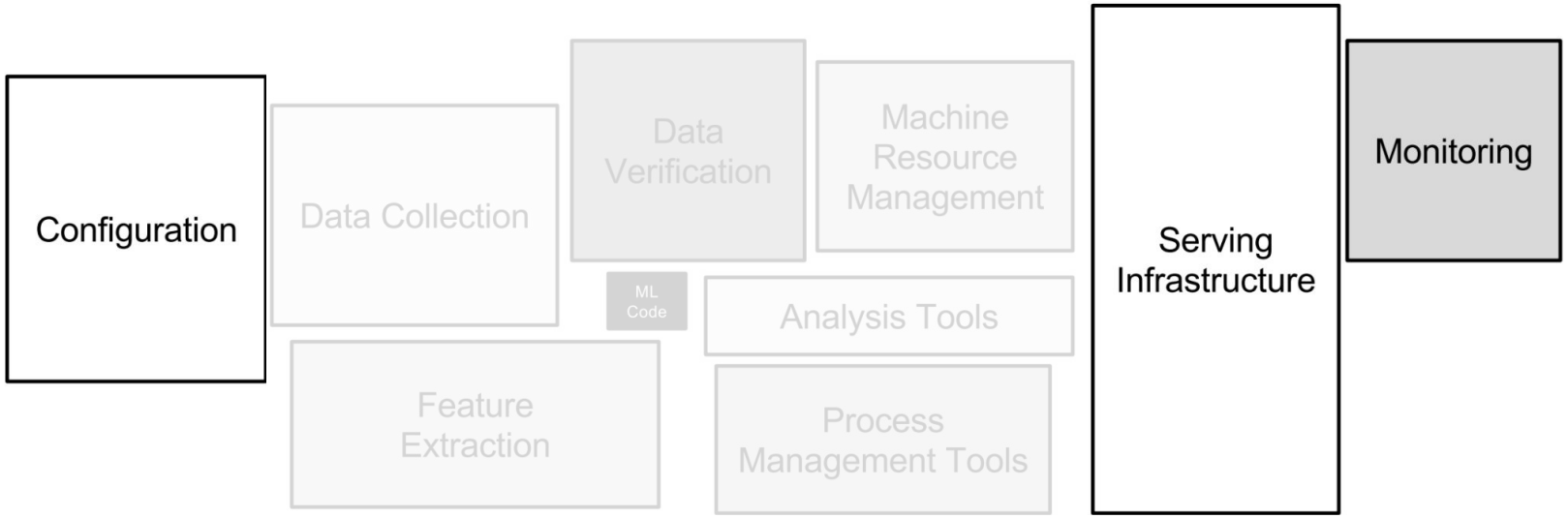
Designons notre pipleine de déploiement en lui assurant ces attributs :

- Idempotente
- Automatisé, pilotable manuellement
- Versionning général, GitOps
- Générique, simple à étendre

Un pilotage plus fin du déploiement permet également :

- Optimization des coûts en diminuant l’empreinte mémoire GPU d’un modèle, un GPU peut exposer plus de modèles
- Itération plus rapide sur les optimisations à l’inférence (quantization, re-write de tooling en Rust)

Composants frontaux



Infrastructure :

- Code : terraform, Kubernetes
- Configuration : GitOps
- Runtime: Container docker standardisé

Serving layer (inférence) :

- Configuration : artefact versionné
- Modèle : artefact(s) versionné
- Code : code applicatif versionné

Monitoring :

- Collecte de métriques
- Alerting
- Autoscaling

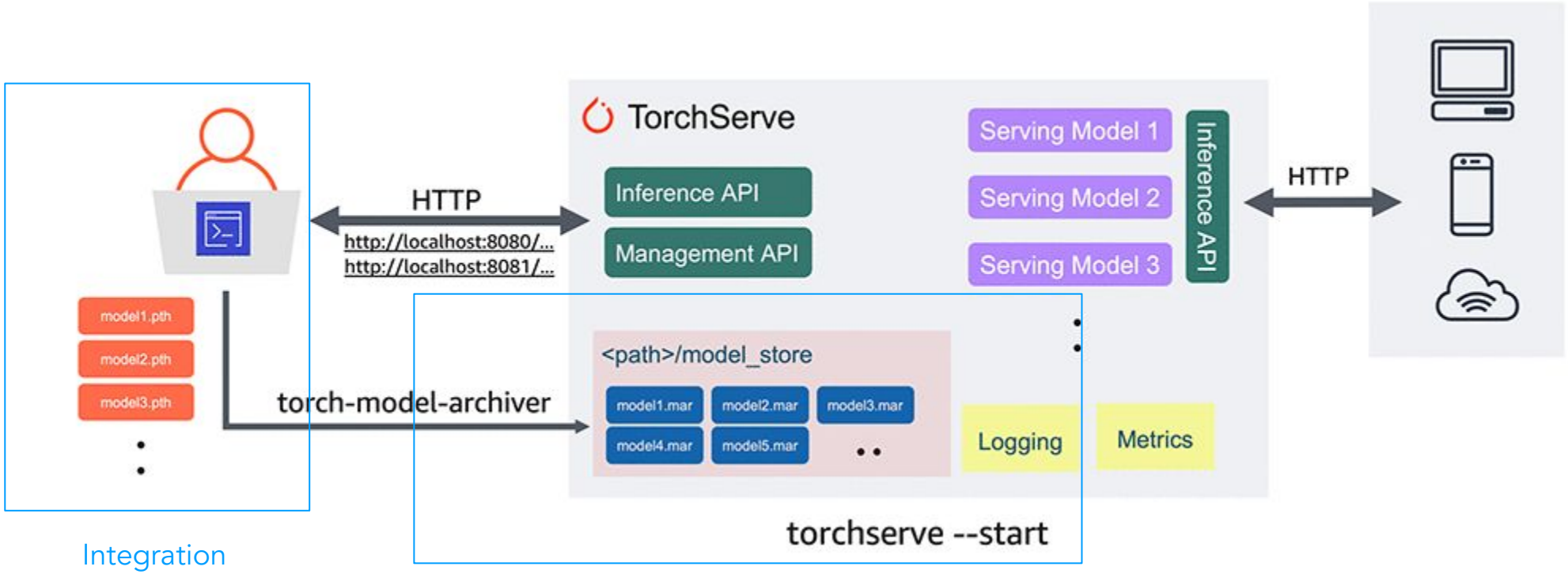
Monitoring

Métriques :

- Technique : État de l'infrastructure (empreinte mémoire, latence des requêtes, etc.)

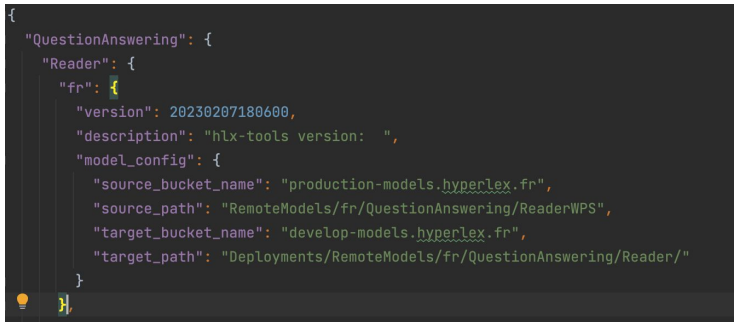
- Fonctionnelle : Pertinence du modèle face aux cas d'usages utilisateurs :
  - Drifts données, drift de concepts (trends des réseaux sociaux, contenu généré par LLM)

Torchserve


TorchServe
Inference API
Management API
Serving Model 1
Serving Model 2
Serving Model 3
Inference API
HTTP
http://localhost:8080/...
http://localhost:8081/...
HTTP
model1.pth
model2.pth
model3.pth
torch-model-archiver
<path>/model_store
model1.mar
model2.mar
model3.mar
model4.mar
model5.mar
Logging
Metrics
torchserve --start
Integration
Deployment

Torchserve: model & configuration artefacts

```
{
  "fr_reader": {
    "1.0": {
      "marName": "fr_reader.mar",
      "defaultVersion": false,
      "minWorkers": 1,
      "maxWorkers": 1,
      "batchSize": 1,
      "maxBatchDelay": 100,
      "responseTimeout": 120
    }
  }
}
```



```
{
  "QuestionAnswering": {
    "Reader": {
      "fr": {
        "version": 20230207180600,
        "description": "hlx-tools version:  ",
        "model_config": {
          "source_bucket_name": "production-models.hyperlex.fr",
          "source_path": "RemoteModels/fr/QuestionAnswering/ReaderWPS",
          "target_bucket_name": "develop-models.hyperlex.fr",
          "target_path": "Deployments/RemoteModels/fr/QuestionAnswering/Reader/"
        }
      },
```

Configuration de service cible :

- Aspects techniques du serveur d'inférence
- Aspects applicatifs, sources des modèles

Model binary & configuration

```
{
  "fr_reader": {
    "1.0": {
      "marName": "fr_reader.mar",
      "defaultVersion": false,
      "minWorkers": 1,
      "maxWorkers": 1,
      "batchSize": 1,
      "maxBatchDelay": 100,
      "responseTimeout": 120
    },
    "2": {
      "defaultVersion": true,
      "marName": "fr_reader_2.mar",
      "minWorkers": 1,
      "responseTimeout": 120
    }
  }
}
```

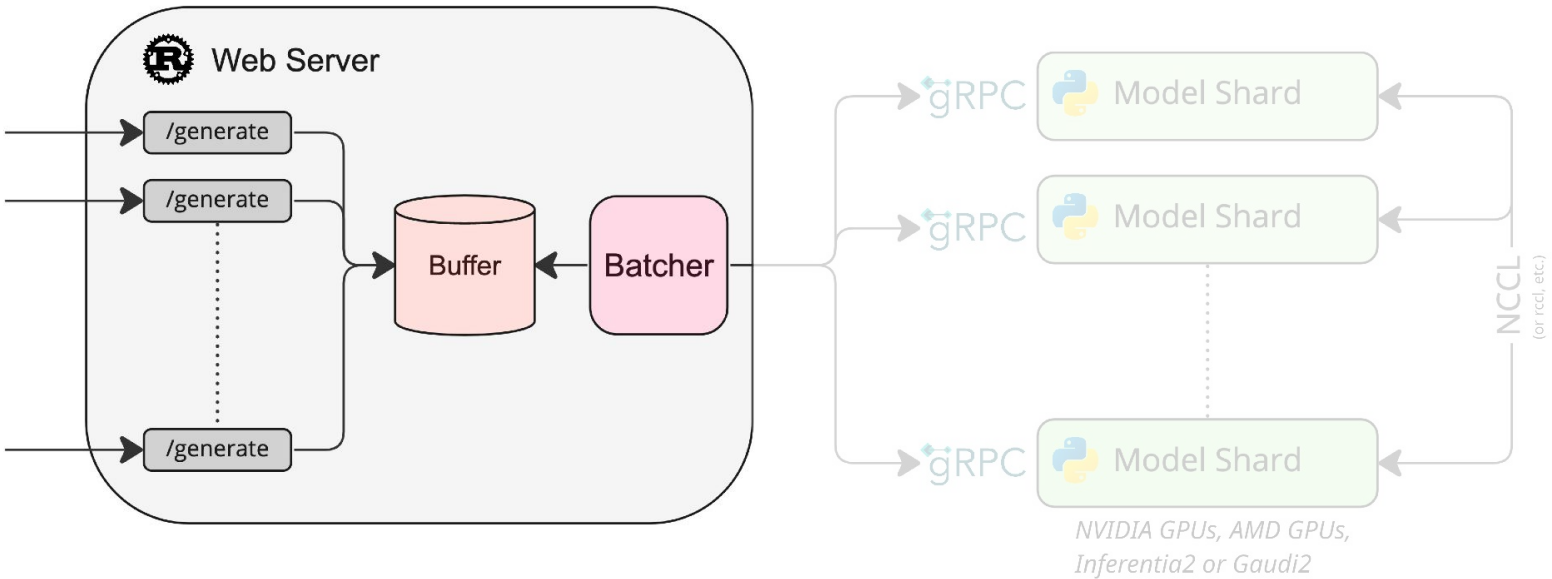
Configuration à déployer

Model archive

Text Generation Inference

Passer l'infrastructure d'inférence à l'échelle, pour exposer des modèles extrêmement performants.

- Back end d'exécution en Rust
- Interfaces en Python
- Intégration des features d'optimisation des performances au fil de l'état de l'art



Quelques techniques d'optimisation

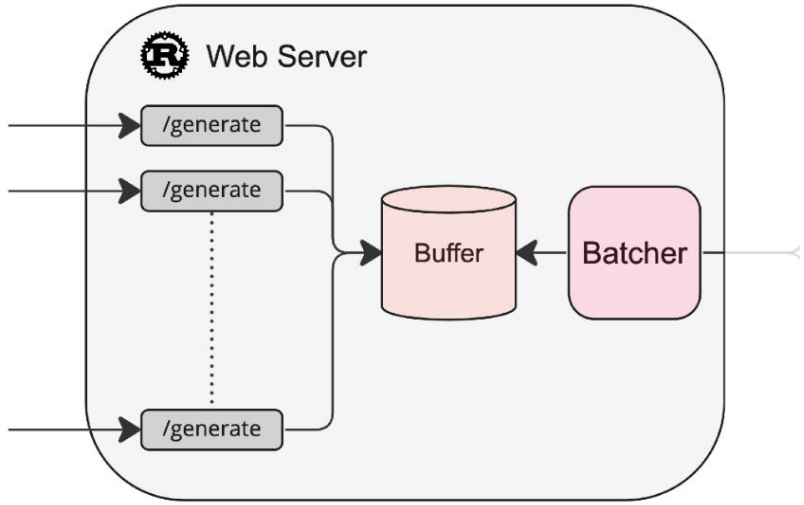
Feature	TGI - Text Generation Interface	vLLM - Versatile Large Language Model
Tensor Parallelism	✓ Utilizes Tensor Parallelism for faster inference	✓ Leverages Tensor Parallelism for high throughput
Optimized Transformers Code	✓ Optimizes transformer code for inference	✓ Employs optimized CUDA kernels for speed
Quantization	✓ Supports quantization methods	✗ Does not support quantization
Continuous Batching	✓ Batches incoming requests continuously	✓ Maximizes GPU utilization with continuous batching
Accelerated Weight Loading	✓ Accelerated weight loading for quick startup	✗ Does not include accelerated weight loading
Logits Warping	✓ Offers logits warping options	✗ Does not provide logits warping features
Custom Prompt Generation	✓ Allows custom prompt generation	✗ Does not support custom prompt generation
Fine-tuning Support	✓ Supports fine-tuned models for higher accuracy	✗ Lacks fine-tuning support
High Throughput	Moderate throughput	Excellent serving throughput
Paged Attention	✗ Does not include Paged Attention	✓ Efficiently manages memory with Paged Attention
Streaming Outputs	✓ Supports streaming outputs	✓ Enables streaming outputs for improved performance
Support for Various Models	✓ Compatible with various LLMs	✓ Seamlessly supports a wide range of Hugging Face models

Text Generation Inference : un déploiement sans aspérités

Docker image

\+

- Batch size
- Nb of token limits
- Speculation parameters
- Optimization parameters



Build du runtime cible

Déploiement du conteneur autonome

Model binary with technical configuration

Model Instance

[Hugging Face TGI Documentation]

Questions

Amine Saboni
@Tundji

dilitrust.com