

Distinct features

samedi 15 août 2020

11:37

Best experience for

* Setup on an enthusiast windows machine
* Setup on the three major cloud vendors
* Lifecycle management and manageability
* Error messages, diagnostics, end to end monitoring
* Resource optimized, GPU is optional
* Collaboration between several teams
* Auditability and security
* Open source librairies and ecosystem integration
* Oustanding documentation, videos and tutorials (integrated)

Assumptions

* You are a datascientist and you know Python
* You are not an infrastructure and operations expert
* You want to keep your data private
* You want to avoid vendor lock-in
* You are not trying to let business users manage production systems
* Easy to integrate AI services and pipelines, but no end-user integrated solutions and UIs

V1 opinionated choices starting with

* Librairies : jupyter, nbdev, fastpages, scikit-learn, Pytorch, fastai, spacy, huggingface, rossmann, rapids only
* Applications : structured, text, images
* Kubernetes : OKD on a single but powerful Windows machine

Later versions

* Cloud deployment
* More applications
* Enterprise security and data sources integration
* Mac support
* Tensorflow support

Value proposition

samedi 15 août 2020

11:45

Setup a complete deep learning development, training and production environments in minutes

* You are a developer with good high level IT culture
* You are willing to learn Python and deep learning framework
* But you are not interested in learning how to manage the underlying environments

Incredible installation, setup and documentation / training experience on the way

Very easy update/upgrade story

AI solutions are a team effort involving IT and business roles

Natively integrated and guided AI services, datasets, models lifecycle management

A distinct experience for each role

Great collaboration features

Focused on individual services and pipelines, not end to end integrated, ready to use solutions

Annotation is the key activity, supported with great graphical annotation tools

The best technologies are integrated to reduce the workload of annotation : self-supervised learning, zero-shot, clustering, active learning, data augmentation

Structured model of the predictions evolves over time : the platform should version it and help with changes

Model and pipeline is a development activity, we don't try to make it accessible to non technical roles

Model hyperparameters tuning support is manual and a technical/expert activity

Auditability, performance + availability + resource consumption monitoring is mandatory everywhere

User feedback / logs and user stats are crucial for the continuous improvement cycle

Great support is included out of the box in the platform

Focus on real-world requirements : filter unrelevant or offensive content, anonymisation, spell checking, autocomplete, suggestions, disambiguations, external services integration, superresolutions

User behaviour in the loop is supported out of the box : let the humans do what they are good at, the AI service is there to integrate and to help

A reproducible and secure production deployment pipeline is mandatory

Versioning, tests, non regression, step by step environments promotion are supported out of the box

High level architecture

samedi 15 août 2020

11:45

6 independent environments

* Management plane
* Sandbox
* Dev
* Train
* Valid
* Prod

Activities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Sandbox | Dev | Train | Valid | Prod |
| OS - librairies  Kubernetes  AI platform | Test |  |  |  |  |
| AI services  Pipelines  Business processes |  | Dev | Test | UAT | Prod |
| Datasets |  | Import  Test | Annotate  Evaluate | Collect | Collect |
| Models |  |  | Train | Valid | Prod |

Base services

* Git code versioning
* Docker images repository
* Python modules repository (pip / conda)
* Python services FastAPI
* Jupyter Notebooks server
* OKD high level API
* SQL database with JSON indexing
* Block storage / filesystem
* .Net API services
* Blazor Bootstrap Uis

Data Model

samedi 15 août 2020

14:13

Organization

Contains organizations

Contains users

Admin users

Can be billed in resources for each service call

Can set limits to the resource consumption of its users

User

Email / Password

Name / Title / Alias / Picture

Users created freely

Can be members of several organizations

Can be request to be added inside an organization

Can be added to an organization by org admin

Recommended :

1. Reflect your company structure as needed

Org level 1 : company

Org level 2 : domain

Org level 3 : team

2. Project teams are automatically created

Project

Contains assets

Projet team organization

Owner organizations

Trusted organisations

Automatic history of all changes

Blog decribing the rules, the lifecycle …

Owner organizations admins grant roles in a project

Assets visible/searchable to owners and trusted Organizations

Assets can be modified / deployed only if you have a role

Other projects can request assets to be Shared or Forked : dependencies are explicitly allowed and are tracked, a dependency is positioned on a tagged version only

Platform defines a built-in projects from which you can import read-to use datasets, services, processes

Role

Roles always granted in the context of a project and to a user

Data engineer

Data analyst

Business expert

End user

For each asset

Dev phase

Train phase

Valid phase

Prod phase

Compute cluster (physical availability)

Datacenter locations

Datacenter rooms

Compute nodes : CPU core, Memory, Local storage, Bandwidth

Networking services : WAN bandwidth

Storage service instance (physical availability)

Datacenter locations

Datacenter rooms

Block : IO, disk space

Database : API calls, disk space

Production zone (logical availability, lifecycle and security isolation)

Compute clusters

=> compute quota in each cluster : cores, GPUs, memory, storage, bandwidth

Storage services (low level)

=> dedicated endpoint in each storage service

High availability, failover and backup strategy

Shared high-level service

Git service

Artifacts repository

Datasets repository

Notebooks launch service

API hosting service

Log database

Recommended production zones

* Sandbox (small HA, few small GPUs)
* Dev / Train (no HA, many big GPUs)
* Valid (no HA, few small GPUs)
* Prod (HA, many small GPUs)

Project environment

Production zone

Compute, Storage, Bandwidth quota reserved in the production zone

HA, failover, backup strategies inherited from the production zone

Secrets for all the resources

Resource consumption monitored per project and per user : CPU time, GPU time, Memory, Bandwidth, Storage

Assets promoted from one environment to another

Strong protection on data access in production zone / transfer of anonymized data back to

Recommended project environments

* Dev / Train / Valid / Prod
* Sandbox only for "technical" test projects

Assets

Code (only in dev environment)

Dockerfiles

Notebook

Python module

Task code : (train) (valid) (prod)

Pipeline code

Service code

Busines process state management service code

Metadata / Interface (update only in dev environment, view in other envs)

Module doc

Task model (tasks can be trainable in the owner project only when shared, fork them otherwise)

Pipeline model

Service model

Business process model

External service calls

Import/export processes

Prod control and continuous improvement model (bias control, perf monitoring)

Data (first ones only in dev/train environments, last ones in valid/prod environments)

Dataset

Trained model weights

Test results

Business process state

Service logs

Business process logs

User feedback

Annotation / continuous improvement

Dataset visualization & exploration UI

Business process visualization & exploration UI

Personalized annotation UI

Training dashboard

Production dashboard

Packaging

Versioned AI service

Versioned business process state service

Packaged trained model for a versioned AI service (interface compatibility check)

Deployment

Environments

CICD pipelines

API management

Deployed service

Deployed business process

Technical dashboards & diagnostics

Alerts

Maintenance tasks / schedules

Concepts

lundi 13 septembre 2021

23:19

Scopes

Language [region]

* Business domain
  + Enterprise vocab
    - Channel specifics

Datasets for each scopes

+ dates : datasets evolve over time

Knowledge corpus

* Categories
  + Concepts
    - Instances
* Contextual knowledge corpus

NLP pipeline

Character normalization

Tokenization rules / Sentences splitting

System entities / Sensitive entities

Vocabularies (frequent words vs infrequent subwords)

Entity types - links to knowledge corpus

Classification

Context elements + NLP pipeline results

=> classification through rules & models