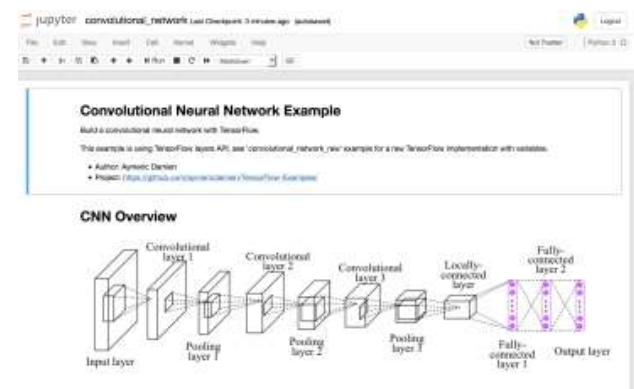


# Watson Machine Learning Accelerator

# Watson Machine Learn – Accelerator

- IBM PowerAI is a packaging of ML/DL frameworks for Linux on Power systems
  - Tensorflow, Caffe, Pytorch....
    - Compiled and optimized for IBM Power Systems
  - Growing number of frameworks since first release
- IBM WML-A is PowerAI + cluster management framework and deep learning platform:
  - IBM Spectrum Conductor and Deep Learning Impact
  - Notebooks, Docker, Distributed Deep Learning, Fabric algorithms



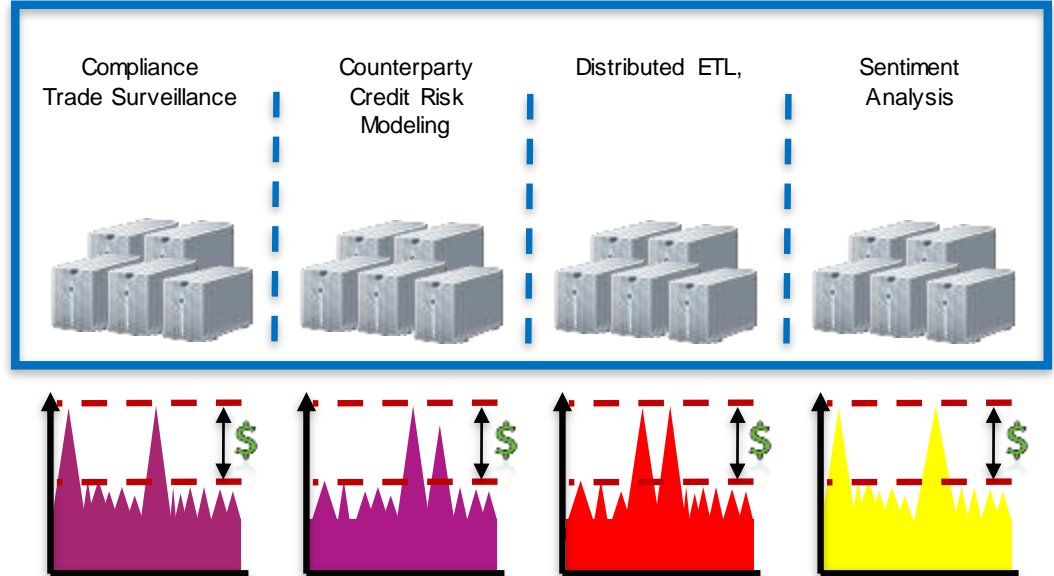
# Apache Spark

- Apache Spark is an open-source cluster-computing framework.
- Spark facilitates the implementation of iterative algorithms and exploratory data analysis.
- Spark schedules jobs through a cluster management system and requires a distributed filesystem.
- Why Spark?
  - Unified Analytics Platform
  - Multi-language (Python, Scala, R, SQL...)
  - Performance: faster than MapReduce
  - Diverse ecosystem
  - Very active open source project



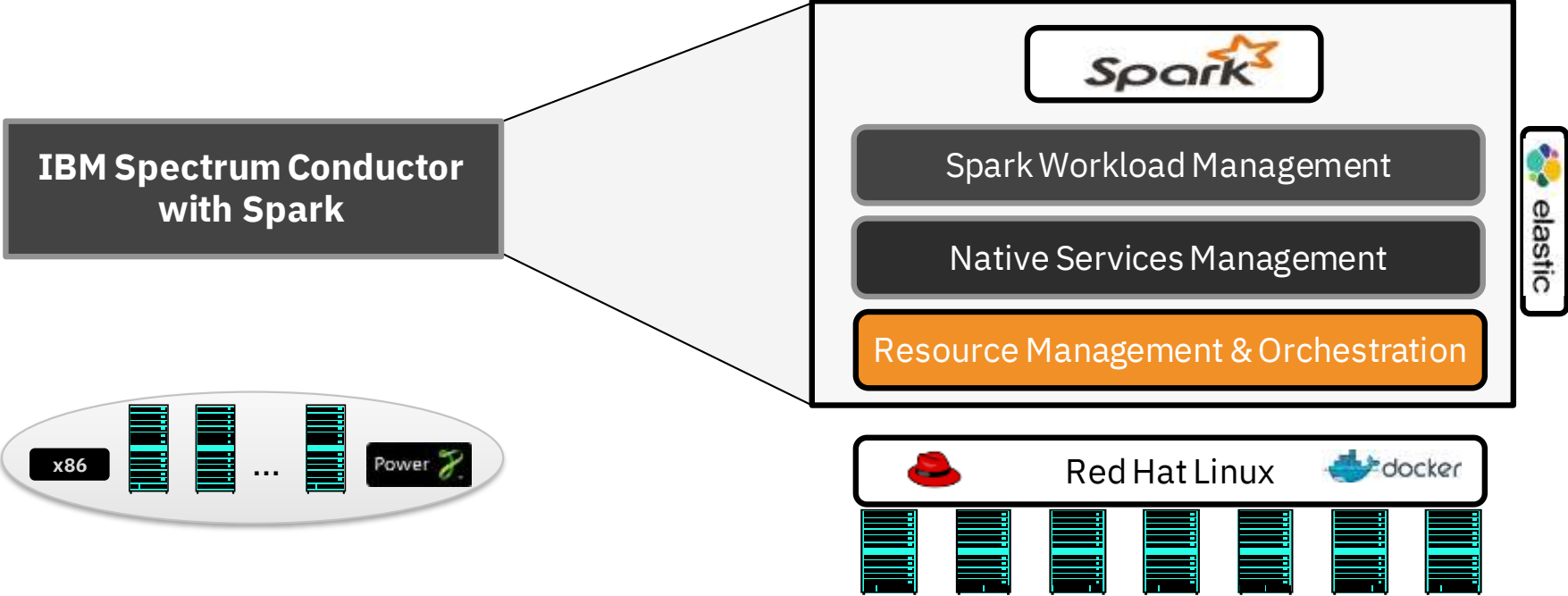
# Challenges managing spark applications

- In a word: siloed environments
- Different Lines of Business
- Multiple Spark versions
- Multiple notebooks and versions
- Security, governance
- SLAs
- Development, test, production
- Diverse data sources



Low utilization → Higher cost

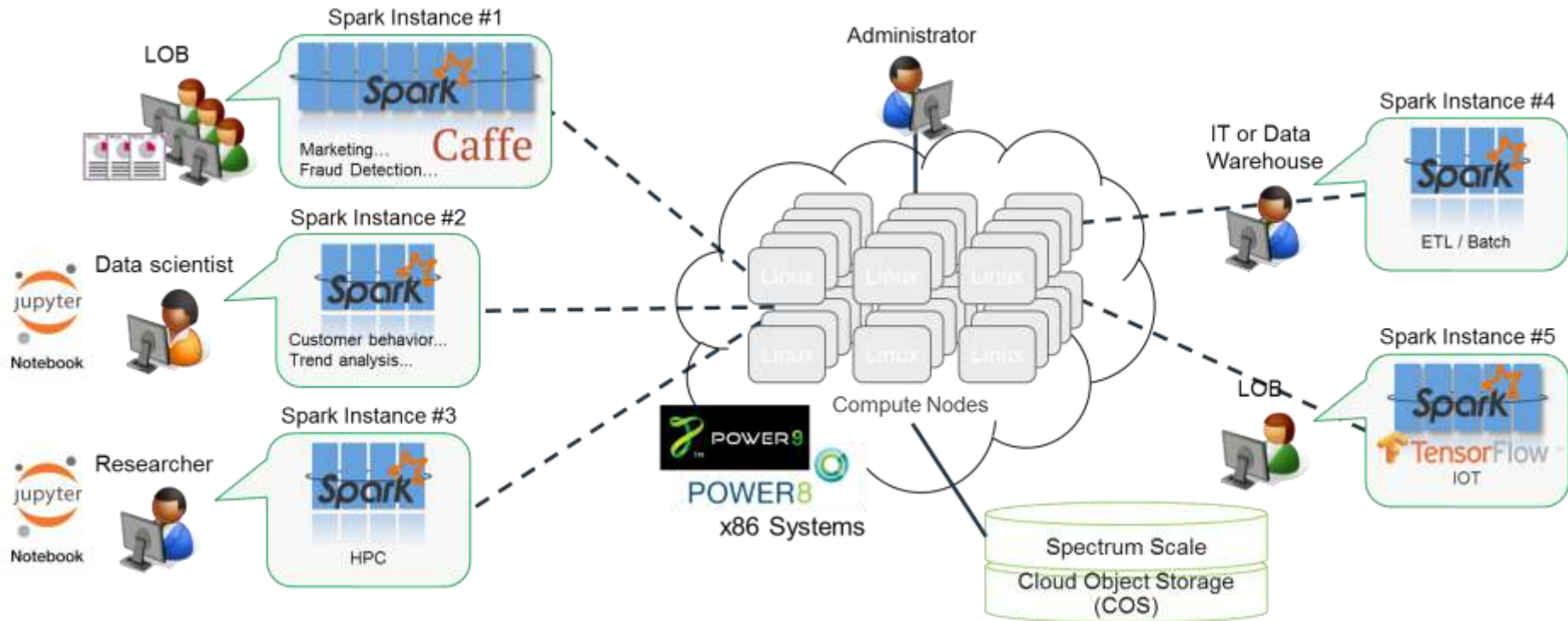
# Spectrum Conductor with Spark



# Key concepts

- Instance groups
  - Defines a spark cluster
  - Introduces multi-tenancy
  - Isolates environments (security)
- Users and consumers
  - How binding is done at the OS level
  - Impersonation of a consumer
- Resource groups
  - Defines a pool of resources
    - » CPU resources
    - » GPU resources
  - Defines slots for resource management
- Resource plans
  - Sharing of resources
  - Reduced silos

# Instance groups



# Resource plans

The screenshot shows the IBM Platform Symphony Advanced Edition interface. The main window displays a 'Resource Plan' for 'ComputeHosts'. The table below shows the resource allocation details:

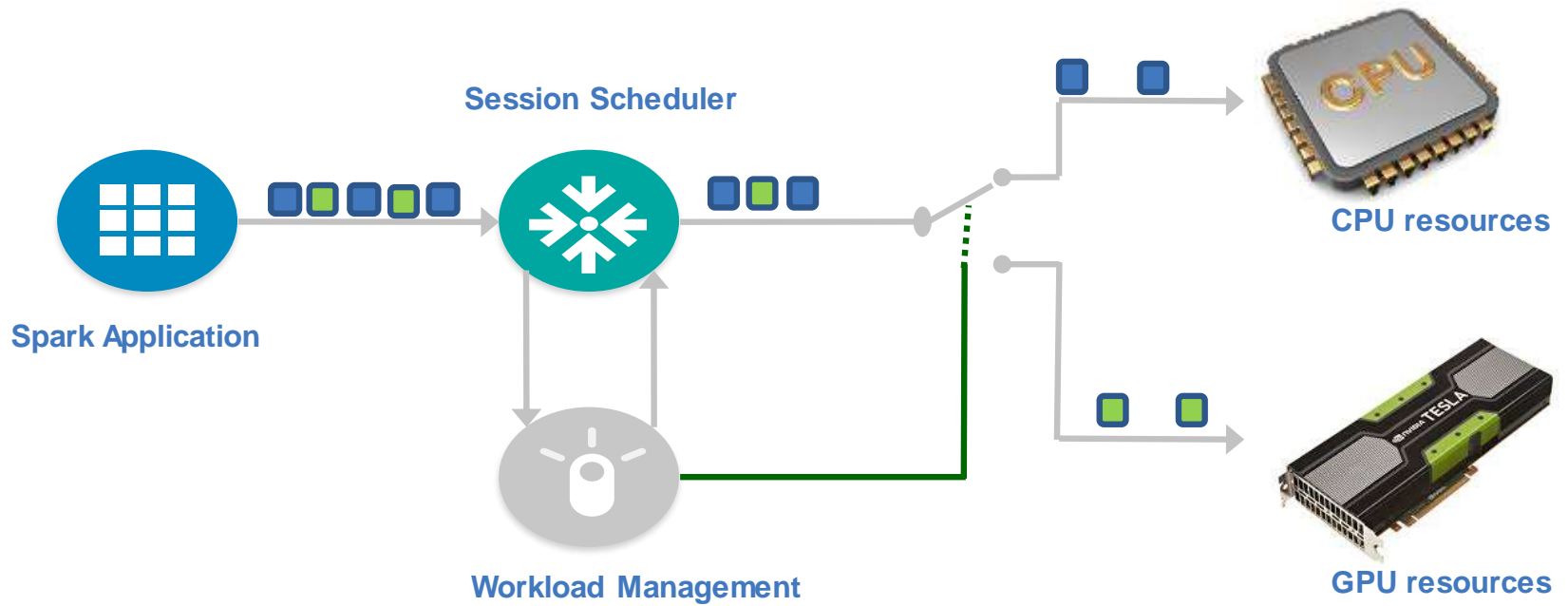
Consumer	Owned Slots	Consumer Rank	Lend   Limit	Borrow   Limit	Share Ratio   Limit
symphdemo	108				
SymTesting	0	0			1
Symping51	0	0			1
Total	0				
Balance	0				
SampleApplications	0	0			1
SOASamples	0	0			1
EclipseSamples	0	50			1
Total	0				
Balance	0				
SymExec	0	0			1
SymExec51	0	0			1
Total	0				
Balance	0				
MapReduceConsumer	40	0			1
MapReduce61	10	0			1
MapReduceHighPriority	30	0			1
MapReduceDefault	0	0			1
Total	40				
Balance	0				
SampleAppOPP	0	0			1
GpuTestApp	0	0			1
Total	40				
Balance	68				

The 'Lend Details' panel shows the configuration for 'MapReduce61' (ComputeHosts, 00:00-24:00). It lists consumers to lend to, including symphdemo, SymTesting, SampleApplications, SymExec, MapReduceConsumer, and GpuTestApp. The 'Borrow Details' panel shows the configuration for 'MapReduceDefault' (ComputeHosts, 00:00-24:00). It lists consumers to borrow from, including symphdemo, SymTesting, SampleApplications, SymExec, MapReduceConsumer, and GpuTestApp. The 'Borrow / Order' column in the Borrow Details panel shows values of 1 and 2 for MapReduce61 and MapReduceHighPriority respectively.

- Sharing of resources while preserving ownership
- Change plan on-the-fly
- Allocations happen in runtime (dynamic allocation)
- Enables SLA management



# GPU support



- Accelerating Spark applications with GPUs
  - Conductor scheduler interfaces with Spark scheduler to ensure that GPU resources are assigned to the applications that can use them.

# Jupyter Notebooks | Docker

- Notebooks are created within an instance group
- Created for a user
- May leverage collaboration
- Fired off from Conductor

The screenshot shows a Jupyter Notebook interface with a title bar 'convolutional\_network' and a 'Logout' button. The main content area features a heading 'Convolutional Neural Network Example' followed by a brief description and a 'CNN Overview' diagram. The diagram illustrates a neural network architecture with three convolutional layers, a locally-connected layer, and two fully-connected layers leading to an output layer. Below the diagram is an 'MNIST Dataset Overview' section with a small image of handwritten digits. At the bottom, a code cell is visible with the following Python code:

```
In [1]: from __future__ import division, print_function, absolute_import

# Import MNIST data
from tensorflow.examples.tutorials.mnist import mnist
mnist = mnist.read_data_sets("./mnist/", one_hot=True)

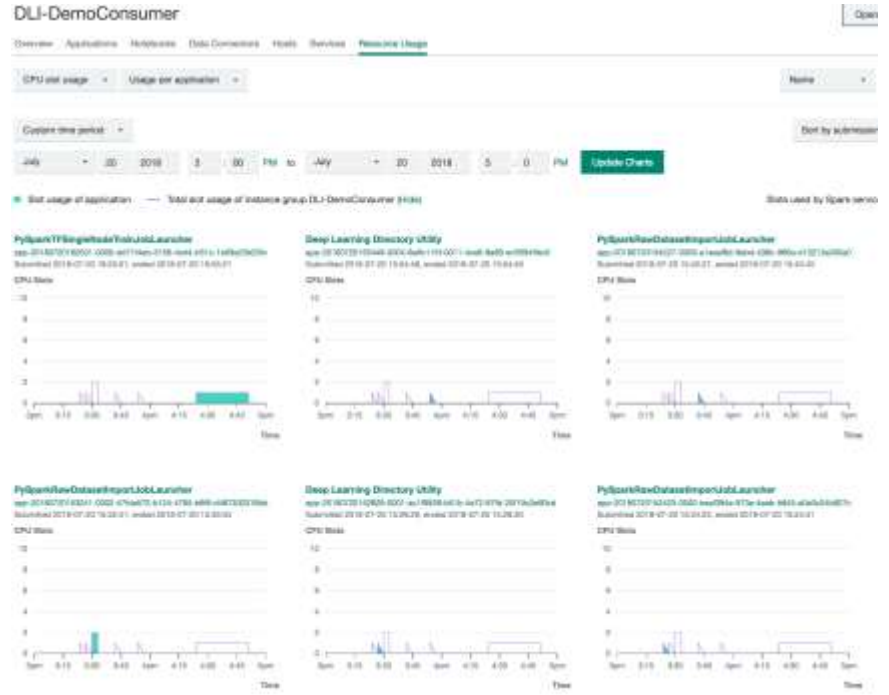
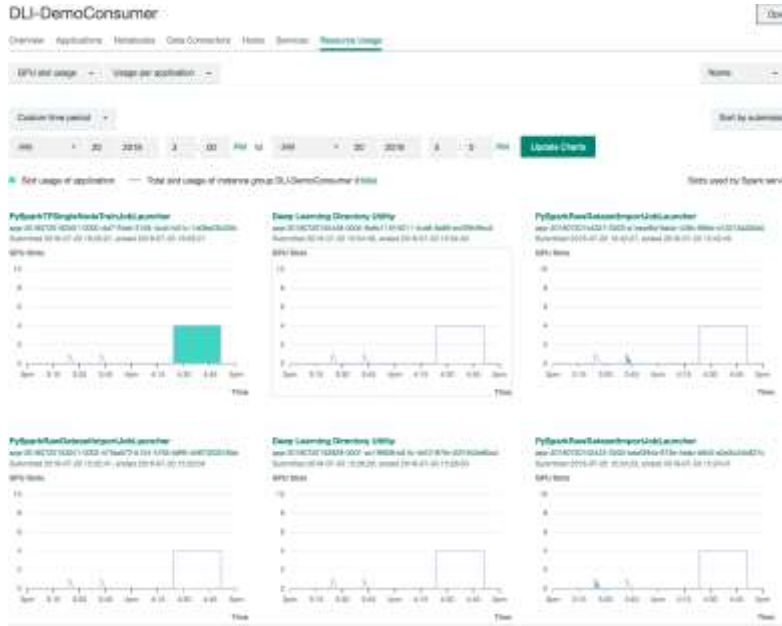
import tensorflow as tf
import tensorflow.python.nn as nn
import numpy as np

# Retrieving /mnist/train-images-idx3-ubyte.gz
# Retrieving /mnist/train-labels-idx1-ubyte.gz
# Retrieving /mnist/train-images-idx3-ubyte.gz
```

- Spectrum Conductor includes full integration with Docker
- Instance groups / notebooks may run in a Docker container



# Monitoring



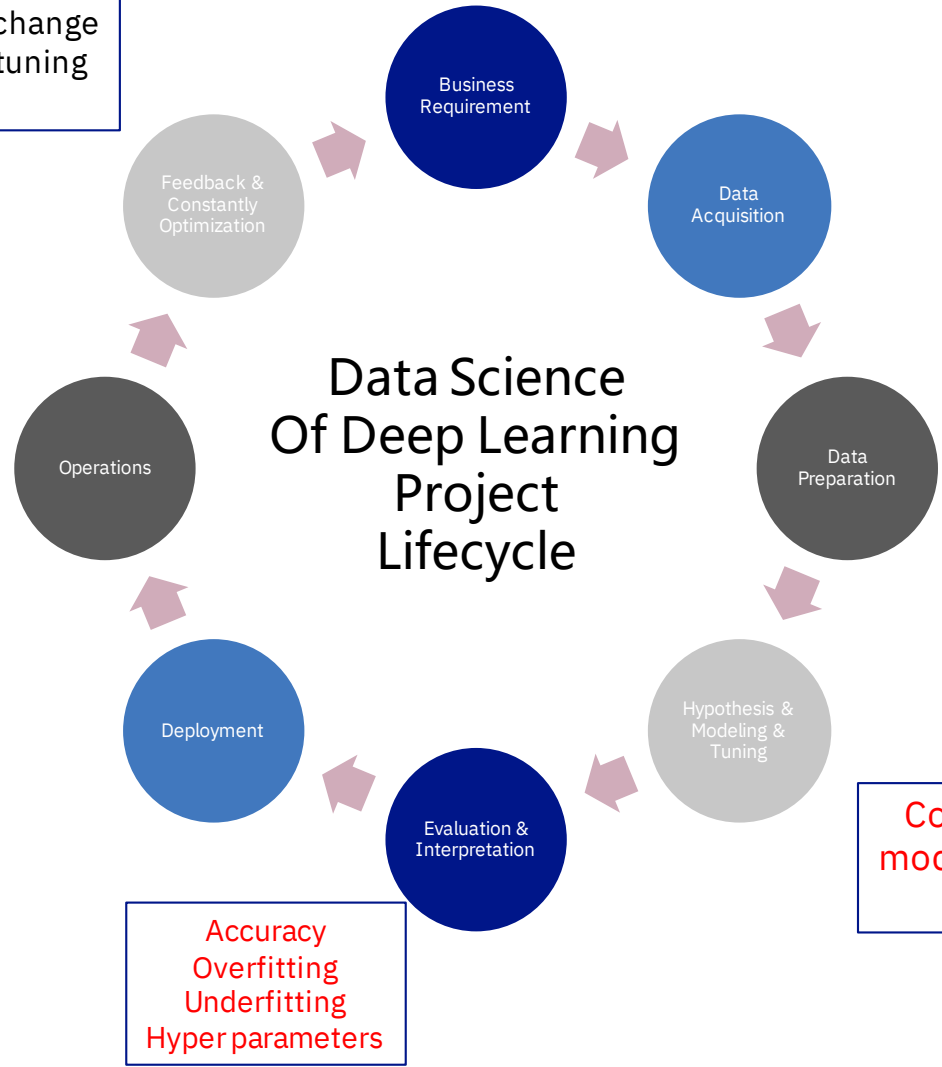
- Integrated Elastic Search, Logstash, Kibana for customizable monitoring
- Built-in monitoring Metrics
  - Cross Spark Instance Groups
  - Cross Spark Applications within Spark Instance Group
  - Within Spark Application
- Built-in monitoring inside Zeppelin Notebook

# Deep Learning Impact

# Challenges of deep learning

Business model / user data change  
→ constant neural network tuning  
or training required

Unified AI platform  
Maximize resource  
utilization



Most time is spent here  
~80%

Core piece. Understanding  
model issues, tuning models,  
long training runs

Accuracy  
Overfitting  
Underfitting  
Hyper parameters

# Spectrum Conductor DLI



## Faster time to results

Distributed training on multiple servers and GPUs includes optimized software and frameworks to accelerate training times.



## Improve accuracy

Greater neural network model accuracy with hyper-parameter search and optimization, and with training visualization and tuning assistance.



## Reduce time preparing data

Less time spent importing, transforming and preparing data. Use Spark to manage data sources and imports.



## Improve ROI with shared resources

Better ROI with multi-tenant access to shared resources, which allow multiple data scientists to run different models at the same time on the same resources.



## Simplify administration

A consolidated framework for deep learning, monitoring and reporting enables you to achieve faster time to results with simplified management.



## Add to IBM Spectrum Conductor

Add a deep learning solution to IBM Spectrum Conductor. This highly available multitenant framework is designed to build a shared, enterprise-class Apache Spark environment.

# Parallel data preparation

## – Transform Data

- Different Data dimension processing
- Resize data to fit the network input layer

## – Algorithm to keep the distribution of data

- Rescaling by cross-entropy loss method
- Hold-out vs Cross validation vs Bootstrapping

## – Parallel Data Import

- Integrate with ETL
- Parallel transfer huge raw data to lmbd or tensor record format

The screenshot displays the 'Deep Learning' interface. On the left, a table lists datasets:

Name	Format	Status
<input checked="" type="checkbox"/> cifar10-1mdb	LMDb	Created
<input type="checkbox"/> cifar10-1bn	Other	Created

The main window shows the 'cif10-1mdb' configuration for 'Image Classification'. It features a bar chart with 'Numbers' on the y-axis (0 to 5.5k) and 'Labels' on the x-axis (0 to 9). Below the chart are input fields for 'Training folder', 'Validation folder', 'Testing folder', and 'Label file'. A warning message states: 'If you want to use this dataset for validation purposes, you must specify a validation folder. If left blank, the dataset will be created without a validation folder and you will not be able to use this dataset for validation.' Buttons for 'Create' and 'Cancel' are at the bottom.

Below this is the 'New Dataset' section with the following options:

- Create a dataset from:
- LMDBs
- TensorFlow Records
- Images for Object Classification
- Images for Object Detection
- Images for Vector Output
- CSV Files
- Other

A 'Cancel' button is located at the bottom right of the 'New Dataset' section.

# Parallel training

– Different optimizers in parallel

– Relationship among:

- Iteration Number( $\tau$ )
- Node number(K)
- GPU Number(n)
- Communication Overhead(s)
- Accuracy

–  $M_a(b,K,n,\tau)$  vs single node

Configuration Network File

\* Model name: cifar10-TF-model-DemoConsumer

\* Training dataset: cifar-TF-DemoConsumer

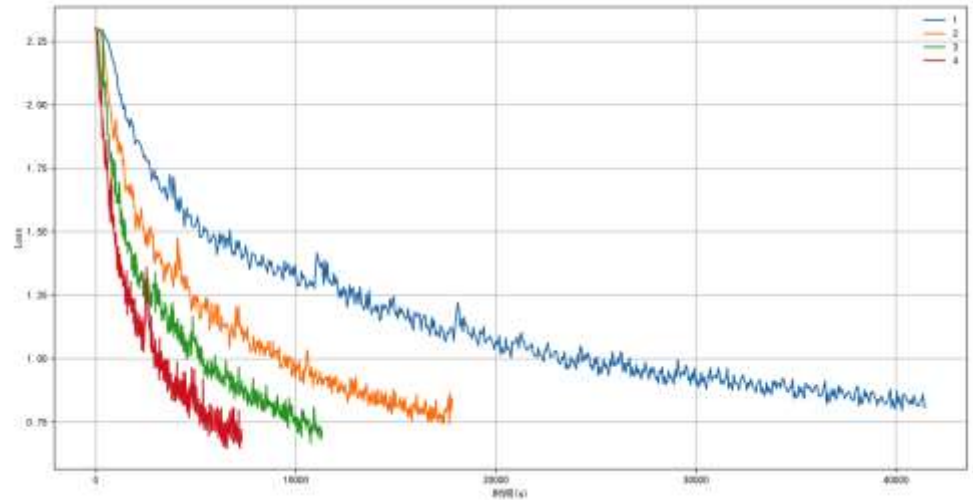
Model description:

Training engine: Distributed training with IBM Fabric and auto-scaling

**Hyperparameter**

\* Learning rate policy: Fixed

Update Model Cancel





# Hyper parameters

– Search:

- Random
  - The optimal solutions is above 5% in the whole space
  - 600 – 800 search may find a solution near the optimal solutions
- Tree-structured Parzen Estimator
  - Modeled by generative process of hyper-parameters, replacing the distributions of the configuration prior with non-parametric densities
  - 10% additional calculation effort than random with around 30% accuracy improvement
- Bayesian Estimator
  - Widely sample data and leverage multivariate Gaussian distribution get the  $\theta$
  - Calculate EI and choose new sample point
  - Bayesian provide better method than TPE to jump out a local optimal solution
  - Better accuracy with massive trained result

- Parameter setting: optimizer, learning rate, weight decay, momentum.
- Workload setting: # of workers and GPUs, iterations, and so on.

Tune Hyperparameters for model:cifar10-single-caffe

A tuning task will launch multiple jobs to search hyperparameters. A new model will be created that contains the tuned hyperparameters.

\* Name of tuning job: cifar10-single-caffe tuning-20171204140301

The new model will be: cifar10-single-caffe tuning-20171204140301

Select search type for hyperparameters

\* Hyperparameter search type: Random Search

Random Search  
TPE (Tree-based Parzen Estimator)  
Bayesian

Tuning Parameter Settings

Input the parameters that will be tuned

\* Optimizer (select at least 1):

- SGD
- AdaDelta
- AdaGrad
- Adam
- Nesterov
- RMSProp

\* Learning rate range: 0.05-0.1

\* Weight decay range: 0.1-0.5

\* Momentum range: 0.5-0.9

\* Max batch size: 64

Tuning Workload Setting

Input the parameters to control tuning jobs and process

\* Number of workers: 1

\* GPUs per worker: 1

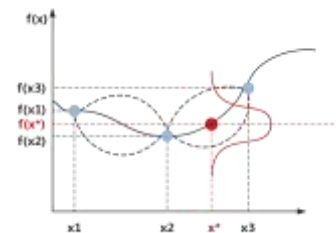
\* Max iterations: 900

\* Total tuning jobs number: 4

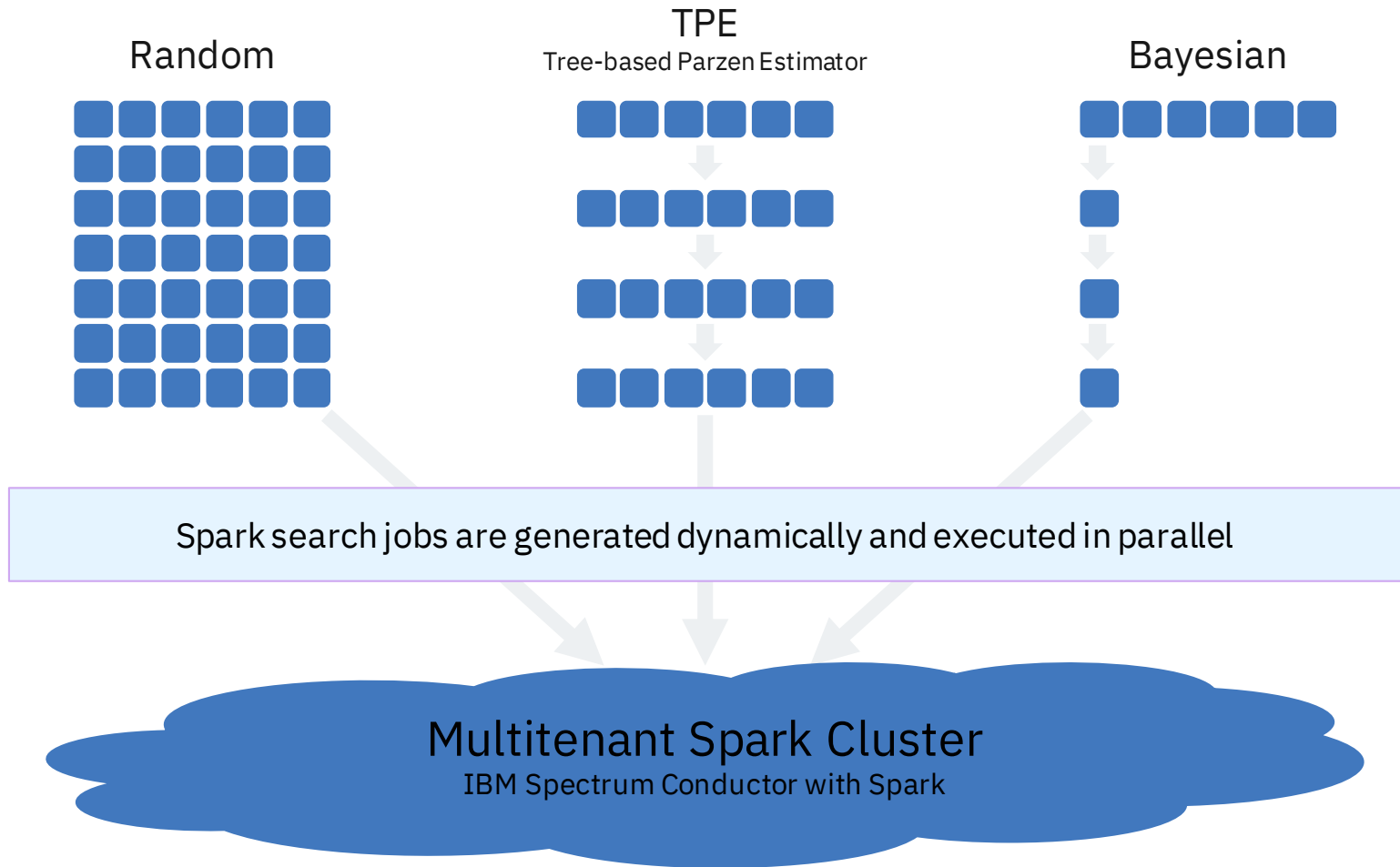
\* Max tuning jobs in parallel: 2

\* Max running time(minutes): 60

Start Tuning Cancel



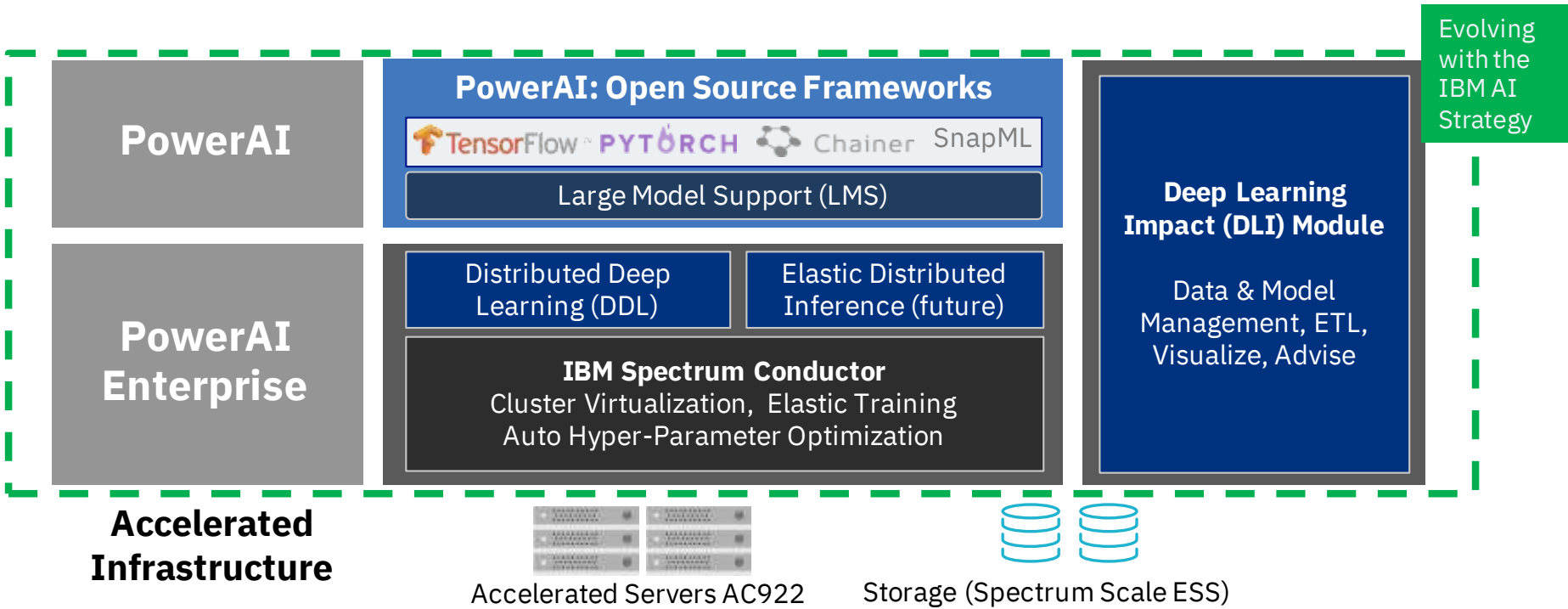
# Hyper parameter search





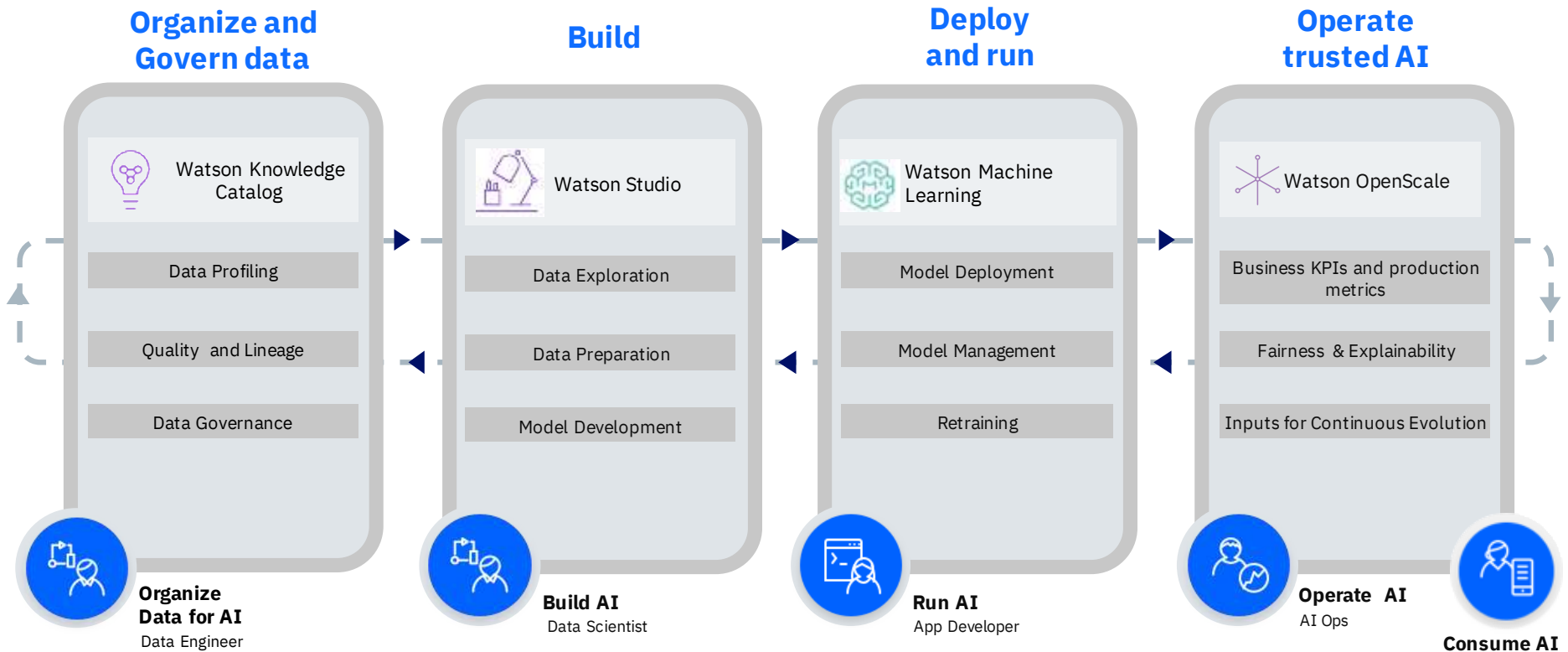
# Recent announcements

# Recent announcements @ THINK

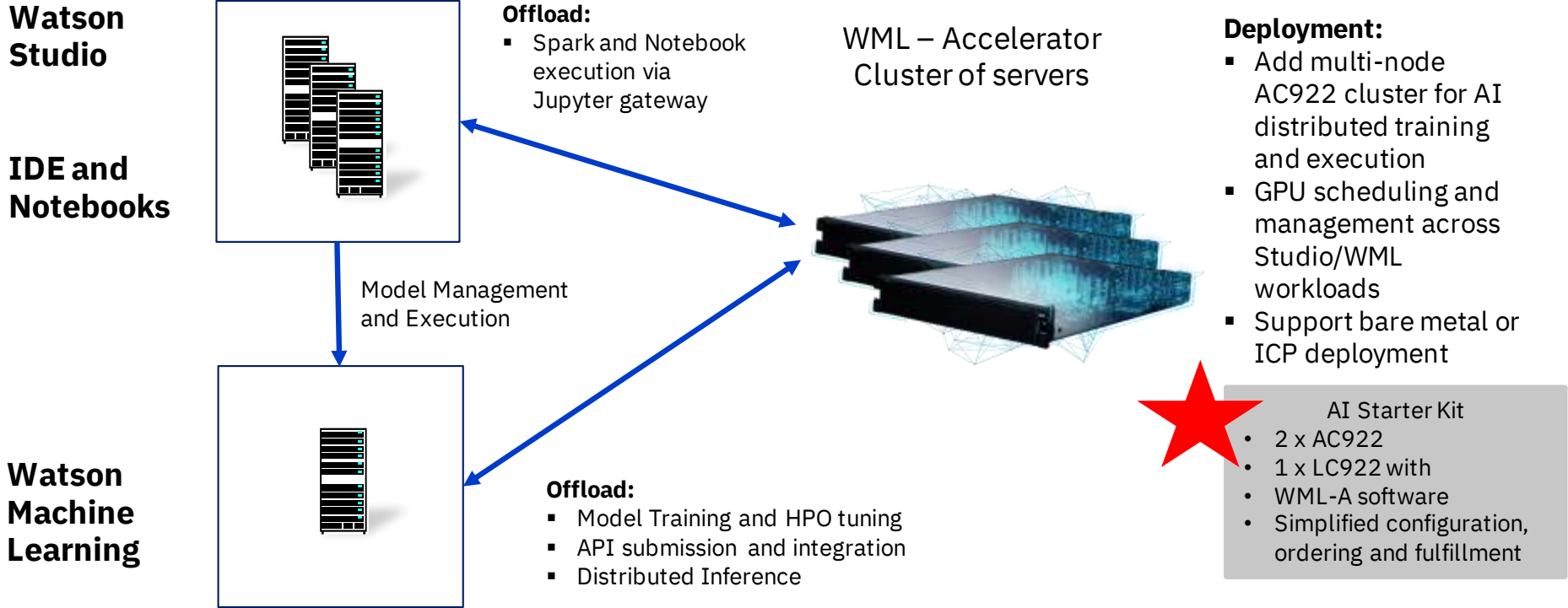


PowerAI Enterprise → Watson Machine Learning – Accelerator  
Integration with Watson suite

# Recent announcements @ THINK



# Recent announcements @ THINK



**IBM**