 [Translated from French to English - www.onlinedoctranslator.com](https://www.onlinedoctranslator.com/en/?utm_source=onlinedoctranslator&utm_medium=docx&utm_campaign=attribution" \o "Doc Translator - www.onlinedoctranslator.com)

**Slide 1**

What is MLOps? This is all things that will help and support the fact of bringing artificial intelligence algorithms into production, it is not about tools or technologies; it is mainly a question of culture and good practice.

Hi, my name is Guillaume Chervet, I'm an ML engineer at AXA France. Basically, I define my role as helping data scientists to bring AI algorithms into production.

I‘m part of in a team that works on real-time automatic document understanding projects. Generally, this kind of projects implements one or more Deep Learning algorithms.

Today, almost 80% of projects fail at the production stage. This does not surprise me because, this kind of projects are on the top complexities in all areas, all factors are met to make them fail.

**Slide 2: trio intro**

The first problem is a human problem, so today most applications require Developers and Operation engineers and it's already not easy to get everyone working together. On AI projects, we should also add a lot of expertise about data. We need Data experts.

Today, it’s almost impossible to be an expert in these 3 areas. Each party will have to trust one another and accept other party’s constraints.

**Slide 3: expensive intro**

Take into consideration that there are even more players (business owners, sponsors, quality, …): we work on projects that are generally extremely expensive.

This is an email classification project that worked very well. For us, it's an example project to follow.

Today the operating cost of the project per month is approximately 9 000 €, that’s a little more than 100,000 € per year. I think that we can still divide the costs by 2. Today on this project we classify about 10,000 emails per day. So, we estimate that each email classification costs about 3 cents euro.

If we decide to carry out one “model retrain” per year, for example, the cost of one email classification will increase to 4 cents euro. Here, I’m excluding the first implementation of the project… which cost a little more than 1 000 000 euros!

**Slide 4: expensive intro**

Here is a functional example of a mail classification.

For your information, all images you see here with a small text below are generated by an artificial intelligence algorithms.

**Slide 5: summary**

As you can see in these projects, we are at the top of the top of technical complexity, all factors are met for failure.

This presentation is divided into 4 parts.

1. In the first part we will introduce Deep learning and related issues.
2. In the 2nd part we will explain what we expect from a project in production at AXA France.
3. Then in the 3rd part we will have fun cooking a recipe. Recipe of what to do to make sure that an ML project will fail. The idea is to present to you some experiences that helped us to improve.
4. Thus, in the 4th part, we learned lessons from these experiences that will allow us to be on the way to make a project run successfully.

**Slide 7 : ML**

One of the subsets of artificial intelligence is machine learning and one of the subsets of machine learning is deep learning.

**Slide 8 : ML**

You will often hear about structured data usually being represented by an Excel file. In general, for this kind of problem where there is a limited number of inputs/outputs. We will use simple machine learning algorithms.

**Slide 9 : Deep**

When we have data like: sounds, images, videos. The number of possible inputs/outputs is generally higher. Consequently, we will generally use Deep learning algorithms which use neural networks.

**Slide 10 : Data intra**

So that's a diagram that came out of the book by François Chollet who is the creator of Keras.

When we work on classic code, in general, we have rules, plus input data. Then we write an algorithm that will allow us to generate answers.

In machine learning, it's a little bit different, we need input data and a lot of answers. Not just a little number, really a lot, thousands of examples 10,000, 100,000 or even more. These inputs allow to write code that will be able to generate the rules: an artificial intelligence, we will technically speak of a generating a “model”.

**Slide 11 : execute**

In production, classic code or AI is used in the same way.

We receive an input, we execute the algorithm, and then we get the result.

**Slide 12: Data is gold**

In the past, we used to talk about project as “model centric”, because we weren't sure we could make an appropriate AI.

Today most of the points on the AI ​​side: at least for our needs at AXA France, already have viable solutions. We know that we will be able to create an AI for our use cases.

On the other hand, the heart of the problem, really the thing that makes a project feasible or not: it is the data. If we have good data, of good quality, inputs and answers, a lot, that will really be game changer to the success of your project. The data is your treasure that will make the difference.

**Slide13 : workflow Dataset**

So now, we are going to explain the workflow of a project. We want to build an email classification project.

The first thing to do is to retrieve data: inputs and answers. On this project, it was easy. We already had emails and answers because in production we have humans who have been classifying the emails manually for several years already. We already have the data and the answers, that's perfect to build up our first dataset.

**Slide14 and 15 and 16 : workflow**

Then, with this data, we will be able to iterate to create an AI and build an API that we can put into production. After that, our application in production can be consumed by client applications.

**Slide 17 : workflow feedback loop**

In production, it is extremely important to set up what is called: the “feedback loop”. A virtuous circle. In the event of a classification error, the client applications must allow the error to be corrected manually. Over time, we will be able to build new dataset of a better quality and we will be able to improve our Artificial Intelligence.

**Slide 18 Demo intro**

We're going to do a demo of how a project workflow based on a well-known cat and dogs dataset from Kaggle.

**Slide 19 Demo Data**

Before starting the demo. Imagine that we have 20,000 files available. We will use 17,000 files for training the model. Then we will keep 3000 files aside for validation, to test our model on data it has never seen. This validation DATASET will also be used to test our production environments.

**Demo Time**

**Slide 20 After Demo Data drift**

If we feed our models with different data. Data very different on which models have learned. Predictions may deteriorate.

For our email classification project, for example, if there is a change of layout for the accident report, it is possible that the AI ​​will fail to recognize automobile claims. In this case, it is necessary to retrain and then re-deploy model in production.

Let's takes a sample with our Demo. Now let's imagine that our users start using plush or inpainting cats and dogs, as you can see the predictions no longer work correctly.

**Slide 21 The Grail**

The Holy Grail for us; is to be able to re-train and re-deploy in production without any manual action

**Slide 22 100% code**

This is a Google slide, a Deep learning project is a little bit less than 5% of data science, however it is 100% code!

**Slide 23: Contents 2nd part**

Now I will explain what is expected from a project in production at AXA France

**Slide 24: Needs**

The most important thing in a project is to meet user’s needs.

* User wants a good prediction quality,
* User wants the cheapest project as possible,
* User doesn’t want to wait for a result. At AXA France we must answer a prediction in less than 10 seconds for our use-cases.
* Safety is non-negotiable.
* We want the projects to be monitored. The idea is that if DATA DRIFT starts, this risk causing a production incident. If we are on a major project that can block AXA France, we are not going to wait for the incident to come.
* So, the last point, we must be able to re-train and redeploy quickly. If we have a major incident that blocks AXA France, we must be able to re-deploy as quickly as possible and not wait months to unblock the situation.

**Slide 25 Block Diagram**

So there, it’s the functional diagram of the workflow which explains how machines reads a French driving license. We also read the old French driving licenses, but for simplicity we do not display them here.

The idea is that we will receive an image or a PDF as an input. For the first step, we extract each PDF page as an image. On each image we will run a zoning algorithm to detect the fronts and backs. Now, let's take the front at the bottom, we will first straighten the image of the front with a specific algorithm. Then run another algorithm to zone the fields. Finally run an OCR to finely extract the text.

If we hosted an API that exposes this service in monolith mode. We would have several problems. The first is the response time, when we call this kind of API in monolith what will happen is that each algorithm will consume all the CPU resources of the machine. The consequence is that the pipeline will run sequentially, if we proceed like that, the response time is greater than 30 seconds.

Another problem is that during these 30 seconds: if another HTTP call arrives to the API; this call will have to wait for the end of the first processing before to start processing. So, you will tell me that we can add machines; but another problem and if we host this in monolith mode, we need machines with about 32 gigabytes of RAM, which is very expensive. That kind of monolith is also very slow to start. It's going to be very complicated to be able to scale quickly and to adapt infrastructure to user’s load.

The solution is to set up microservices. We will host each algorithm using different resources which allow us to finely configure the necessary RAM and CPU. We will be able to play the algorithms in parallel and to scale quickly. Response time will be quicker.

**Slide 26: Function**

How does it technically work? We use functions. Each algorithm is hosted on a function. This function listens to a Queue as input and as soon as it has free time, it will be able to pop the messages that come from the Queue and to process them.

For example, let’s take the SPLITTER which extracts each page of a PDF an image, the function will retrieve a message with a file identifier from the Queue, the files are stored in REDIS. With the identifier, we will fetch the document from REDIS to run our algorithms and for example if we have a PDF with 2 pages, we will extract 2 images then generate 2 identifiers; push the images into REDIS with them. We finally add 2 messages in the downstream QUEUE to be able to call the following service, in this case the Zoning of the French driver license.

**Slide 27: Scale**

The advantage of working like this. You can scale quickly and support peak loads. Imagine that suddenly, we receive 500 French driver licenses, the number of functions will increase to be able to unstack the messages more quickly. There is also a financial advantage because you only consume what you need.

**Slide 28: Architecture**

So that's the architecture diagram. If you find it complicated, that's normal because it's complicated.

I will not describe the entire schema, but in summary, it is an asynchronous architecture.

Look at the top left of the schema. The first POD will receive a document and push it straight into REDIS then, with the file identifier it will add a message in the upstream QUEUE of the splitter. Then the Splitter will retrieve the file, process it, etc. etc.

This architecture works for real time and in Batch mode.

**Slide 29: Difficulties**

**These projects are not without challenges.**

I will present you the 2 most important difficulties that we meet.

**Slide 30: Triptych**

First, there is an iron triangle of choice. A cursor to position between the prediction quality, the response time and the cost of the infrastructure in production.

If we want to have the best quality of prediction we can agree to pay more or lose some response time.

But, if you want to be able to respond more quickly, you must be able to accept either to pay more or to lose prediction quality.

Finally, if we want the project to cost less in production, we must either agree to extend the response processing time or reduce the quality of prediction.

**Slide 31: Find the differences?**

So, do you see the differences here?

The differences in these images, we cannot see them, however the AI ​​can!

I had first experience of this problem on our very first project. Pillow library was used to resize images, and which was no longer up to date. As we must always maintain libraires up to dates. As a good developer, I updated it. What happened next, is that predictions did not work anymore. So, I looked at the pictures. Images was identical to my eyes. It was impossible for me to understand what happened. And in fact, for AI images were very different. AI received different bits as an input. The consequence of this problem is that if in production we do not use the same libraries version, python version, and operating system than during training, it may not work.

Back to the demo. The AI ​​algorithm was trained with images resized to 200\*200 pixels via Pillow library.

Story: On the driver's license project, we changed all images resize from Pillow library to OpenCV, which allowed us to save a little less than 2 seconds in the global pipeline time.

In this demo I have also implemented a version where images in production are resized with OpenCV before prediction.

Watch, if I predict with images resized via Pillow and then with via OpenCV I don't always get the same results.

I carried out a test with 12500 images and we have around 500 different predictions.

**Slide 32: Sliding impact**

This only makes 3% prediction differences, but if you take a complex pipeline, where we chain algorithms with different AIs. 3% will propagate thought the pipeline, so at the end the prediction rate becomes very low.

**Slide 33: Contents part 3**

In this part we are going to give feedbacks based on different experiences. This allowed us to learn what NOT to do. To make it fun, we are going to open a pizzeria and we are going to organize to serve pizzas to our customers. We're going to put everything in place to make our pizzeria fails to deliver a good meal.

**Slide 34: Elizabeth**

To start, we will organize our team. In an AI project, there are 3 main phases. The first phase is the exploration phase, we will test if the project is feasible, then we move on to an industrialization phase and finally we have a “Deployment and Run” phase.

So here, I present to you Elisabeth, Elisabeth is a Data Scientist. We asked her to work alone on the exploration phase.

**Slide 35: Elizabeth**

So, Elisabeth is an excellent cook, we asked her to knead the pizza dough and spread it. She does it very well.

**Slide 36: Elizabeth**

Now, Elisabeth is done.

**Slide 37: Elisabeth - Hicham**

I present you, Hicham, his role is to distribute the ingredients on the pizza dough.

**Slide 38: Elisabeth – Hicham Hop**

Now Elisabeth, sends the pizza dough and the ingredients to Hicham. As you can see on the left, Elisabeth forgot to give the ham to Hicham.

**Slide 39: Hicham**

So now, Hicham, it's his turn to play and it's up to him to spread the ingredients on the pizza dough. However, he realizes that the ham has been forgotten and he expected that Elisabeth would have pre-cut the ingredients. Unfortunately, he does not have any knife.

Hicham as a good developer does everything he can. He takes the ingredients and spreads them as he can on the pizza.

**Slide 40: Hicham**

Good job for Hicham.

**Slide 41: Hisham - Khalid**

Now, I present you Khalid; it's our Ops. He is responsible for baking the pizza then to deliver it to the customers.

**Slide 42: Hisham – Khalid Hop**

Hop, Hicham give the pizza to Khalid.

**Slide 43: Khalid**

Damn, we forgot to give the instructions to Khalid. He doesn't know how to cook it (what temperature, and for how long). So, Khalid try to cook the pizza as he can.

**Slide 44: Khalid**

Unfortunately, now the pizza – without ham, and with badmy cut ingredients – is burnt. Good luck eating that!

**Slide 45: Team summary**

To sum up, here we have organized our team in silo, each role working separately, and also at the same time. It’s not a team mode.. If you want to wreck your project, this is the right way to organize.

You should know that the very first choice of libraries from the start of the project has a direct impact on production. Not working together, not worrying about production from the beginning of the project, are silver bullets to make project fail.

**Slide 46: Lilian**

I forgot to mention Lilian. Lilian came this morning to deliver some salt. It's one of the raw materials for our pizzas; but he was wrong; he delivered sugar instead! It will be a very good pizza.

**Slide 47: Table**

Now we're going to set up the table for our customers. So, if you want to ruin a project it can be a very good idea not to focus on the user needs.

First example, on the driving license project, we were asked to read the documents in less than 10 seconds. When we started working on it, we deploy on production environment the very first algorithm that Data Scientists gave to us, the Splitter of PDF. We quickly realized that the times of one extraction took around 15 to 30 seconds, which was very problematic. The Data Scientists did not notice it because they work on overpowered machines which perform the operation very quickly.

Second example, when we were a little further on the project. We realized that some readings were very slow and clogged the systems. By looking a little more in detail at that the documents. We found that, in general, there were almost unreadable document event for a human. For this kind of case, it would have been a good idea to go see the users and to ask them if it would not be a better idea to report immediately that the document is not clean enough. It would have been better for the system and final users which wait 40 seconds before getting very bad results.

**Slide 48: Aperitif.**

Let’s start the aperitif!

So here, we are finally going to serve the aperitif to our first customers. When you start a project, especially if you want to do agile and have a fast Time To Market. It may be a good idea to always test all classic code approaches rather than immediately carrying out Deep learning. To use the right for the right job.

Implementing a Deep Learning algorithm is more complicate and longer, also you need a lot of data and it's expensive.

First of all, the classic code is faster to set up. This makes it possible to iterate quickly and in general the processing time is faster and requires less CPU.

The classic code however will not allow us to be as good at prediction as Deep learning. Deep learning will make it possible to solve very complex problems in a much better way, but the iterations are longer.

In general, if you start with classic code and you can't achieve the desired results, it's not wasted time, because this classic code will allow you to save time on the labelling part. You can pre-label your data and you will also have a fallback system in case the AI ​​is not able to predict well.

**Slide 49 Aperitif 1**

So that's an example we had on the driving license project. To straighten a front license, we used 2 AI algorithm. We had a total of 5 algorithms. The problem is that these algorithms were slow and expensive in production. We were able to replace them with traditional algorithms with the same quality of prediction. The consequences: gain in response time and money.

**Slide 50 Aperitif 2**

We had the same problem with the email classification project. We had a deep learning algorithm to straighten out the documents. It wasn't doing it well and it took even longer than reading the document itself. So, we insisted on looking for algorithms and we managed to replace it without AI. Now execution is fast, less than 0.3 seconds and with very few resources consumed.

And who knows, if tomorrow we want to improve the performance and if we have the budget, we can train an AI.

**Slide 51 Aperitif 3**

Last example for our aperitif, for the text detection, we tried to replace these Custom Deep Learning algorithms by open-source algorithm available on internet. The only difference, Open-Source Algorithm are worst at detecting text on unreadable documents by humans.

If we had excluded these illegible documents from the start, we could have used this algorithm and thus gained TimeToMarket and made a lot of money.

**Slide 52: Salad**

Salad time. It may be a good idea to not industrialize your training code. If you do not industrialize your training code. When you start the industrialization phase, you will need to adapt some code then you will have to retrain the models.   
If you did not industrialize the training code and you are still on Jupyter Notebook without data versioning and code version. You will need again, almost the full time of the project to train the new models.

So now we can serve our pizza to our customers.

**Slide 53: Spaghetti**

Actually no, we were wrong, it's not a pizza; we served a plate of spaghetti. Code quality, unit testing today are extremely important for project maintenance. Don't do this, if you don't want your project to succeed.

For example, on the email classification project, we had a phase of extracting data from a lot of emails which took around 2 weeks of processing time. There were 2 possible choices, wait 2 weeks to validate a code change, or take 20 minutes to perform a unit test. With this example, Data Scientists understood the interest of Unit Tests. Thanks to this, we were able to iterate very quickly and now, our processing time is less than half a day.

**Slide 54: Code without testing**

Do you understand this code? Without unit test, without sample of inputs and outputs. I am personally unable to know what this function does.

**Slide 55: Salt**

A little pinch of salt on our pizza.

If you really want to fail your project, it might also be a good idea to work on separate GIT. A single GIT is simple and pragmatic, it allows you to work in team mode via PullRequest and to focus on production. This minimizes the number of manual actions and thus simplifies automation. So, if you don't want to work as a team and complicate your already complex project, separating the GITs is a good recipe for failure.

Separating the GITs is also great for having code mismatches between training and production.

**Slide 56: Dessert**

Icing on the cake for dessert!

Not monitoring your production. That's simple, you have a major project, you haven't monitored it. The model starts predicting garbage and you are unaware of it. You will encounter a major incident that will block your business and may go bankrupt.

**Slide 57: Coffee**

Small coffee

It may be a good idea to focus only on statistics. Statistics are a good indicator. However, by experience, there is nothing better than visualizing the data. With the data, we see exactly what happen in a complex pipeline. When there is a problem, you always must go back to the heart of the problem: The data.  
Also an enterprise project is not in a Kaggle competition, there are many ways to improve prediction without only DataScience.

**Slide 58 Addition**

Time to pay the bill!

You don't have to wait for someone to come to you to start paying attention if the project is too expensive. The planet has limited resources, so please, you should always think about consuming as few resources as possible.

If your project is too expensive, it may be stopped.

**Slide 59: Contents 4th part**

With these experiences, we have learnt. We can try to maximize our chances of success.

**Slide 60: Experimentation**

We will start from the exploration phase. What is important from the beginning in the organization of the project is to make all the actors work together toward a single objective: make the project work in production.

Focus on humans, make sure everyone share their own problems, make your everyone understands the problems of others, make sure everyone is working together on their resolution. This is really the heart of the success of a project. This is also why the title of this presentation is called MLOPS is a human adventure, it is really a culture of sharing associated with the fact of trusting each other.

In this exploration phase, will make it possible to estimate whether the project is feasible or not.

**Slide 61: Experiment Set up**

At AXA France we work on AI issues directly on production with production data. To secure this, we will ask Khalid our Ops to deploy a development environment on production network. Only few people will have access to this environment on AzureML.

**Slide 62: Elisabeth and Hicham experiment**

From the beginning Elisabeth and Hicham will work together from GIT and a DataLake. It will mainly be Elisabeth who will be active on the first phase but accompanied by Hicham who knows production difficulties.

**Slide 63: Experiment Annotation**

Quite quickly we will have to start a labelling phase. We're going to find Lilian again. It is very important to work in collaboration with the team that is going to label the data. Why? Data is your gold.

What we will do, is that we will first make small batches of data. Which will be used to refine the labelling rules. It is extremely important that the labels are carried out in the same way by everyone.

Imagine you have to Zone some cats on images. Imagine you have a cat that is on a table and its tail disappears under the table and becomes visible again on the other side. An annotator may Zone the cat without the piece of tail and another one may select the cat with the full tail.

With, this kind of data, the AI ​​will not be able to learn well. It is very important to choose common rules, to refine the labelling rules according to each specific case that we encounter.

Once the rules are well defined, the volume of labelling data can be increased.

**Slide 64: Model Experiment**

Once we have labelled the data, we can train a model. Then, to automate and save time; we will set up a continuous integration which will allow us to be able to replay the training automatically, almost industrially. The deliverable: it is not the model. It's the code in GIT plus the DATA versioned in a DataLake.

**Slide 65: Experimentation Deliverables**

We can see that we already have a well-spread pizza dough with all our ingredients and a knife ready to use, so, we really have everything we need.

At the end of this step, we must all together be able to estimate the metrics.

* Prediction quality
* Production infrastructure
* Response time
* We already have a versioned quasi-industrial training code and the versioned data, this is our deliverable

**Slide 66: Industrialization**

We will be able to enter to the industrialization phase where the goal will be to develop and bring into production APIs that host the models.

**Slide 68: Docker industrialization**

We are going to set up continuous integration which will allow us to build docker images. We retrieve models from a model registry then build our APIs inside a DOCKER image then push it into a docker registry.

**Slide 69: GitOps industrialization**

At AXA France, we deploy our docker images on OpenShift. We will ask our OPS to set up the development and staging environments. We deliver using GITOPS method. We use 4 GIT branches, one for the development environment, one for the staging, 2 others for pre-production and production.

This will allow us to be able to deploy our applications via GIT manipulation.

**Slide 70 Industrialization GitOps ml-cli**

Today we have tools that allow us to validate the proper functioning of the application in the environments. We check that we have the same quality of prediction as in training. We also check the response time and finely configure the sizing of the infrastructures.

Once the development environment is validated. We can deploy in staging.

To do this, we make a PullRequest on the staging branch.

**Slide 71: AzureML Production industrialization**

At that point we will enter to another phase of the project. We will ask our Ops to create a new AzureML training environment still on production. On this environment, we will only have read-only access.

**Slide 72: Big CI Industrialization**

We are going to set up a very BIG pipeline that allow us to retrain the whole pipeline automatically.

**Slide 73: GitOps Dev Industrialization**

We can now, via GIT manipulations, deploy on the development environment and then validate it.

**Slide 74 Industrialization GitOps Rec**

We deploy in staging and validate it.

**Slide 75 Industrialization Ready**

At this stage, we already have a pizza that is almost pre-cooked.

**Slide 76 GitOps industrialization Pre-production**

For safety, we do not have the rights to deliver in preproduction. Only the OPS can do it and they will deliver this pre-cooked pizza in pre-production. Then we validate the environment.

**Slide 77 Industrialization GitOps Production**

Then Khalid deploys to production. Yeah 😊

**Slide 78: Deployment & run**

The model API is deployed and follows the monitoring

**Slide79: Deployment & run monitoring**

For the moment, for monitoring Data Drift we are using Prometheus and Grafana

This monitoring allows us to easily link the metrics send some alerts. Alerts are sent to another team specialized in production.

Subsequently, we work on setting up the feedback loop and we will rather connect to this feedback loop to set up much finer metrics and alerting.

**Slide 80: End**

Today we have 2 ways to share code inside the same GIT.

We set up a directory named production. As in python everything is a module, the training code uses some modules called production to perform inference for scoring.

**Slide 81: End**

Another technique, we are testing to share code by making local python packages.

**Slide 82: Open source**

MLOPS is not centered on tools, however thanks to our projects, we have been able to set up tools and open source them. It is a pleasure to share them with you.

* The daily clean allows us to automatically shut down Kubernetes environment.
* Ecotag allows us to make labelling and comply with GDPR rules.
* ml-cli is our integration test tool.

**Slide 83: Conclusion**

To conclude, MLOPS is above all a culture of sharing.

Today we have projects that are at the top of the top of the complexity and if we do not put everything in place to help experts to work together.

To understand that the problem of one is the problem of the other.

It can't work.

A production ready project should also be a final focus.

The key to success comes in this sharing and working together.