MLOps = More Money

Calculating and communicating the value of MLOps to non-technical stakeholders

Who am I

I am Sam

I like startups

I co-founded DareData Engineering which is a consulting / outsourcing company specializing in data science and data engineering

I have mostly a technical background and in the last few years have been spending more time in coordinating and decision making roles

Table of contents

Part 1

Motivation, context and goal of this talk

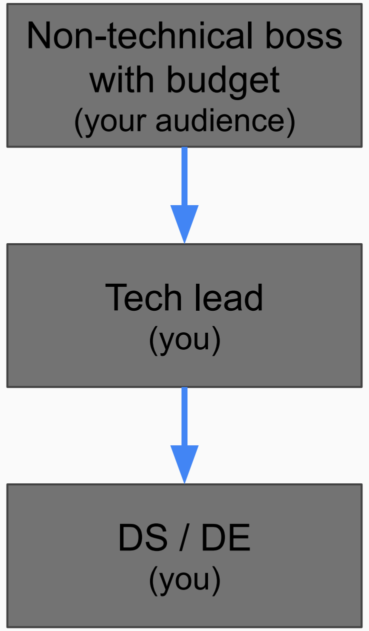
Let’s set the scene

Before continuing, we need to know: **who is the audience for this talk?**

The talked is based upon a blog post that is meant to be read by a non-technical person.

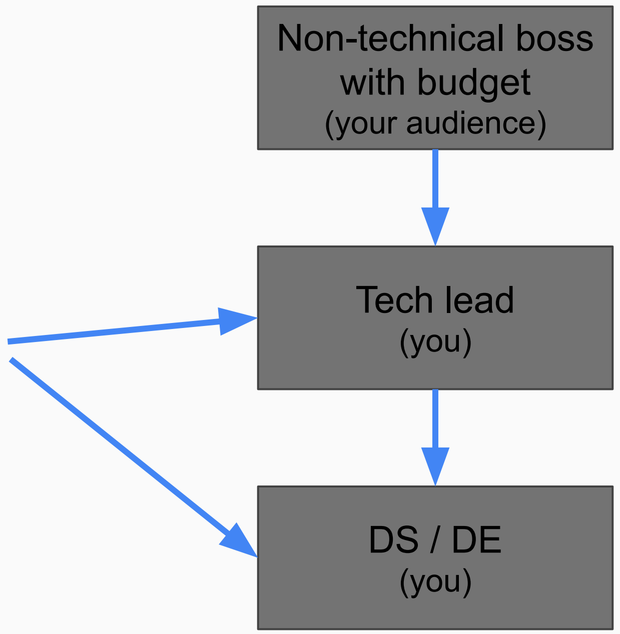
For this talk though, we are taking the perspective of **you**, the technical worker who is trying to communicate the return on investing in MLOps.

A hierarchy that is found almost everywhere



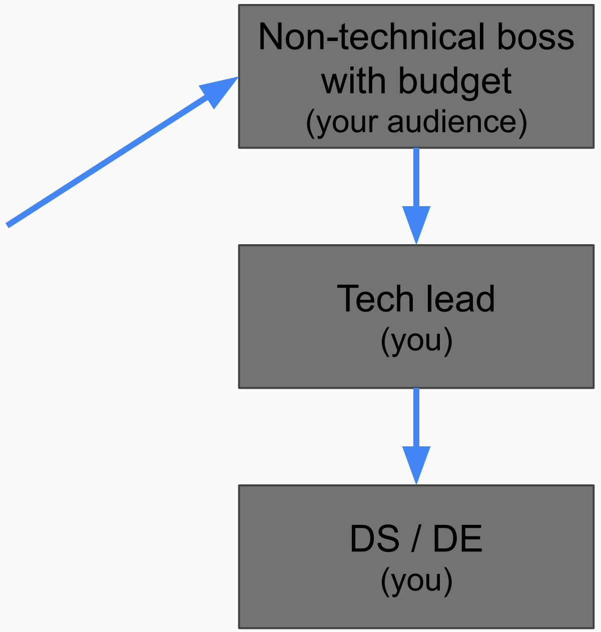
A hierarchy that is found almost everywhere

You are one of these people



A hierarchy that is found almost everywhere

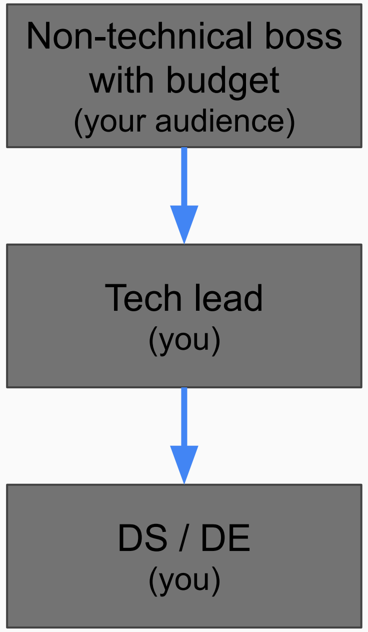
You need to convince this person of something technical



What is the convincing you need to do?

This talk assumes the following:

- You are a technical worker
- You are deploying ML models and / or developing ETLs
- You don't have proper automation around your production and dev environments
- There are non-technical stakeholders somewhere above you in the chain
- These non-technical stakeholders need to be convinced of an infrastructure overhaul in order to bring best practices to your work



The goal of this talk

Is to give you, the technical worker, a framework that you can use to approach non-technical stakeholders with the value of MLOps.

Part 2

Communication style with non-technical decision makers

Who is your audience?

Why do they need to be convinced?

- They are responsible for what is delivered at the end of the day
- They don’t have the technical expertise to make an informed decision
- They have probably been burned in the past with an infrastructure overhaul that went way over budget
- They are not seeing an emergency or anything else that would justify slowing down progress on the roadmap

Most importantly

Despite not knowing anything about the subject, they will be held responsible for it at the end of the day if something goes wrong.

That's a tough position to be in.

And the fact is

If they said “yes” to every single request for infrastructure changes that developers asked for, the org would suffer.

We are technical workers, we have a very narrow perspective compared to them and the stakes are higher for them.

Convincing this person is a HUGE job

If you explaining it looks like this, you've already lost

For success, some general guidelines

- It requires a delicate balance between firmness, respect, and empathy
- It must be done with a minimum (preferably zero) technical language
- It must be communicated in terms of money
- Business people understand money so if you can frame your argument in terms of money, you have a chance

Part 3

Types of costs incurred by not having proper MLOps

Let’s talk in terms of costs

Remember, we need to **speak in terms of costs** for everything that we do here.

So as a first step, let’s identify the different kinds of costs that are associated with developing and maintaining a production data ecosystem.

Types of costs

- One-off cost
- Monthly costs
  - Manual tasks
  - Mechanical time
  - Non-technical engineer hours
  - Technical engineer hours
- Employee turnover costs
- Time to market opportunity cost

One-off costs

These are essentially all costs with getting a project off the ground and there’s not much you can do about those.

They are mostly dictated by how well your organization is structured and how technically competent your people are.

They might be high but, if the project results in success, will ultimately represent a small portion of the budget.

One-off costs

- Any time you are proposing an infrastructure change, you are going to experience one-off costs and these could end up being high.
- These one-off costs are the primary reason that non-technical stakeholders are scared of infrastructure changes.
- If you are proposing an infrastructure change, it’s going to be VERY important to make sure that the one-off costs are paid down by the savings brought through improved automation.
- **Calculate these carefully and stash them away so that you can come back to them later to objectively show when a positive ROI should be achieved.**

So where does this ROI come from?

It comes from savings in the following areas

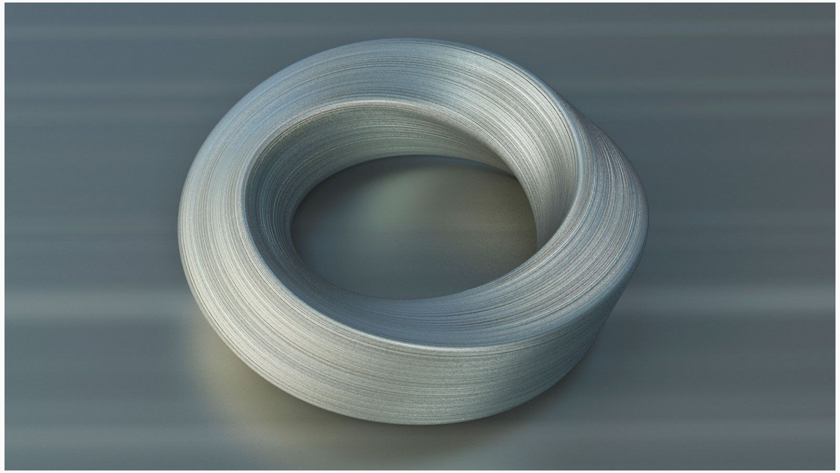
- ~~One off costs~~
- **Monthly costs**
  - **Manual tasks**
  - **Mechanical time**
  - **Non-technical engineer hours**
  - **Technical engineer hours**
- **Employee turnover costs**
- **Time to market opportunity cost**

Notice how the one-off costs are only **one category of cost**. Simply communicating this can help non-technical stakeholders understand the situation.

Monthly costs

Monthly costs are **terrifying**.

Much like this mobius strip they have no end in sight.



Monthly costs

Here's a simple and familiar example to help communicate the importance of seemingly small monthly improvements:

We all know the difference between a flat $100 / year investment for unlimited users and a $10 / person / month subscription fee. When you're getting started you might only have a few people that you need to pay this fee for. It's very tempting to go for the $10 / user / month.

Even if you only keep a single user, paying the $100 up-front and you've already broken even in less than a year. Considering a few years out after you've grown the team and all of a sudden the monthly subscription is costing you several multiples more.

Investing in MLOps is choosing to pay the $100 / year flat fee because of this category of monthly costs.

Let’s check out the monthly costs

Okay now that you’ve set up the analogy, it’s time to get concrete on what these costs actually are in terms that can be understood by a non-technical stakeholder.

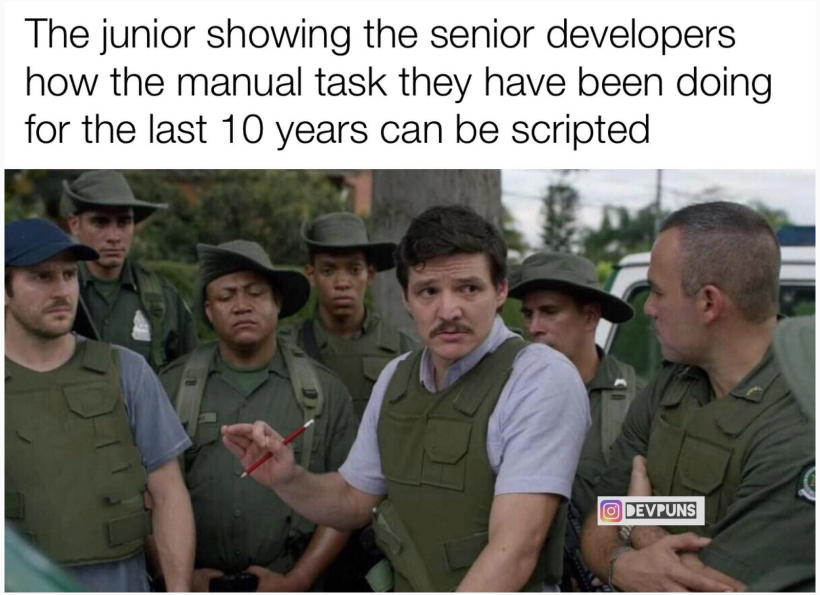
Types of monthly costs

Monthly cost types - manual work

- When a technical worker cannot automate part of their job, they must do manual work.
- This manual work is the death of the soul of anybody that works with code for a living.
- Plus they interrupt your concentration. Even seemingly small 20 minute per week tasks can easily add up if you are not given the tools to automate them away.

Monthly cost types - manual work

- Before you know it, you are spending a part-time job amount of time executing manual tasks.
- Multiply this across a team and the costs explode.
- Since people are usually your biggest cost, you should be able to calculate, using your salary, how much manual tasks are costing your company.


The junior showing the senior developers how the manual task they have been doing for the last 10 years can be scripted
DEVPUNS

Monthly cost types - mechanical time

When a DS or DE hits a button to process some data or train a model, it needs to be fast.

With DE this is a bit different than app development because the operations themselves can take much longer and require more computing power.

So it actually becomes really important for the data processing operations to be FAST.

Monthly cost types - mechanical time

You might think: well if they are slow, the engineer can just work on something else in the meantime, right?

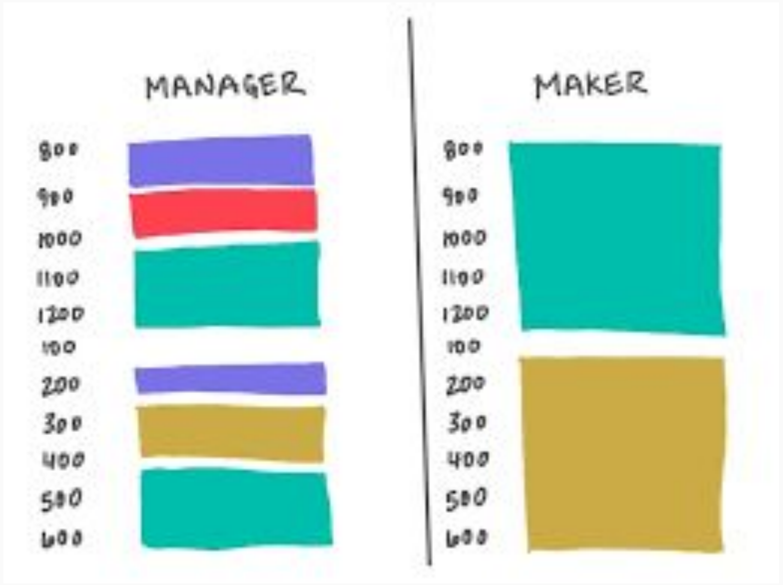
This is true to some extent but breaks down pretty quickly.

Monthly cost types - mechanical time

There is a difference between a maker's schedule and a manager's schedule.

Engineers and scientists require longer amounts of unbroken time to do their work. The longer they have to wait for something to run, the more it breaks their train of thought.

So we need to take into account how long it takes for an engineer to run an operation.



Monthly cost types - non-technical engineer hours

- No matter how technical of a worker you are, if you are working in industry you are having meetings.
- One of the mantras of DevOps (and thus MLOps) is self-service. **The technical worker should own their own deployments.**
- When the technical worker owns their deployments, they have less meetings with other teams to achieve the same outcome. This is good.
- **The less MLOps you have, the more meetings you have.** The more meetings you have, the less your technical workers can ship.

Monthly cost types - technical engineer hours

This one innocent looking section in the middle of this blog post is what many people think is the entire job description.

This section is a question of how many technical tasks can a technical worker complete given that they can get an unbroken stream of time to work on it.

Monthly cost types - technical engineer hours

This essentially boils down to the feature completeness of your infrastructure. The more features it has that you need, the fewer things that your engineers will need to develop themselves.

Example: Setting up and maintaining a Spark cluster in-house and on-prem for the first time can take months whereas getting access to the same functionality through DataBricks can be achieved in an afternoon.

Other types of recurring costs

Recurring cost - employee turnover

Fact: the worse your tooling, the more employee turnover you have.

Many technical workers are motivated by learning. Others are more outcome oriented and like to deliver.

In both cases, an absence of MLOps may cause your Data Scientists / Data Engineers to become bored and frustrated with their lack of ability to ship in a modern way.

Recurring cost - employee turnover

It is well studied that the turnover costs associated with technical workers can result in non-trivial delays, human hours, and actual costs.

Depending on the culture of your organisation, you will only have a certain amount of control over this. **Tooling should be one element of control over this that you retain.**

Recurring cost - time to market opportunity cost

The longer a project sits in the pipeline the higher your opportunity cost.

If you are a large organisation with many customers, a single ML model might represent 6 or 7 figures in yearly revenue.

There are two factors that you should take into account with respect to your technical workers and they are:

1. How long does it take to develop and release a model
2. How many projects can a single technical worker maintain in production

Recurring cost - time to market opportunity cost

1. How long does it take to develop and release a model
2. How many projects can a single technical worker maintain in production

(1) is obvious but (2) generally gets less consideration.

Each time a model or dashboard is put into production, everything that went into creating it must be maintained.

The models must be re-trained. The ETLs must be re-run on an hourly, daily, weekly, or monthly basis.

The less MLOps that you have, the more time the technical worker must spend executing manual tasks that they should be able to automate.

Recurring cost - time to market opportunity cost

We regularly find that improved tooling can literally double (or more) the number of projects a single technical worker can maintain in production.

So yeah, you can double the ROI on your technical people's salaries with the right tooling.

So what’s the ask?

Assuming you've made a compelling argument

Your non-technical is going to respond with "Okay, I'm convinced. So exactly what are you asking for?"

They are going to need to sign off on a plan as well as some costs. You'll need to pick your battles here very carefully. If you ask for something that's not necessary, they sign off on it, then it comes back to bite them in the future it's gonna be rough times.

The following slides are the types* of tech that you'll need to ask for and why.

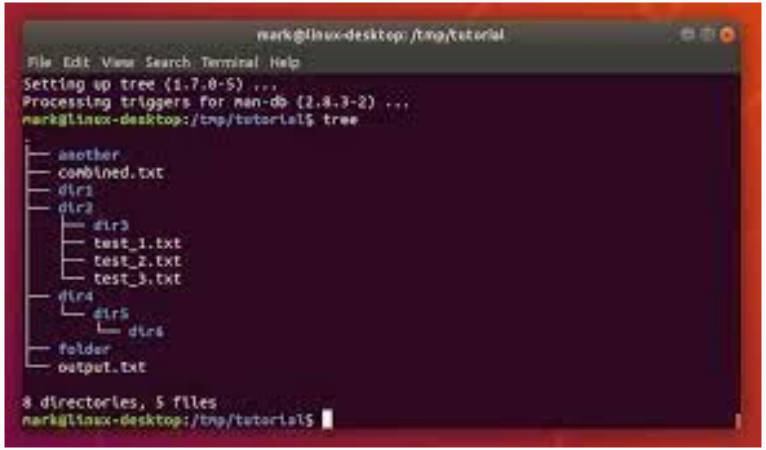
** We say "types" because there are multiple tools in each category that can achieve the same effect. Giving yourself and your non-technical stakeholder some flexibility in these areas will show good will as well as increasing the chances that you find the right stack without being too dogmatic.*

The tech - a linux command line

Everything must be runnable via a command. A command is a string of characters that represents a button you can push.

Every button that you push on a computer should have a corresponding command.

If it does not have a corresponding command it means you are locked into something called a graphical user interface (GUI).


mark@linux-desktop: /tmp/tutorial
File Edit View Search Terminal Help

```
Setting up tree (1.7.0-5) ...
Processing triggers for man-db (2.8.3-2) ...
mark@linux-desktop:/tmp/tutorial$ tree
.
├── another
├── combined.txt
├── dir1
├── dir2
│   ├── dir3
│   ├── test_1.txt
│   ├── test_2.txt
│   └── test_3.txt
├── dir4
│   └── dir5
│       └── dir6
├── folder
└── output.txt

8 directories, 5 files
mark@linux-desktop:/tmp/tutorial$ 
```

The tech - a linux command line

A GUI requires a human operating a mouse and clicking on it to run. If something requires a human clicking a mouse then you will never be able to automate it away.

If you can run it as a command then it can be included in things called CI/CD pipelines which are the primary method of automation in the world of DevOps and MLOps.

Command lines available in a linux or unix environment are the most feature complete, well-studied, and robust so this is what DareData recommends. If you are a windows shop then this task is made more difficult but PowerShell does exist so there is a path.

The tech - model and data versioning

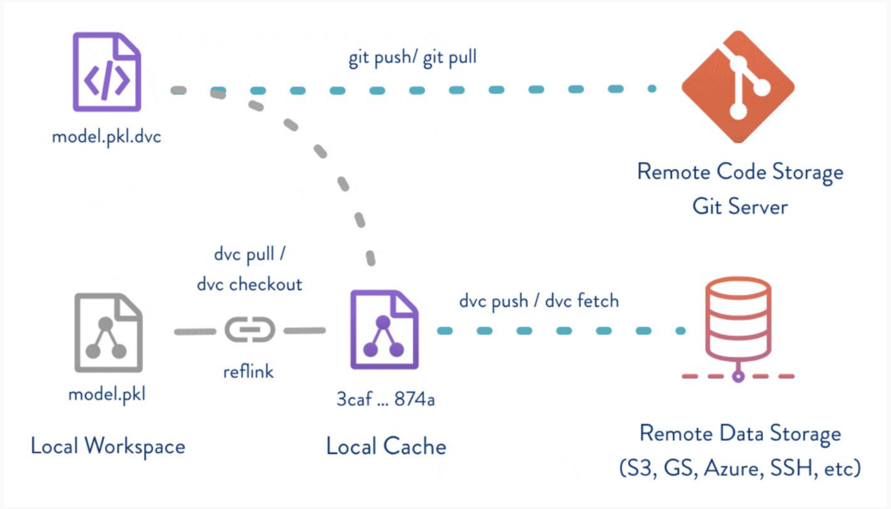
Unlike most other things in MLOps this one does not have a direct parallel in traditional DevOps.

This is because data and models have different characteristics than anything in the application world because of their size and nature.

The tech - model and data versioning

There is one set of tools used to track versions of your code but you need a different set of tools to track versions of your data and models.

Tools like MLFlow, DVC, and github lfs tackle these needs.

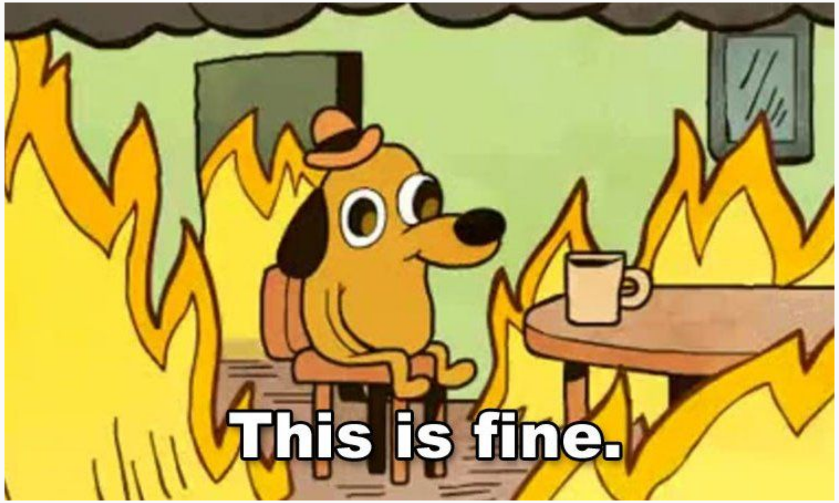


The tech - ETL automation and scheduling

That's because with ETLs, there are just so many things that can go wrong.

The more that can go wrong, the more time you spend doing manual work to fix it.

Before you know it, you're spending all of your time putting out fires without knowing you've descended into madness.



The tech - ETL automation and scheduling

You can't just run a complex ETL on a simple schedule.

- You need to have it retry if something goes wrong.
- You need access to organised logs to debug it.
- You may need to coordinate data flow between multiple systems with arbitrary triggers.
- You may need some workflows to depend on others.
- You may need to perform arbitrary logic in the middle of it which requires the flexibility of an open source programming language.

There's enough needs that usage of tools like digdag or airflow is absolutely essential. Without it you significantly increase the maintenance hours of a deployment, resulting in soul-destroying manual tasks.

The tech - CI/CD pipelines

What we are trying to achieve at the end of the day is the ability for your technical workers to "push" new code to a repo and have it automatically checked for errors and deployed to production.

CI/CD pipelines are how you do this.

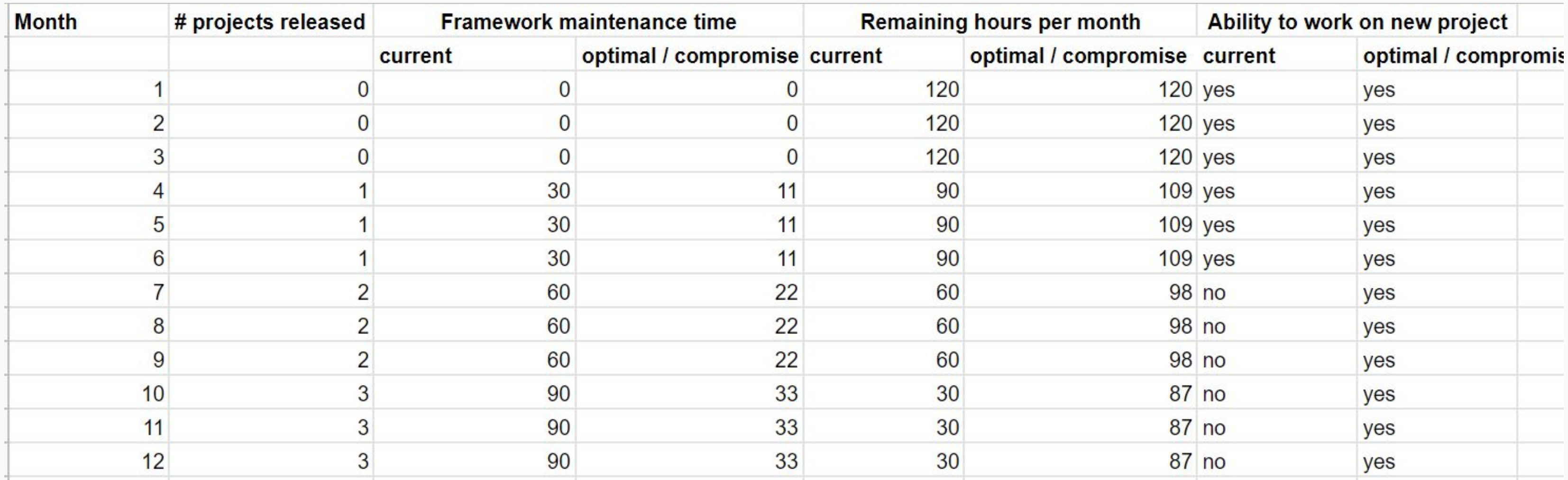
We recommend using gitlab for your code versioning as well as CI/CD since it was built from day one with DevOps in mind.

Make it real

With numbers

We’ve got a spreadsheet here that you can use as inspiration for putting this into real numbers for your non-technical stakeholders.

The end goal is to accurately simulate how well you will be able to scale your machine learning output.



Month	# projects released	Framework maintenance time		Remaining hours per month		Ability to work on new project	
		current	optimal / compromise	current	optimal / compromise	current	optimal / compromis
1	0	0	0	120	120	yes	yes
2	0	0	0	120	120	yes	yes
3	0	0	0	120	120	yes	yes
4	1	30	11	90	109	yes	yes
5	1	30	11	90	109	yes	yes
6	1	30	11	90	109	yes	yes
7	2	60	22	60	98	no	yes
8	2	60	22	60	98	no	yes
9	2	60	22	60	98	no	yes
10	3	90	33	30	87	no	yes
11	3	90	33	30	87	no	yes
12	3	90	33	30	87	no	yes

Questions?