optimyze

professional service FAQ - January 2019

Q: What is the general flow of a project?

A: The general flow of a project with optimyze:

1. We discuss your cloud footprint and your workload with you to understand what your business does, what computation you perform, and what your constraints are. This usually involves inspecting your cloud bill to find the "big-ticket items".
2. We agree with you on some stream metric that is a good representative of your business and your cloud usage -- such as "number of client requests" or something similar. In the end, we are trying to help you optimize your margins, and this metric will be used to measure our success (and remove seasonal fluctuations etc.). We measure and record your "cloud spend pro-rated to this metric".
3. We begin gathering performance and usage data from your cloud footprint to identify which areas of your code and your infrastructure will provide the "most bang for the buck".
4. We analyze your code, your infrastructure, you base images etc. and generate a "menu of optimization options".
5. We present the menu of optimization options to you - including estimates of complexity and estimates of cost reduction. You and your engineers decide which of these optimizations you are comfortable with - some will be more intrusive than others, and nobody can judge feasibility, risk, and convenience better than you.
6. You choose elements from the menu, and we work in conjunction with your team to implement these optimizations.

7. Once the optimizations are implemented, we re-measure your “cloud spend pro-rated to this metric”. Our remuneration is based on this - e.g. we are paid 50% of the savings we realize for you over the next 24 months; if we do not manage to save you money and enhance your margins, our work was free.

Our goal is to find the best alignment between your interests and our interests - a true win-win situation.

Q: What is the minimum cloud spend for a company to become a client?

A: Our target market are companies with significant cloud spend - ideally in excess of 50k$ / month. This depends on your growth perspective: We try to pro-rate the savings we produce to some "volume measure" (cost-per-customer, cost-per-transaction etc.) so that demand fluctuations on your side are evened out (seasonal fluctuations being the biggest driver). If your cloud load is growing, a minimum spend of 50k$ / month is not a hard limit.

Q: What other qualifications are important for a potential client?

A: The Cloud workload should almost entirely be based on Linux or FreeBSD; our optimization expertise does not extend to Windows setups and heavy use of third-party closed-source components.

Q: What level of system access needs to be provided?

A: The amount of system access depends on the granularity and nature of existing monitoring infrastructure - and on what level of access you are comfortable with.

From past experience, we need the following to be most effective:

Profiling and optimizing CPU / IO / Network usage.

For profiling and optimizing CPU / IO / Network usage, we usually work with the following:

- Read access to production monitoring data:
    - Read access to monitoring data from all production systems (equivalent data to what a Prometheus node_exporter or the DataDog SystemCheck exports). This allows us to diagnose CPU, RAM, IO and network load on the different systems without needing local access. For active Prometheus users, the easiest method is usually just providing us with a snapshot of the raw Prometheus time series database, or direct access to the stream metrics.
- Read access to output from the "perf" command with some command line flags from running production systems (to obtain stack trace information under real-world load); access to the output from a few other diagnostic commands from the production systems.
    - Depending on the client setup and languages deployed, obtaining stack traces may require slight modifications to the client setup - particularly for JVM-based languages, Node.js, and Python.
- Read access to package lists from the Linux production images. This is needed to make sense of the stack traces (if no debug packages are on the production images).
- Access to the code running in production:

  - Read access to the executables on the production image; copies of the production images/containers are usually the easiest.
  - Read access to the custom code the client is deploying on his production image - ideally the ability to read the client git repository (or other source control system).
  - If applicable, read access to the symbols for the deployed executables.
- Interactive root access to one exemplary "worker" for each type of "worker" in the distributed system. This does not need to be a production machine, but should ideally run under some workload that is representative of the production workloads. We use this to perform fine-grained profiling and experimentation. **In essence, we need the ability to run workloads that closely resemble production workloads** (albeit at smaller scale).

If monitoring infrastructure such as Prometheus is not in place, several possibilities exist: Granting us more access so we can gather data without a monitoring infrastructure, working with us to help you implement better monitoring, or finding other ways of providing the data that does not involve more access.

Storage / bandwidth / compression optimization:

For quite a few customers, the cost of S3 or GCS or other cloud storage solutions (and bandwidth in accessing it) can be significant. We help you choose optimal compression algorithms and settings. To do this, we usually need:

- Representative datasets for the sort of data that is stored.
- Information about expected number of compression / decompression / network transfers / expected storage lifetime of the data.
- Information about the precise setup of Databases, ElasticSearch clusters etc.

We then use our proprietary methods to find the setup that minimizes your cost.

Q: How much time / attention from client engineers does optimyze need?

A: We try to be as lightweight as possible. In addition to the data / access described above, we normally need some initial guidance to understand the bigger picture. For this, a 60 minute kickoff meeting where we get an overview of the architecture, where to find additional documentation and code etc. is usually good start. We then gather all the data, start inspecting the code, and try to understand your setup and requirements.

We will normally develop follow-up questions - both on the technical setup, but also on business constraints. Example follow-up questions are:

- “What latency is acceptable for operation XYZ” ?
- “We see this cluster of machines here that looks overprovisioned; is this in anticipation of work spikes?”

We try to determine both the constraints and the design rationale of the existing deployment, and to ensure that we understand the constraints under which everything operates.

In order to clarify questions, we try to schedule another 60-90 minute meeting about 48h after the kick-off.

As our understanding of the infrastructure deepens, another meeting or two may become necessary - we try hard to work from documentation, code, and monitoring data, but it is common that important details are not available in writing.

Our goal is to work mostly without imposing on the customer's time; the mode of operation is extended analysis of code / artifacts, and sparingly (and in batched fashion) questions for the existing engineers - either via Email, or Meetings, depending on client preference.

If your engineering teams already use Slack or IRC to communicate, a dedicated low-traffic channel for questions and discussion is also extremely helpful, and our preferred mode of operation.

Q: How are optimization proposals implemented?

A: We strive to fit into your existing workflow and processes. For optimization measures that are not very intrusive, we aim to minimize the amount of work you have to do.

In an ideal scenario, after an optimization proposal has been agreed upon, we work to provide you with “pull requests” that your engineers need to review, approve and apply: Changes to Dockerfiles, Terraform scripts, installation scripts or the actual code bases that implement the proposed optimizations. We then work through your regular change management processes to deployment.

For non-intrusive changes, going through a regular commit / canary / deployment process should work fine.

Depending on the intrusiveness of the proposed optimization, some amount of help from you and your team is required - this is particularly true for migration of storage infrastructure (and stored legacy data) to a new, more cost-effective compression algorithm, or other optimizations that require re-processing of stored data or changes in the way that data is stored.

We normally give priority to the less-intrusive optimizations - the less friction for you, the better for all parties.

If larger changes in architecture are decided upon, we work with your team as “one of the engineers” to help the execution. Details of this are decided on a case-by-case basis jointly with you.

Q: What are the timelines for data gathering, analysis, and execution?

A: The timeline for data gathering, analysis, and execution is very dependent on your infrastructure. If you have a highly homogenous computational load (many copies of essentially the same service running in parallel) the effort in understanding the infrastructure will be lower than highly complex systems of interacting services.

We will work with you to focus to the areas where you suspects (or know) most of your cloud spend is.

As a rule of thumb, we try to obtain all the required data within the first two weeks of an engagement. We strive to achieve a good overview of your infrastructure in 4 weeks.

The detailed analysis work then spans another 2 weeks while we prepare a menu of optimization suggestions along with savings estimates per option.

This means that after a maximum of 6 weeks, the options for optimization can be proposed and discussed.

Please note that these timelines are subject to quite some variance depending on the complexity of your setup.

Similarly, the execution phase varies widely depending on the complexity and intrusiveness of the agreed changes. Simple changes can often be deployed within 2 weeks (bringing total time between “start” and “first cost reduction” to about 8 weeks), but depending on the amount of engineering required, the expected cost savings, and the criticality of the changes, another month or two are not out of the question. All in all, a project will take approximately 3 months end-to-end, with the option to spend more time if more savings can be realized.