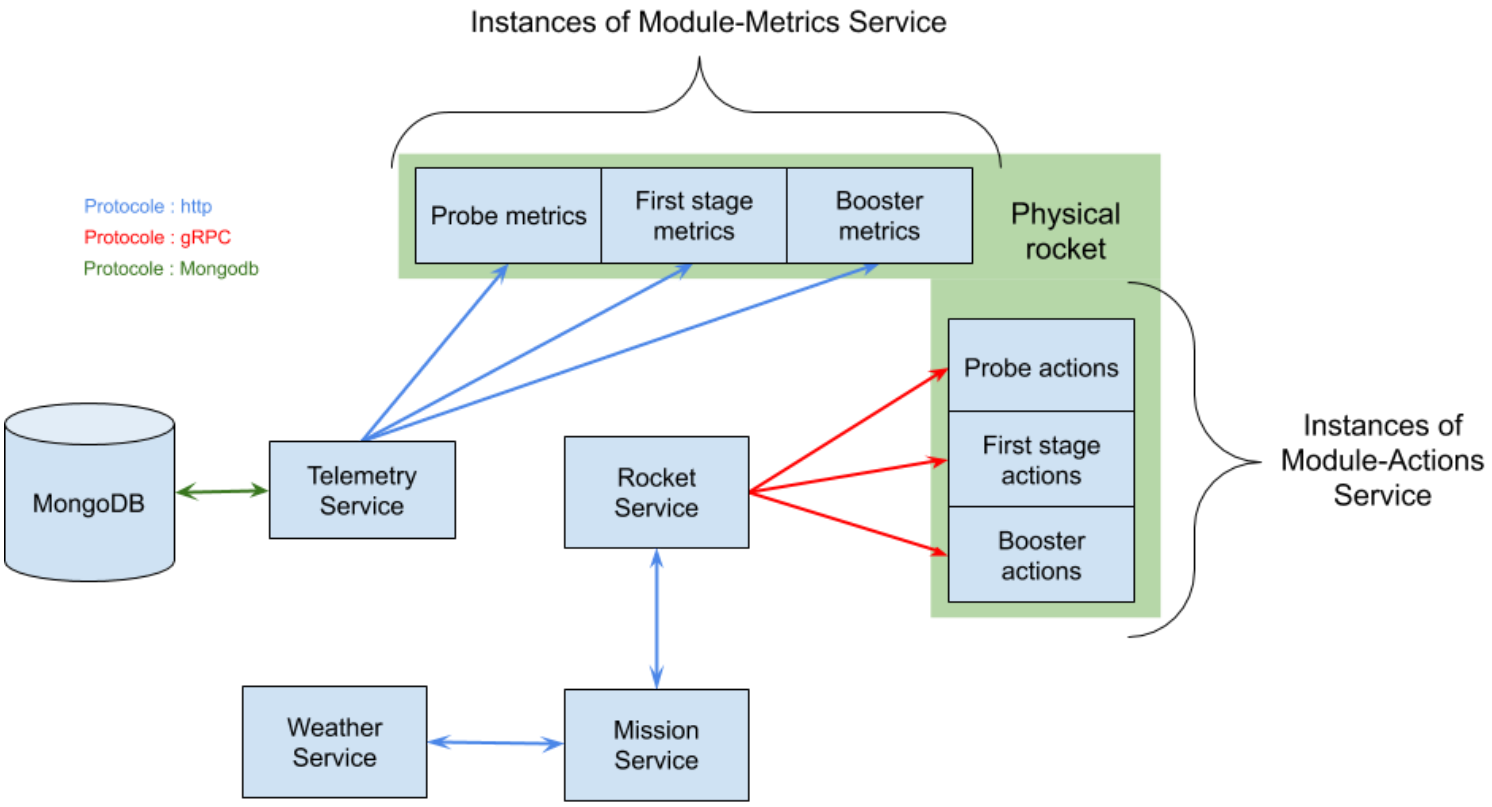
Blue Origin X

Team B

Masia Sylvain
Montoya Damien
Peres Richard
Rigaut François

2020 – 2021

Architecture



Responsibility

After a long thought, we came to the above architecture for this MVP. It is composed of 4 ground based services : telemetry, rocket, weather and mission; a database and two services in every rocket part (Booster, 1st Stage and Payload). Each of the ground based services will be contacted by the different actors of our system. Thus, the mission commander will use the mission service to perform the go / no-go poll. The chief rocket department and the chief payload department will both use the rocket service to do their actions. The launch weather officer will be able to check the weather and answer to the service mission thanks to the weather service. Finally, the telemetry officer will contact the telemetry service to get the metrics of each rocket part (also called modules) stored.

Each rocket module contains two services. The first one is dealing with every action that the module can encounter. For example the booster module has a method to allow the chief rocket department to manually detach this module from the rest of the rocket. Every module also provides a method to explode in case of problem, and a method to adjust the speed.... The second service in the module is about the metrics. This last one will collect every data available about it's sensors, such as the speed of the module, the altitude.... Those data will be stored in local and the telemetry service will contact the module to get them and persist them.

Explanation

Telemetry service implementation

When designing the architecture we thought that adding a dedicated service to manage the telemetry was a good idea because it could allow us to delegate the metrics storage part to a single service.

For now, we use HTTP routes to fetch the telemetry data (i.e. the telemetry service has to explicitly request the data from each module-metrics). We did this because, in the future, we are planning on adding a Prometheus broker to continually and autonomously fetch the data from every module with integrated persistence, load-balancing and protocol encoding. Also, a UI with Grafana would easily be added to this part for better visualization of the telemetry data.

Thus, as Sockets would be too complex to implement just to switch on another system afterwards, we stuck to simple HTTP routes so that we can convert afterwards.

Rocket management implementation

Considering we had to track the metrics of each part of the rocket, we've decided to split it into modules and to keep our rocket service as a facade to interact with the different parts of the rocket as one unit.

Regarding these rocket modules, we chose to split each module in two services for two reasons. First of all, splitting the module into a telemetry and an action part has allowed to split the responsibilities of each service. The second argument is that if a service crashes for any reason, the other one will possibly be still running. For example, if the telemetry service is no longer responding, the chief rocket departement may be always able to explode the concerned module to prevent any bigger problem.

To communicate between the rocket service and each rocket module, we are using the gRPC protocol because we are not dealing with any data therefore we only send actions to be done. Furthermore, by using gRPC we are adding more security with a contract enforcement.

Constraints and limits

First of all, in our actual system we are limited to use a single rocket. In fact, the rocket and the telemetry services are not designed to handle multiple rockets.

Secondly, as we said previously in the telemetry service implementation part, we are using http to fetch the telemetry data, instead of having an implementation which allows data streaming, even though the modules continuously send data to the telemetry service (for now).