author: Michal Stefaniuk, Software Engineer at Devapo (www.devapo.io)

Simple business case of an Azure platform for quick Camunda enablement

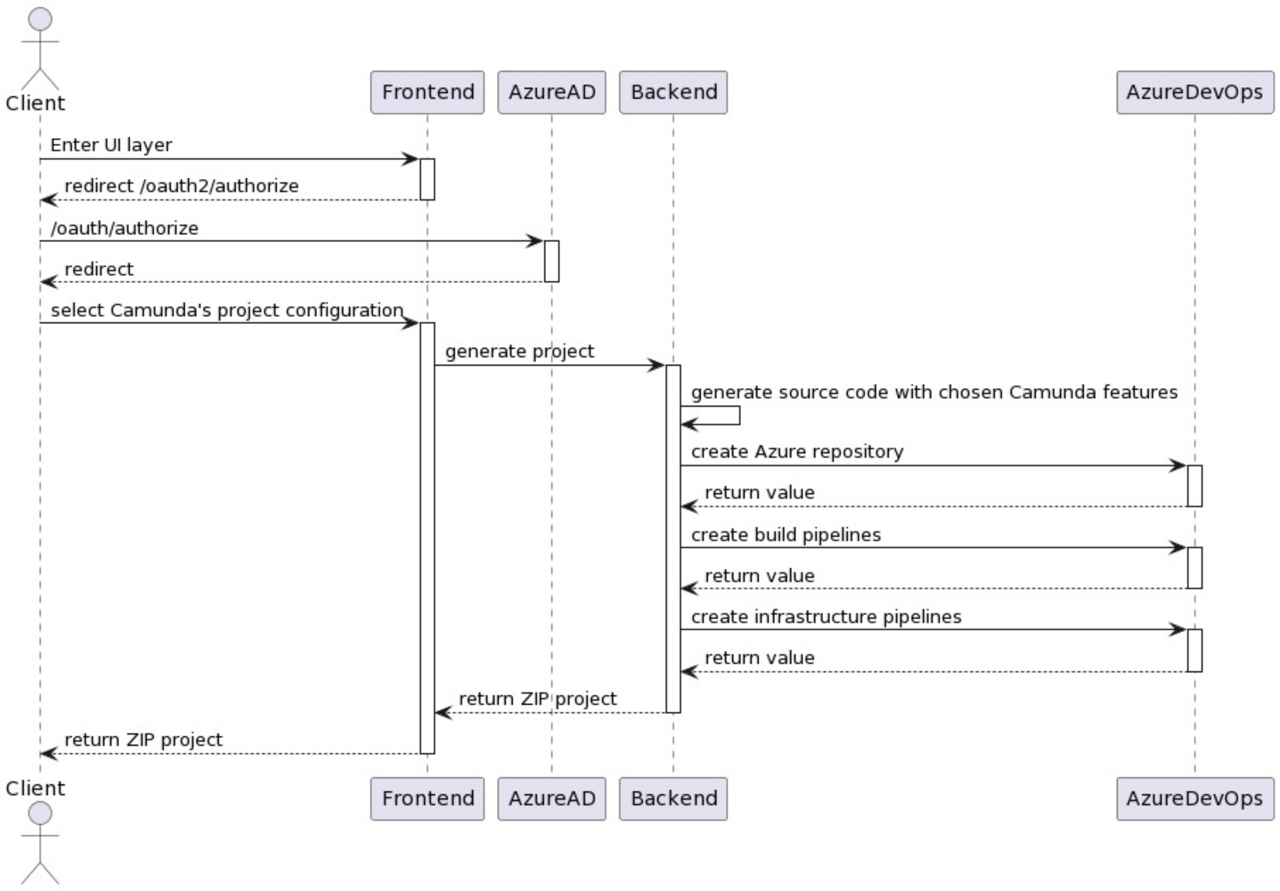
Nowadays many companies from the banking sector are moving towards business process automation. Where there is a process, a tool like Camunda could be introduced. Another approach which seems to be very trendy these days would be migrating to the cloud. These two solutions combined
present a huge scale of possibilities, but as they say, with great power comes great responsibility.

Introducing new tools into your architecture sometimes might be really tricky, especially if your project is already live in the production environment. Moving your product to the cloud might be even more challenging, as this usually requires
major changes in the existing implementation.

Here at Devapo, as an official Camunda partner, for one of our clients from the banking sector we develop a solution, which intends to enable developer teams to quickly move to Azure cloud with Camunda. In this article I would like to present this concept on its technical level as well its business implications.

How is the platform implemented? The concept's walkthrough

The goal of the platform was to allow developer teams to quickly move with their Camunda products to Azure cloud. The platform is mainly focused on automating the process of creating Azure repository, pipelines, including build and infrastructure ones thorugh Azure DevOps REST API.
Its user interface allows to simply select the configuration and particular features that the client is interested in. This concept bridges the gaps between integrations and bringing custom Camunda features to production.



Once the user accesses the fronted of the platform, authentication and authorization against the Azure AD service begins. After a successful authorization , the user is prompted with a form that allows to select the project configuration. Among those options, the client can choose particular dependencies, their versions, custom features and the environment's setup. The platform
allows generating only the projects with embedded edition of Camunda, however the development effort is reduced significantly. The custom features that the platform provides are for instance

- Camunda Custom Identity service,
- Camunda with LDAP,
- Camunda with Azure AD authentication and authorization,
- Camunda with Azure SQL Database and automated database migrations,
- Camunda event-driven by Azure Functions,
- Camunda with secrets stored in Azure Key Vault,
- Camunda with process definitions stored in Azure Blob,
- Camunda delegates with Azure Service Bus connectivity

and many others.

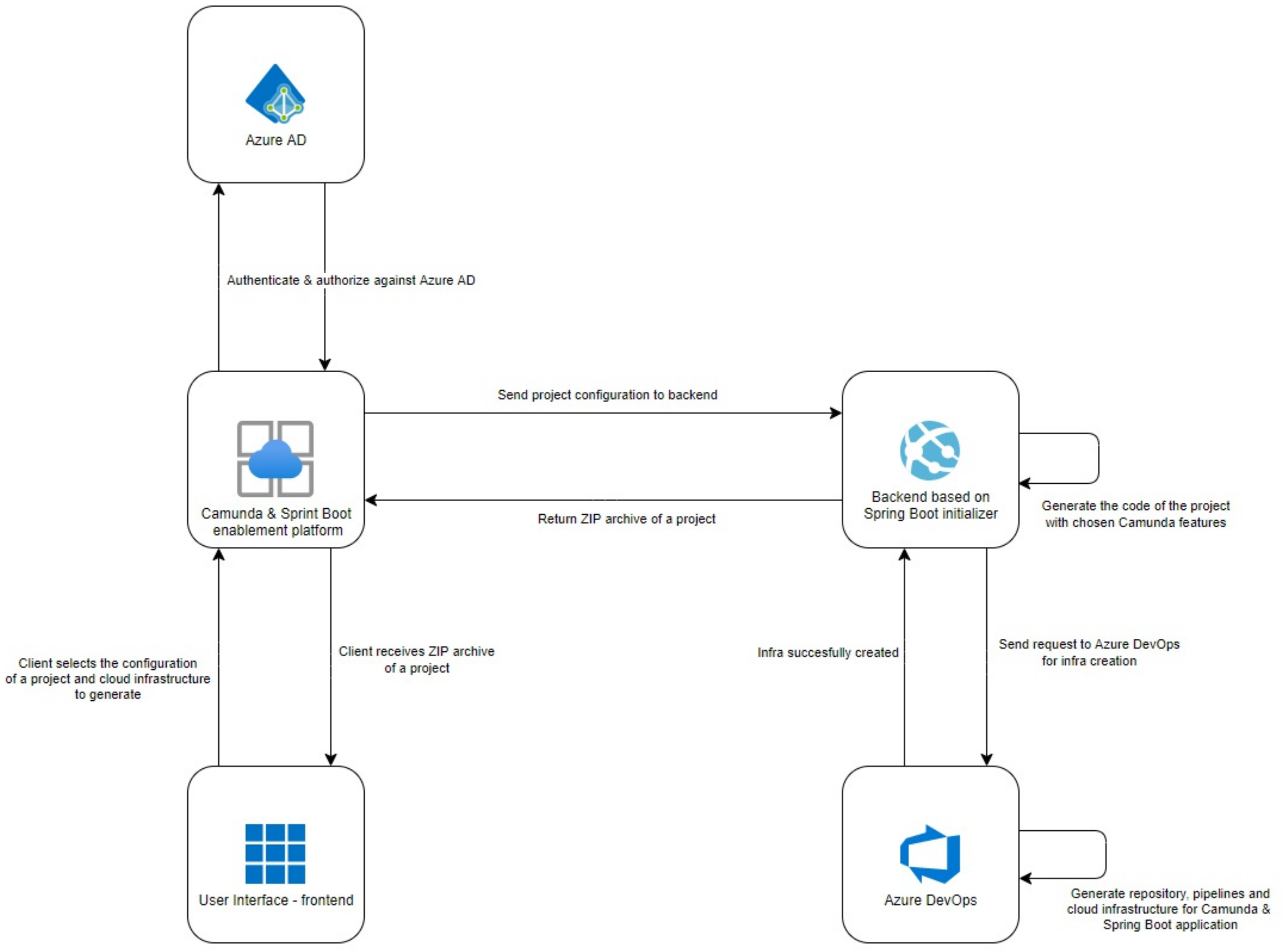
The backend does the trick

Once the platform's client chooses the particular configuration, the frontend layer communicates with the backend through a http POST, that contains in its the body all the configuration that the client chose. The backend layer is very simple - it is a Spring Boot application that implements the official Spring initializr project (https://github.com/spring-io/initializr).

From a technical point of view, it contains three packages.

- generator - contains java Beans that implement so called `contributor` classes provided by Spring intializer and they describe the way that the project should be generated regarding the configuration in the HTTP Post request. It is responsible to utilize the logic on which of the custom Camunda features should be included in the project generation.
- metadata - this package holds the template of the project files which will be used in the project generation (like *java files, *bpmn files or any other static resources necessary for a Spring Boot & Camunda application). This package is also responsible for generation of Spring Boot configuration and it will be used by the contributor classes called in `generator` package.
- web - the last package is the one that holds a bunch of controllers and is responsible for receiving HTTP requests. It also delegates the project generation to the particular contributor class of the `generator` package after parsing HTTP requests body content.

Before the zip file of the freshly generated project can be returned by the backend, it does one more thing - through the REST template it communicates with the Azure DevOps API and it creates the Azure repository with the chosen Spring Boot and Camunda application and it creates build and infrastructure pipelines which are ready to be used. This part is extremely important - we need to notice that at this point the client is not only provided with a Spring Boot & Camunda application with custom features, but also with a pipelines, that are ready to be run, so the effort that usually needs to take place when creating cloud resources and whole deployment infrastructure is gone. This approach is called IAAC (infrastructure as a code) and is a huge enablement for the developer teams. Once this part is finished, the client ends up with the necessary cloud infrastructure and a Spring Boot & Camunda application which is already in the Azure repository, but in addition a client also gets to download a ZIP archive of the project.


Azure AD
Authenticate & authorize against Azure AD
Camunda & Sprint Boot enablement platform
Send project configuration to backend
Return ZIP archive of a project
Backend based on Spring Boot initializer
Generate the code of the project with chosen Camunda features
Client selects the configuration of a project and cloud infrastructure to generate
Client receives ZIP archive of a project
Infra succesfully created
Send request to Azure DevOps for infra creation
User Interface - frontend
Azure DevOps
Generate repository, pipelines and cloud infrastructure for Camunda & Spring Boot application

Example scenario

Now let's try to imagine an example scenario - as a team of developers you are trying to launch another microservice in your cloud infrastructure that will automate the business processing management, and it will be orchestrated by the Camunda platform embedded in Spring Boot. For the sake of an example, let's imagine that the process will be taking care of a new customer's onboarding. At this point the usual challenge would be to adjust existing cloud infrastructure with a proper configuration that will enable a new instance of another microservice in our already existing system. The next step, if necessary, would be to implement custom features and let's say we want to utilize our Camunda platform with Azure SQL database. Now the effort is significantly reduced as all it takes is to select this feature in the UI form of the enablement platform and then, the generated project will be enhanced with all necessary implementation for Azure SQL database support, as well as the generated pipelines will be enhanced with creating cloud infrastructure like the database itself, following the IaaC pattern. Now the solution is ready to be deployed (also through generated pipelines) and after proper adjustments - it's ready for production use. This is just a simple example on how developer teams are accelerated by this platform, when it comes to bringing Camunda to the Azure cloud.

The platform IS a big success

The business case for this platform was rather simple. In a large organization of a company from the banking sector, with several dozen developer teams, most of them decided to go for business process automation with Camunda. The challenge was that most of those teams had to face similar issues, regarding setting up proper cloud infrastructure for Camunda & Spring Boot applications and of course to be aligned with the developer standards in the organization, many custom Camunda features had to be introduced. This platform allows to be ahead of most of these challenges. It occurred to be a huge success to have a possibility to enable projects with custom Camunda features on Azure cloud with a blink

of an eye and it turned out to be an investment that reduced development cost significantly. Now, not only the challenge was solved but also the platform exists up until today and is used by clients also for placing orders for new custom Camunda features, which also allows them to
be more concentrated on external vendor's particular business cases, while their casual struggle for providing custom Camunda features on Azure is not theirs to worry anymore. One could say that this platform is extremely similar to the Camunda initializer,
however it is enhanced with custom Camunda based features and instead of only generating the zip archive, it also brings your freshly generated project, and its infrastructure to Azure. How cool is that?

Conclusion

It is fair to say that the generic platform for Camunda solutions on cloud was a huge success in the organization. To summarize:

- the platform is easily accessible
- it is a guidance for developers who are trying to introduce Camunda projects on cloud
- the platform saves not only a lot of engineering effort, but an estimated cost of millions of euro
- the presence of the platform creates developer friendly standards in the company organizations
- the platform reduces the need of customizing Camunda through boilerplate code by using production ready solutions or solution patterns
- this approach allows to quickly introduce highly scalable Camunda based applications on cloud without a lot of specialized knowledge

In the end automation usually means success. From our point of view automation happens not only when it comes to external vendor requirements, but also within the organization that depends on quick implementation of Camunda solutions on cloud.