

Forum:[Hack Forums](#)

Thread: [Service-Oriented Architecture and how it can help you.](#)

Shprqness - January 2, 2021, 3:04 am

Before we begin, I've recorded a video version of this, however, this thread will contain more details:
<https://www.youtube.com/watch?v=9rTnycgSk00>

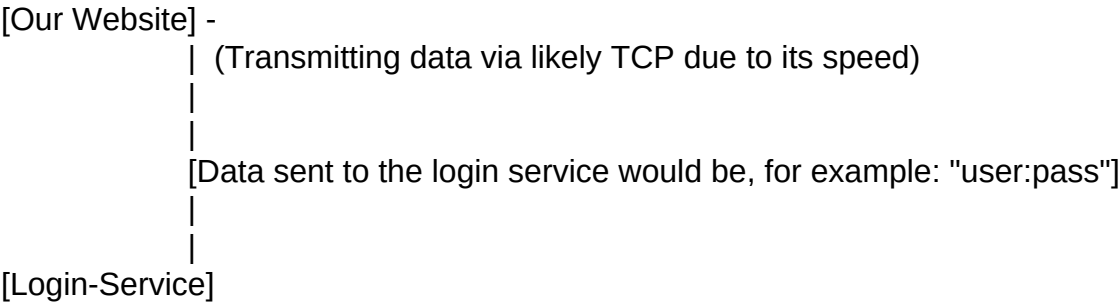
Key:
SOA = Service-Oriented Architecture

Summary
In layman's terms, SOA is one of the best forms of architecture for backend systems, it essentially entails creating multiple services for parts of your system, this would dramatically improve performance and resource consumption.

The fun part
Ideally, a good way to integrate this is to have one, or many different servers running different services, this is because, depending on your service load you may be required to spread out your services in order to lower resource consumption on a single node, which is why a lot of large systems are required to do this.

Let's assume we are developing a large e-commerce website, due to the amount of traffic we would be receiving we may be required to outsource our login, register, and purchasing parts of our website into microservices that will handle, process, and deal with data accordingly.

The way this would work is as follows (as an example): [For our Login Service]



Our login service would then process this data, check if its valid and would return a response confirming the validity of these credentials, this would relieve the processing stress from our web server dramatically as all it is required to do is send out some very light data and wait for a response rather than create database queries, etc.

Now here is an image of an even more complicated Service-Oriented Setup:

Any questions leave below.

Note - OUR MS HANDLER will keep open sockets to each service, this means you don't need to create multiple socks over and over again, you just create one that waits for data while the MS handler sends through open socket bridges its made to our services, an ideal situation would be to

have multiple handlers for different services based on weight.

Maxie - January 2, 2021, 3:16 am

This is generally not how it's done in bigger microservice architecture.

For example in AWS it is done by using an authentication method, and verify that the user is authenticated at the API Gateway ("MS Handler") and then sending the request through to the correct microservice. This microservice might in turn run software to be able to identify the user to make sure that the user gets access to the correct resources.

The database generally also tends to be a part of the microservice, so that everything belonging to that microservice is contained together.

Shprqness - January 2, 2021, 3:22 am

(Jan 2, 2021 04:16 AM)Maxie Wrote: [\[CITE\]](#) This is generally not how it's done in bigger microservice architecture.

For example in AWS it is done by using an authentication method, and verify that the user is authenticated at the API Gateway ("MS Handler") and then sending the request through to the correct microservice. This microservice might in turn run software to be able to identify the user to make sure that the user gets access to the correct resources.

The database generally also tends to be a part of the microservice, so that everything belonging to that microservice is contained together.

Yes, my initial thought of making this was to approach it at a beginner level, as I understand not many people will be entering this form of architecture with experience with this sort of architecture, if you get what I am saying. But I agree with you, larger-scale systems do use more complex methods etc. I did not want to approach authentication in this thread however as I felt like for an experienced developer, that was self-explanatory when building your own servers.

Shadow_Cloud - January 4, 2021, 5:53 am

So I'm about to state a very unpopular opinion, but here goes :

"Most software should just be built as a monolith."

In my experience, most software projects fail. The overhead of getting into complicated architectures for the benefits you'll gain from growth is almost never utilized. Instead focus on good SOLID principles from the start so you can easily split into whatever architecture supports your longer term goals later.

So yes, this SOA example will provide a single point of failure in the DB and its very tempting to write queries that can join different tables leading to contextual bleed down the line which makes it very hard to change your application later, but the simpler the architecture for a POC the more likely you'll be able to build a bunch of software so you can eventually create software that takes of.