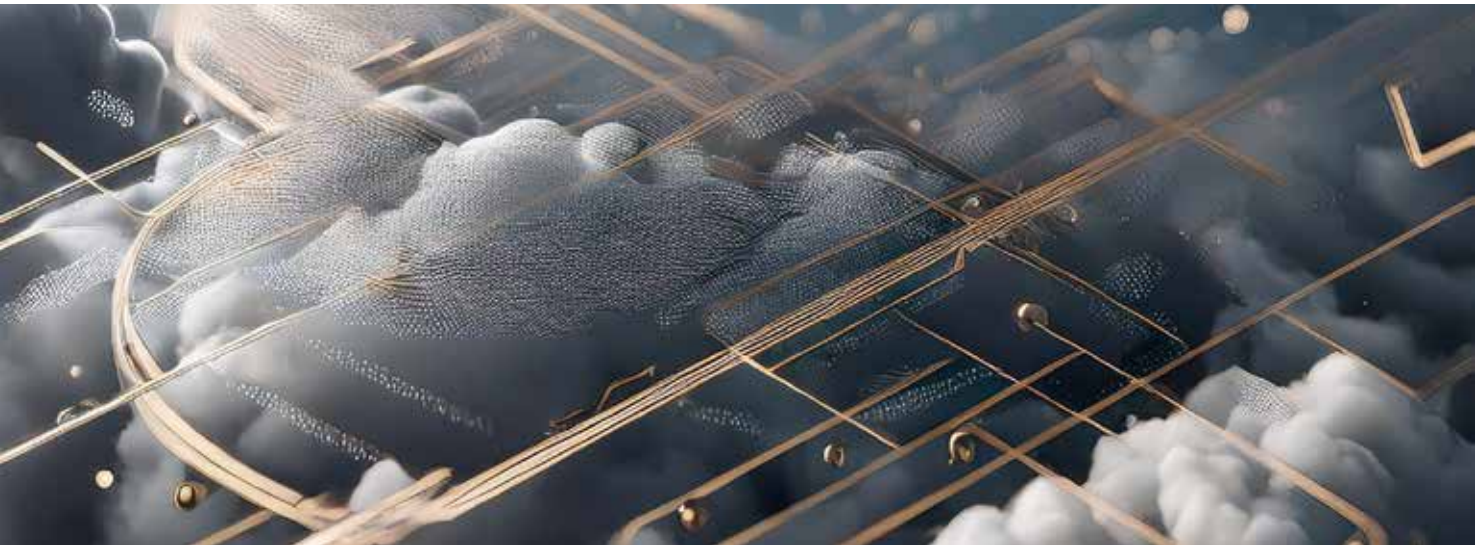
Cloud or on-premises: What's in favour of which solution?

The component-based system architecture of PDTec AG (Karlsruhe, Germany) gives the customer the choice of when to migrate to the cloud and how long to pursue a hybrid approach consisting of on-premises and cloud implementation. A look at the software system architecture shows the various advantages.



Components of the system architecture for cloud compared to on-premises

The system architecture is characterised not by differences, but by similarities. PDTec pursues the strategy of being able to optimally operate a software platform and thus the entire application suite both on the basis of an on-premises infrastructure and in the cloud without compromise. To this end, application services are systematised and classified. Regardless of their actual function, the services can be divided into three classes:

- The web services contain both domain-specific business logic and an HTTP/S-based service interface. Examples of web services are browser-based user interfaces or REST APIs.
- Worker services also contain domain-specific business logic, but no HTTP/S interface that is available to the outside world. In contrast to web services, they perform their functionality in the background (batch processing) and are controlled via a batch job queue. Examples are data export and import procedures, data converters or programs for running simulations.
- Infrastructure services do not contain any business logic, but perform a specific task in a (technical) context of the overall system. Examples are a vault service (file storage service) or database service.

Based on this classification, the application services can be transferred to the respective technical representation of the deployment variant (on-premises or cloud). With on-premises deployment on IIS or Windows servers, web services become IIS applications, worker services become Windows services and infrastructure services are installed within the server infrastructure. With cloud deployment, however, web services become web containers, worker services become worker containers, and for infrastructure services, the services already available in the cloud infrastructure are used wherever possible. For example, a vault service can be replaced by AWS S3 or Azure Blob Storage, a database service by Azure SQL or a PostgreSQL service. With appropriate tailoring of the services, it is possible to implement the service only once, but package and operate it in both variants.

Procedure for a hybrid approach

The architectural approach described allows the same data management software to be implemented both in a cloud infrastructure and conventionally on on-premises servers. It is also technically possible to operate a hybrid (mixed) form of the two environments. The specialized application services and the database can be installed in a cloud environment. A vault service for the management and transfer of user data — which can generate very large amounts of data in the design and simulation domains — can be operated in an on-premises installation with its own storage infrastructure.

The advantage of operating application services in the cloud is that you can work in a homogeneous environment and do not have to take care of the details of software operation such as installation, load balancing and high availability yourself.

Using a storage service for binary content via the cloud platform such as AWS S3 or Azure Blob Storage is also convenient for operation. Here the management service is provided via a service interface and used by the application. It is not important for the user to know how the Hyperscaler implements the storage and availability of the data. It simply provides a service with a certain service quality in terms of latency, bandwidth and capacity. However, this also involves pricing that depends on these parameters.

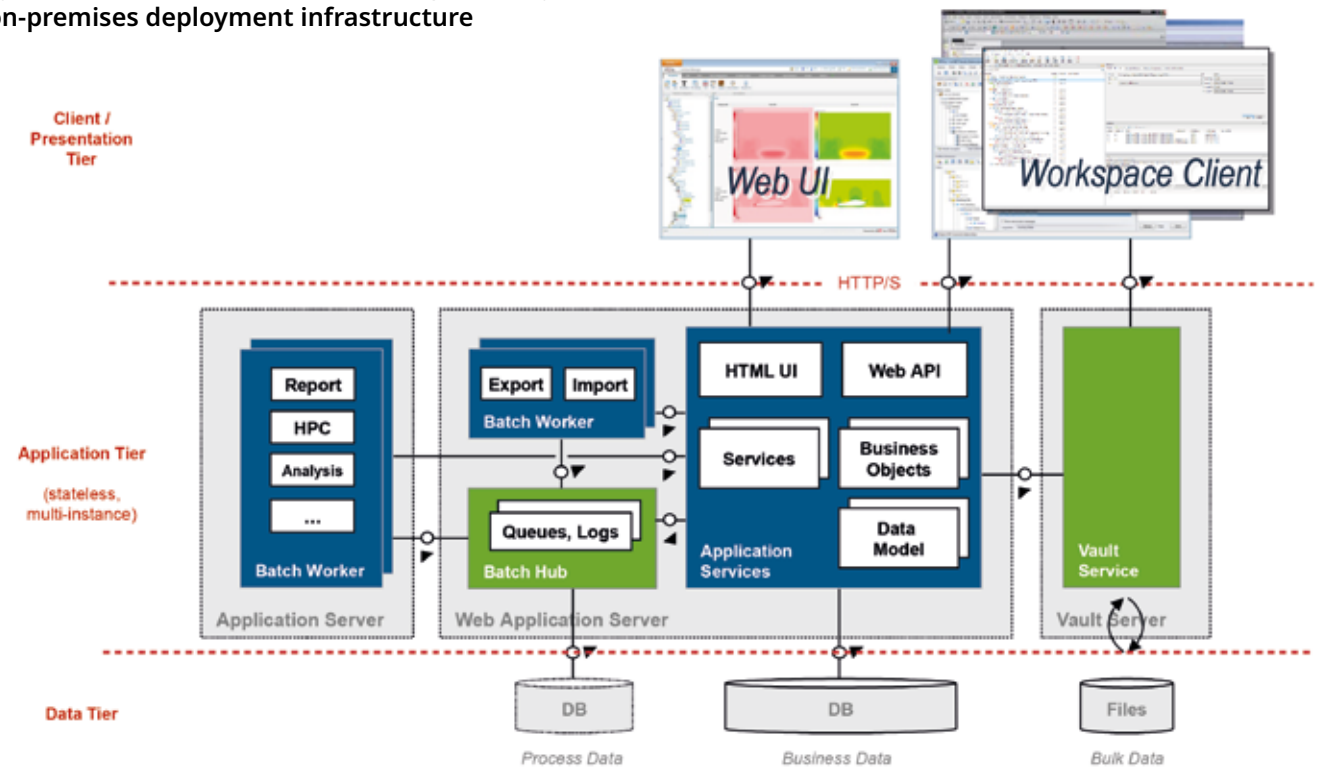
With the hybrid approach, for example, the operational IT can have a direct influence on the parameters of this service by deciding to operate the storage service explicitly in an on-premises variant and thus also directly control the associated costs and the service quality. This makes sense, for example, when managing raw results in the simulation, which are not so critical to availability because they are regularly replaced by new simulations.

What to consider when introducing cloud infrastructure

On the software side, cloud operation requires a consistently designed software architecture with stateless services (see for example (1)), classification of services, appropriate customisation of services according to the microservice principle (2).. The PDTec platform ensures a uniform structure of the services, regardless of their technical function. Parts of the services are generated by software development tools (such as Business Objects Builder, Service Builder) and ensure technical compatibility in the platform.

Other challenges include the smart implementa-

System architecture of a data management system in a classic on-premises deployment infrastructure



The figure shows how different application services are operated and interact on servers. Purely technical infrastructure services (authentication, firewall, load balancing, etc.) are not shown.
Source: PDTec AG

tion of the individual services, which, for example, are not allowed to access temporary background storage but use the uniform caching infrastructure provided by the platform. Furthermore, the system must be configured uniformly using environment variables and database entries. 'Free-flying' configuration data have to be avoided.

In cloud environments, integrated authentication methods such as OIDC must also be supported. However, this is now also finding its way into on-premises infrastructures.

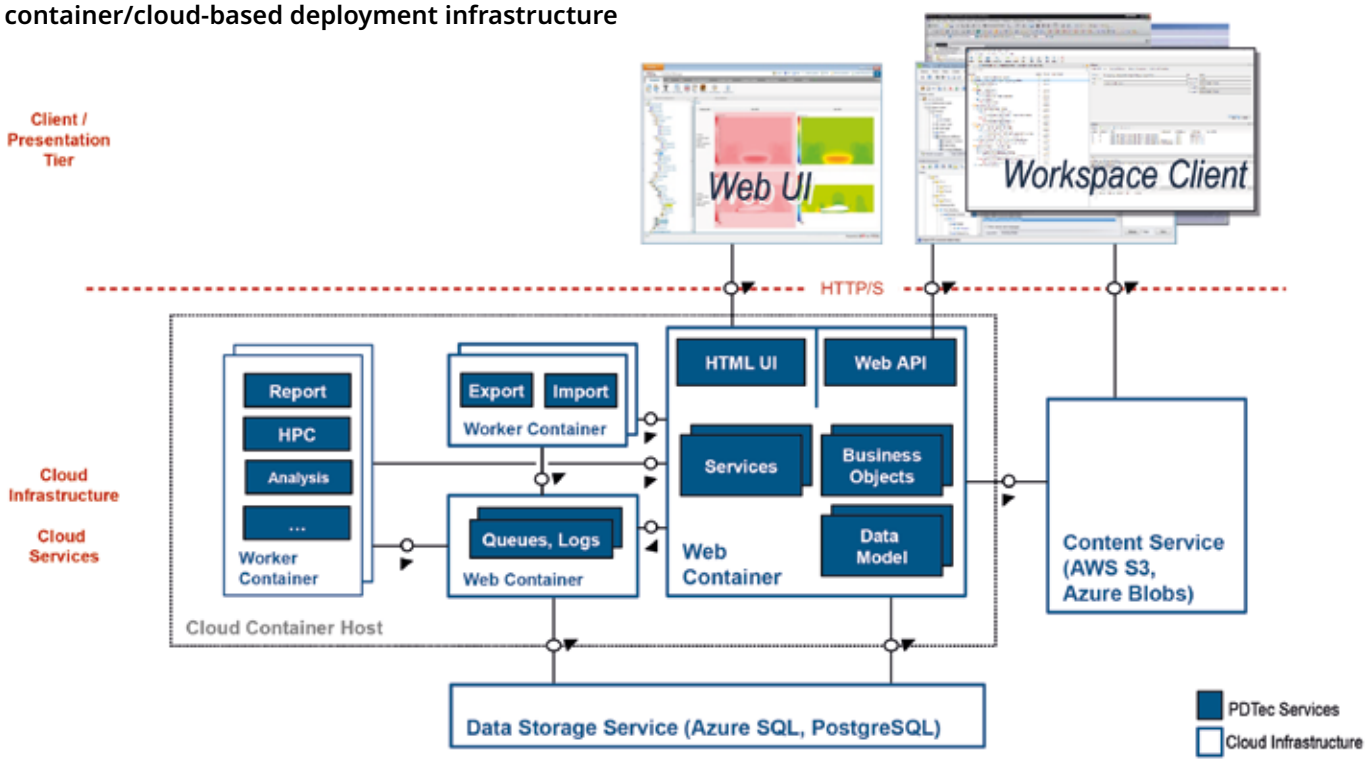
Outlook: Roadmap at a glance

PDTec's cloud strategy is being implemented in several stages. In the first step, the software is upgraded so that it can run in a cloud environment. To this end, all legacy technologies that impair deployment in containers and the ability to run independently of the operating system are replaced by state-of-the-art, cloud-capable technologies.

In the second step, the services are tailored and internally structured according to a platform architecture based on the principles already mentioned. On the basis of this modularization, the application components can now be packaged for cloud operation.

In the third step, it is ensured that the application services run in a neutral

System architecture of the same data management system in a container/cloud-based deployment infrastructure



The figure shows that the structure of the application services and their communication do not differ significantly from those of the classic deployment (left). The differences lie in the hosting (container environment instead of servers / virtual machines) and in the infrastructure services (content storage, database, etc.).

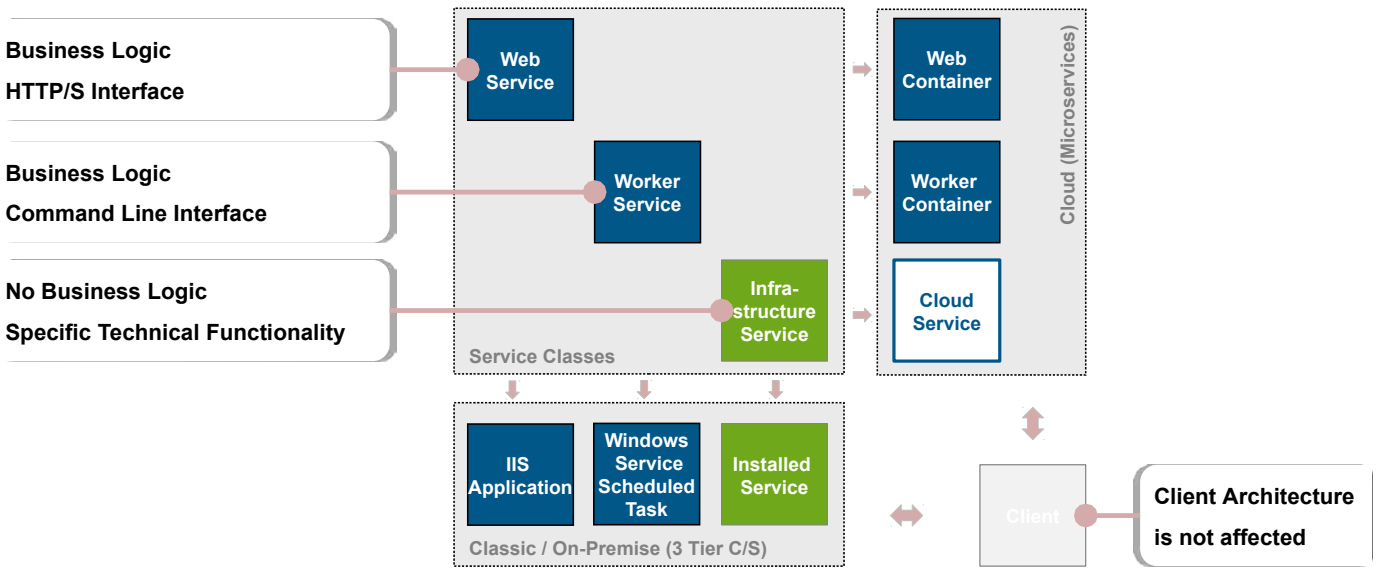
Source: PDTec AG

container environment and interact with each other (orchestration). It is initially irrelevant whether the container infrastructure is actually operated in a public cloud or in an internal container host.

If the ability to run in a container infrastructure is guaranteed, the cloud deployment for the specific cloud environment (for example Kubernetes on MS Azure) is configured and put into operation in the fourth step. For private cloud environments of larger customer installations, corresponding container registries can also be supplied with container images instead of a direct deployment. The following is also worth mentioning: As container and cloud technologies in particular are subject to constant change, attention is paid to the continuous optimisation of cloud operations

References
(1) en.wikipedia.org/wiki/Service_statelessness_principle
(2) www.geeksforgeeks.org/10-microservices-design-principles-that-every-developer-should-know/

Classification of application services and mapping of service classes in the respective deployment infrastructure (on-premises or cloud)



The figure shows the three service classes (middle) and the technical characteristics of the service implementations in the two deployment variants on-premises (below) and container/cloud (right)

Source: PDTec 2024