

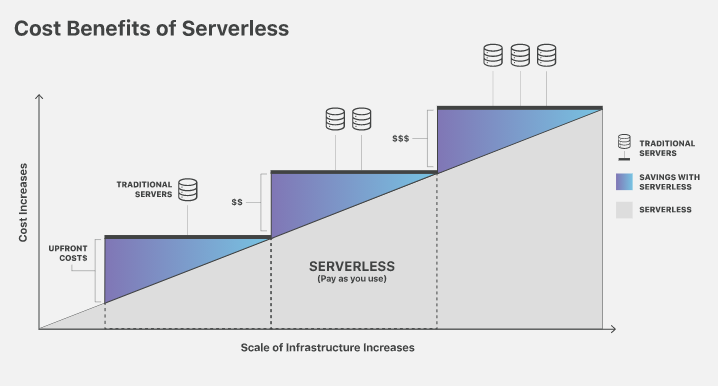
<https://www.cloudflare.com/learning/serverless/what-is-serverless/>

**What is serverless computing?**

Serverless computing is a method of providing backend services on an as-used basis. A serverless provider allows users to write and deploy code without the hassle of worrying about the underlying infrastructure. A company that gets backend services from a serverless vendor is charged based on their computation and do not have to reserve and pay for a fixed amount of bandwidth or number of servers, as the service is auto-scaling. Note that despite the name serverless, physical servers are still used but developers do not need to be aware of them.

In the early days of the web, anyone who wanted to build a web application had to own the physical hardware required to run a server, which is a cumbersome and expensive undertaking.

Then came cloud computing, where fixed numbers of servers or amounts of server space could be rented remotely. Developers and companies who rent these fixed units of server space generally over-purchase to ensure that a spike in traffic or activity will not exceed their monthly limits and break their applications. This means that much of the server space that gets paid for can go to waste. Cloud vendors have introduced auto-scaling models to address the issue, but even with auto-scaling, models to address the issue, but even with auto-scaling an unwanted spike in activity, such as a DDoS Attack, could end up being very expensive.



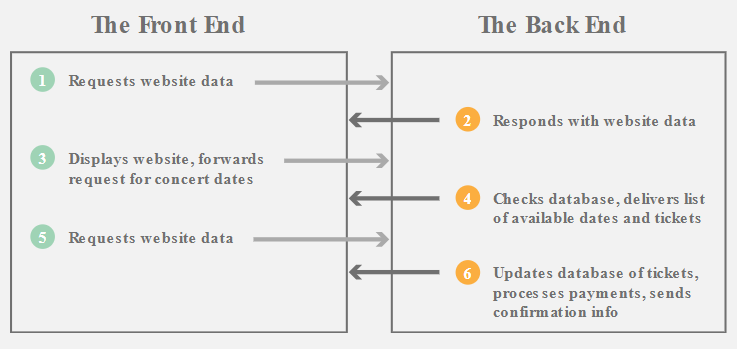
**Benefits of Serverless**

Serverless computing allows developers to purchase backend services on a flexible ‘pay-as-you-go’ basis, meaning that developers only have to pay for the services they use. This is like switching from a cell phone data plan with a monthly fixed limit, to one that only charges for each byte of data that actually gets used.

The term ‘serverless’ is somewhat misleading, as there are still servers providing these backend services, but all of the server space and infrastructure concerns are handled by the vendor. Serverless means that the developers can do their work without having to worry about servers at all.

**What are backend services? What’s the difference between frontend and backend?**

Application development is generally split into two realms: the frontend and the backend. The frontend is the part of the application that users see and interact with, such as the visual layout. The backend is the part that the user doesn’t see; this includes the server where the application's files live and the database where user data and business logic is persisted.



**Frontend vs Backend of an Application**

For example, let’s imagine a website that sells concert tickets. When a user types a website address into the browser window, the browser sends a request to the backend server, which responds with the website data. The user will then see the frontend of the website, which can include content such as text, images, and form fields for the user to fill out. The user can then interact with one of the form fields on the frontend to search for their favorite musical act. When the user clicks on ‘submit’, this will trigger another request to the backend. The backend code checks its database to see if a performer with this name exists, and if so, when they will be playing next, and how many tickets are available. The backend will then pass that data back to the frontend, and the frontend will display the results in a way that makes sense to the user. Similarly, when the user creates an account and enters financial information to buy the tickets, another back-and-forth communication between the frontend and backend will occur.

**What kind of backend services can serverless computing provide?**

Most serverless providers offer database and storage services to their customers, and many also have Function-as-a-Service (FaaS) platforms, like Cloudflare Workers. FaaS allows developers to execute small pieces of code on the network edge. With FaaS, developers can build a modular architecture, making a codebase that is more scalable without having to spend resources on maintaining the underlying backend. Learn more about FaaS >>

**What are the advantages of serverless computing?**

* Lower costs - Serverless computing is generally very cost-effective, as traditional cloud providers of backend services (server allocation) often result in the user paying for unused space or idle CPU time.
* Simplified scalability - Developers using serverless architecture don’t have to worry about policies to scale up their code. The serverless vendor handles all of the scaling on demand.
* Simplified backend code - With FaaS, developers can create simple functions that independently perform a single purpose, like making an API call.
* Quicker turnaround - Serverless architecture can significantly cut time to market. Instead of needing a complicated deploy process to roll out bug fixes and new features, developers can add and modify code on a piecemeal basis.

**How does serverless compare to other cloud backend models?**

A couple of technologies that are often conflated with serverless computing are Backend-as-a-Service and Platform-as-a-Service. Although they share similarities, these models do not necessarily meet the requirements of serverless.

Backend-as-a-service (BaaS) is a service model where a cloud provider offers backend services such as data storage, so that developers can focus on writing front-end code. But while serverless applications are event-driven and run on the edge, BaaS applications may not meet either of these requirements. Learn more about BaaS >>

Platform-as-a-service (PaaS) is a model where developers essentially rent all the necessary tools to develop and deploy applications from a cloud provider, including things like operating systems and middleware. However PaaS applications are not as easily scalable as serverless applications. PaaS also don’t necessarily run on the edge and often have a noticeable startup delay that isn’t present in serverless applications. Learn more about PaaS >>

Infrastructure-as-a-service (IaaS) is a catchall term for cloud vendors hosting infrastructure on behalf of their customers. IaaS providers may offer serverless functionality, but the terms are not synonymous. Learn more about IaaS >>

**What is next for serverless?**

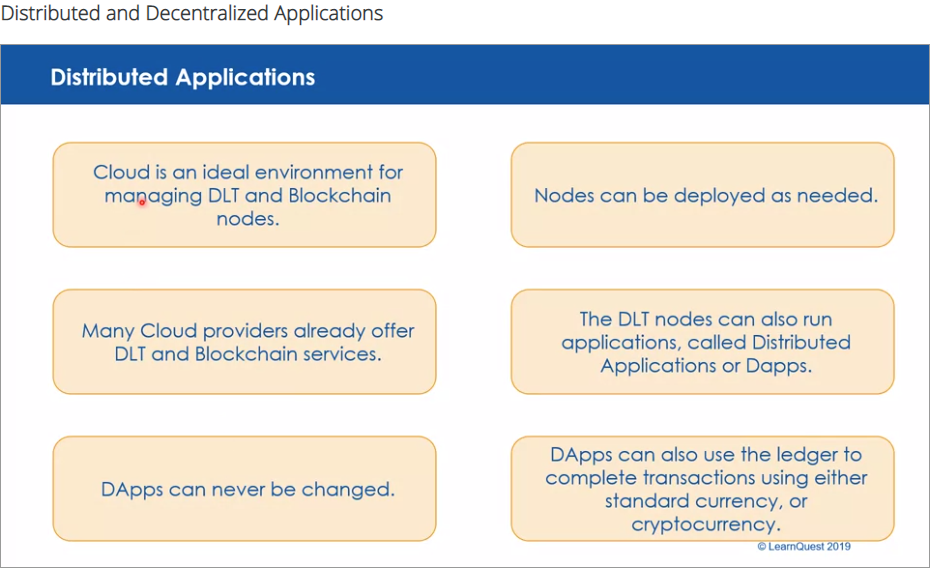
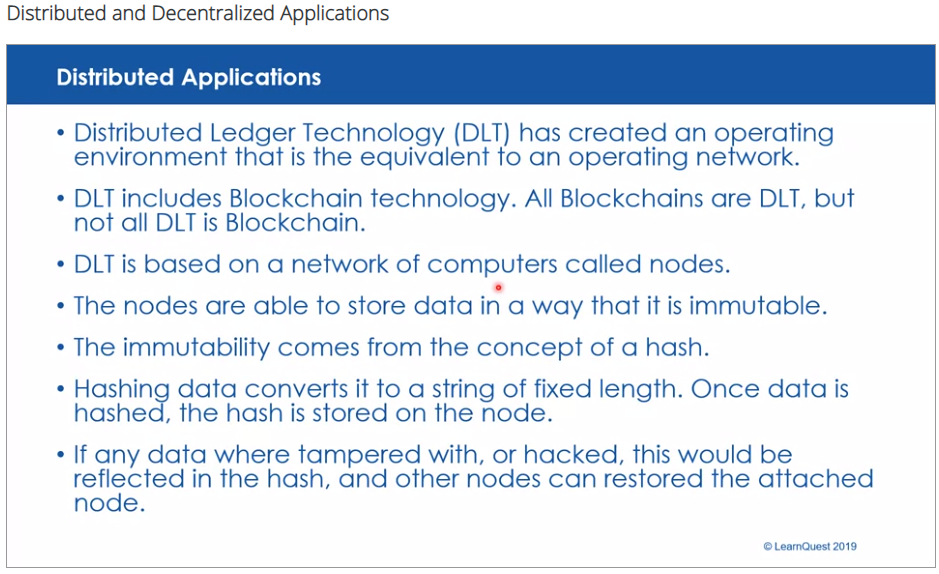
Serverless computing continues to evolve as serverless providers come up with solutions to overcome some of its drawbacks. One of these drawbacks is cold starts.

Typically when a particular serverless function has not been called in a while, the provider shuts down the function to save energy and avoid over-provisioning. The next time a user runs an application that calls that function, the serverless provider will have to spin it up fresh and start hosting that function again. This startup time adds significant latency, which is known as a ‘cold start’.

Once the function is up and running it will be served much more rapidly on subsequent requests (warm starts), but if the function is not requested again for a while, the function will once again go dormant. This means the next user to request that function will experience a cold start. Up until fairly recently, cold starts were considered a necessary trade-off of using serverless functions.

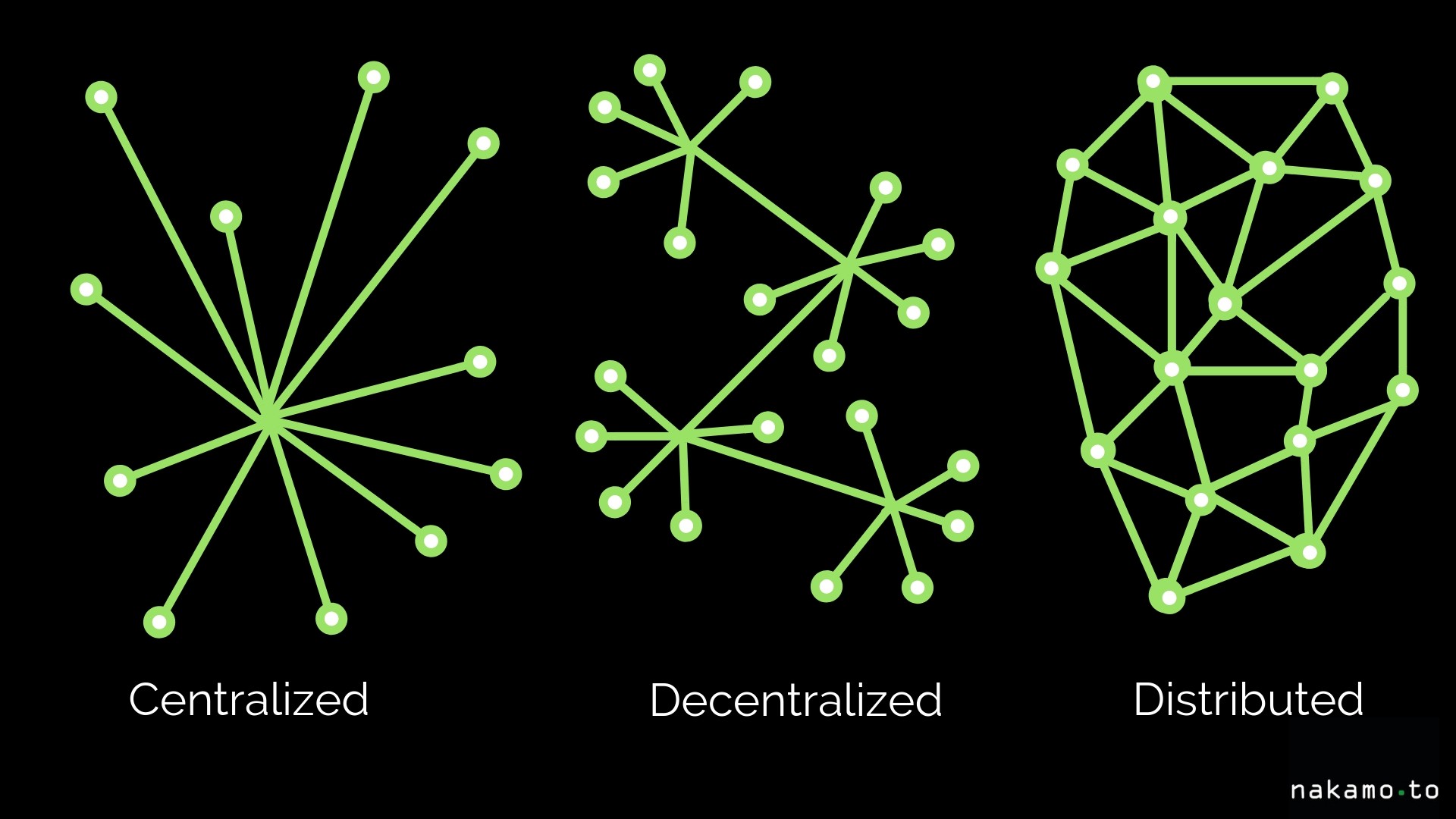
Cloudflare Workers has addressed this problem by spinning up serverless functions in advance, during the TLS handshake. Since Workers functions spin up at the edge in a very short amount of time, even shorter than the time required to complete the handshake, the result is an FaaS platform with zero cold starts.

As more and more of the drawbacks of using serverless get addressed and the popularity of edge computing grows, we can expect to see serverless architecture becoming more widespread.



**What’s the difference between Decentralized and Distributed ?**

And how does this relate to Private vs. Public blockchains?



The terms ‘decentralized’ and ‘distributed’ are often used when talking about blockchains — and often confused, for the difference is not always obvious.

Bitcoin’s blockchain protocol, for instance, is a decentralized system for exchanging digital cash — but it’s also an example of distributed ledger technology. What’s going on?

**Centralization and decentralization refer to levels of control.**

In a centralized system, control is exerted by just one entity (a person or an enterprise, for example). In a decentralized system, there is no single controlling entity. Instead, control is shared among several independent entities.

**Distribution refers to differences of location.**

In a non-distributed (or co-located) system, all the parts of the system are in the same physical location. In a distributed system, parts of the system exist in separate locations.

**Allow me to clarify using examples.**

Imagine you are writing a document using the Microsoft Word application on a PC running the Microsoft Windows operating system. This setup is both centralized and non-distributed. One entity, Microsoft, controls your application and operating system (centralized). Both the application and the operating system are located on your PC, i.e. in one physical location (non-distributed).

Now, suppose you run an open source operating system like Linux on your PC with open source word-processing software. Different people and organizations contribute to the evolution of both items (the setup is now decentralized). On the other hand, you are still using one physical PC to run all the software (it’s still non-distributed).

A distributed, but centralized system may sound contradictory, but if we use the definitions above based on control and location, we’ll see how this works. Consider a cloud service provider offering a data storage service. Physically, your data could be shared and replicated on different machines according to resource availability and resiliency(distributed). However, wherever the machines and data storage facilities happen to be, the cloud service provider still controls them all (centralized).

That leaves one last case, which is the decentralized and distributed system. We’ll use Bitcoin as our example. Bitcoin is a blockchain system that cannot be altered by any one entity (decentralized). It also runs as a peer-to-peer network of independent computers spread across the globe (distributed).

We can now make more sense out of discussions about how much certain blockchains are truly decentralized or distributed.

Whereas public blockchains tend to be decentralized, private blockchains are typically controlled by one entity like an enterprise. Some people then suggest that private blockchains should not be called blockchains, for this reason (i.e., they are centralized).

Decentralized blockchains using a mining consensus system like proof of work may lose their decentralization if one miner (or group of miners) becomes more powerful than all the others, leading to the threat of a 51% attack.

Cryptocurrency exchanges serving users of otherwise decentralized blockchains may introduce centralization and risk. Consider the Mt. Gox exchange, once the largest online exchange for users to buy and sell Bitcoins. It was controlled by one entity, lacked security, and was hacked several times before it finally shut down.

Even the ‘decentralized-by-design’ blockchains of Bitcoin and Ethereum have been rated as ‘not particularly decentralized’ in a study conducted at Cornell University. The study found that power was concentrated among relatively few miners or mining pools. It also considered these blockchains to be less distributed than one might think, with many blockchain nodes physically located in just a few large data centers .

To conclude, there are degrees of decentralization and distribution, rather than hard divisions. How much decentralization or distribution is desirable then depends on your objectives. Hopefully with the definitions given here, you’ll be able to have a meaningful discussion on the matter and a constructive exchange of points of view.

