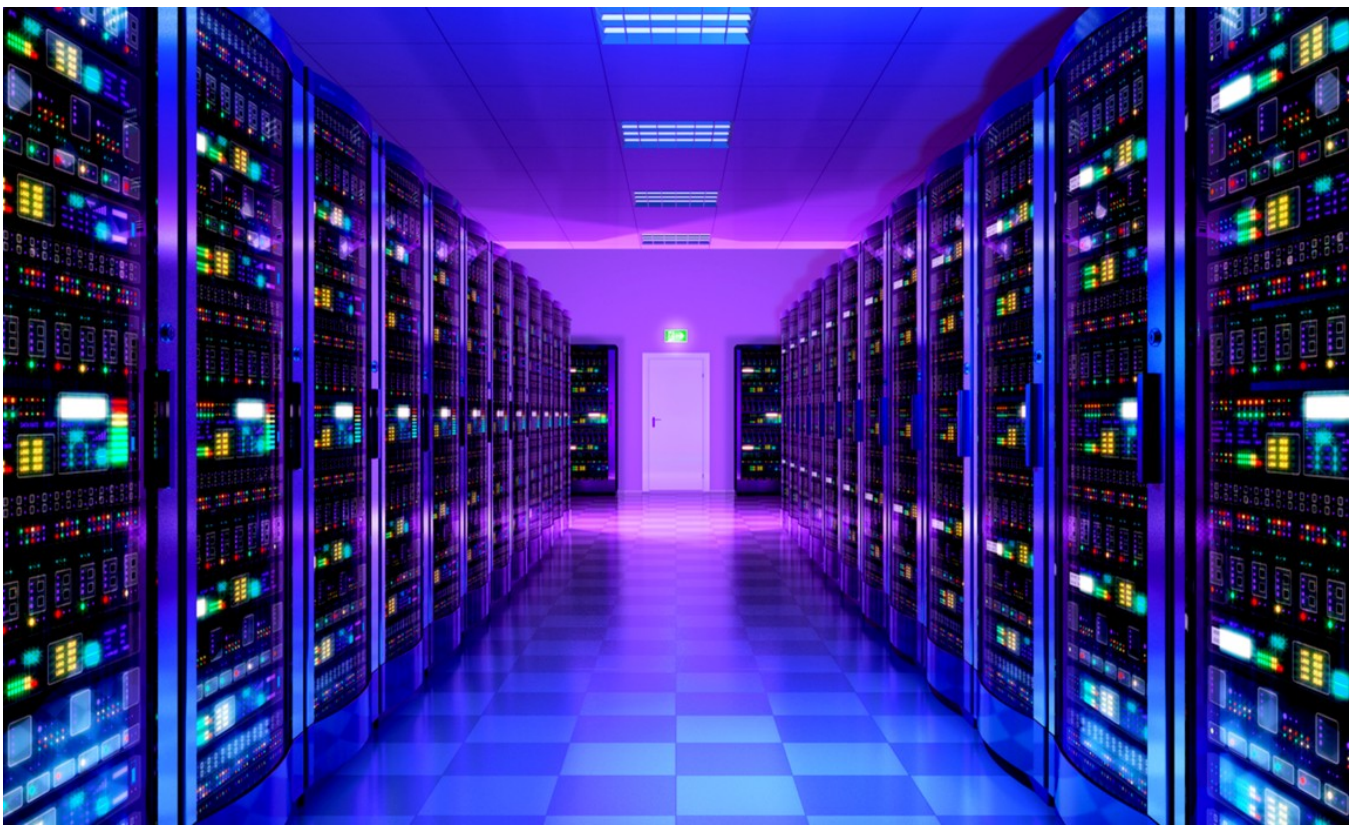
Benefits of Serverless and Event Driven Design

The server is dead, long live serverless — by Craig Godden-Payne@beardy.digital

Craig Godden-Payne
May 10 · 4 min read ★



You can define serverless by its name. You care less about the server and only really about the software that is running on it.

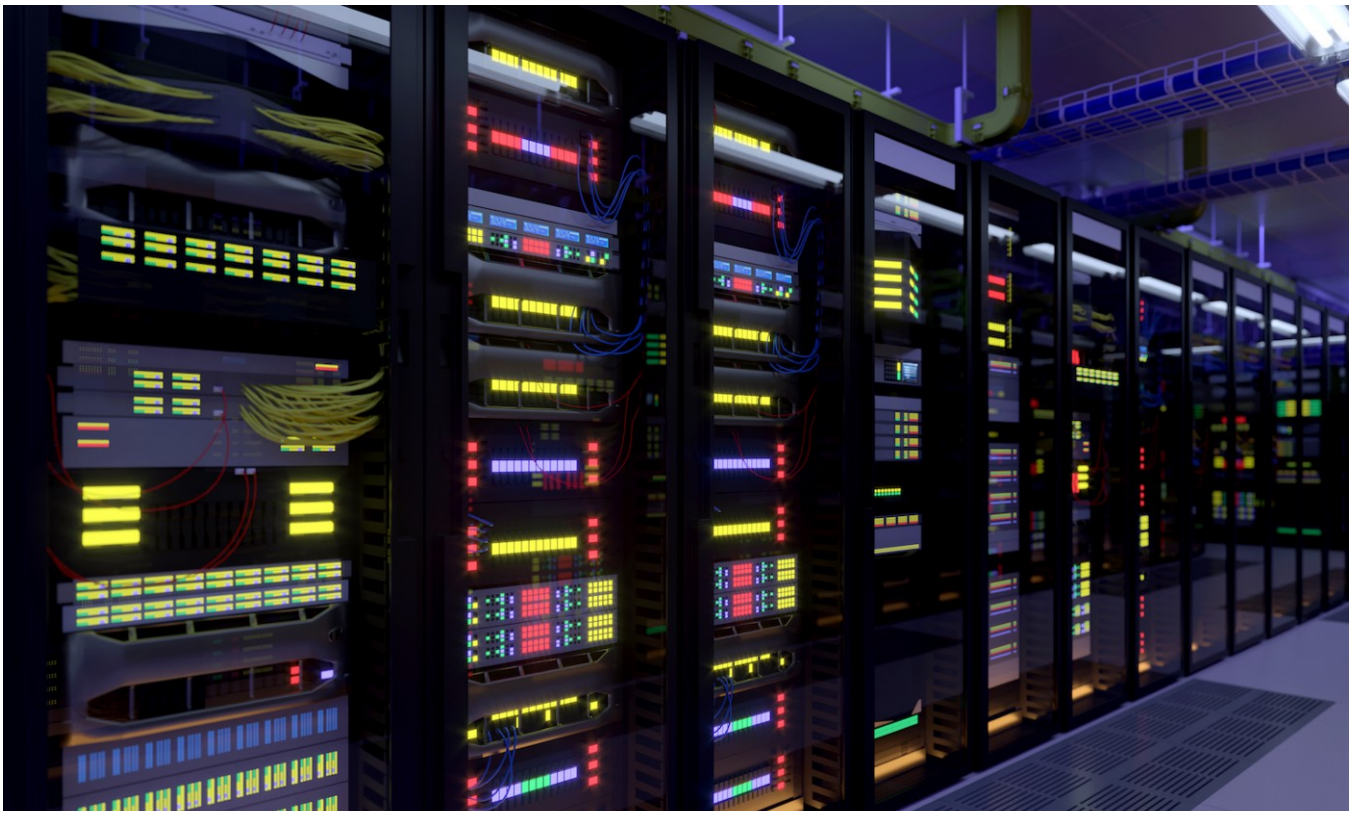
Gone are the times when an admin would have to keep the server up to date, adding new patches and making sure it runs as it should. This is now all handled by the cloud platform.

. . .

What are the benefits

The two main features of serverless are computing are cost and elasticity.

Since you no longer have to hire a team of experts to administer a server, you effectively save on this cost.



. . .

Cost benefits of serverless

You don't need to worry about idle costs, as most cloud providers will provide a so called `function as a service` where you pay for the amount of execution time, over the amount of resources consumed.

AWS offers many different kinds of serverless computing, but the cost varies. Some examples would be:

- AWS lambda — where you pay only for execution time, but is limited due to running within a set of boundaries, i.e. A node, python, or dotnet application which runs on a particular version of a runtime. You have access to most functionality, the exception usually being low level system calls, there also is a max execution time which cannot be exceeded.
- AWS Fargate — where again you only pay for the execution time, but has more freedom since it runs a container image, meaning pretty much anything that can be containerised can be run. Fargate doesn't have a max executing time, so effectively you can run serverless 24/7.

. . .

Elasticity benefits of serverless

Using serverless offerings, you can essentially support scaling from zero. This means if your workload is not needed, you do not need to pay for idle servers. The challenge with this would be in the scheduling and optimising of these cloud resources.

It also means your compute power can scale as your business needs grow, today you may only need 1 container running 24/7, but tomorrow after launching an ad campaign, you may want to increase your capacity to match.

Luckily, most cloud platforms will allow you to scale on certain factors, such as CPU or Memory load, time of day, or custom factors such as network utilisation.

. . .

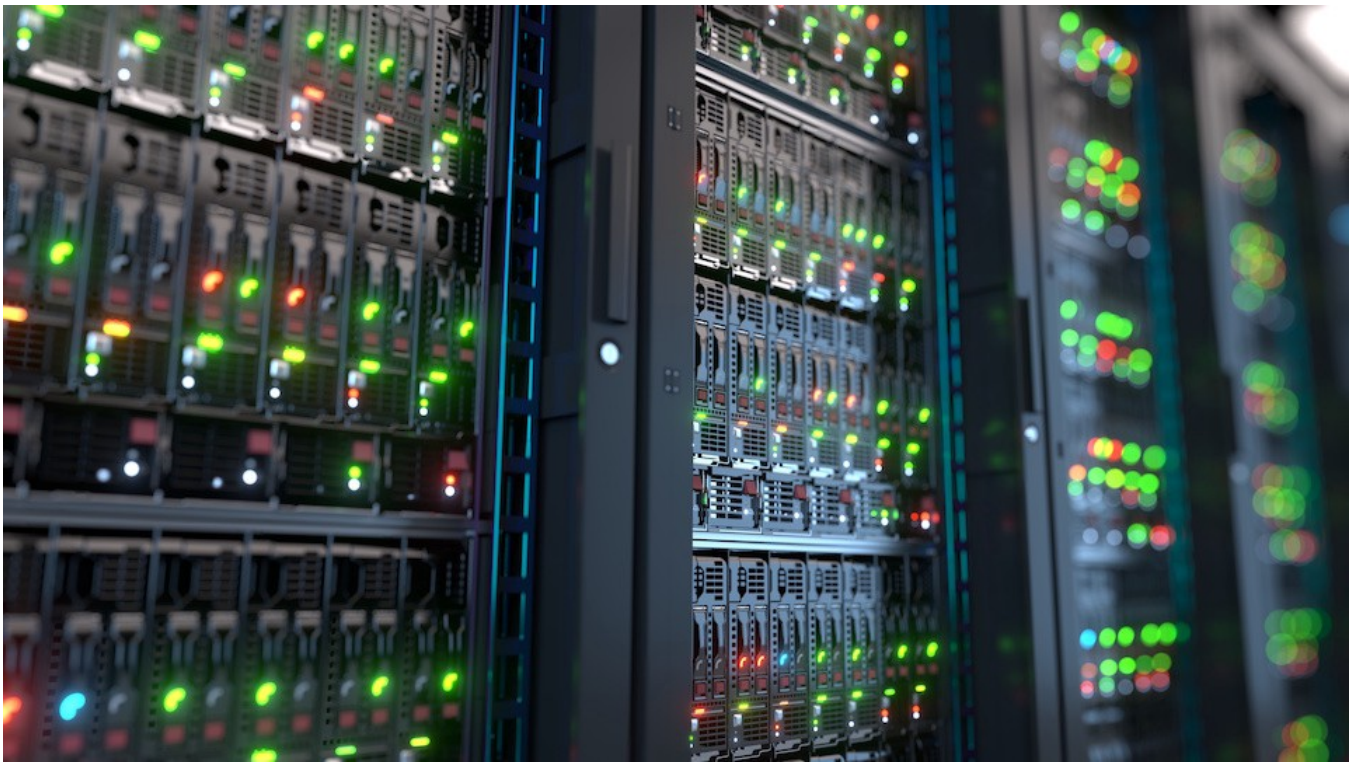
Where does event driven design come into play?

Serverless and Stateless go hand in hand. Services such as AWS Lambda can expect to run asynchronously, and not feed back to something waiting for an operation to complete.

By using these small amounts of computation, you can easily scale and maintain a relatively small cost compared to traditional microservices, which are running 24/7, waiting for the request to hit the server.

Event driven design does add a layer of complexity, in terms of the architecture of an application is likely to be spread out across a cloud, rather than isolated within say an EC2 instance.

There are many patterns that work well with serverless and event driven design, and some not so.

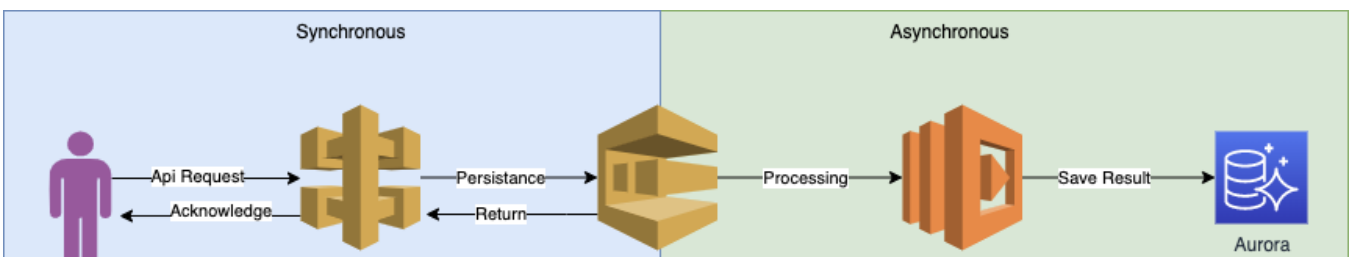


. . .

Some serverless design patterns:

A simple serverless web request with some processing and storing a result.

A good example of this would be a contact us form on a webpage.

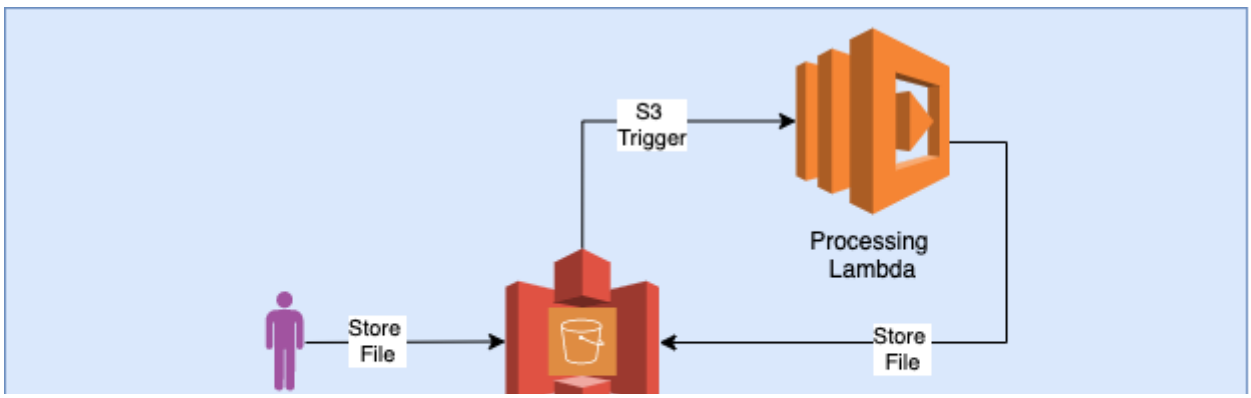


1. An S3 static website, which the user submits a webpage, and data is sent to Api Gateway
2. Api Gateway passes this data onto an SQS queue.
   At this point, it returns to Api Gateway, and the user is returned a 200 status code.
3. The SQS triggers a lambda to process the message. This could be anything from cleaning the data, sending an email etc.
4. The lambda also stores the data in a serverless database, this is just a form of persistence, and could be used for detecting duplicates etc.

. . .

An action performed on a file on the back of an upload

A good example would be image processing on an uploaded file.



1. A file is uploaded to S3 via a web application or by some kind of user interaction
2. Once the file upload is complete, an S3 trigger as invoked, which calls a Lambda
3. The Lambda performs some kind of processing operation on the file which was uploaded
4. The result is then stored back in S3

Serverless Event Driven Architecture Software Development DevOps