# Web Application

An important aspect to LogiSteps is the ability to portray user data generated from their shoes in a clean, easy to use manner. While there may be several approaches for displaying the data (mobile application, native application, etc.), one of the most dynamic and flexible options is using an HTML based web user interface - doing so presents numerous advantages over other display options. While the advantages are numerous and vary from use case to user case, a few important advantages to consider are the following:

* **Cross Platform** – by serving user interfaces from a web server, it allows the application to be designed without worrying about compatibility across Windows, MacOS, and Linux. UX code will render and run in the browser.
* **Installation-less** – Since the user interface is being served from a web server, there is no need to design software for installing and booting the user interface. End users only need to know the right URL for accessing the application.
* **Updates** – Updates can be easily deployed to a centralized location without the need to push updates software to numerous distributed users.
* **Administrative Overhead** – by deploying the end user application to a web server, our project will be able to mitigate much of the administrative overhead placed by companies such as Apple.
* **Fast Prototyping/Development** – The abundant availability of front end frameworks help get the project into deployable state without needing to design UI components from the ground up.

Of course, there are more advantages than the ones considered, however they help illustrate the case for using a web application for presenting data to users.

While there are numerous advantages to this approach, one must also consider valid disadvantages, and factor them into the decision-making process. The use of a web server for processing and displaying user data lacks behind mobile and native apps when considering the following issues:

* **Processing Overhead** – Using front end and back end frameworks for building an interactive application for a user comes with a lot of extra files. Web applications often require an extensive number of files for rendering and processing data, much of which may not be used if the user does not navigate to that respective part of the application. Additionally, since the application is abstracted from the system, it often runs slower than native desktop and mobile apps. The speed of the application also becomes dependent on the web browser being used, an aspect of design that the developer cannot control.
* **Programming Language** – Since the front-end portion of the application is being run in a browser, development is limited to javascript, HTML, and CSS. Additionally, the back-end server will also be limited to a language supported by web frameworks and web servers – typically NodeJS, Python, Ruby, etc.
* **Resources** – The application will not have access to many of the client’s system resources. This will somewhat limit the design flexibility for displaying information to the user.
* **Security** – Resources will be more viable to attack and probe from outside sources. Additionally, data being transmitted over the public internet may be vulnerable to spies. Security becomes a much larger problem when using the public internet to serve and present resources to users.

One last thing to note that by presenting information to users using a web application, the application becomes decoupled, with a distinct separation of client side and server-side software. This can be an advantage if designed correctly but can make it more difficult to design and develop as well. The server and client may not be able to easily access resources that could be used if both aspects of the application were closely coupled (like a native mobile/desktop app).

The web application will be a composition of 4 logically separated components – the front-end framework, the back-end framework, the long-term storage, and the web server. A critical review of available resources are discussed in the following sections of the Web Application technology and concept report.

# Front-End Framework

Choosing a front-end web framework for a web application can be a difficult task due to the vast multitude of available frameworks – each claiming to be better than the next. Each framework excels aspects that other frameworks may lack, and lack in others. After doing initial technology research, team Omicron discovered that three of the most popular front-end web frameworks available to developers are Angular, ReactJS, and VueJS. Other notable front end frameworks worth mentioning are Ember, Elm, knockout, and more.

When considering front-end frameworks, several attributes were explored, some of which included available documentation, architecture, data binding, load times, syntax, DOM manipulation, element expression, supporting community, learning curve, and more. Each of these attributes were closely examines, and used to compare each framework, helping drive a choice. Each of the comparable attributes are examined in finer detail below.

## Documentation

Angular, React, and Vue are three of the most popular front-end frameworks in the world. As a result, each has an extensive amount of rich, detailed documentation and tutorials. All three frameworks have the necessary documentation to begin development and reference if there is any trouble in development. Of all three, it is difficult to choose which framework has the best documentation, but it appears that React may lag behind both Angular and Vue in documentation, with simple explanations of the software[\*\*\*CITATION HERE\*\*\*]. This claim appears to be backed up by several members of the software development community as well, with many claiming that React lacks in some aspects of its documentation. Having little experience in web development, documentation is an essential attribute for Team Omicron to consider when choosing a proper front-end framework.

## Architecture

Architecture of a framework has a large impact on how flexible a developer’s application can be. Through Team Omircron’s research, the group found Angular to be highly opinionated on how a developer’s application should be structured, with React and Vue providing a much more flexible approach, allowing developers to structure their application in a manner that best fits it’s needs. This has much to do with the fact that React and Angular are only View layer libraries, while Angular uses a model, view, view model structure with structured components and services.

Having a strict/strongly opinionated framework can make design work simpler by limiting the developer’s decisions, but it limits the flexibility of the application, while also adding overhead that is often unnecessary. This makes Angular a much heavier framework than Vue and React, introducing a lot of complexity that is likely unneeded. On the other hand, having a structure to develop off can sometimes be beneficial in implementing an application. Due to this, both Vue and React have templates and third-party plugins for modeling client-side data and structuring an application. This makes it possible to integrate model layers into a front-end application without requiring an application to adhere by a strict set of rules (as it is in Angular). The benefits of structure can be exploited without dependency on it.

React – V (view) - <https://medium.com/@cabot_solutions/flux-the-react-js-application-architecture-a-comprehensive-study-fd2585d06483>

Vue – V (view)

Angular – MVVC

## Data Binding

Data binding is a property that maps underlying data to the view layer of the application. This attribute relates to the architecture of the framework being used. In the frameworks that are primarily the view layer of an application, data modeling is typically one way, from the model to the user interface. Vue and React follow this pattern; data has a downward flow, and in React child elements don’t have any effect on parent data. This can be advantageous because it keeps the application logic simple – data originates and the state of the data lives in only one place. The disadvantage of this attribute is that it makes it more difficult to directly change the state of data from user interaction. Other tools and features of the framework must be exploited to update the underlying data.

Angular differs from Vue and React in that it offers two-way data binding. Two-way data binding is a powerful feature that allows actions by the user to directly update and change the state of underlying data models. This becomes possible in Angular because of its model and view model layers underneath the view layer of the framework. Although this feature can be powerful in applications, it can lead to bugs that are difficult to test, find, and fix. It also adds extra logic to the application. For applications where data is primarily viewed, with no great need for user manipulation, this feature of the framework becomes less useful.

Since LogiSteps will be using the user interface to primarily display user data, two-way data binding is likely an unnecessary feature. One-way data binding will most likely provide a cleaner conduit for conveying data to the end user.

## Performance and Load Times

A noticeable metric for an end user of a web application is the performance of the web page and how quick the initial load time is. John Hannon did a study on the performance and load times for various front-end web frameworks, and he found that for keyed implementations of the frameworks, Vue beat out Angular and React significantly in both typical operations as well as initial load times. Angular and React appeared to perform similarly for typical operations, with React slightly edging out Angular in most metrics. When comparing initial load times, React was significantly faster than Angular, but still significantly slower than Vue.

The web application for LogiSteps is not expected to be an enormous web application performing complex algorithms. Due to this, there is an expectation that the web page should be fast and respond to users quickly. The page should also load quickly when a user navigates to the page to view their statistics. An analysis of this metric points to Vue as being the best option when picking from Angular and React.

## Syntax

Syntax is an important aspect of a framework, as it directly correlates to the learning curve associated with the framework; the more unfamiliar a syntax is, the longer it will take to begin development of the application. Based on research done by team Omicron, Angular and React present syntaxes unlike others found in other languages and frameworks. React represents all UI elements using JSX and rendering functions. JSX is an XML like syntax that is used within JavaScript. While JSX can be powerful, allowing JavaScript to be interwoven with UI, it means that using React would require learning a new language for expressing UI elements, presenting a steep learning curve to all members of Team Omicron. Vue, on the other hand, supports JSX, but allows UI elements to be expressed with standard HTML, CSS, and JavaScript.

Angular, like React, provides a rich syntax that is complex in nature. Additionally, Angular makes use of Typescript rather than JavaScript. This means the developers using Angular would need to be learn the Angular specific syntax for element expression, as well as Typescript for data modeling. This would also present a large learning curve to developers on Team Omicron. Vue keeps syntax simple, using HTML, CSS, and JavaScript for UI element expression, while also picking a select few expressions used in AngularJS for more powerful UI expression.

Vue provides a simple syntax for its core functionality, while expanding upon the strengths of other frameworks to allow for more powerful element expression and HTML based templates.

## Community Support

Another important aspect of a front-end framework is the amount of third party and community support that the framework has. This metric was checked by team Omicron by comparing Github statistics. Angular, React, and Vue has the following community activity.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Angular | React | Vue |
| Github Stars | 40,903 | 111,824 | 114,636 |
| Github Contributors | 731 | 1,241 | 193 |
| Stack Overflow Tags | 132,505 | 103,665 | 23,029 |

Table 1 - Community Support Statistics for Angular, React, and Vue.

From the quick metics collected in table 1, Angular and React appear to have massive following in the developer community. Angular has a significantly lower Github star count than the others, but this is likely a consequence of Angular migrating versions. Both Angular and React also have a large amount of questions on stack overflow, meaning that many potential questions likely have answers available on stack overflow. Vue has a significantly lower number of Github contributors and stack overflow questions. This means that it may be more difficult to find answers to questions during the development process.

## Other

Angular, React, and Vue all provide several other features as well which have been considered when performing technology research. All three frameworks provide view templating routing, and expressive element expression. When considering framework transfer size, Vue wins out, beating both React and Angular based on a study performed by Jacek Schae.

## Conclusion

All 3 front end frameworks provide similar speed performance, however, Angular typically has a longer initial load time. In terms of learning curve, both Angular and React have steep learning curves, with Angular having complex syntax, and React using JSX for UI development. Vue offers the smallest learning curve by using native javascript, CSS, and HTML to implement rich, expressive user interfaces. Vue also has advantage in terms of size. Due to Angular's heavy use of libraries, complex syntax, and other features, it becomes bloated. Both Vue and React are much smaller in size, with libraries being included as they are required. Angular provides more structure, but this structure can be restrictive at times; React and Vue allow for much greater flexibility. The flexibility of Vue and React can sometimes make it more difficult to start projects, but Vue offers a powerful CLI that helps setup projects with various configuration settings available. Vue is currently less popular than Angular and React - leading to a smaller supporting community and a smaller collection of libraries, but Vue is quickly growing in community size and popularity. In the end, Vue likely offers the best option for our project due to its small size, and smaller learning curve. Team Omicron’s front-end UI is meant to be simple, providing a portal for viewing statistics. The added size of Angular, and complexity of React and Angular most likely make them a less beneficial option for our project.

# Back-End Framework

Back-end frameworks are numerous, with each build upon differing languages. A backend framework provides the API and processing of user data, without a need to be closely coupled with the user interface. Some of the most popular languages used for writing web application back ends are NodeJS (server-side JavaScript), Python, Ruby, PHP, C#, Java, and more. Back-end frameworks use these underlying languages to build a platform for rapid development and prototyping. The features of a programming language often make one more optimal than others, depending on the needs of the web application. When researching and choosing a backend framework, this, plus other factors such as application structure, database support, and libraries were considered and compared.

After performing technology and concept research, the best potential back-end frameworks for the LogiSteps application were narrowed down to Express, Django, and Flask. All 3 are very popular frameworks used to implement web applications in NodeJS and Python. Some of the advantages and disadvantages of each are discussed in the following subsections.

## Express

Express is a framework that uses nodeJS for development. NodeJS is an event driven, non-blocking I/O language, which makes it ideal for applications driven by user interaction and events. LogiSteps is predicted to be mostly a data driven application, with I/O events occurring in regular, infrequent intervals. Due to this, NodeJS is most likely not an idea language for writing the LogiSteps web application.

Express is also a lightweight framework that comes with support for both SQL and noSQL databases, allowing easier integration of data for applications. Additionally, Express offers features such as routing, view caching, and middleware chaining. Essentially, express is a minimalist web framework that has as little functionality as possible, while providing a series of middleware function calls for executing code, changing request and response objects, ending the request/response cycle, and calling the next middleware in the pipeline. This gives applications using Express a log of flexibility, without bogging down an application in express specific code. Additionally, by using express and NodeJS, developers have access to NPM – the largest open source library in the world.

## Django

Django differs from express in a multitude of ways, providing more structure, and utilizing a completely different programming language. Django is a framework that helps build back-end applications using Python, a common language used for data driven applications. While Express was a framework with little requirements for structure, Django enforces rigid application structure, following the MVC pattern for representing objects and state. While a stricter structure requirement can limit the possibilities of an application, Django provides several features that frameworks such as Express and Flask do not provide. In particular, Django has a relational database interface that is build into the framework. This provides access to a build in object-relational mapper, support for SQL database managers, the ability to quickly switch between different DBMS, and allows the application to be closely coupled with the backend database.

Additionally, Django advertised several more features that frameworks such as Express and Flask do not advertise. Some of these features include built in Security (SQL injection, cross-site scripting, cross-site request forgery, clickjacking), user authentication, administrative controls, scalability, and more. Django provides an all-in-one solution, and this helps assist in quicker application development. In frameworks such as Express and Django, these features may only be available through third party plugins and libraries. Django is designed for building data driven applications, such as the web application for LogiSteps.

## Flask

Flask is a back-end framework for applications written in Python (similar to Django), but with limited functionality (similar to Express). Flask was designed to provide a web framework that is focused on simplicity, minimalism, and fine grain control. As a result, the Flask framework has a smaller community than Django, but is less restrictive. Further, due to Flask’s simplicity, it does not contain an object-relational mapper, which allows for greater flexibility, but more work implementing data driven applications. While Flask is considered a “microframework”, it still provides some of the fundamental features that web frameworks provide, such as minor security, routing, middleware, and scalability.

## Conclusion

Express, Django, and Flask are 3 of the most popular frameworks used for creating backend web services. Choosing between them requires knowledge of the task at hand. Express is built on NodeJS and is designed for efficient event driven programs. Express also integrates well with NoSQL databases such as MongoDB, which may be beneficial for storing complex data structures. Django and Flask, on the other hand, are built on Python, which offers a core API much more equipped for data processing. While Django and Flask share a common underlying programming language, they are quite different. Django offers a "batteries included" approach, providing several tools for quick and efficient prototyping. Additionally, Django specializes in providing an interface for integrating relational databases into an application. This reduces the effort required for performing CRUD operations and querying for data. Flask on the other hand, offers a bare-bones, minimalism approach. This offers inherit benefits, such as increased freedom and size, but can make it more difficult to rapidly prototype and develop. This provides increased flexibility for the type of database and management systems being used for storing and querying data, but increases the effort required for integrating databases into the application. Our project will mostly be a data driven application, serving data visualization views to clients. Because of this, it makes more sense for our back-end technology to utilize the Django framework, which assists in easy data mapping and database querying using the relational object mapper. Using this, our relational databases can be easily integrated into the application, Python can be used for efficient data manipulation, and other Django features such as user authentication, security, and more can be used to support development.

## Database

The LogiSteps product will generate data pertaining to a user’s movement originating from their shoe. Data will be relayed from the shoe’s sensors, to a microcontroller, relayed through a mobile application, to a web server, which will place the data into long term storage. The medium of the long-term storage will be a database able to perform efficient storage and querying on timeseries data. Application data is being considered timeseries data since data will likely be indexed and queried using time in most cases.

Choosing the right database for long term storage and querying of application data is highly dependent on the structure of the data, it’s rate of generation, how the data is being used, and other factors. For LogiSteps, the database needs to scale well as the number of rows increases with little or no deletions. Additionally, the LogiSteps application will need to query large amounts of data based on the time that the data was generated. A good database for the LogiSteps application is one which allows for easy, efficient querying of data indexed optimized for timeseries data.

Some of the most popular database choices that exist are MongoDB, MySQL, PostgreSQL, and more. When researching databases, they were compared based on their ability to efficiently store timeseries data and retrieve subsets of the data through powerful commands.

## SQL Databases

SQL databases, such as MySQL and PostgreSQL, are advantageous in that they make use of powerful, long established standards for querying and manipulating data. As a result, users of the database can access and view subsets and supersets of the database, helping answer analytical questions and present applicable data to a user. Additionally, SQL databases are accessed using a standard SQL language, which can make it easier to access data without having to write a lot of code. While SQL databases can excel in querying and keeping consistent data, they often do not scale well. An experiment done by a TimeScale engineer showed that the insertion rate for a PostgreSQL database decreased at a near linear rate when increasing the number of records in the database, beginning to level out near 400 million rows. This could present a problem as LogiSteps scales up, increasing the number of users logging data into the database. Another disadvantage of using a SQL database is the inability to store complex data structures such as objects; a SQL database requires all columns to be a single scalar attribute. While this likely won’t impact LogiSteps, the inability to scale with dataset size presents a problem.

## NoSQL Databases

A popular database alternative that scales well are noSQL databases such as MongoDB. noSQL databases have taken off in popularity in previous years due to their various advantages over SQL databases such as

* Ability to handle well structured, semi-structured, and unstructured data
* Quick prototyping and development (structure does not need to be well defined)
* Ability to store objects
* Efficient, scale-out architectures

While these advantages can be beneficial, LogiSteps only needs the ability to scale up, as well as quick development and prototyping. In addition to the advantages of a noSQL database,noSQL present several disadvantages as well. Most notably, noSQL databases lack many of the business intelligence and analytic features that SQL databases are able offer or paired with. While MongoDB provides a service for timeseries data, it lags behind more specialized solutions designed for timeseries data.

## TimescaleDB

Timescale is a SQL based database which is designed specifically for timeseries data. Specifically, Timescale is build on PostgreSQL, and as a result, is compatible with any tools that work with PostgreSQL. Being built on a SQL database allows Timescale data to benefit from the same advantages of traditional SQL databases, but it is designed to scale like a noSQL database. Essentially, Timescale attempts to bridge the gap and offer some of the most notable benefits of a traditional SQL database and a noSQL database. To achieve scalability similar to that of a noSQL database, Timescale partitions data into time-based chunks which allows for faster performance at scale. Some of the most notable advantages of this database are

* SQL database designed to be compatible with SQL compatible tools
* Scales like a noSQL database
* Comes with built in timeseries specific analytical functions
* Abstracts data as one continuous table for simplified queries.
* Can be managed like a traditional PostgreSQL database
* TimescaleDB integrates directly into the PostgreSQL query planner and execution engine

## Conclusion

LogiSteps aims to provide a user interface for customers to interact with and view their fitness data. As a result, a powerful query language will be needed that will make it possible to easily calculate aggregate data and find subsets of their data based on certain parameters. Additionally, the database used to store user data will likely need to interface with a third-party visualization library or an object relational mapper. Such tools are often designed to interface with SQL databases due to its standardization. These needs are well suited by a SQL database, however, the SQL database will likely no scale well for LogiStep’s timeseries data. A noSQL database would likely be best for scalability but would sacrifice the benefits of the SQL database. To fulfill both requirements, TimescaleDB appears to be the best option for LogiSteps. TimescaleDB is a database designed specifically for timeseries data that is built to scale, but is built on, and takes advantage of, the SQL properties in PostgreSQL. Other timeseries databases with similar features, such as influx, may be used if needed.

## Web Hosting

One of the final pieces for creating the web application is hosting the application publicly so that any user across the world can access it. While it is technically feasible to host a dynamic website on a development PC or server, it would be vulnerable to a multitude of security concerns, and would not be able to scale well. To solve this issue, there are a multitude of web hosting services that allow a developer to host their database, web server, and website publicly. Some of the top competitors in this industry are Google Cloud Platform, Amazon Web Services, and Heroku. All 3 of these options offer free versions for active development.

Google Cloud Platform and Amazon Web Services are similar in that they offer infrastructure for hosting web applications as a service. This allows a developer to choose the correct plan based on the needs of the application, and offers limited free trials for initial development. Amazon web services have been around the longest of all 3 options (Heroku actually uses Amazon Web Services), but often it can be difficult to predict the price for hosting a web application on Amazon. Additionally, based on experimentation, Amazon Web Services appear to be the most difficult of the three options for navigating and finding all features. AWS claims to have free tier with the following features:

* Amazon Cognito – Mobile user identification and synchronization
* Amazon DynamoDB – 25 GB noSQL database
* Amazon EC2 – 750 hours of cloud compute capacity per month
* AmazonMQ – 750 hours of broker service for Apache ActiveMQ per month
* AmazonRDS – 750 hours of Managed relation database service for MySQL, PostgreSQL, etc.
* Elastic Load Balancing – 750 hours of traffic distribution

The Google Cloud Platform also has a free tier that gives a 12-month $300 credit to developers. Google Cloud offers many of the same features as AWS, but with a more intuitive interface for developers. One of Google Cloud’s most valuable features is its Compute Engine, which provides scalable, high performance virtual machines. Using this, custom software can be installed and used to host a web application. Google Cloud also offers services for load balancing applications, removing a necessary piece of deployment. Additionally, Google Cloud offers several predefined production environments that a web application can be deployed to, removing the necessary administrative tasks involved with a custom web stack. While Google offers a $300 credit to its customers, the following services are always free

* Google App Engine – platform for building scalable web applications
* Highly-scalable noSQL databases
* Google Compute Engine – scalable, high performance virtual machines (web hosting too)
  + 30 GB of HHD
  + 1 f1-micro instance per month
* Google Cloud Pub/Sub – real-time and reliable messaging and streaming data
* Cluster Management
* Google Stackdriver – monitoring, logging, and diagnostics for applications on Cloud Platform
* GCP Marketplace – pre-configured free production grade solutions

**Pricing**

Google cloud will likely be free for the duration of Senior Design due to the requirements of the web application. If LogiSteps were scaled up, pricing would be heavily dependent on the number of users. A server with 4 cores and 15GB of memory would cost $97.09 per month, and a server with 1 core and 3.75GB memory would cost $24.27 per month.

Heroku is a cloud application platform that supports building, deploying, and managing apps. Heroku allows apps to be run inside of what they call dynos, which are fully managed runtime environments for applications. Heroku is designed to be as easy as possible for developers to easily, quickly, and frequently deploy applications; Heroku then provides a dashboard for managing all applications. Heroku helps handle scale, similar to Amazon Web Services and Google Could Platform. Heroku offers several different database solutions such as Heroku PostgreSQL but requires extensions and add-ons to work with other database systems, making it slightly more difficult to deploy data driven applications. Heroku is likely an excellent source for active development but appears to lack the extensive functionality that can be achieved using a Google Cloud Compute virtual machine. Heroku also sleeps after inactivity for free accounts, leading to large latency for requests made when the dyno is sleeping.

**Pricing**

* Free - $0
  + Core platform features
  + Sleeps after 30 minutes of inactivity
  + 512 MB RAM | 1 web/1 worker
* Hobby Level - $7/dyno/month (pay for time used to the second)
  + Core platform features
  + Never sleeps
  + Free SSL & Automated Certificate Mangement
  + 512 MB RAM | 10 Process Types
* Standard - $25 - $500/dyno/month (pay for time used to the second)
  + All hobby features+
  + Horizontal scalability
  + Threshold alerts
  + Preboot
  + 512 MB RAM or 1 GB RAM
* Performance - $25 - $500/dyno/month (pay for time used to the second)
  + All standard features+
  + Dedicated
  + 2.5GB RAM or 14GB RAM
  + Infinite process types

## Conclusion

For the development of LogiSteps, Google Cloud Platform will likely be the best options. GCP not only offers a free plan to begin development, but it also offers a robust administrative web interface for managing the deployment. Google Cloud also presents a more straightforward pricing model which coincides with a pricing calculator to help determine cost of hosting, depending on the needs of the web application. Additionally, Google’s compute Engine allows a customized web stack to be developed and deployed without having to worry about compatibility. This presents a huge advantage over services such as Heroku. It is worth noting that there are several other web hosting services which appear to be extremely cheap up front (such as hostgator), but these services often attempt to lock customers into long term deals and offer limited hosting options, severely limiting the web stack.

<https://blog.timescale.com/how-to-store-time-series-data-mongodb-vs-timescaledb-postgresql-a73939734016>

<https://www.mongodb.com/scale/advantages-of-nosql>

<https://blog.timescale.com/timescaledb-vs-6a696248104e>

<http://flask.pocoo.org/docs/1.0/>

<https://www.djangoproject.com/start/overview/>

<https://www.quora.com/What-are-the-advantages-and-disadvantages-of-SQL>

<https://expressjs.com/en/guide/using-middleware.html>

<https://medium.freecodecamp.org/a-real-world-comparison-of-front-end-frameworks-with-benchmarks-2018-update-e5760fb4a962>

<https://javascriptreport.com/js-frameworks-fastest/>

<https://012.vuejs.org/guide/>

[https://aws.amazon.com/free/?awsf.Free%20Tier%20Types=\*default](https://aws.amazon.com/free/?awsf.Free%20Tier%20Types=*default)

<https://cloud.google.com/free/>

<https://aws.amazon.com/ec2/pricing/on-demand/>

<https://cloud.google.com/compute/pricing>

<https://www.heroku.com/pricing>