Dyalog North America Meetup, 11 April 2024

APL and Web Services

Brian Becker
APL Tools Architect
Dyalog, LTD

Web Services

- What
- Consuming using `HttpCommand`
- Providing using `Jarvis`
- Ask questions!

What is a Web Service?

- Let's ask a web service what a web service is...

What is a Web Service?

```
      c←OpenAI.Chat.Completion 'you are a helpful assistant' 'what is a web service?'
      c.Run
[rc: 0 | msg:  | HTTP Status: 200 "OK" | ≠Data: 1 (namespace)]
      c.Conversation
```

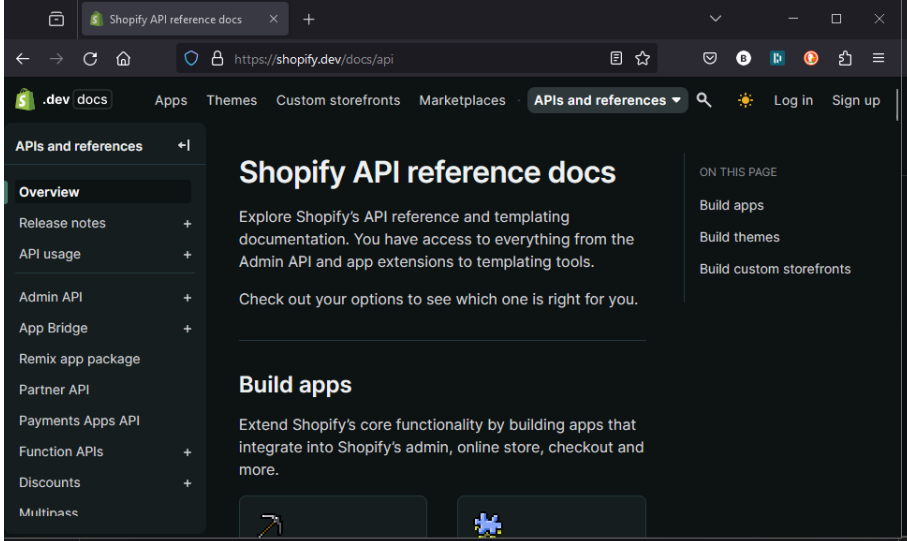
system	you are a helpful assistant
user	what is a web service?
assistant	A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It allows different applications to communicate with each other over the internet using standard protocols such as HTTP. Web services typically provide a way for different systems to exchange data and perform actions without needing to know the internal workings of each other. They are commonly used for integrating different systems, sharing data, and automating business processes.

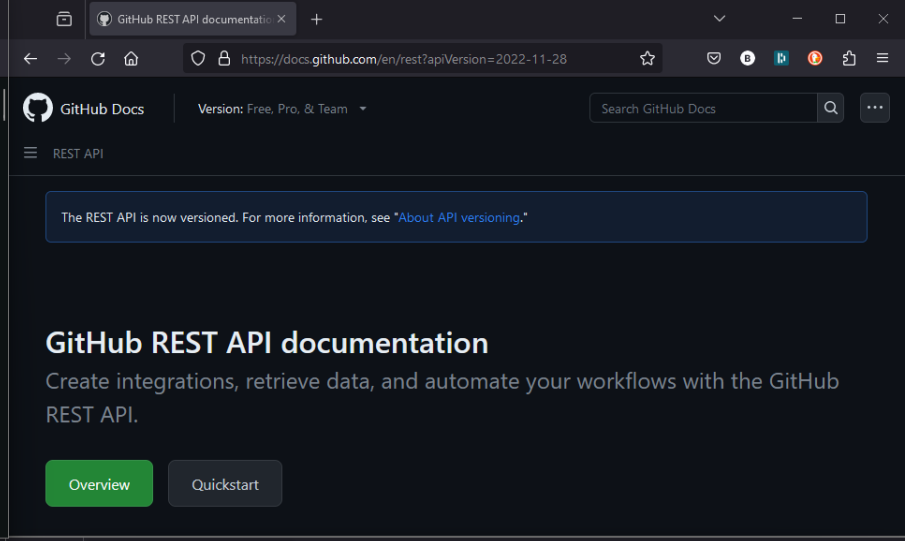
What is a Web Service?

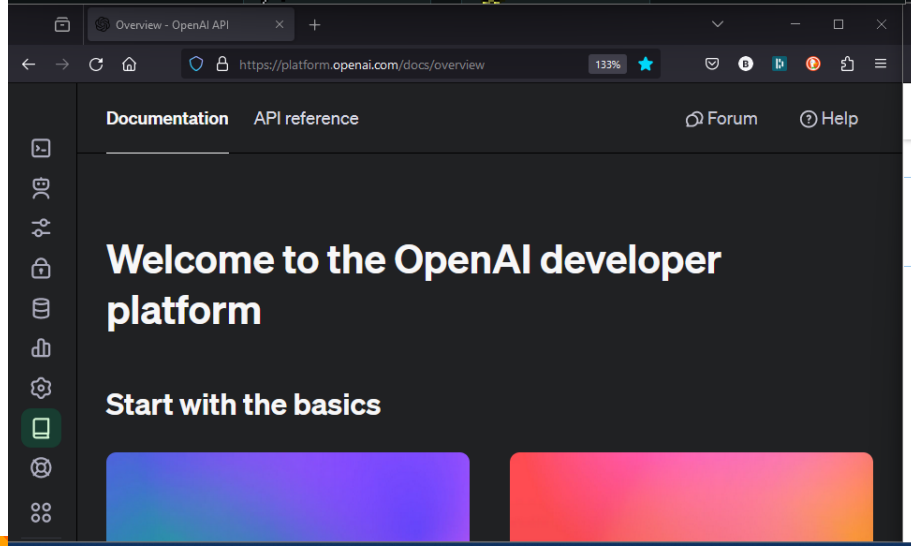
A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It allows different applications to communicate with each other over the internet using standard protocols such as HTTP. Web services typically provide a way for different systems to exchange data and perform actions without needing to know the internal workings of each other. They are commonly used for integrating different systems, sharing data, and automating business processes.

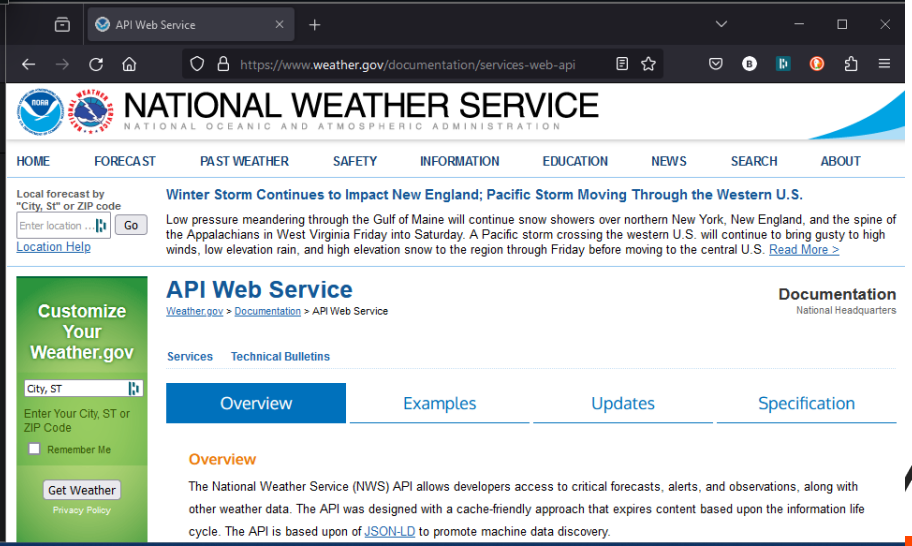
Web Services

- Machine to machine
- Use a standard protocol (HTTP, HTTPS)
- Expose an Application Programming Interface (API)


Shopify API reference docs
https://shopify.dev/docs/api
.dev docs
Apps
Themes
Custom storefronts
Marketplaces
APIs and references
Log in
Sign up
APIs and references
Overview
Release notes
API usage
Admin API
App Bridge
Remix app package
Partner API
Payments Apps API
Function APIs
Discounts
Shopify API reference docs
Explore Shopify's API reference and templating documentation. You have access to everything from the Admin API and app extensions to templating tools.
Check out your options to see which one is right for you.
ON THIS PAGE
Build apps
Build themes
Build custom storefronts
Build apps
Extend Shopify's core functionality by building apps that integrate into Shopify's admin, online store, checkout and more.


GitHub REST API documentation
https://docs.github.com/en/rest?apiVersion=2022-11-28
GitHub Docs
Version: Free, Pro, & Team
Search GitHub Docs
REST API
The REST API is now versioned. For more information, see "About API versioning."
GitHub REST API documentation
Create integrations, retrieve data, and automate your workflows with the GitHub REST API.
Overview
Quickstart


Overview - OpenAI API
https://platform.openai.com/docs/overview
133%
Documentation
API reference
Forum
Help
Welcome to the OpenAI developer platform
Start with the basics


API Web Service
https://www.weather.gov/documentation/services-web-api
NATIONAL WEATHER SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
HOME
FORECAST
PAST WEATHER
SAFETY
INFORMATION
EDUCATION
NEWS
SEARCH
ABOUT
Local forecast by "City, St" or ZIP code
Enter location ...
Go
Location Help
Winter Storm Continues to Impact New England; Pacific Storm Moving Through the Western U.S.
Low pressure meandering through the Gulf of Maine will continue snow showers over northern New York, New England, and the spine of the Appalachians in West Virginia Friday into Saturday. A Pacific storm crossing the western U.S. will continue to bring gusty to high winds, low elevation rain, and high elevation snow to the region through Friday before moving to the central U.S. Read More >
Customize Your Weather.gov
City, ST
Enter Your City, ST or ZIP Code
Remember Me
Get Weather
Privacy Policy
API Web Service
Weather.gov > Documentation > API Web Service
Documentation
National Headquarters
Services
Technical Bulletins
Overview
Examples
Updates
Specification
Overview
The National Weather Service (NWS) API allows developers access to critical forecasts, alerts, and observations, along with other weather data. The API was designed with a cache-friendly approach that expires content based upon the information life cycle. The API is based upon of JSON-LD to promote machine data discovery.

Web Services

- Machine to machine
- Use a standard protocol (HTTP, HTTPS)
- Expose an Application Programming Interface (API)
  - Server doesn't necessarily know what the client is
    - Web page/JavaScript, Phone App, C#, .NET, APL
  - Client doesn't know the server's internal workings

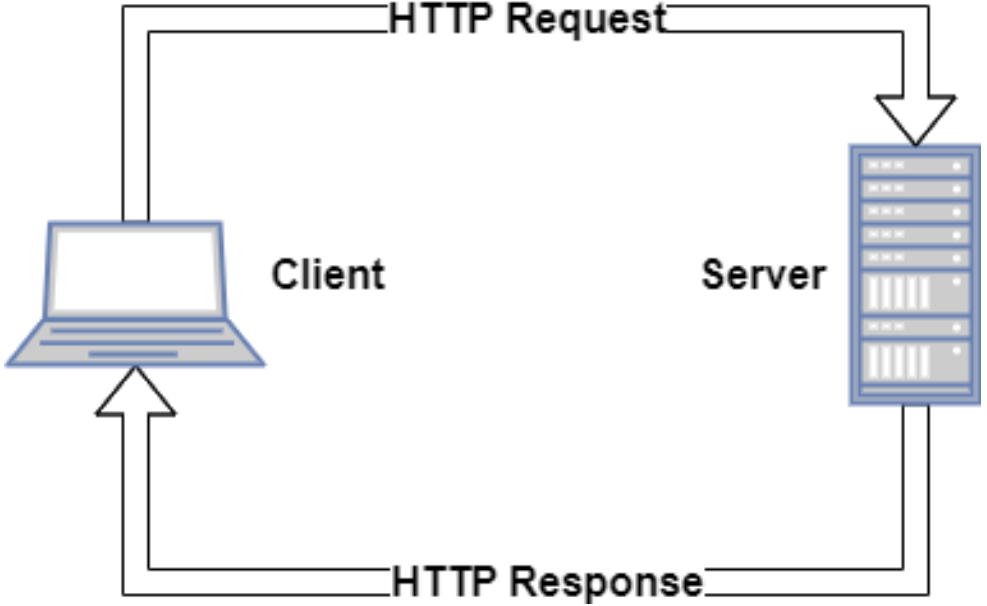
HTTP Communications 101

- HTTP is a request-response protocol
- A client sends a request to a server
- The server receives the request
- The server runs an application to process the request
- The server sends a response back to the client
- The client receives the response

Client Examples:
A web browser,
`HttpCommand`, cURL,
JavaScript, Python

Server Examples:
IIS, Apache, Nginx,
`Jarvis`,
`DUI/MiServer`

HTTP Communications 101



Client Examples:
A web browser,
`HttpCommand`, cURL,
JavaScript, Python

Server Examples:
IIS, Apache, Nginx,
`Jarvis`,
`DUI/MiServer`

Web Service API Usage Patterns

- Find the documentation
- Determine if you need authentication credentials
  - Register and obtain an API key
  - Many web services provide a free, rate-limited, level of access
- Construct and send your request
- Process the service's response

HttpCommand

`HttpCommand` is a utility that is well-suited to enable the APLer to interact with web services because it:

- Allows you to specify an HTTP request in a manner that is conducive to an APLer
- Sends a properly formatted HTTP request to the server
- Receives the server's response
- Decomposes the response in a manner that is conducive to an APLer
- Minimizes the need for you to learn a lot about HTTP

Typical `HttpCommand` Usage

- Create a new `HttpCommand`
- Specify:
  - HTTP Method (GET, PUT, POST, etc)
  - URL (https://api.github.com/repos)
  - Any additional necessary headers (Content-Type, Authorization, etc)
  - Any payload
- Send the request
- Examine and process the response

Example

```
      h          ← HttpCommand.New ''
      h.URL      ← 'dyalog.com'
      h.Command ← 'get'
      ⊢ r        ← h.Run
[rc: 0 | msg:  | HTTP Status: 200 "OK" | ≠Data: 24139]
```

Shortened Example

```
      h          ← HttpCommand.New 'get' 'dyalog.com'
      ⊢ r        ← h.Run
[rc: 0 | msg:  | HTTP Status: 200 "OK" | ≠Data: 24139]
```

Demo Time...

- Grabbing a web page
- Using a REST web service (GitHub)
- Using a non-REST web service (OpenAI)

JSON AND REST SERVICE

JARVICE

JARVIS

Jarvis

Much like `HttpCommand`, `Jarvis` is designed with the APLer in mind:

- Client requests are POST requests with JSON payloads
- Web Service Endpoints are APL functions
- They take an APL array as a right argument
- They return an APL array as their result
- `Jarvis` handles all the rest

More Demos

- Web service in 5 Minutes
- Limit endpoints
- Authentication
- Sessioning/State maintenance

Think about it...

- Anything that can "speak" HTTP can talk to your web service
  - Web page
  - Phone app
  - Another process
  - Even HttpCommand ☺

Security

- If you are sending sensitive content - use HTTPS
- Jarvis supports HTTP Basic authentication out of the box
- You can implement whatever authentication makes sense
- Sessions are independent and cannot see one another, unless you do so in your application code.

Performance

- Jarvis itself has very little overhead
- Performance may be impacted by
  - Number of requests
  - Size of requests
  - Application code
  - How much "state" is maintained on the server and for how long

Scalability

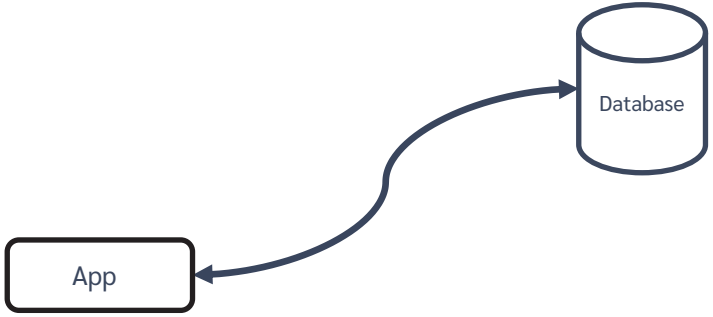
At Dyalog'22, Morten and Brian ran a half-day workshop. We:

- took an application
- made it a Jarvis web service
- ran it in a Docker container
- moved it to the cloud (AWS)
- scaled it
- load-balanced it
- ran it securely using HTTPS

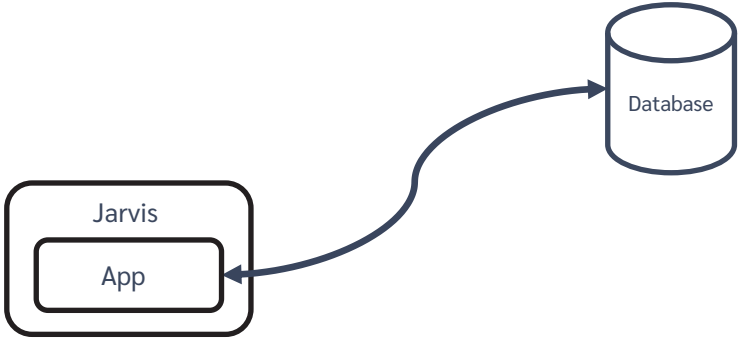
The Plan Visualized... (from Dyalog'22)

The Plan Visualized...

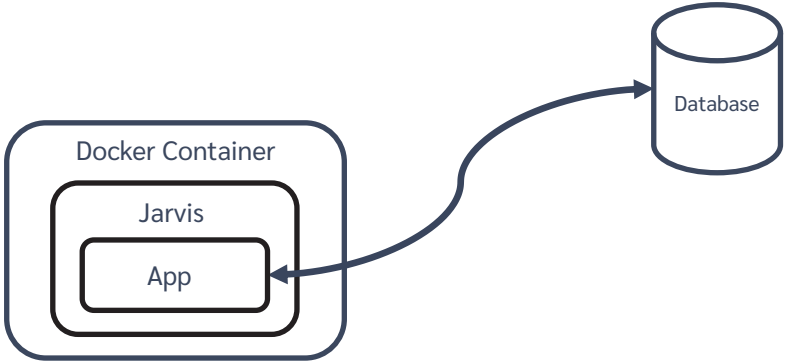
In the beginning, there was an Application...



Run the app as a service

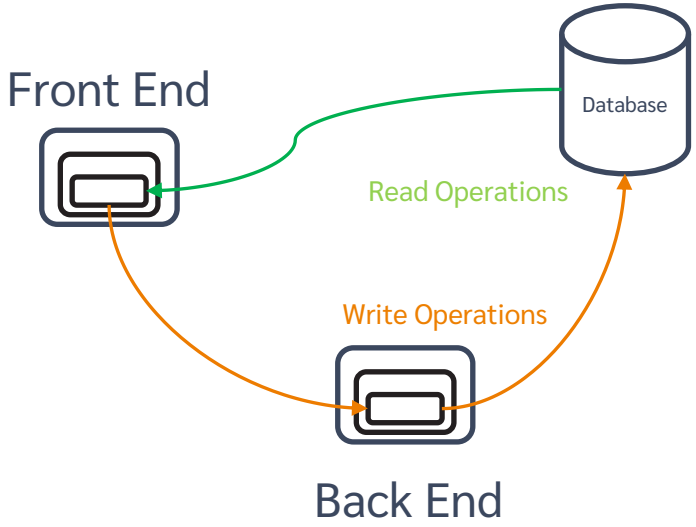


Run it in a container

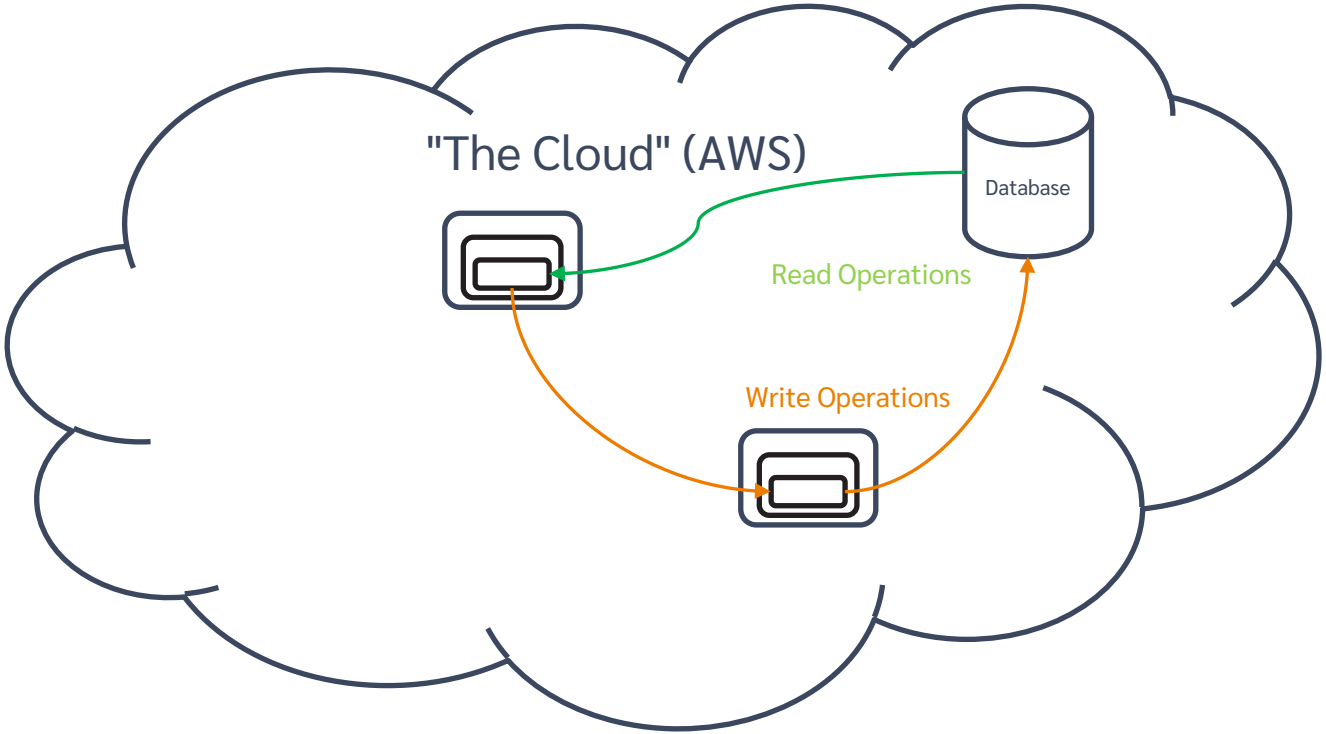


Split into Front and Back Ends

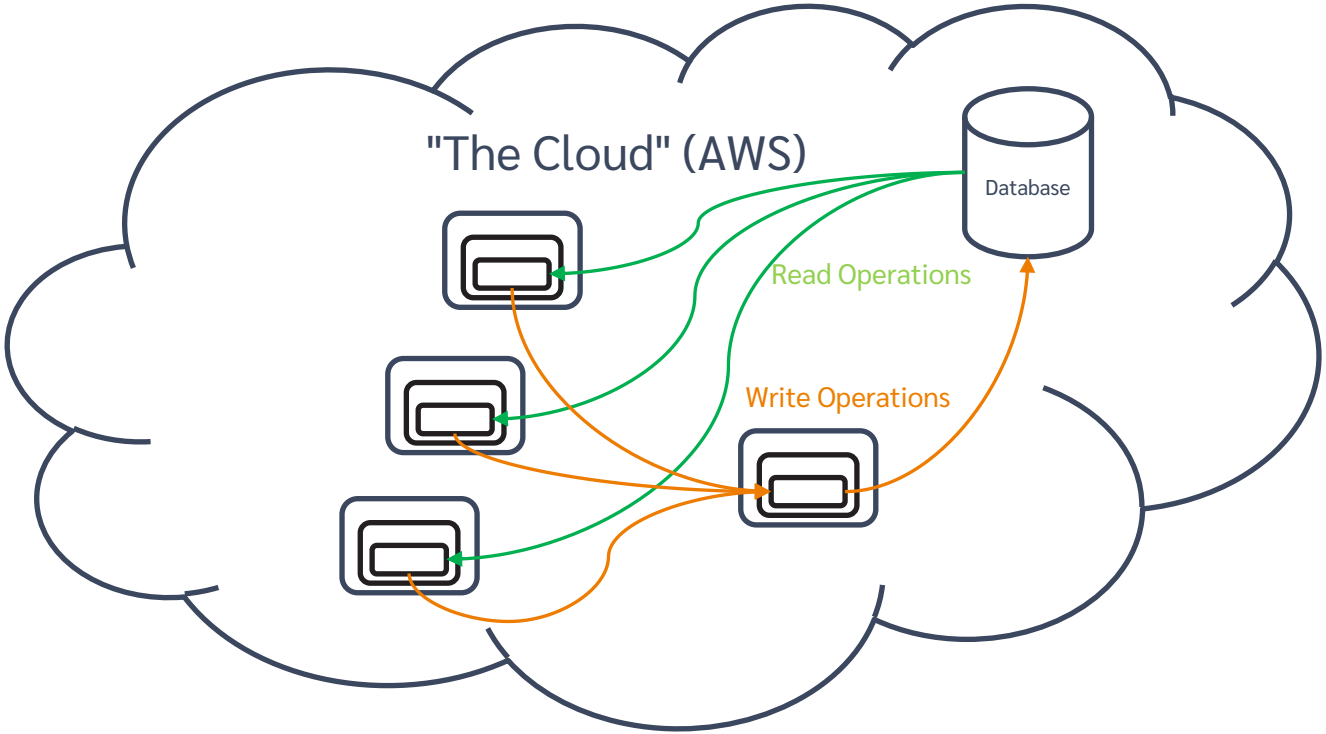
We'll call this "Two-Tier"



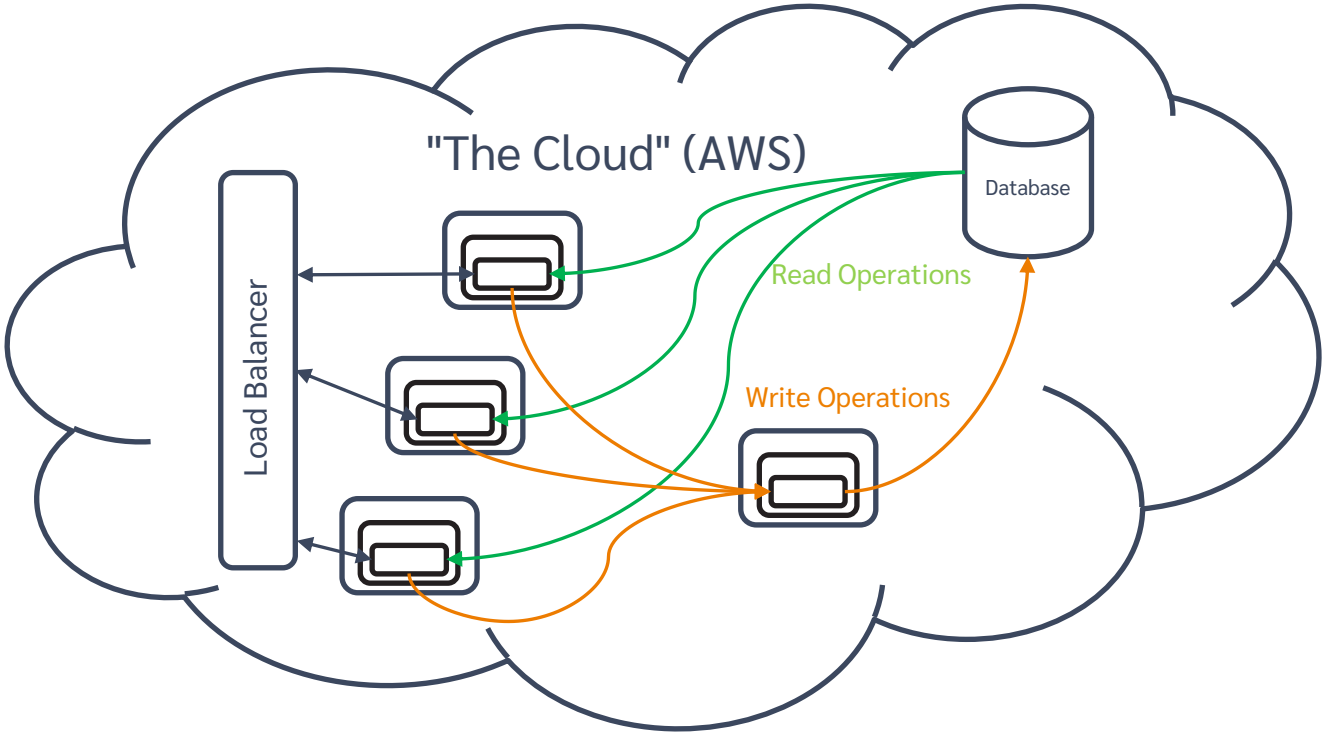
Try it in the cloud



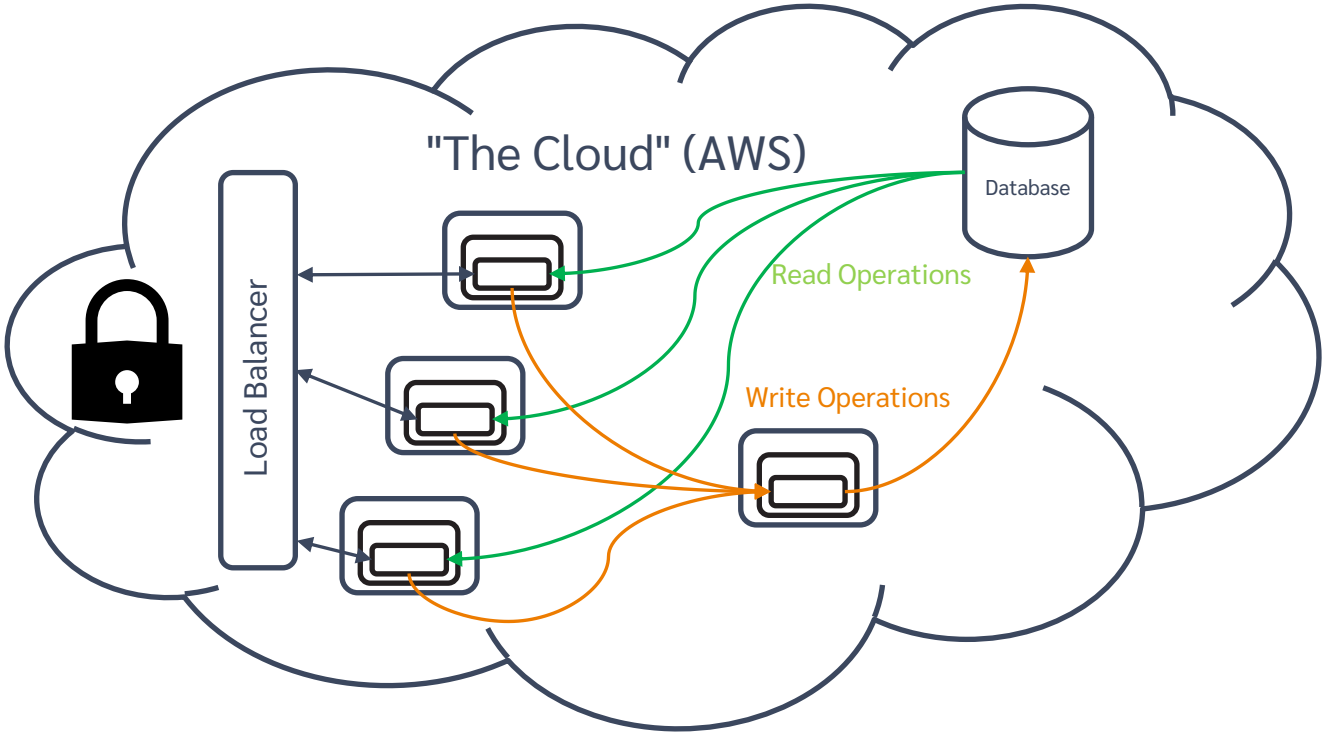
Scale it up



Load balance it



Secure it


"The Cloud" (AWS)
Database
Load Balancer
Read Operations
Write Operations

Jarvis and REST

- Jarvis can serve REST web services
  - Instead of "functional" endpoints, you write a function for each HTTP method your service will support
  - Each function will parse the requested resource and take appropriate action
- To me as an APLer, the JSON paradigm seems more natural
- If you have an interest in the REST paradigm, ask me

In the Jarvis Pipeline

- Finish the documentation!
- Add more logging and management capability
- JAWS – Jarvis And Web Sockets

Questions?