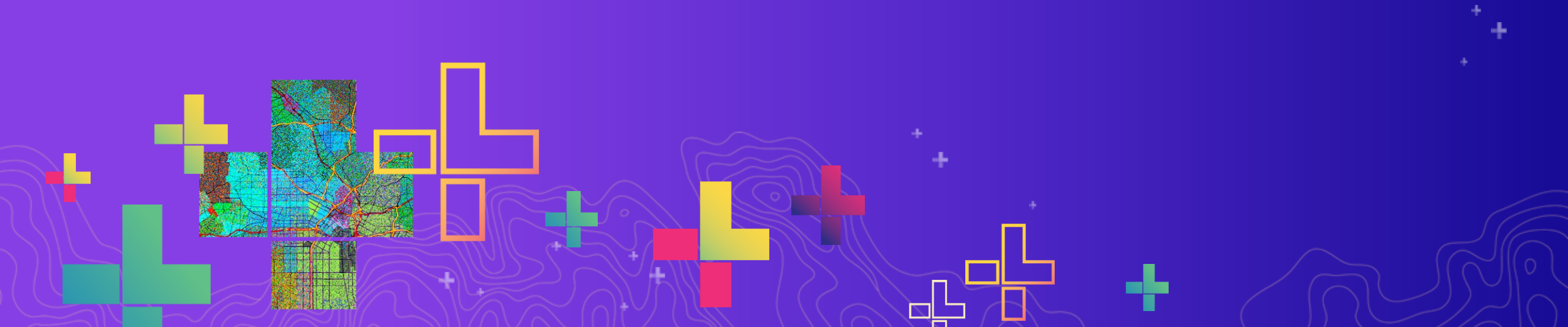




# Use Koop to Connect Any Data Source to ArcGIS

Haoliang Yu, Software Engineer  
Rich Gwozdz, Software Engineer  
ArcGIS Hub | Koop

2020 ESRI DEVELOPER SUMMIT | Palm Springs, CA



**Koop is an open-source webserver used to extract and transform geospatial data on-the-fly.**





# Overview

- **A RESTful web server written in Node.js that performs**
  - **Geodata extraction and transformation**
  - **On demand and on the fly via HTTP requests**
  
- **An engine with plugin architecture**
  - **Geoservices output for ArcGIS clients**
  - **Published plugins available on npm**
  - **Allows custom plugins for your own specific needs**
  
- **Apache license 2.0: free of use**



# When to use Koop?

- **You want to use data in ArcGIS clients**
  - Source data format is tied to the business application
  - Source data format is controlled by a 3rd party
  - No existing SDE or ETL system support
- **You want to expose a source data in multiple formats**

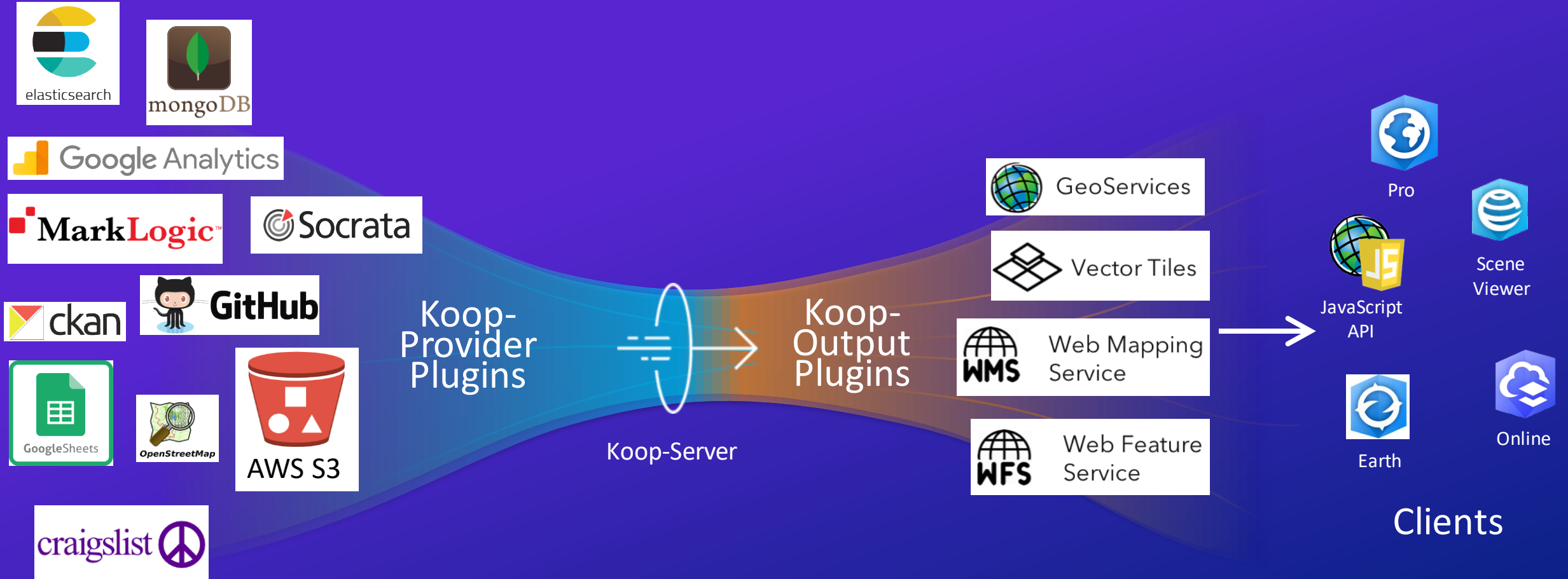


# Koop Concepts

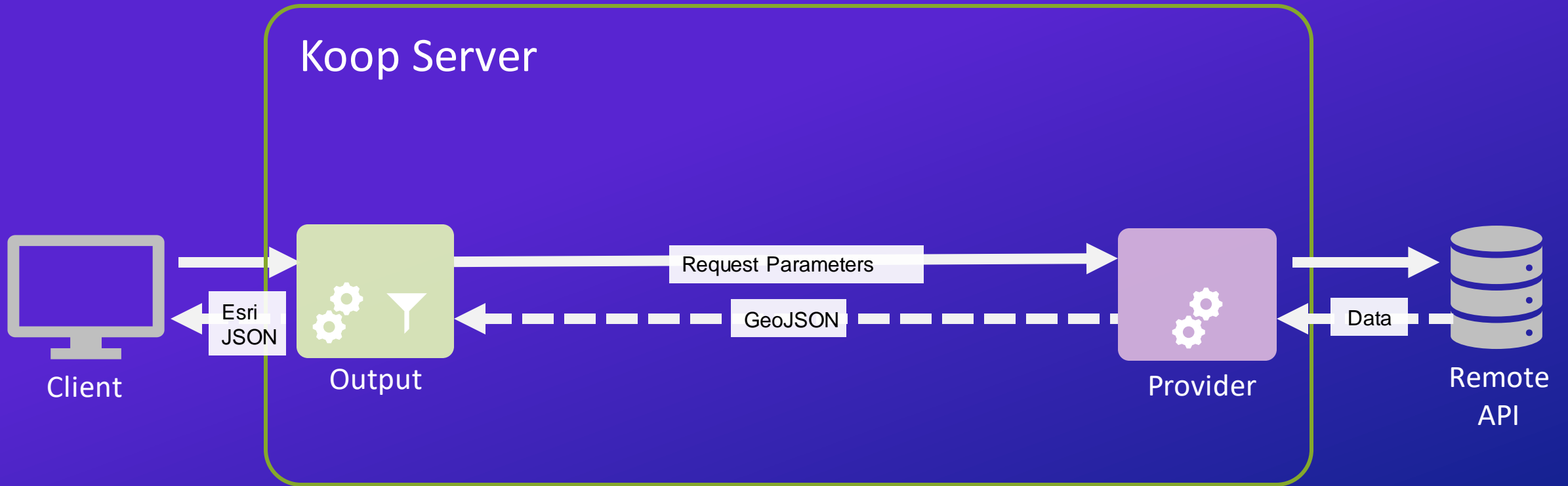
- **Koop core**
  - A wrapper around a light-weight Express.js server instance
  - Handles registration of plugins
  - Manages data flow
- **Provider**
  - A plugin to connect with data source(s)
- **Output**
  - A plugin to transform data into specific output formats
- **Cache**
  - A plugin to stores data initially requested from the provider



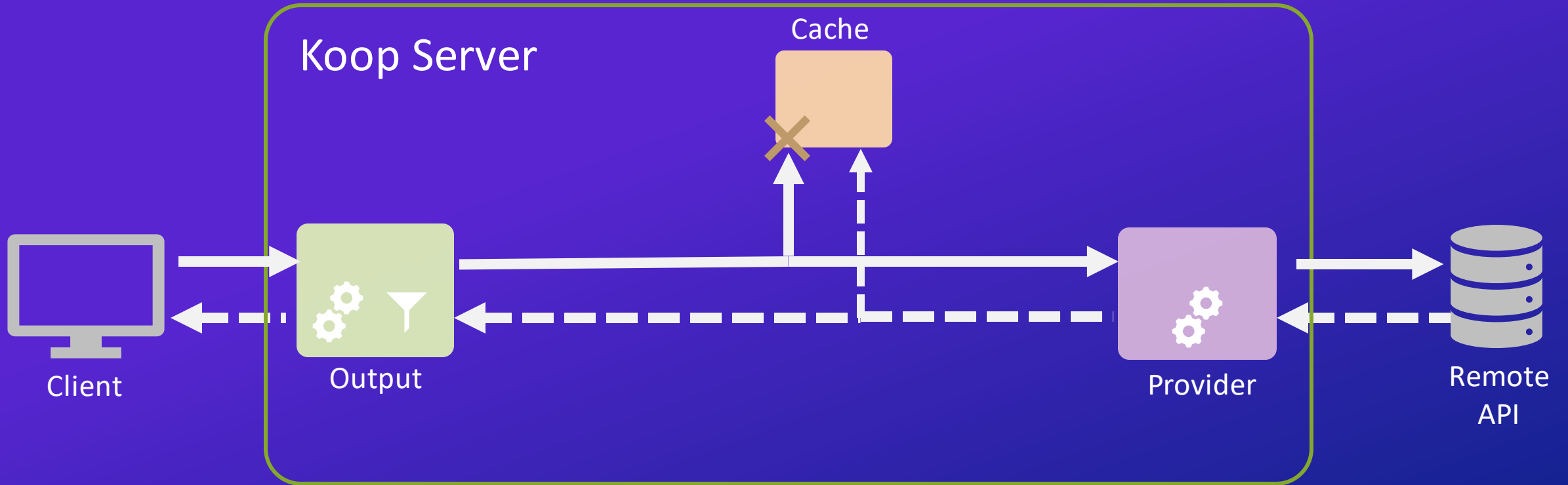
# Koop plugins



# Koop request lifecycle

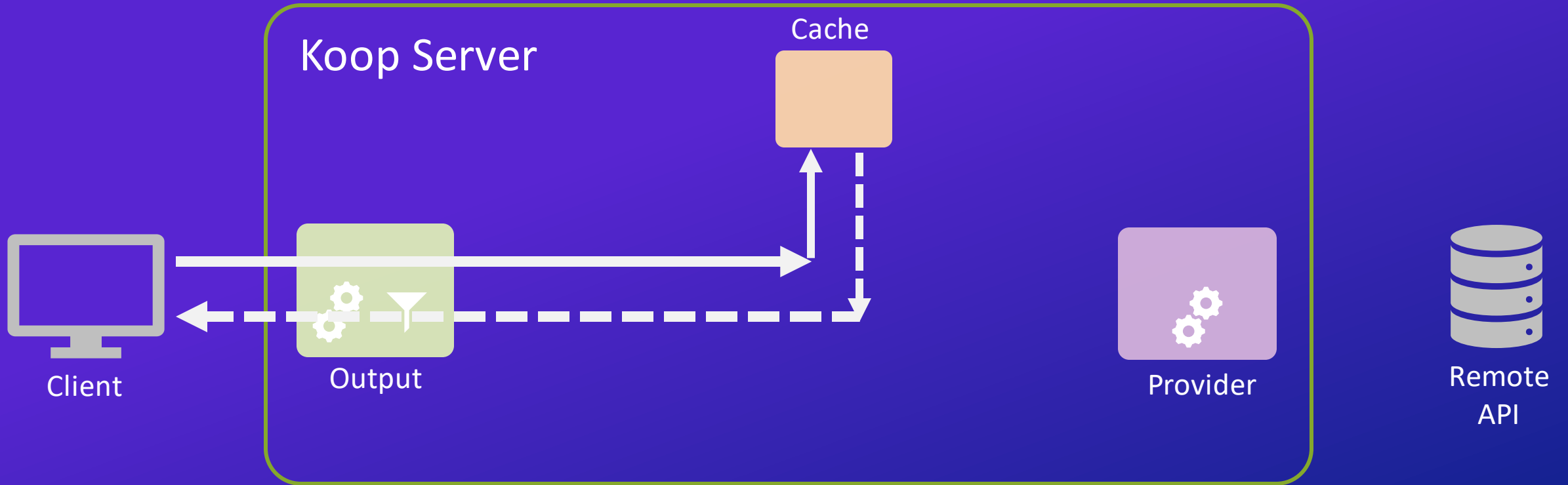


# Koop request lifecycle – with cache

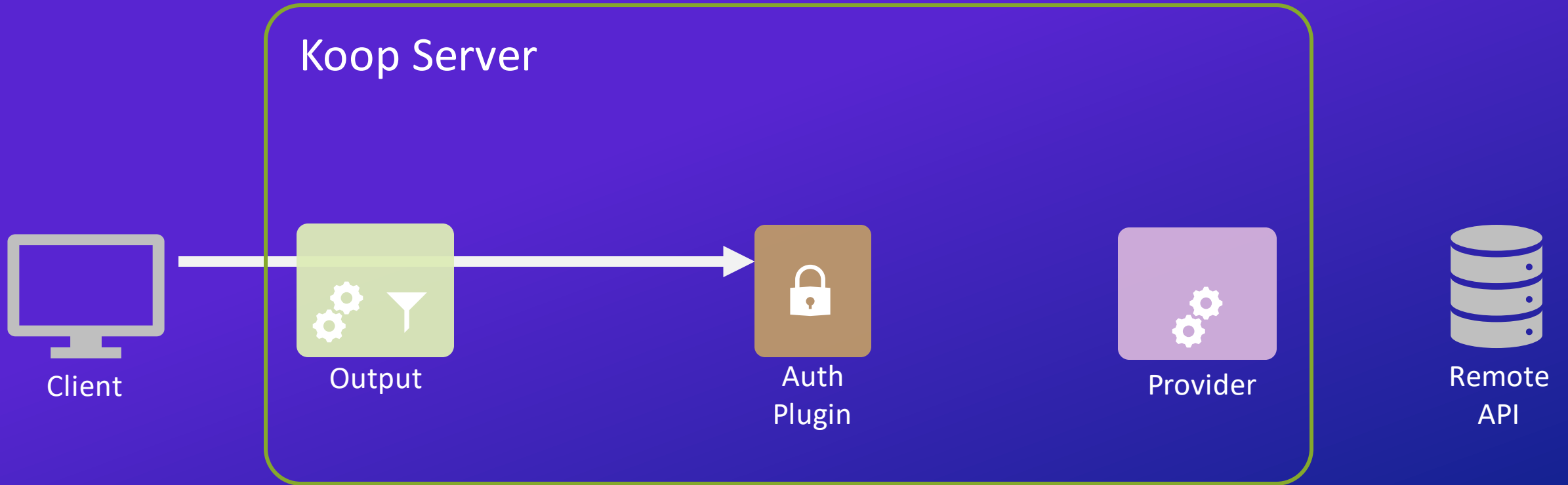




# Koop request lifecycle



# Koop request lifecycle



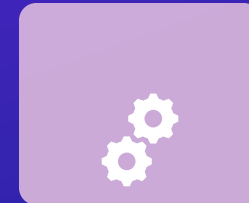
# Inside Koop

## Koop Server



Output  
GeoService

(Data transformation)



Provider  
CSV

An output plugin provides one or many public APIs for the client

A provider plugin knows how to read data from the data source

Example route

<http://localhost:3000/provider-csv/:id/FeatureServer/0/query>

provider fragment

output fragment

# One output, multiple sources

## Koop Server

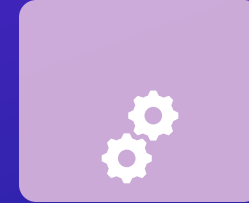


Output  
GeoService

(Data transformation)



Provider  
S3



Provider  
CSV

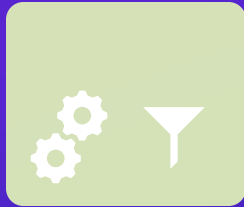
Example route

<http://localhost:3000/provider-s3/{id}/FeatureServer/0/query>

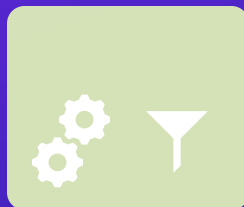
<http://localhost:3000/provider-csv/{id}/FeatureServer/0/query>

# One source, multiple outputs

## Koop Server

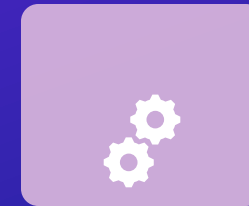


Output  
GeoService



Output  
VectorTile

(Data transformation)



Provider  
CSV

Example route

<http://localhost:3000/provider-csv/{id}/FeatureServer/0/query>

<http://localhost:3000/provider-csv/{id}/VectorTileServer/{z}/{x}/{y}.pbf>

## Actual code

```
const Koop = require('koop')
const csvProvider = require('koop-provider-csv')
const vtOutput = require('@koopjs/output-vector-tiles')

// initiate a koop app
const koop = new Koop()

// register a vector-tile output plugin
koop.register(vtOutput)

// register a CSV provider plugin
koop.register(csvProvider)

// start the server
koop.server.listen(3000, () =>
  koop.log.info(`Koop server listening at 3000`)
)
```

# Actual routes

```
"Geoservices" output routes for the "provider-csv" provider  Methods
-----
/provider-csv/rest/info                                     GET, POST
/provider-csv/tokens/:method                              GET, POST
/provider-csv/tokens/                                     GET, POST
/provider-csv/rest/services/:id/FeatureServer/:layer/:method  GET, POST
/provider-csv/rest/services/:id/FeatureServer/layers        GET, POST
/provider-csv/rest/services/:id/FeatureServer/:layer        GET, POST
/provider-csv/rest/services/:id/FeatureServer              GET, POST
/provider-csv/:id/FeatureServer/:layer/:method             GET, POST
/provider-csv/:id/FeatureServer/layers                    GET, POST
/provider-csv/:id/FeatureServer/:layer                    GET, POST
/provider-csv/:id/FeatureServer                           GET, POST
/provider-csv/rest/services/:id/FeatureServer*            GET, POST
/provider-csv/:id/FeatureServer*                          GET, POST
/provider-csv/rest/services/:id/MapServer*               GET, POST
/provider-csv/:id/MapServer*                              GET, POST

"VectorTileServer" output routes for the "provider-csv" provider  Methods
-----
/provider-csv/:id/VectorTileServer/:z([0-9]+)/:x([0-9]+)/:y([0-9+)].pbf  GET
/provider-csv/:id/VectorTileServer/tiles.json                       GET
/provider-csv/rest/services/:id/VectorTileServer/:z([0-9]+)/:x([0-9]+)/:y([0-9+)].pbf  GET
/provider-csv/rest/services/:id/VectorTileServer                   GET, POST
/provider-csv/rest/services/:id/VectorTileServer/                   GET, POST
/provider-csv/rest/services/:id/VectorTileServer/resources/styles/root.json  GET
```

# Providers

- CSV, GeoJSON, S3, Google Sheets, Socrata, Yelp, etc.
- Search npm
- Find in the provider list
  - <https://koopjs.github.io/docs/available-plugins/providers>
- Create your own

The screenshot shows the npm search results for 'koop provider'. The search bar contains 'koop provider' and the results show 55 packages found. The top results are:

- koop-provider-csv**: Koop provider for CSV. Published by haoliang 3.0.1, 9 months ago.
- @koopjs/provider-github**: Github provider for Koop. Published by rgwozdz 3.0.0, a year ago.
- koop-provider-carto**: Provider to transform layers hosted on Carto servers to ArcGIS Feature Layer.

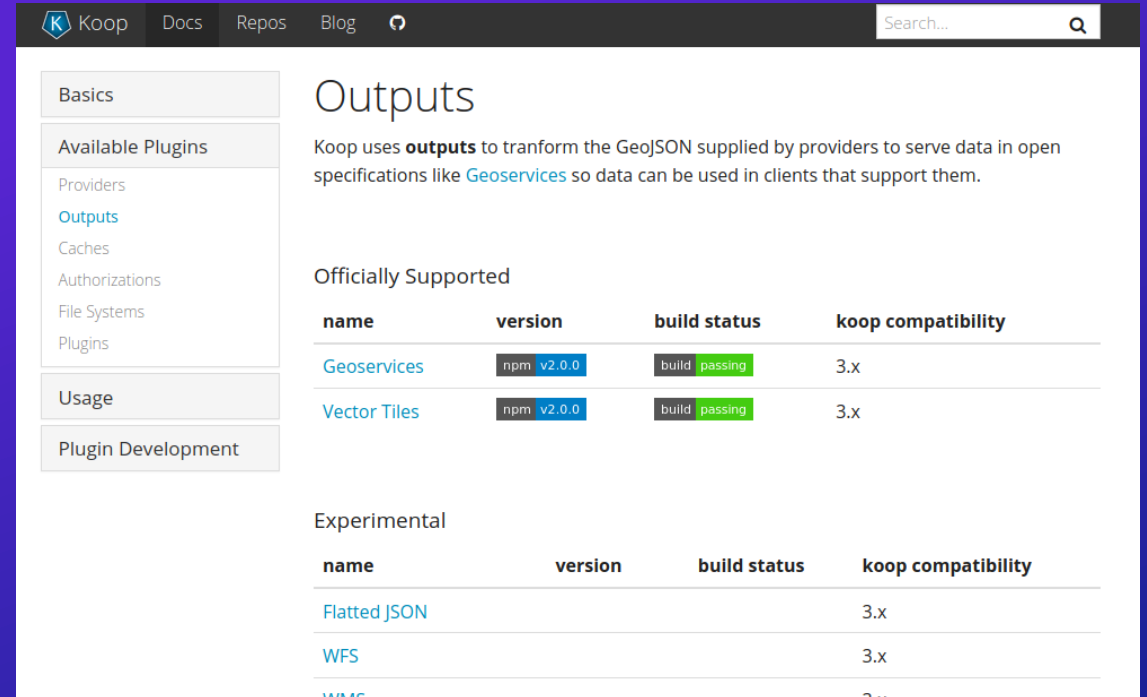
The screenshot shows the Koop Providers documentation page. The page title is 'Providers' and it explains that Koop uses providers to transform data from different sources to GeoJSON. Below the introduction is a table of 'Officially Supported' providers.

name	version	build status	compatibility
ArcGIS Online	npm v3.14.4	build passing	3.0.0-alpha.29
AWS S3 Select	npm v1.1.1	build passing	3.x
File GeotJSON	npm v1.0.1	build passing	3.x
Gist	npm v2.0.0	build passing	2.x
GitHub	npm v2.0.0	build passing	3.x
Google Analytics	npm v1.2.2	build passing	3.x
Google Fusion Tables	npm v1.0.0	NA	3.x
Google Sheets	npm v1.2.0	build passing	3.x



# Outputs

- Geoservices
  - Based on an open-source implementation
  - Support query, pagination, layer/service metadata
  - Default output in koop-core
- Others: VectorTile
- Find in the output list
  - <https://koopjs.github.io/docs/available-plugins/outputs>
- Create your own



The screenshot shows the 'Outputs' page on the Koop.js documentation website. The page is divided into sections for 'Officially Supported' and 'Experimental' outputs. The 'Officially Supported' section contains a table with columns for name, version, build status, and koop compatibility. The 'Experimental' section also contains a table with the same columns. The navigation menu on the left includes 'Basics', 'Available Plugins', 'Usage', and 'Plugin Development'. The 'Available Plugins' section is expanded to show 'Outputs' as the selected option.

Koop uses **outputs** to transform the GeoJSON supplied by providers to serve data in open specifications like [Geoservices](#) so data can be used in clients that support them.

Officially Supported

name	version	build status	koop compatibility
<a href="#">Geoservices</a>	npm v2.0.0	build passing	3.x
<a href="#">Vector Tiles</a>	npm v2.0.0	build passing	3.x

Experimental

name	version	build status	koop compatibility
<a href="#">Flatted JSON</a>			3.x
<a href="#">WFS</a>			3.x
<a href="#">WMS</a>			3.x

# Deployment

- Since Koop is a simple Node.js server in essence, it can be deployed to anywhere.



Google Cloud

kubernetes



and more...

# Getting started with using the Koop CLI

- Koop CLI
  - A command line tool for Koop developers
    - allows quick setup Koop project (app/plugin)
    - runs dev server for testing
    - works on Windows/MacOS/Linux



# Getting started with using the Koop CLI

- Check if the plugin exists
- Read Koop specification on the home page
- Use Koop CLI
  - A command line tool for developers
    - setup Koop project (app/plugin)
    - run dev server for testing
    - works on Windows/MacOS/Linux

# Building a Koop app with CLI

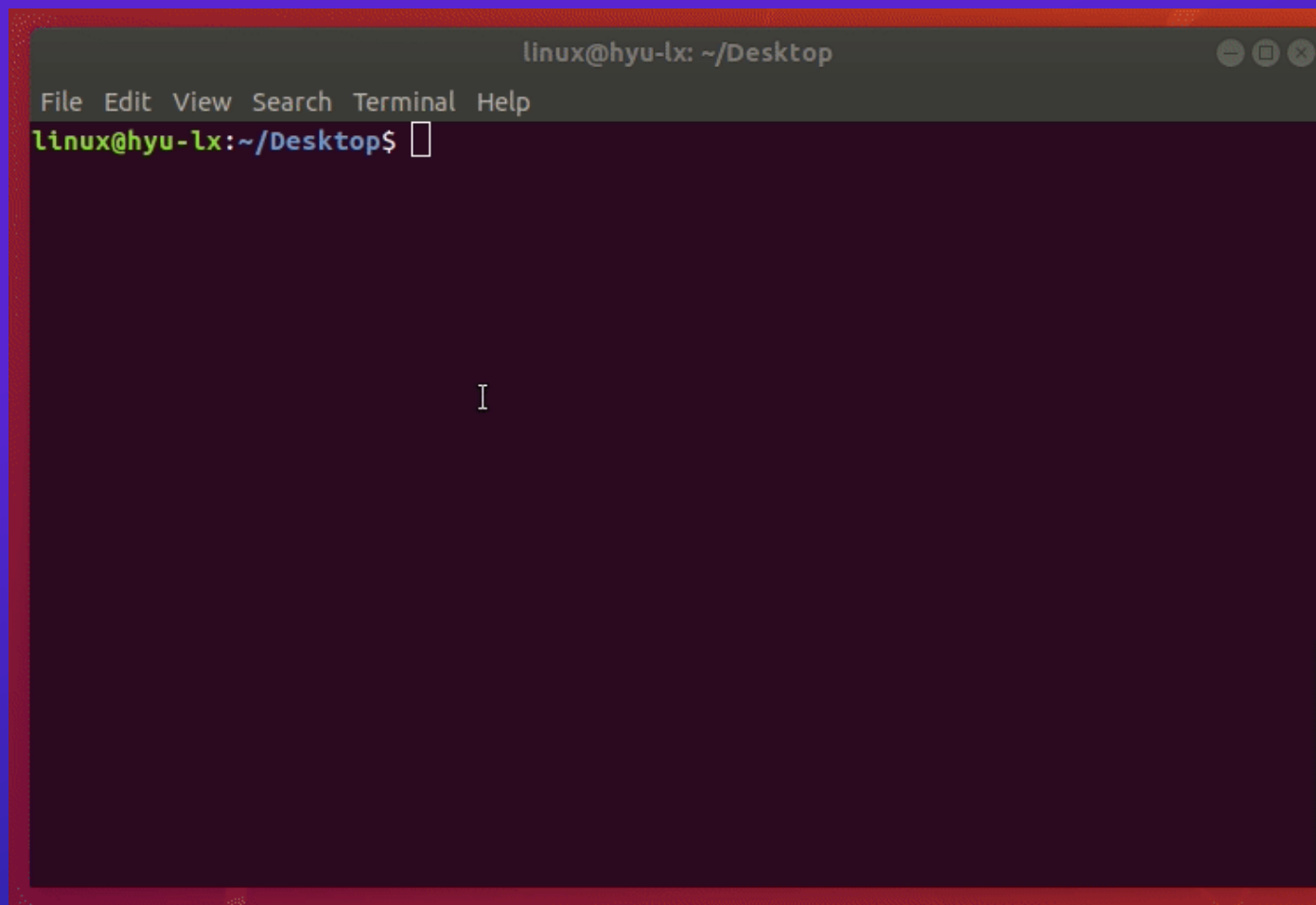
```
# create a new Koop demo app with boilerplate
koop new app demo-app

# go into the demo app directory
cd demo-app

# install a provider
koop add provider @koopjs/provider-socrata

# test it out
koop serve
```

# A live demo



# Building a Koop provider

- Use the Koop CLI to create the project from template
- Modify the provider "getData" method
  - Add code to fetch data from the data source
  - Convert the raw data into GeoJSON

```
8
9  function Model (koop) {}
10
11  Model.prototype.getData = function (req, callback) {
12
13    // Add code to fetch data from the remote API
14
15    // Add code to transform fetched data to GeoJSON
16
17    // Execute the callback
18    callback(null, geojson)
19  }
20
21  module.exports = Model
22
```

# Summary

- **Koop is an open-source and lightweight ETL solution**
- **Geoservice output supports many feature of ArcGIS clients**
- **There are many Koop provider and output plugins published and ready for use**
- **Koop can be deployed anywhere**
- **Some development may be necessary to set up a custom Koop server**
  - **But Koop's plugin architecture and CLI can reduce much effort**





# Resources

- Koop home page: <https://koopjs.github.io>
- Koop CLI: <https://github.com/koopjs/koop-cli>
- Examples
  - App: <https://github.com/koopjs/koop-app-example>
  - Provider: <https://github.com/koopjs/koop-provider-example>
  - Docker: <https://github.com/koopjs/koop-docker-example>
  - AWS Lambda: <https://github.com/koopjs/koop-serverless-example>