Introduction to APIs

ACM Dev

acmcsuf.com/api-meeting-deck

What is an API?

- API stands for application programming interface
- APIs allow us to access data or features of another program, application, or service
- In the context of the web, it is a way for standalone applications to communicate with each other over the network

REST: Representational State Transfer

A flexible API architecture that can be implemented in almost any language, transfer several types of data, and communicate over HTTP.

REST APIs adhere to the six REST design principles

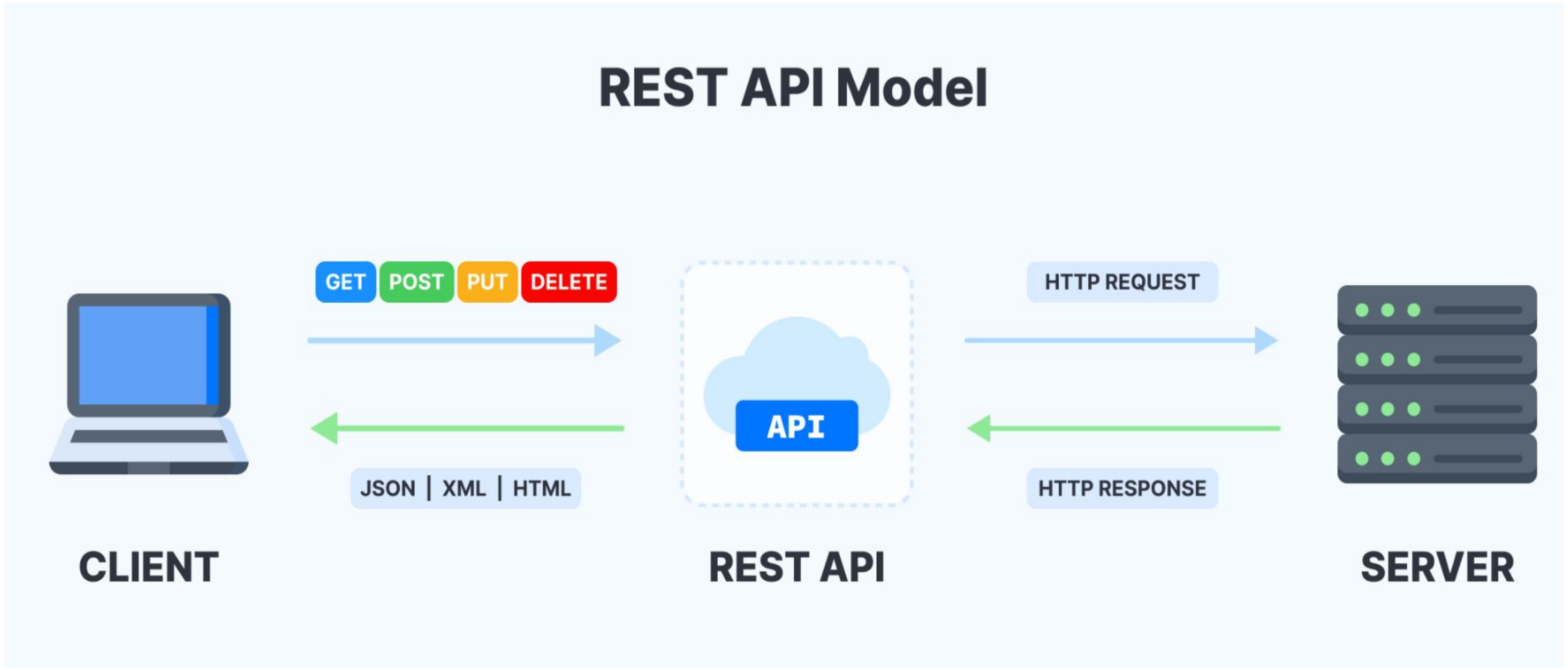
- **Uniformity**: every request to a specific resource looks the same
- **Decoupling**: client and server should be separate
- **Statelessness**: includes all information needed to process request
- **Cacheability**: Should be cacheable on both sides
- **Layered system**: endpoints don’t necessarily communicate directly
- **Code on demand**: only send code/scripts as needed

CRUD: Create, Read, Update, Delete

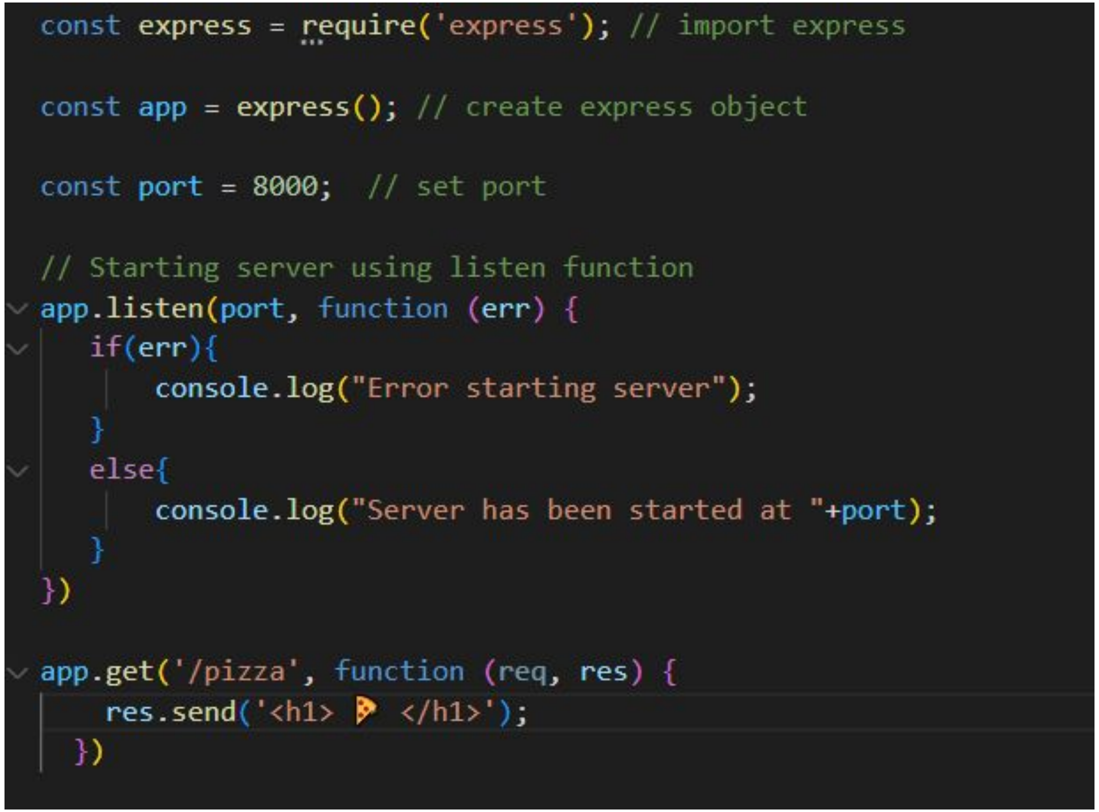
Because REST APIs are built on top of HTTP, we can use HTTP methods to interact with resources on our server.

- **GET** - access a resource
- **POST** - create a resource
- **PUT** - update (the entire) resource
- **DELETE** - …wonder what this one does?
- There are also a few additional HTTP methods that we haven't covered, such as PATCH, HEAD, TRACE, CONNECT, and OPTIONS. To learn more visit this link!

Rest API diagram

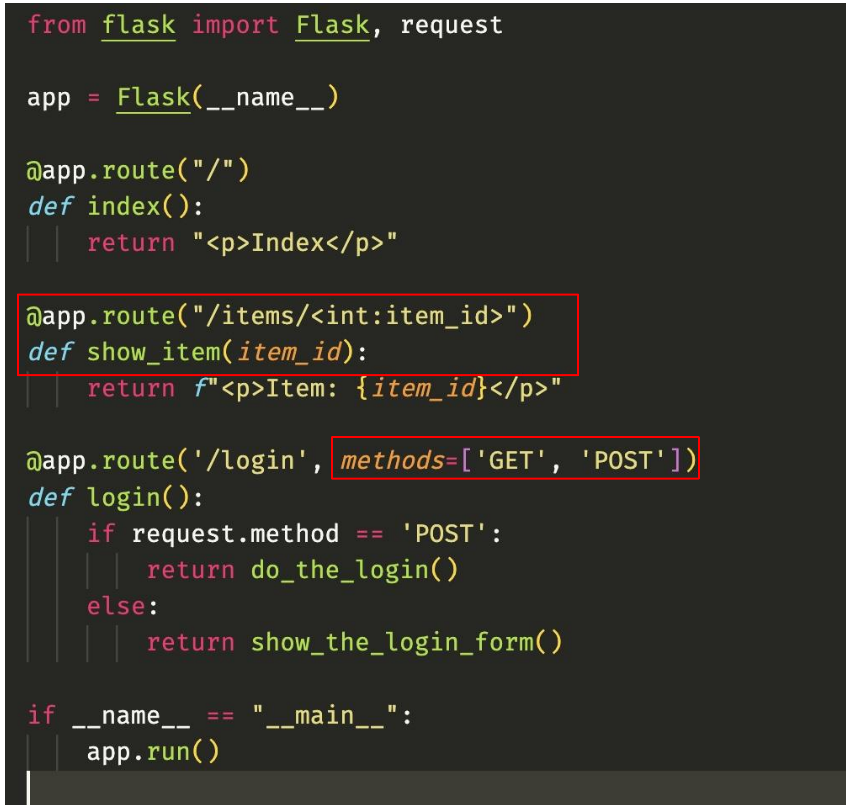


Example Node API



```
const express = require('express'); // import express

const app = express(); // create express object

const port = 8000;  // set port

// Starting server using listen function
app.listen(port, function (err) {
    if(err){
        console.log("Error starting server");
    }
    else{
        console.log("Server has been started at "+port);
    }
})

app.get('/pizza', function (req, res) {
    res.send('<h1> 🍕 </h1>');
  })
```

Example Flask API



```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/")
def index():
    return "<p>Index</p>"

@app.route("/items/<int:item_id>")
def show_item(item_id):
    return f"<p>Item: {item_id}</p>"

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()

if __name__ == "__main__":
    app.run()
```

Deploying on Ngrok

- Ngrok runs a small client process on your machine
- This creates a private connection tunnel to the cloud service.
- Your localhost development server is mapped to an ngrok.io sub-domain, which a remote user can then access
- server-->link

Using Ngrok

Overview of Steps:

- Download -> https://dashboard.ngrok.com/get-started/setup
- Configure Ngrok auth token (you shouldn't have to?)
- Use `npm i -g ngrok` or `brew install ngrok` or `choco install ngrok`
- Run `ngrok http localhost:yourPort`
- You should get a link that will allow other people to call your API!

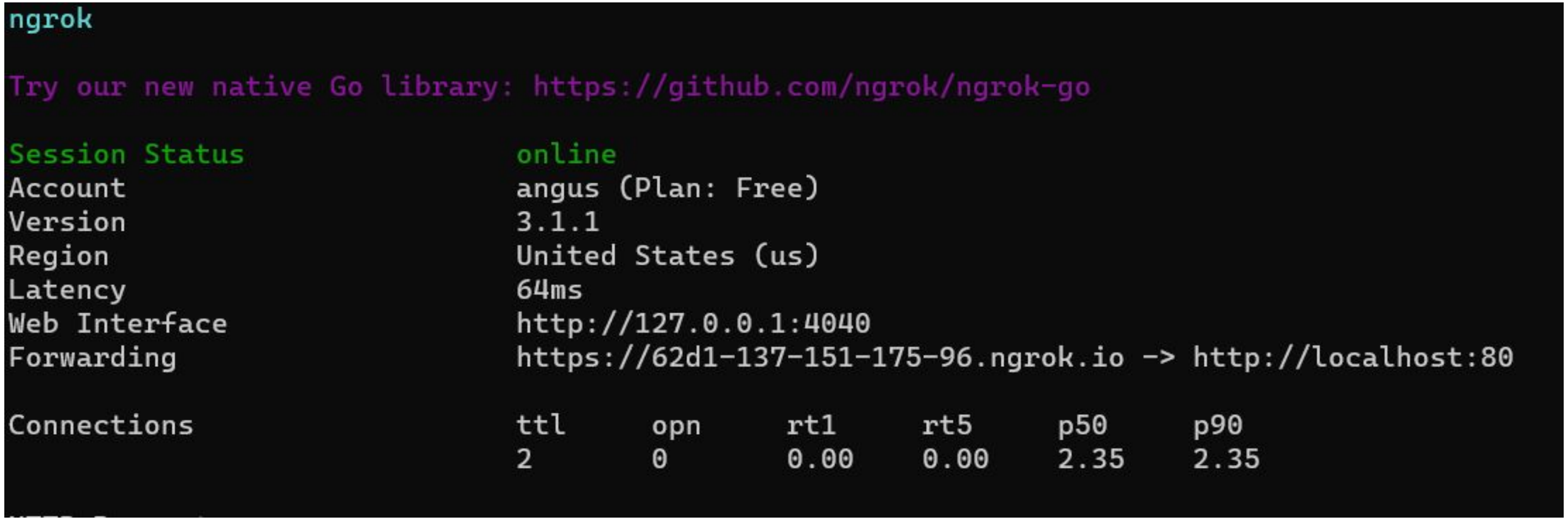
Using srv.us

srv.us is similar to ngrok but uses the built-in ssh tool.

Overview of Steps:

- Run `ssh -R 1:localhost:${port} srv.us`

Live demonstration



```
ngrok

Try our new native Go library: https://github.com/ngrok/ngrok-go

Session Status                online
Account                       angus (Plan: Free)
Version                       3.1.1
Region                        United States (us)
Latency                       64ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://62d1-137-151-175-96.ngrok.io -> http://localhost:80

Connections                   ttl     opn     rt1     rt5     p50     p90
                              2       0       0.00    0.00    2.35    2.35
```