# CSCI 466: Networks

Cloud Computing, Building a basic API

Reese Pearsall

Fall 2024

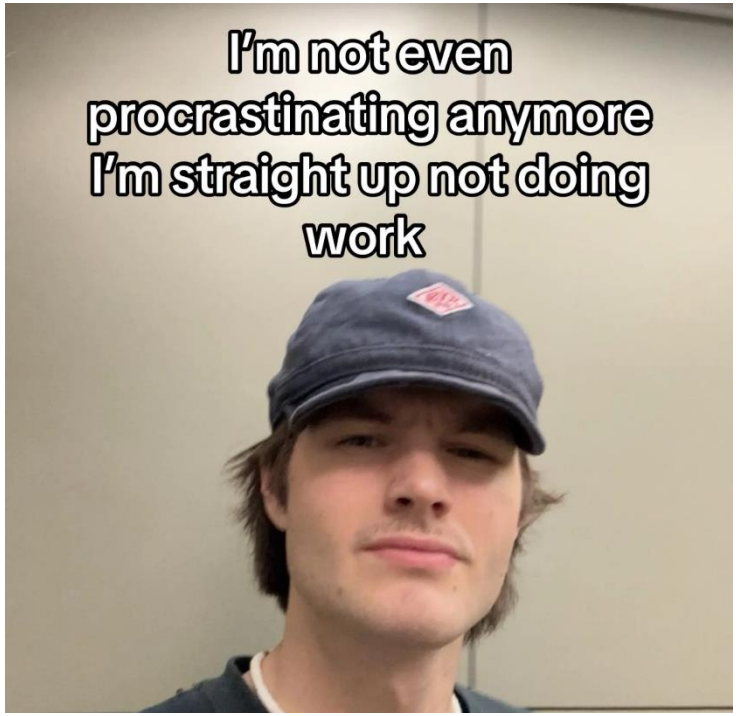
# Announcements

Wireshark Lab 4 due **Wednesday** at 11:59 PM

PA4 due Sunday 11/24 @ 11:59 PM

Wednesday will also be lecture recording (no in-person lecture)

Friday will be a work day for PA4 (no lecture)

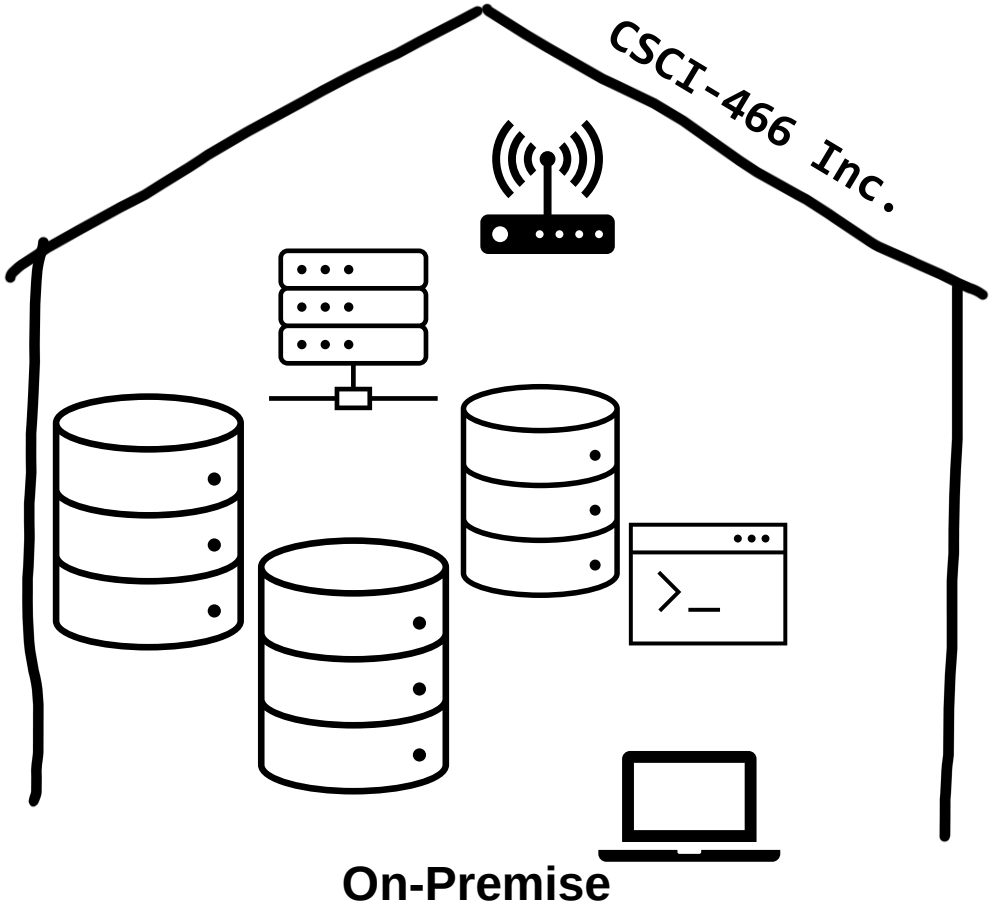



| | |
|---|---|
| Monday December 2nd | PA5 work day (No lecture) |
| Wednesday December 4th | Wrap Up + Review |
| Friday December 6th | Final Quiz (**in person**) |

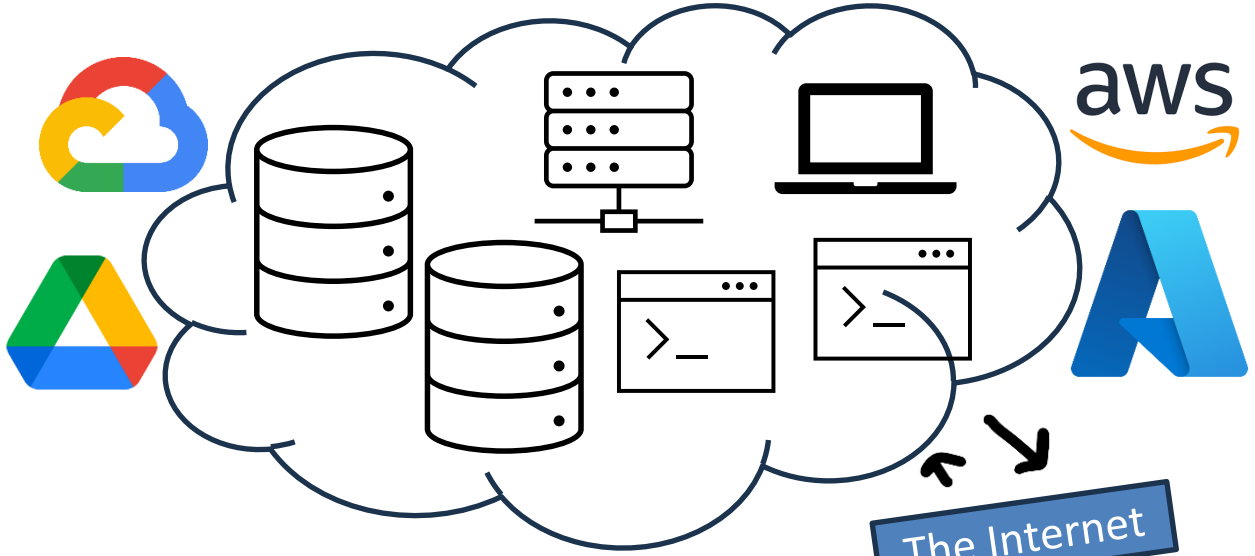
**Final week of CSCI 466**

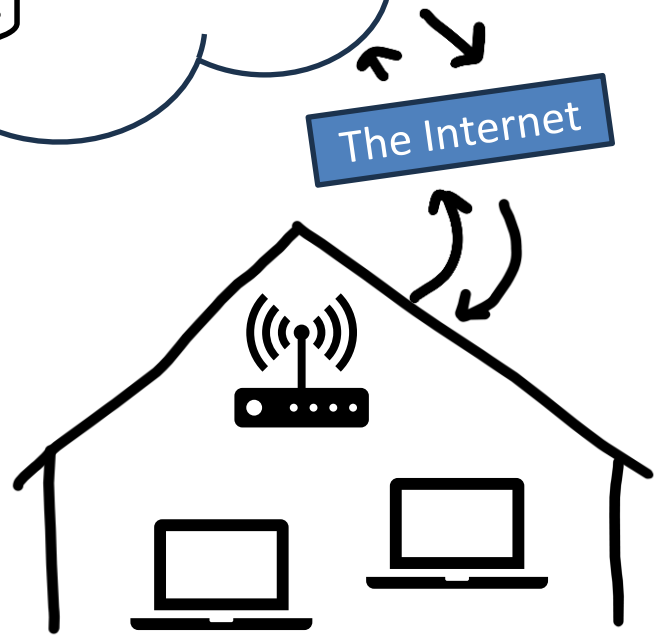
# Wireshark Lab 4

**Cloud computing** is the on-demand service of computing resources over the internet instead of **on-premise**




On-Premise




Cloud Computing

# Benefits of Cloud Computing

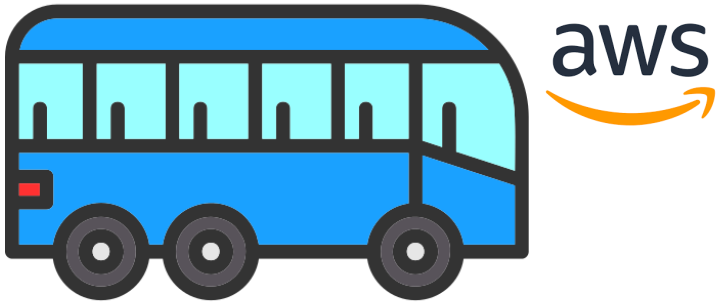
- **Cost Efficiency** – Hosting your services on a cloud system will generally be less expensive than purchasing and maintaining your own servers
  → *Pay-as-you-go* model
- **Scalability**- Resources can quickly be scaled up or down based on demand
- **Accessibility** – Cloud providers offer high uptime guarantees across multiple data centers
- **Backups** – Cloud services provide regular backups in case of hardware failures or accidents
- **Security + Compliance** – Trusted cloud providers generally provide solid security + guarantees on *compliance*

# Cons of Cloud Computing

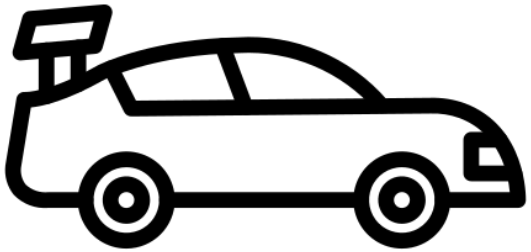
- **Control**– less control over infrastructure and configuration
- **Security-** Cloud systems are a common target in cyber attacks
- **Unexpected Costs**

# Deployment Types

**Public Cloud-** shared environment where resources are owned and operated by third-party cloud provider and accessible through the public internet



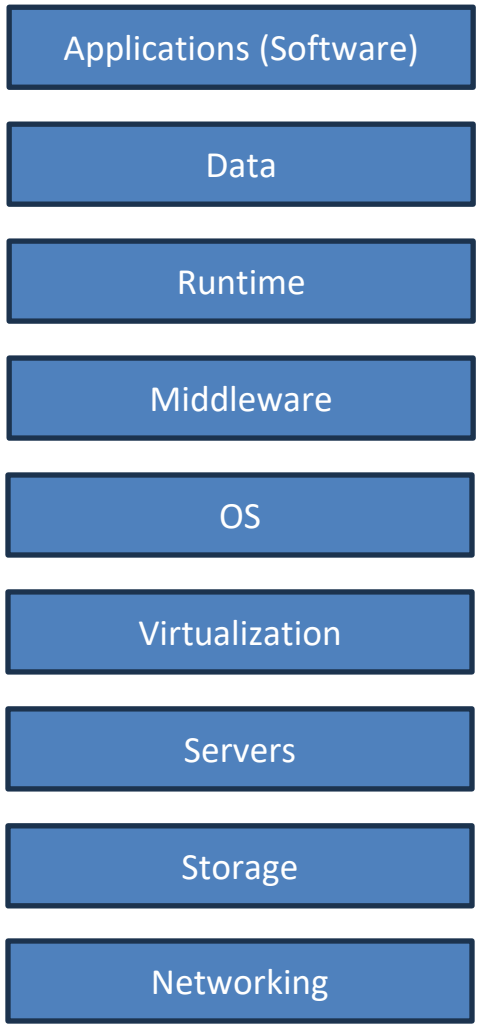

**Private Cloud-** computing resources are exclusively dedicated to one organization. Not accessible through the public internet



**Hybrid Cloud-** Combination of public and private. Information and Data flows between public and private cloud servers

# Cloud Computing Services

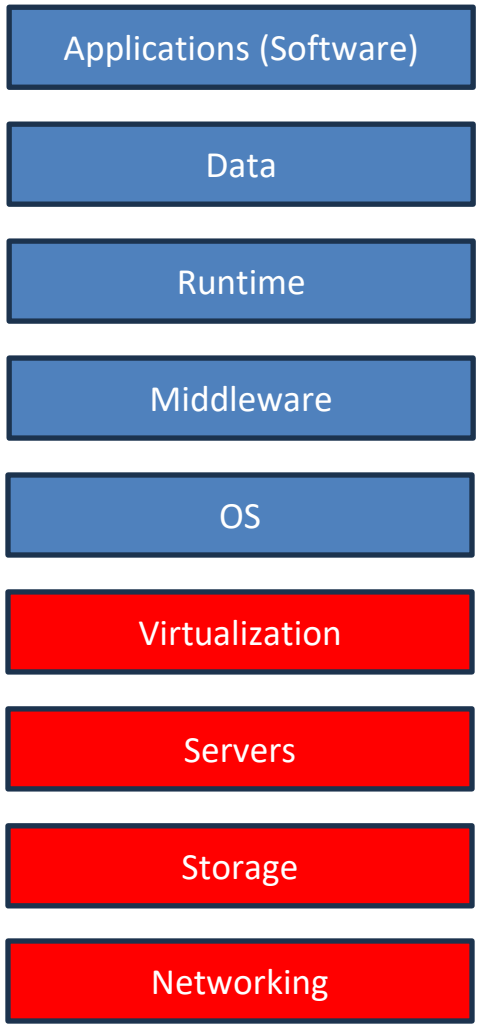
These are all the computing resources in a normal tech stack




*In an on-premise model, the organization is responsible for managing all of these*

# Cloud Computing Services

These are all the computing resources in a normal tech stack

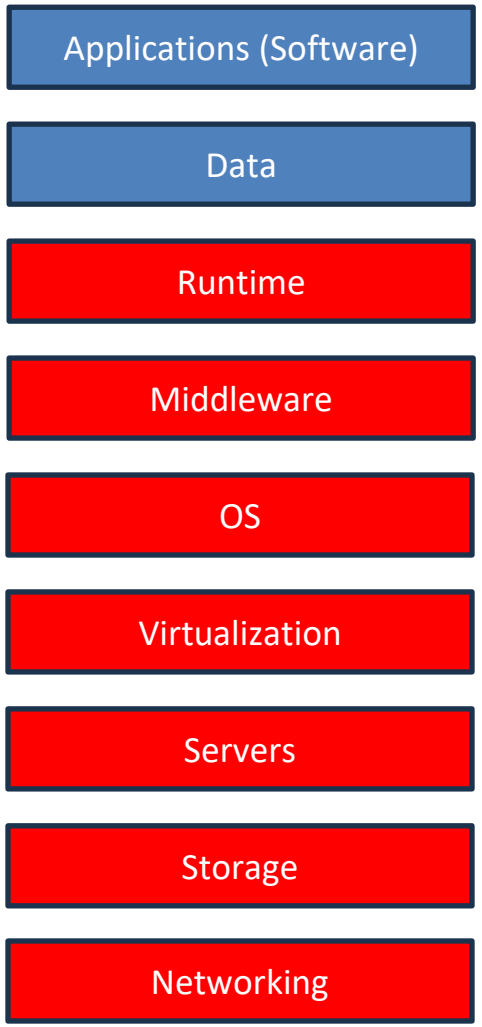



**Infrastructure as a Service (IaaS)** provides virtualized computing resources and foundational IT building blocks over the internet

aws will manage these

# Cloud Computing Services

These are all the computing resources in a normal tech stack




**Platform as a Service (PaaS)** provides hardware and software tools needed for application development

Write code without having to worry about underlying infrastructure

aws will manage these

# Cloud Computing Services

These are all the computing resources in a normal tech stack

- Applications (Software)
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Servers
- Storage
- Networking

**Software as a Service (SaaS)** delivers fully-functional applications over the internet

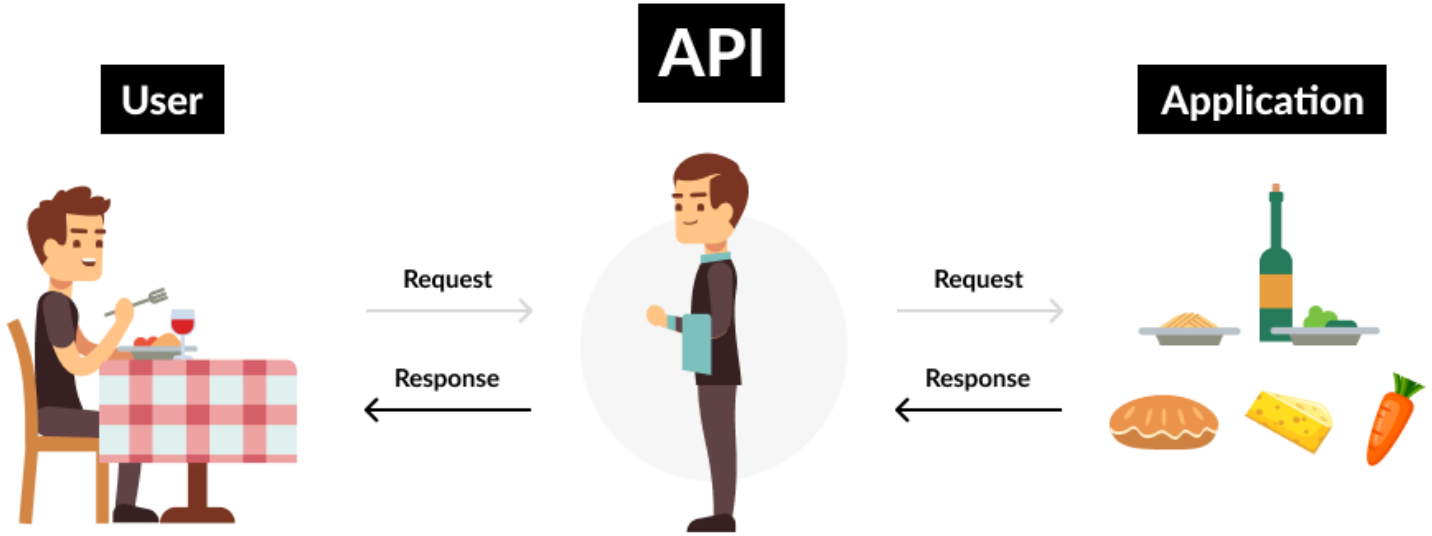
Most commonly used for web applications (Google Drive)


will manage these

# API

An **Application Programming Interface (API)** is a very common application to deploy on the cloud

An API accepts requests, and communicates with a backend service to retrieve/update/delete data

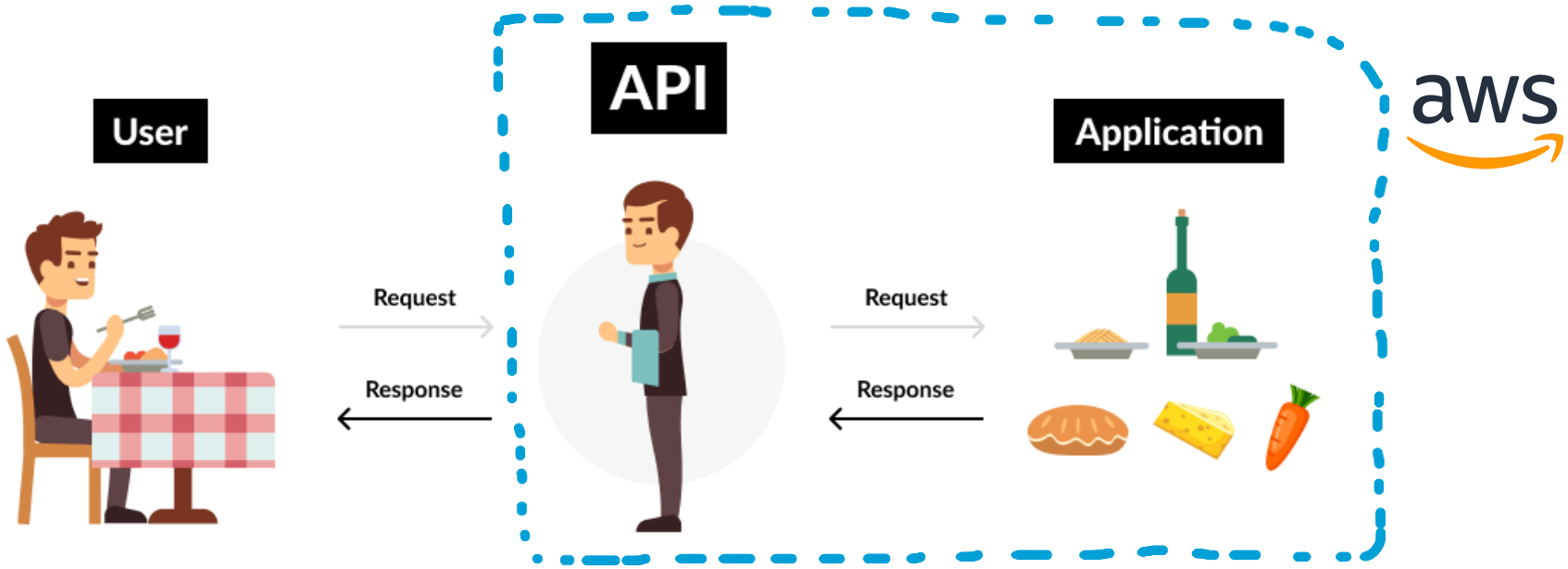



APIs typically listen for HTTP requests, perform some actions, and then send data back in an HTTP response

# API

An **Application Programming Interface (API)** is a very common application to deploy on the cloud

An API accepts requests, and communicates with a backend service to retrieve/update/delete data




APIs typically listen for HTTP requests, perform some actions, and then send data back in an HTTP response

If our API is on the cloud, it will be accessible 24/7 and it doesn't have to use *our* computational resources

# API

Today and Wednesday (and on Programming Assignment 5), our goal will be to build a basic API, and then deploy it to the cloud

We will first build it locally, and then deploy to Amazon Web Service (**AWS**)

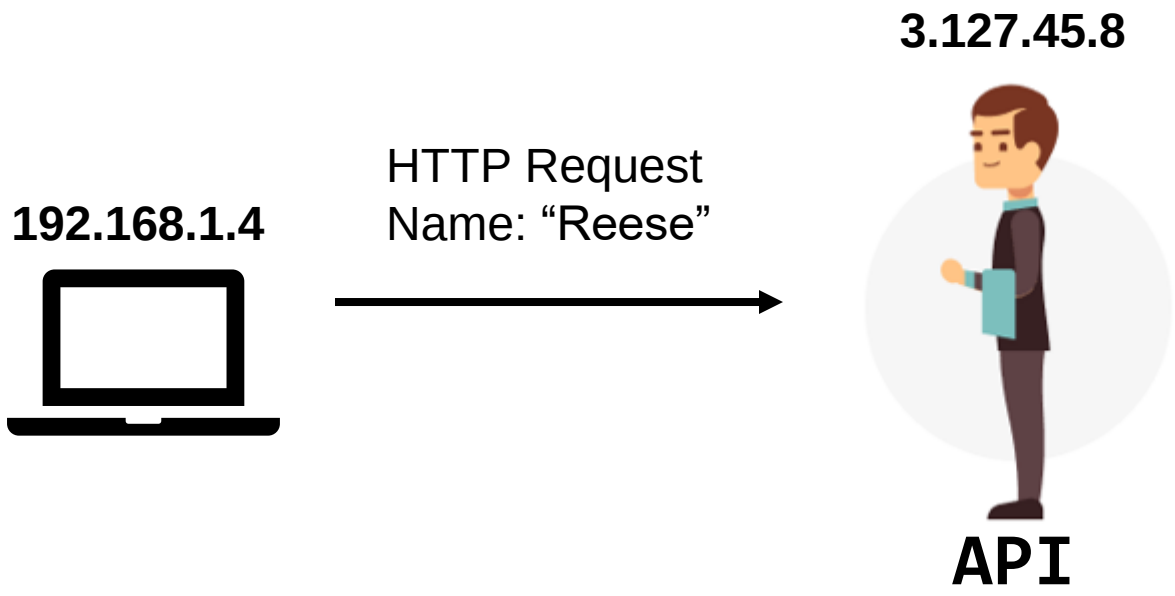
**3.127.45.8**

**API**

# API

Today and Wednesday (and on Programming Assignment 5), our goal will be to build a basic API, and then deploy it to the cloud

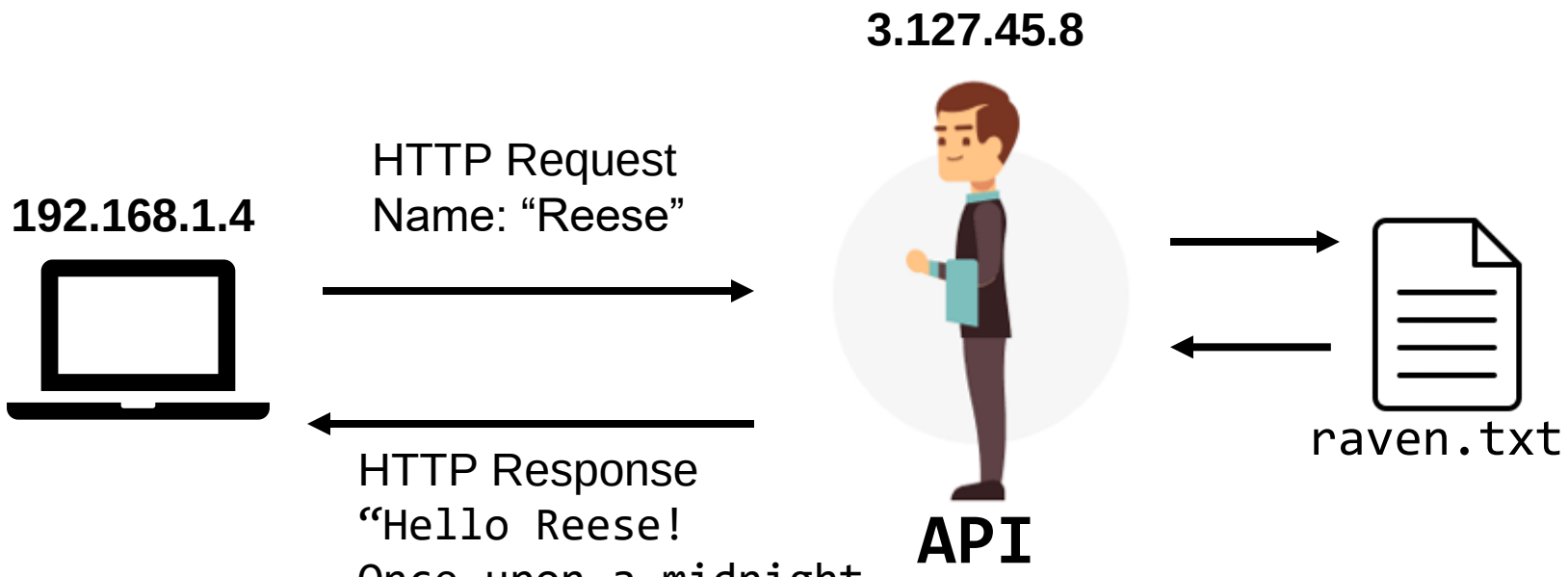
We will first build it locally, and then deploy to Amazon Web Service (**AWS**)



# API

Today and Wednesday (and on Programming Assignment 5), our goal will be to build a basic API, and then deploy it to the cloud

We will first build it locally, and then deploy to Amazon Web Service (**AWS**)



# Flask


**Flask** is a popular python framework for creating basic web applications and API

- Easy to learn (maybe too easy?)
- Powerful routing system for different endpoints
- Built-in handling and templating of HTTP requests/responses
- RESTful API support
- Can be combined with many other web frameworks

# Anatomy of Flask Program

```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello_world():
    return 'Hello World'

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

```
* Running on http://127.0.0.1:5000
```

If I visit this IP in my web browser, or send an HTTP request to it, I will get “Hello World” returned

```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello_world():
    return 'Hello World'

@app.route('/reese')
def reese_page():
    return 'Reese page'


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

```
* Running on http://127.0.0.1:5000
```

This app now has two endpoints you can send requests

`http://127.0.0.1:5000/reese` will return “Reese page”

# Anatomy of Flask Program

```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello_world():
    return 'Hello World'

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

Creates a new endpoint

value that gets returned when HTTP request is sent to that endpoint

```
* Running on http://127.0.0.1:5000
```

If I visit this IP in my web browser, or send an HTTP request to it, I will get “Hello World” returned

```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello_world():
    return 'Hello World'

@app.route('/reese')
def reese_page():
    return 'Reese page'


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

```
* Running on http://127.0.0.1:5000
```

This app now has two endpoints you can send requests

`http://127.0.0.1:5000/reese` will return “Reese page”

# Nevermore API

```python
from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def example():
    file = open("raven.txt","r",encoding="utf8")
    filetext = file.read()
    name = request.headers["name"]
    answer = "Hello, " + name + "\n \n" + filetext
    return answer

@app.route('/reese')
def reese():
    return "this is the reese page!"


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

Sending HTTP requests to API

```
C:\Users\reese>curl --header "name: reese" 127.0.0.1:5000
Hello, reese

Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
    While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
“’Tis some visitor,” I muttered, “tapping at my chamber door—
```