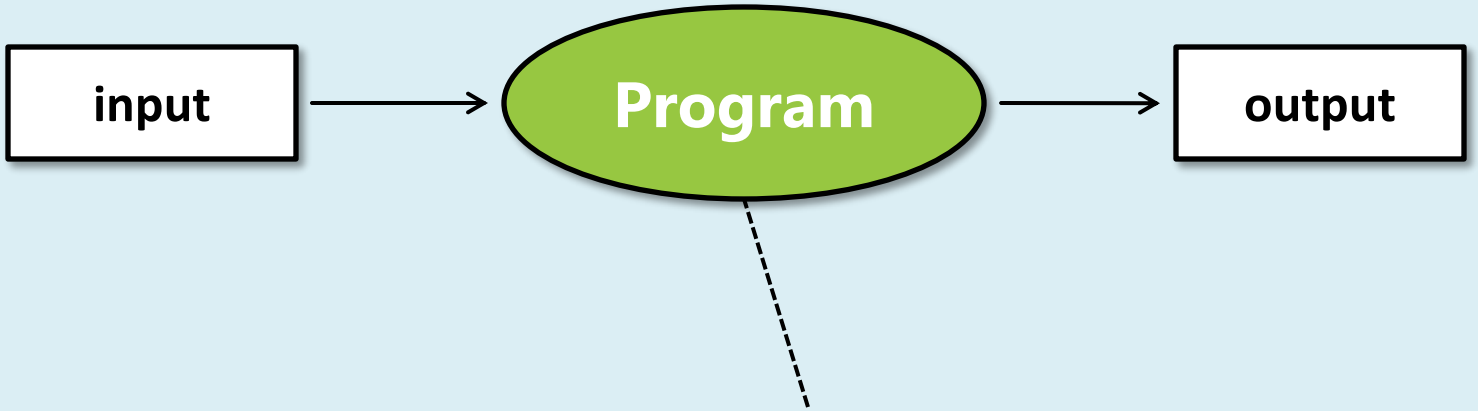
CS 310 : Lecture 01

- **Introduction to the Cloud**
  - The cloud and Amazon Web Services (AWS)
  - Modern software architectures
  - Web servers
  - SOA
  - AWS S3

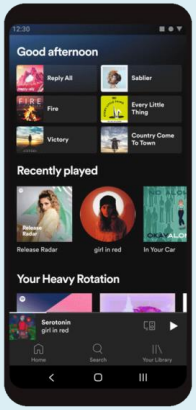
Stand-alone software

- **Most software you've written is "stand-alone"**
  - *CS 111 / 150 / 211 / 214 / 3xx*



Multi-tier

- Most software you use is *not* stand-alone
- Example: Pandora or Spotify

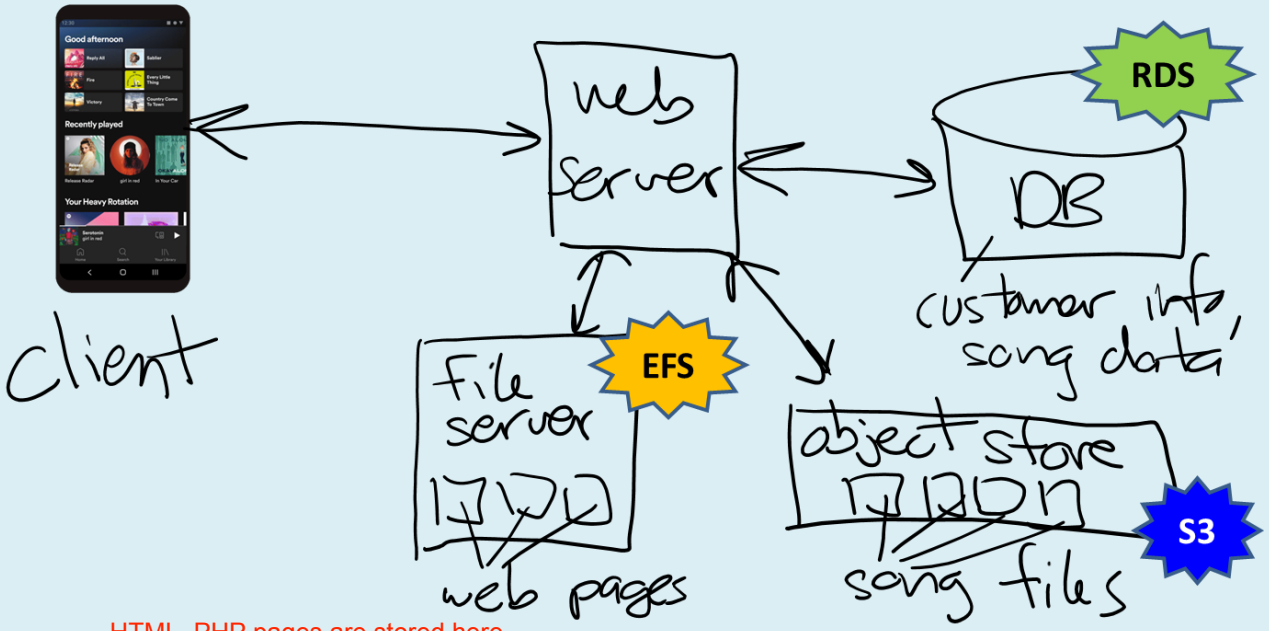


Trade-offs

- **AWS offers at least 3 services for storing data**
  - *Database (RDS), File system (EFS), and Object store (S3)*
- **Why?**

Discuss Cost vs Speed vs Reliability
to answer difference between different storage

Why songs are not in Database ?
Answer: Fast but expensive. Only Critical information that can not be recovered or is no big deal if lost is stored in database. Eg. Passwords, Credit Cards, Money Amount etc are stored in Database



HTML, PHP pages are stored here
EFS : Elastic FileSystem

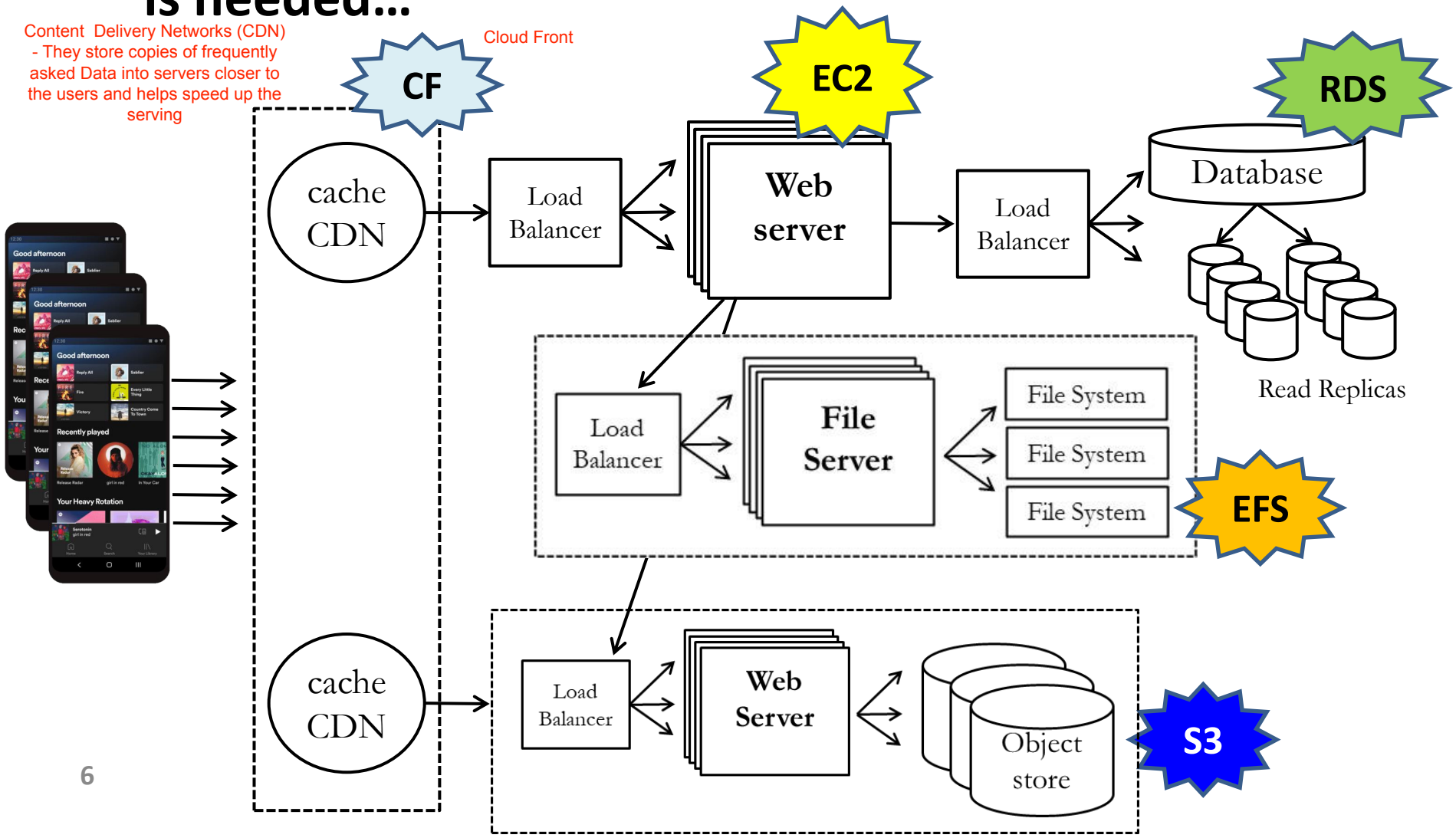
Faster than s3 and slower than DB. Cost more money than S3. Useful when we need to serve something fast.

Simple Storage Service
Cheaper, Slower and are useful to store non-critical huge stuff which otherwise will cost huge amount if stored in DB.

Scalable version

With replication, if we do changes at one place, it gets replicated everywhere.

- **To support millions of users, caching and replication is needed…**



Question

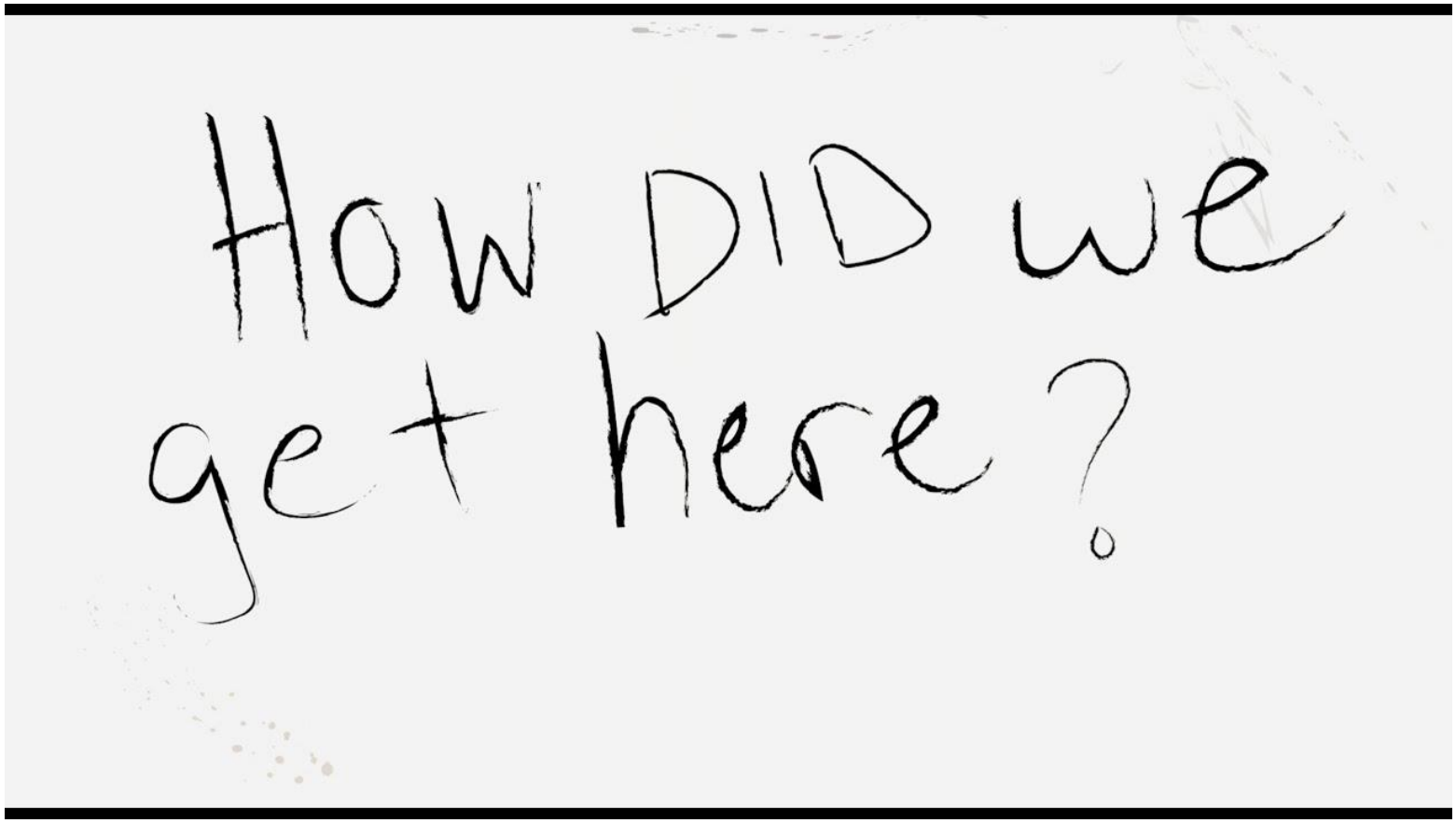
- **How many Amazon regions are there?**
- **How many CloudFront sites does Amazon run?**

– *Use google or favorite chatbot to answer…*

34 Regions: link

600+ CloudFront sites: link

Note: each region = 3+ data centers for redundancy

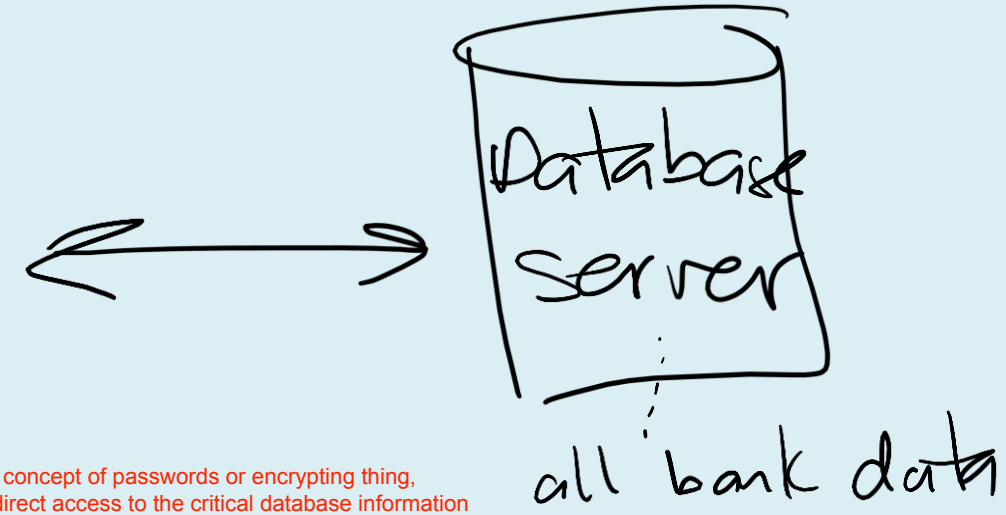

How DID we
get here?

Client-server

- **Initial designs were client-server**
  - *Two tiers*
  - *Nearly all the computation on the client --- the server typically stored data*
  - *Often called "Thick" client*

Example

- **Banking software (before you were born :-)**



Unsecure - No concept of passwords or encrypting thing,
a user /banker has direct access to the critical database information

The client software is a custom app that does everything --- classic "thick" client

The software is only available to bank employees because:

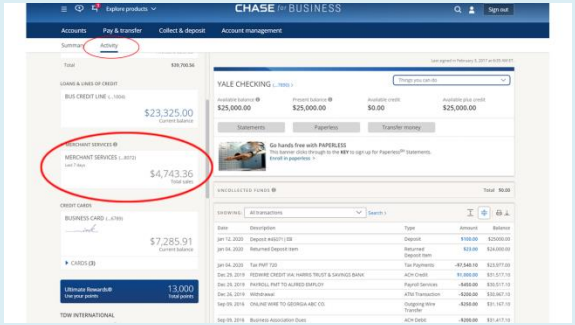
1. the database is not safe to put on the internet
2. the software is designed for bank employees, not customers

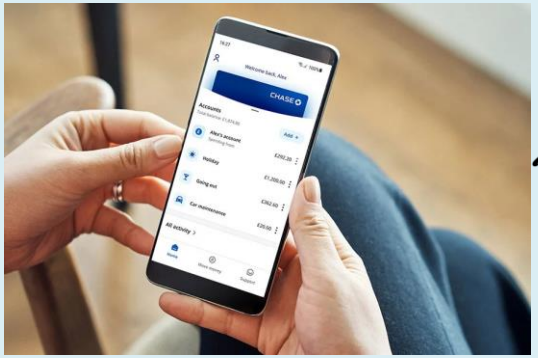
Multi-tier

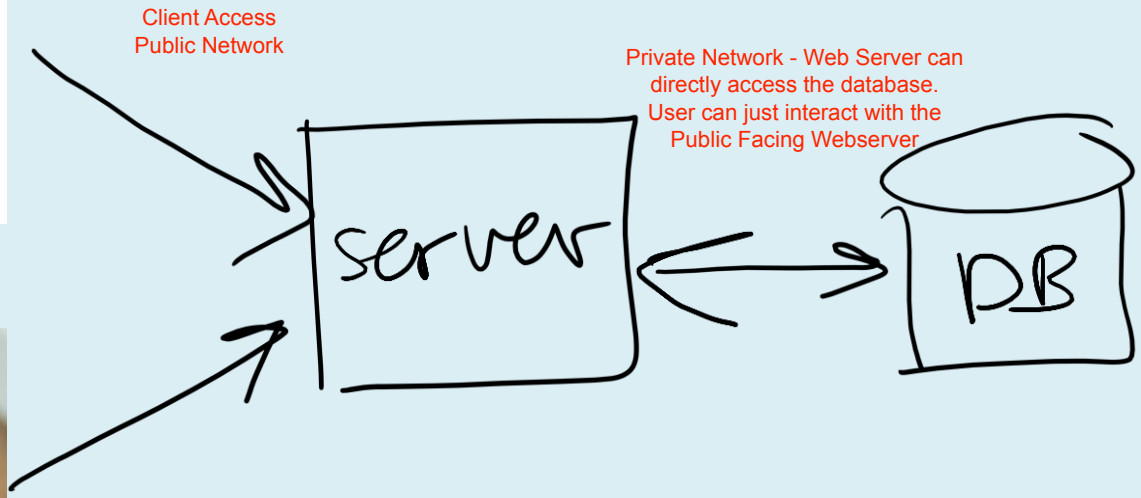
- **Designs evolved to multi-tier**
  - *3+ tiers*
  - *Enhances security by hiding data behind a server…*

Example

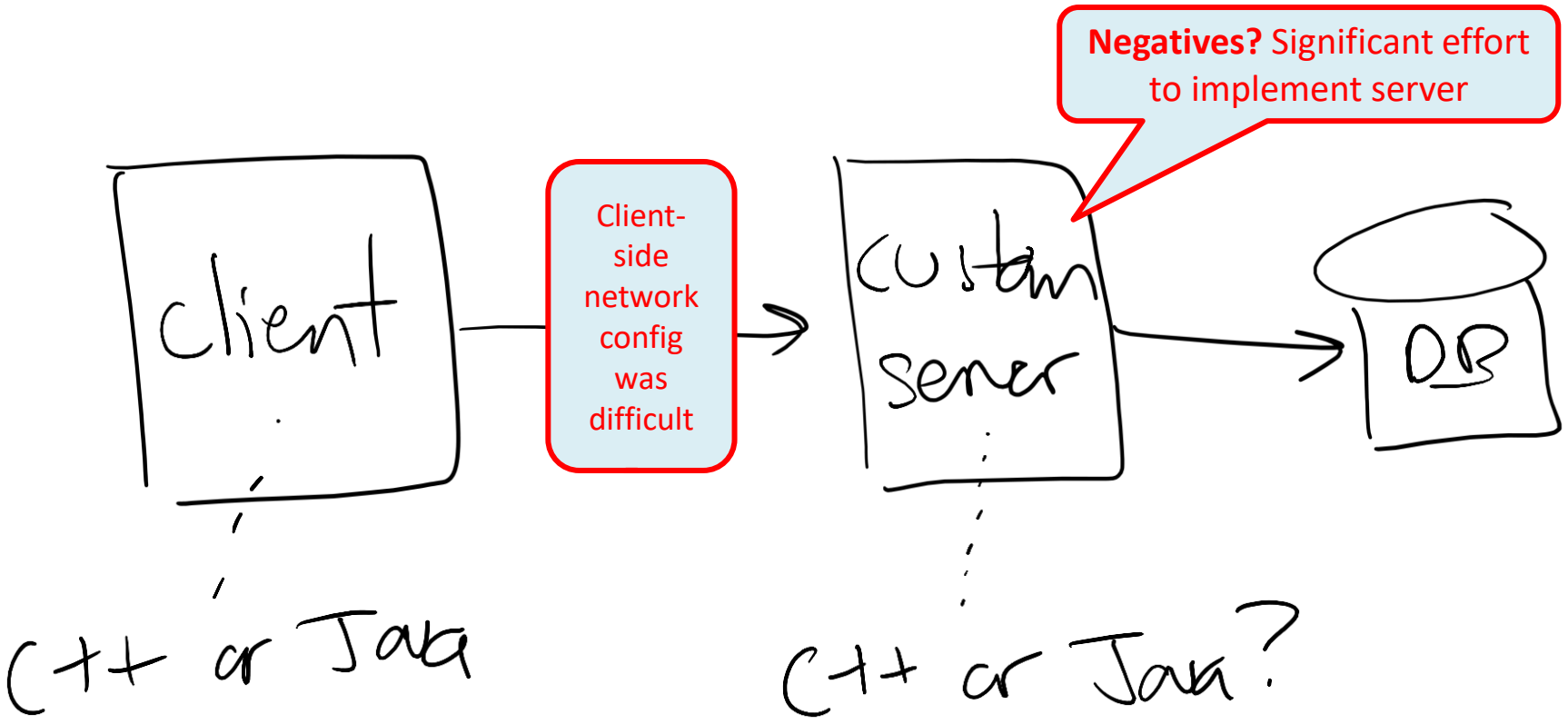
- **Modern banking software**
  - *Client software focuses on UI ("thin client")*
  - *Server software performs most of the processing*







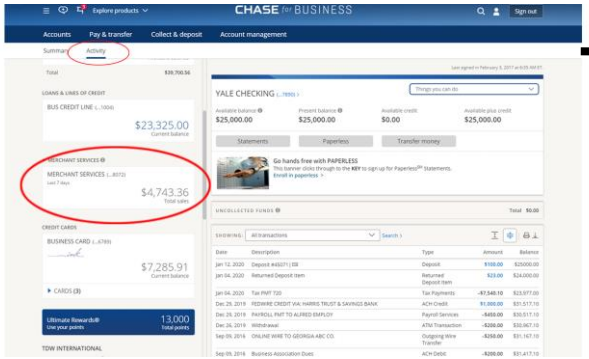
Early designs were custom built



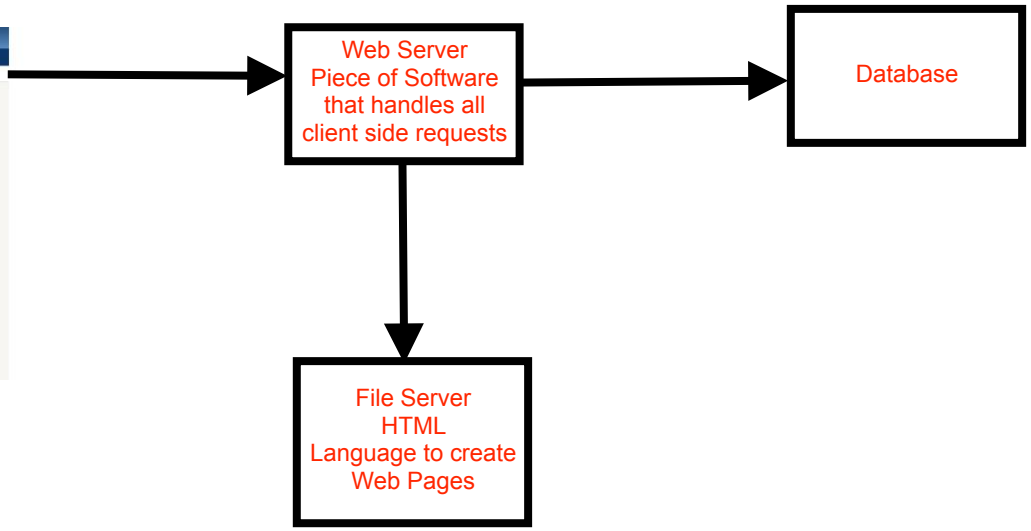
Earlier all servers were custom built, different types of authentication mechanisms, passwords or encryptions, so could not scale much since no set standards were there and therefore was very hard to write server side codes.

Multi-tier web apps

- **With popularity of web, multi-tier => web apps**
  - *The browser became the client…*



Browser -> Client



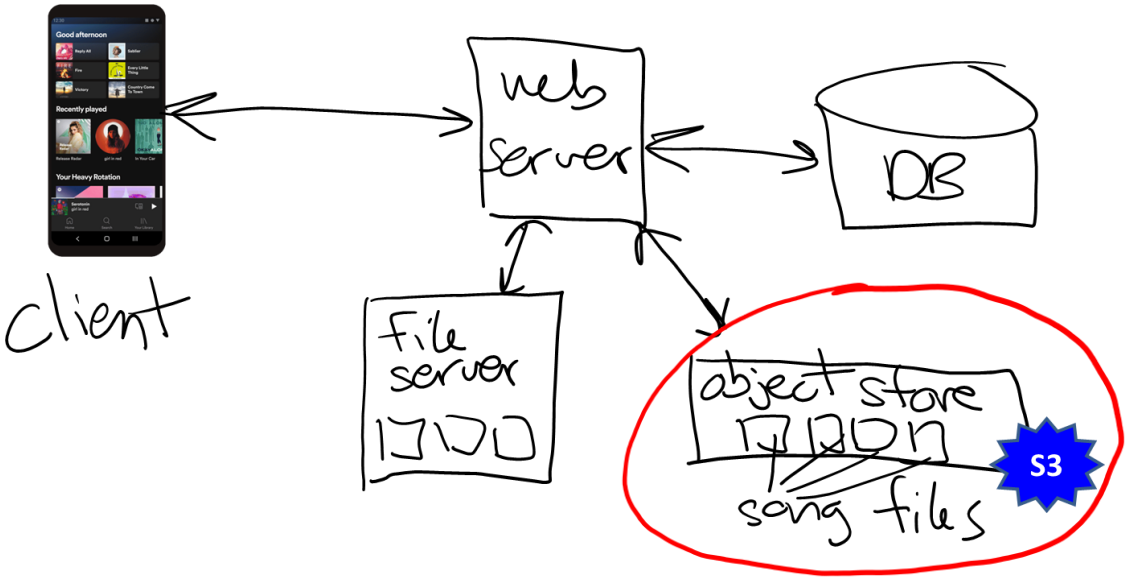

EXAMPLE

AWS S3 object store

- **S3 = Simple Storage Service**

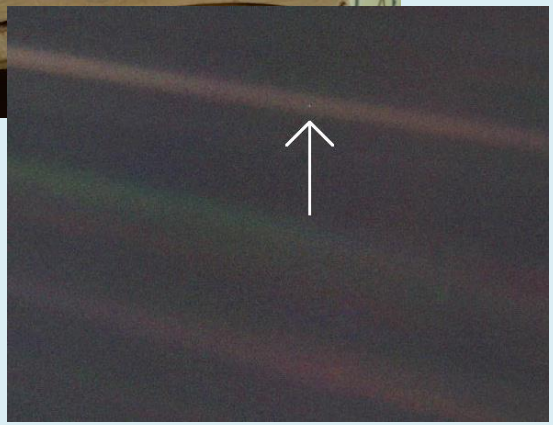
s3 is implemented as a web server. It is being used to serve data and not the HTML content

- **Use cases?**
  - *Least expensive cloud storage*
  - *Streaming video, music, etc.*
  - *Large datasets and documents (e.g. legal documents)*
  - *Unstructured data (e.g ML training sets)*



Demo

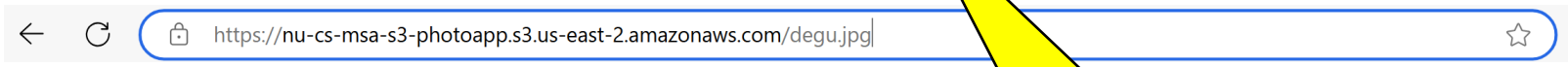
- I have some images up in S3…



exercise

- **Open a browser and type this URL**

https://nu-cs-msa-s3-photoapp.s3.us-east-2.amazonaws.com/degu.jpg



US-East => Ohio

- **You got an image back…**

- **What does this tell us about how S3 works?**

Browser is the client and s3 is a webserver that takes user request and returns back the asked image.

Exercise part 02

- **Try some variations…**

It is the base content for the s3 bucket. It returns XML that shows the content of the bucket and not the web page

https://nu-cs-msa-s3-photoapp.s3.us-east-2.amazonaws.com/

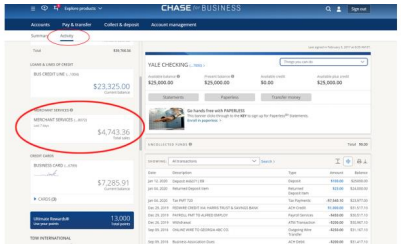
https://nu-cs-msa-s3-photoapp.s3.us-east-2.amazonaws.com/?acl

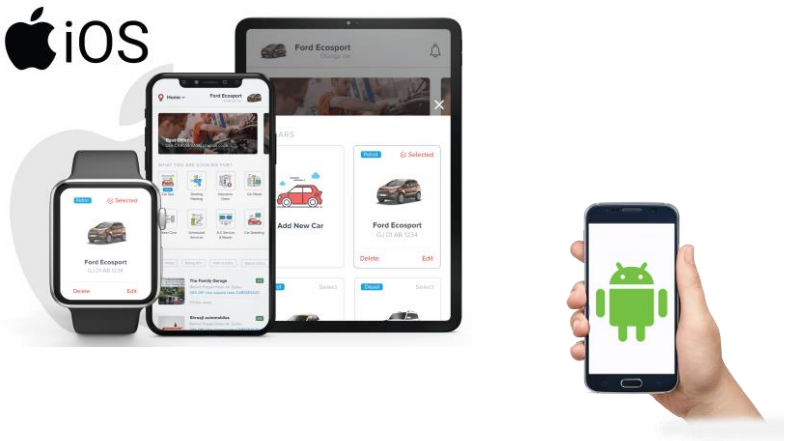
https://nu-cs-msa-s3-photoapp.s3.us-east-2.amazonaws.com/rollcloud.jpg


What's
Going On?

Service-oriented architecture (SOA)

- **Multi-tier evolved to providing services rather than web pages**
  - ***Banking**: login, deposit, withdrawal, transfer, statement*
  - *The service is viewed as a set of functions, allowing client software to have complete freedom over UI*





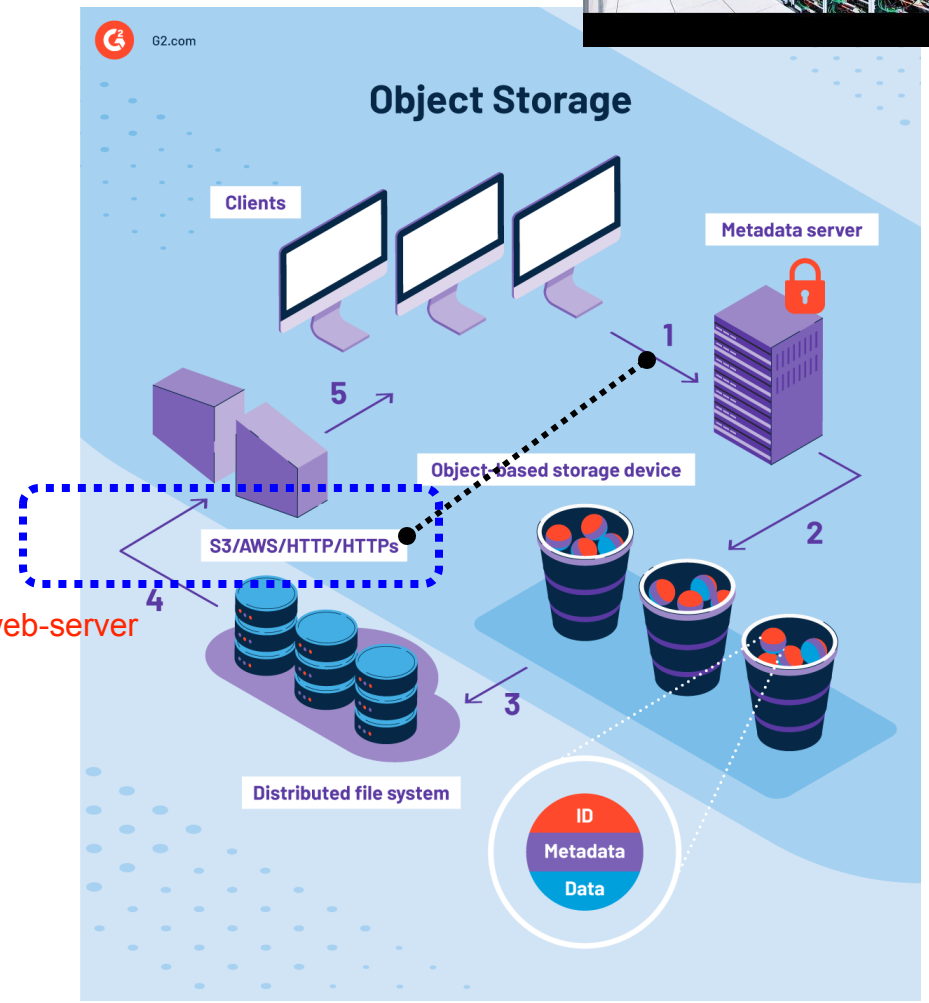
Web servers as a design choice

- **Multi-tier designs often based on web server**
  - *Web server software are a proven technology*
  - *Web servers provide secure communication via HTTPS*
  - *Nearly all devices pre-configured to access web servers*
    - No ports need to be opened in the firewall
    - Network ports 80 (HTTP) and 443 (HTTPS) are open by default

Webserver has API that are functions. These APIs are being called by client and they do some computation and return the data

S3 design choice

- **Amazon chose to implement S3 using web server technologies…**
  - *For all the reasons given on the previous slide*
  - *Even though a custom server would be faster…*



Serving data as web-server

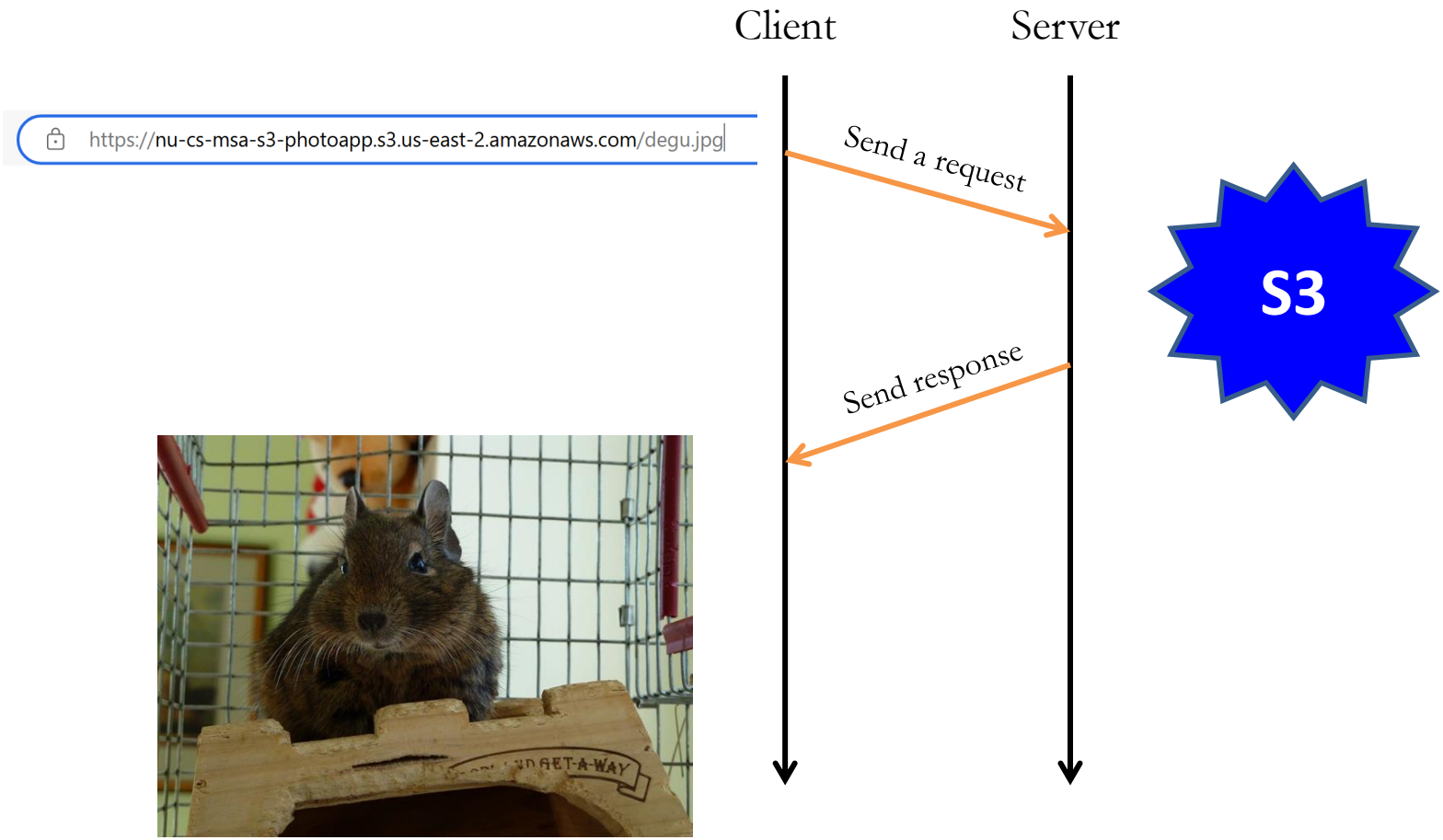
Live coding using trinket.io

https://trinket.io/embed/python3

- **Free site for Python programming**
- **No install or login required**

Service-oriented architecture (SOA)

- **Services adhere to request-response protocol**



Programmatic access: using HTTP get

- **Let's confirm it's a web service by issuing a "GET" request**
- **Grab the Python code from S3 and paste into trinket.io**
  - *https://nu-cs-msa-s3-photoapp.s3.us-east-2.amazonaws.com/main-get.py.txt*

```
import requests

import matplotlib.pyplot as plt
import matplotlib.image as img

##########################################
##
## main
##
print('starting')
print()
#
# eliminate traceback so we just get error message:
#
sys.tracebacklimit = 0

try:
  baseurl = "http://photoapp-nu-cs310.s3.us-east-2.amazonaws.com/"

  #
  # available images: degu.jpg, earth.jpg, rollcloud.jpg, social-n
  #
  prompt = "Enter image name (or press ENTER)> "
  imagename = input(prompt)

  #
  # if no input, default to degu image:
  #
  if imagename == "":
    imagename = "degu.jpg"

  url = baseurl + imagename

  print()
  print("url:", url)
  print()

  #
  # make get request to web service:
  #
  res = requests.get(url)
```

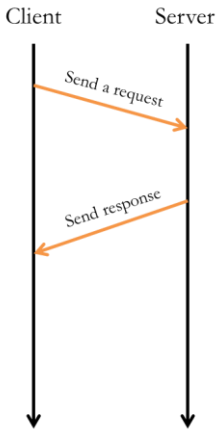
- **Observations:**
  - *Supports 1,000's of users*
  - *Fast even though we are talking to a server in Ohio (250+ miles away)*

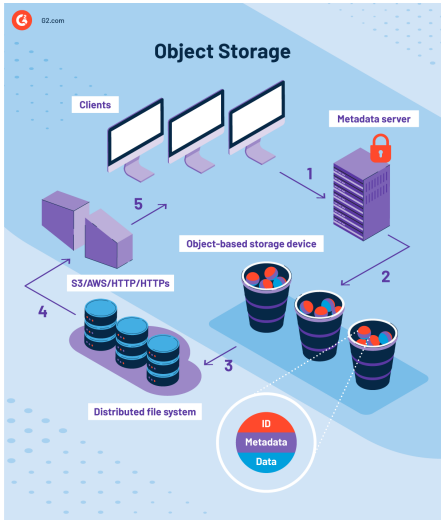
Live coding using boto

- **Amazon provides a Python library called "boto" to hide the low-level HTTP request-response protocol…**

AWS boto library

```
70  #
71  # gain access to CS 310's public photoapp bucket:
72  #
73  try:
74    s3 = boto3.resource(
75      's3',
76      region_name='us-east-2',
77      # enables access to public objects:
78      config=Config(signature_version=UNSIGNED))
79
80    bucket = s3.Bucket('photoapp-nu-cs310')
81
82  except Exception as e:
83    logging.error(e)
84    sys.exit(-1)
85
86  #
87  # loops through the test folder and displays each image
88  #
89  # NOTES:
90  #   1. make sure "output" tab is open to see image
91  #   2. click "X" in upper-right of image to advance
92  #
93  folder = "test/"
94
95  #
96  # TODO
97  #
98
```






```
for asset in bucket.objects.filter(Prefix=folder):
  if not asset.key.endswith('jpg'):
    pass # skip folders, non-images:
  else:
    print('downloading', asset.key)

    filename = 'temp.jpg'

    bucket.download_file(asset.key, filename)

    image = img.imread(filename)
    plt.imshow(image)
    plt.show()
    os.remove(filename)
```

That's it, thank you!