- POST api endpoint
  POST http://localhost/api/v1/scan
  ```
  {
          Hostname:[
                  "Freecodecamp.org"
                  "14"
          ]
  }"
  ```

    - **Implemented -** This endpoint will take a list of hostnames or ip addresses as input and will scan for 3 ports 80,443 and 843 and return current status if it's open or closed.
    - **Todo -** End idea is to take a list of ports in the api as a separate parameter by hostname. And if none ports are provided then scan all common ports list.

Logic Processing

- We will process input list (hostnames) as parallel in go channels initially and when we start processing more than 5 ports for each host then we will have channels for ports as well with ranges
    - Go(){
        - Process ip address - in one channel with a port range of 70 to 1000

Why go lang?

- Planning to support more than 1 million requests per second, Golang offers simpler multi-threading and has better support to utilize multi-core. Another possibility will be python. Also there is nmap package in go which can be used to write your own nmapper.
- Also its open source with a lot of community support. It has lot of cloud and containerized examples.

Data Store

- Initially I would start with RDBMS something like postgres and store below entities.
- Scan id will store information related to each scan and will be used as FK to get previous status of the run and network.

    - scans
      ```
      -----
        scan_id (PK)
        start (this must be a datetime)
        end (this must be a datetime)
        version
        arguments
      ```

    - hosts
      ```
      -----
      ```

  host_id (PK)
  scan_id (FK to scans.scan_id)
  name
  ip_address
  mac_address
  Status

- ports
  -----
    port_id (PK)
    host_id (FK to hosts.host_id)
    number
    Status

- This will give us the option to separate out api endpoints for hosts information as well as getting historical data.

Scaling for Future

- This app will be containerized and hosted on a private cloud. Reason to choose private cloud as this app is scanning networks for some other hosts and stores that information.
- It will be easier to provide protection and easier to iterate over the data or features.
- As app will be containerized we can leverage techs like kubernetes or docker and host them on cloud or on prem(my recommendation is on prem for this api).
- Api will be hosted behind Load balancers to provide High availability and scalability. App can be scaled based on the number of requests being processed by a single container. We can use either request count or number of thread (cpu) usage to scale the app.
- We are using RDBMS and it can become a bottleneck once we grow a business with more than 1 million requests per second. So storing some common information like ports list can be stored in a cache like redis.
- Also based on IPaddress and if its been scanned more than one time in the last 1 hour then we can store its information in Cache as well. For example the entry IP address of any network can be stored in cache and keep checking deltas and see if the ip address network values are changed then we can call process it and update DB.
- Eventually moving hosts and ports DB to no sql will make sense as it will become read heavy in the future as IP addresses might start repeating. But again relational db with proper data sharding across different regions can be enough.
- App will be hosted based on region to minimize too many network hops, and data can be shared using a queue between regions to provide resilience and scalability.