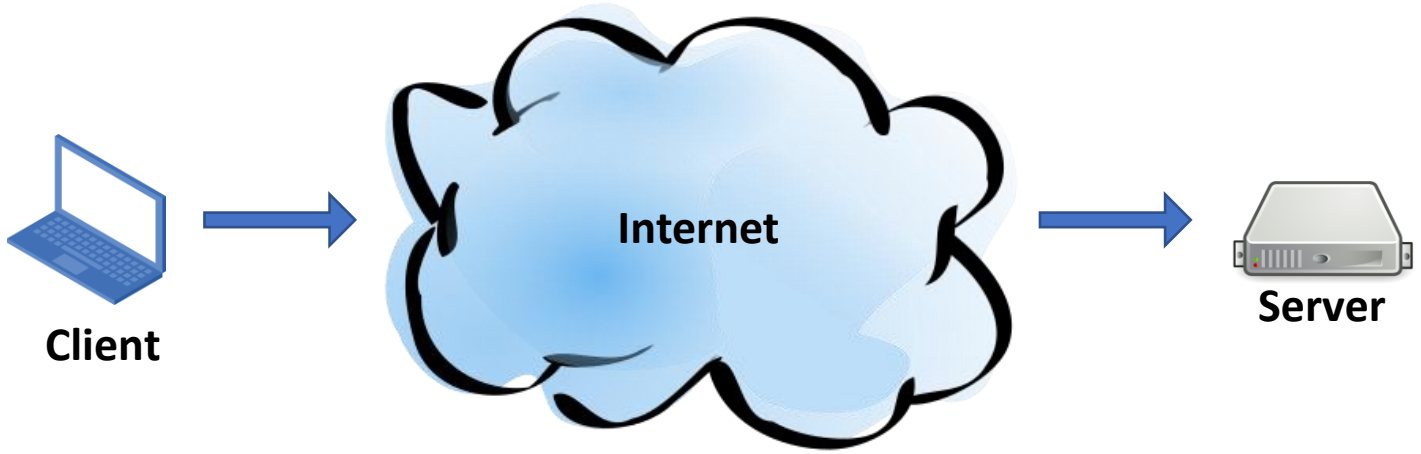
CompSci 401: Cloud Computing

Client-Server Applications

Prof. Ítalo Cunha

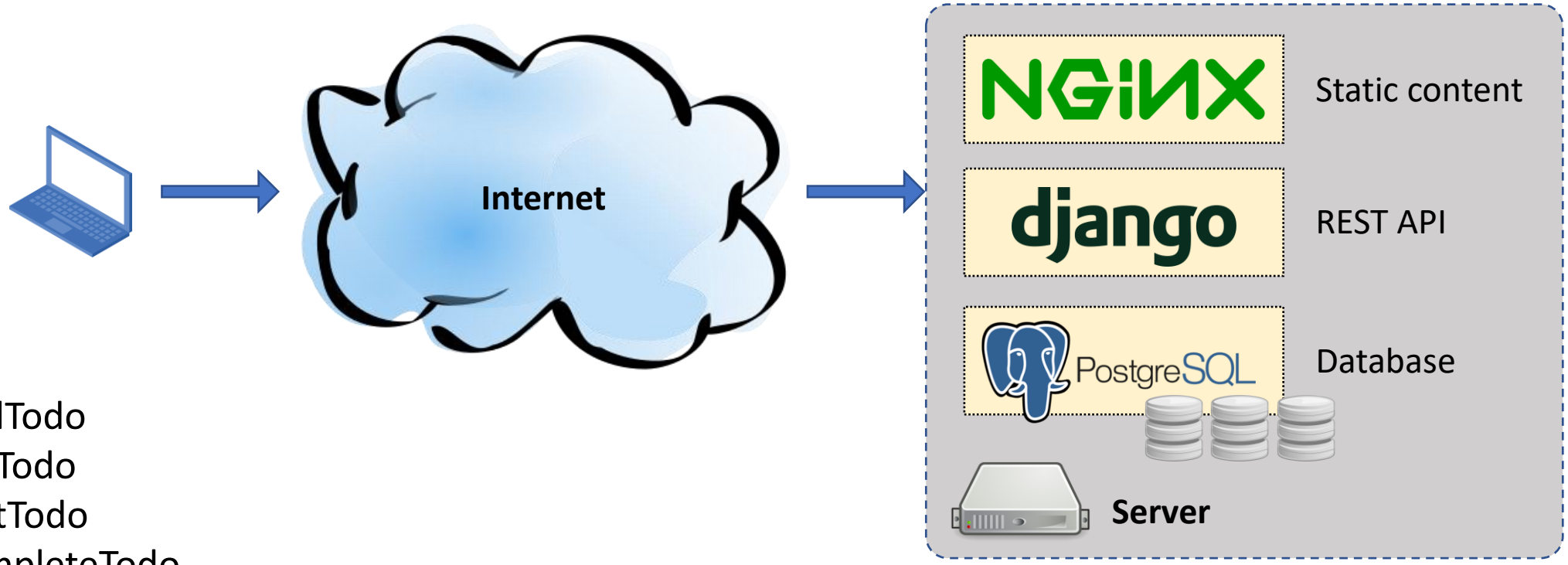
昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Server listens for client connections



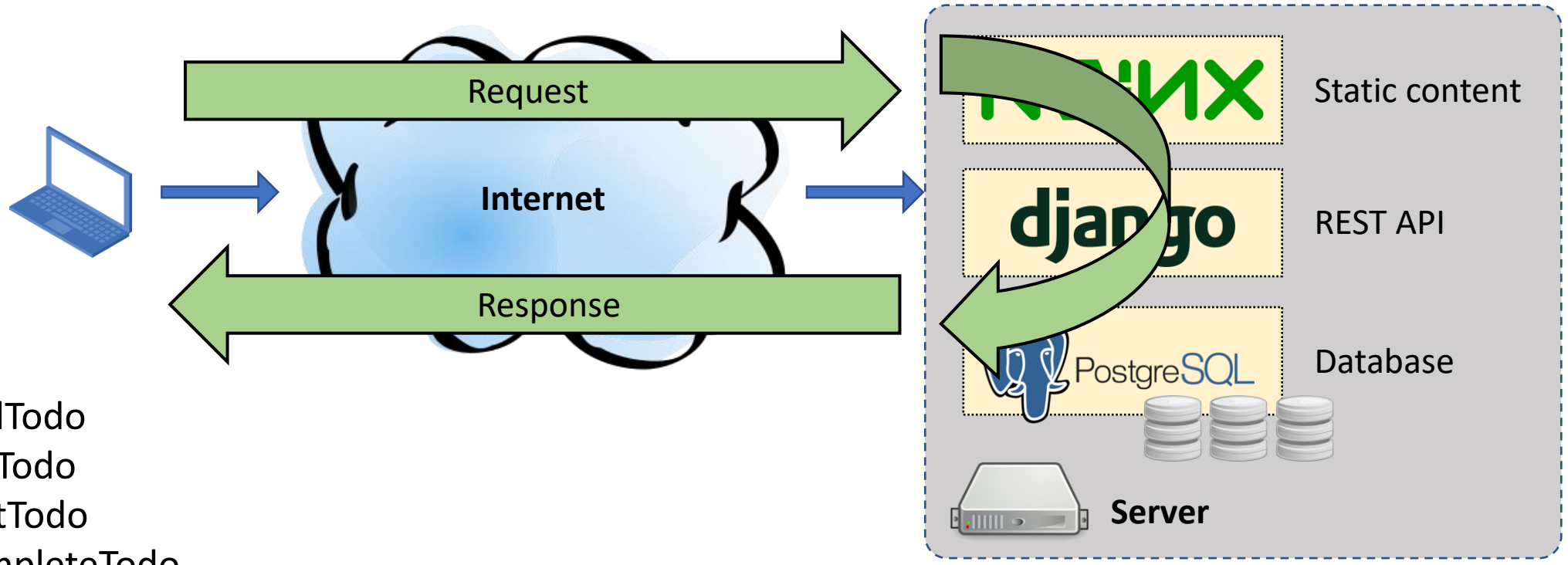
1. Server starts, listens for connections
2. Client starts later, starts a connection
3. After connection is established, data can flow in both directions

Example “to-do list” app with a REST interface



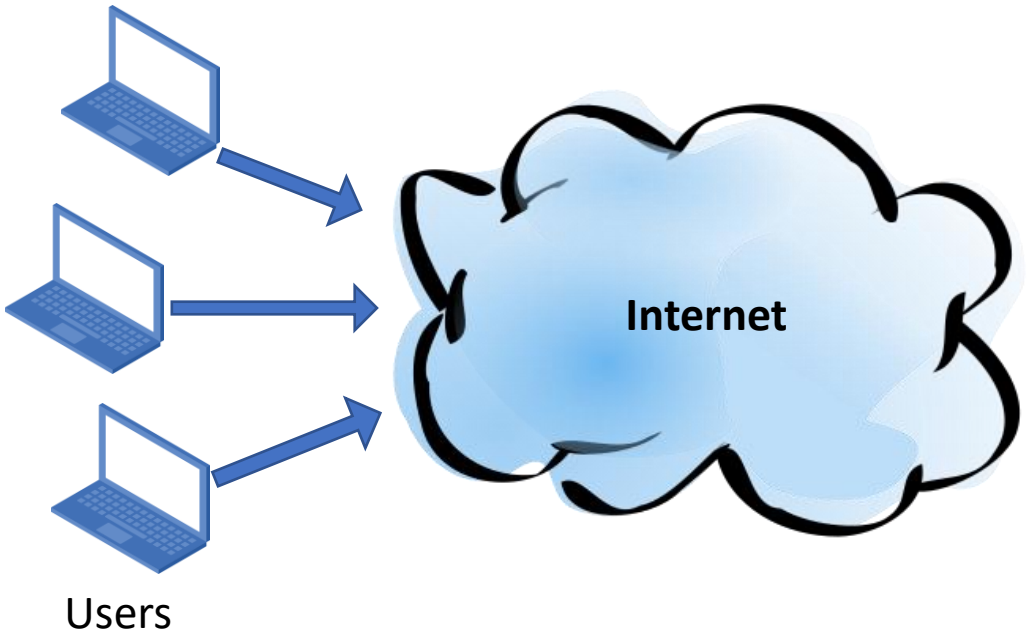
- addTodo
- getTodo
- editTodo
- completeTodo
- deleteTodo
- listAllTodo

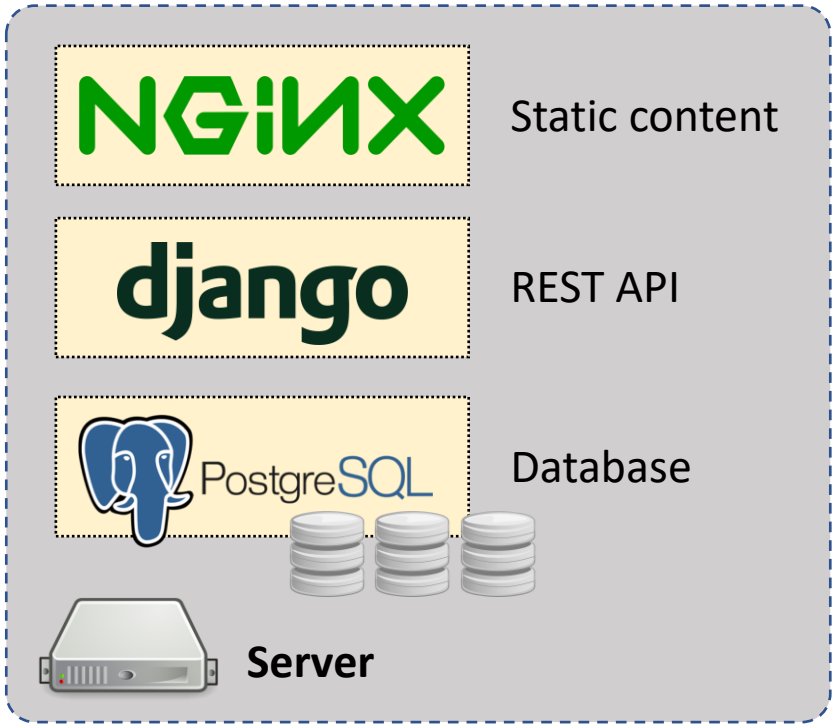
Example “to-do list” app with a REST interface



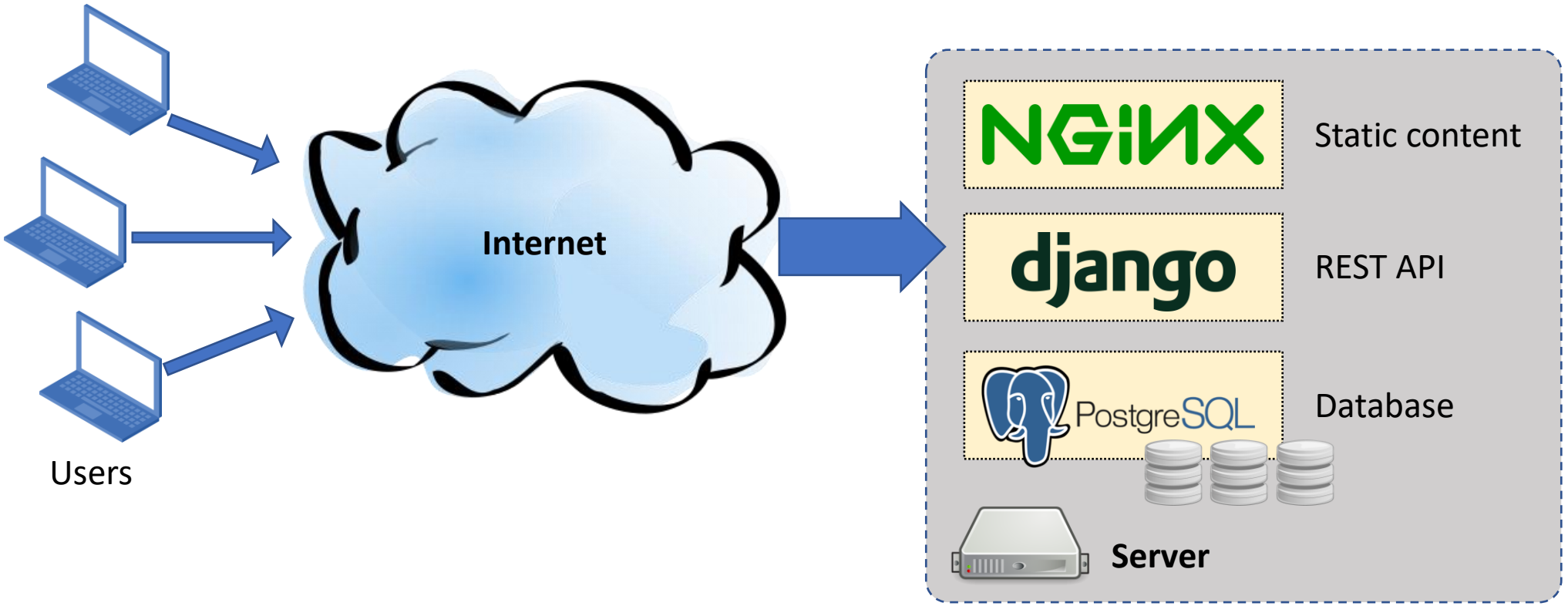
- addTodo
- getTodo
- editTodo
- completeTodo
- deleteTodo
- listAllTodo

How to handle multiple clients?



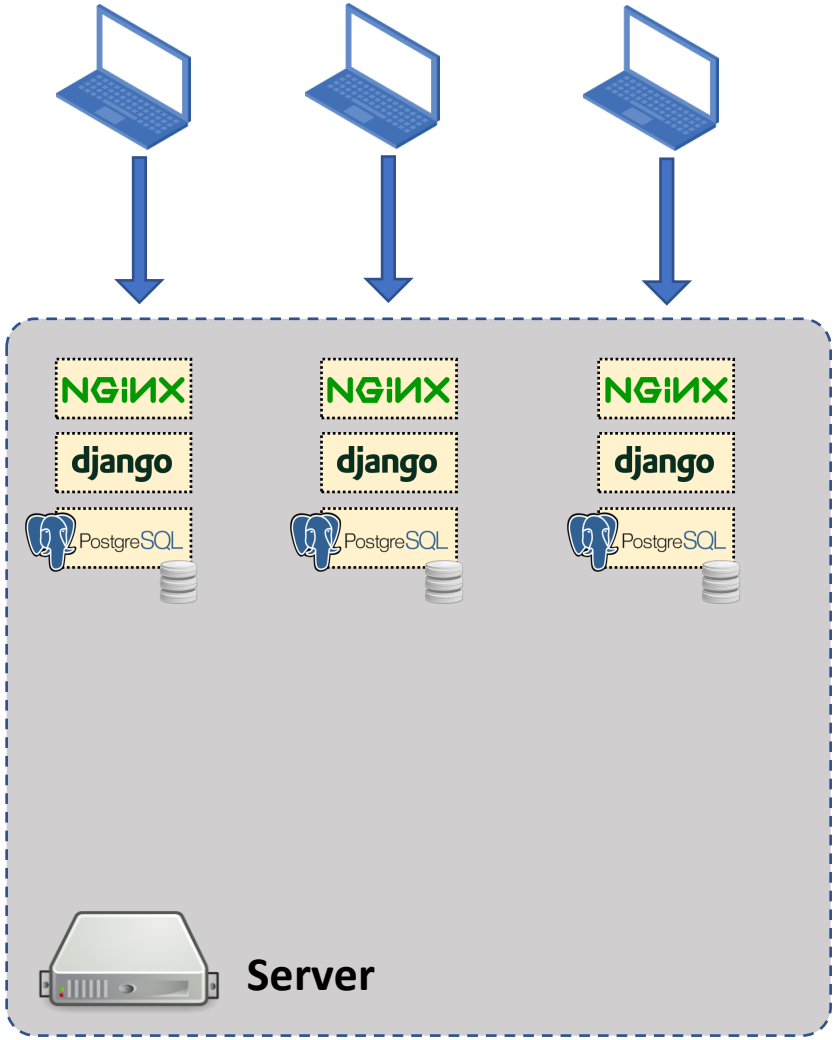


How to handle multiple clients?



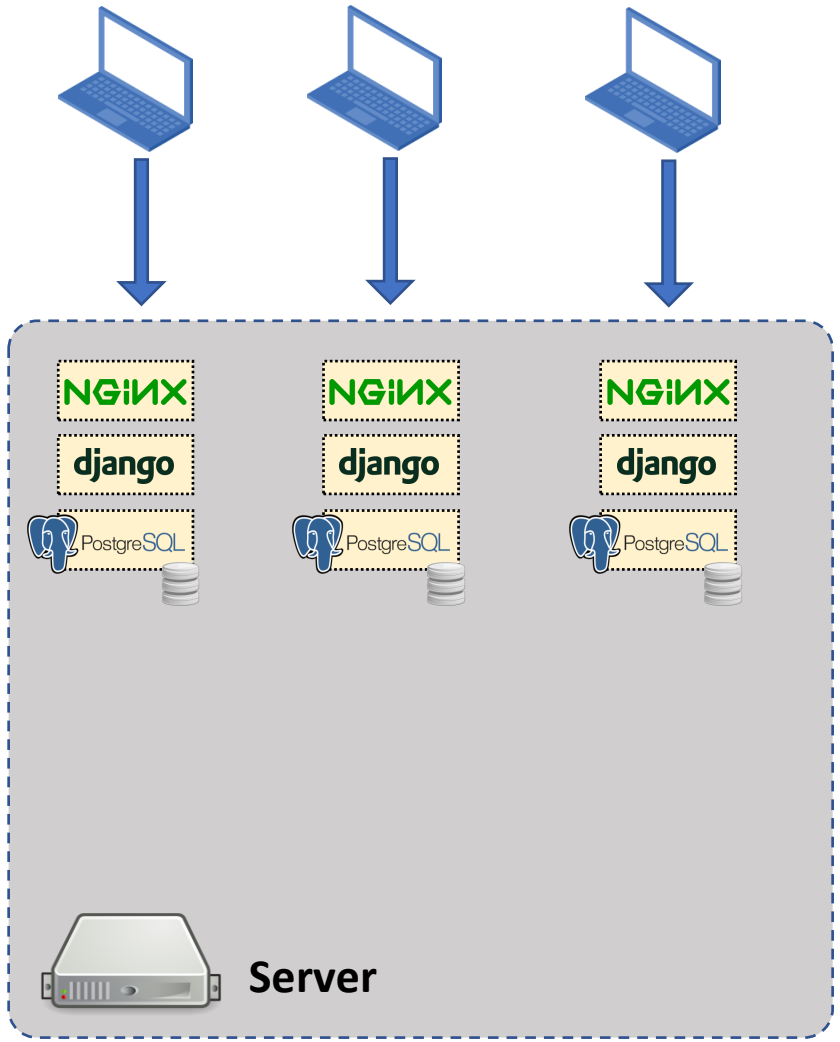
How to handle multiple clients?

Synchronous architecture with threads/processes

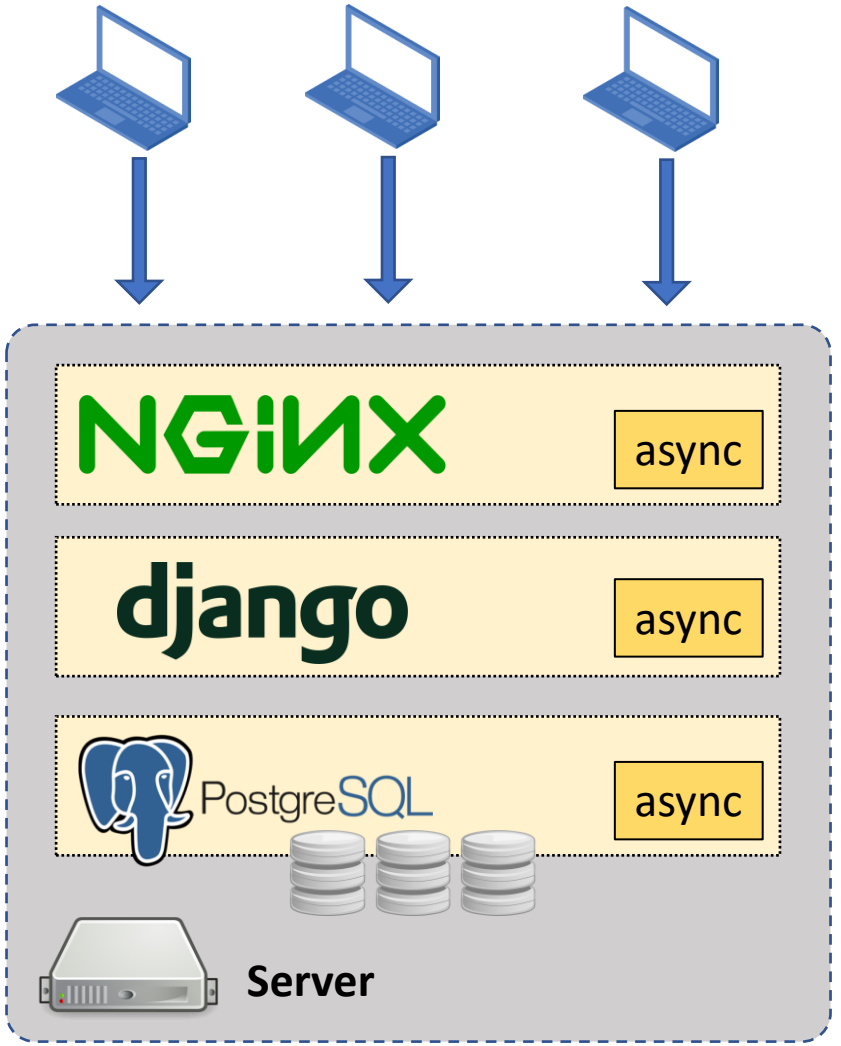


How to handle multiple clients?

Synchronous architecture with threads/processes

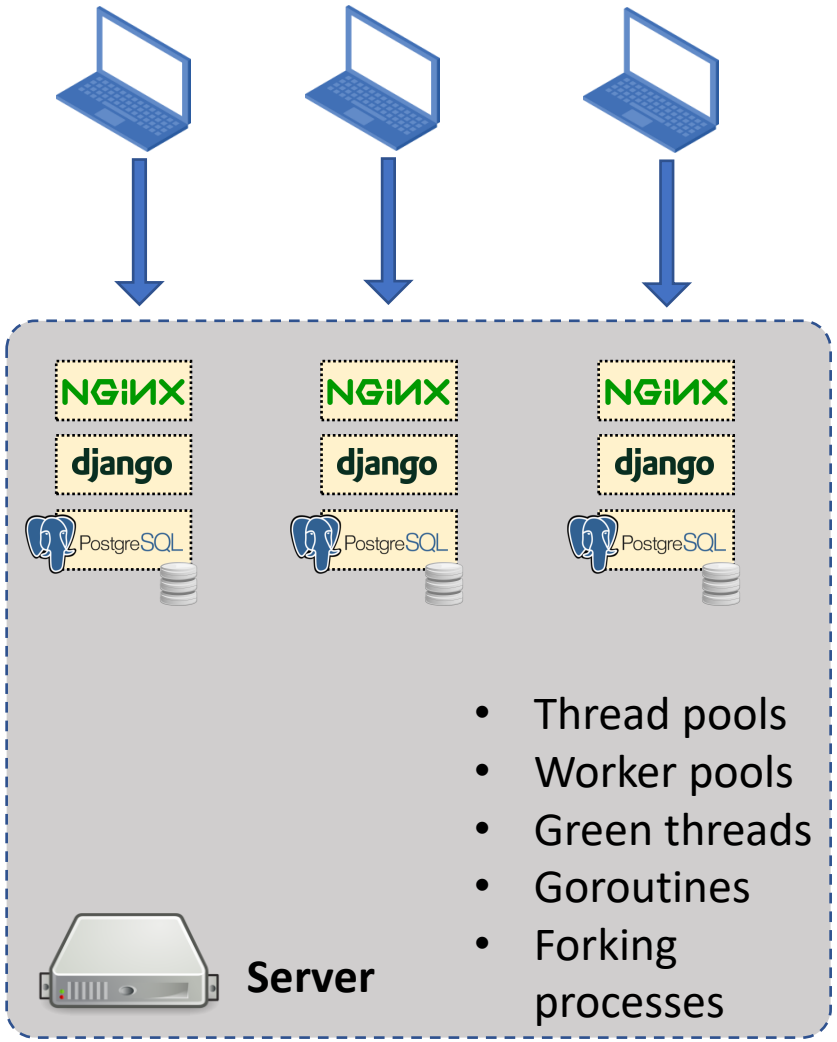


Asynchronous architecture



How to handle multiple clients?

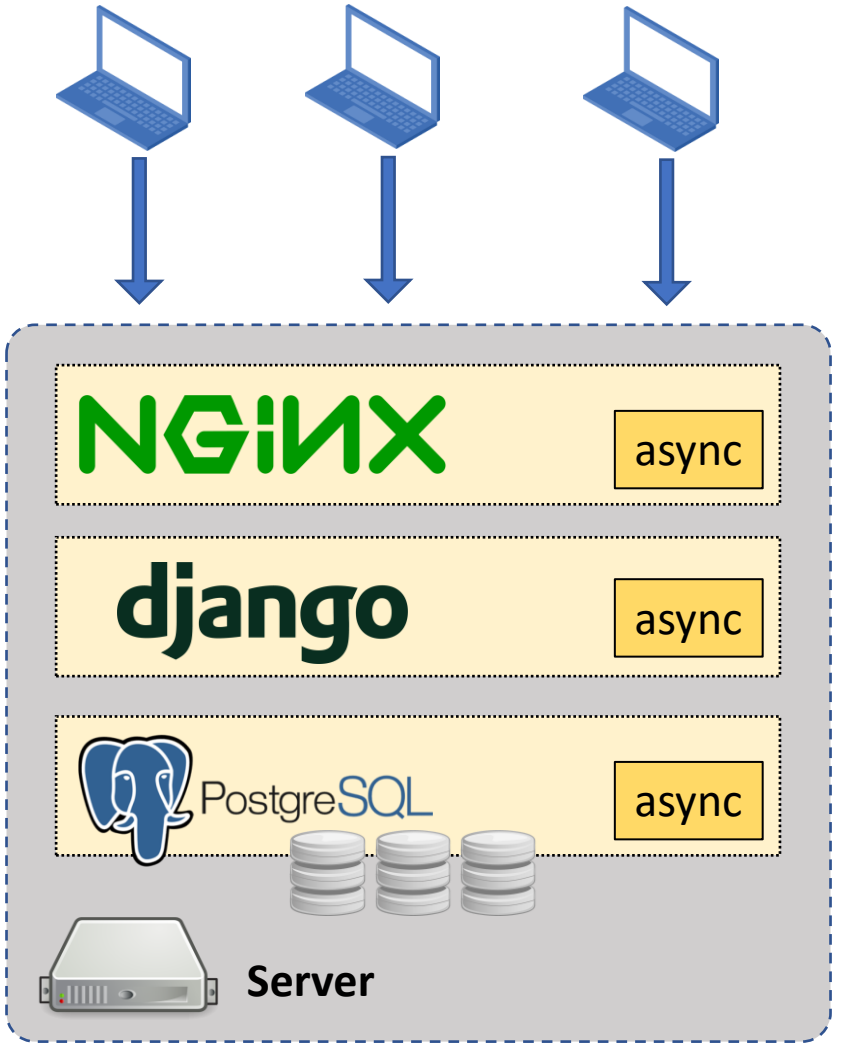
Synchronous architecture with threads/processes



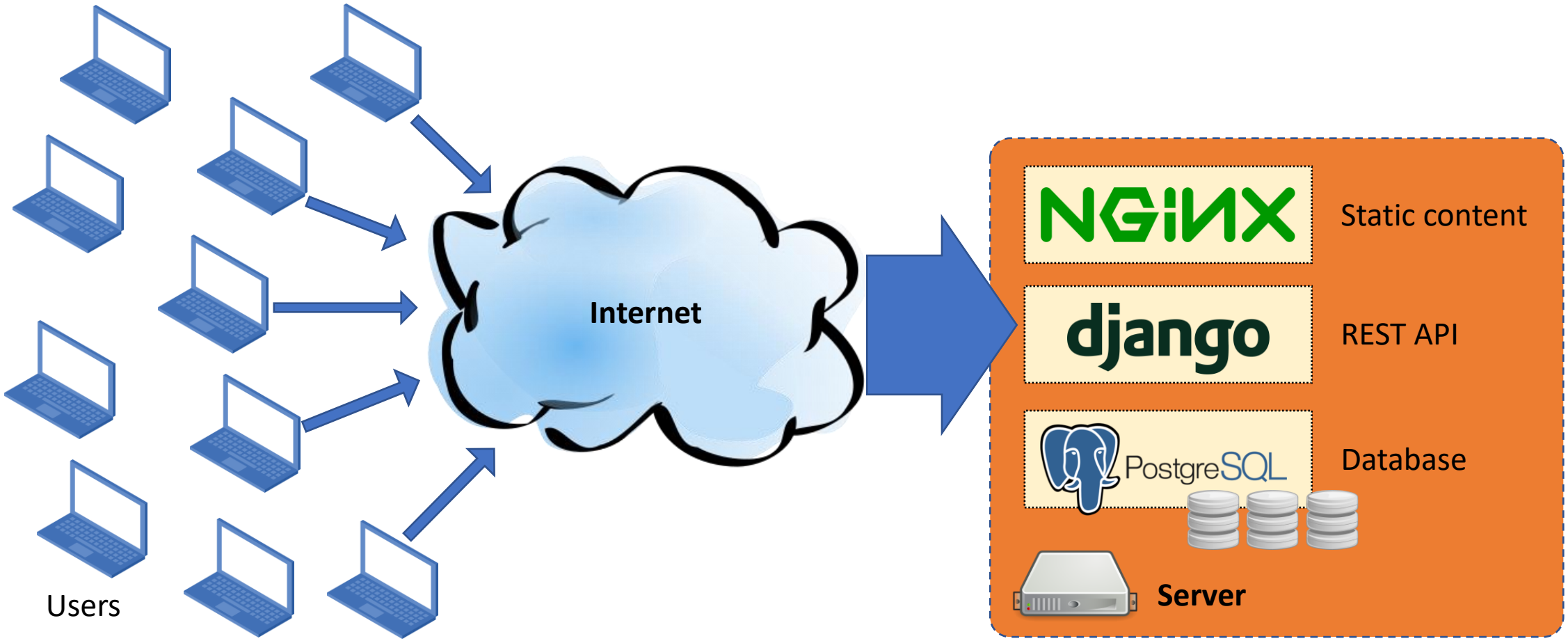
- Thread pools
- Worker pools
- Green threads
- Goroutines
- Forking processes

- select()/poll()
- Async libraries
- Event loops
- Callbacks
- Futures

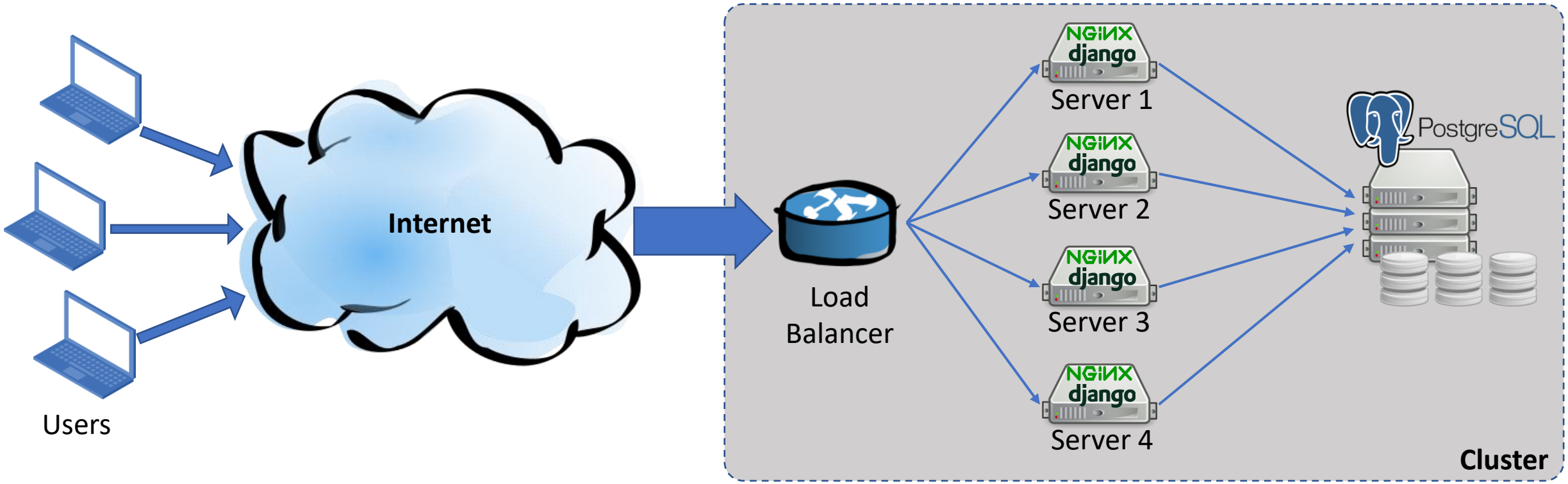
Asynchronous architecture



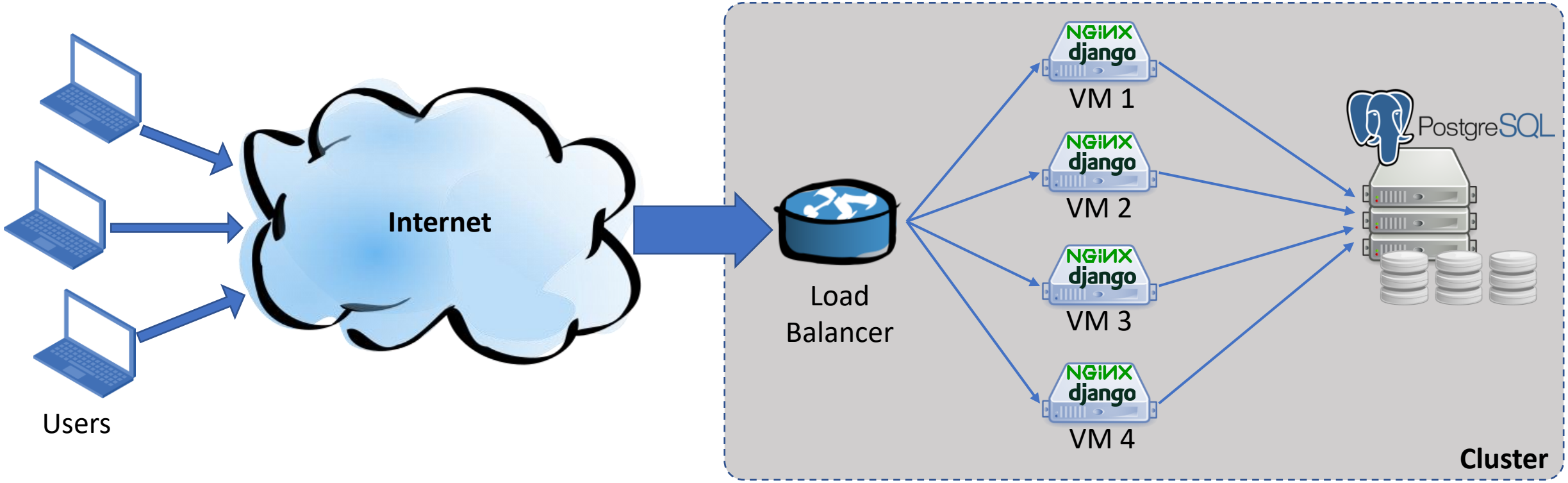
A server can handle only so many clients



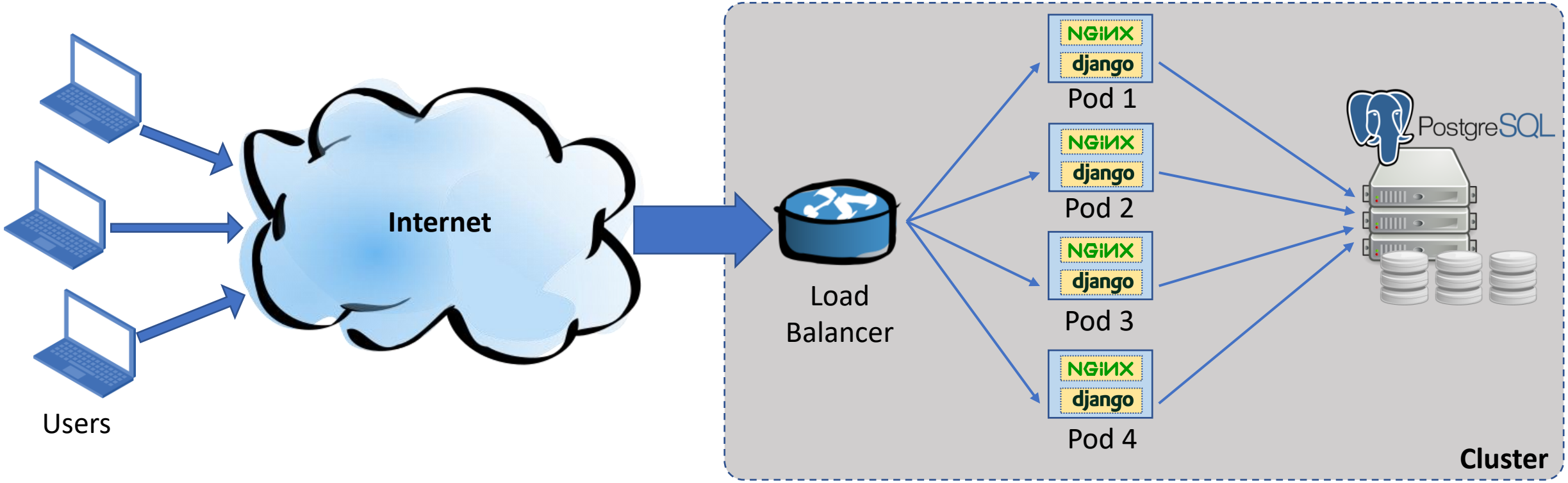
Distributed Web services



Distributed Web services in virtual machines



Distributed Web services as microservices



Properties of the microservice approach

- Need a minimum number of instances running
  - Regardless of load
  - Incurs cost
- Some components may be hard to scale
  - For example, the database
- Development complexity
  - Load balancing
  - Pods, containers, packaging Nginx and Django
  - Configuring and managing Postgres

Properties of the microservice approach

- Need a minimum number of instances running
  - Regardless of load
  - Incurs cost
- Some components may be hard to scale
  - For example, the database
- Development complexity
  - Load balancing
  - Pods, containers, packaging Nginx and Django
  - Configuring and managing Postgres

Worse for virtual machines

CompSci 401: Cloud Computing

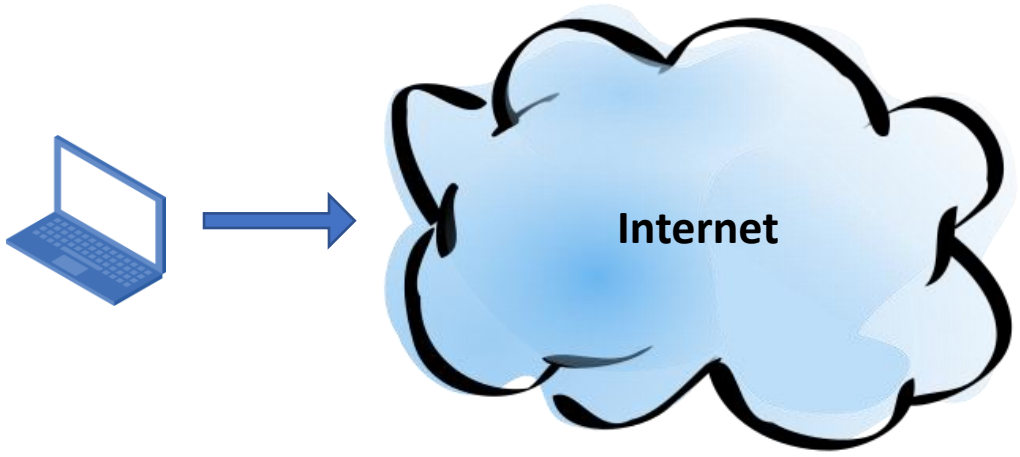
Serverless Computing

(or Function as a Service)

Prof. Ítalo Cunha

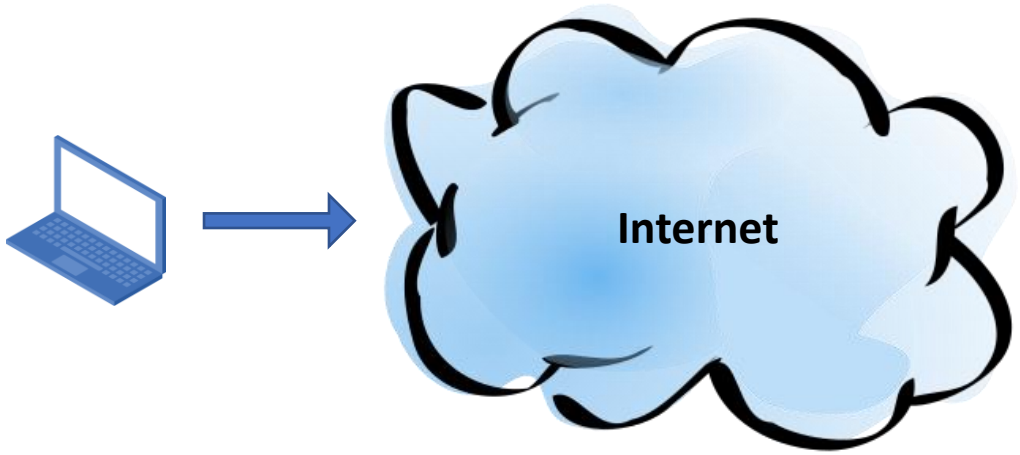
昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Example “to-do list” app with a REST interface



- addTodo
- getTodo
- editTodo
- completeTodo
- deleteTodo
- listAllTodo

Example “to-do list” app with a REST interface



- addTodo
- getTodo
- editTodo
- completeTodo
- deleteTodo
- listAllTodo

addTodo λ

getTodo λ

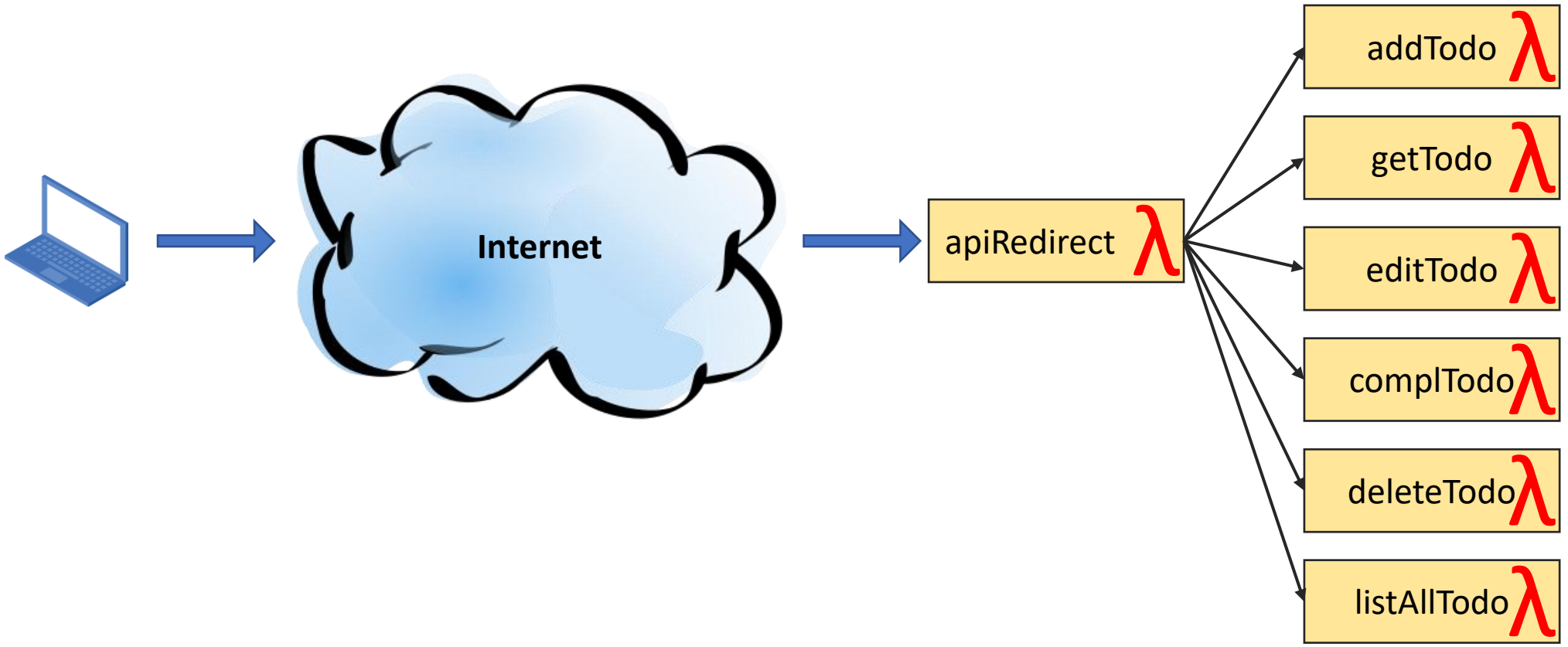
editTodo λ

complTodo λ

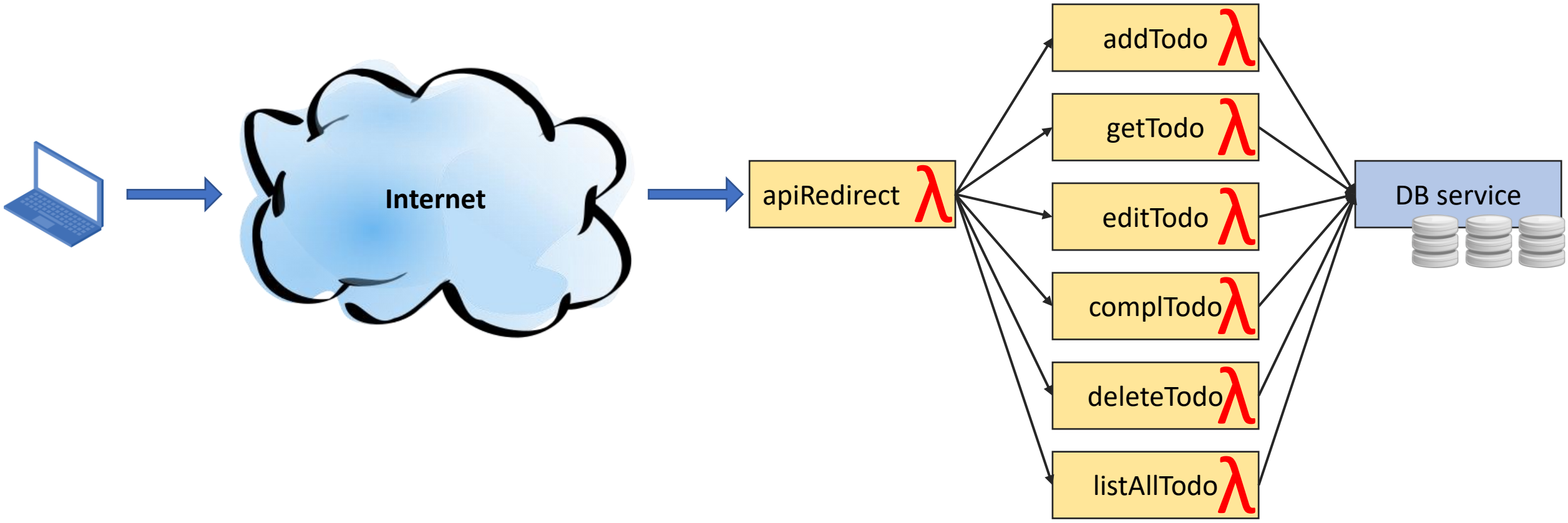
deleteTodo λ

listAllTodo λ

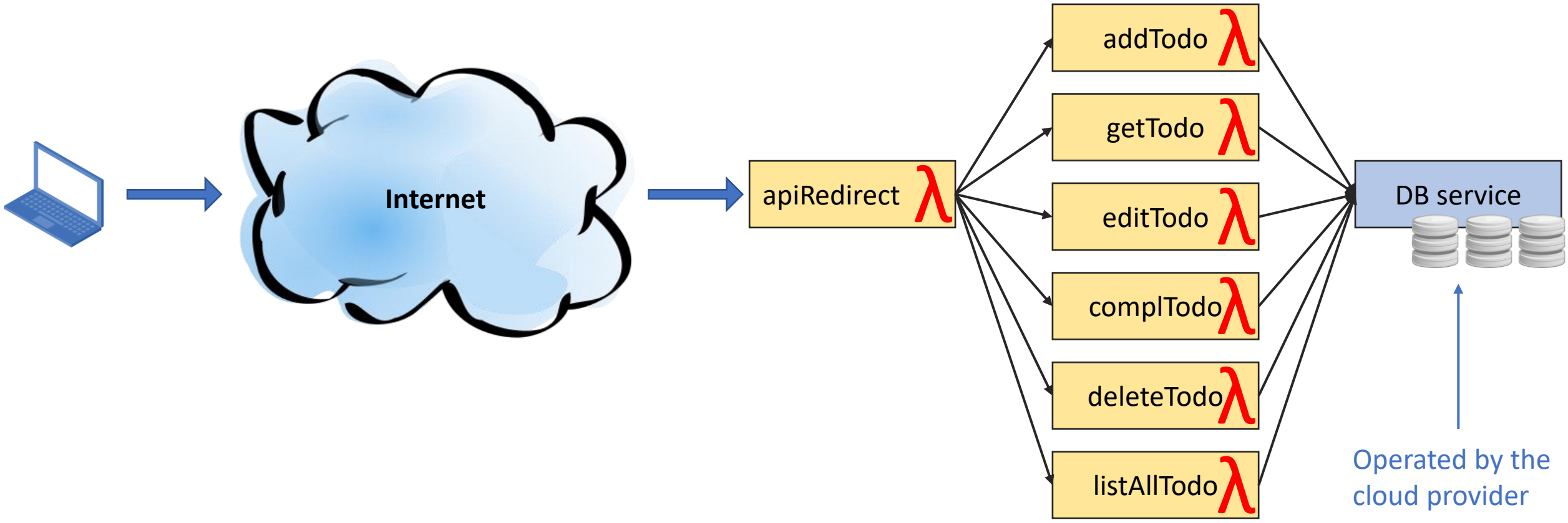
Example “to-do list” app with a REST interface



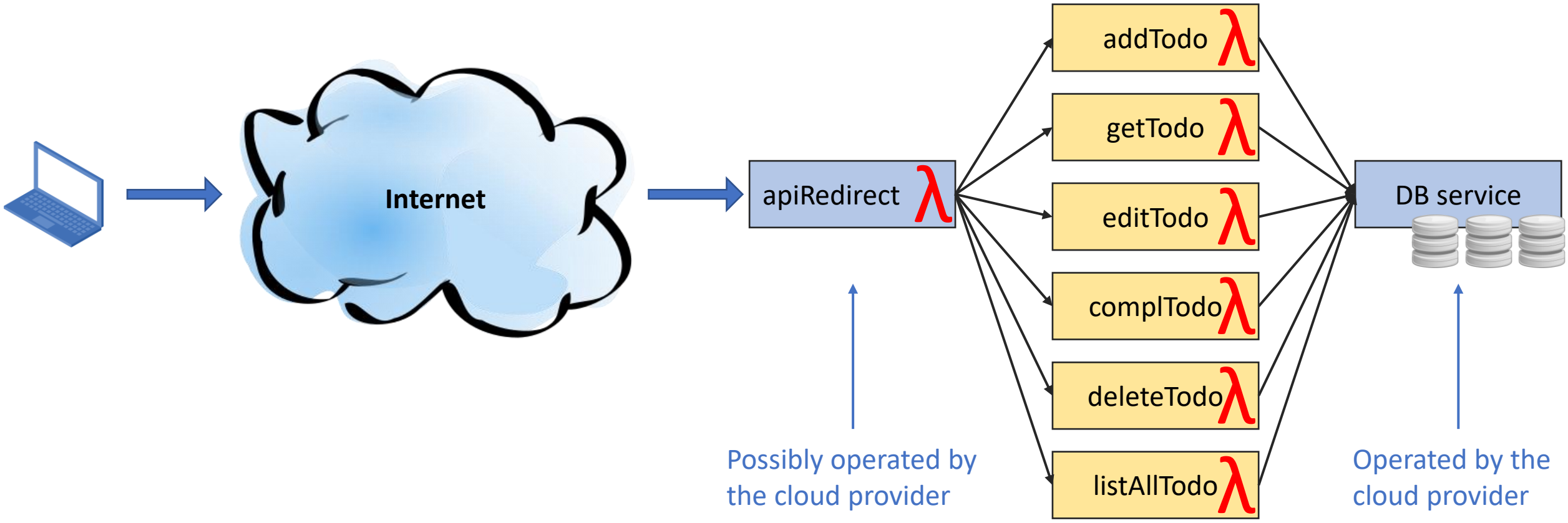
Example “to-do list” app with a REST interface



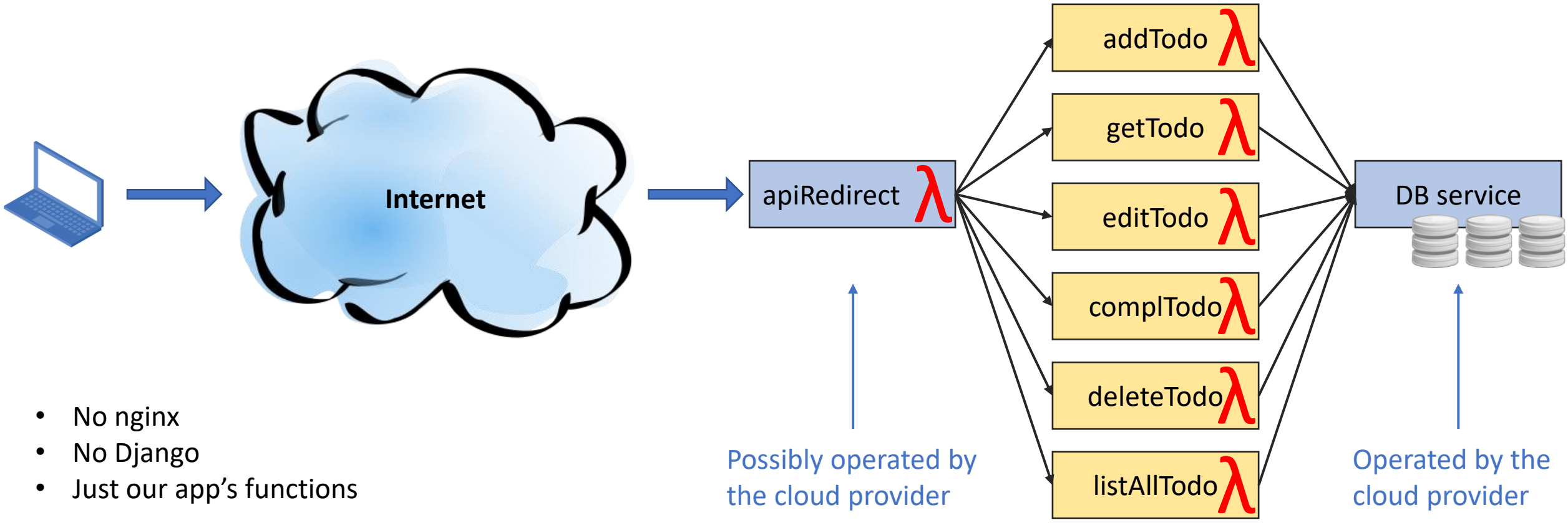
Example “to-do list” app with a REST interface



Example “to-do list” app with a REST interface



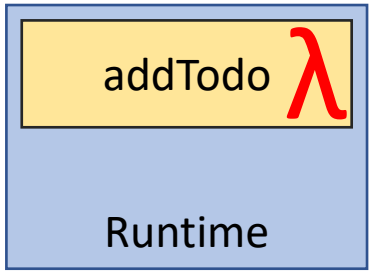
Example “to-do list” app with a REST interface



- No nginx
- No Django
- Just our app’s functions

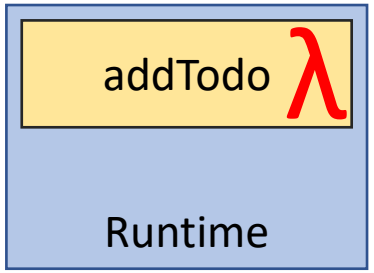
Magic? Custom runtime by the cloud provider

- Functions run inside a runtime
  - Runtime has monitoring and facilities required by orchestration
  - Reads incoming messages/events and calls our functions
  - Forwards results to other functions or databases
- Runtime imposes constraints on functions



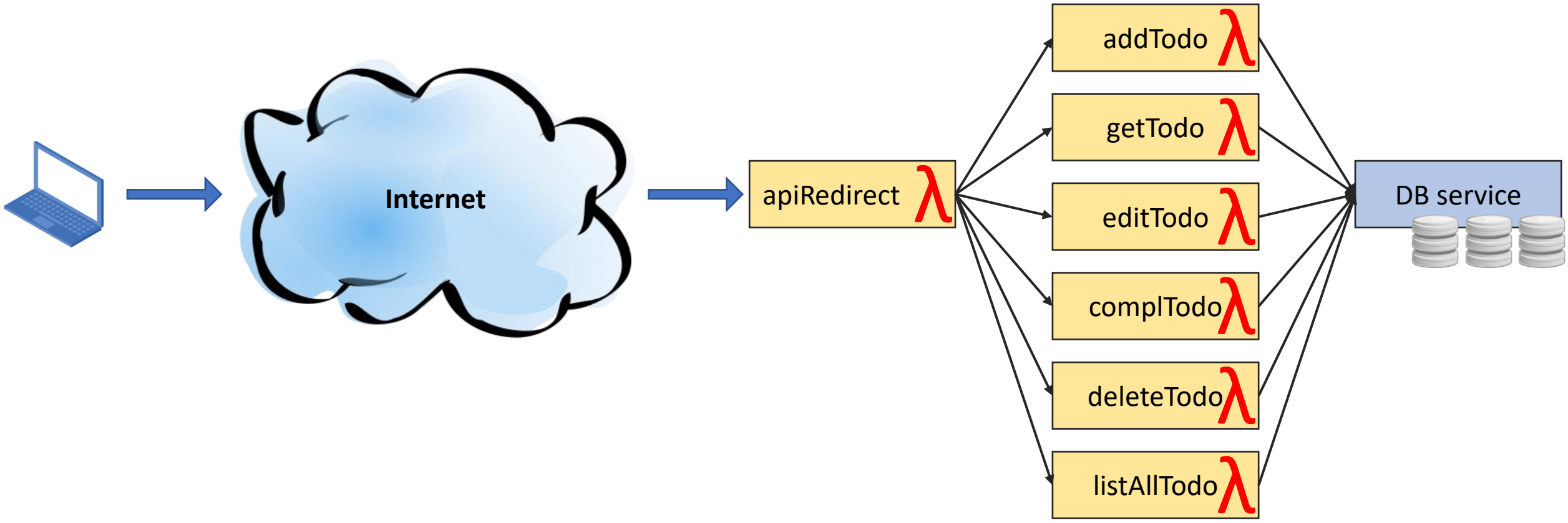
Magic? Custom runtime by the cloud provider

- Functions run inside a runtime
  - Runtime has monitoring and facilities required by orchestration
  - Reads incoming messages/events and calls our functions
  - Forwards results to other functions or databases
- Runtime imposes constraints on functions

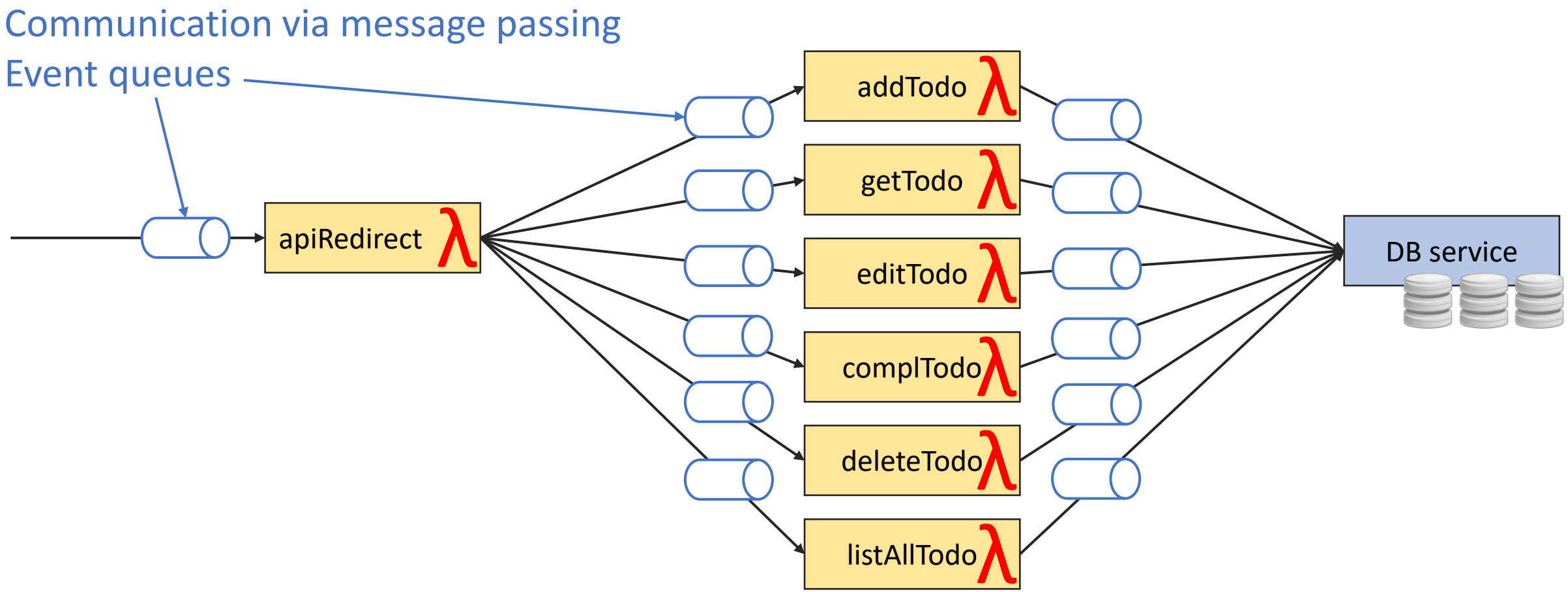


Amazon AWS Lambda natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby

Event-driven and stateless functions



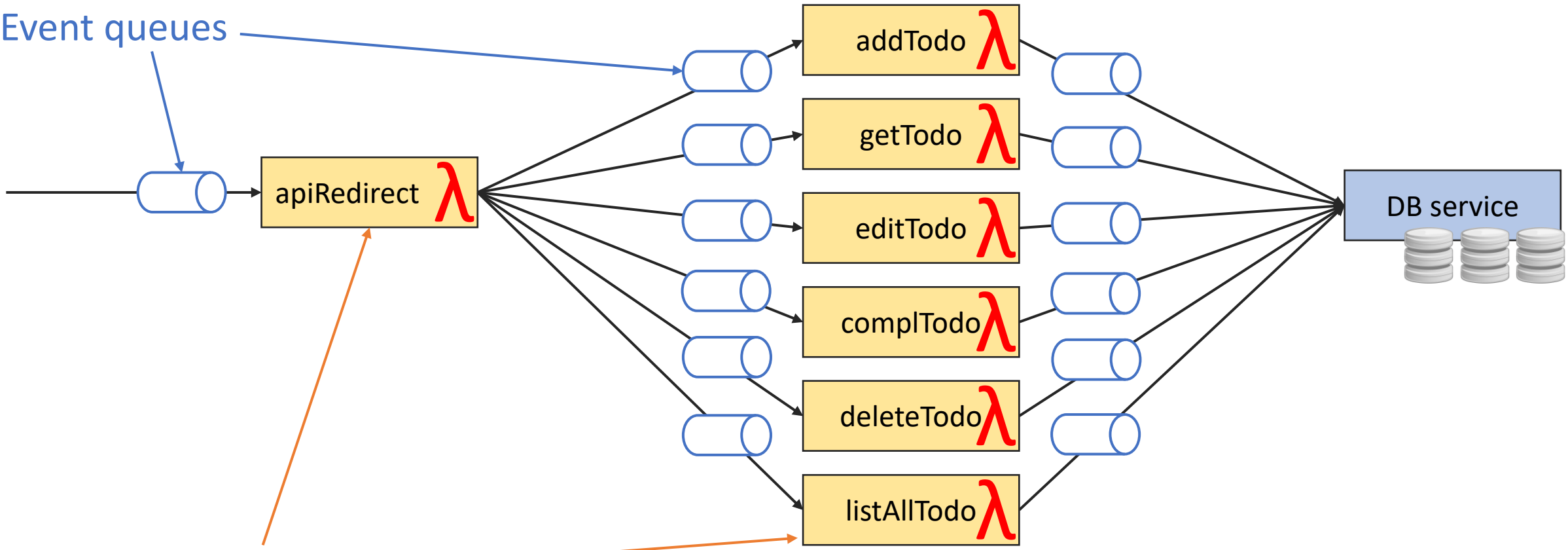
Event-driven and stateless functions



Event-driven and stateless functions

Communication via message passing.

Event queues



Stateless functions

Functions may be shut down if there are no events. All state in database.

Properties of serverless computing

- Developers can focus on their application business logic
- Reduced management complexity
  - No containers
  - No VMs
  - Built-in orchestration
  - Can avoid configuring databases
- The serverless computing paradigm has built-in scalability
  - Possibly lower costs
  - Scale to zero means the function may have no instances and use no resources
  - Infinite scalability to arbitrary workloads

Disadvantages of serverless computing

- Runtime limitations on functions
  - Programming languages are supported natively
  - Resource limits (e.g., RAM) on functions
- Starting functions to process an event incurs additional delay
  - May be impactful for delay-sensitive applications
- Possibly harder debugging
  - Limited support for introspection, closed proprietary runtime
- Vendor lock-in