Inter-Service Messaging

Created	@February 27, 2022 8:45 PM
Type	Lecture
Week	10
Lecture #	1
Lecture URL	https://youtu.be/ffHIEp4smlQ
Notion URL	https://21f1003586.notion.site/Inter-Service-Messaging-c55d06d5f29b4dc2a4ebea0d1ce657f1

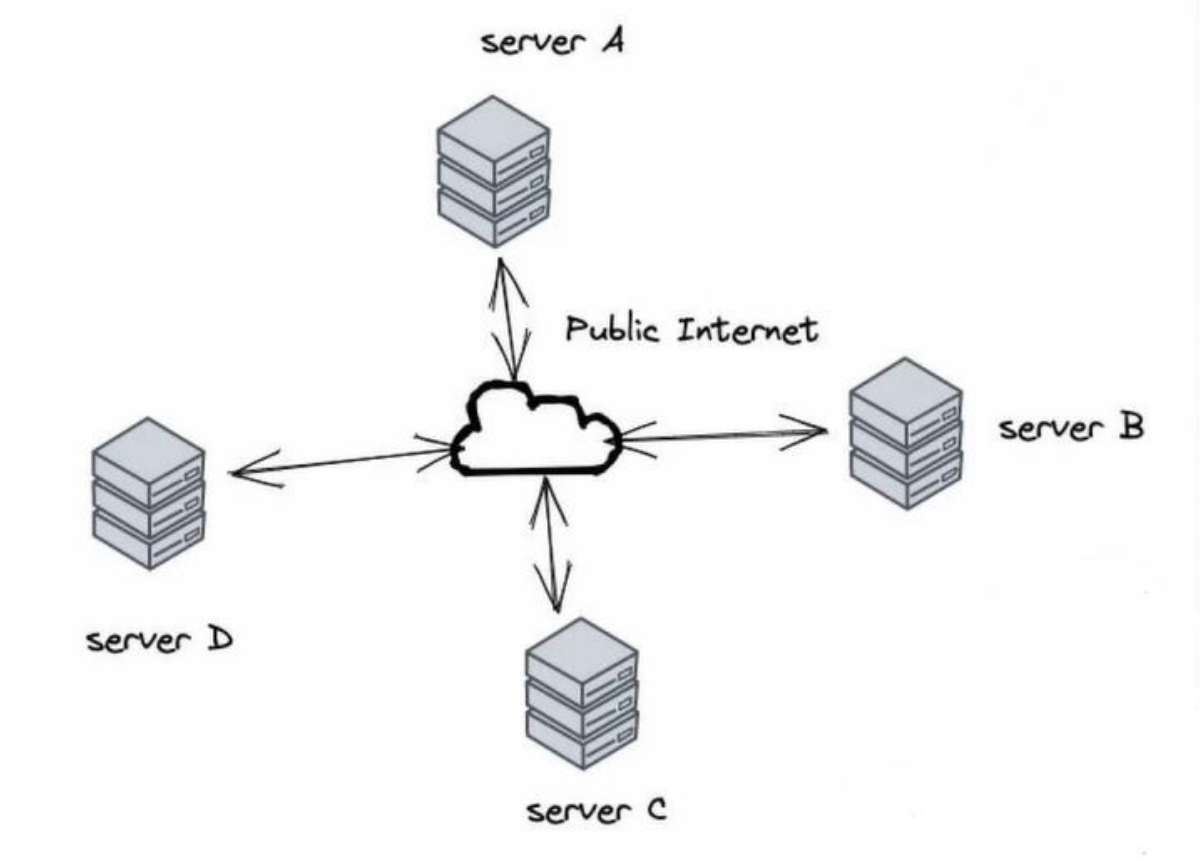
Message Queues

- Multiple services
  - Closely coupled
  - Running on same or closely related servers
  - Example: frontend, email, database, image processing
- Asyncronous message delivery

- Guarantees of delivery
- Ordered transactions

Internet-distributed Services

- No common message broker
- Servers expose services for public use
- May not need delivery or order guarantees
- Lightweight messaging



Lightweight API calls

- Server exposes certain endpoints
- Meant for others to PUSH messages, not retrieve data
- Request usually through POST, maybe GET
- Data payload may be trivial on even non-existent

Why? to receive message from others

Examples

- Every time there is a commit pushed to GitHub, send a message on Google Chat room
  - GitHub allows us to register a URL
  - We create a server to receive the request and then push to GChat
- Use Twilio to send several messages
  - Twilio calls you back when done with messages
  - We don’t have to keep checking status from Twilio

Webhooks

- Use the existing web infrastructure to send messages
- Server to server communication
  - Usually ... can also be direct client invoking a hook on the server
- Simpler than message queues

Webhooks

Created	@February 27, 2022 9:10 PM
Type	Lecture
Week	10
Lecture #	2
Lecture URL	https://youtu.be/vVXiXZhn7hM
Notion URL	https://21f1003586.notion.site/Webhooks-f90228839c1d4653a6fac88f17207c34

What is a webhook?

A way foran “app” to provide other “apps” with realtime information

- Also called a web callback or HTTP Push API
- Regular HTTP request
  - Uses standard HTTP protocol
  - Usually either POST or GET
- Sometimes called reverse API

    - Similar specifications to regular APIs
    - Usually only to push information, not retrieve data
- Synchronous
    - No store, retry, etc. — may trigger other behaviour, but webhook response must be immediate

  https://sendgrid.com/blog/whats-webhook/

Example webhook: gitlab

- Setting up a receiver
    - https://requestbin.com/r/enhd2m5q5rmi8
    - https://replit.com/@nchandra/webhooktest#main.py

More about Webhooks

Created	@February 27, 2022 9:21 PM
Type	Lecture
Week	10
Lecture #	3
Lecture URL	https://youtu.be/pQVX4kn6AXU
Notion URL	https://21f1003586.notion.site/More-about-Webhooks-6b8d25f073304d6a895234abc79d0d1d

Message contents

- Entirely application dependent
- Keep to minimum
  - Not meant for transfer of large amounts of data
  - Only as a message
- Request body

Message response

- Webhooks are “machine called”
  - Invoked by another server, not a human client
- Response should just indicate status
  - 200 for success, 4xx for failures
  - Minimal data returned — mostly will de discarded

Webhooks vs ...

- Websockets
  - Realtime 2-way communication vs One-way
  - Keep connection open
  - Custom protocol vs standard HTTP
- Pub/Sub
  - Message queue — but public: Google cloud, Apache Kafka, etc
  - Asynchronous
  - More oriented to message delivery and processing — may be overkill
- Polling
  - Periodic requests to check status
  - PULL, not PUSH
  - Server can be overloaded when client numbers increase
- APIs
  - Collection of endpoints
  - Any client can send request to API
  - Usually used for retrieving data
  - Webhook uses a form of API more suited for pusing messages

How to ...?

- Consume webhook
  - Create URL endpoint, register with provider (eg. gitlab, twilio, ...)

- Debugging
  - requestbin: dummy endpoint to retrieve data and see content
  - curl or postman: API debugging tool
  - ngrok or cloudflare argo tunnel: public URL endpoint for private code without needing public IP
- Securing webhook
  - Restrict at IP access level? Difficult for public facing
  - Use API key/access token/header: eg. X-Gitlab-Token header

Push to Client

Created	@February 27, 2022 9:29 PM
Type	Lecture
Week	10
Lecture #	4
Lecture URL	https://youtu.be/Rn-gHqpDEll
Notion URL	https://21f1003586.notion.site/Push-to-Client-8539d1969a184184b29c47b9225c122d

Message targets

- Server to server push
  - Server A requests some messages to be sent
  - Server B calls back webhook after sending all messages

Client-side updates

- Requires persistent connection to/from client

- Pull vs Push
  - Client can pull updates
  - Server can push updates
- Original HTTP spec provided no support for this
  - Connection is stateless
  - Client-side pull easy — always possible with page refresh

Polling

- Client repeatedly sends requests
- Fixed interval
  - easy to implement at client
  - Server may not have any updates! Unnecessary work
  - Can overwhelm server if too many clients
- Variable interval: long poll
  - Server blocks until it has something to update
  - Keep connection open — data sent only when needed
  - Occupies server resources
- https://replit.com/@nchandra/longpoll#main.py
- https://replit.com/@nchandra/simplechat#main.py
- https://javascript.info/long-polling#demo-a-chat

Server-sent events

- Mechanism for server to “push” events
- Requires WebWorker (service worker) on client
- Service worker can continue running in the background
- Receive update, push to page

Needs more work on server side, but true push possible

Push notifications

- JS Push API

    - https://developer.mozilla.org/en-US/docs/Web/API/Push_API
- Web Push Protocol
    - https://datatracker.ietf.org/doc/html/draft-ietf-webpush-protocol
- Message urgency, priority possible
- Service worker on client receives and updates

Public push notification provides

- Alternatives
    - Firebase Cloud Messaging (previously Google Cloud Messaging)
    - Apple Push
- Authenticated and registered with app
- Web apps vs native apps
    - Web tech (HTTP) vs custom connections (TCP)

Summary: push messaging important part of user experience