

CS 253: Web Security

Local HTTP server security

The most dangerous code you run every day

```
// Anyone can connect to the server at http://<your-ip>:4000  
server.listen(4000)
```

```
// Only your device can connect to the server  
server.listen(4000, '127.0.0.1')
```

Zoom Zero Day: 4+ Million Webcams & maybe an RCE? Just get them to visit your website!

A vulnerability in the Mac Zoom Client allows any malicious website to enable your camera without your permission. The flaw potentially exposes up to 750,000 companies around the world that use Zoom to conduct day-to-day business.



Jonathan Leitschuh [Follow](#)
Jul 8 · 16 min read



CVE-Numbers

- DOS Vulnerability — Fixed in Client version 4.4.2 — [CVE-2019-13449](#)
- Information Disclosure (Webcam) — Unpatched — [CVE-2019-13450](#)

UPDATE — July 9th (am)

As far as I can tell this vulnerability also impacts Ringcentral. Ringcentral for their web conference system is a white labeled Zoom system.

Zoom zero day

"This vulnerability allows any website to forcibly join a user to a Zoom call, with their video camera activated, without the user's permission"

"On top of this, this vulnerability allowed any webpage to DOS (Denial of Service) a Mac by repeatedly joining a user to an invalid call"

"Additionally, if you've ever installed the Zoom client and then uninstalled it, you still have a localhost web server on your machine that will happily re-install the Zoom client for you, without requiring any user interaction on your behalf besides visiting a webpage. This re-install 'feature' continues to work to this day"

Zoom zero day

"Let me start off by saying having an installed app that is running a web server on my local machine with a totally undocumented API feels incredibly sketchy to me"

"Secondly, the fact that any website that I visit can interact with this web server running on my machine is a huge red flag for me as a Security Researcher"

"Having every Zoom user have a web server that accepts HTTP GET requests that trigger code outside of the browser sandbox is painting a huge target on the back of Zoom"

Demo: How does a site communicate with a local HTTP server?

Demo: How does a site communicate with a local HTTP server?

- With the following local HTTP server:

```
const COMMAND = 'open /System/Applications/Dictionary.app'

app.get('/', (req, res) => {
  exec(COMMAND, err => {
    res.set('Access-Control-Allow-Origin', '*')
    if (err) res.status(500).send(err)
    else res.status(200).send('Success')
  })
})
```

- Any site can send a GET request to `http://localhost:4000` to launch the Dictionary application

Demo: How many servers are running on your computer?

Demo: How many servers are running on your computer?

```
$ lsof -i -P | grep -i "listen"
```

rapportd	408	feross	4u	IPv4	0x97025599d3aa176b	0t0	TCP	*:57054 (LISTEN)
rapportd	408	feross	5u	IPv6	0x97025599ed02e613	0t0	TCP	*:57054 (LISTEN)
CommCente	421	feross	26u	IPv6	0x97025599ed02b513	0t0	TCP	[2907:fa90:5c0:906e:a1a0:f0b3:9732:fa7a]:5060 (LISTEN)
Spotify	27013	feross	62u	IPv4	0x9702559a0aa233db	0t0	TCP	*:57343 (LISTEN)
Spotify	27013	feross	64u	IPv4	0x97025599ee8ae3db	0t0	TCP	*:57621 (LISTEN)

Problems with Zoom's local server

- Any site, not just `zoom.us`, can send a GET request to open the app and join the user to the given conference
 - `http://localhost:19421/launch?action=join&confno=###`
- Conference host can decide to automatically enable video for participants
- The local server remains installed after the user uninstalls Zoom and it has the ability to re-install Zoom
- Vulnerable to UI denial-of-service

Schedule a Meeting

Topic

Jonathan Leitschuh's Zoom Meeting

Date

7/ 6/ 2019 ▾

11:00 PM ▾

to

7/ 6/ 2019 ▾

11:30 PM ▾

Time Zone

(GMT-04:00) Eastern Time (US and Canada) ▾

Recurring meeting

Video

Host

On Off

Participants

On Off

Audio

Telephone Computer Audio Telephone and Computer Audio

Dial in from United States [Edit](#)

Options

Require meeting password

[Advanced Options](#) ▾

Calendar

iCal

Google Calendar

Outlook

Other Calendars

Cancel

Schedule

Zoom UI denial-of-service

```
// It's actually better if this number isn't a valid zoom conference number
const confNum = '694138052'

setInterval(() => {
  new Image().src =
    'http://localhost:19421/launch?action=join&confno=' +
    confNum + '&' + Date.now()
}, 1)
```

MUST READ: Google's new AI tool could help decode the mysterious algorithms that decide everything

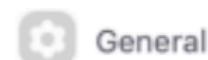
Zoom defends use of local web server on Macs after security report

Local web server will also reportedly reinstall Zoom if a user removes the application and joins a meeting.

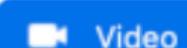


By [Chris Duckett](#) | July 9, 2019 --
01:28 GMT (18:28 PDT) | Topic:
[Security](#)

Settings



General



Video



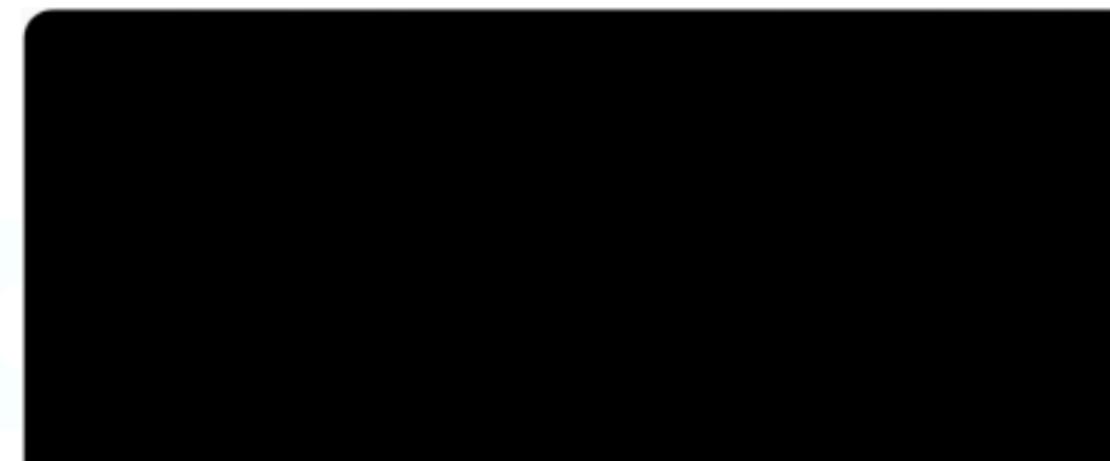
Audio



Chat



Virtual Background



RECOMMENDED FOR YOU

Tech Pro Research: Using Tech to Make Shopping Easier and More Enjoyable

Downloads provided by [TechRepublic.com](#)

DOWNLOAD NOW

adjeh

MORE FROM CHRIS DUCKETT

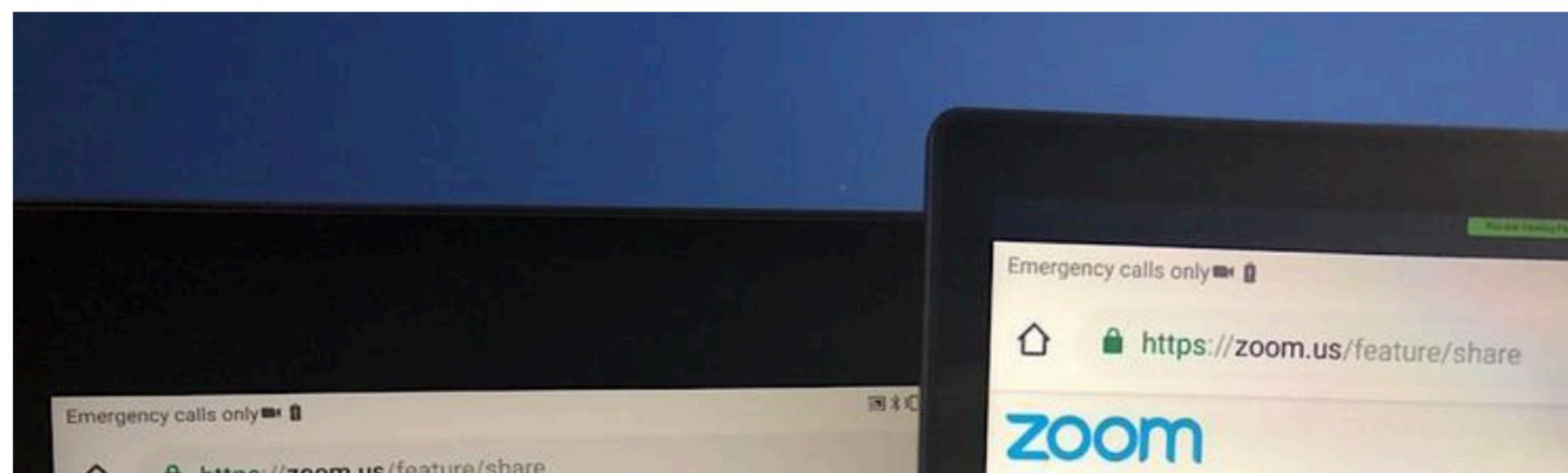
MUST READ: Google's new AI tool could help decode the mysterious algorithms that decide everything

Zoom reverses course to kill off Mac local web server

Less than a day after backing its approach to get around Safari restrictions on Mac, Zoom's local web server is no more.



By [Chris Duckett](#) | July 10, 2019 --
00:50 GMT (17:50 PDT) | Topic:
[Security](#)



RECOMMENDED FOR YOU

TechRepublic Premium
Budget Template: Year-round IT budgets

Downloads provided by TechRepublic Premium

[DOWNLOAD NOW](#)

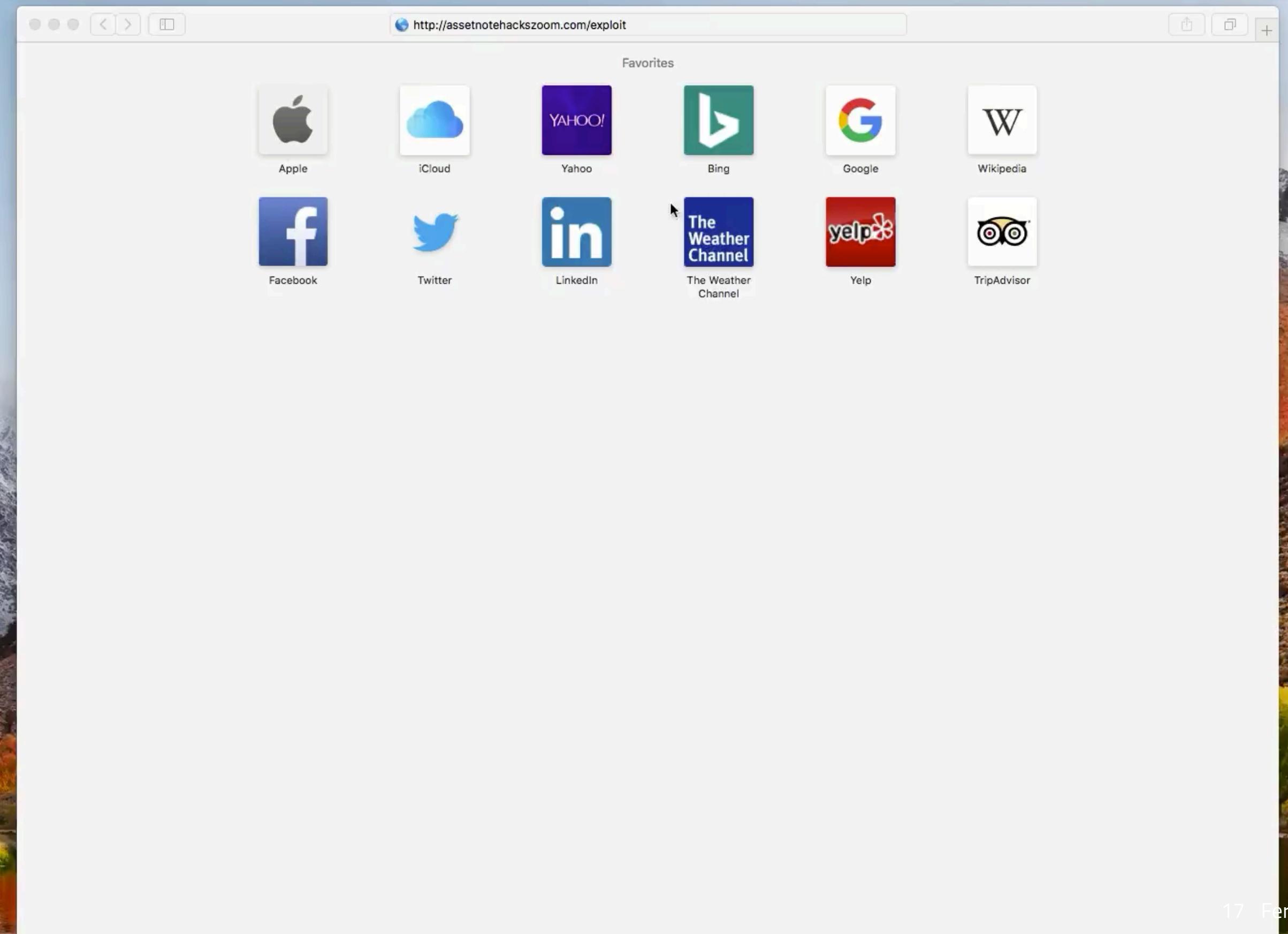
hadjeh

Cleaning up the mess

- Zoom issued an updated app which uninstalled the local HTTP server and added a new UI prompt to confirm that you want to join a meeting
- User who did not open the app for a while would be vulnerable until they installed the update
- Users who previously uninstalled Zoom would not get the update, so they'd be stuck with the vulnerable local server

Remote Code Execution (RCE)

- Around 1 week after the local server issue came to light, another research team discovered a RCE vulnerability
- The complete exploit allowed a zero-interaction RCE just by visiting a malicious site – yikes!



Apple is silently removing Zoom's web server software from Macs

48 

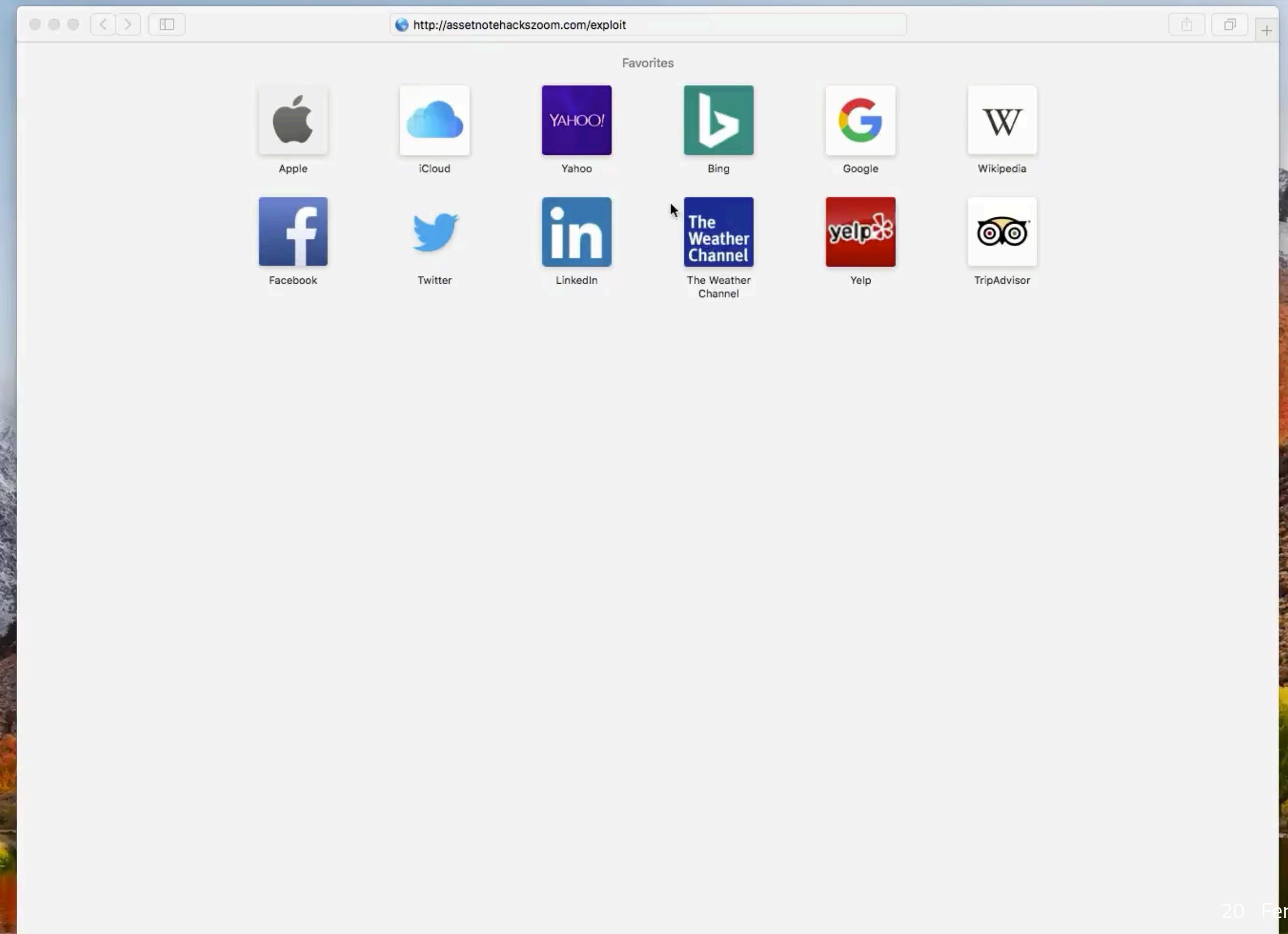
For users who haven't seen all the drama

By Dieter Bohn | @backlon | Jul 10, 2019, 7:12pm EDT



Apple takes steps

- macOS has a silent update mechanism for disabling specific executable files (Malware Removal Tool)
- No OS update required – checks for new banned executables in the background, regularly
- Useful for disabling fast-spreading malware or vulnerable software affecting lots of users



Zoom doesn't understand CORS

User joins a zoom call (vulnerable)

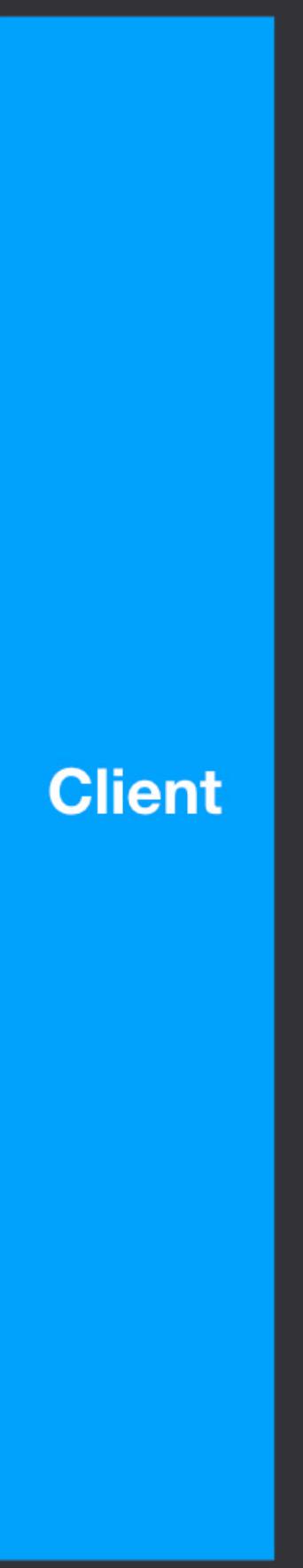
Local Server
localhost:19421

Client

Server
zoom.us



Local Server
`localhost:19421`

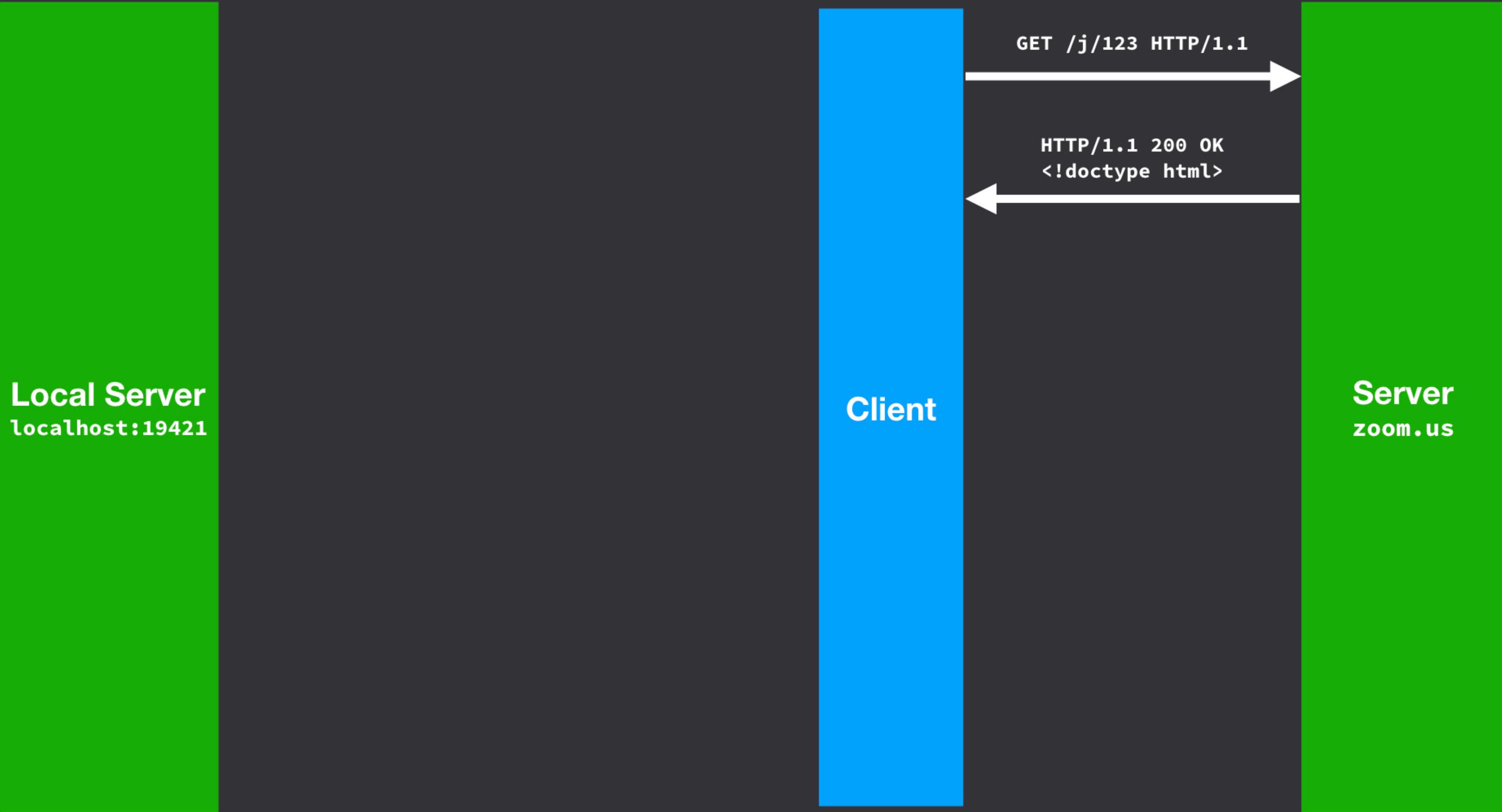


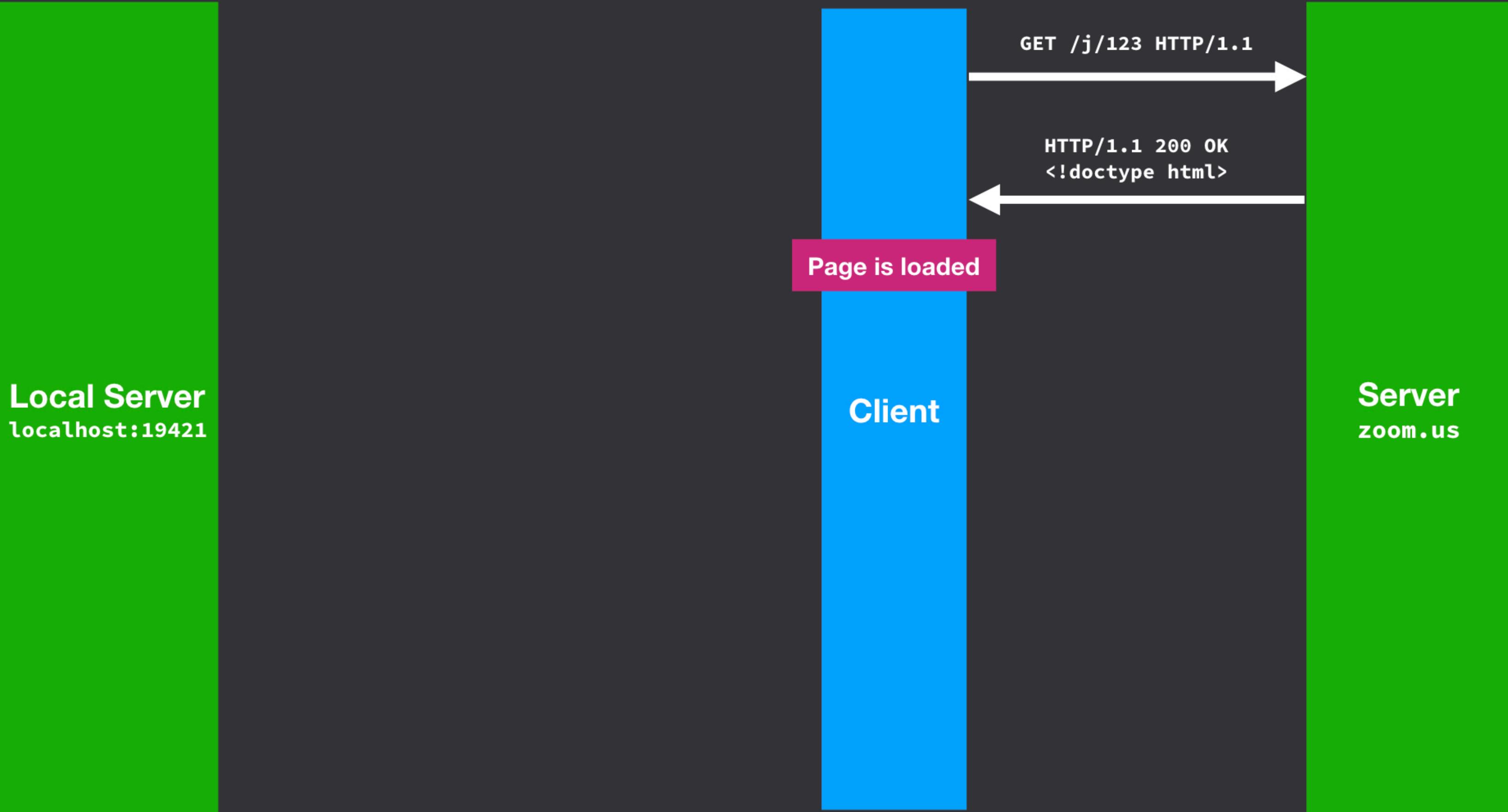
Client

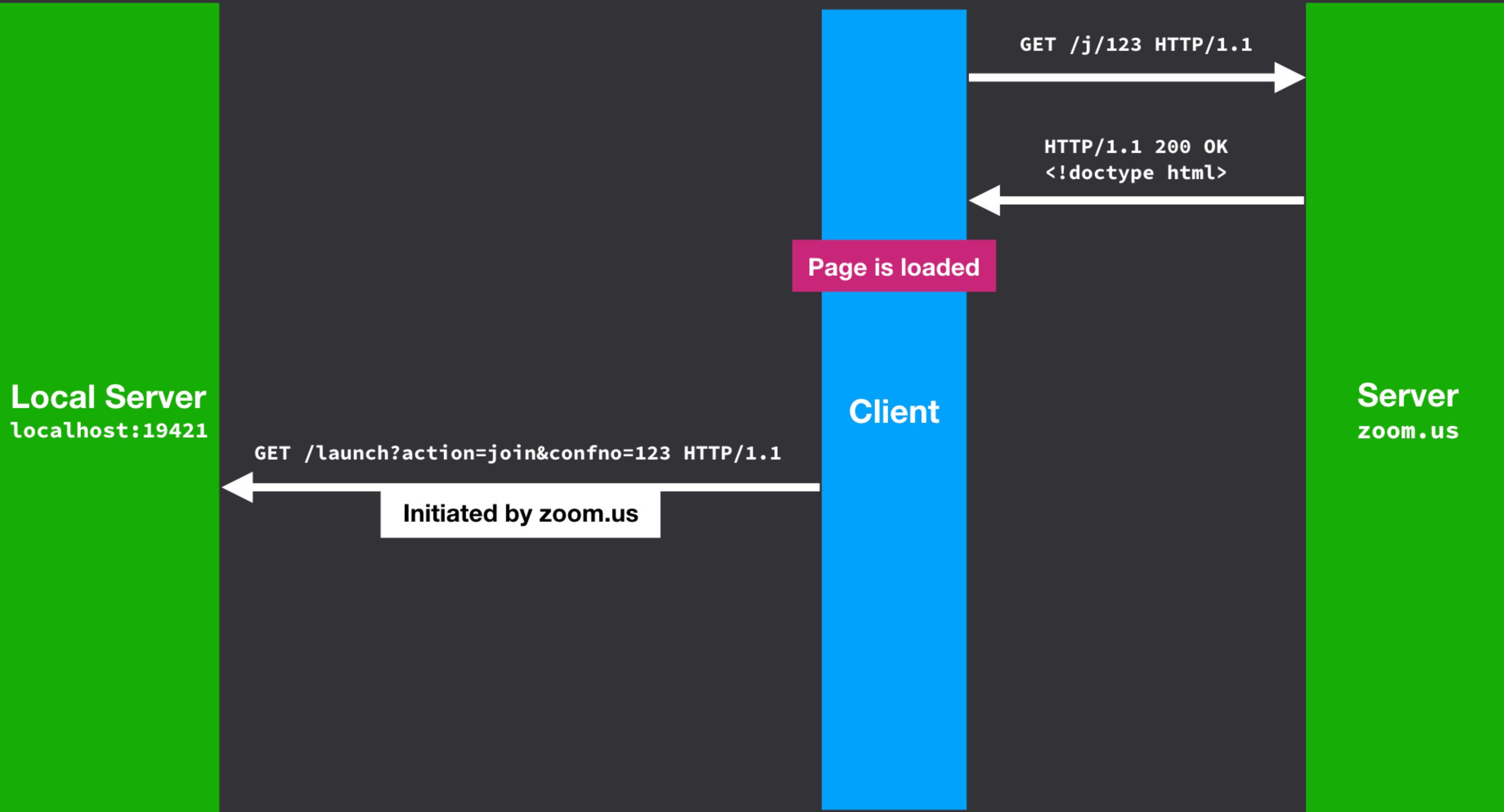


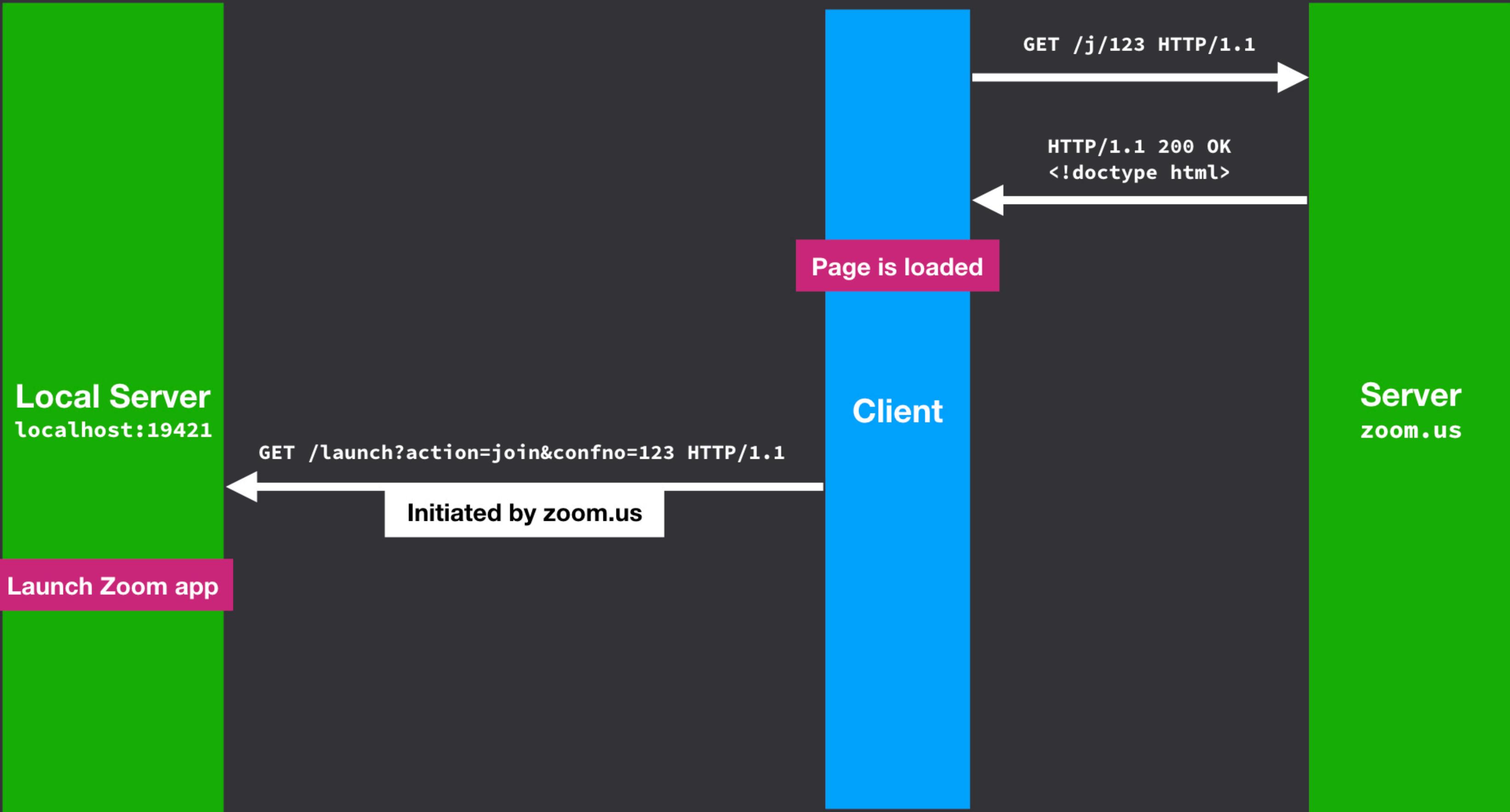
Server
`zoom.us`

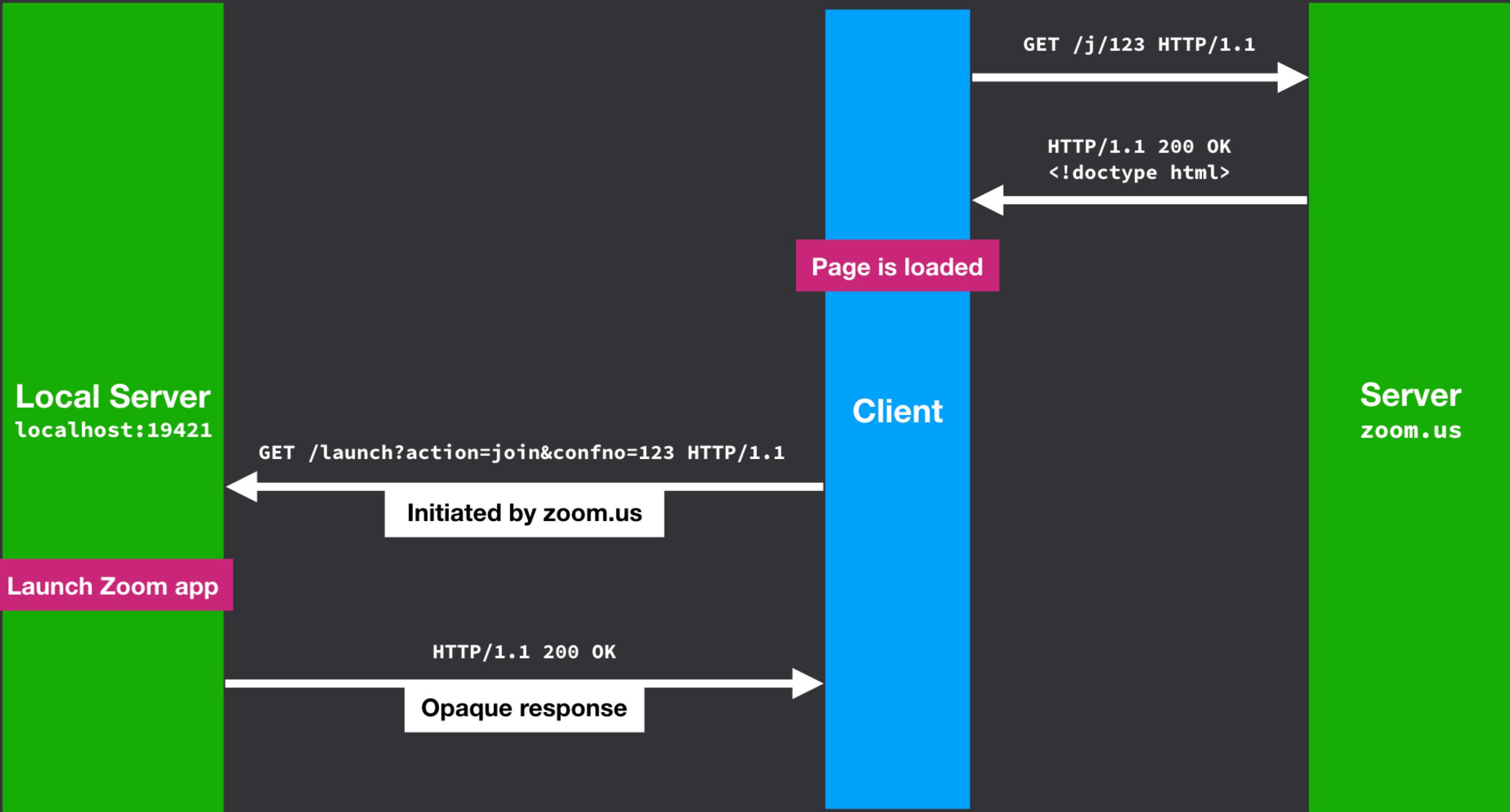


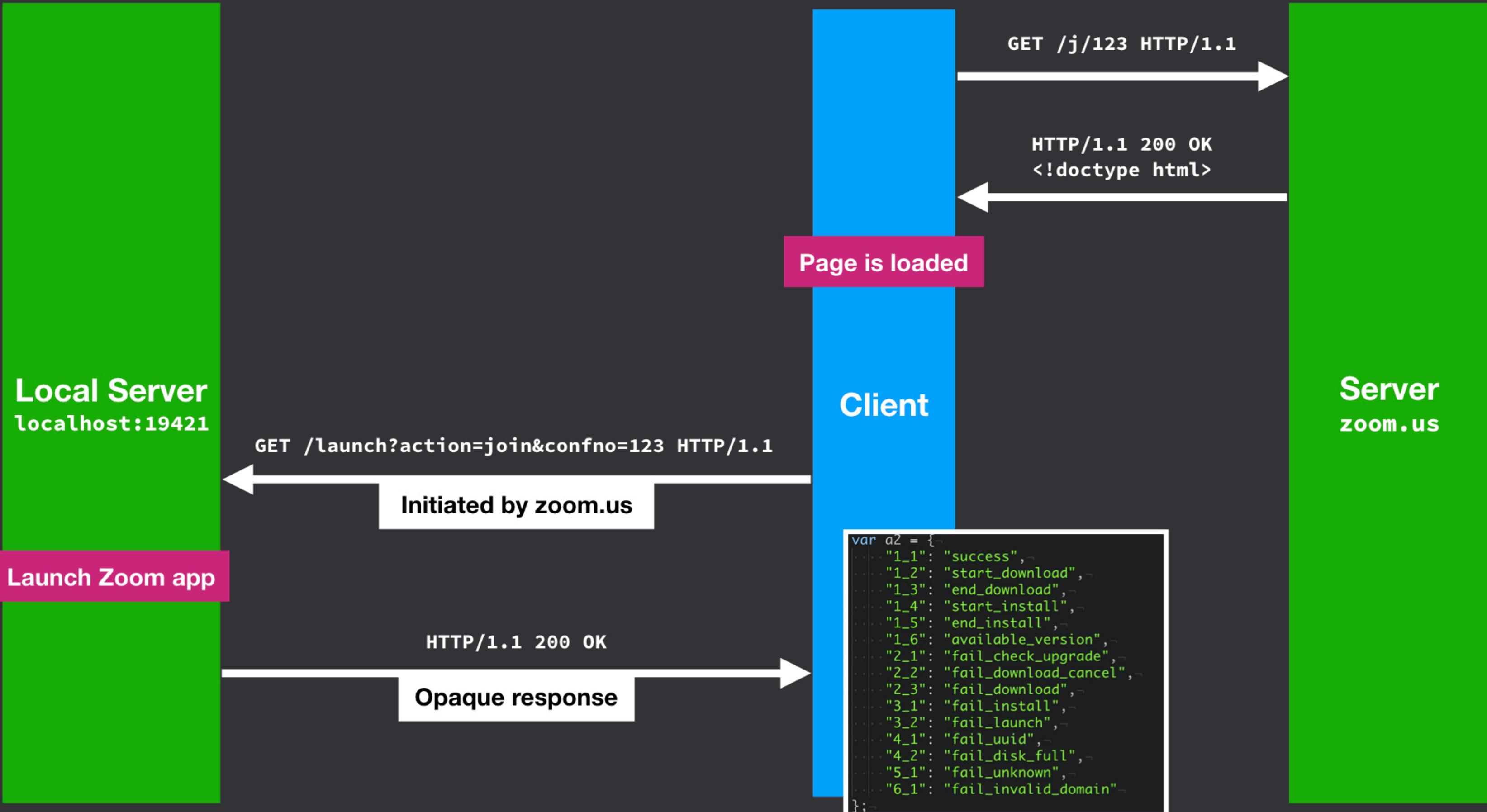












Zoom doesn't understand how CORS works?

- The `http://localhost:19421/launch?action=join&confno=###` endpoint returns information about whether the request succeeded, but since it's triggered from `https://zoom.us` the same origin policy doesn't allow **reading** the response
- So, they returned an image with different widths/heights to "leak" information to the site that triggered the request
- They could have just used **Access-Control-Allow-Origin** to specify particular sites which would be allowed to read the response

User joins a zoom call (with CORS endpoint) (vulnerable)

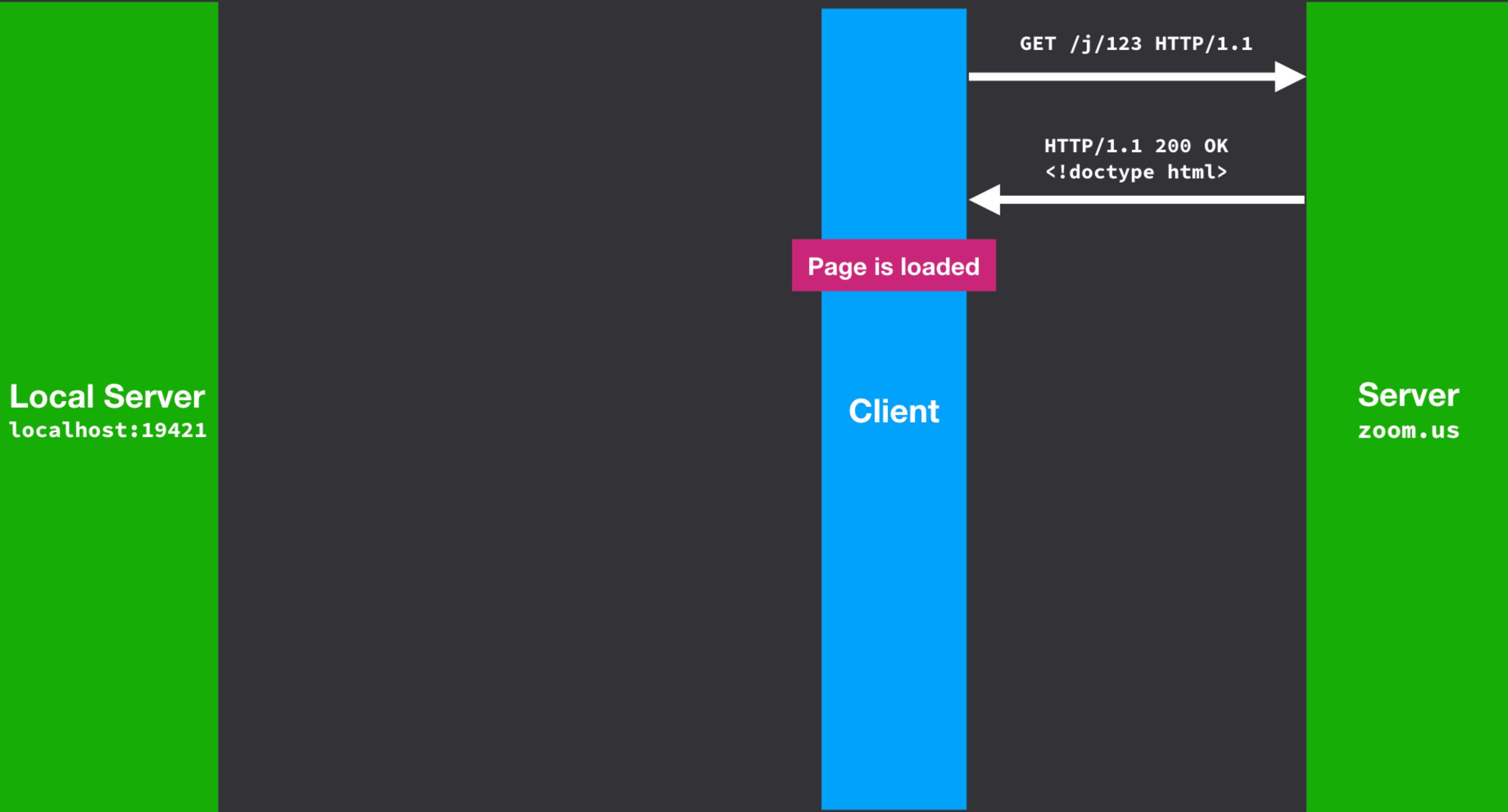
Local Server
localhost:19421

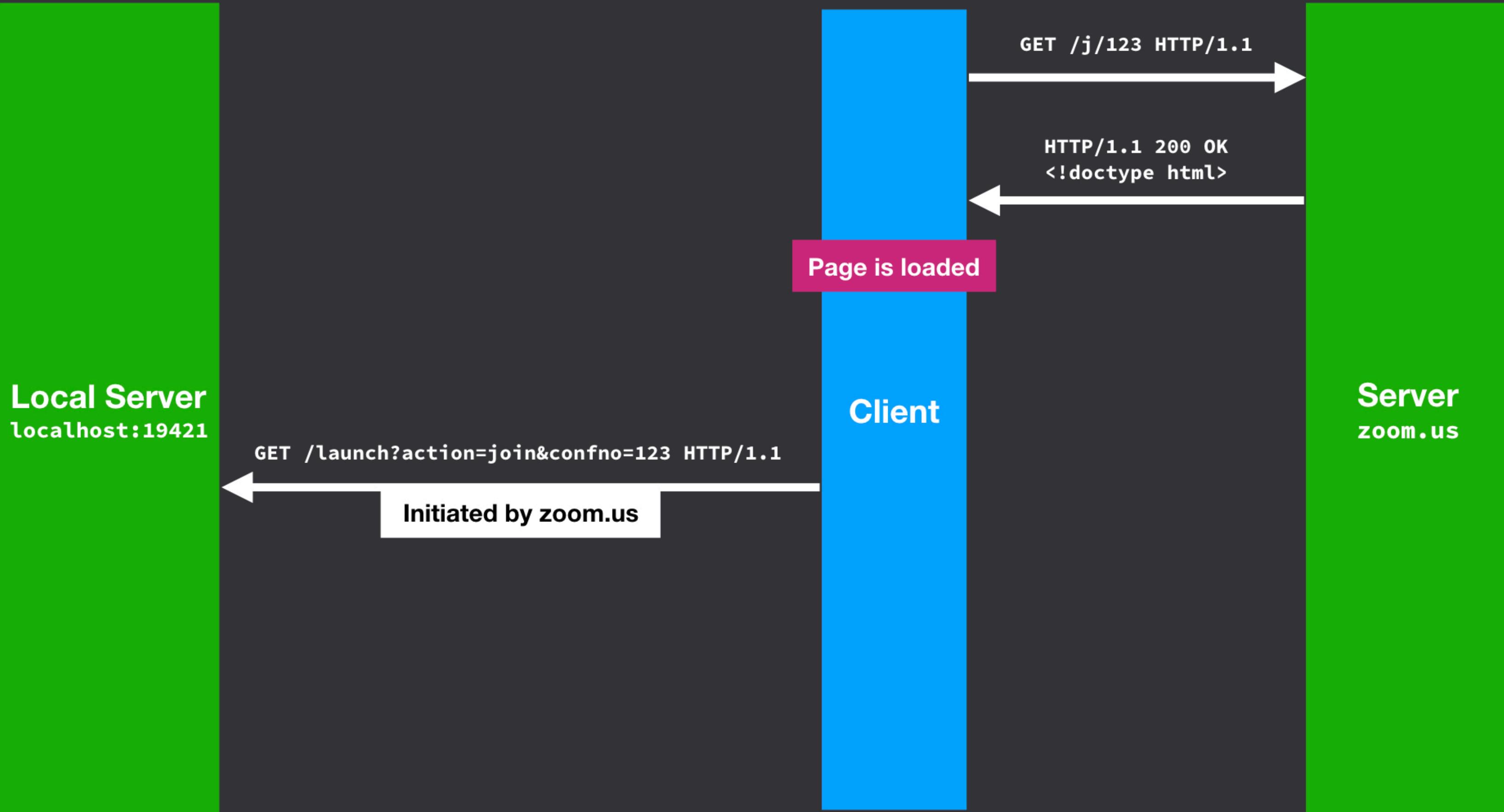
Page is loaded

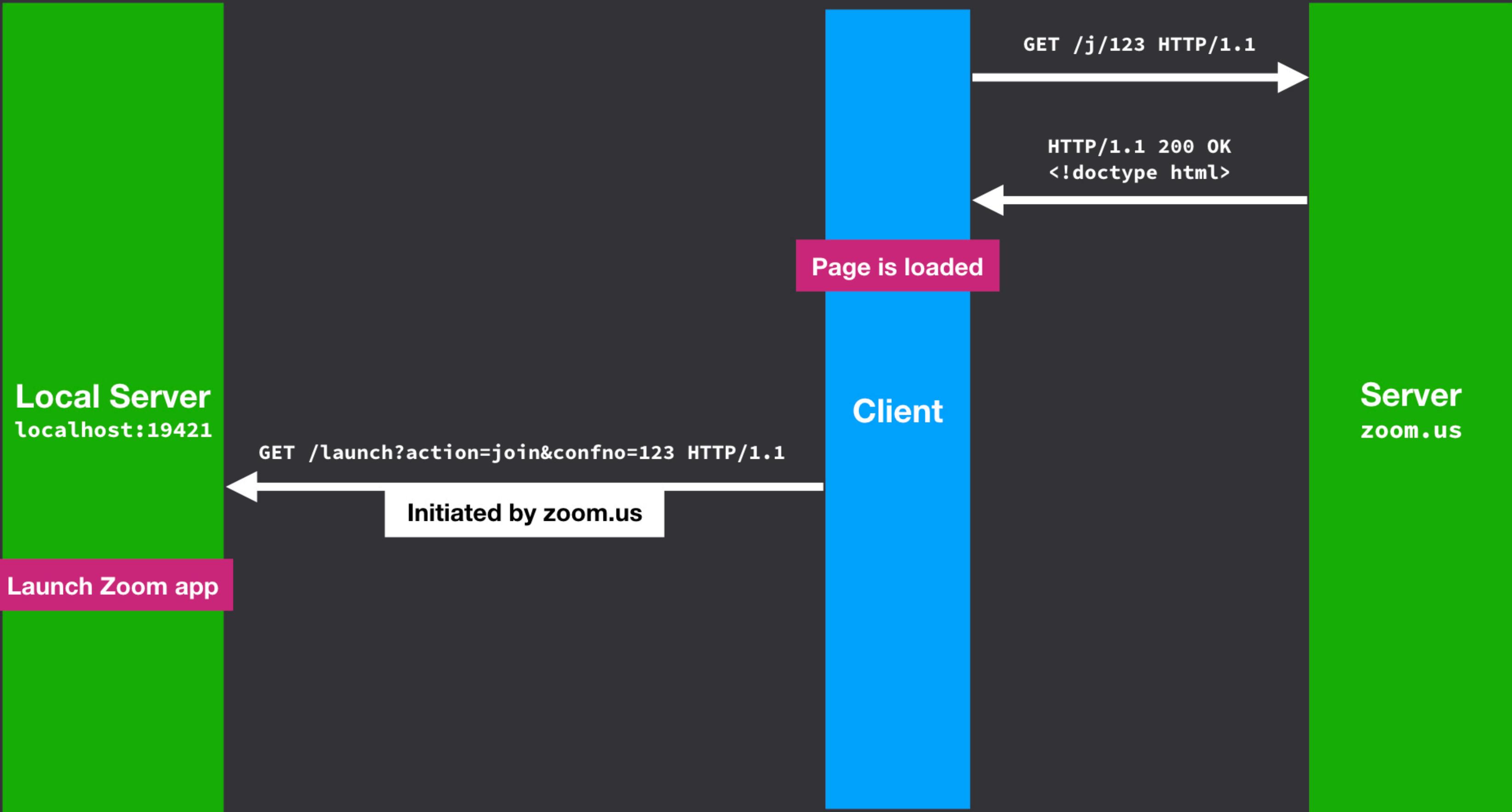
HTTP/1.1 200 OK
<!doctype html>

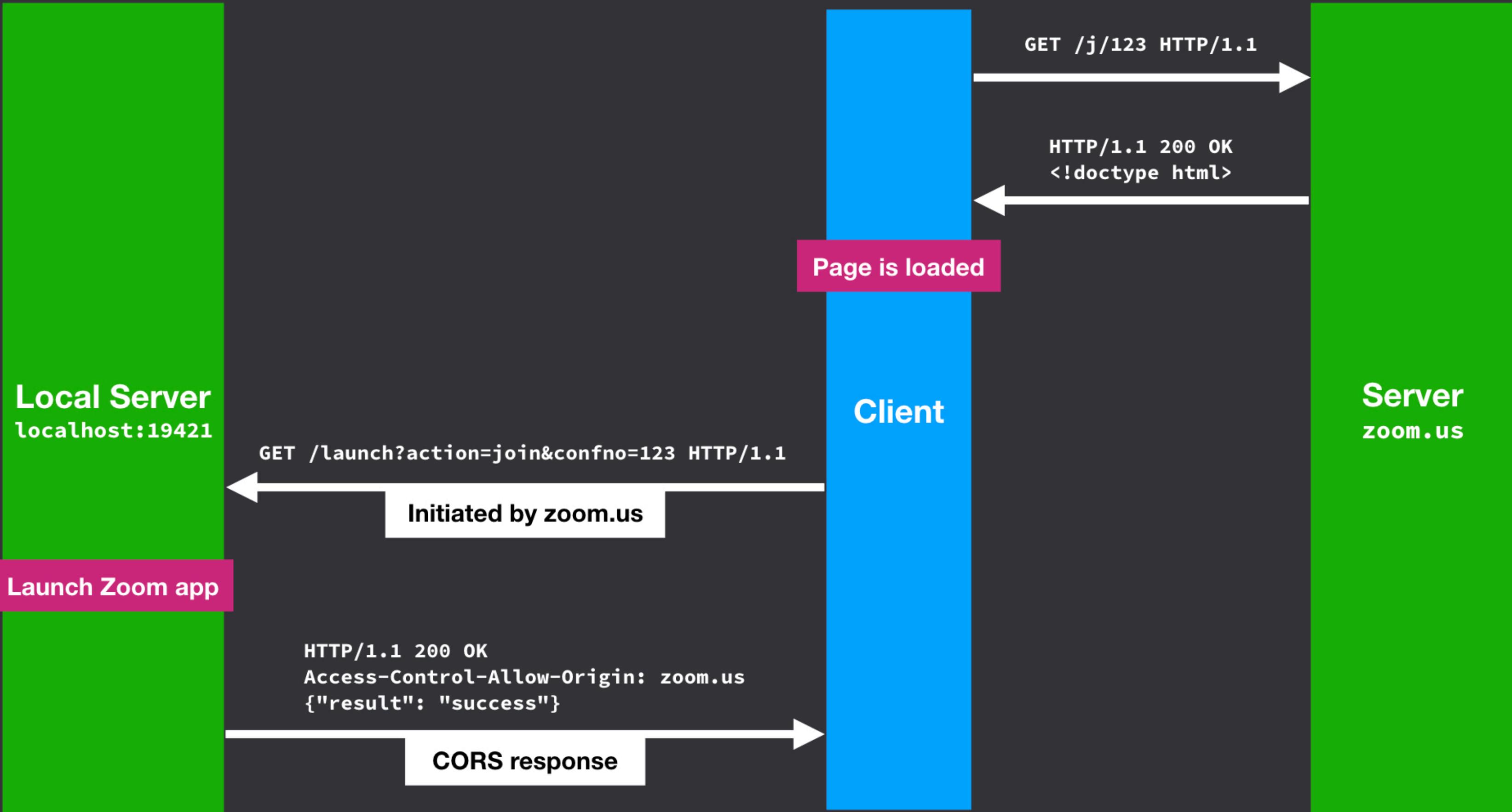
Client

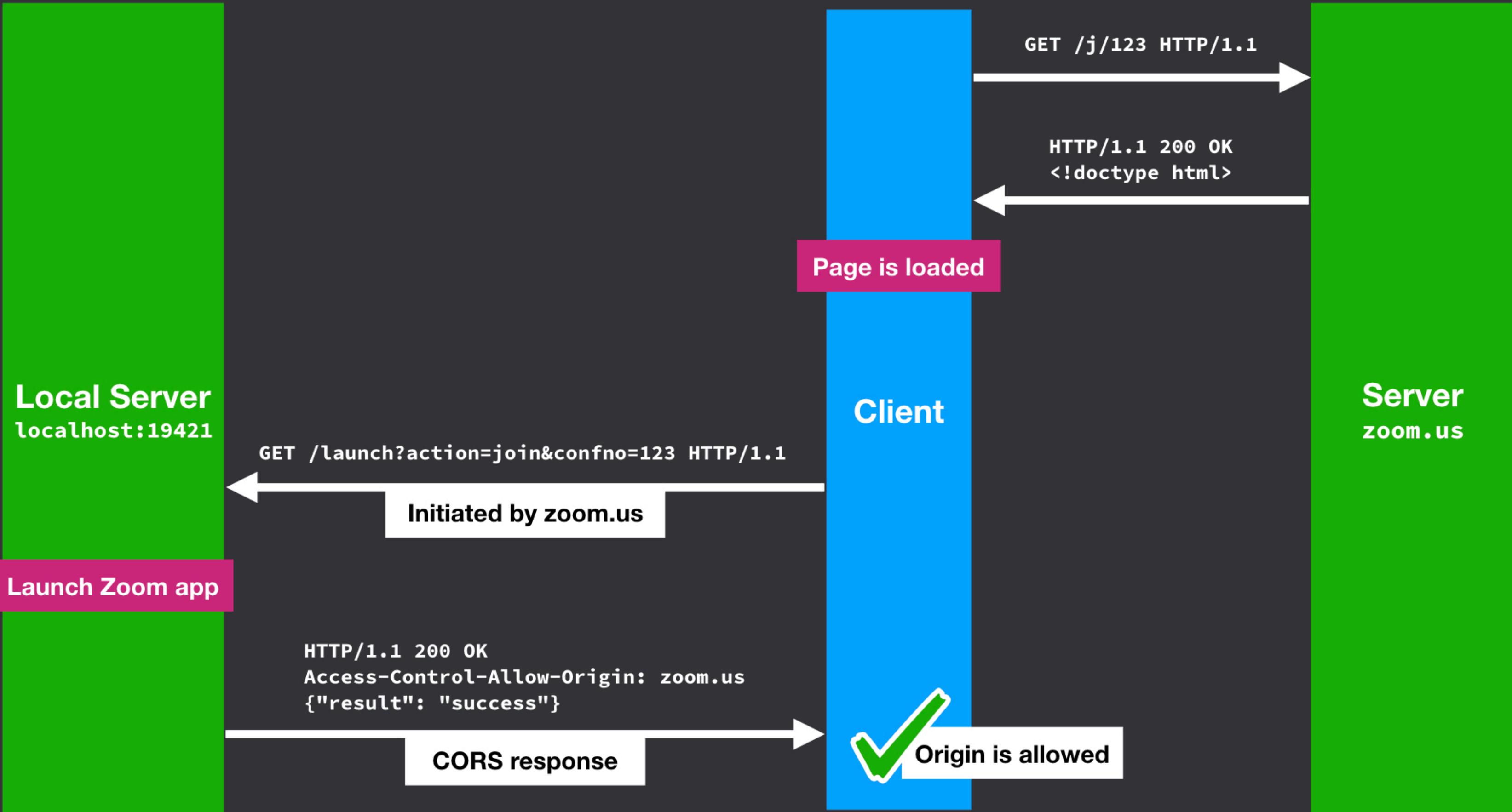
Server
zoom.us











Attacker joins user into a zoom call

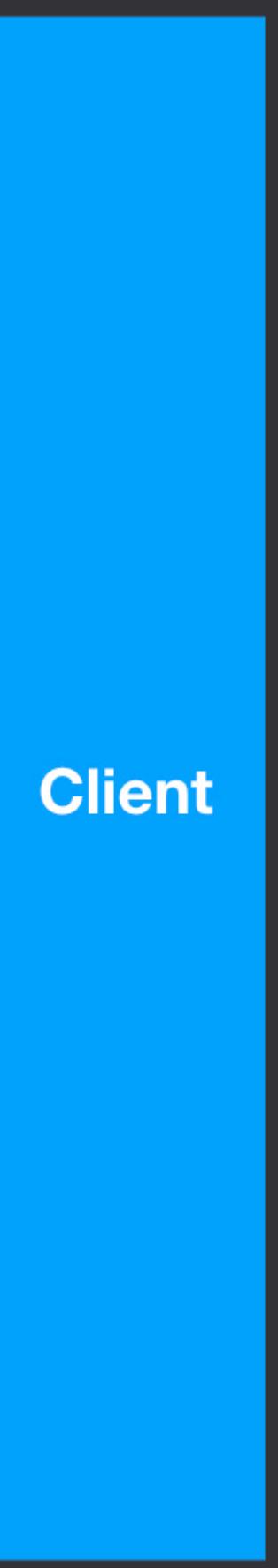
Local Server
localhost:19421

Client

Server
attacker.com



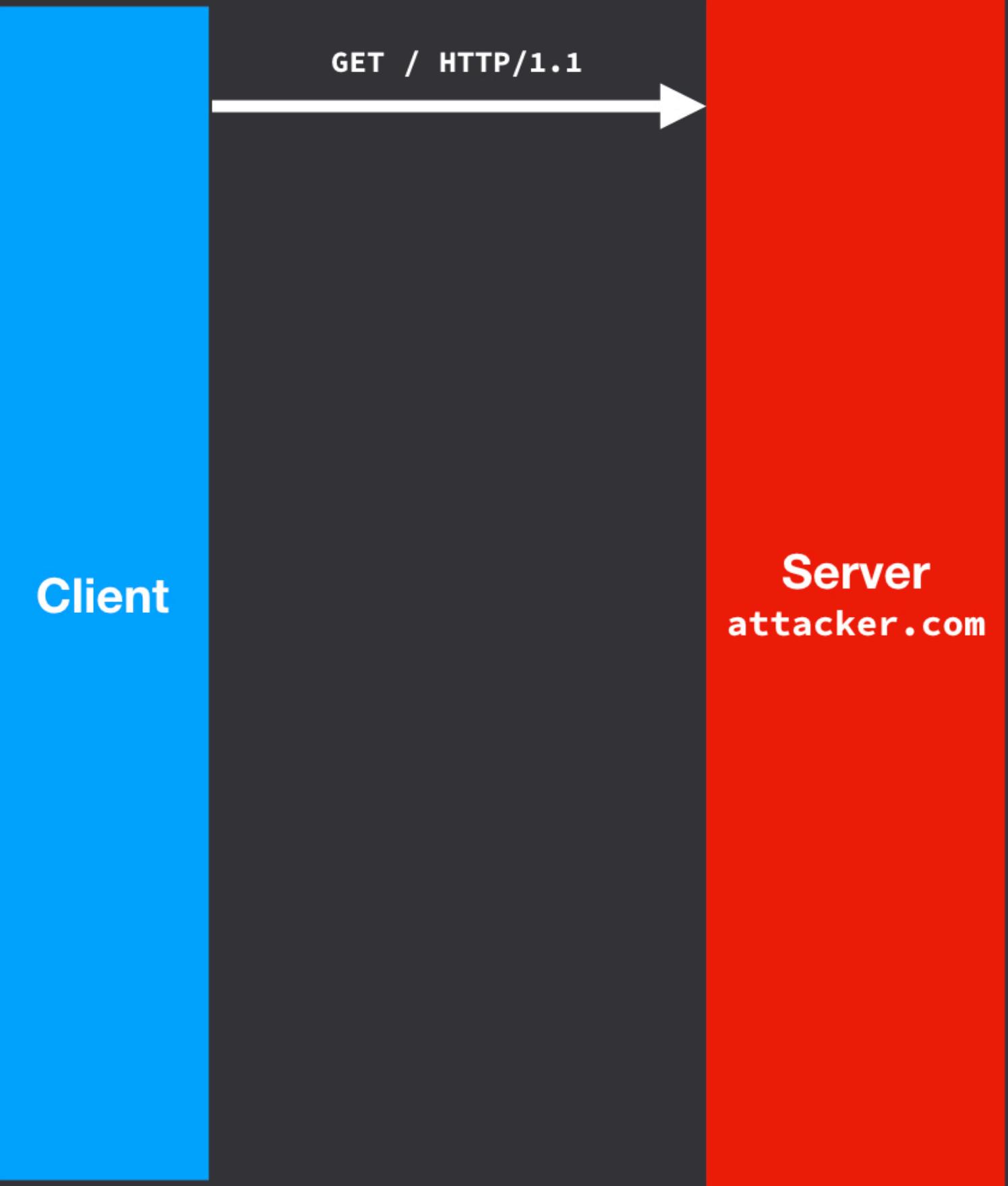
Local Server
`localhost:19421`



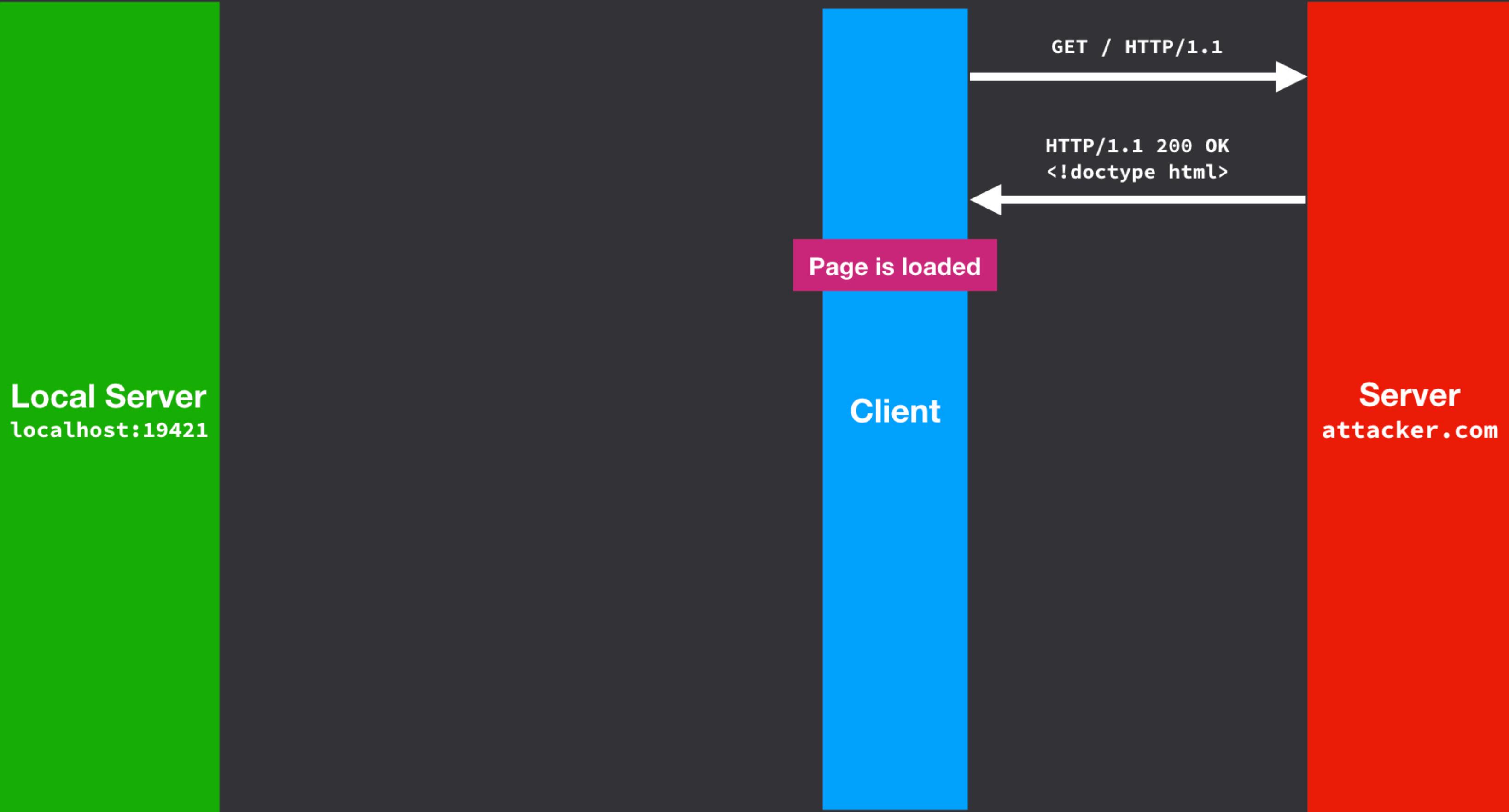
Client

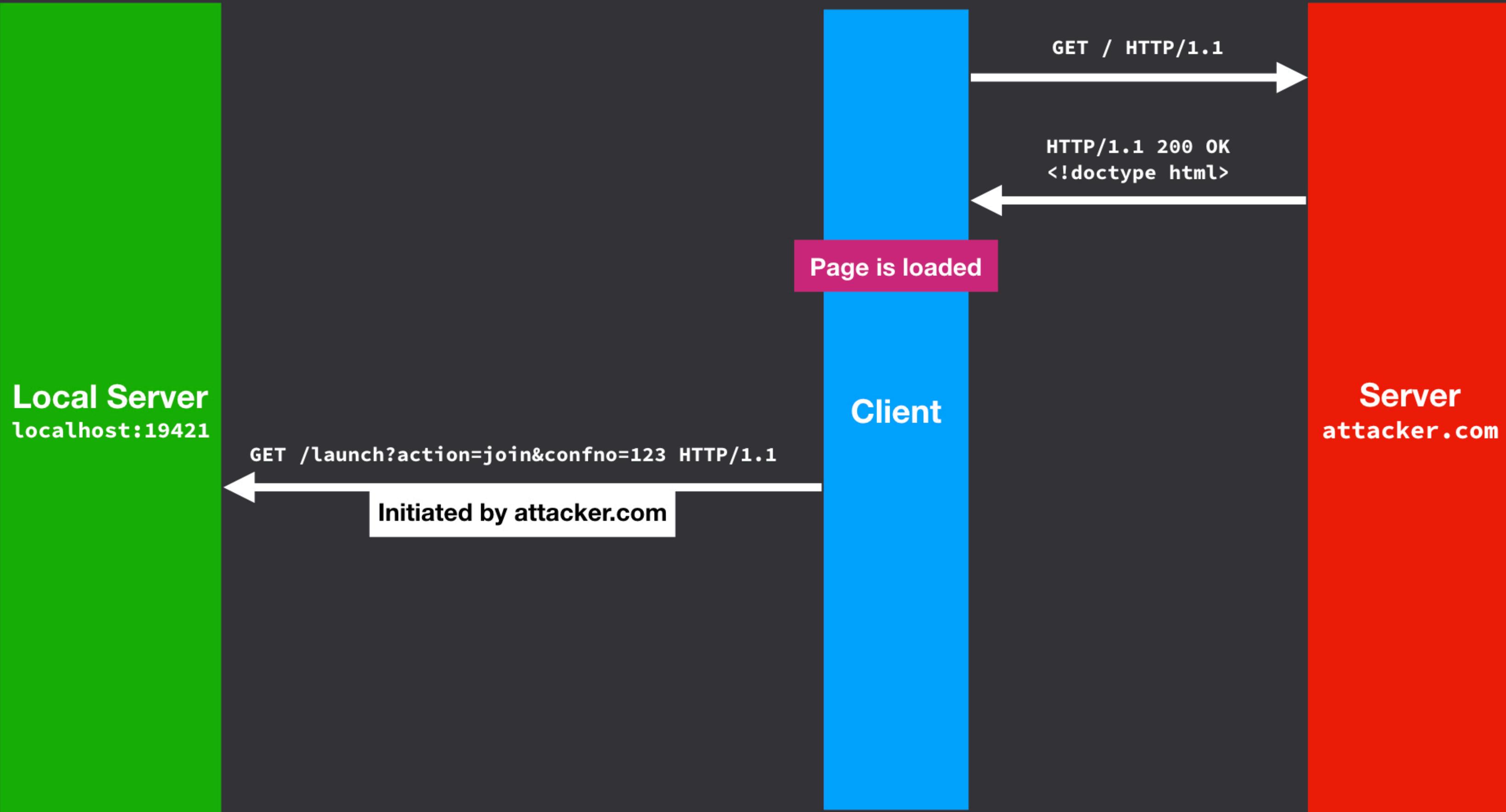


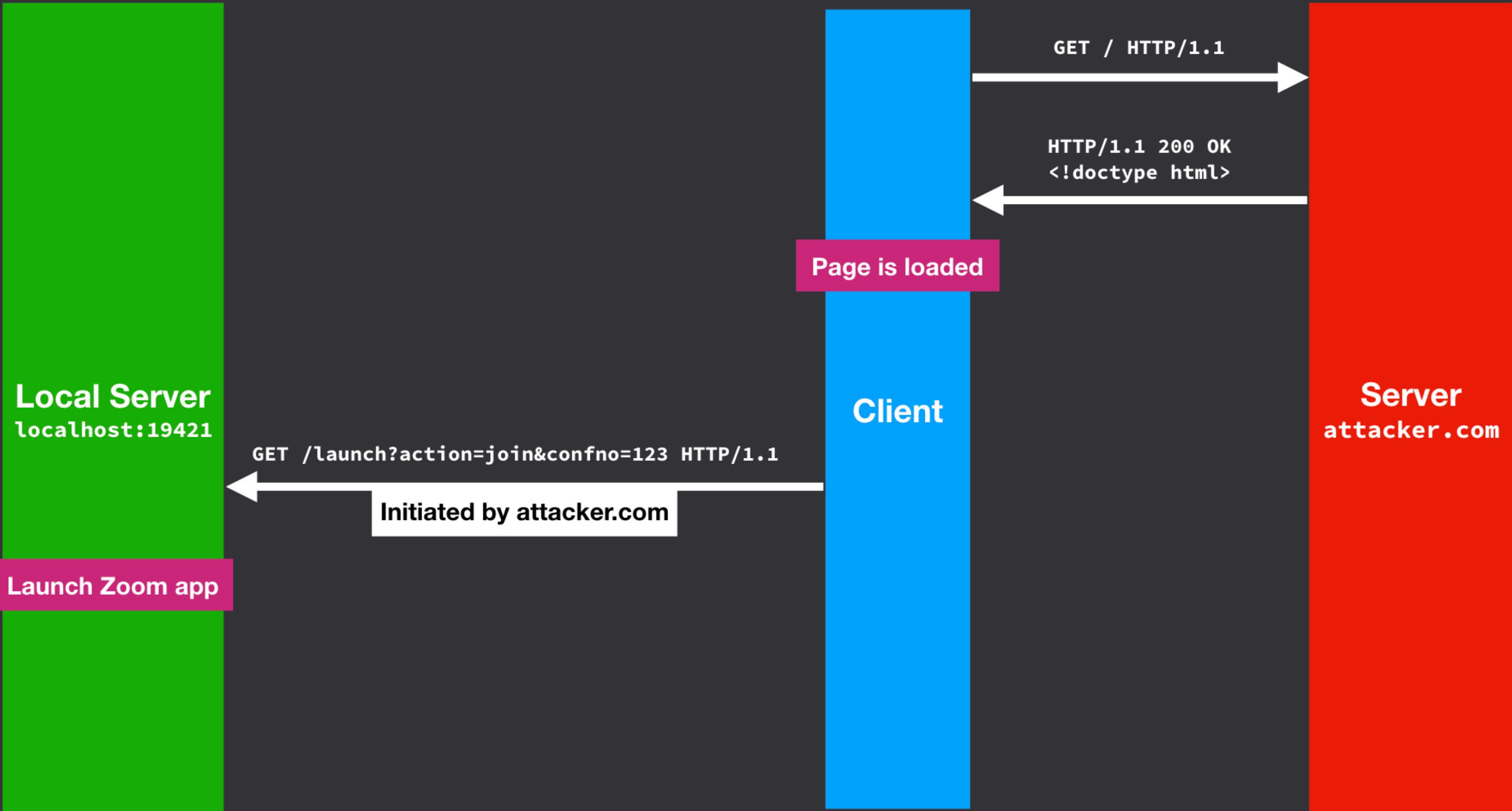
Server
`attacker.com`

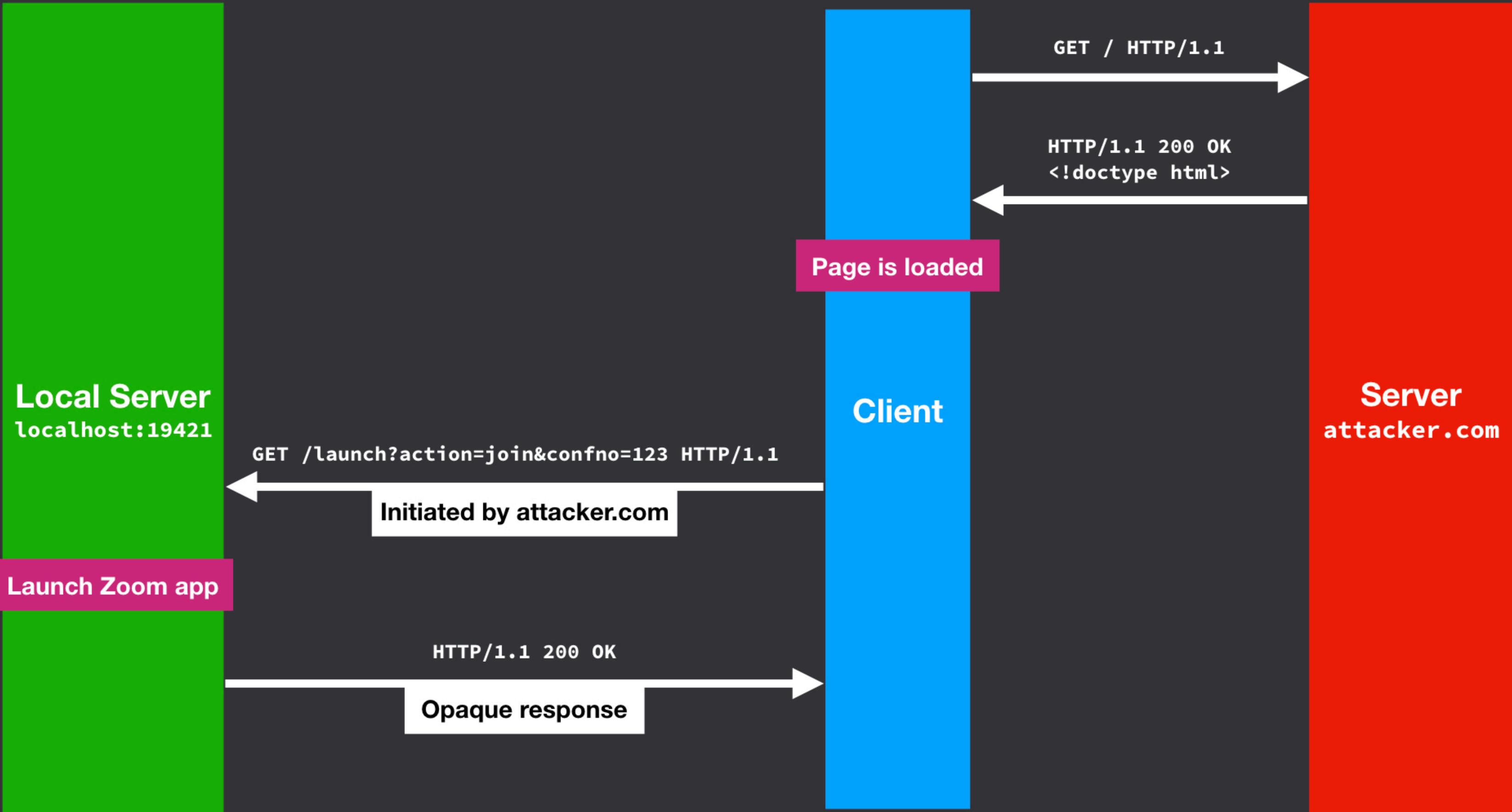












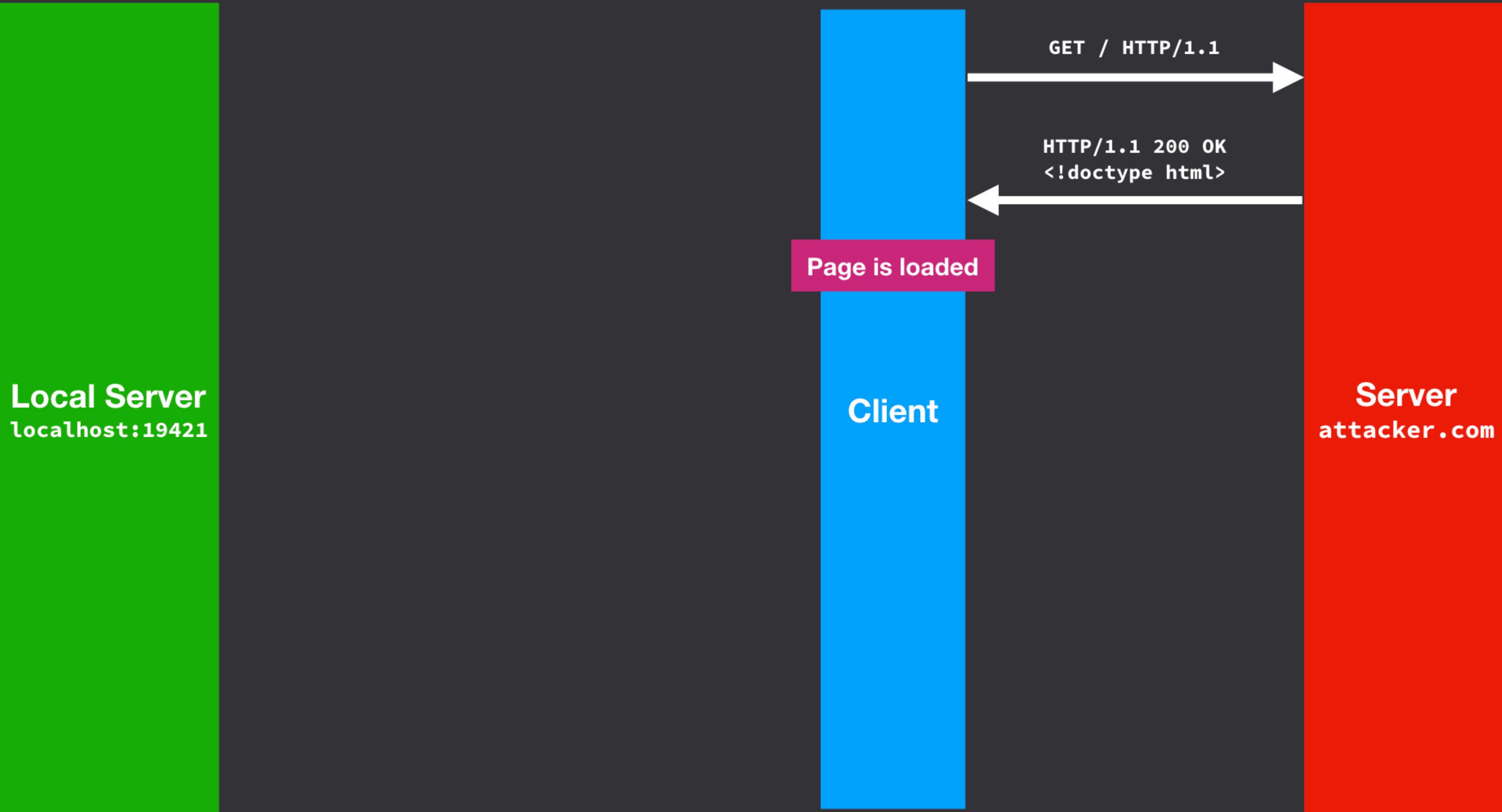
Attacker joins user into a zoom call (with CORS endpoint)

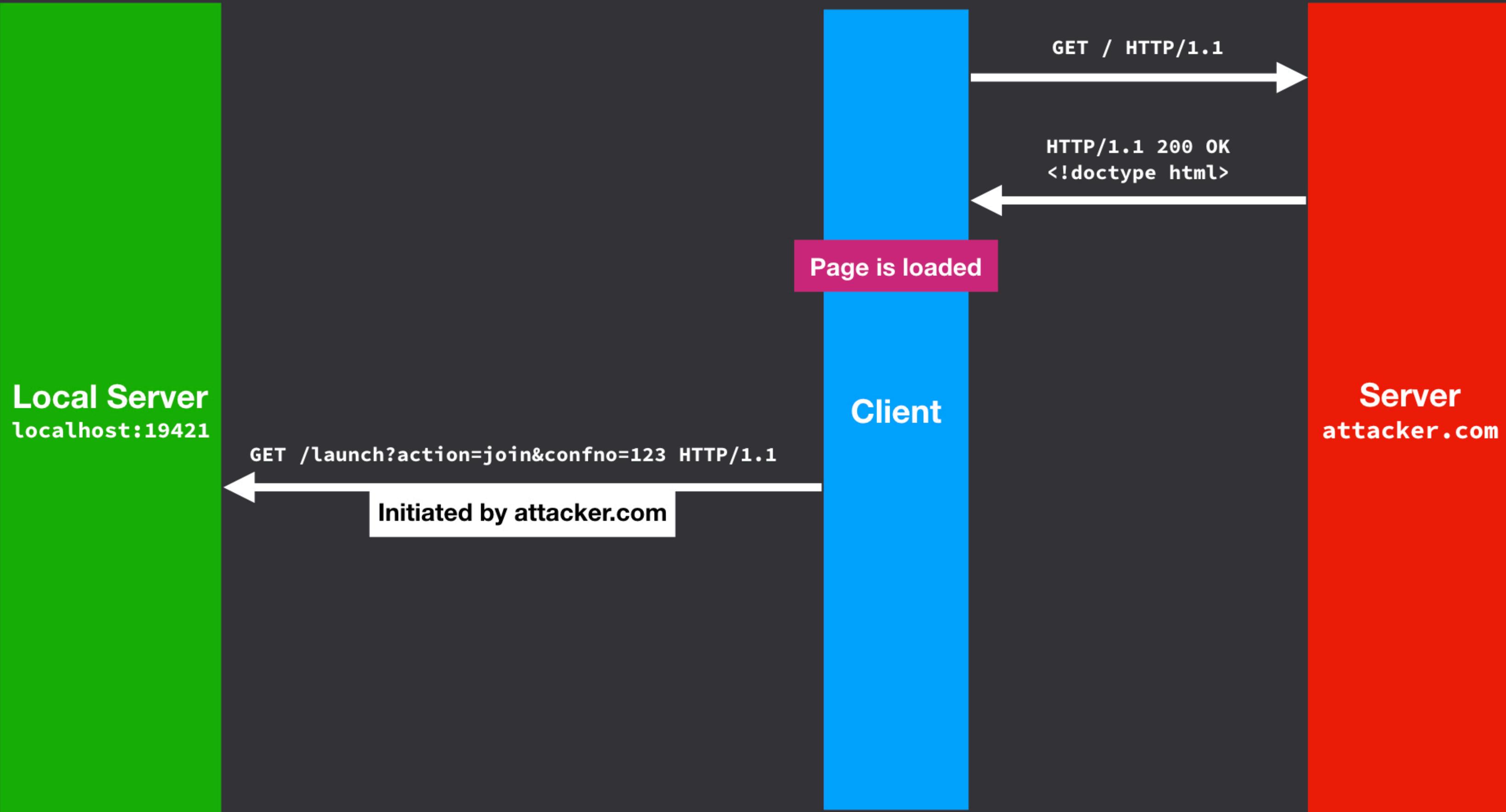
Local Server
localhost:19421

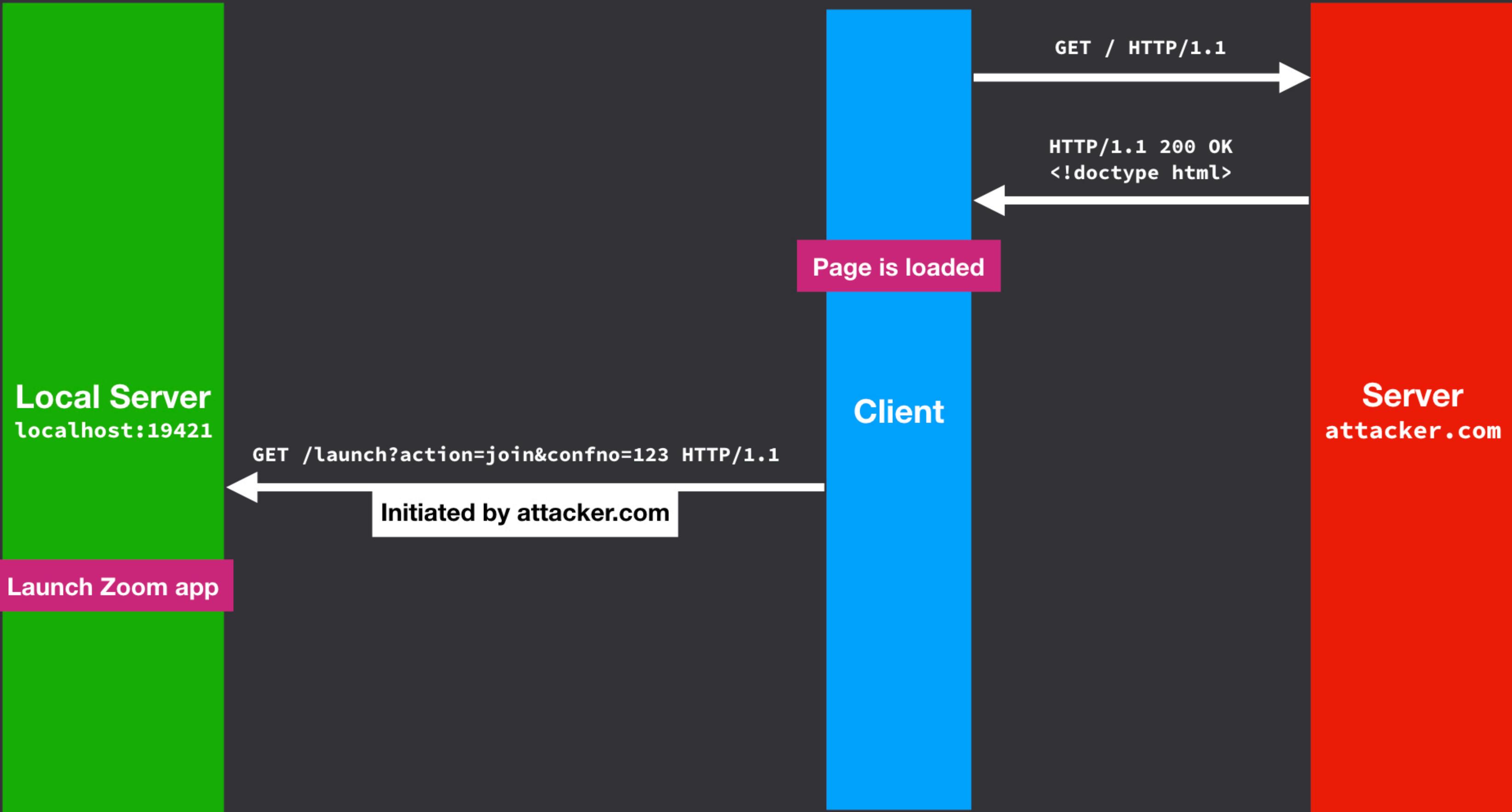
Page is loaded

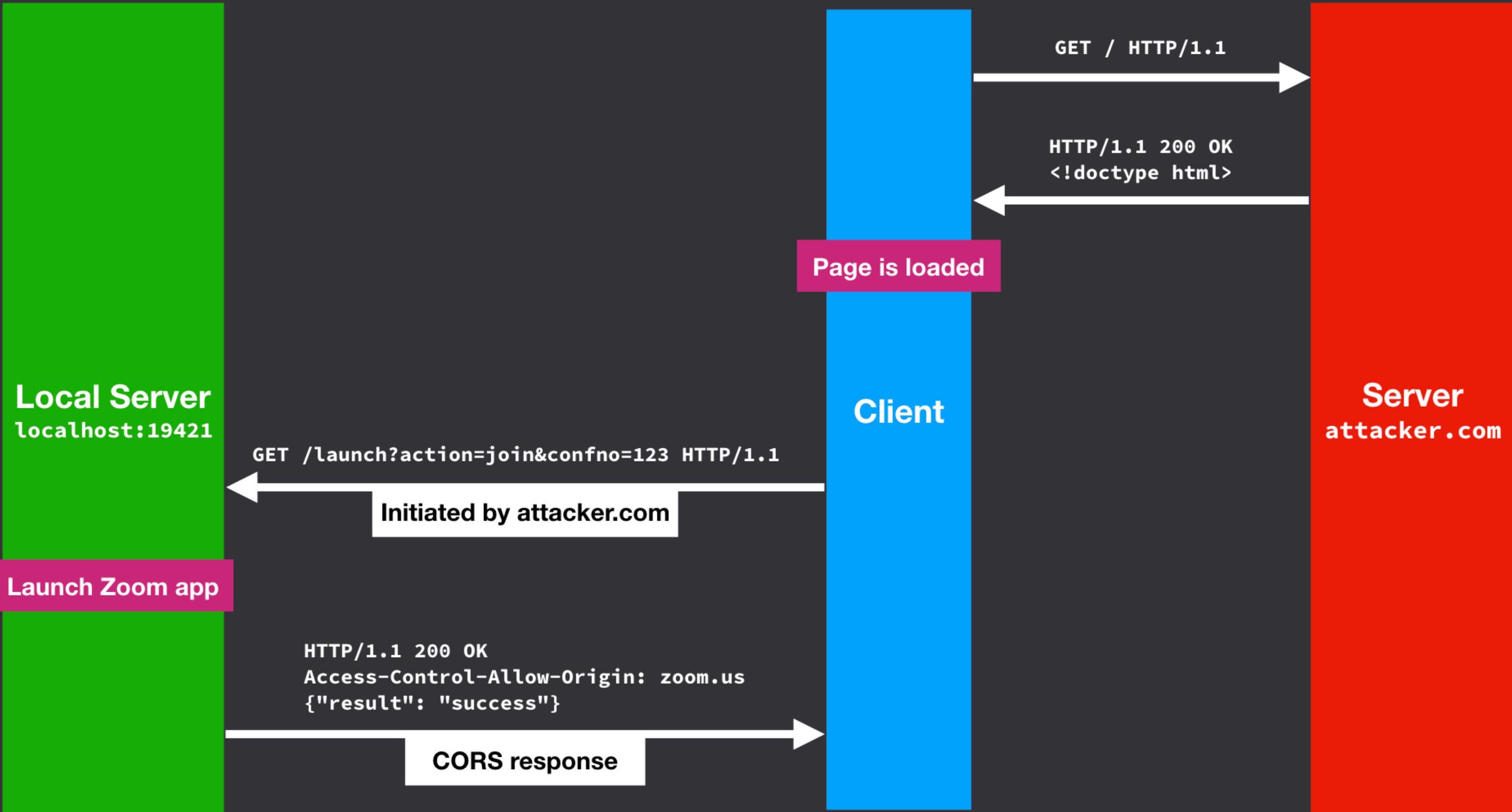
Server
attacker.com

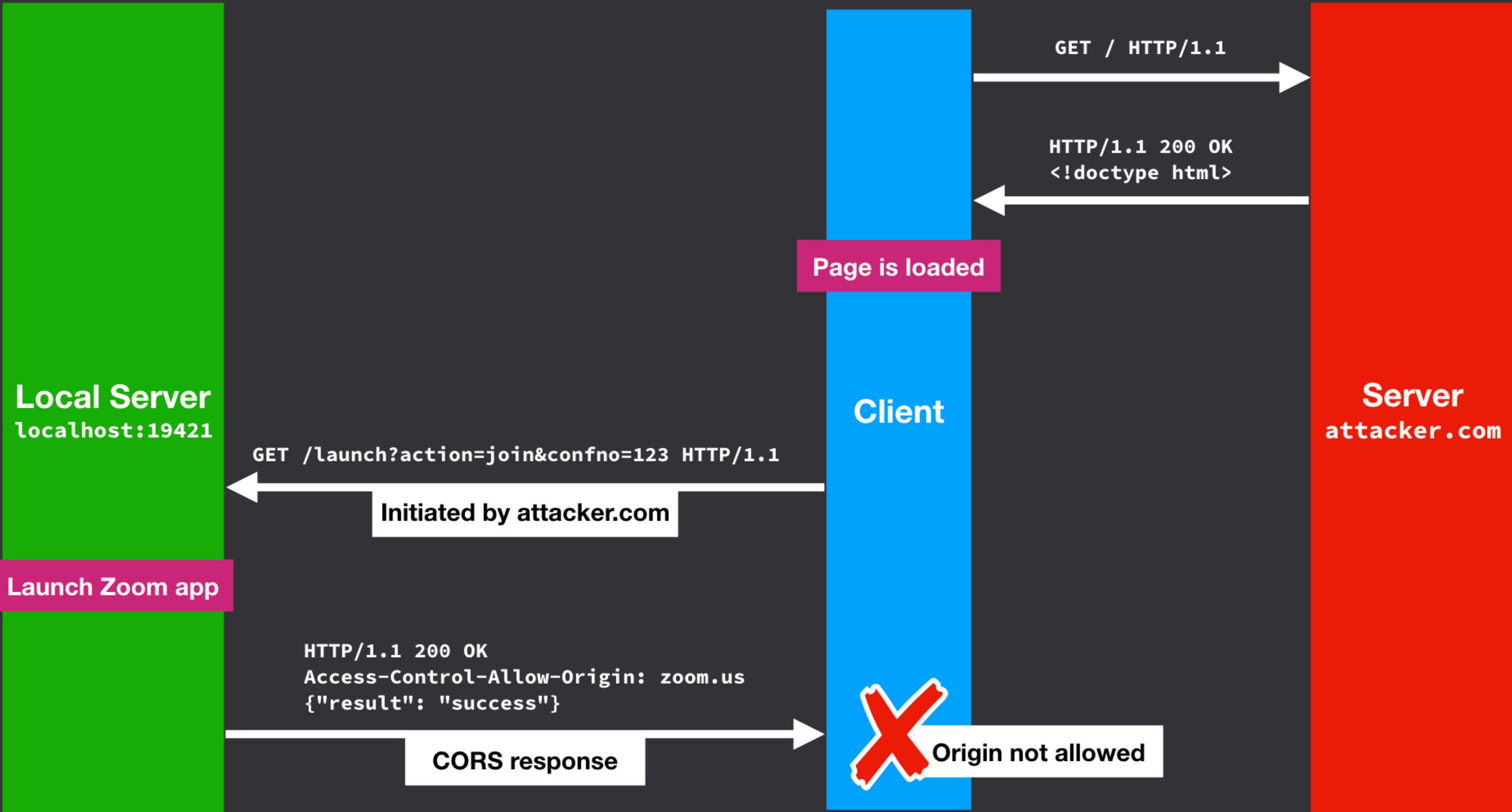
HTTP/1.1 200 OK
<!doctype html>











Let's fix the issue

- **Best solution:** remove the local HTTP server and just register a `zoom://` protocol handler
- However, let's assume we need to keep the local HTTP server (probably a bad idea)
 - How can we secure it?
- Ideas:
 - Require user interaction before joining, don't allow host to automatically enable video
 - Only allow `zoom.us` to communicate with the local server

User joins a zoom call (local server inspects Origin header)

Local Server
localhost:19421

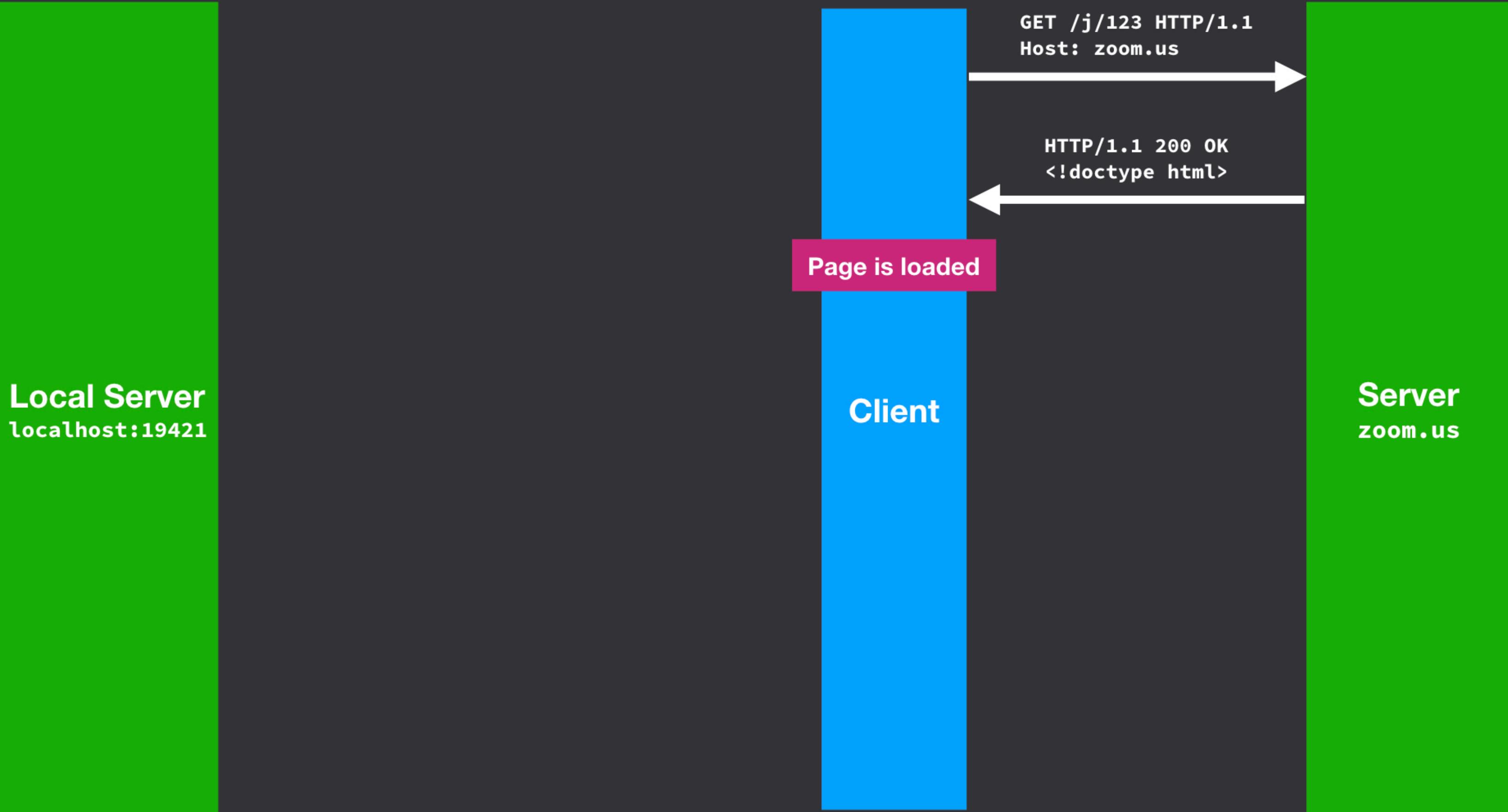
Page is loaded

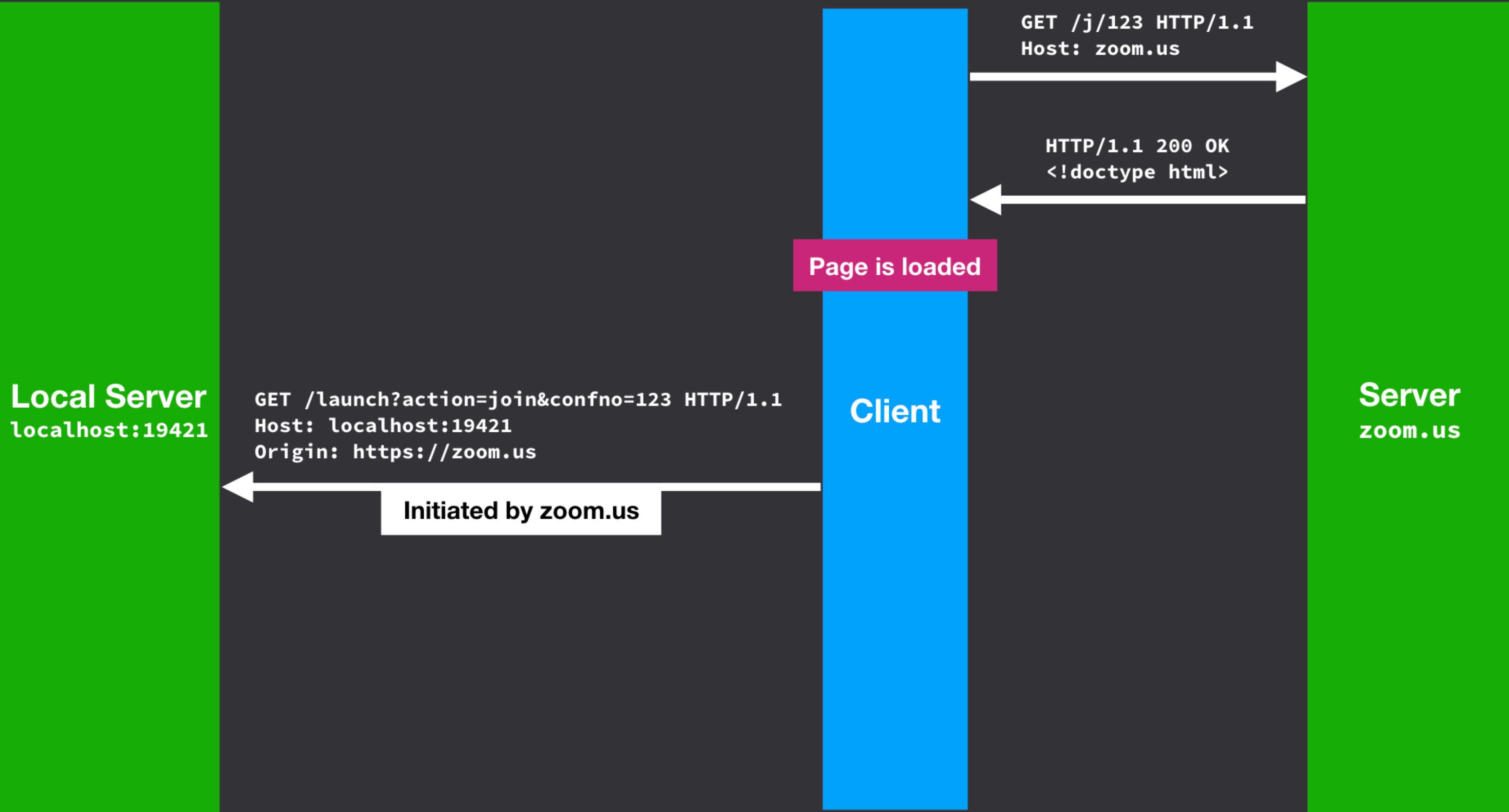
GET /j/123 HTTP/1.1
Host: zoom.us
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36
Origin: http://localhost:19421
Accept: */*
Referer: http://localhost:19421/join?token=123
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Sec-GPC: 1
TE: trailers

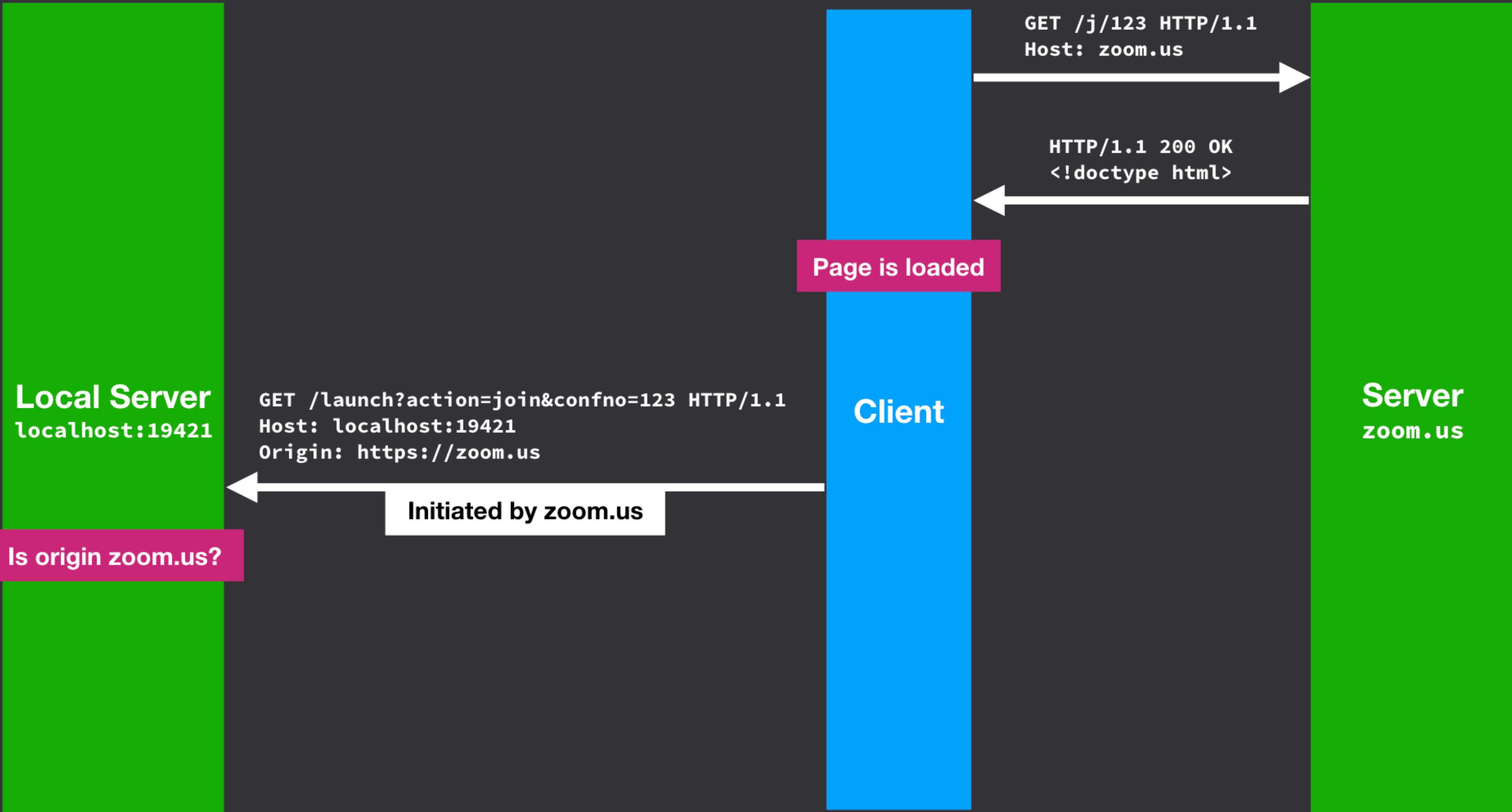
HTTP/1.1 200 OK
<!doctype html>

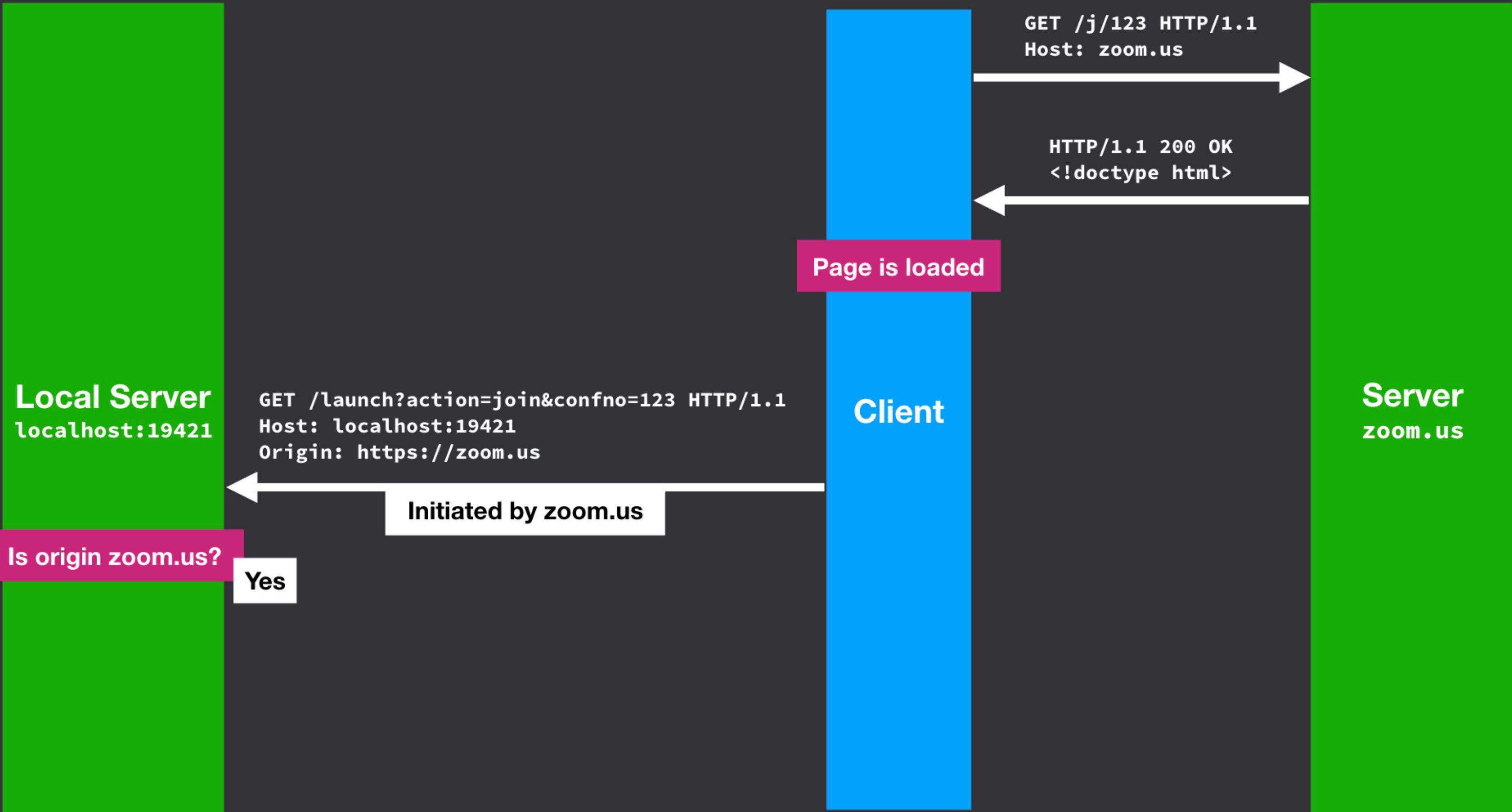
Client

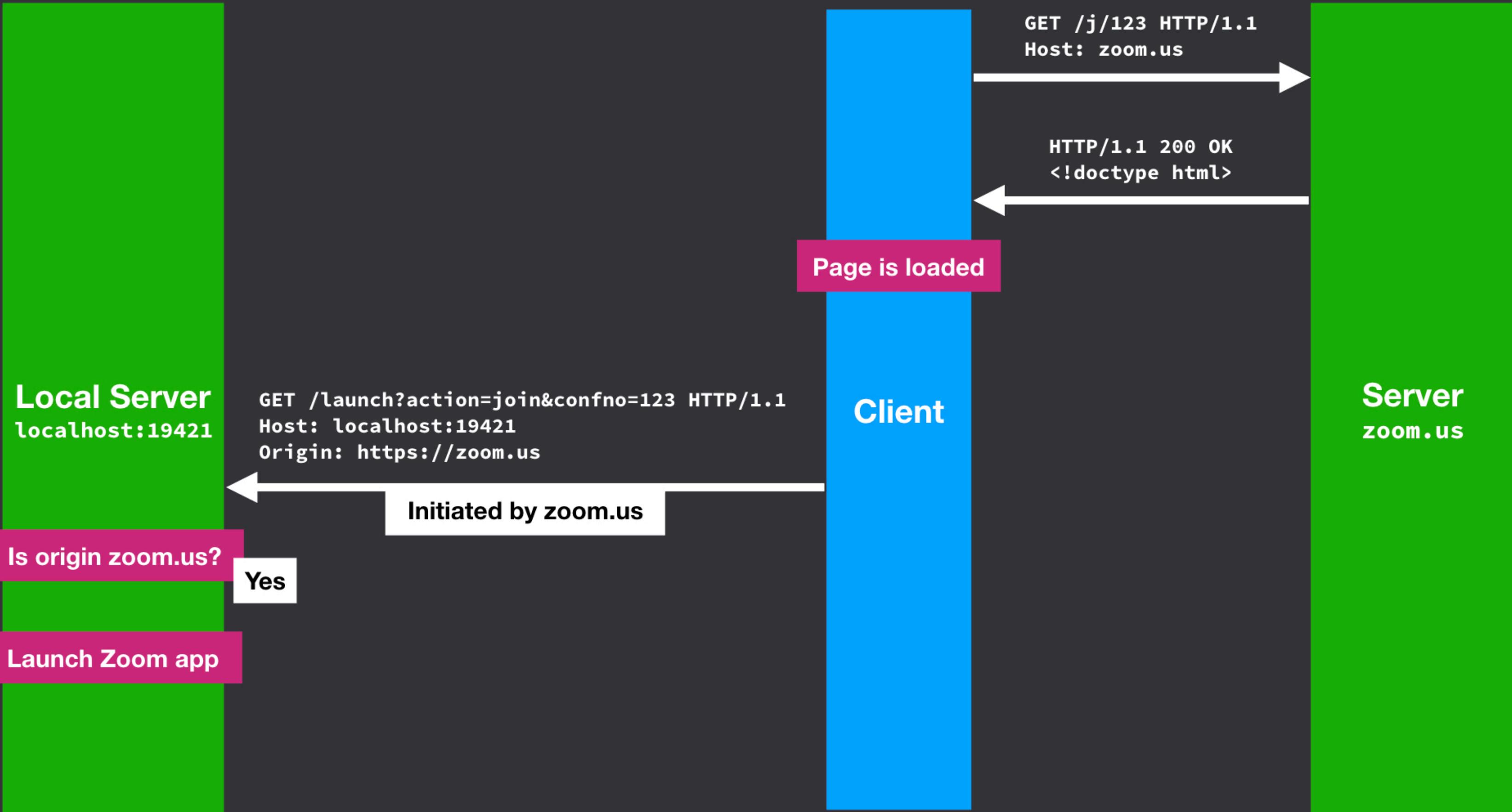
Server
zoom.us

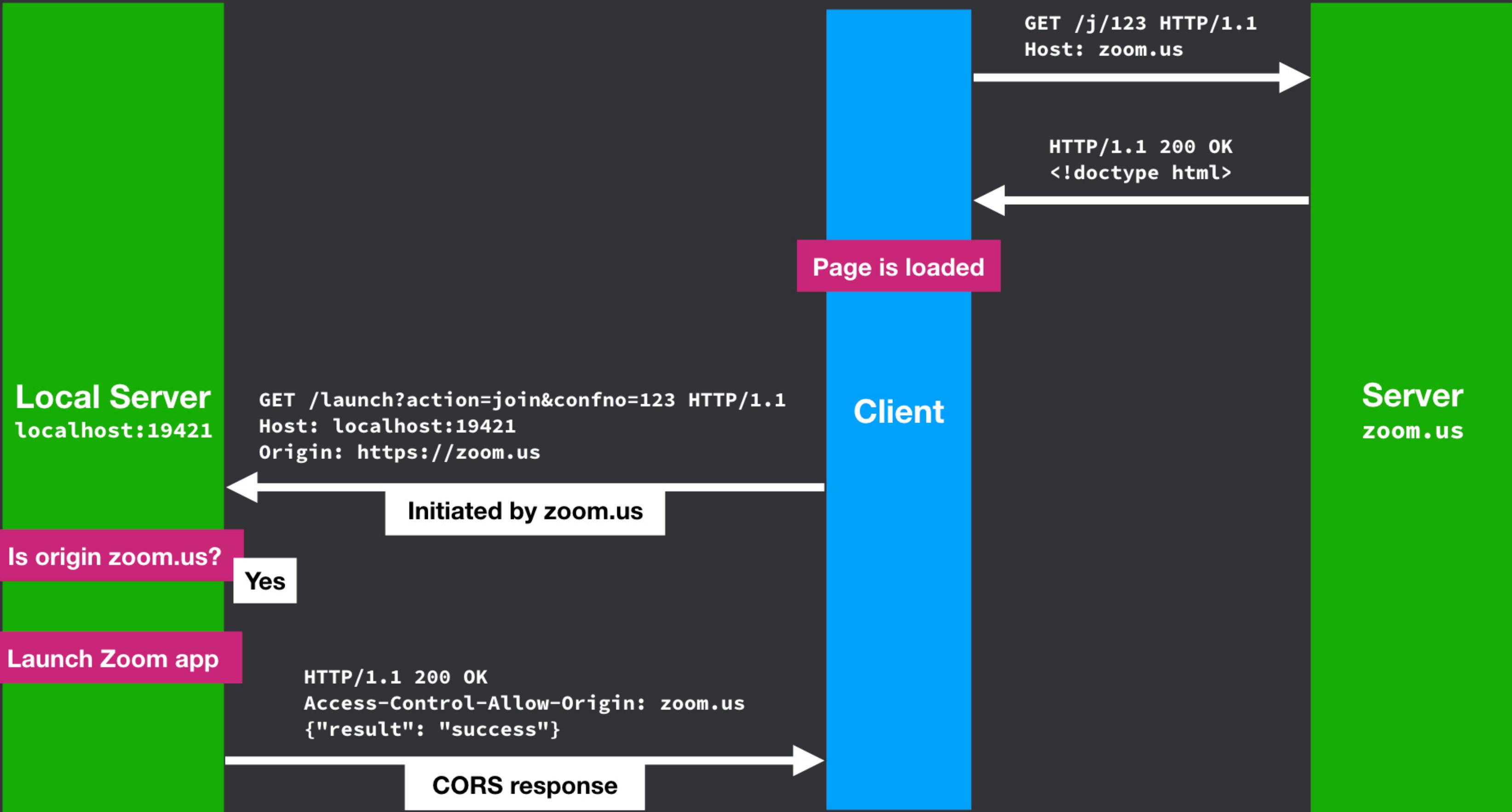


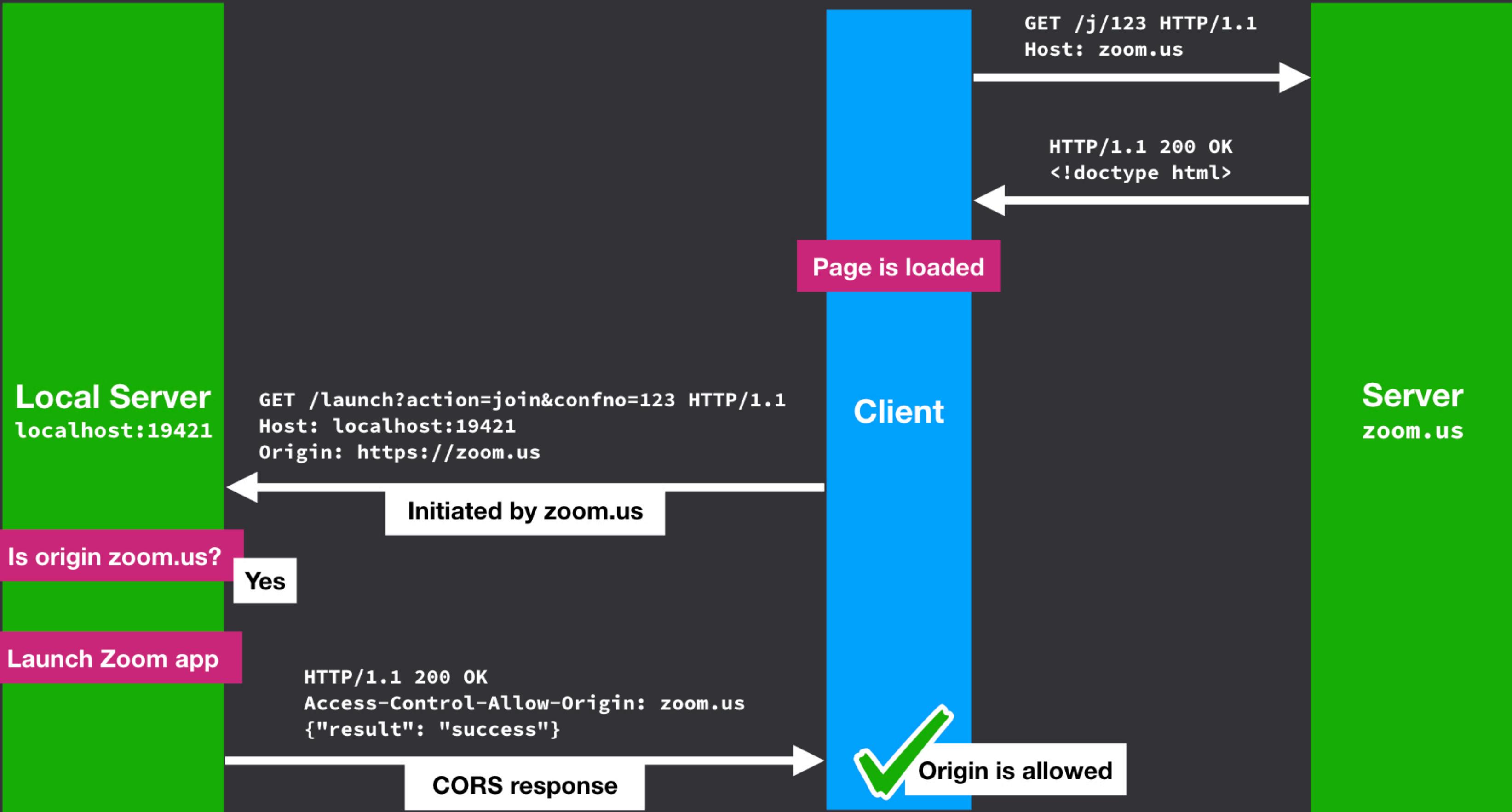




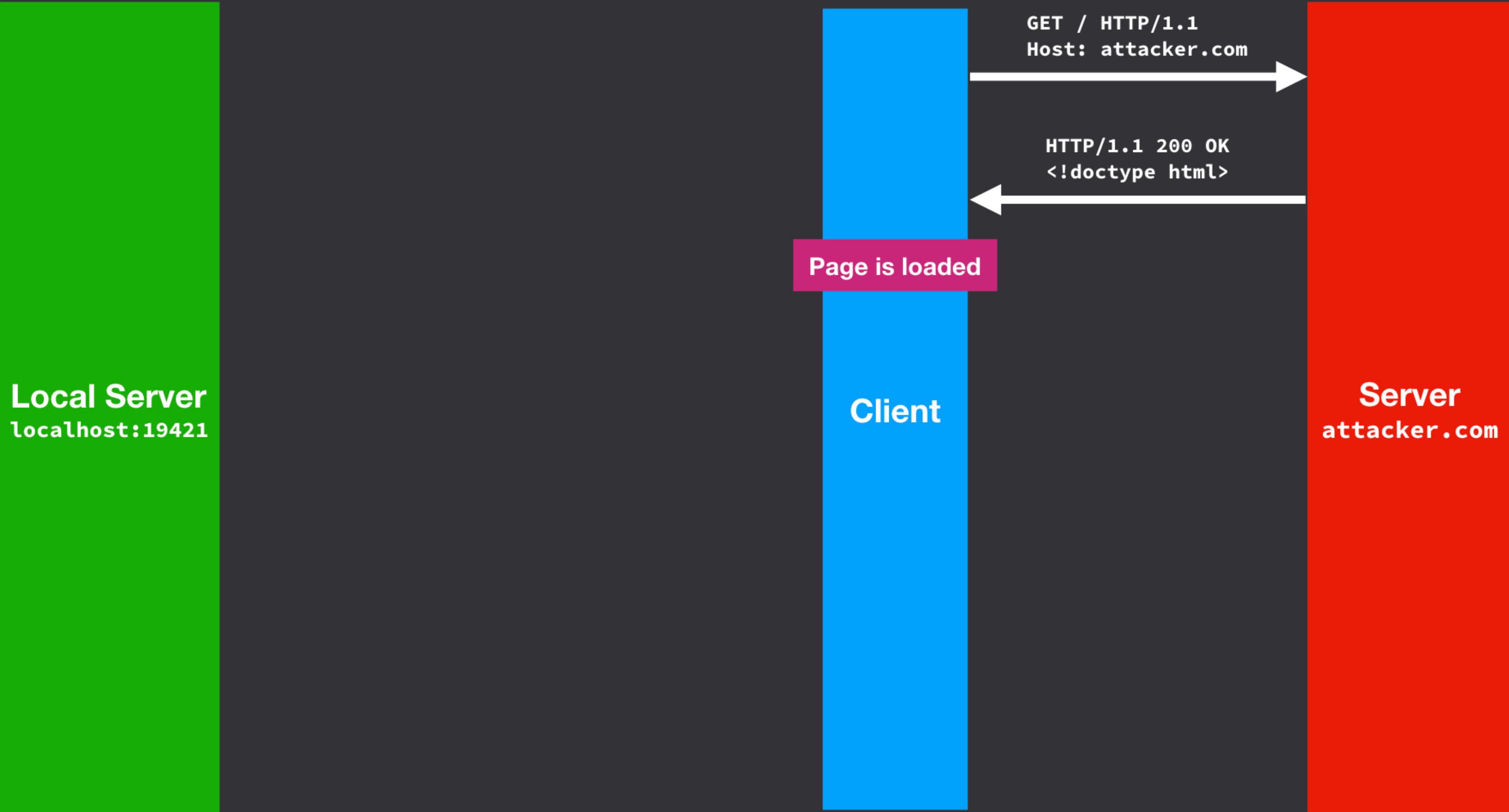


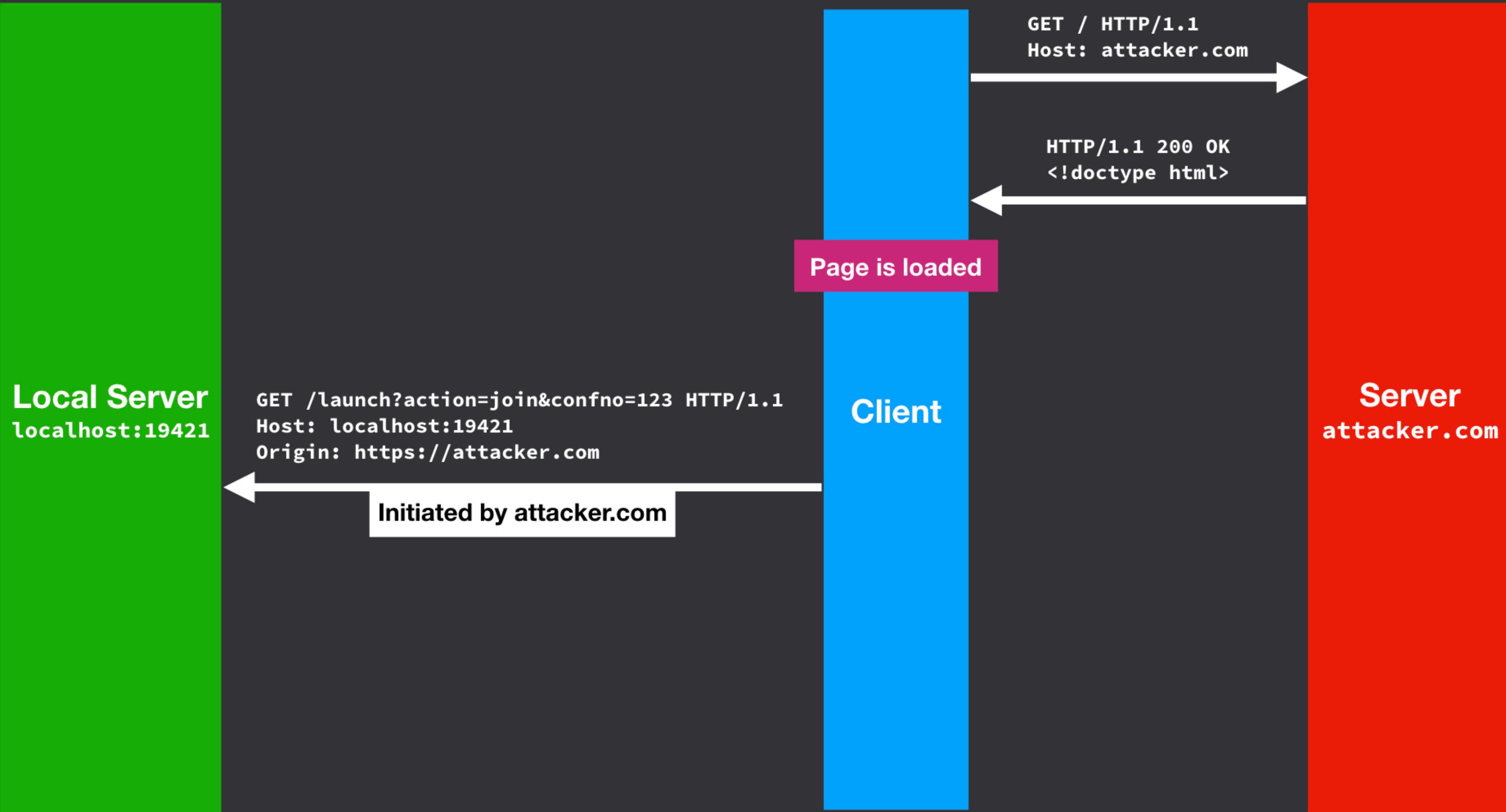


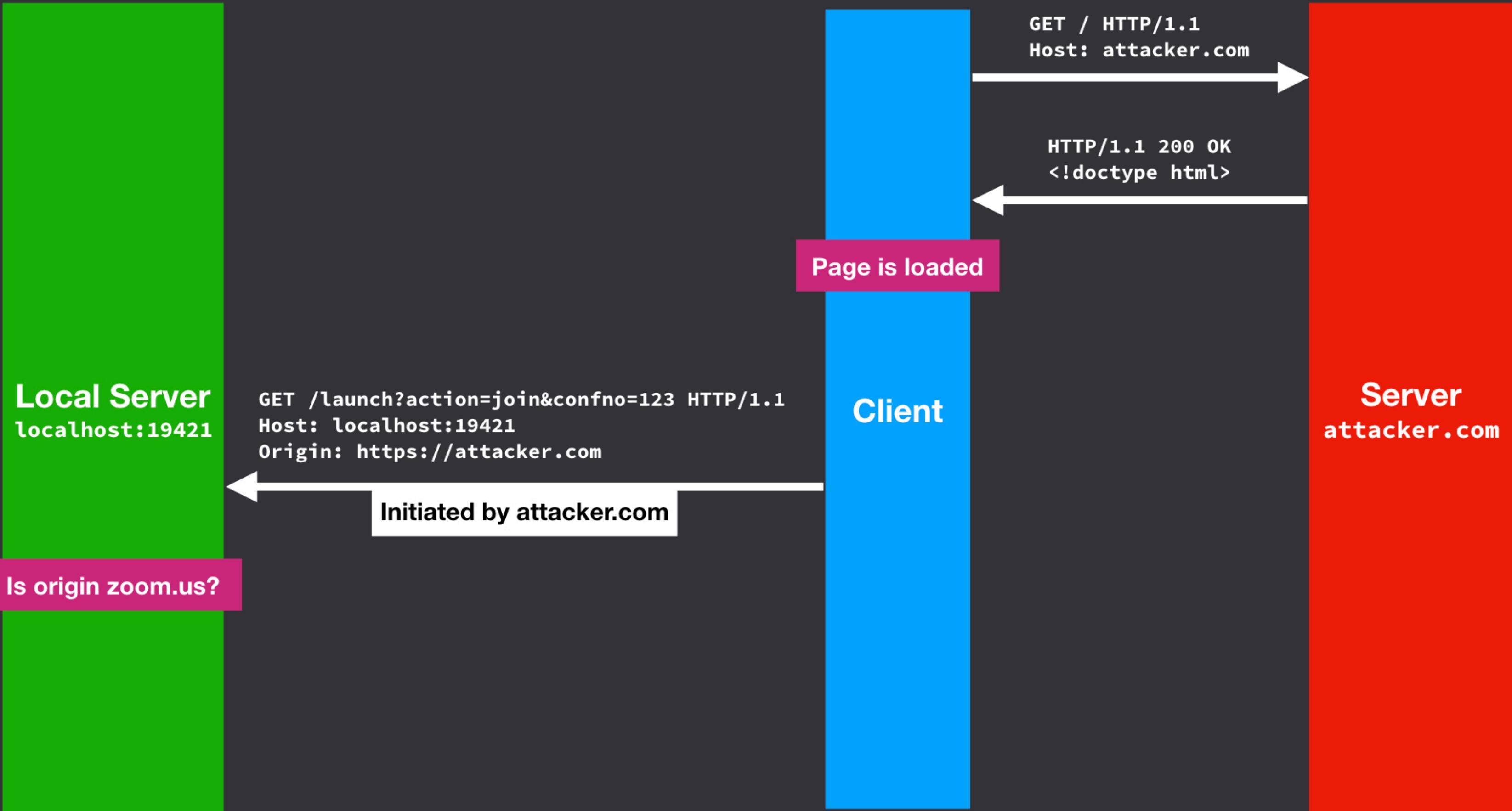


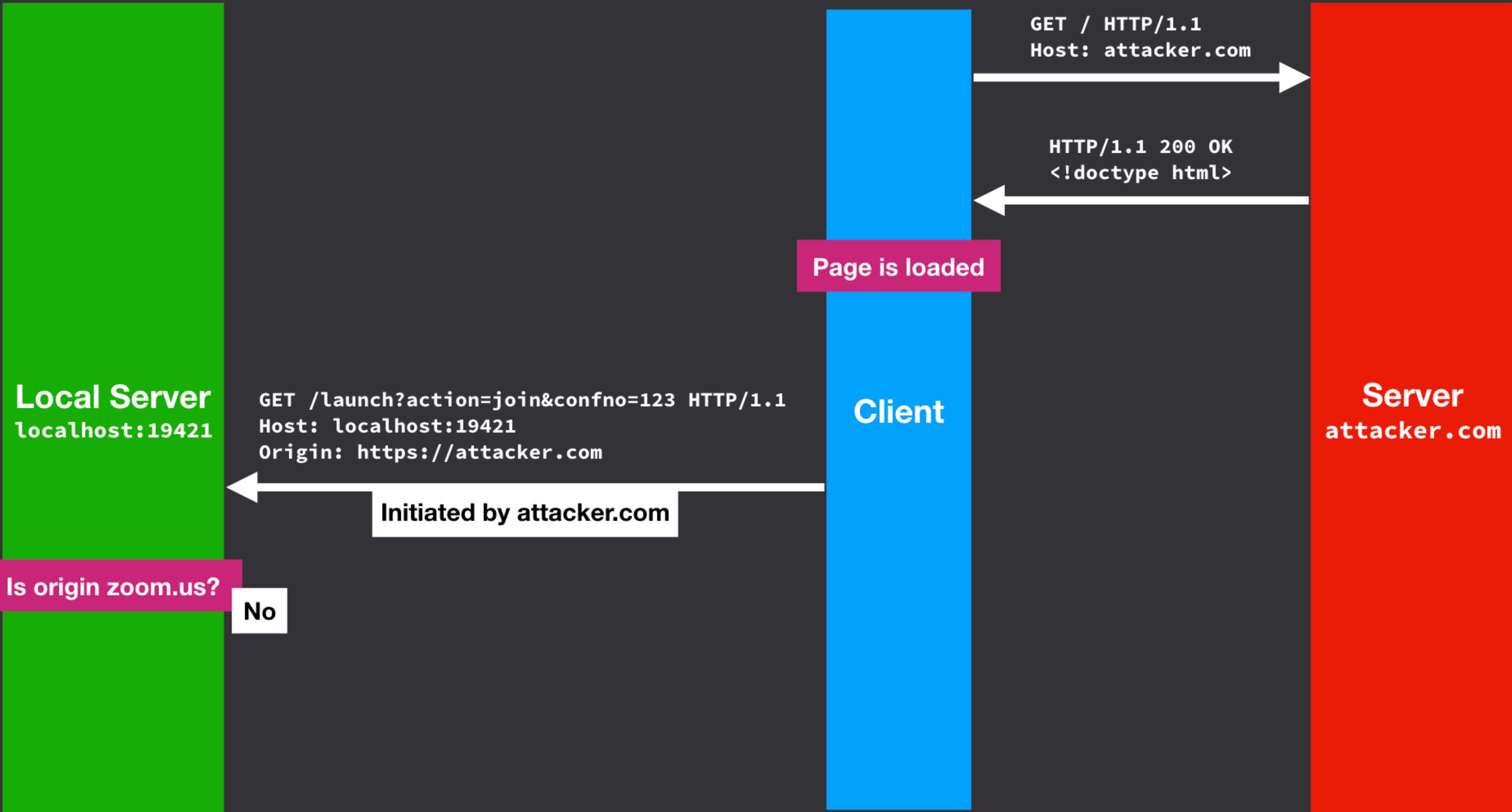


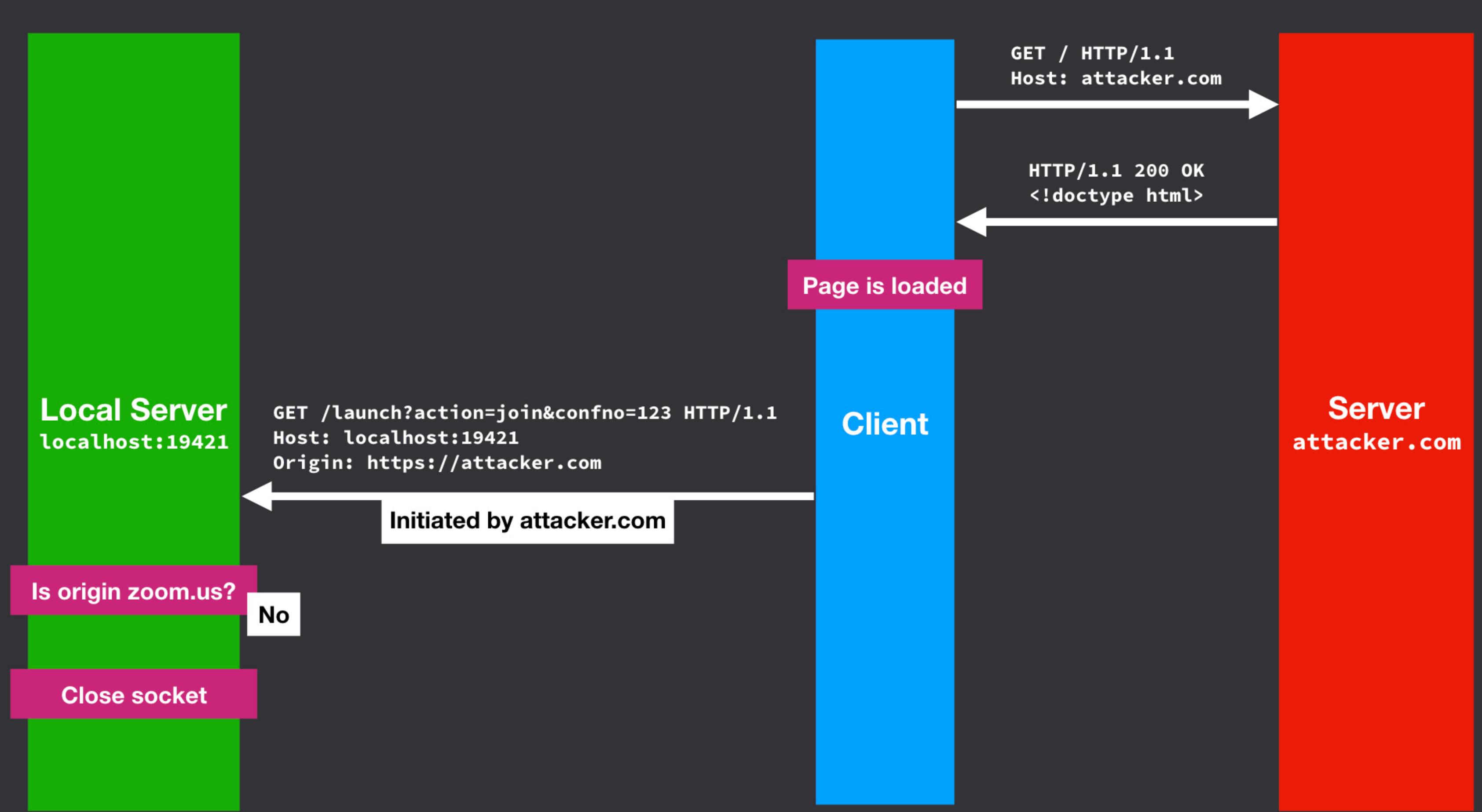
Attacker joins user into a zoom call (local server inspects Origin header)











Problems with inspecting Origin header

- **Problem:** Browser doesn't always add **Origin** header
 - for "simple" requests (e.g. `` or `<iframe>` tags)
 - for same origin requests (e.g. `fetch()` to same origin)
 - Very old browsers
- **Solution:** block requests where **Origin** header is omitted
- **Solution:** change the endpoint to require a "preflighted" request so that **Origin** header is always sent (e.g. change the HTTP method to PUT)

"Simple" HTTP requests

- An HTTP/1.1 **GET**, **HEAD** or a **POST** is the request method
- In the case of a POST, the **Content-Type** of the request body is one of **application/x-www-form-urlencoded**, **multipart/form-data**, or **text/plain**
- No custom HTTP headers are set (or, only CORS-safelisted headers are set)

"Preflighted" HTTP requests

- Before a "preflighted" requests can be sent to the target server, the browser must check that it is safe to send
- So it first sends an HTTP request with the **OPTIONS** method to the same URL

What happens if the browser does not preflight "non-simple" requests

Client

Server
victim.com

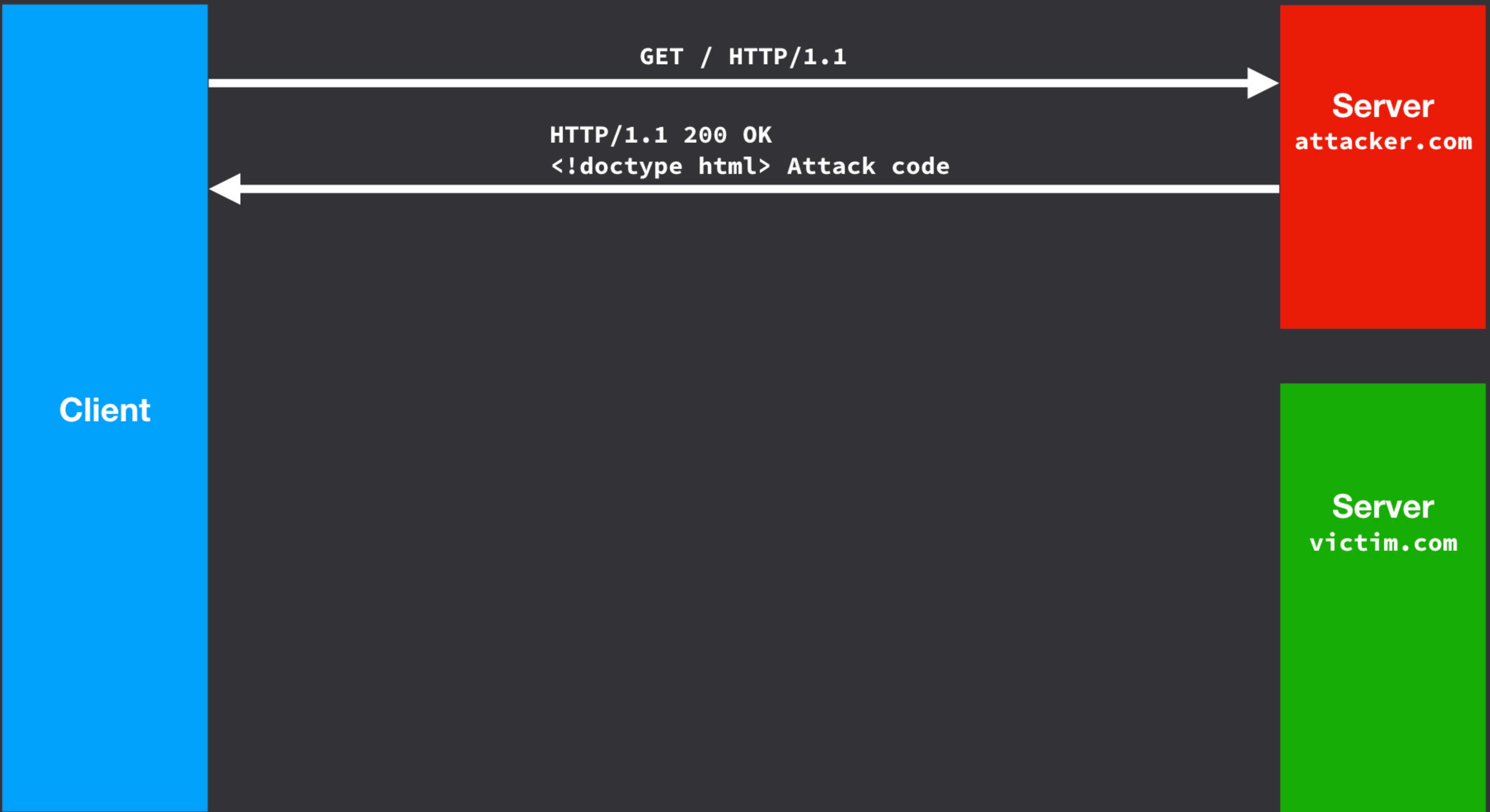
Server
victim.com

Client

Server
attacker.com

Server
victim.com















Introducing the OPTIONS request

- Browser sends **OPTIONS** request first to ask the server if the request we want to send is okay
- If server doesn't support **OPTIONS** (either because it is old or because it doesn't want to support preflighted requests) then, preflighted requests are **denied**
- Let's see how it can protect our local HTTP server

User joins a zoom call (local server requires "preflighted" request)

Local Server
localhost:19421

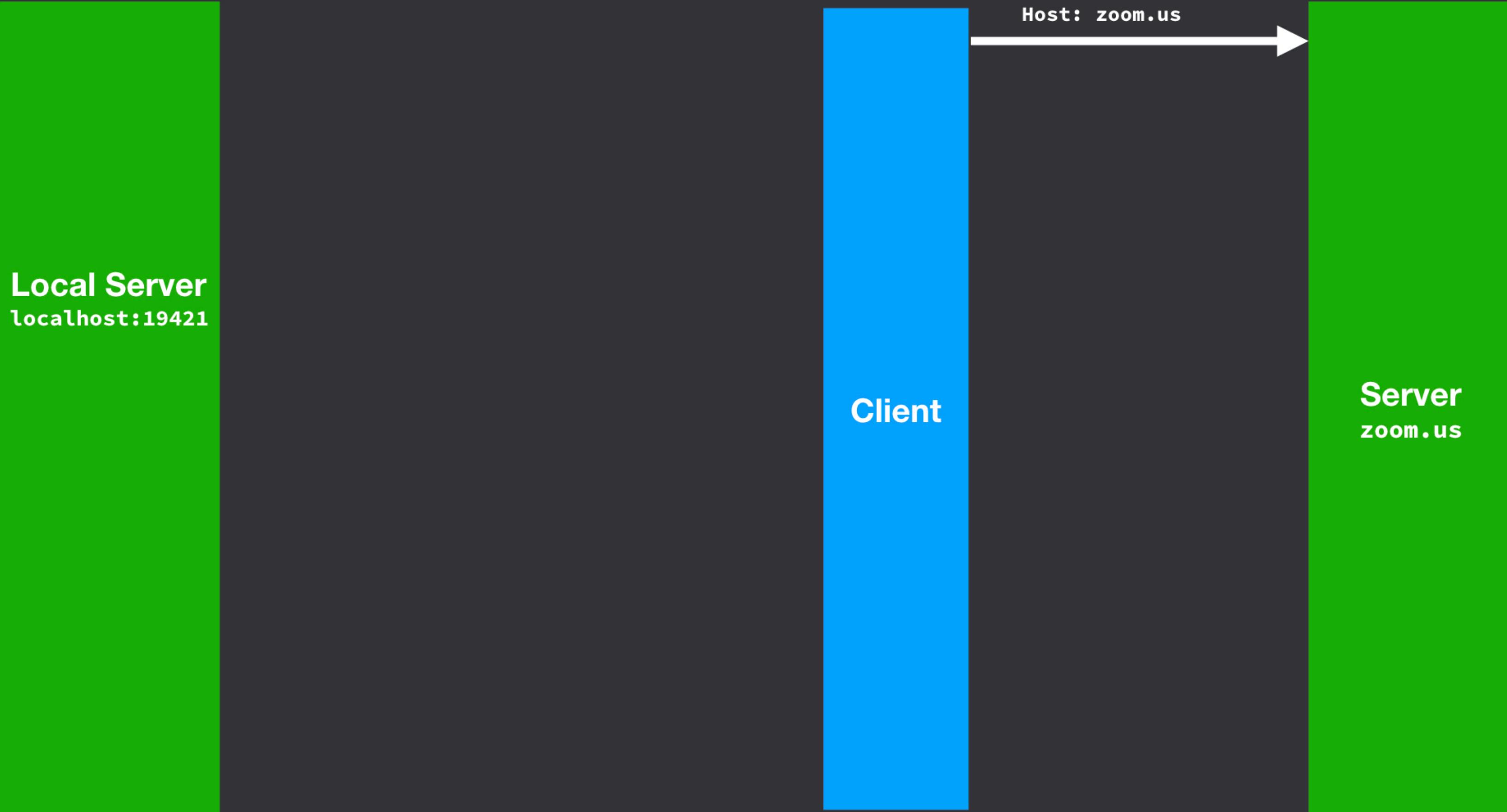
Client

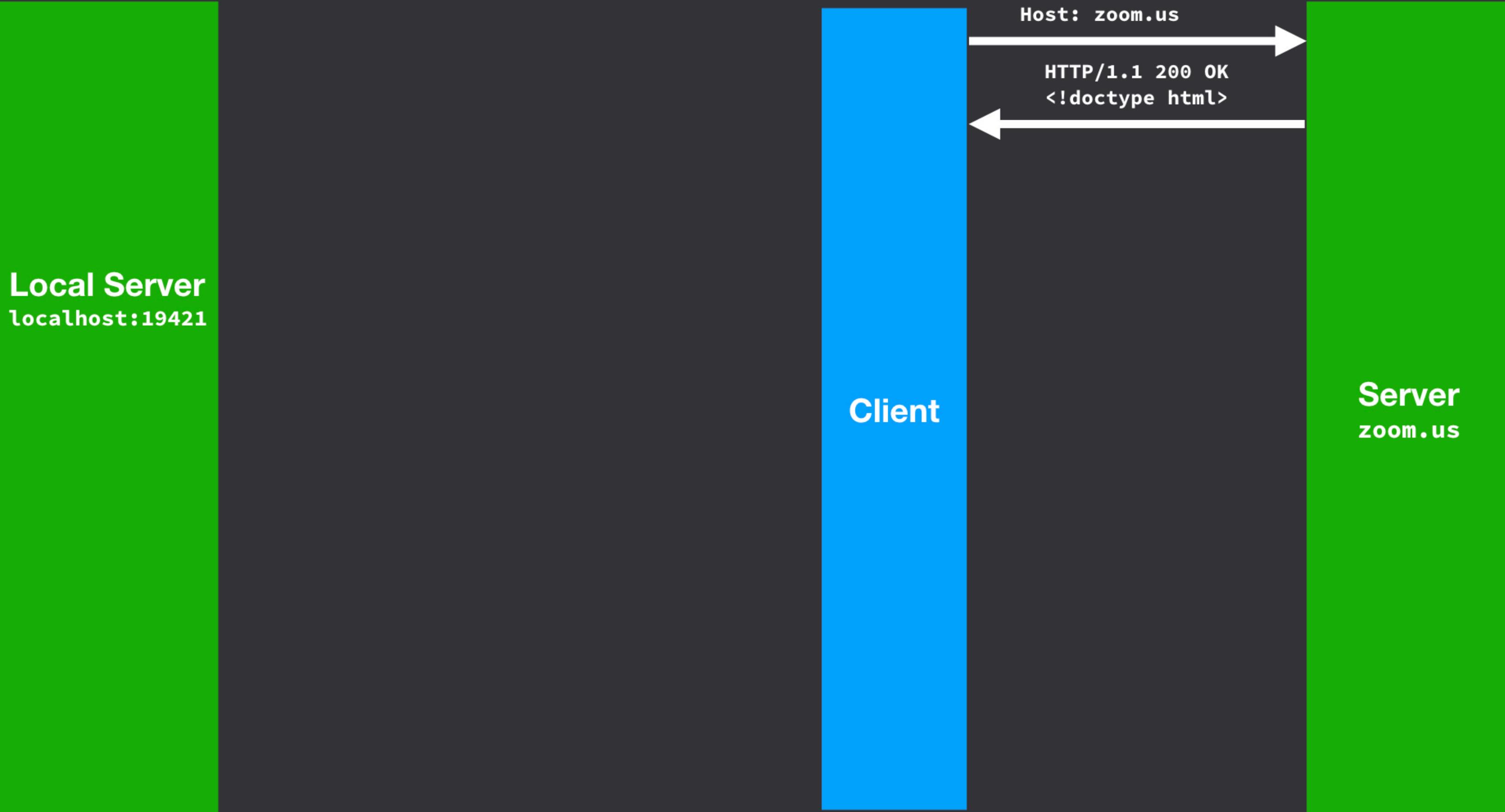
Server
zoom.us

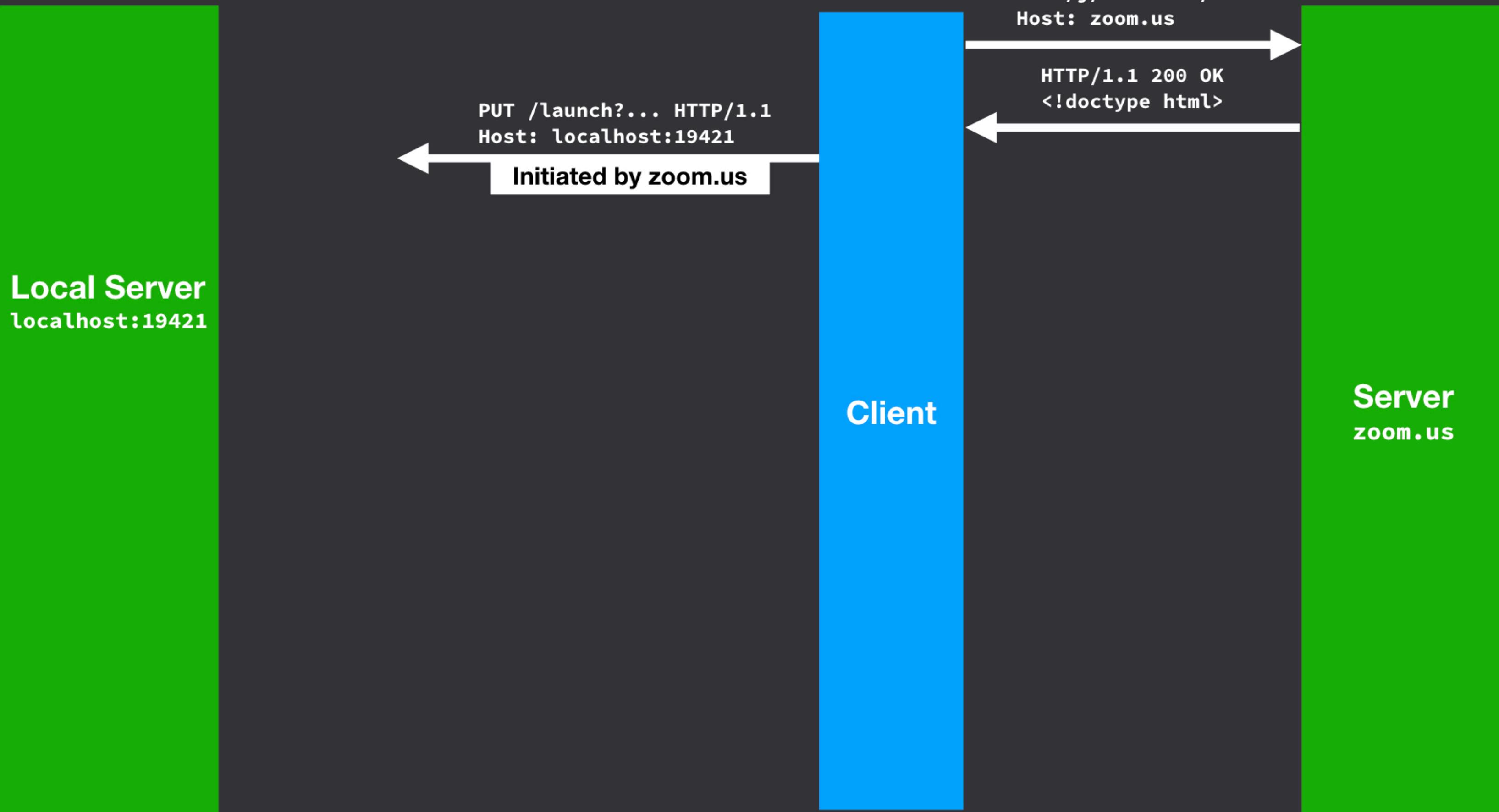
Local Server
`localhost:19421`

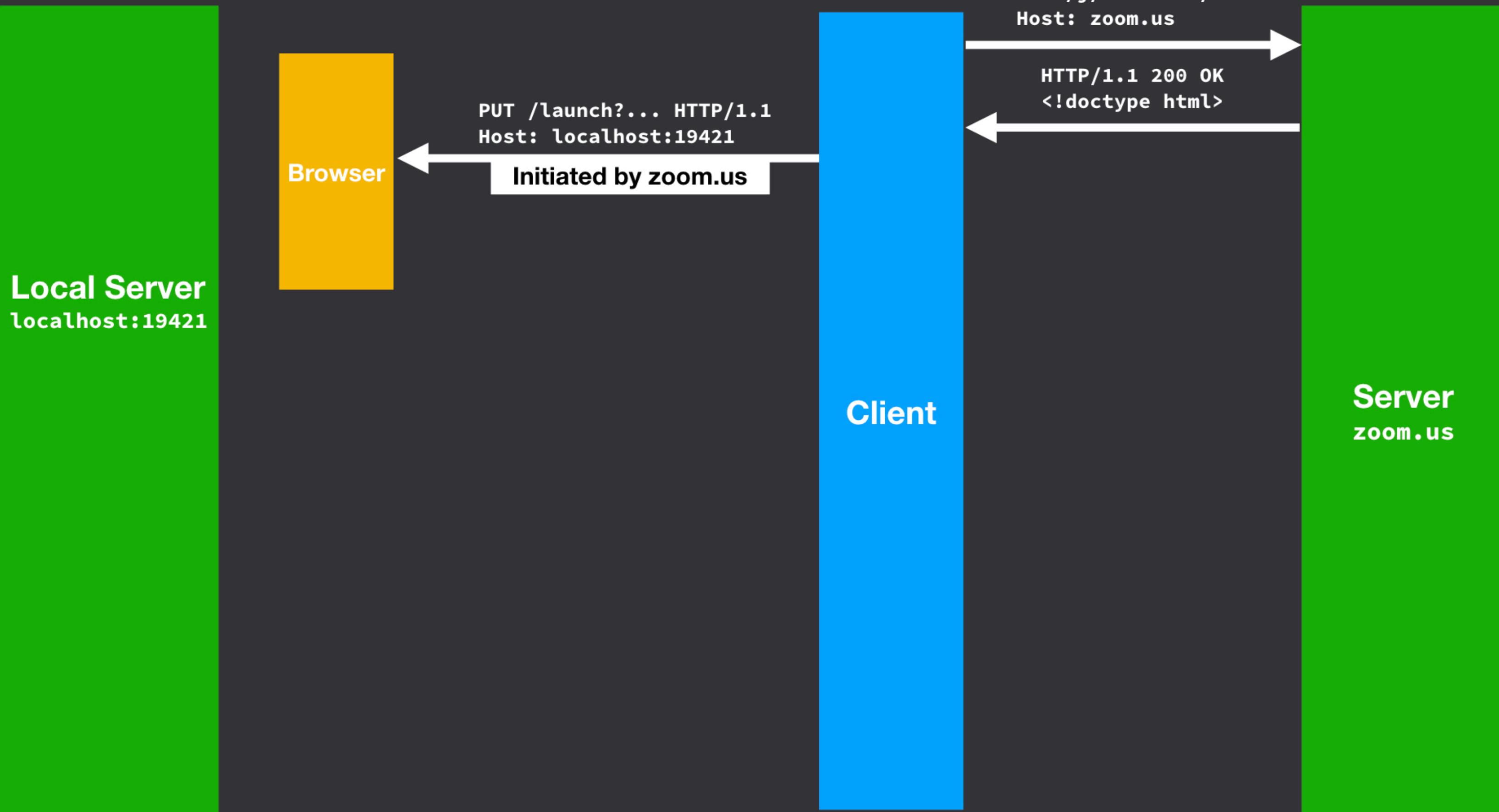
Client

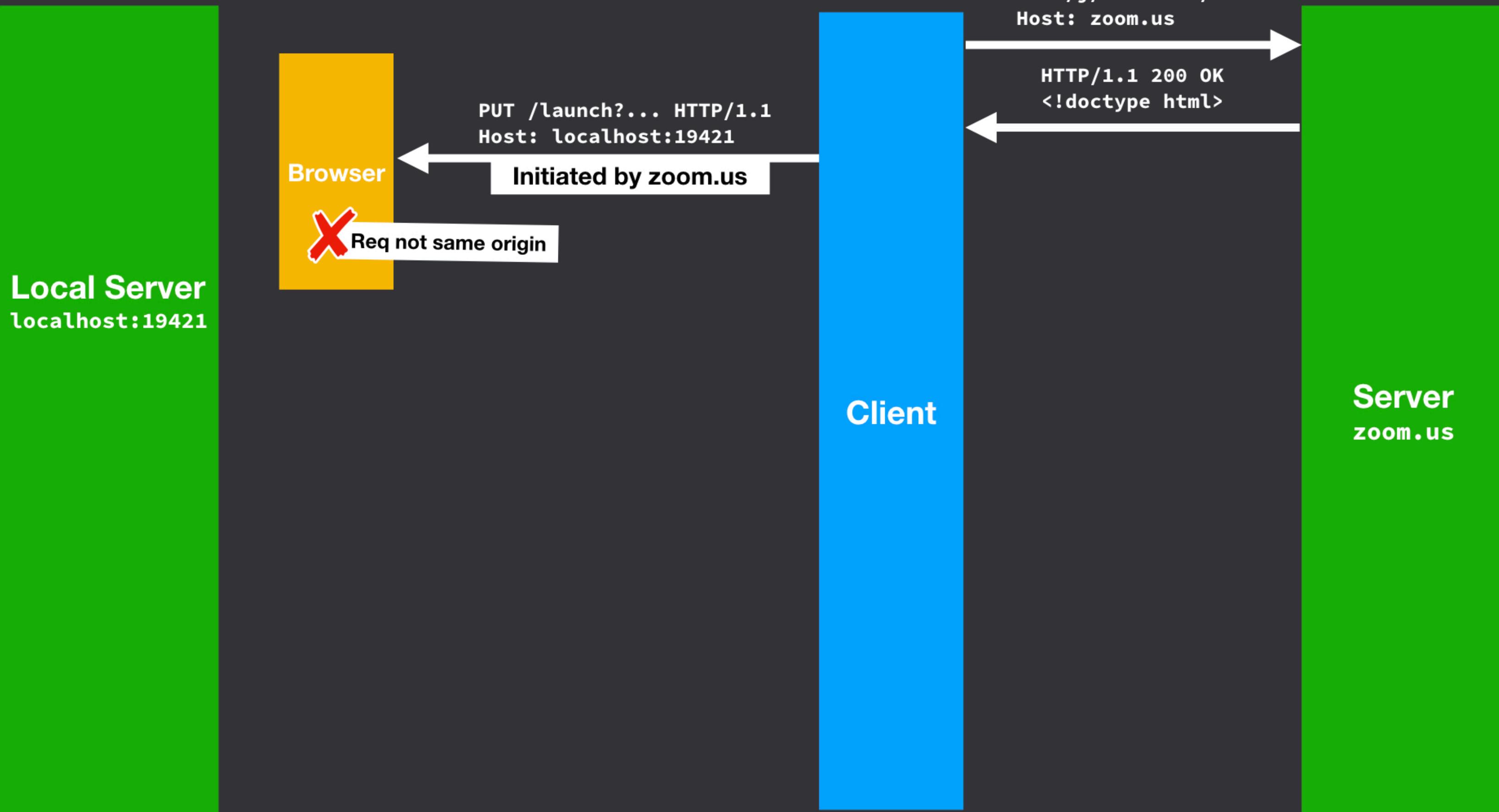
Server
`zoom.us`

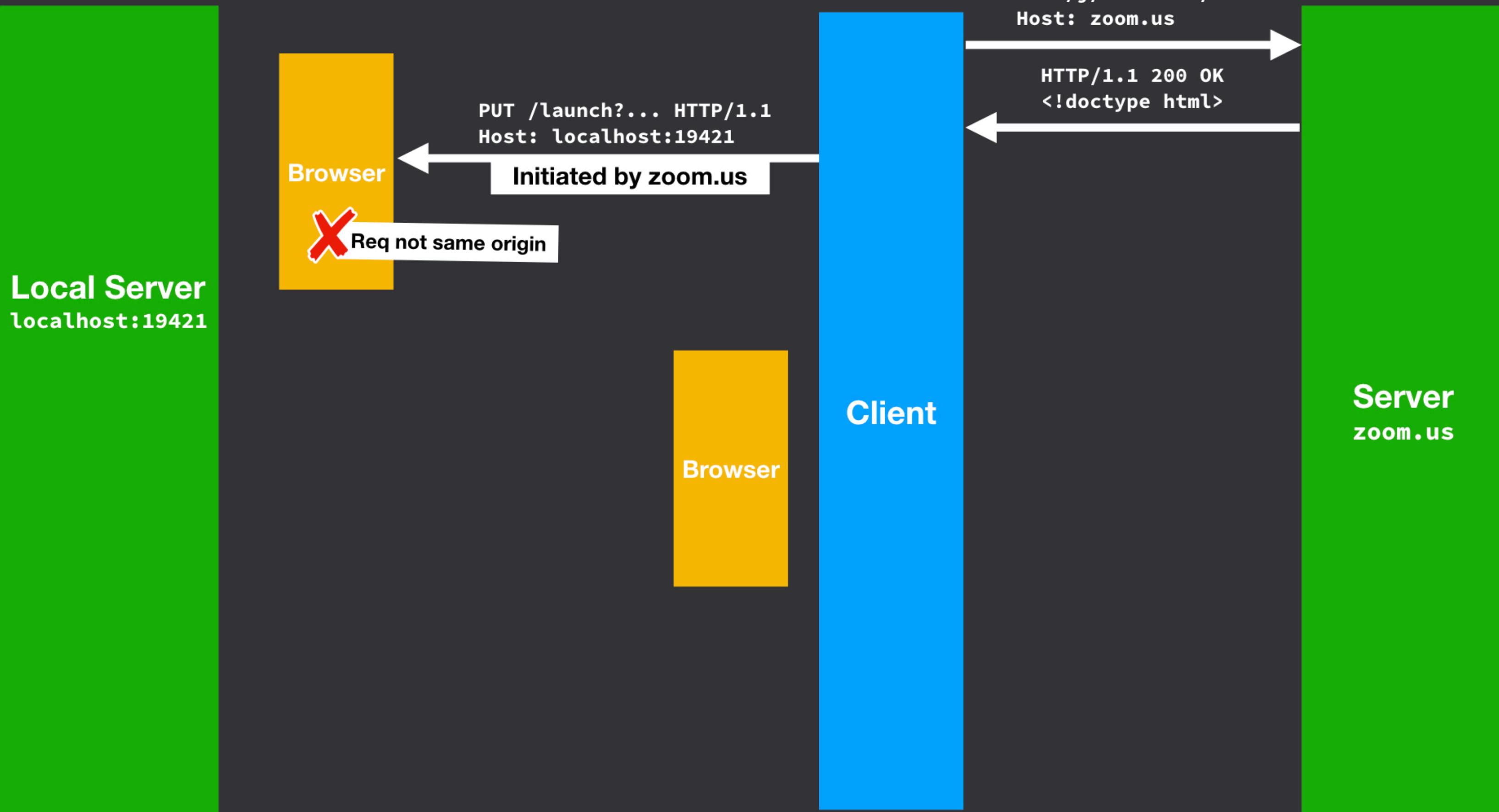


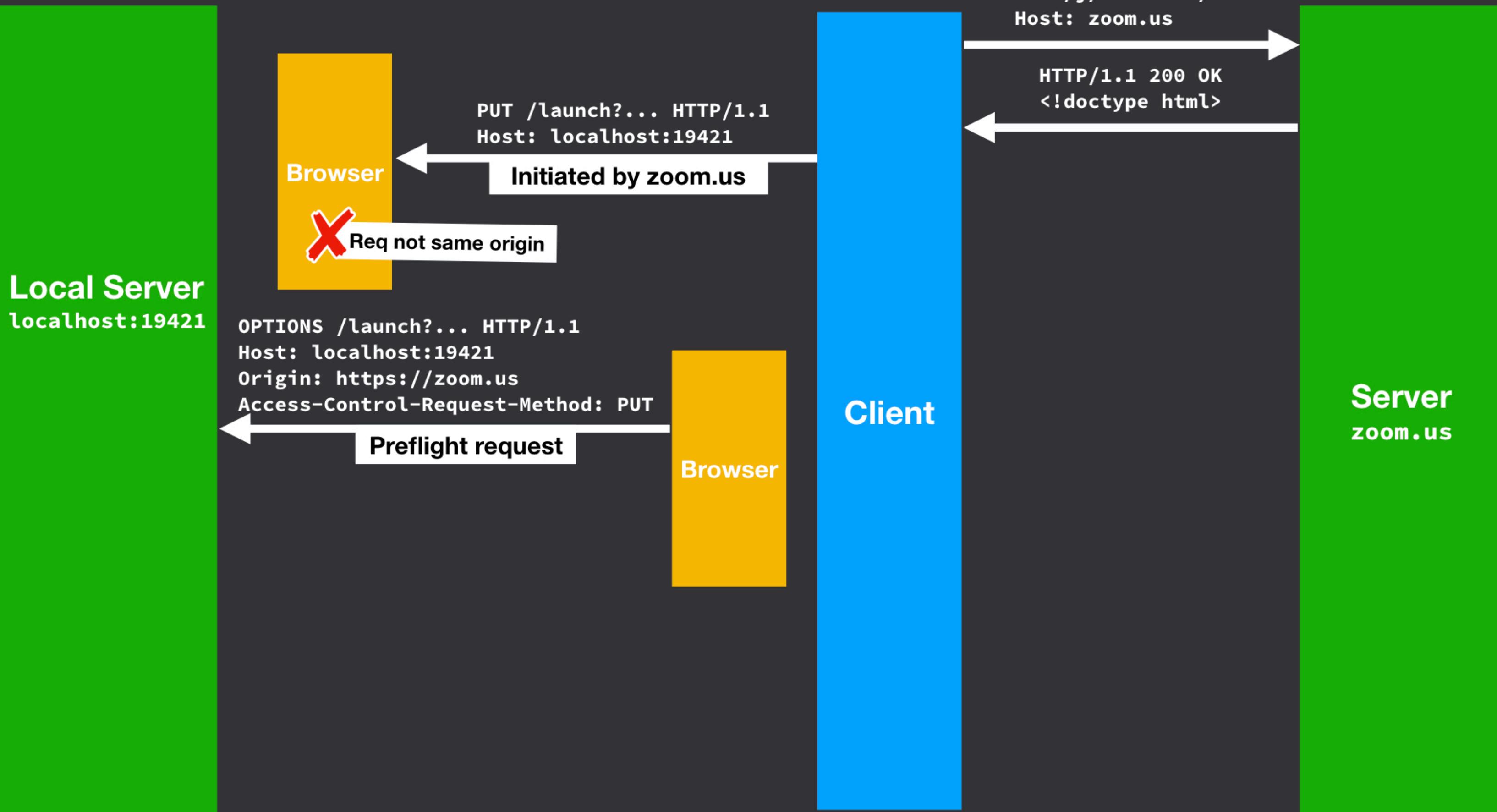


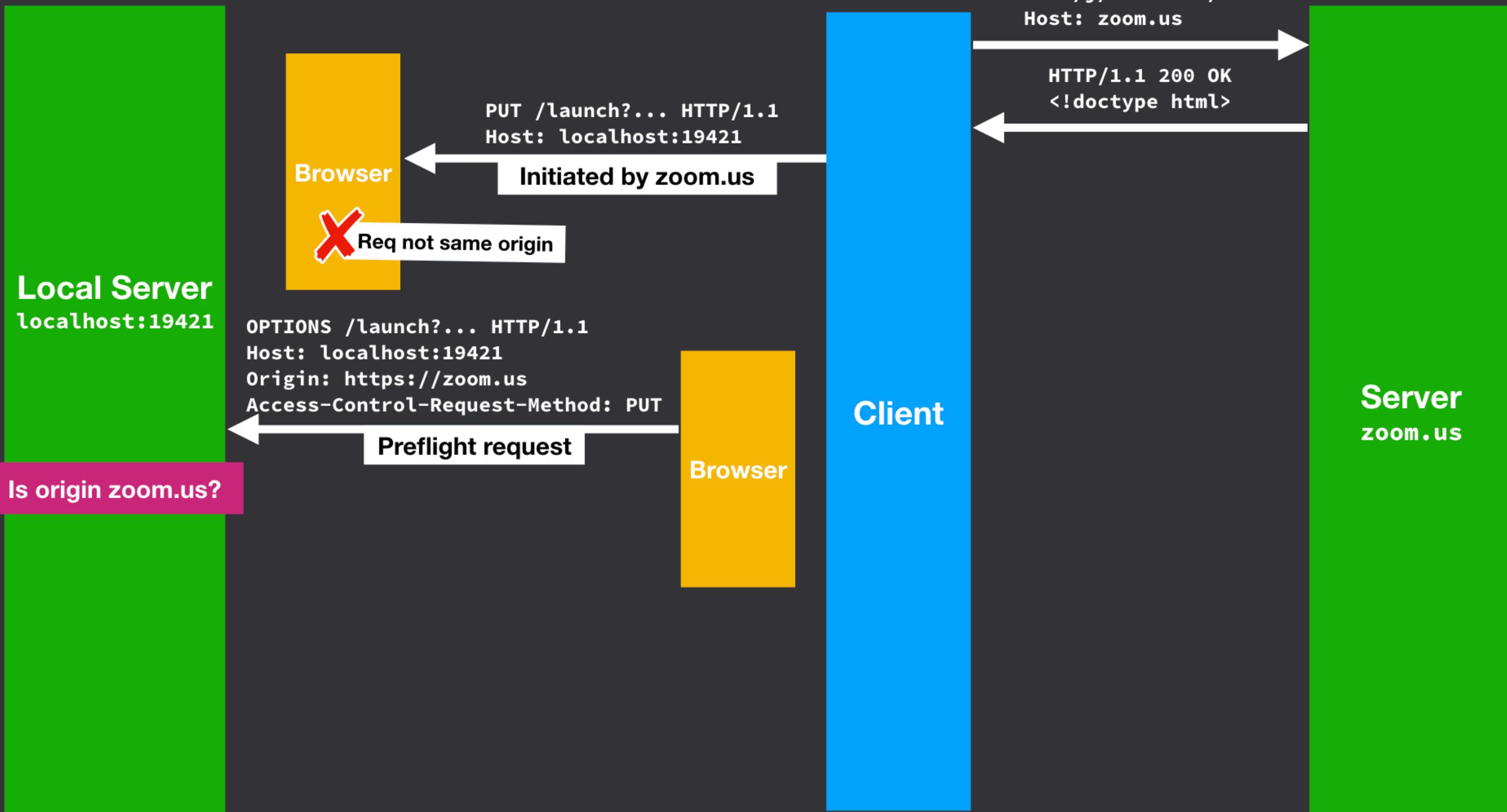


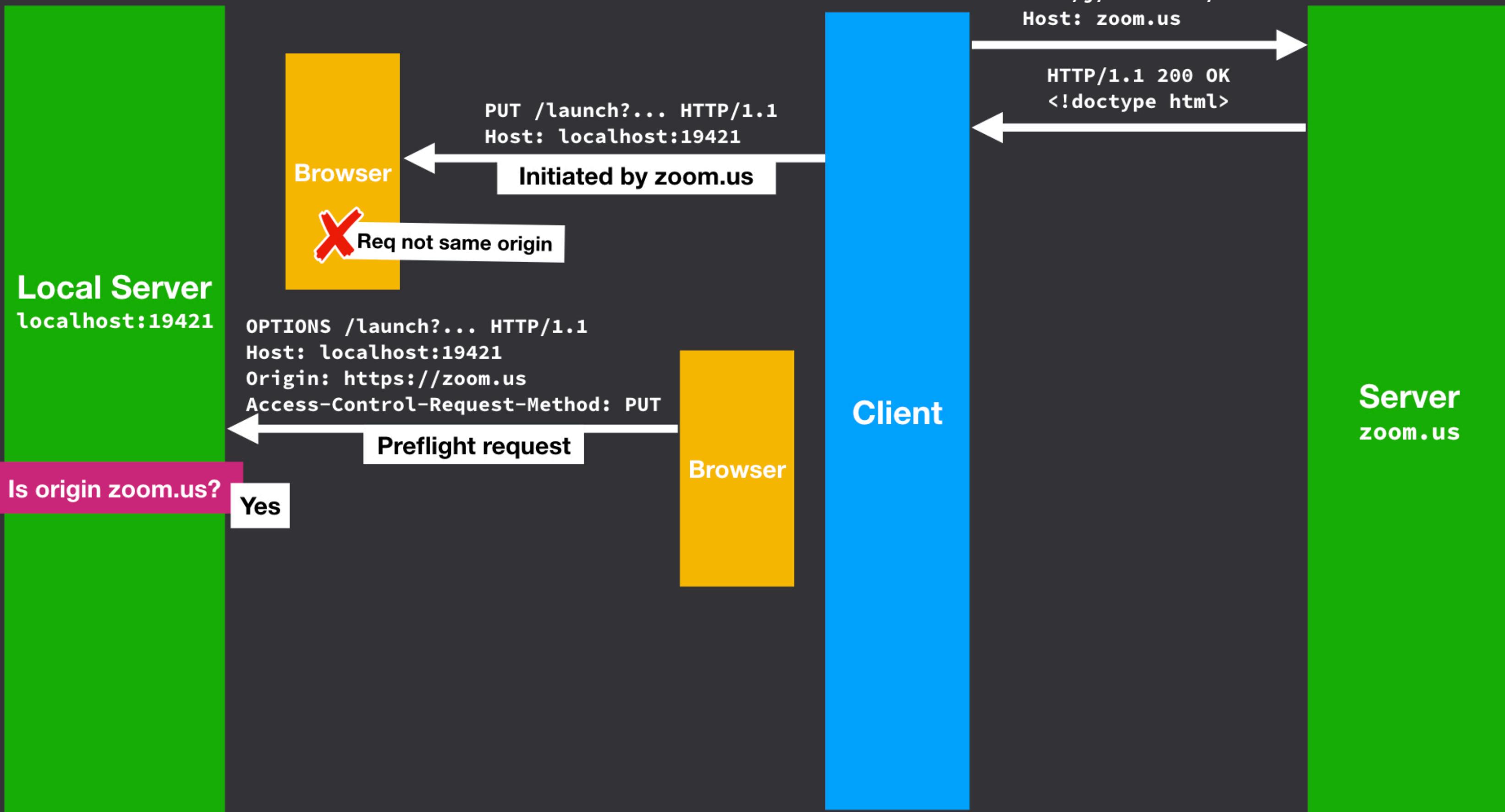


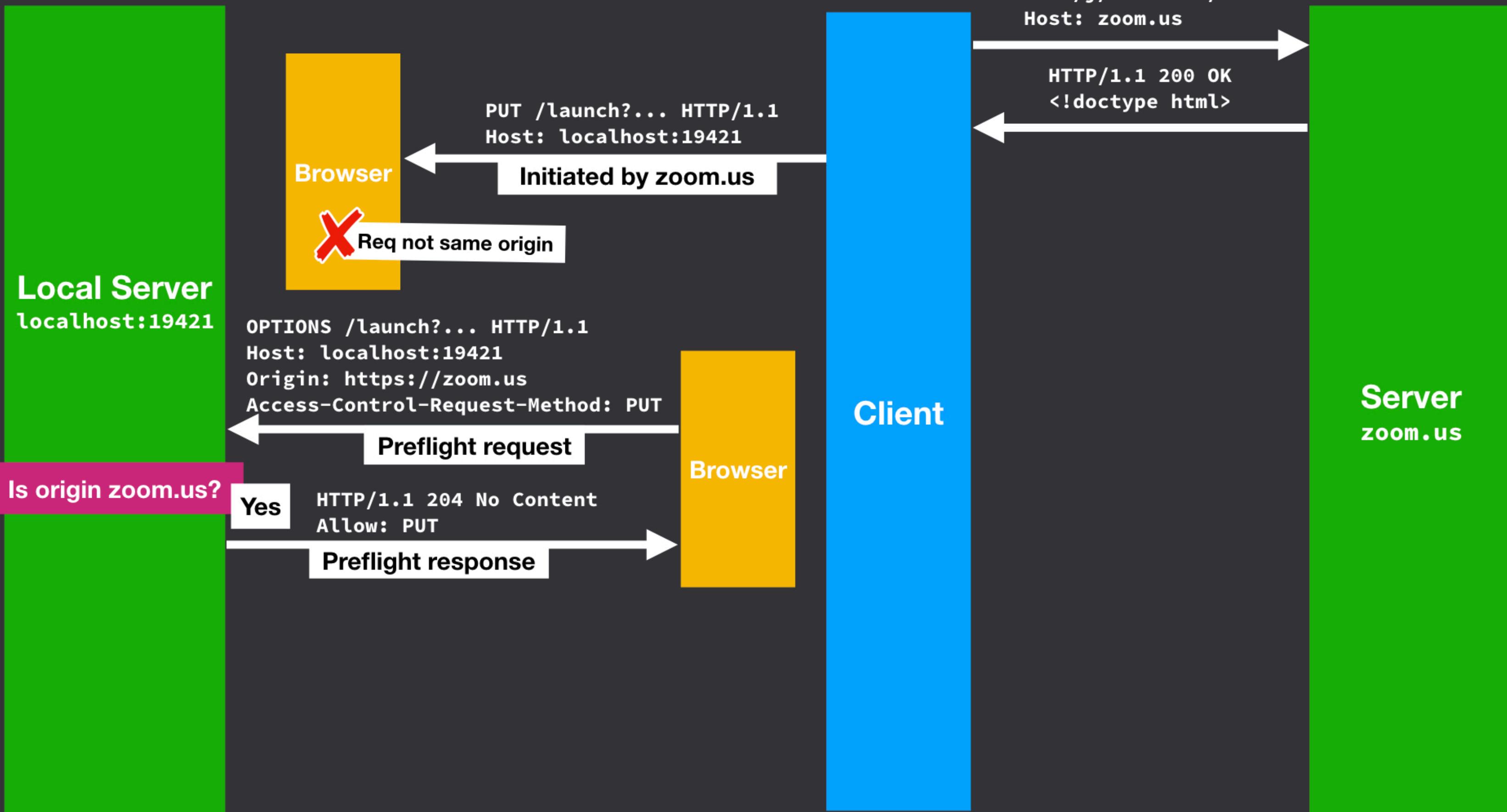


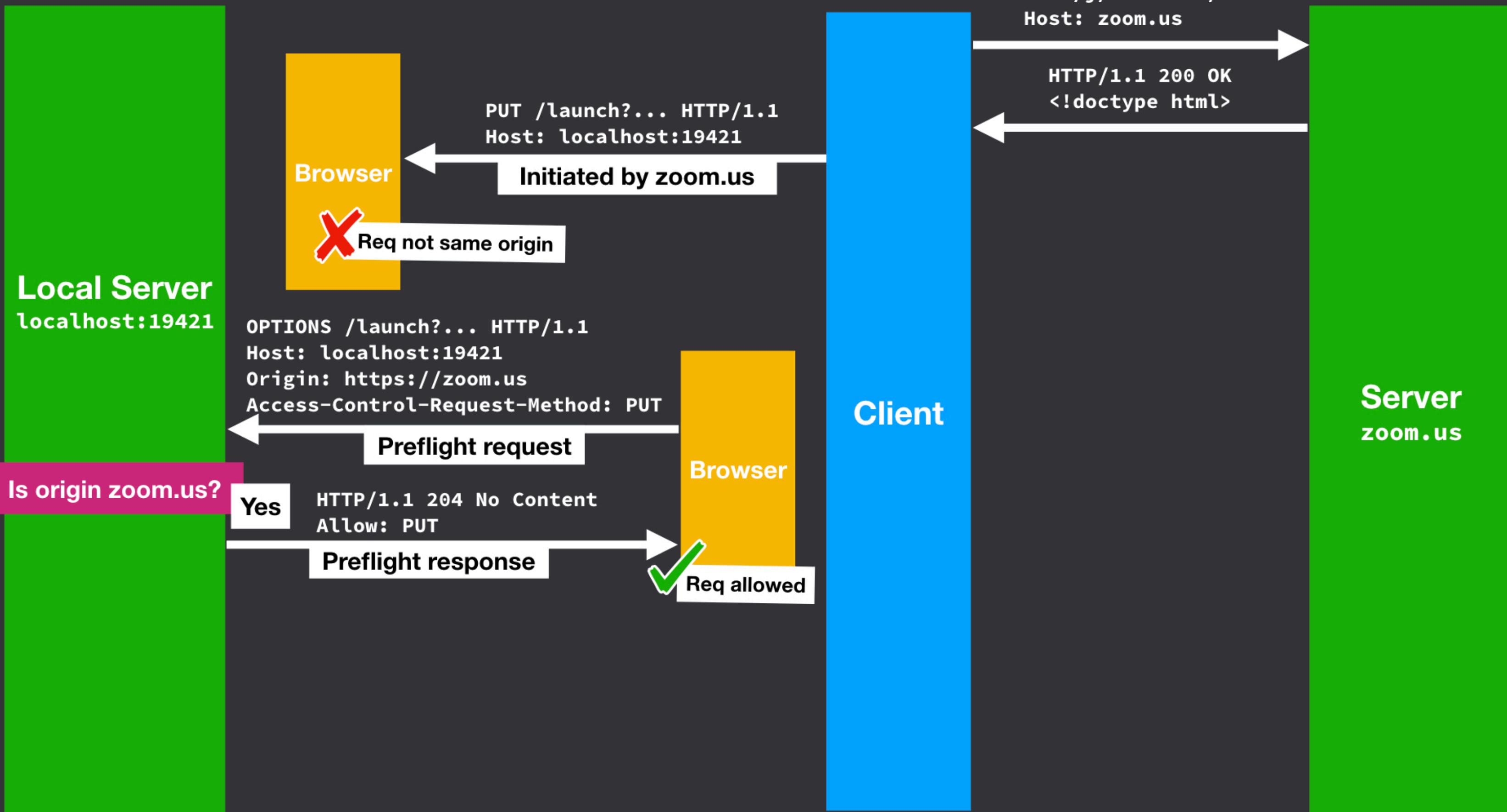


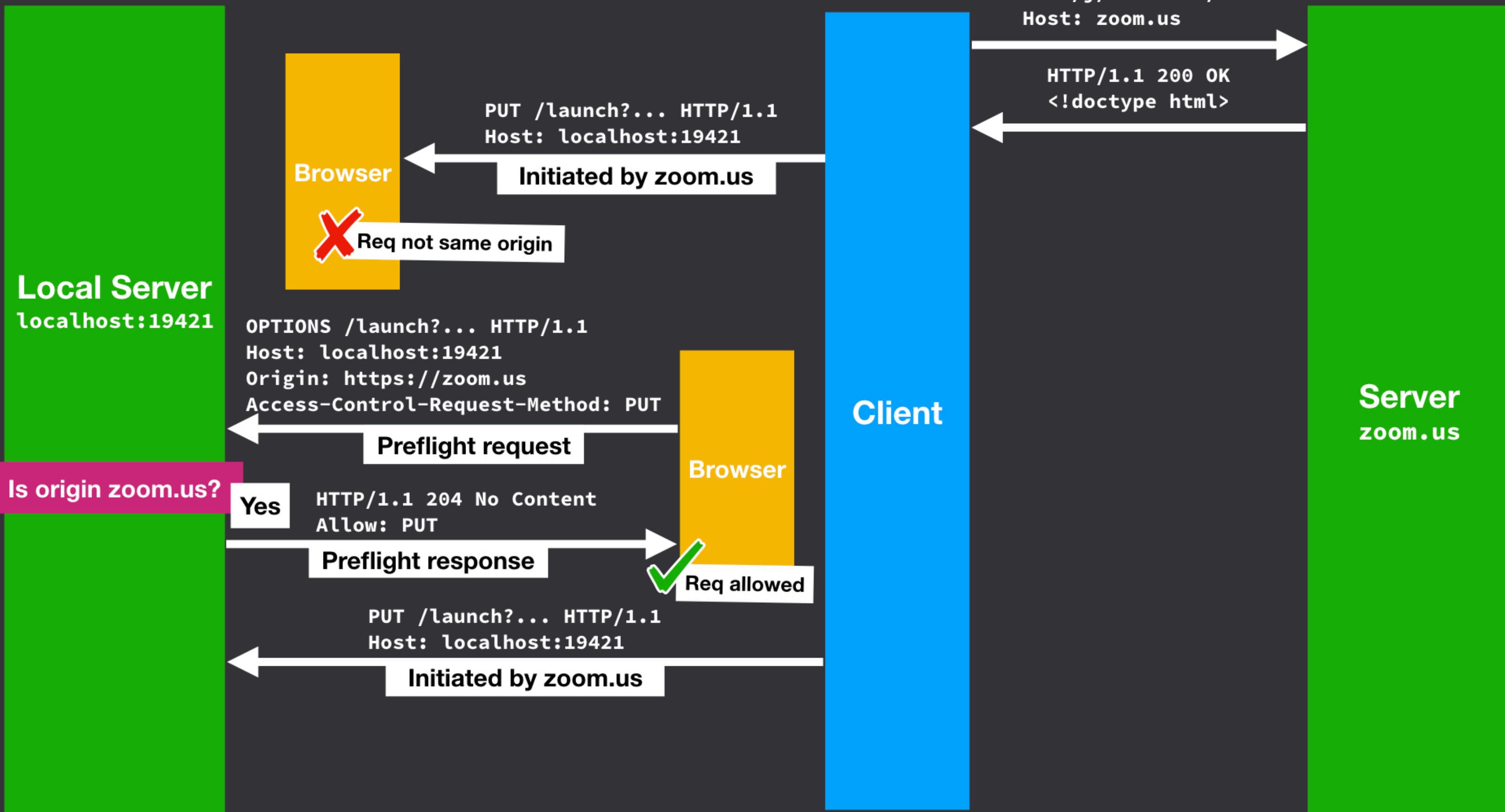


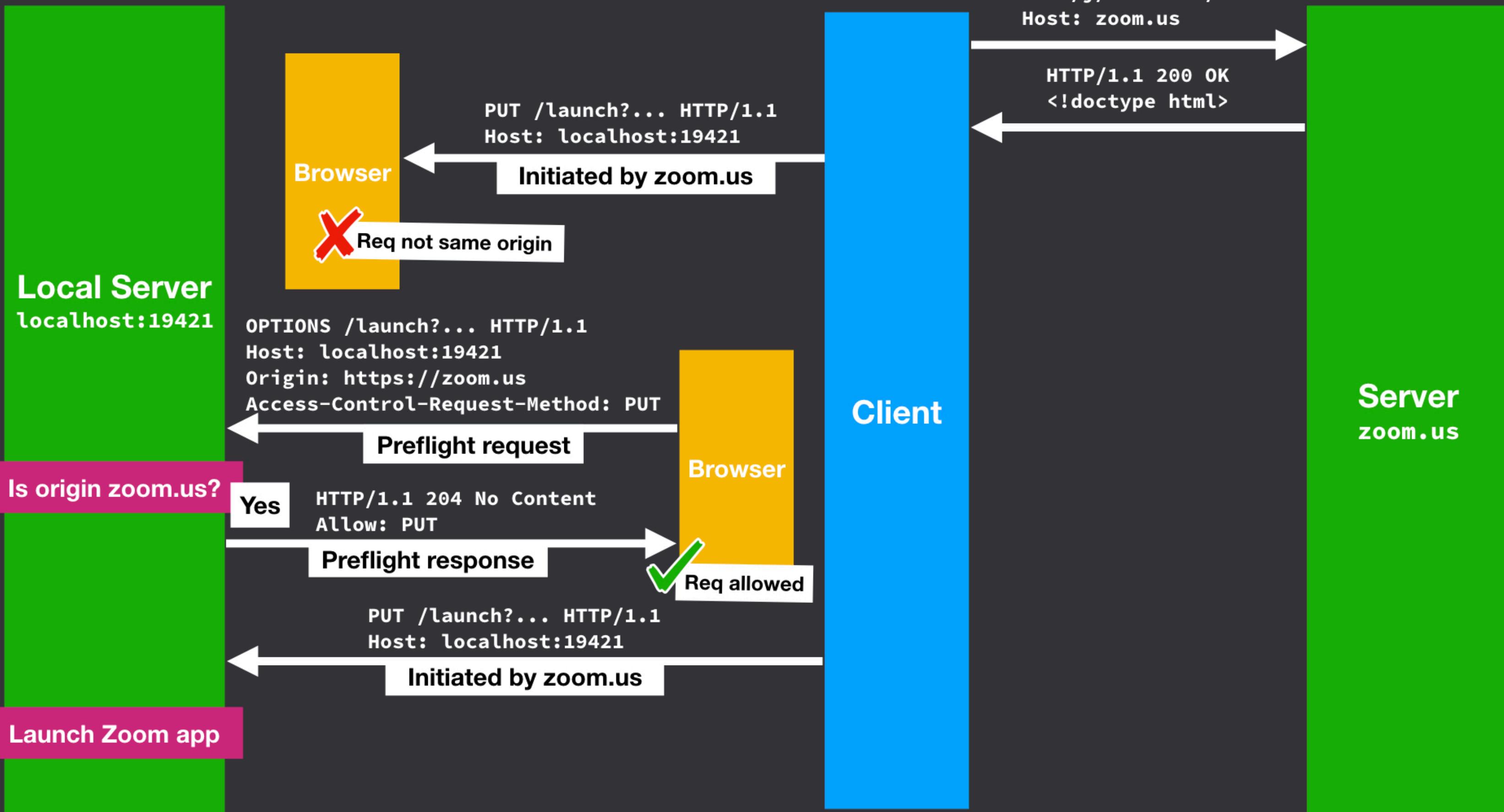


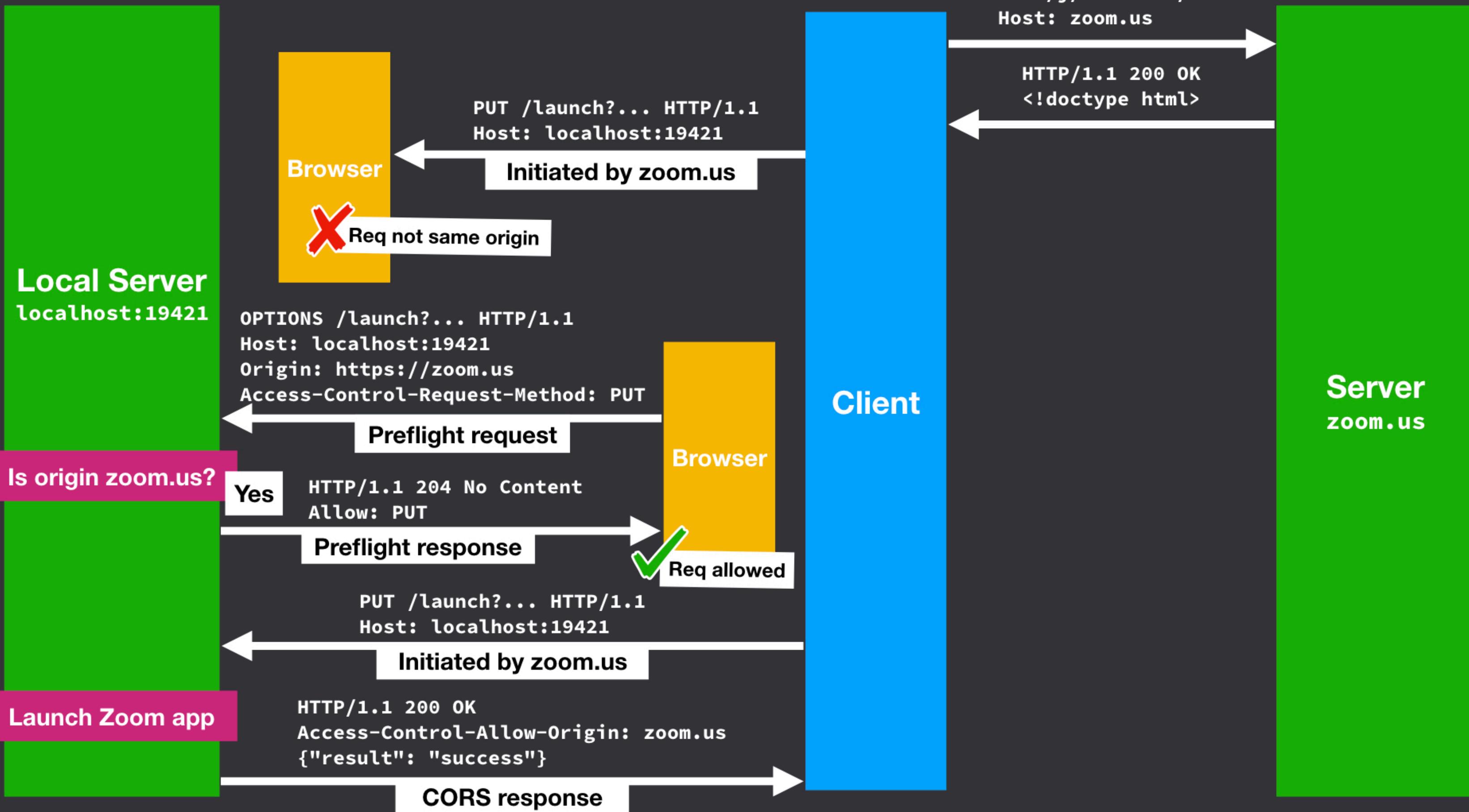


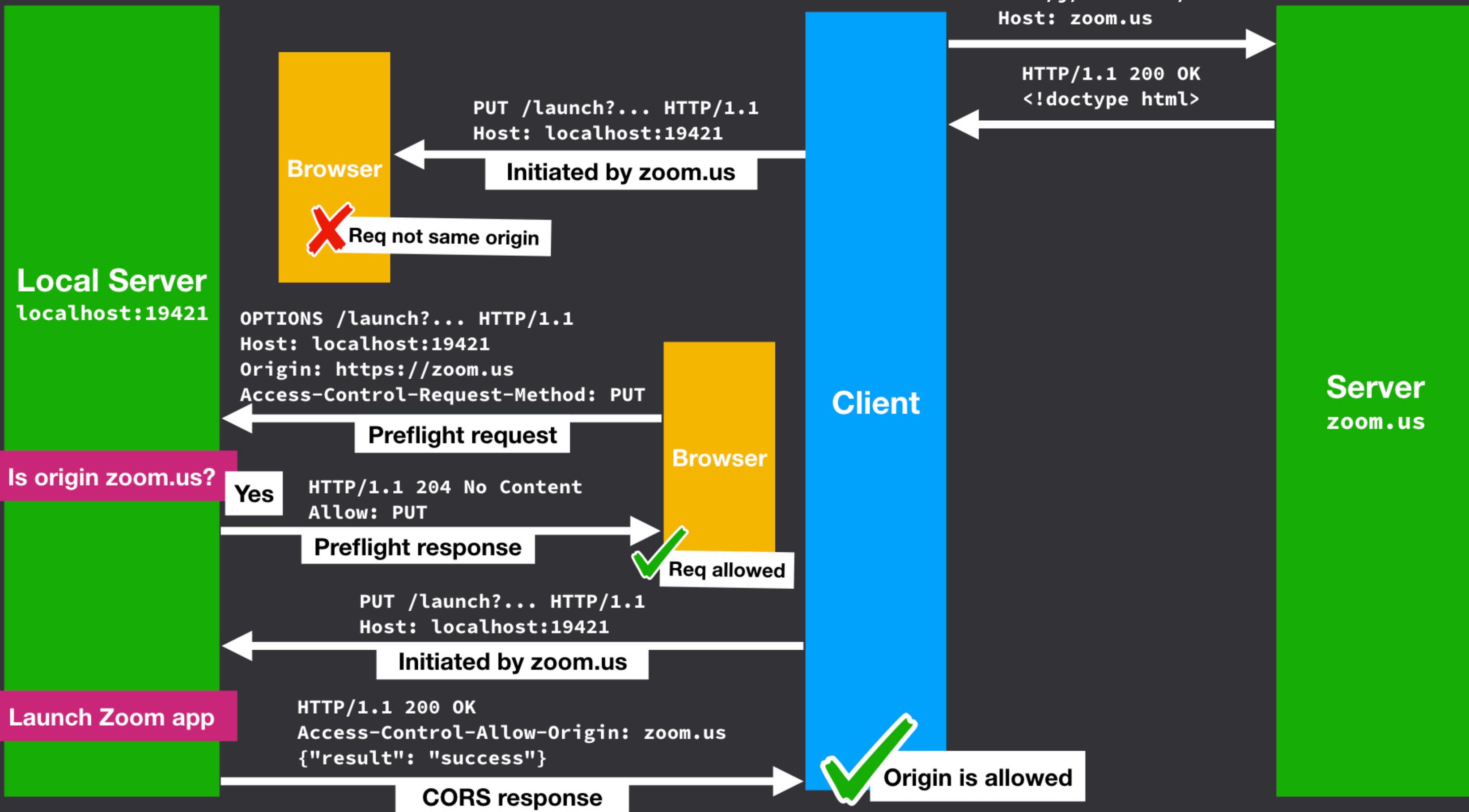












Attacker joins user into a zoom call (local server requires "preflighted" request)

Local Server

localhost:19421

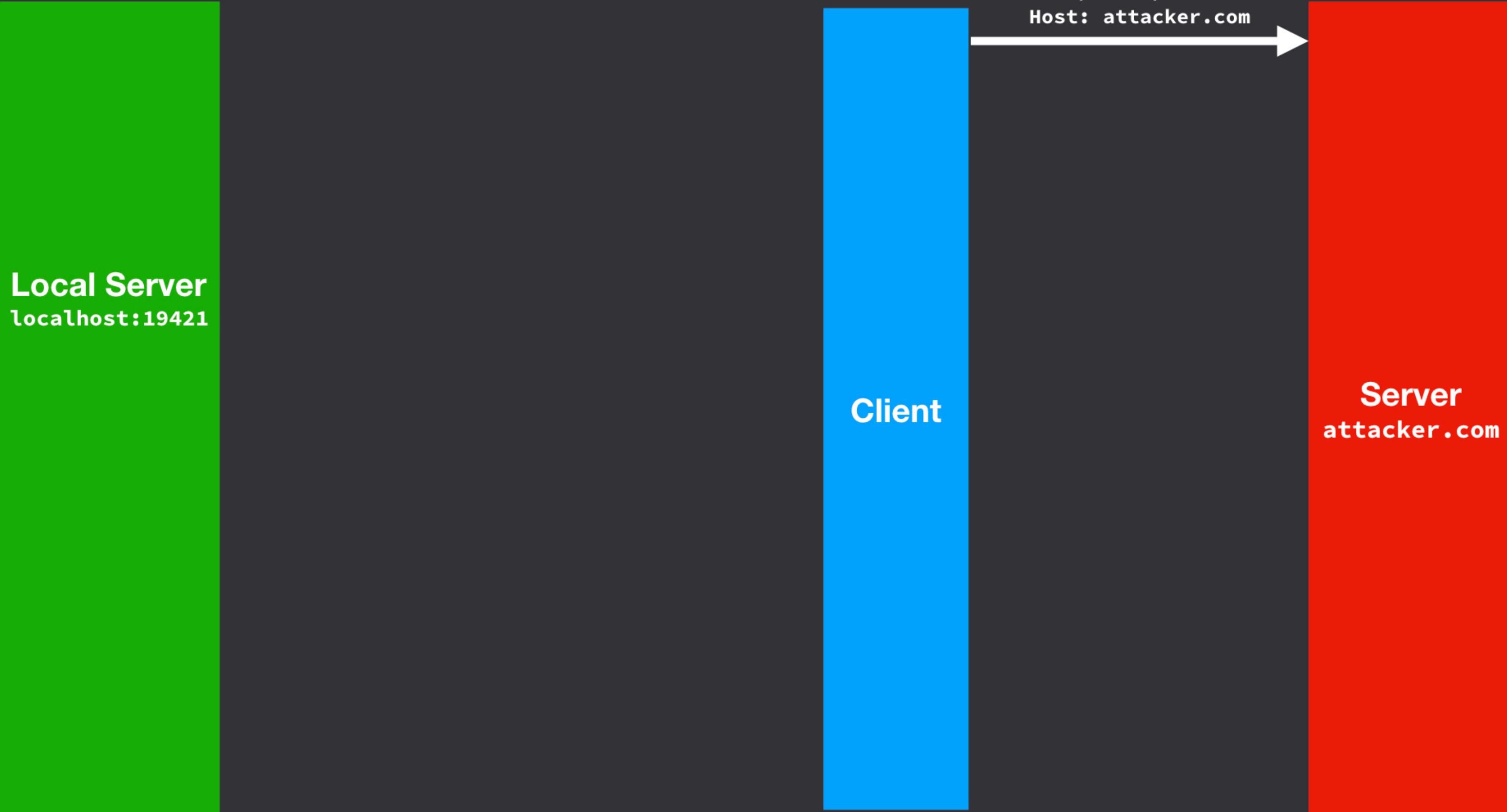
Client

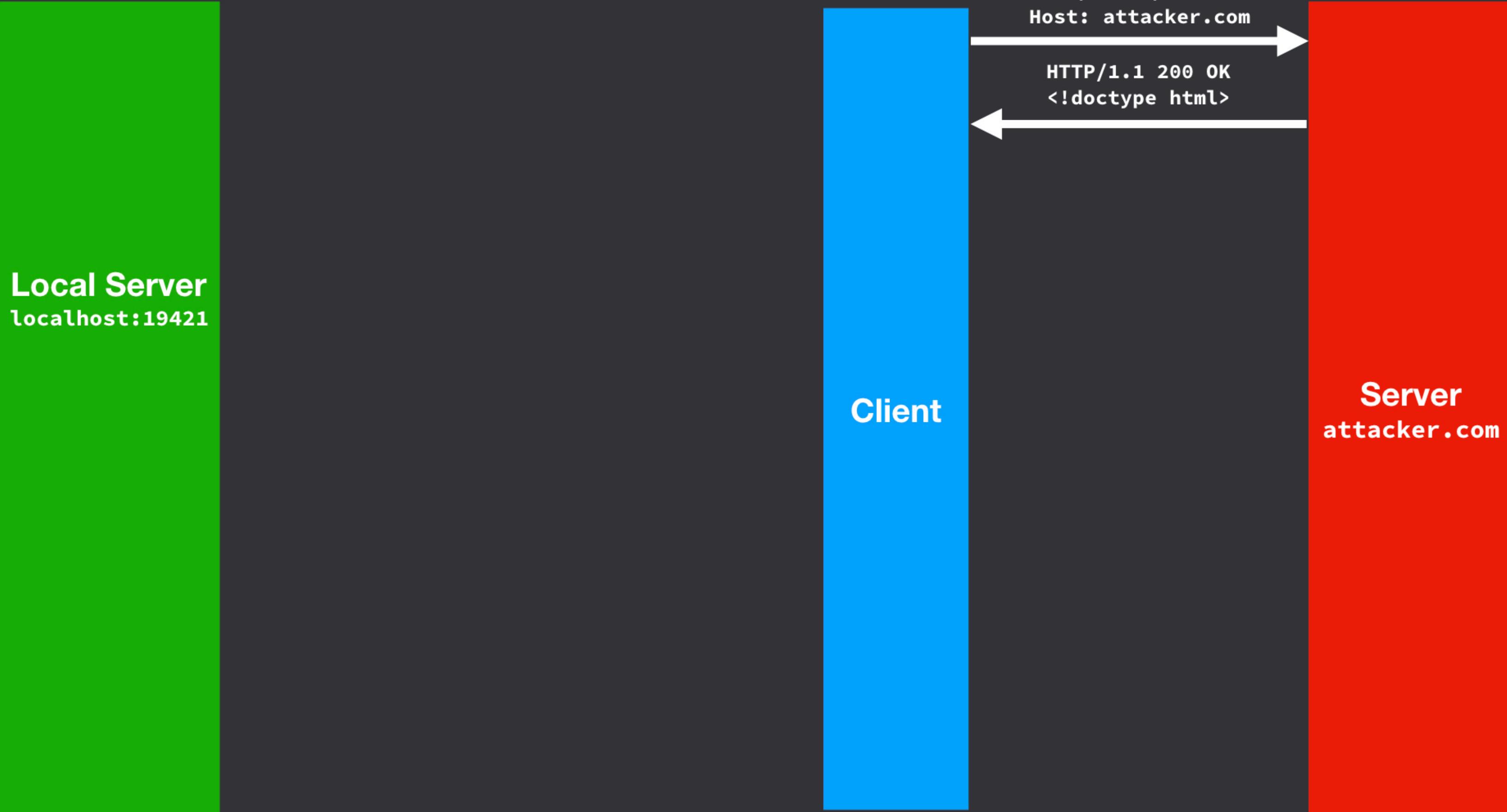
Server
attacker.com

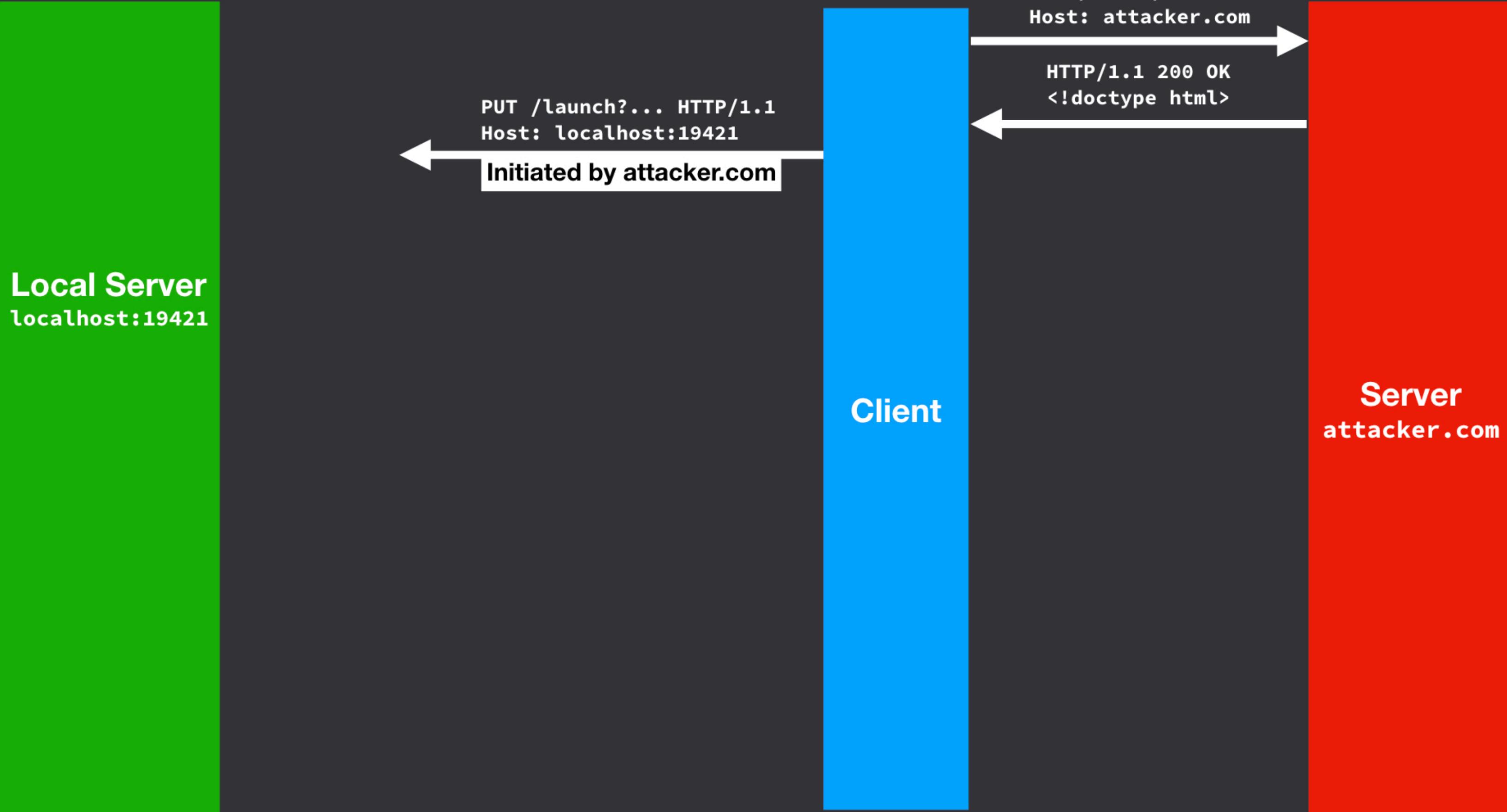
Local Server
localhost:19421

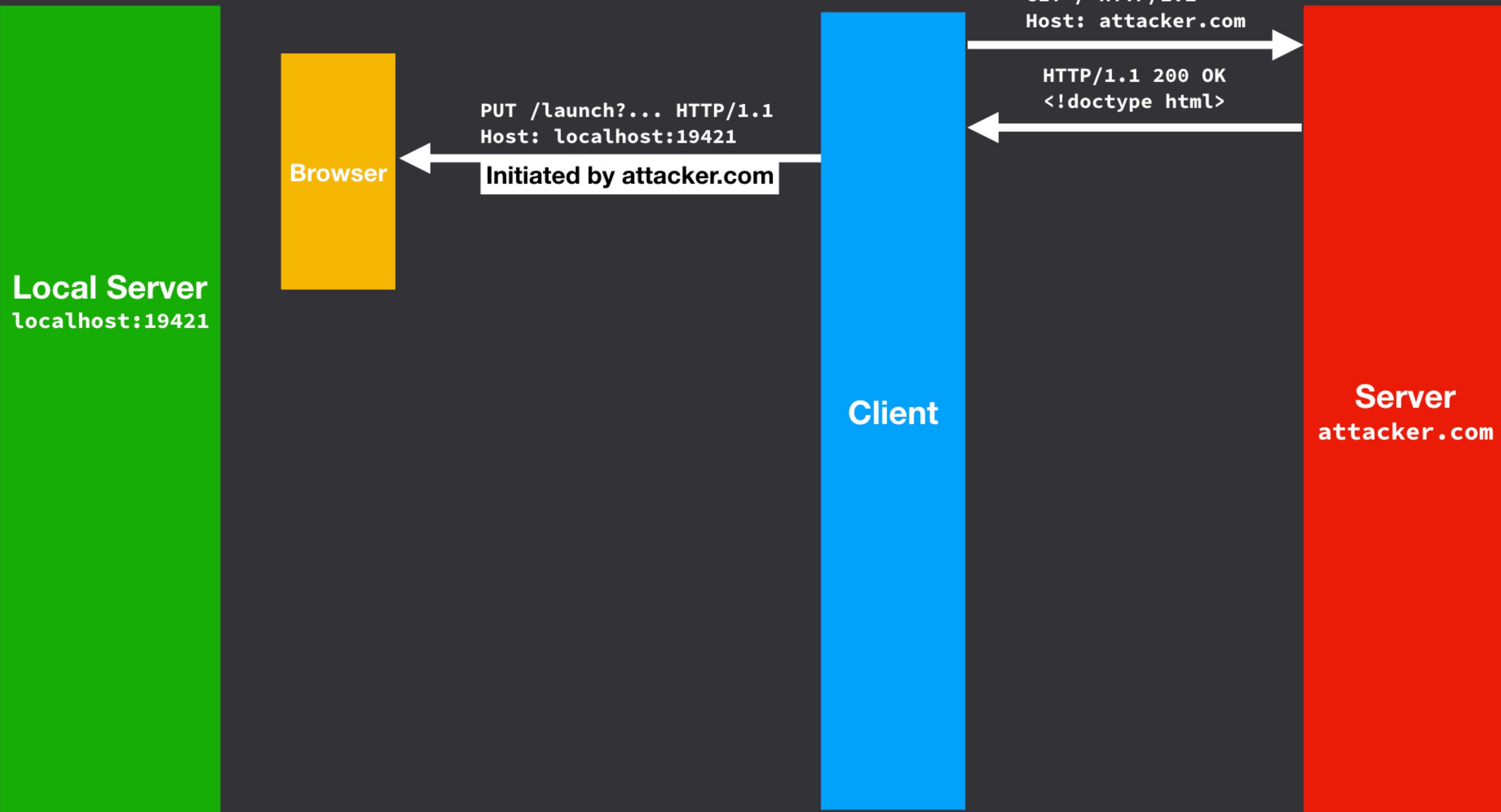
Client

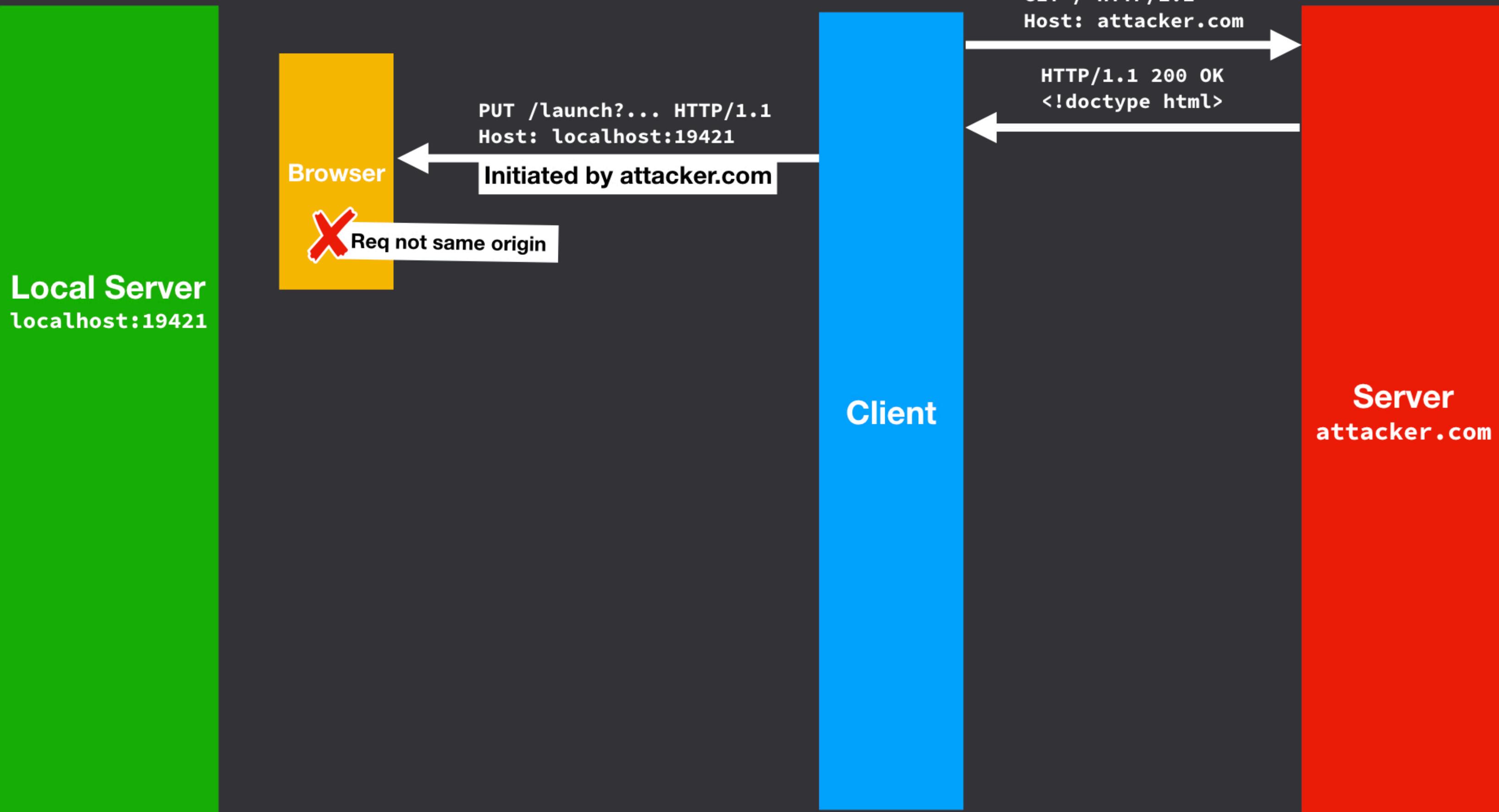
Server
attacker.com

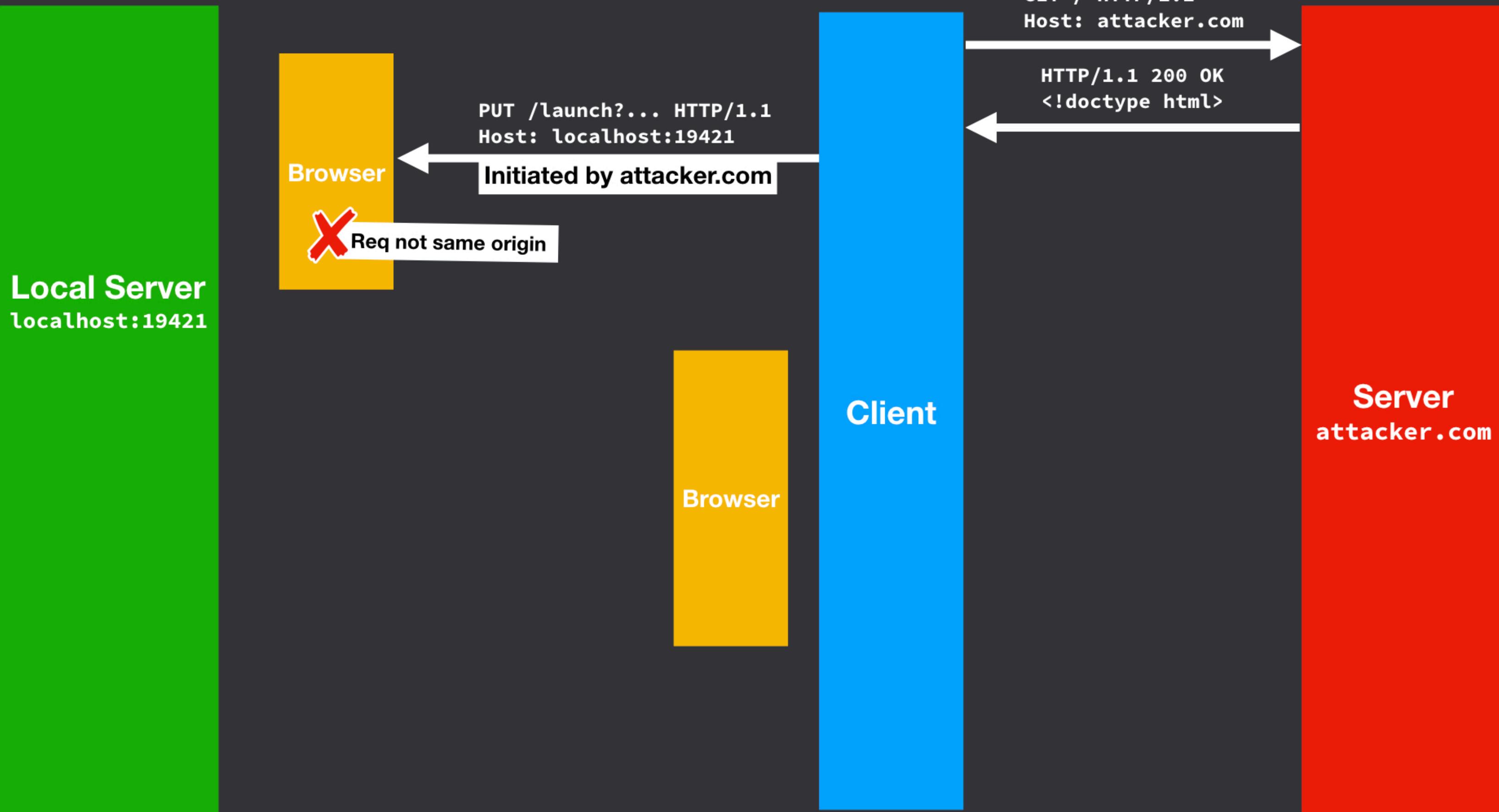


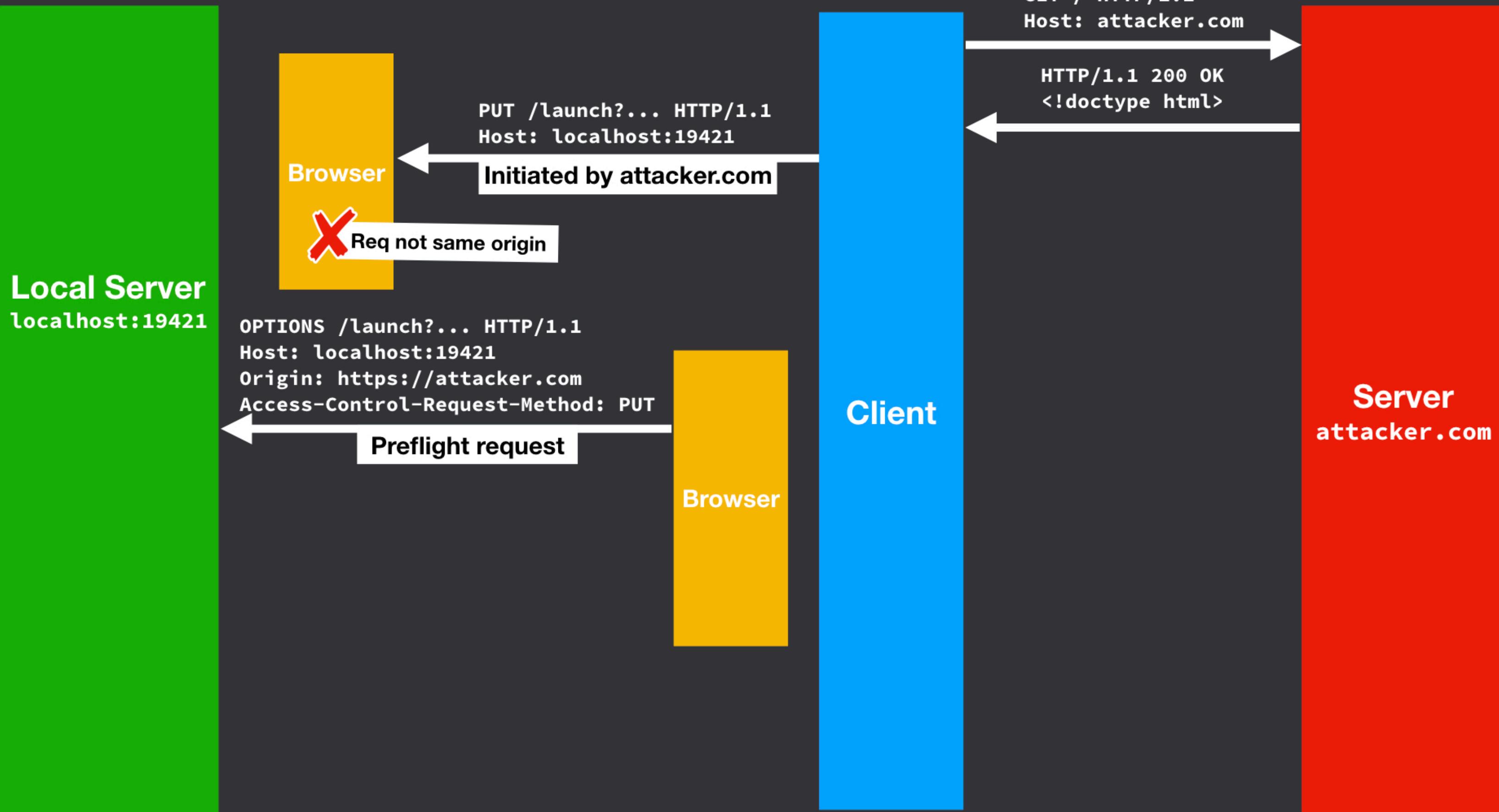


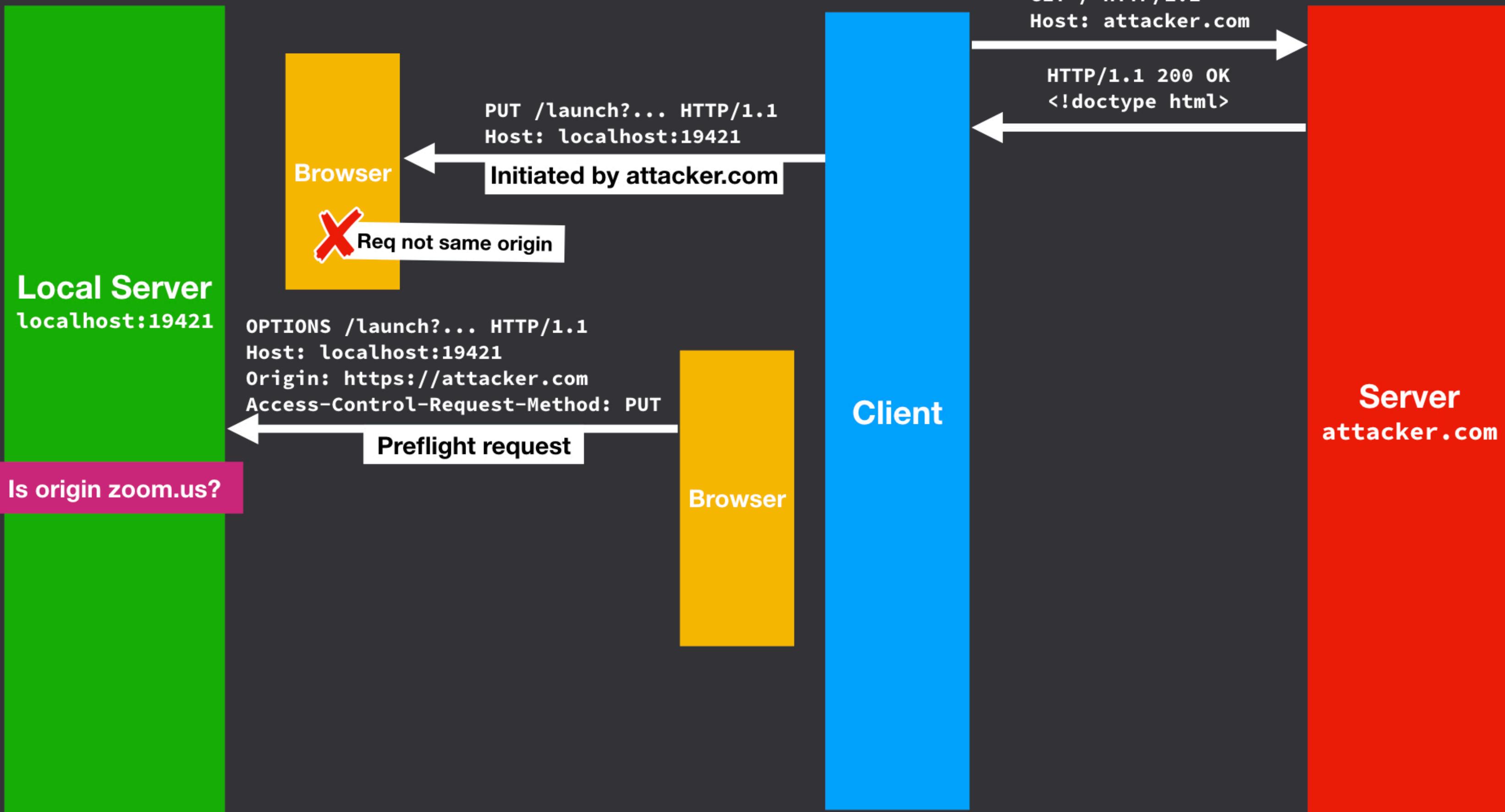


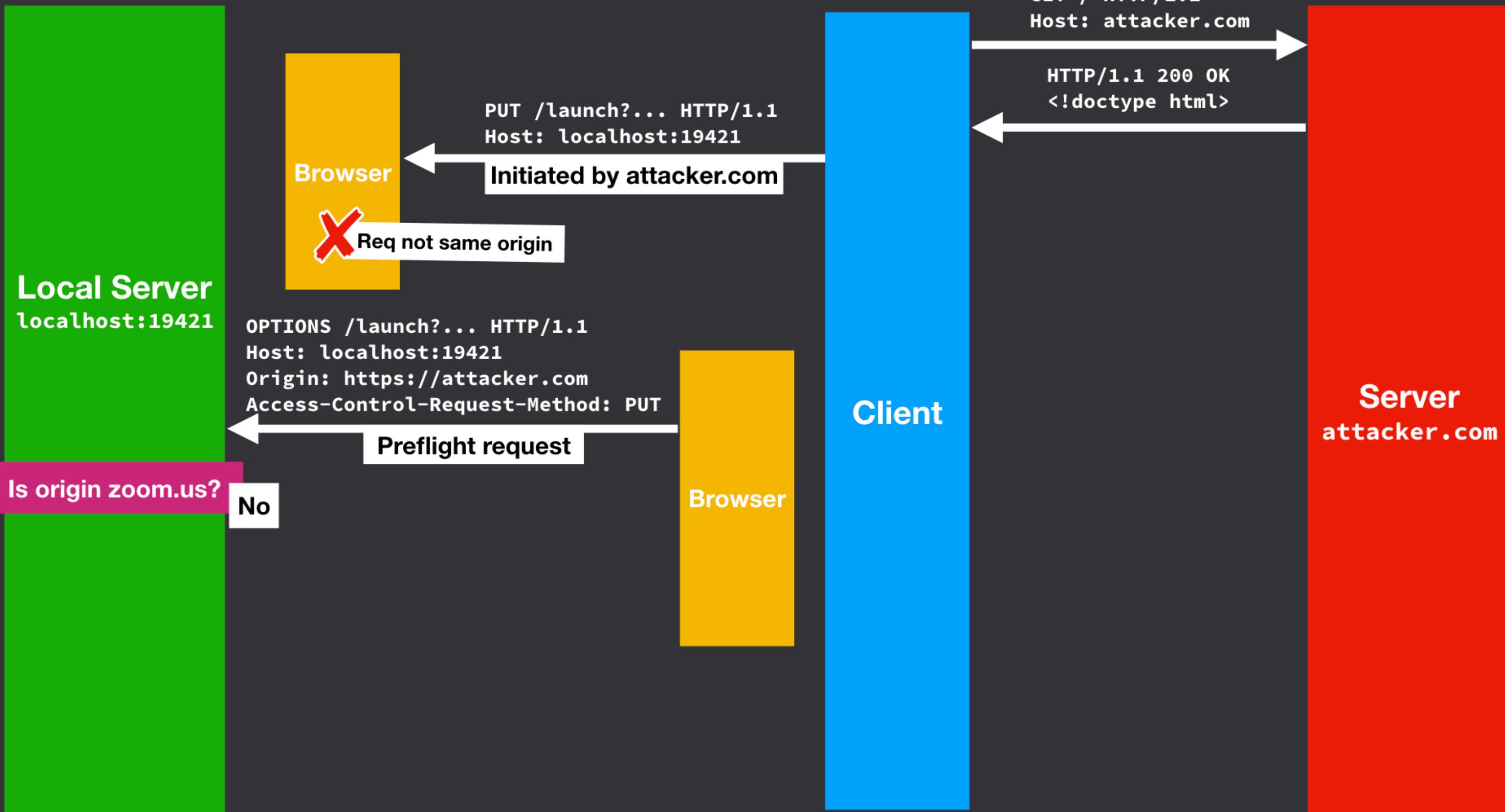


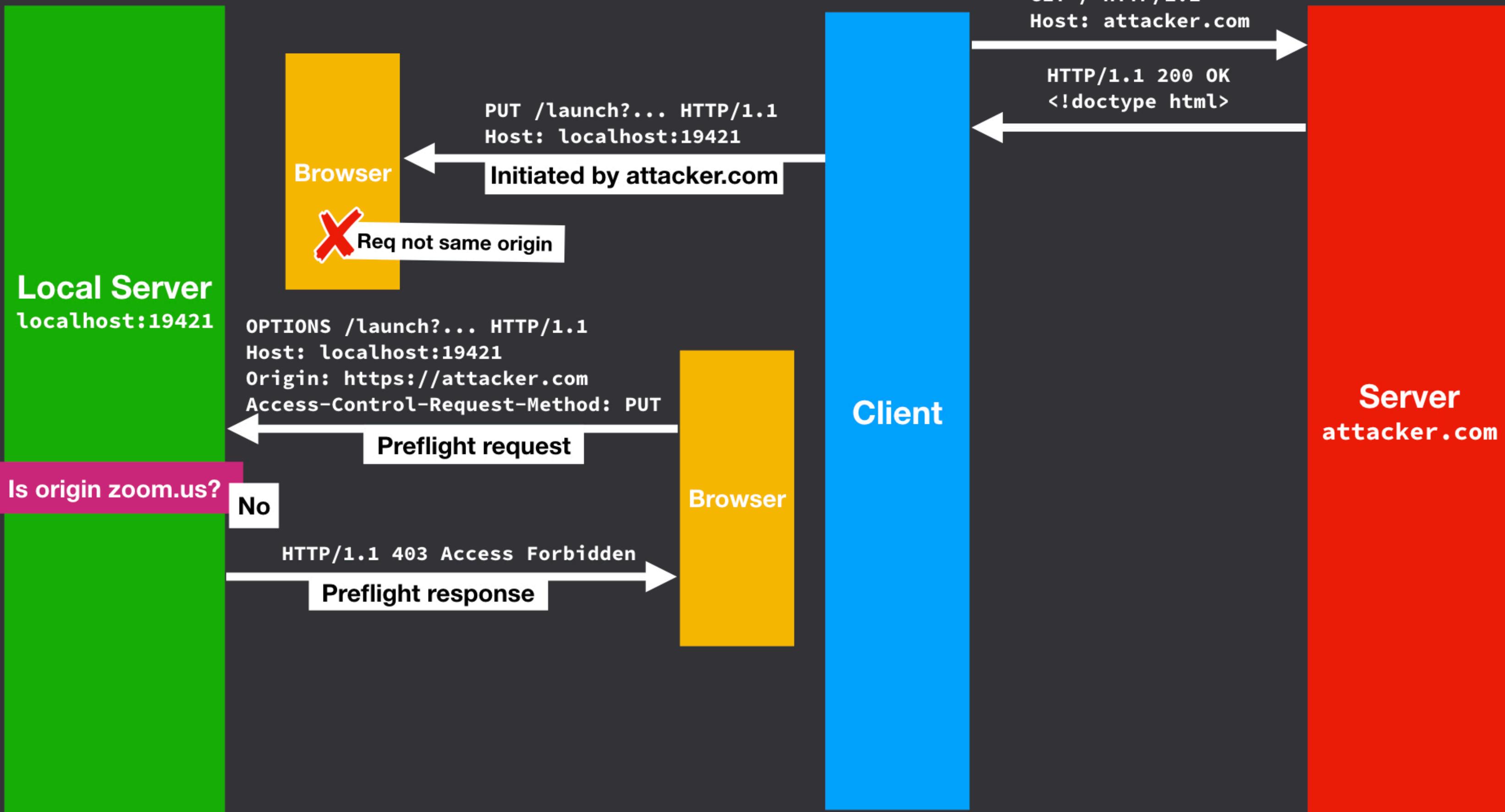


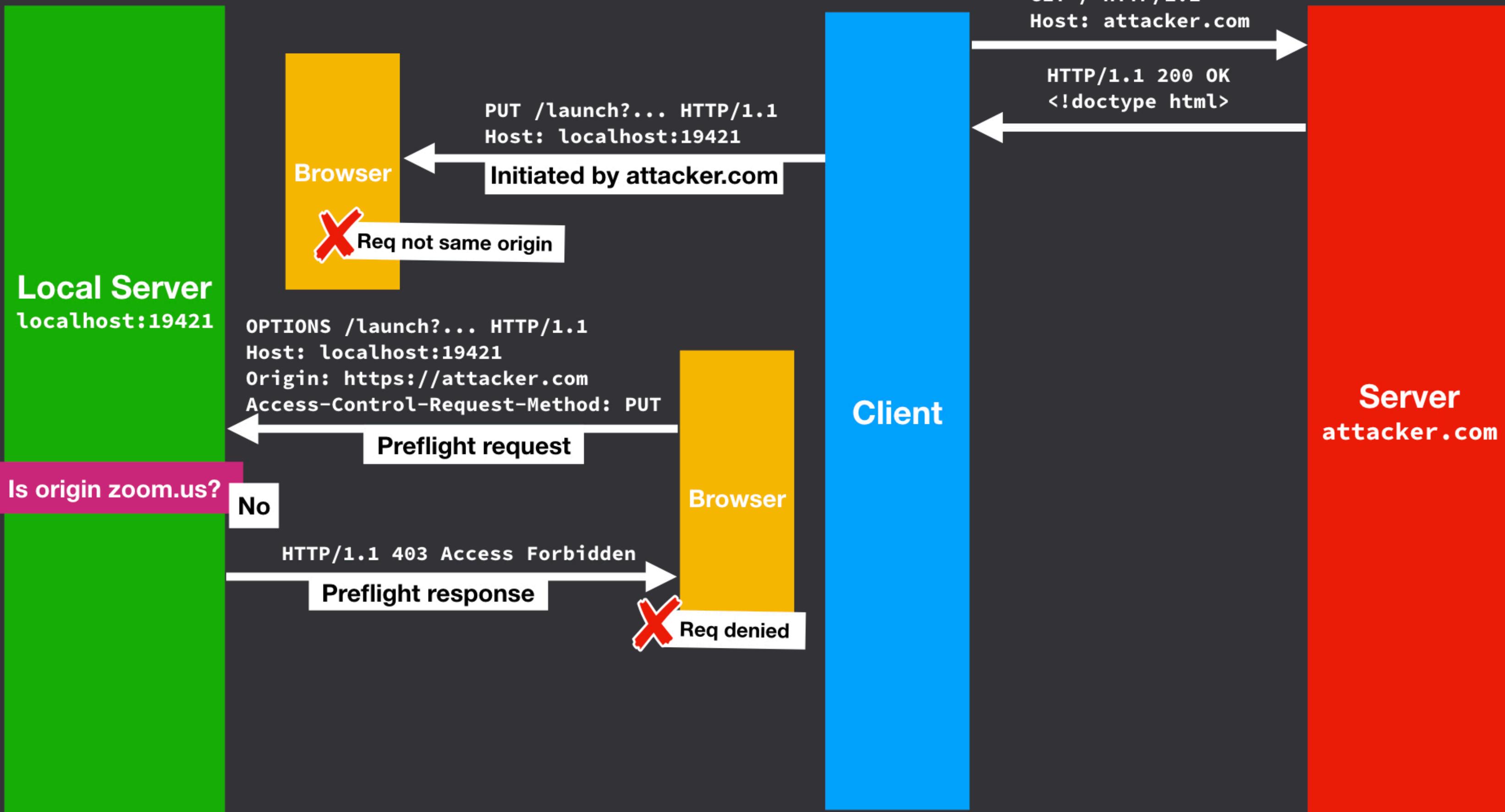












Who can still launch the app from the local server?

- Preflight requests seems to allow the local server to distinguish requests from **zoom.us** and those from random sites
- However, other native apps running on the same device can still fool the local server
 - The browser enforces that sites can't tamper with the **Origin** header, but a native app (e.g. a Node.js or Python script) can make a request and set the **Origin** header to **https://zoom.us**

One more thing....

- Every site on the web can send requests to our local HTTP server!

- Works against the server that required "preflighted" requests as well as the server which just checked the **Origin** header
- Next time... we'll discuss **DNS rebinding attacks** ✨

END

Credits:

<https://medium.com/bugbountywriteup/zoom-zero-day-4-million-webcams-maybe-an-rce-just-get-them-to-visit-your-website-ac75c83f4ef5>

<https://blog.assetnote.io/bug-bounty/2019/07/17/rce-on-zoom/>