

# **WebSockets**

**Yeah, about that whole “stateless” thing...**

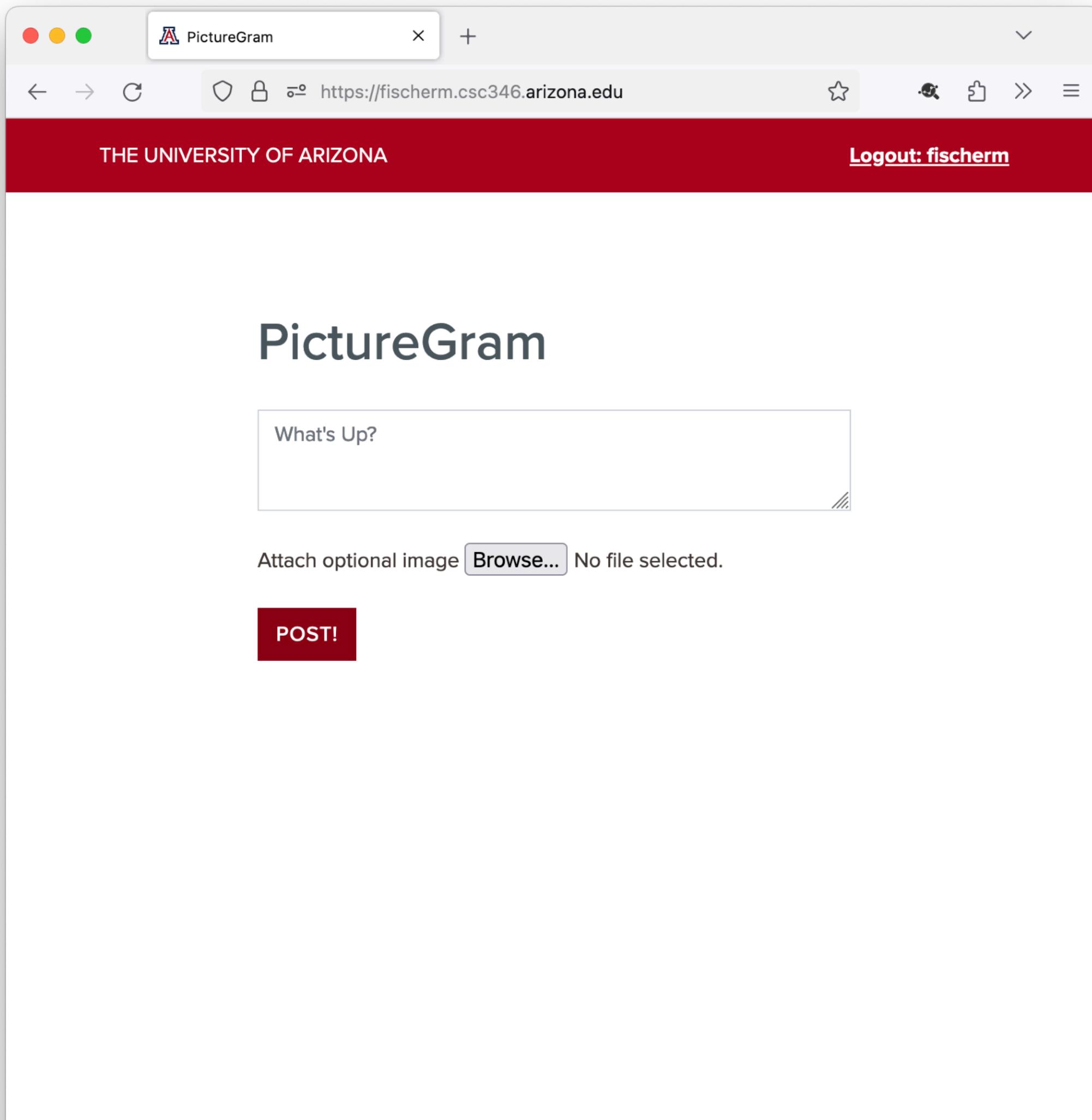
# WebSockets

**Sometimes you just need a constant connection**

- Recall that the HTTP protocol is stateless.
  - Each HTTP request is separate and isolated from any other ones.
  - We've repeated this more than a few times this semester 😜
- What are some of the use cases where a stateless network model starts to fail?

# Chat

## How does our Chat App get new chat messages?



PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

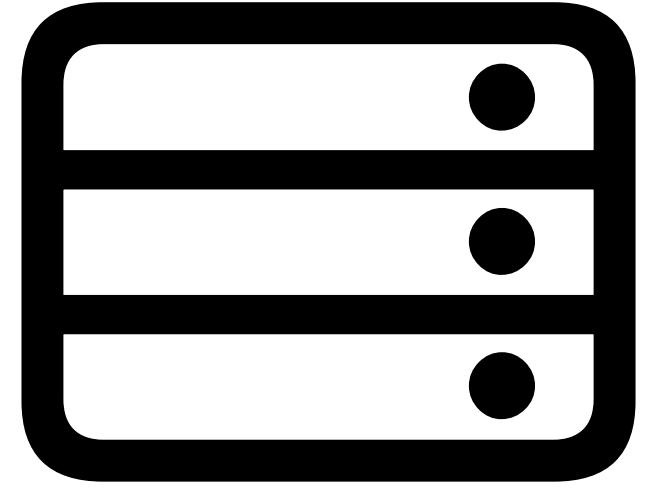
PictureGram

What's Up?

Attach optional image  No file selected.

https://fischerm.csc346.arizona.edu

On Page Load:  
Request New Posts Messages



api.csc346.arizona.edu

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

API Responds with new post JSON data

The diagram illustrates the communication flow between the client application and the API. On the right, a black rounded rectangle labeled "api.csc346.arizona.edu" contains a night photograph of a stadium with blue lights and a large "A" logo. An arrow points from this box to a white rectangular box on the left labeled "PictureGram". Inside the "PictureGram" box, there is a text input field containing "What's Up?", a file upload section with "No file selected.", and a red "POST!" button. A callout box above the "POST!" button contains the text "API Responds with new post JSON data".

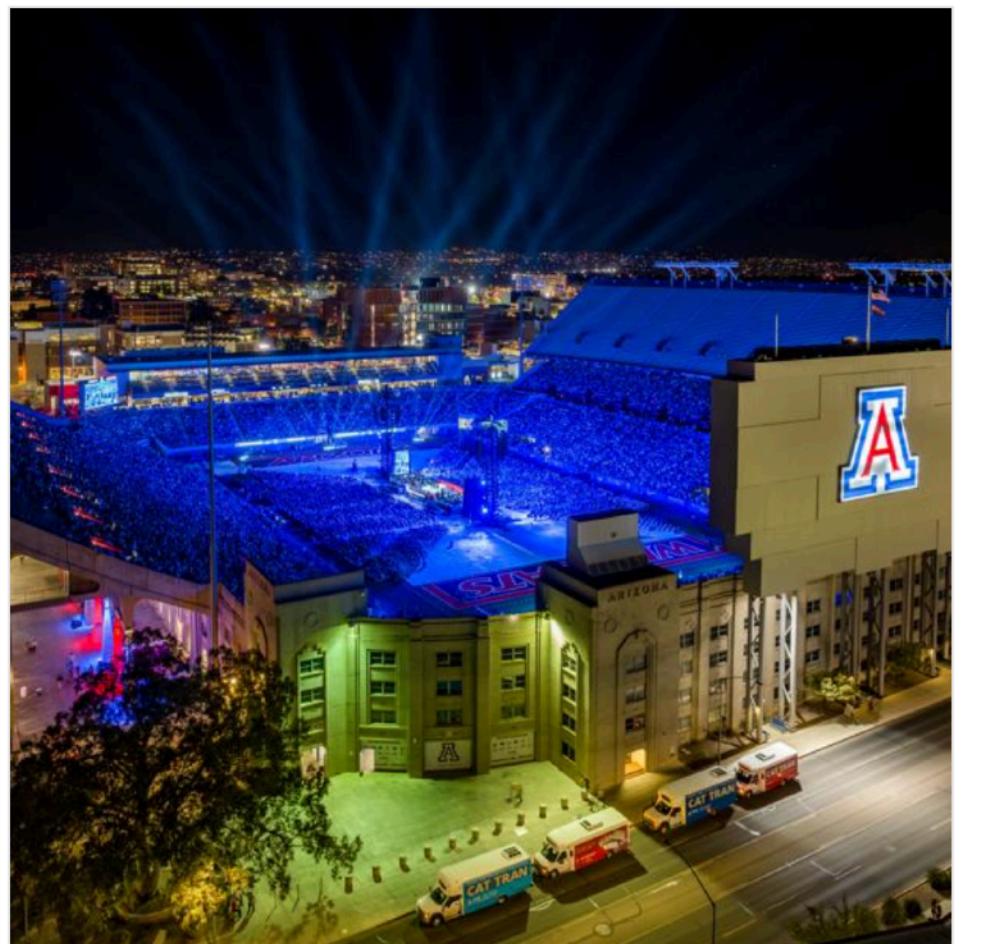


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

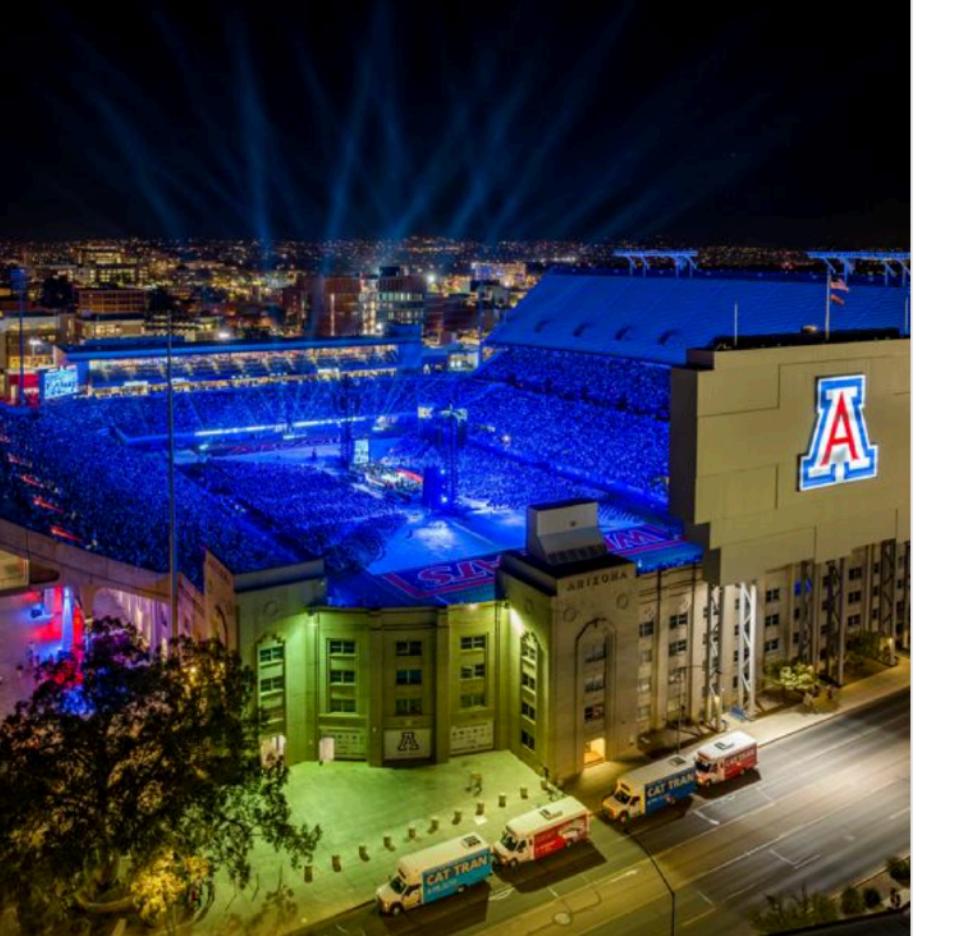
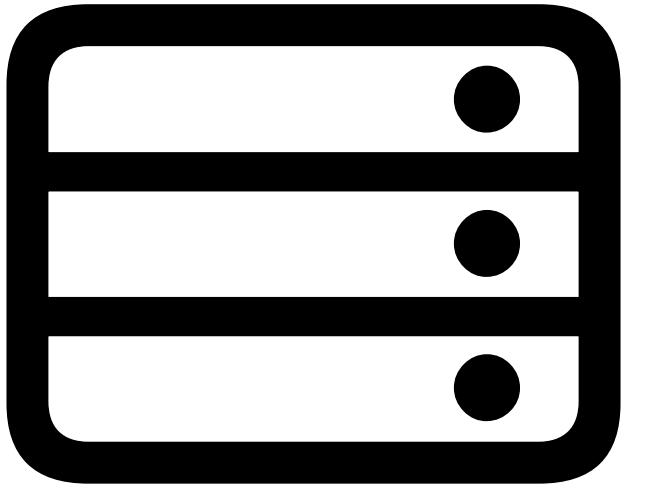


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)



api.csc346.arizona.edu

A PictureGram

THE UNIVERSITY OF ARIZONA

Logout: fischerm

# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

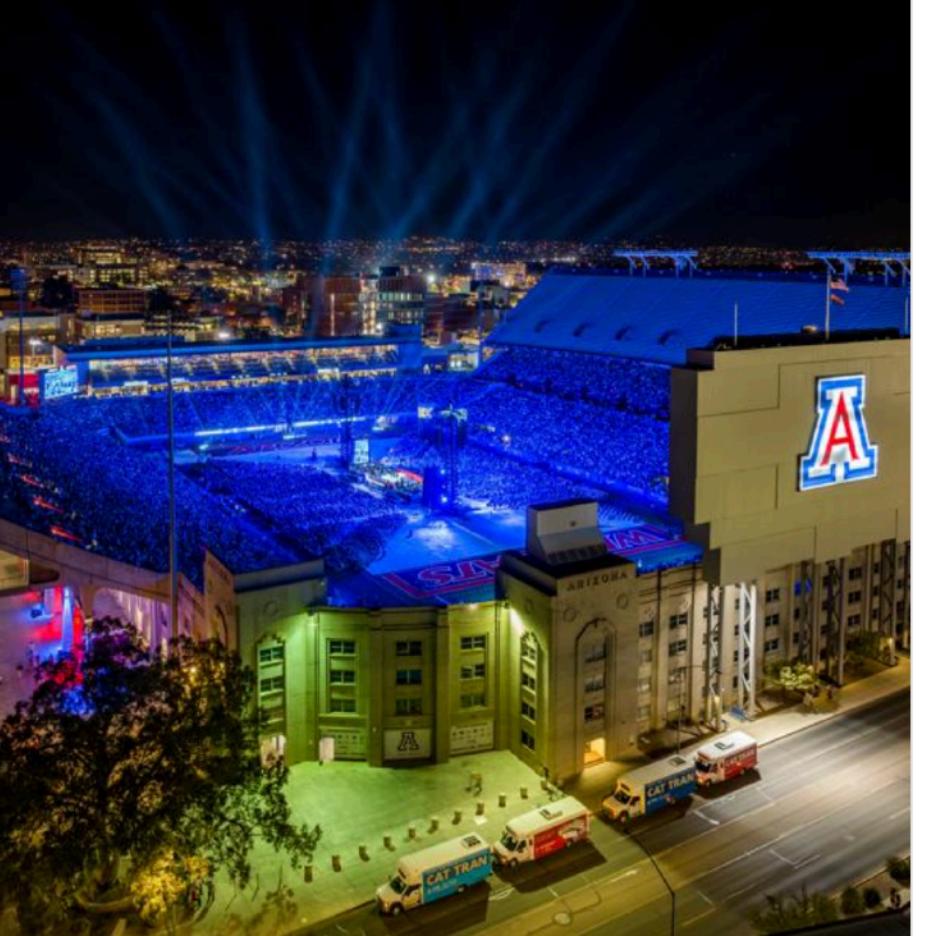
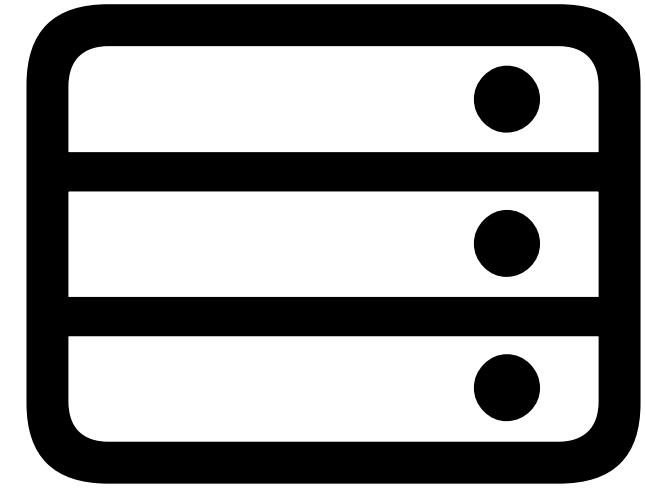
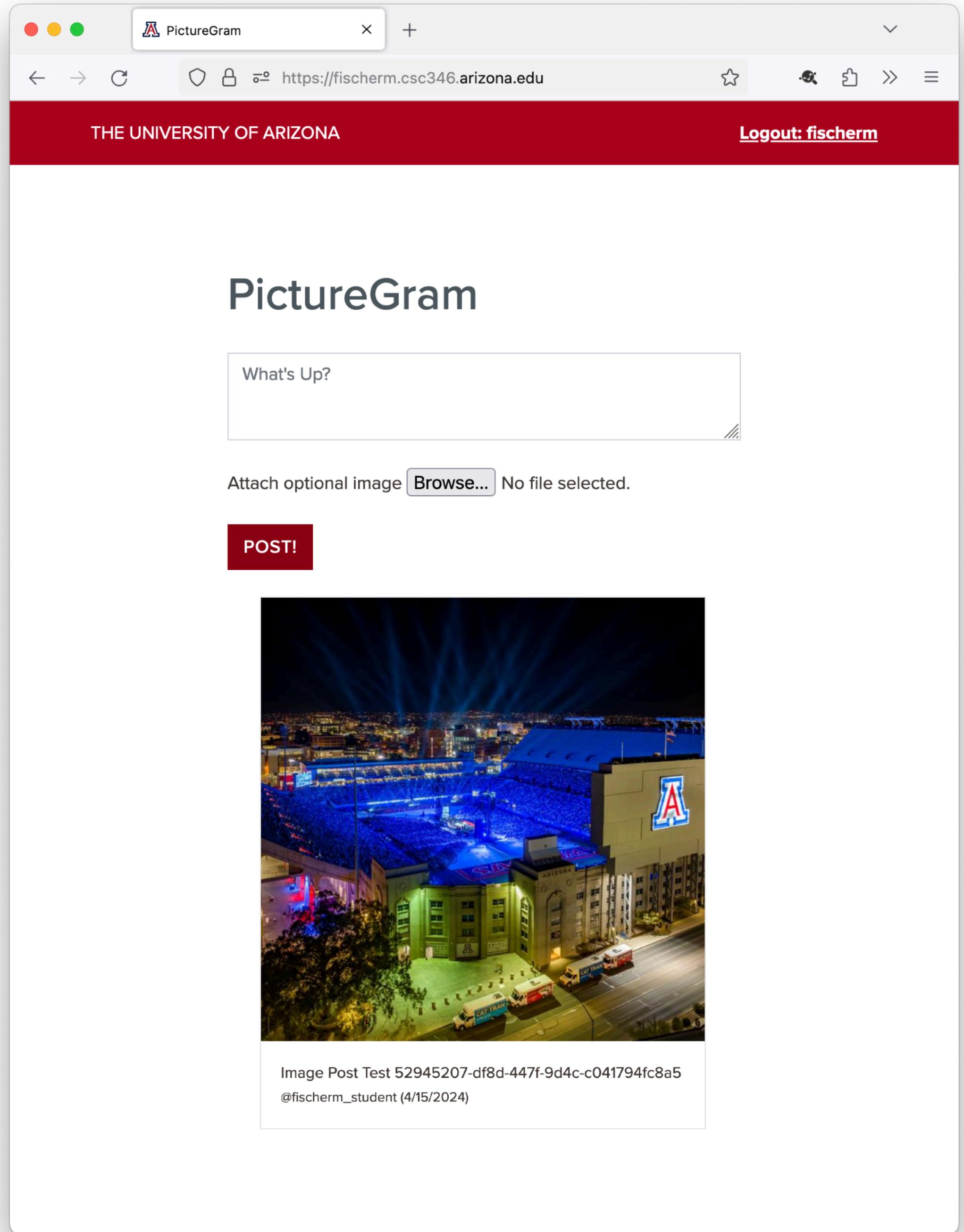


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

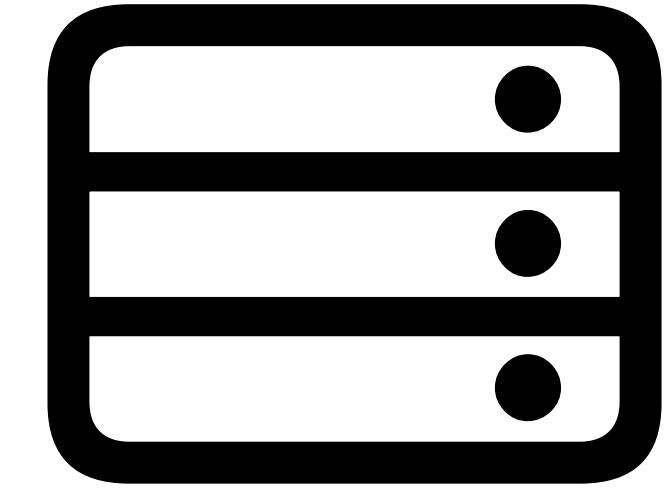


api.csc346.arizona.edu

- Now What?
- If new messages are posted by someone else, how does this browser get them?
- Currently you have to reload the page

A screenshot of a web browser window titled "PictureGram". The address bar shows the URL "https://fischerm.csc346.arizona.edu". The page header includes "THE UNIVERSITY OF ARIZONA" and a "Logout: fischerm" link. The main content area is titled "PictureGram" and contains a text input field with "What's Up?", a file upload section with "Attach optional image" and a "Browse..." button (showing "No file selected"), and a red "POST!" button. Below these is a large image of a stadium at night with blue lights and the University of Arizona logo on a building. At the bottom of the image is a caption: "Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5" and "@fischerm\_student (4/15/2024)".

# Polling



api.csc346.arizona.edu

- A common approach is known as polling
- The browser checks with the API on a timer and asks for new chat messages

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

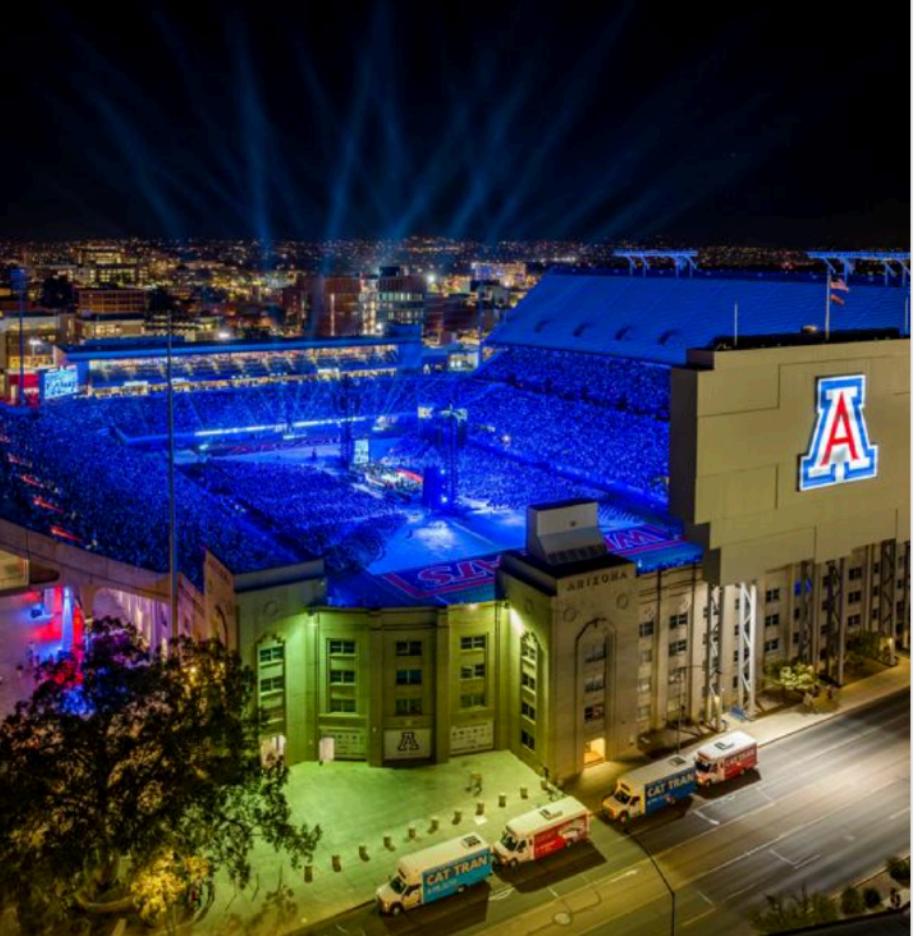
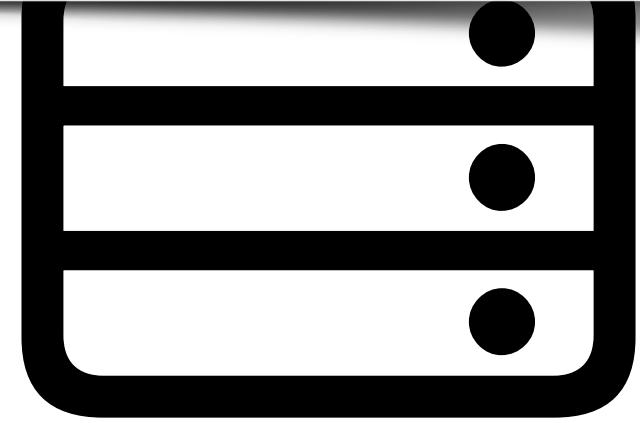


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

```
setTimeout( () => {  
    loadPosts(newestPostTimestamp, null)  
, 10000)
```



api.csc346.arizona.edu

- The `setTimeout()` function will call the first argument at an interval specified in the second argument
- Here the second argument is 10000ms, so every 10 seconds the `loadChats(...)` function is called

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

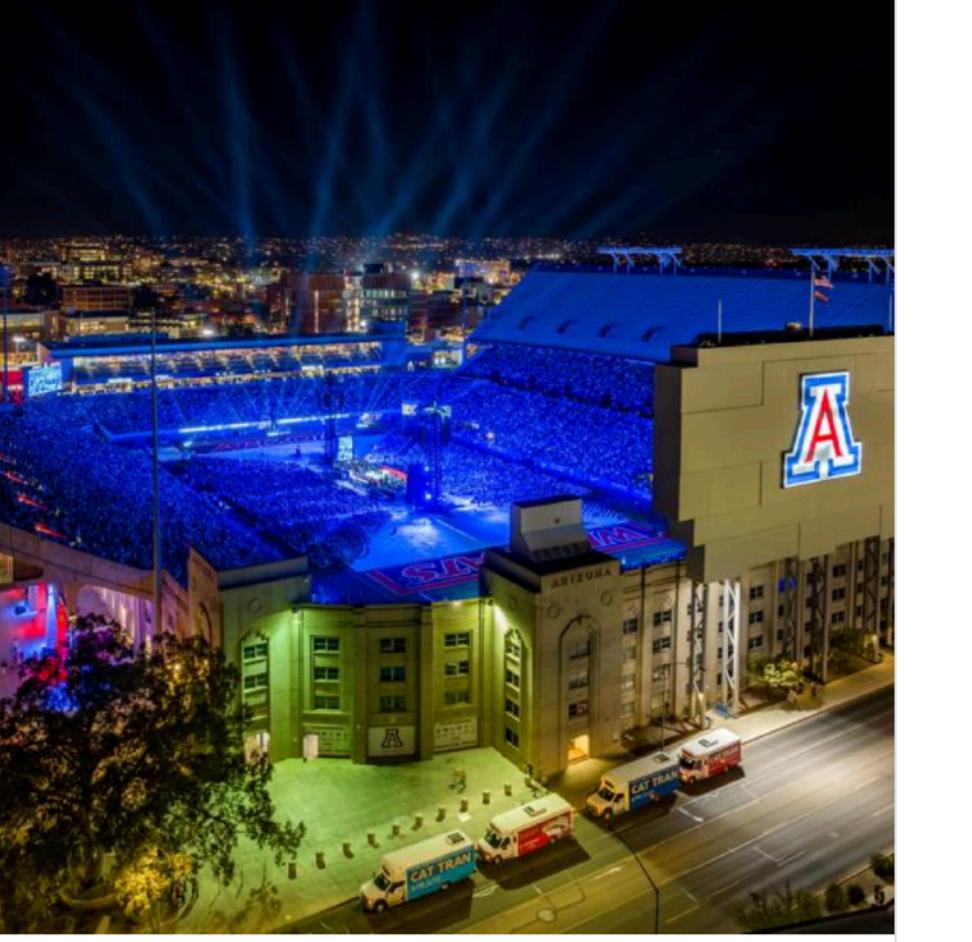
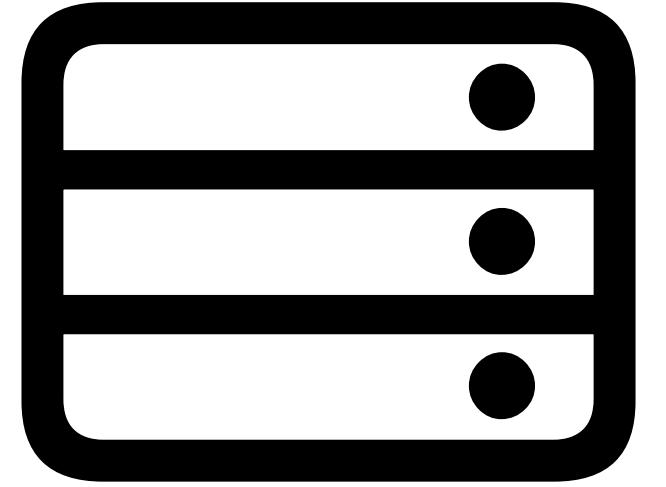


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

“Got any new posts?”



api.csc346.arizona.edu

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

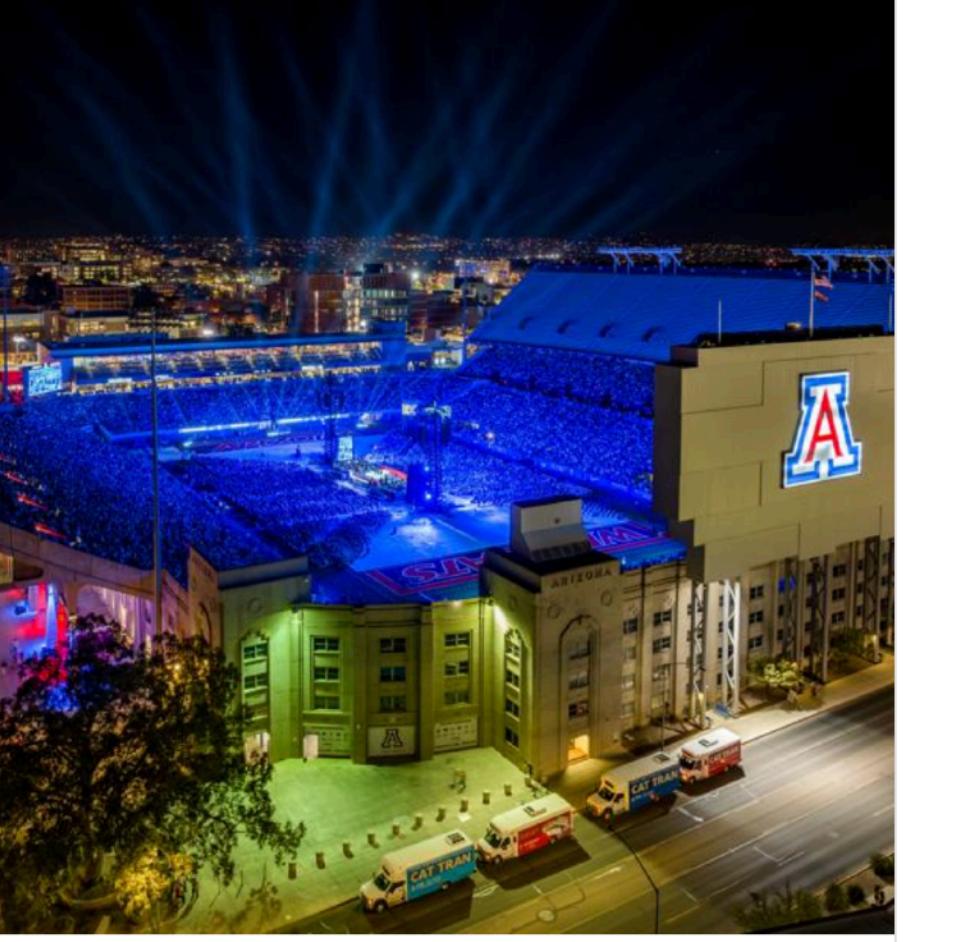


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)



**10  
seconds  
later**

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

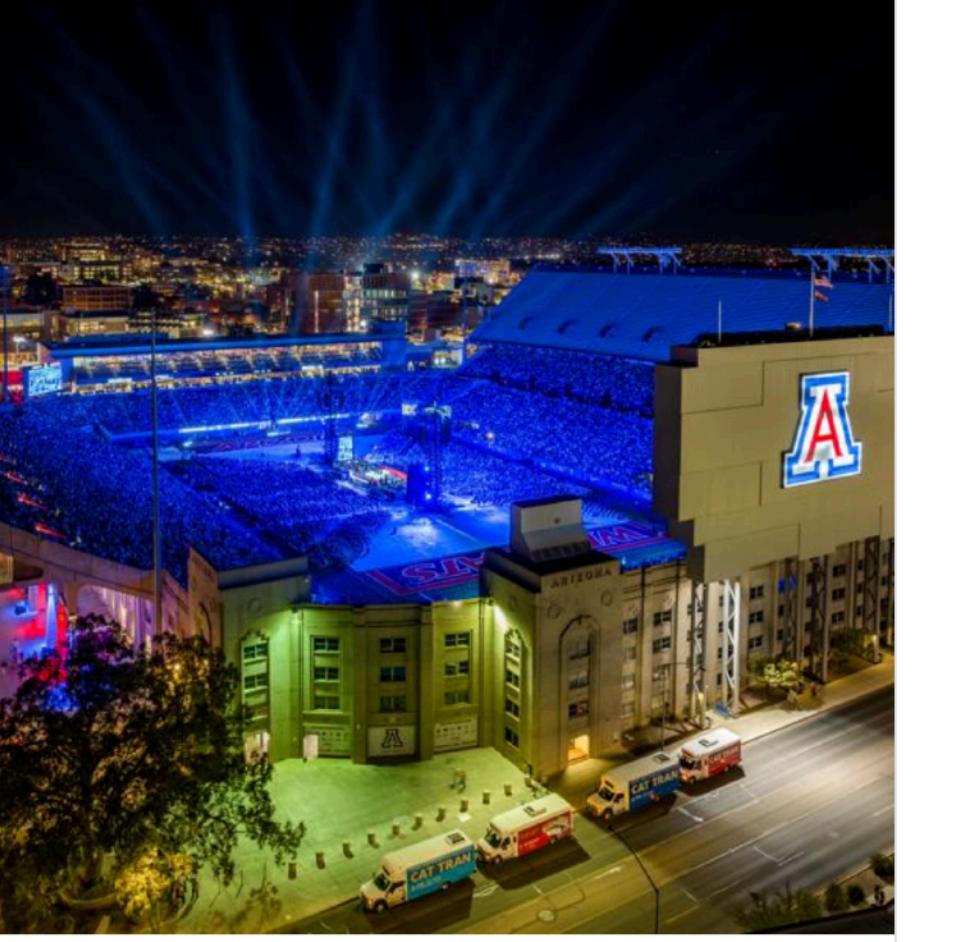
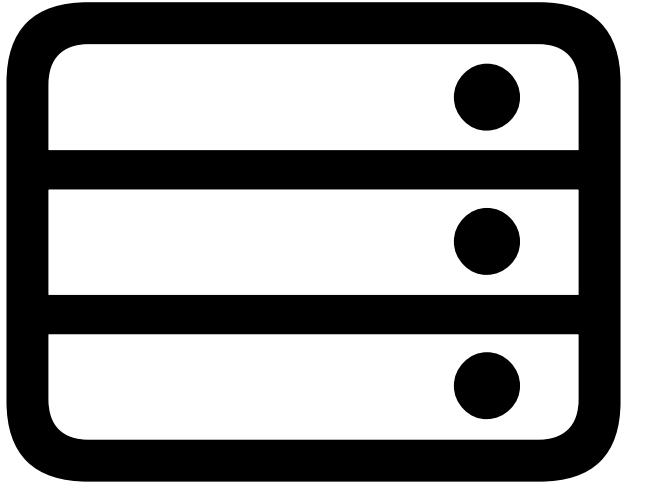


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

“Got any new posts?”



api.csc346.arizona.edu

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

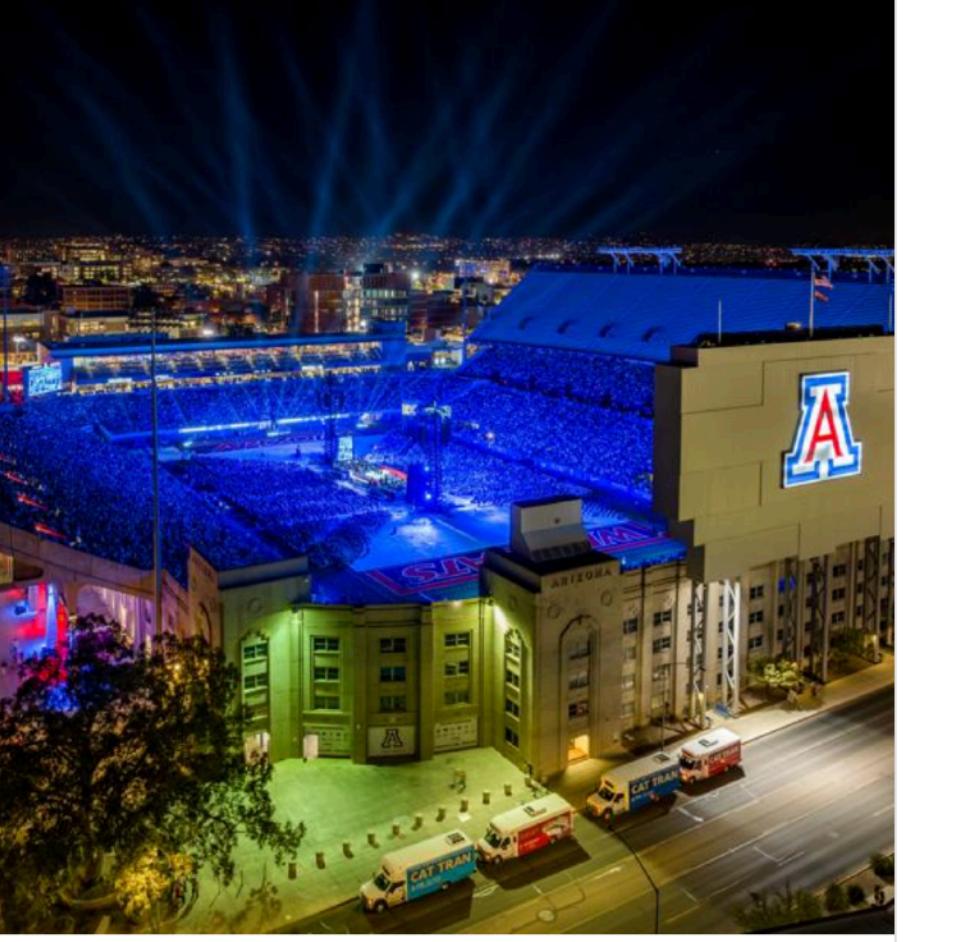


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)



**10  
seconds  
later**

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

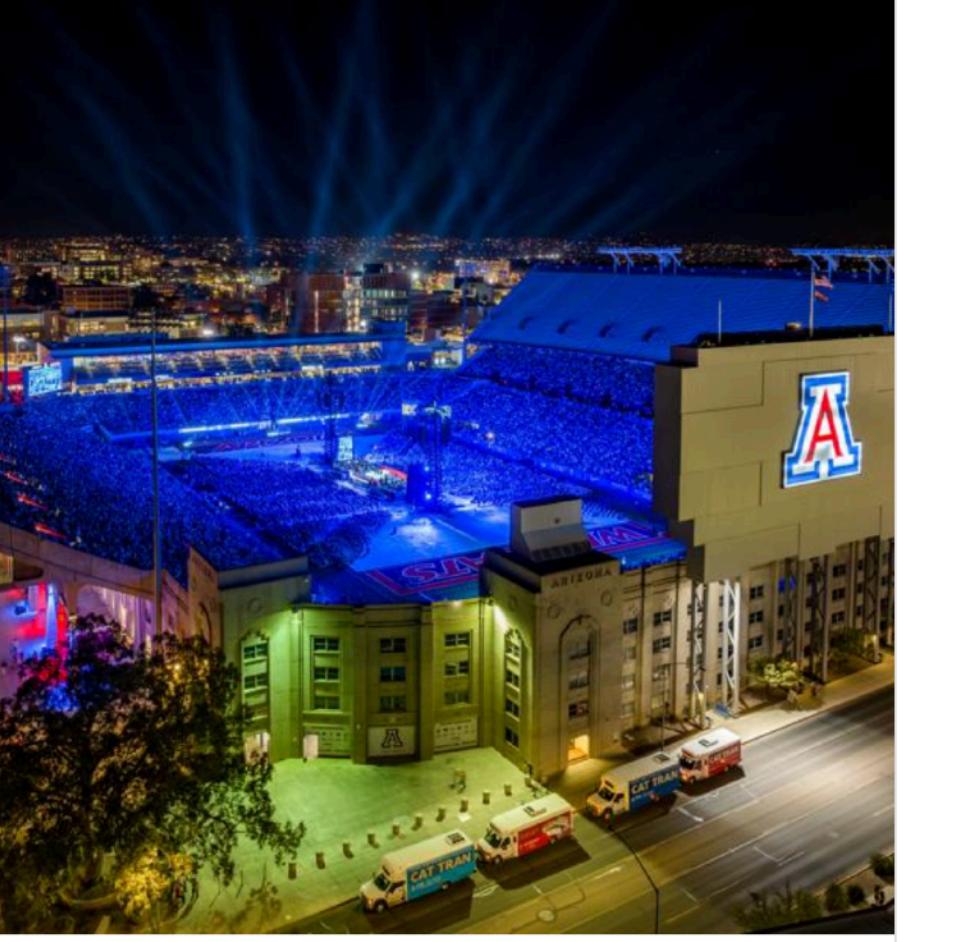
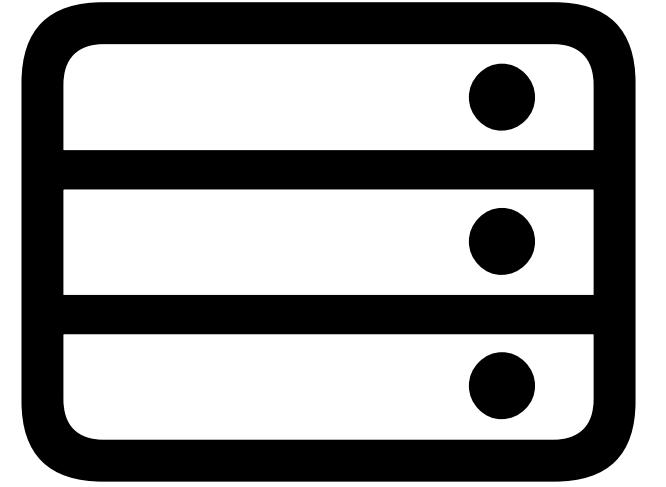


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

“Got any new posts?”



api.csc346.arizona.edu

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.

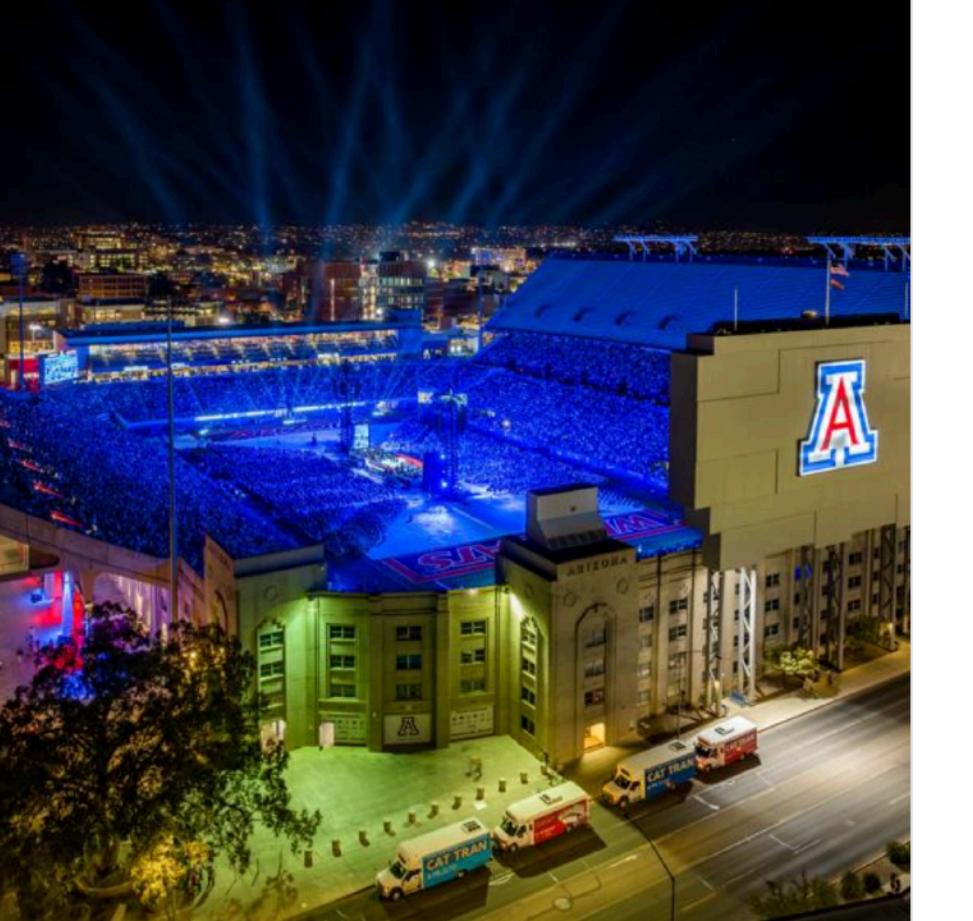


Image Post Test 52945207-df8d-447f-9d4c-c041794fc8a5  
@fischerm\_student (4/15/2024)

Logout: fischerm

api.csc346.arizona.edu

Hey I do! Here you go



Image Post Test da834b5e-1039-47c7-9e79-260ef8aafed28  
@fischerm\_student (4/15/2024)

PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

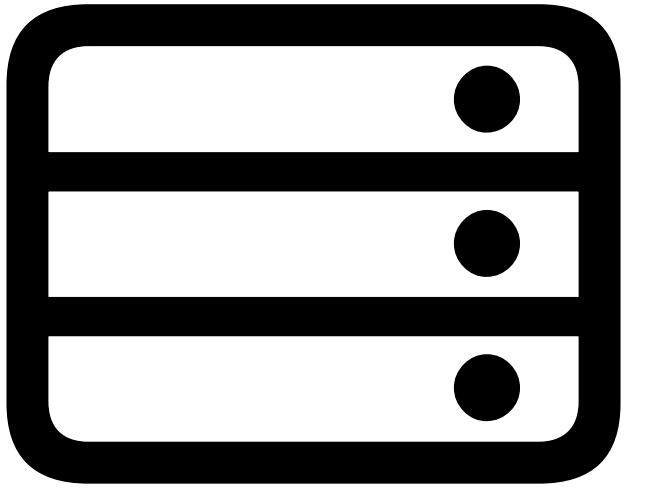
What's Up?

Attach optional image  No file selected.

**POST!**



Image Post Test  
da834b5e-1039-47c7-9e79-260ef8aaf28  
@fischerm\_student (4/15/2024)



api.csc346.arizona.edu

A PictureGram

THE UNIVERSITY OF ARIZONA

Logout: fischerm

## PictureGram

What's Up?

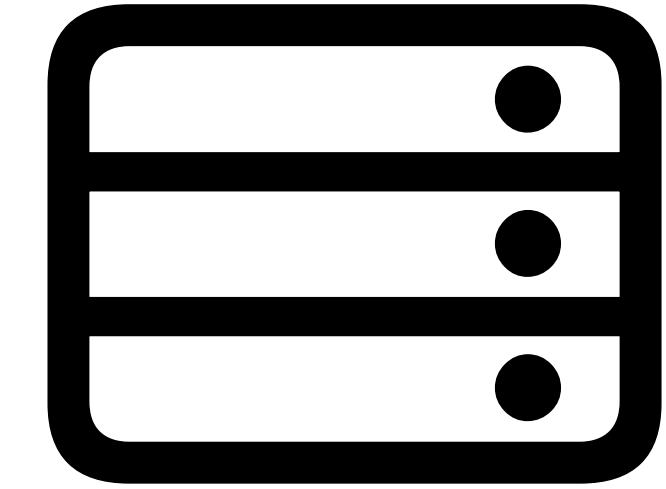
Attach optional image  No file selected.

**POST!**



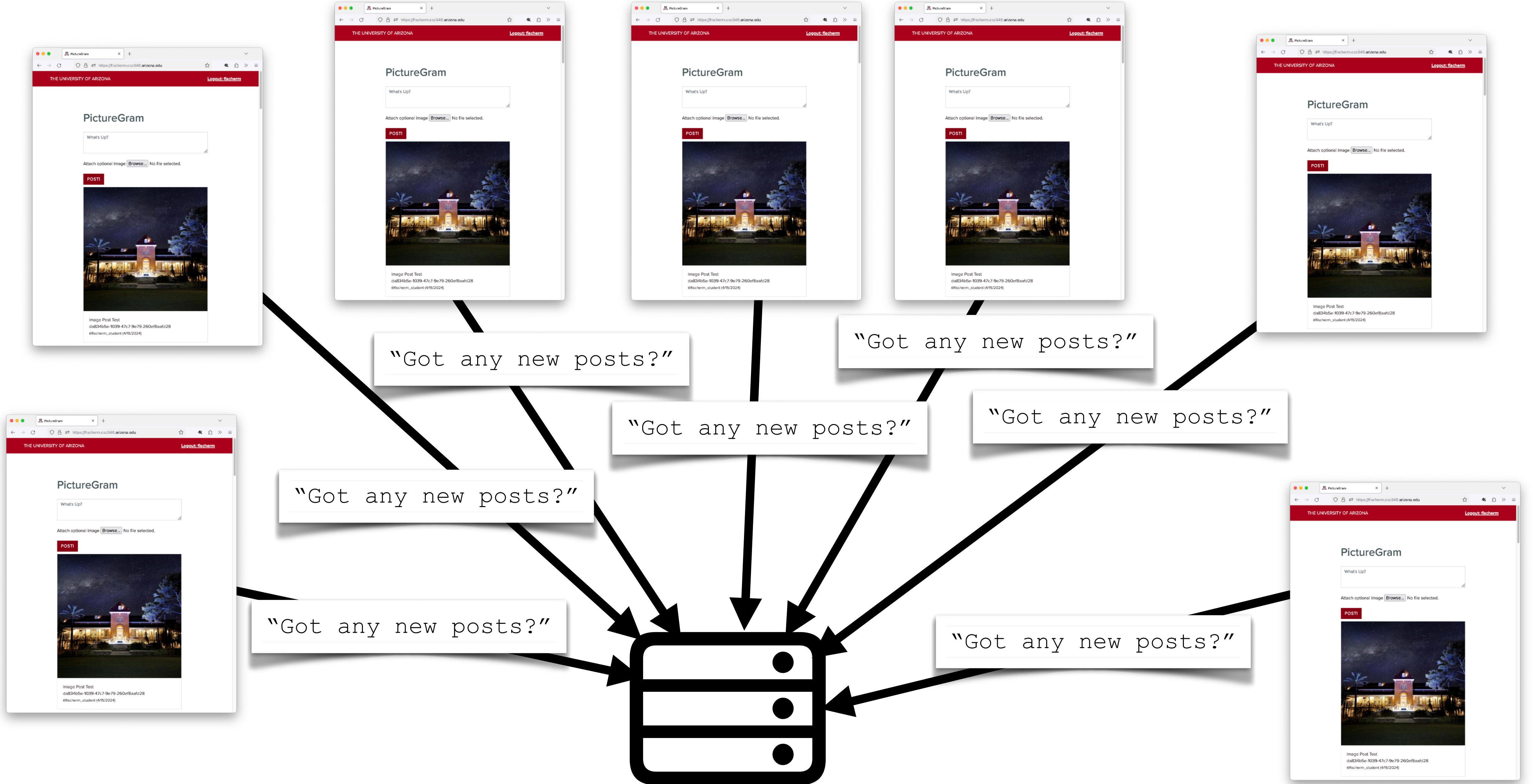
Image Post Test  
da834b5e-1039-47c7-9e79-260ef8aaf28  
@fischerm\_student (4/15/2024)

# Polling



api.csc346.arizona.edu

- This works OK for small numbers of infrequent polling
- What happens when there are many clients?



# Polling

## Has its downsides

- Polling requires each client to constantly ask the API for new data
- Short polling intervals can overwhelm the API host with incoming requests for updates
- Long polling intervals can result in significant delay getting new data out to clients
  - The Host may know there's a new message, but it has to wait for a client to ask for it

# WebSockets

**All that is old is new again**

- What if we could establish a long-lived network connection between the client and the host?
- This is what WebSockets does

# WebSockets

- So are WebSockets just regular TCP Sockets?
- Spoiler, No
- Conceptually, WebSockets and TCP Sockets have similar goals
  - Support Long-Lived Connections
  - Two-Way Communication
  - Not Request Based
- However they are not related technologically
  - WebSockets are an extension to the HTTP Protocol that runs on top of a TCP Socket

# WebSockets

## Challenges

- Low-level socket programming is hard
- Many network situations only permit “web” traffic over ports 80 or 443
- Session and state information about web application logins are already using Cookies, we don’t want a new way of handling state
- Security and encryption are already established for HTTPS communications, developing an additional model would be annoying

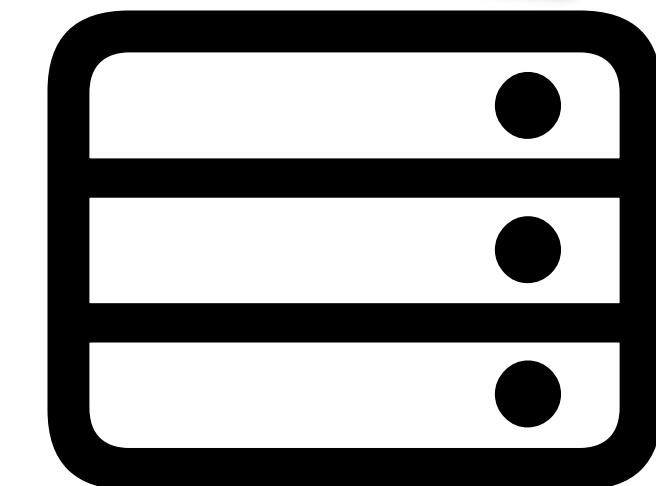
# WebSockets

## Solutions

- Implement a new type of HTTP request
- New request creates a “socket” inside an HTTP request
- Can stay open forever
- Bi-directional comm (not request/response)
- Relatively inexpensive (server memory, network)
- Uses standard HTTP mechanisms for encryption, cookies, etc.
  - Uses standard HTTP/HTTPS ports

The screenshot shows a web browser window titled "PictureGram" with the URL "https://fischerm.csc346.arizona.edu". The page has a red header bar with "THE UNIVERSITY OF ARIZONA" on the left and "Logout: fischerm" on the right. Below the header is the "PictureGram" logo. A text input field labeled "What's Up?" is present, along with a file upload section that says "Attach optional image" and a "Browse..." button. A "No file selected." message is displayed below the button. At the bottom is a red "POST!" button.

```
GET /chat HTTP/1.1  
Host: chat-api.csc346.arizona.edu  
Upgrade: websocket  
Connection: Upgrade  
Sec-WebSocket-Key: dGhlIHNhbXBsZSSub25jZQ==  
Sec-WebSocket-Version: 13
```



api.csc346.arizona.edu

- A regular HTTP request initiates the WebSocket handshake
- Additional headers are sent, telling the host that the client would like to upgrade this connection to a WebSocket
- Passes along a client key
  - This is just an identifier, not a cryptographic key

A PictureGram

THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

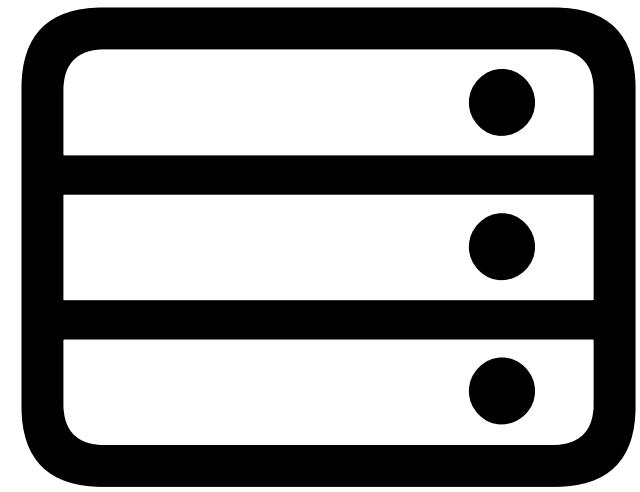
# PictureGram

What's Up?

Attach optional image  No file selected.

https://fischerm.csc346.arizona.edu

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzhZRbK+xOo=
```



api.csc346.arizona.edu

- If the server supports WebSockets, it responds with the correct headers
- The Sec-WebSocket-Accept response header is calculated in a seemingly overcomplicated way, but exists so that it's obvious to the client whether the server supports WebSockets

THE UNIVERSITY OF ARIZONA

Logout: fischerm

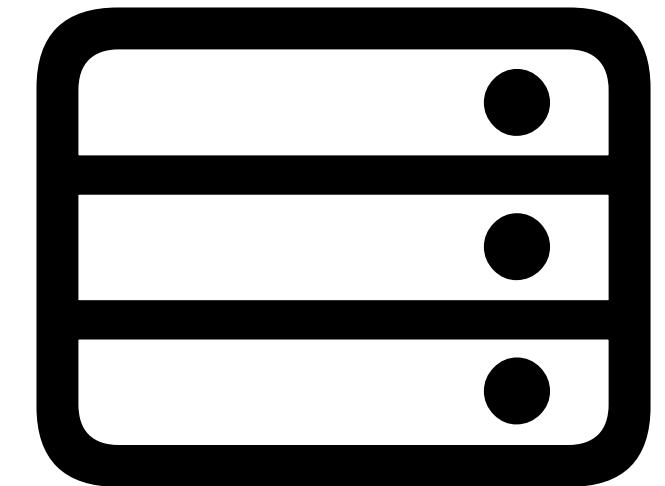
# PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzhZRbK+xOo=
```



api.csc346.arizona.edu

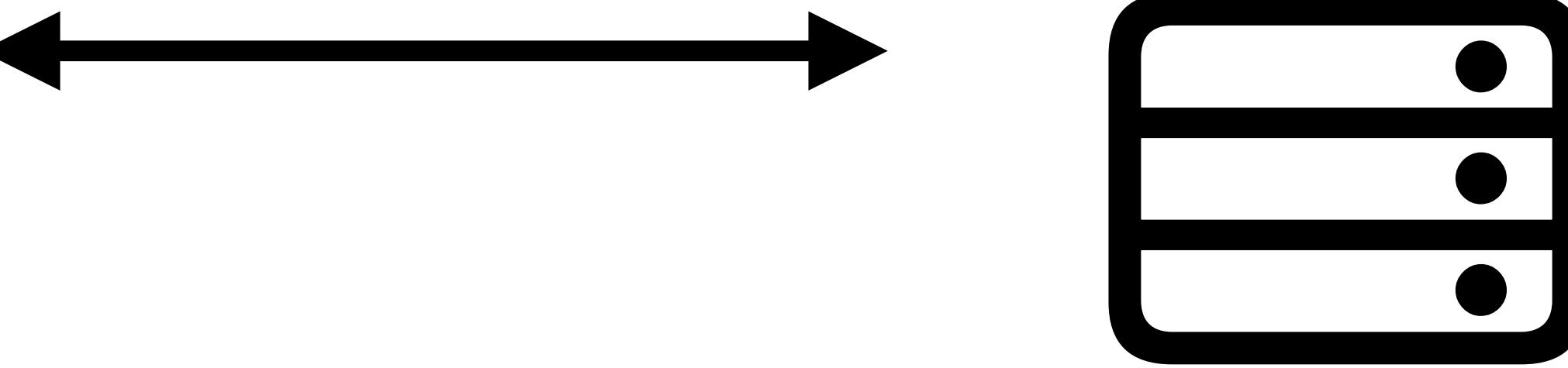
- The Sec-WebSocket-Accept header is important in that the server must derive it from the Sec-WebSocket-Key that the client sent to it.
- To get it, concatenate the client's Sec-WebSocket-Key and the string "258EAFA5-E914-47DA-95CA-C5AB0DC85B11" together, take the SHA-1 hash of the result, and return the base64 encoding of that hash.
- You likely will never have to do this unless you want to implement a WebSockets compliant HTTP server. Still useful to know that it's part of the handshake.

PictureGram

What's Up?

Attach optional image  No file selected.

**POST!**



api.csc346.arizona.edu

- From that point on, there is a persistent connection between the client and host
- Connection remains open until one side or the other explicitly closes it
- Data can be sent and initiated in either direction by either the client or the host at any time
- Data transfer is now a binary format

A PictureGram

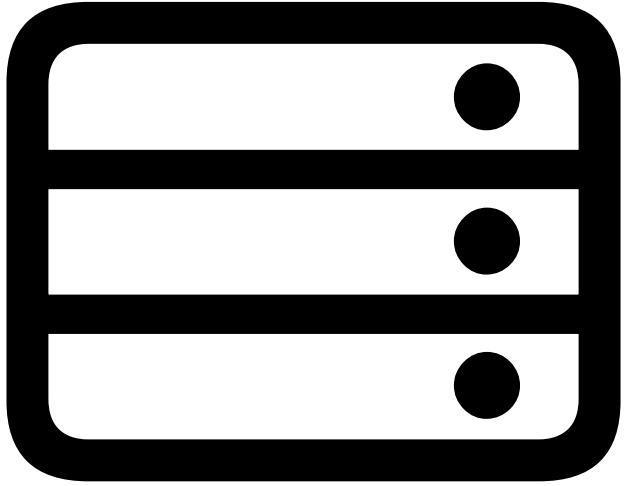
THE UNIVERSITY OF ARIZONA

[Logout: fischerm](#)

# PictureGram

What's Up?

Attach optional image  No file selected.



api.csc346.arizona.edu

[https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API/Writing\\_WebSocket\\_servers](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers)

# WebSockets

## Using with JavaScript

- Handshake details are handled by the browser
- Presents a JavaScript interface to us: `new WebSocket(...)`

```
const apiHost = "wss://chat-api.csc346.arizona.edu/chats"
const exampleSocket = new WebSocket(apiHost)
```

# WebSockets

## Using with JavaScript

- New Protocol prefix: ws:// and wss://
  - ws:// kicks off a handshake over http://
  - wss:// kicks off the handshake over https://

```
const apiHost = "wss://chat-api.csc346.arizona.edu/chats"
const exampleSocket = new WebSocket(apiHost)
```

# WebSockets

## Sending messages to the server

```
const apiHost = "wss://chat-api.csc346.arizona.edu/chats"
const exampleSocket = new WebSocket(apiHost)

exampleSocket.send("Message to the server")

data = {
  "type": "newchat",
  "message": "Here's a new chat message",
  "user": "fischerm"
}

exampleSocket.send(data)
```

# WebSockets

## Listening for incoming messages

```
const apiHost = "wss://chat-api.csc346.arizona.edu/chats"
const exampleSocket = new WebSocket(apiHost)

exampleSocket.addEventListener('message', (event) => {
    console.log('Message from server ', event.data);
});
```

# **WebSockets**

**MTG Card Demo**

# **WebSockets**

## **From the Server's Side**

# WebSockets

## Server Responsibilities

- The server side has a few duties
  - Accept HTTP Connections and look for the `Upgrade: websocket` and `Connection: Upgrade` headers
  - Calculate the correct `Sec-WebSocket-Accept` response value
  - Keep the WebSocket open
  - Keep track of all open WebSockets, and allow an API to send messages to specific clients

# WebSockets

## AWS API Gateway

- Most Cloud Providers have a managed service for WebSockets
- AWS API Gateway supports multiple API specifications
  - REST
  - Basic HTTP
  - WebSockets

<https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-overview.html>

# WebSockets

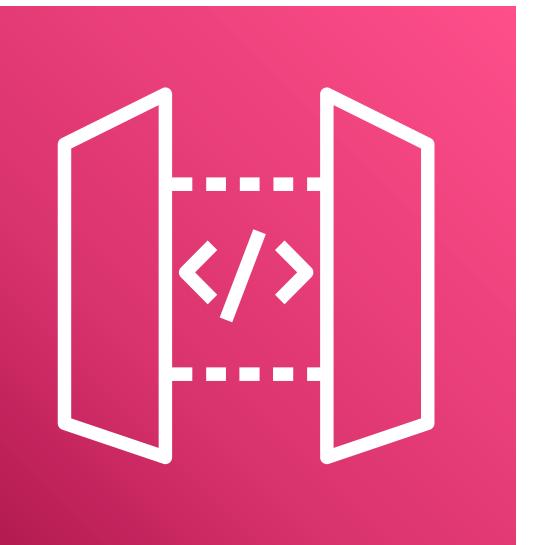
## AWS API Gateway

- API Gateway takes care of all the protocol level work associated with WebSockets
  - Accepts and Upgrades WebSocket connections
  - Calculates Sec-WebSocket-Accept responses
  - Keeps Socket connections open
  - Assigns Connection IDs to each open WebSocket and tracks activity
  - Sends activity to a backend processor, ie Lambda

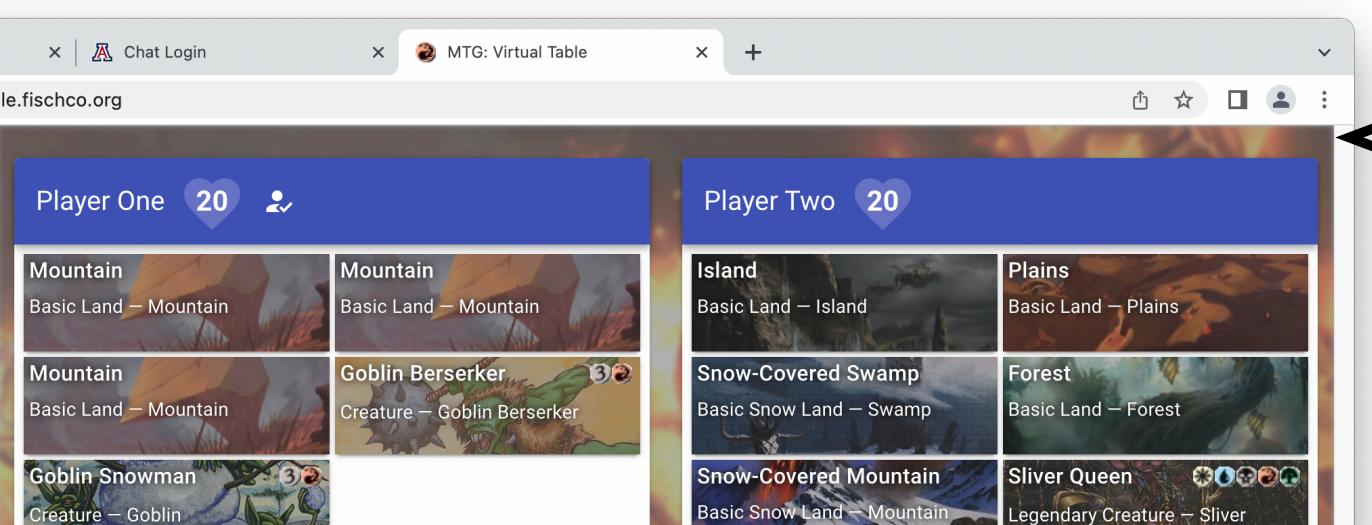
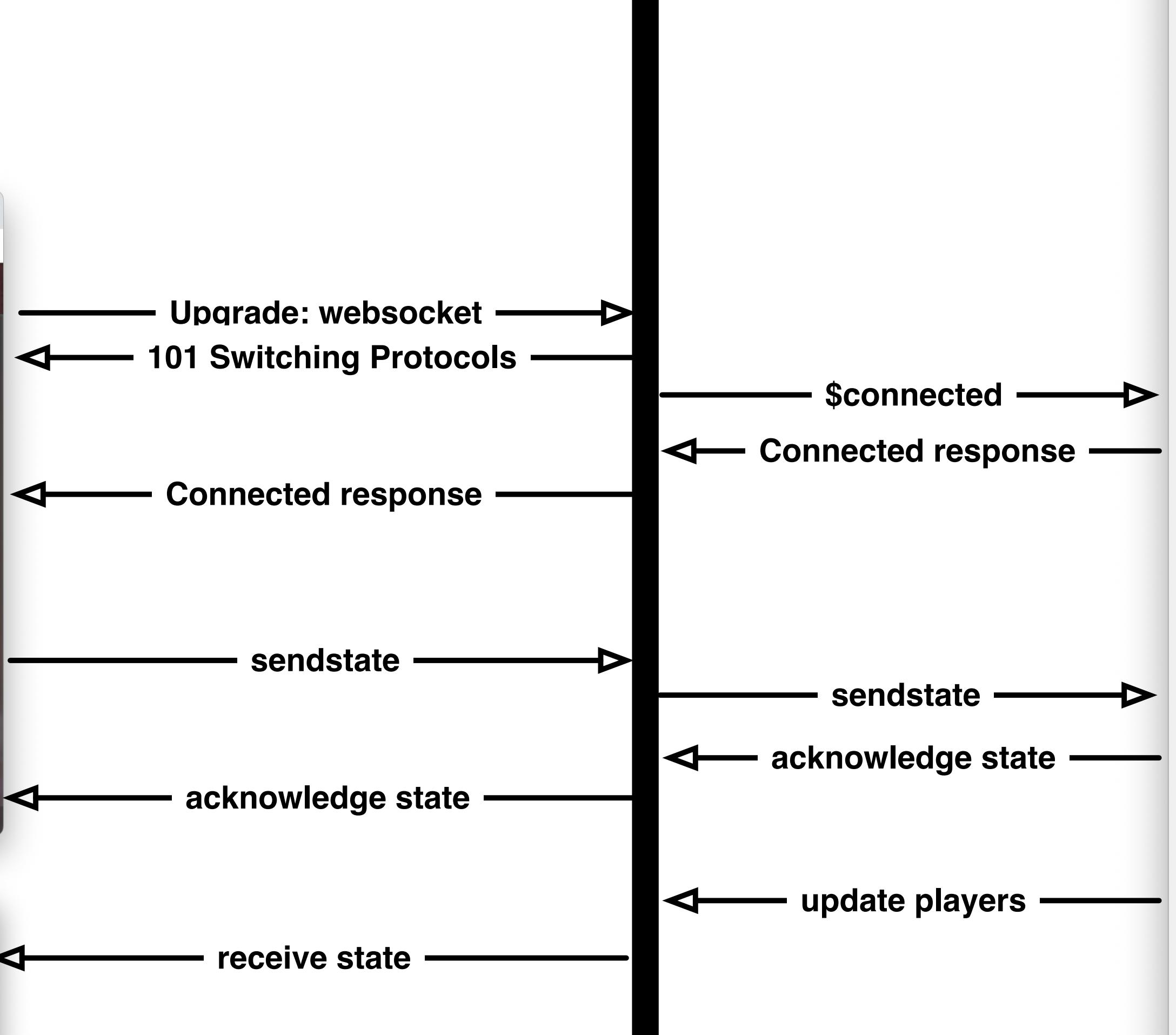
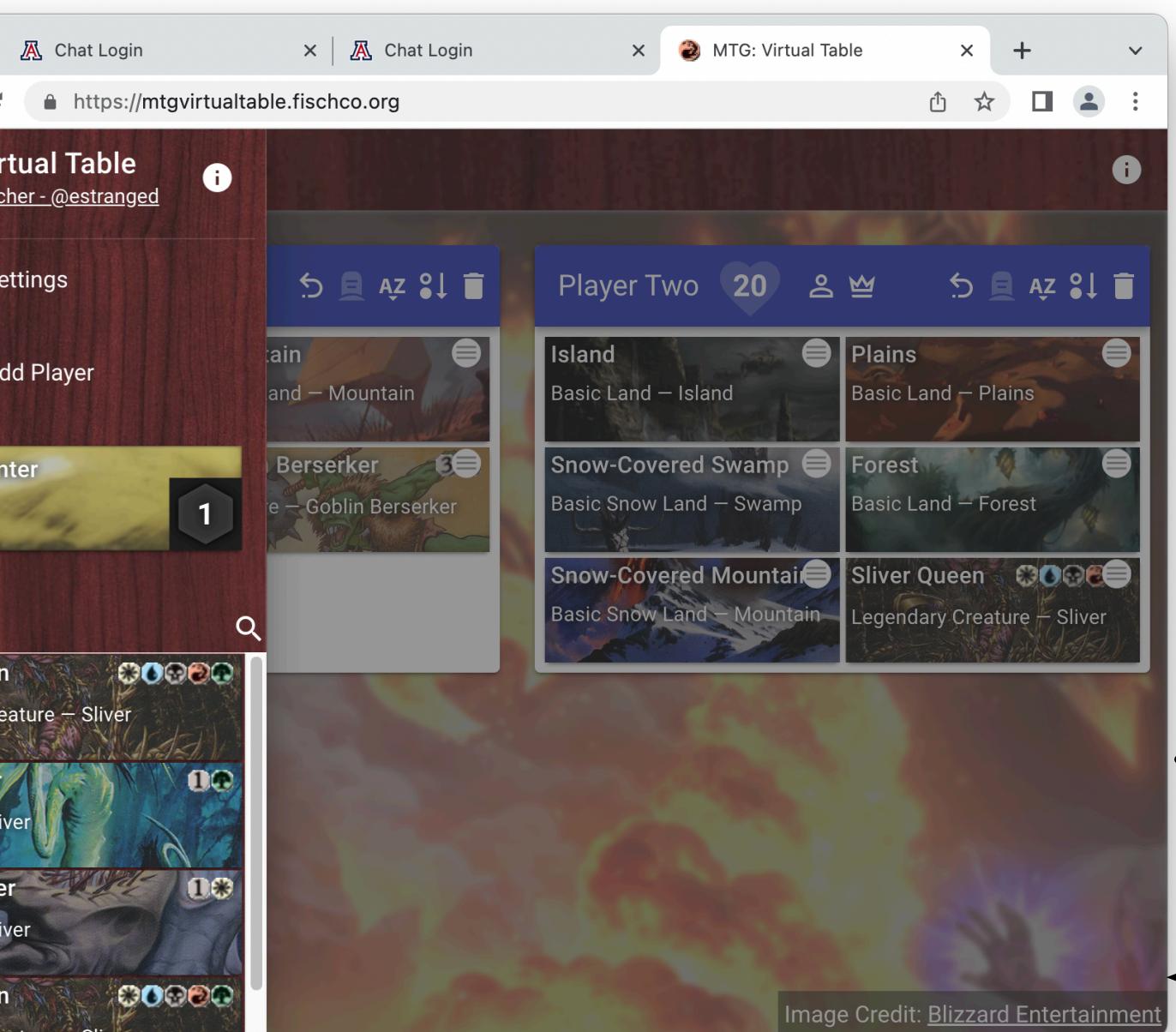
# WebSockets

## AWS API Gateway

API Gateway



Lambda



```
connection.py
serverless > connection.py > connection.py
239 def handler(event, context):
240
241     logger.info(json.dumps(event))
242
243     action = event.get('event', None)
244     request_context = event.get('requestContext', None)
245     if not request_context:
246         return _format_response(f'Missing requestContext', 400)
247
248     route_key = request_context.get('routeKey', None)
249     if not route_key:
250         return _format_response(f'Missing routeKey', 400)
251
252     connection_id = request_context.get('connectionId', None)
253     if not connection_id:
254         return _format_response(f'Missing connectionId', 400)
255
256     domain_name = request_context.get('domainName', None)
257     if not domain_name:
258         return _format_response(f'Missing domainName', 400)
259
260     body = event.get('body', None)
261     if body:
262         body = json.loads(body)
263
264     if route_key == "$connect":
265         return _format_response(f'Connected', 200)
266
267     if route_key == "$disconnect":
268         return _format_response(f'Bye', 200)
269
270     if route_key == "$default":
271         resp = {}
272
273     if not body:
274         return _format_response(f'Missing body', 400)
275
276     action = body.get('action', None)
277
278     if not action:
279         return _format_response(f'Missing action', 400)
300
301     if action == "updatePlayers":
302         # Logic to update players
303
304     if action == "receiveState":
305         # Logic to receive state
306
307     if action == "sendState":
308         # Logic to send state
309
310     if action == "acknowledgeState":
311         # Logic to acknowledge state
312
313     if action == "disconnect":
314         # Logic to disconnect
315
316     if action == "connect":
317         # Logic to connect
318
319     if action == "default":
320         # Logic for default action
321
322     if action == "error":
323         # Logic for error handling
324
325     if action == "pong":
326         # Logic for pong
327
328     if action == "ping":
329         # Logic for ping
330
331     if action == "close":
332         # Logic for close
333
334     if action == "open":
335         # Logic for open
336
337     if action == "upgradeToWebSocket":
338         # Logic for upgrade to websocket
339
340     if action == "downgradeFromWebSocket":
341         # Logic for downgrade from websocket
342
343     if action == "checkConnection":
344         # Logic for check connection
345
346     if action == "getDomainName":
347         # Logic for get domain name
348
349     if action == "getRouteKey":
350         # Logic for get route key
351
352     if action == "getConnectionId":
353         # Logic for get connection id
354
355     if action == "getHttpRequest":
356         # Logic for get http request
357
358     if action == "getHttpMethod":
359         # Logic for get http method
360
361     if action == "getHttpPath":
362         # Logic for get http path
363
364     if action == "getHttpQueryString":
365         # Logic for get http query string
366
367     if action == "getHttpHeaders":
368         # Logic for get http headers
369
370     if action == "getHttpRequestVersion":
371         # Logic for get http request version
372
373     if action == "getAwsRequestId":
374         # Logic for get aws request id
375
376     if action == "getAwsIdentity":
377         # Logic for get aws identity
378
379     if action == "getAwsIdentityArn":
380         # Logic for get aws identity arn
381
382     if action == "getAwsRegion":
383         # Logic for get aws region
384
385     if action == "getAwsLogGroup":
386         # Logic for get aws log group
387
388     if action == "getAwsLogStream":
389         # Logic for get aws log stream
390
391     if action == "getAwsMemoryLimit":
392         # Logic for get aws memory limit
393
394     if action == "getAwsRequestId":
395         # Logic for get aws request id
396
397     if action == "getAwsIdentity":
398         # Logic for get aws identity
399
400     if action == "getAwsIdentityArn":
401         # Logic for get aws identity arn
402
403     if action == "getAwsRegion":
404         # Logic for get aws region
405
406     if action == "getAwsLogGroup":
407         # Logic for get aws log group
408
409     if action == "getAwsLogStream":
410         # Logic for get aws log stream
411
412     if action == "getAwsMemoryLimit":
413         # Logic for get aws memory limit
414
415     if action == "getAwsRequestId":
416         # Logic for get aws request id
417
418     if action == "getAwsIdentity":
419         # Logic for get aws identity
420
421     if action == "getAwsIdentityArn":
422         # Logic for get aws identity arn
423
424     if action == "getAwsRegion":
425         # Logic for get aws region
426
427     if action == "getAwsLogGroup":
428         # Logic for get aws log group
429
430     if action == "getAwsLogStream":
431         # Logic for get aws log stream
432
433     if action == "getAwsMemoryLimit":
434         # Logic for get aws memory limit
435
436     if action == "getAwsRequestId":
437         # Logic for get aws request id
438
439     if action == "getAwsIdentity":
440         # Logic for get aws identity
441
442     if action == "getAwsIdentityArn":
443         # Logic for get aws identity arn
444
445     if action == "getAwsRegion":
446         # Logic for get aws region
447
448     if action == "getAwsLogGroup":
449         # Logic for get aws log group
450
451     if action == "getAwsLogStream":
452         # Logic for get aws log stream
453
454     if action == "getAwsMemoryLimit":
455         # Logic for get aws memory limit
456
457     if action == "getAwsRequestId":
458         # Logic for get aws request id
459
460     if action == "getAwsIdentity":
461         # Logic for get aws identity
462
463     if action == "getAwsIdentityArn":
464         # Logic for get aws identity arn
465
466     if action == "getAwsRegion":
467         # Logic for get aws region
468
469     if action == "getAwsLogGroup":
470         # Logic for get aws log group
471
472     if action == "getAwsLogStream":
473         # Logic for get aws log stream
474
475     if action == "getAwsMemoryLimit":
476         # Logic for get aws memory limit
477
478     if action == "getAwsRequestId":
479         # Logic for get aws request id
480
481     if action == "getAwsIdentity":
482         # Logic for get aws identity
483
484     if action == "getAwsIdentityArn":
485         # Logic for get aws identity arn
486
487     if action == "getAwsRegion":
488         # Logic for get aws region
489
490     if action == "getAwsLogGroup":
491         # Logic for get aws log group
492
493     if action == "getAwsLogStream":
494         # Logic for get aws log stream
495
496     if action == "getAwsMemoryLimit":
497         # Logic for get aws memory limit
498
499     if action == "getAwsRequestId":
500         # Logic for get aws request id
501
502     if action == "getAwsIdentity":
503         # Logic for get aws identity
504
505     if action == "getAwsIdentityArn":
506         # Logic for get aws identity arn
507
508     if action == "getAwsRegion":
509         # Logic for get aws region
510
511     if action == "getAwsLogGroup":
512         # Logic for get aws log group
513
514     if action == "getAwsLogStream":
515         # Logic for get aws log stream
516
517     if action == "getAwsMemoryLimit":
518         # Logic for get aws memory limit
519
520     if action == "getAwsRequestId":
521         # Logic for get aws request id
522
523     if action == "getAwsIdentity":
524         # Logic for get aws identity
525
526     if action == "getAwsIdentityArn":
527         # Logic for get aws identity arn
528
529     if action == "getAwsRegion":
530         # Logic for get aws region
531
532     if action == "getAwsLogGroup":
533         # Logic for get aws log group
534
535     if action == "getAwsLogStream":
536         # Logic for get aws log stream
537
538     if action == "getAwsMemoryLimit":
539         # Logic for get aws memory limit
540
541     if action == "getAwsRequestId":
542         # Logic for get aws request id
543
544     if action == "getAwsIdentity":
545         # Logic for get aws identity
546
547     if action == "getAwsIdentityArn":
548         # Logic for get aws identity arn
549
550     if action == "getAwsRegion":
551         # Logic for get aws region
552
553     if action == "getAwsLogGroup":
554         # Logic for get aws log group
555
556     if action == "getAwsLogStream":
557         # Logic for get aws log stream
558
559     if action == "getAwsMemoryLimit":
560         # Logic for get aws memory limit
561
562     if action == "getAwsRequestId":
563         # Logic for get aws request id
564
565     if action == "getAwsIdentity":
566         # Logic for get aws identity
567
568     if action == "getAwsIdentityArn":
569         # Logic for get aws identity arn
570
571     if action == "getAwsRegion":
572         # Logic for get aws region
573
574     if action == "getAwsLogGroup":
575         # Logic for get aws log group
576
577     if action == "getAwsLogStream":
578         # Logic for get aws log stream
579
580     if action == "getAwsMemoryLimit":
581         # Logic for get aws memory limit
582
583     if action == "getAwsRequestId":
584         # Logic for get aws request id
585
586     if action == "getAwsIdentity":
587         # Logic for get aws identity
588
589     if action == "getAwsIdentityArn":
590         # Logic for get aws identity arn
591
592     if action == "getAwsRegion":
593         # Logic for get aws region
594
595     if action == "getAwsLogGroup":
596         # Logic for get aws log group
597
598     if action == "getAwsLogStream":
599         # Logic for get aws log stream
600
601     if action == "getAwsMemoryLimit":
602         # Logic for get aws memory limit
603
604     if action == "getAwsRequestId":
605         # Logic for get aws request id
606
607     if action == "getAwsIdentity":
608         # Logic for get aws identity
609
610     if action == "getAwsIdentityArn":
611         # Logic for get aws identity arn
612
613     if action == "getAwsRegion":
614         # Logic for get aws region
615
616     if action == "getAwsLogGroup":
617         # Logic for get aws log group
618
619     if action == "getAwsLogStream":
620         # Logic for get aws log stream
621
622     if action == "getAwsMemoryLimit":
623         # Logic for get aws memory limit
624
625     if action == "getAwsRequestId":
626         # Logic for get aws request id
627
628     if action == "getAwsIdentity":
629         # Logic for get aws identity
630
631     if action == "getAwsIdentityArn":
632         # Logic for get aws identity arn
633
634     if action == "getAwsRegion":
635         # Logic for get aws region
636
637     if action == "getAwsLogGroup":
638         # Logic for get aws log group
639
640     if action == "getAwsLogStream":
641         # Logic for get aws log stream
642
643     if action == "getAwsMemoryLimit":
644         # Logic for get aws memory limit
645
646     if action == "getAwsRequestId":
647         # Logic for get aws request id
648
649     if action == "getAwsIdentity":
650         # Logic for get aws identity
651
652     if action == "getAwsIdentityArn":
653         # Logic for get aws identity arn
654
655     if action == "getAwsRegion":
656         # Logic for get aws region
657
658     if action == "getAwsLogGroup":
659         # Logic for get aws log group
660
661     if action == "getAwsLogStream":
662         # Logic for get aws log stream
663
664     if action == "getAwsMemoryLimit":
665         # Logic for get aws memory limit
666
667     if action == "getAwsRequestId":
668         # Logic for get aws request id
669
670     if action == "getAwsIdentity":
671         # Logic for get aws identity
672
673     if action == "getAwsIdentityArn":
674         # Logic for get aws identity arn
675
676     if action == "getAwsRegion":
677         # Logic for get aws region
678
679     if action == "getAwsLogGroup":
680         # Logic for get aws log group
681
682     if action == "getAwsLogStream":
683         # Logic for get aws log stream
684
685     if action == "getAwsMemoryLimit":
686         # Logic for get aws memory limit
687
688     if action == "getAwsRequestId":
689         # Logic for get aws request id
690
691     if action == "getAwsIdentity":
692         # Logic for get aws identity
693
694     if action == "getAwsIdentityArn":
695         # Logic for get aws identity arn
696
697     if action == "getAwsRegion":
698         # Logic for get aws region
699
700     if action == "getAwsLogGroup":
701         # Logic for get aws log group
702
703     if action == "getAwsLogStream":
704         # Logic for get aws log stream
705
706     if action == "getAwsMemoryLimit":
707         # Logic for get aws memory limit
708
709     if action == "getAwsRequestId":
710         # Logic for get aws request id
711
712     if action == "getAwsIdentity":
713         # Logic for get aws identity
714
715     if action == "getAwsIdentityArn":
716         # Logic for get aws identity arn
717
718     if action == "getAwsRegion":
719         # Logic for get aws region
720
721     if action == "getAwsLogGroup":
722         # Logic for get aws log group
723
724     if action == "getAwsLogStream":
725         # Logic for get aws log stream
726
727     if action == "getAwsMemoryLimit":
728         # Logic for get aws memory limit
729
730     if action == "getAwsRequestId":
731         # Logic for get aws request id
732
733     if action == "getAwsIdentity":
734         # Logic for get aws identity
735
736     if action == "getAwsIdentityArn":
737         # Logic for get aws identity arn
738
739     if action == "getAwsRegion":
740         # Logic for get aws region
741
742     if action == "getAwsLogGroup":
743         # Logic for get aws log group
744
745     if action == "getAwsLogStream":
746         # Logic for get aws log stream
747
748     if action == "getAwsMemoryLimit":
749         # Logic for get aws memory limit
750
751     if action == "getAwsRequestId":
752         # Logic for get aws request id
753
754     if action == "getAwsIdentity":
755         # Logic for get aws identity
756
757     if action == "getAwsIdentityArn":
758         # Logic for get aws identity arn
759
760     if action == "getAwsRegion":
761         # Logic for get aws region
762
763     if action == "getAwsLogGroup":
764         # Logic for get aws log group
765
766     if action == "getAwsLogStream":
767         # Logic for get aws log stream
768
769     if action == "getAwsMemoryLimit":
770         # Logic for get aws memory limit
771
772     if action == "getAwsRequestId":
773         # Logic for get aws request id
774
775     if action == "getAwsIdentity":
776         # Logic for get aws identity
777
778     if action == "getAwsIdentityArn":
779         # Logic for get aws identity arn
780
781     if action == "getAwsRegion":
782         # Logic for get aws region
783
784     if action == "getAwsLogGroup":
785         # Logic for get aws log group
786
787     if action == "getAwsLogStream":
788         # Logic for get aws log stream
789
789     if action == "getAwsMemoryLimit":
790         # Logic for get aws memory limit
791
792     if action == "getAwsRequestId":
793         # Logic for get aws request id
794
795     if action == "getAwsIdentity":
796         # Logic for get aws identity
797
798     if action == "getAwsIdentityArn":
799         # Logic for get aws identity arn
800
801     if action == "getAwsRegion":
802         # Logic for get aws region
803
804     if action == "getAwsLogGroup":
805         # Logic for get aws log group
806
807     if action == "getAwsLogStream":
808         # Logic for get aws log stream
809
809     if action == "getAwsMemoryLimit":
810         # Logic for get aws memory limit
811
812     if action == "getAwsRequestId":
813         # Logic for get aws request id
814
815     if action == "getAwsIdentity":
816         # Logic for get aws identity
817
818     if action == "getAwsIdentityArn":
819         # Logic for get aws identity arn
820
821     if action == "getAwsRegion":
822         # Logic for get aws region
823
824     if action == "getAwsLogGroup":
825         # Logic for get aws log group
826
827     if action == "getAwsLogStream":
828         # Logic for get aws log stream
829
829     if action == "getAwsMemoryLimit":
830         # Logic for get aws memory limit
831
832     if action == "getAwsRequestId":
833         # Logic for get aws request id
834
835     if action == "getAwsIdentity":
836         # Logic for get aws identity
837
838     if action == "getAwsIdentityArn":
839         # Logic for get aws identity arn
840
841     if action == "getAwsRegion":
842         # Logic for get aws region
843
844     if action == "getAwsLogGroup":
845         # Logic for get aws log group
846
847     if action == "getAwsLogStream":
848         # Logic for get aws log stream
849
849     if action == "getAwsMemoryLimit":
850         # Logic for get aws memory limit
851
852     if action == "getAwsRequestId":
853         # Logic for get aws request id
854
855     if action == "getAwsIdentity":
856         # Logic for get aws identity
857
858     if action == "getAwsIdentityArn":
859         # Logic for get aws identity arn
860
861     if action == "getAwsRegion":
862         # Logic for get aws region
863
864     if action == "getAwsLogGroup":
865         # Logic for get aws log group
866
867     if action == "getAwsLogStream":
868         # Logic for get aws log stream
869
869     if action == "getAwsMemoryLimit":
870         # Logic for get aws memory limit
871
872     if action == "getAwsRequestId":
873         # Logic for get aws request id
874
875     if action == "getAwsIdentity":
876         # Logic for get aws identity
877
878     if action == "getAwsIdentityArn":
879         # Logic for get aws identity arn
880
881     if action == "getAwsRegion":
882         # Logic for get aws region
883
884     if action == "getAwsLogGroup":
885         # Logic for get aws log group
886
887     if action == "getAwsLogStream":
888         # Logic for get aws log stream
889
889     if action == "getAwsMemoryLimit":
890         # Logic for get aws memory limit
891
892     if action == "getAwsRequestId":
893         # Logic for get aws request id
894
895     if action == "getAwsIdentity":
896         # Logic for get aws identity
897
898     if action == "getAwsIdentityArn":
899         # Logic for get aws identity arn
900
901     if action == "getAwsRegion":
902         # Logic for get aws region
903
904     if action == "getAwsLogGroup":
905         # Logic for get aws log group
906
907     if action == "getAwsLogStream":
908         # Logic for get aws log stream
909
909     if action == "getAwsMemoryLimit":
910         # Logic for get aws memory limit
911
912     if action == "getAwsRequestId":
913         # Logic for get aws request id
914
915     if action == "getAwsIdentity":
916         # Logic for get aws identity
917
918     if action == "getAwsIdentityArn":
919         # Logic for get aws identity arn
920
921     if action == "getAwsRegion":
922         # Logic for get aws region
923
924     if action == "getAwsLogGroup":
925         # Logic for get aws log group
926
927     if action == "getAwsLogStream":
928         # Logic for get aws log stream
929
929     if action == "getAwsMemoryLimit":
930         # Logic for get aws memory limit
931
932     if action == "getAwsRequestId":
933         # Logic for get aws request id
934
935     if action == "getAwsIdentity":
936         # Logic for get aws identity
937
938     if action == "getAwsIdentityArn":
939         # Logic for get aws identity arn
940
941     if action == "getAwsRegion":
942         # Logic for get aws region
943
944     if action == "getAwsLogGroup":
945         # Logic for get aws log group
946
947     if action == "getAwsLogStream":
948         # Logic for get aws log stream
949
949     if action == "getAwsMemoryLimit":
950         # Logic for get aws memory limit
951
952     if action == "getAwsRequestId":
953         # Logic for get aws request id
954
955     if action == "getAwsIdentity":
956         # Logic for get aws identity
957
958     if action == "getAwsIdentityArn":
959         # Logic for get aws identity arn
960
961     if action == "getAwsRegion":
962         # Logic for get aws region
963
964     if action == "getAwsLogGroup":
965         # Logic for get aws log group
966
967     if action == "getAwsLogStream":
968         # Logic for get aws log stream
969
969     if action == "getAwsMemoryLimit":
970         # Logic for get aws memory limit
971
972     if action == "getAwsRequestId":
973         # Logic for get aws request id
974
975     if action == "getAwsIdentity":
976         # Logic for get aws identity
977
978     if action == "getAwsIdentityArn":
979         # Logic for get aws identity arn
980
981     if action == "getAwsRegion":
982         # Logic for get aws region
983
984     if action == "getAwsLogGroup":
985         # Logic for get aws log group
986
987     if action == "getAwsLogStream":
988         # Logic for get aws log stream
989
989     if action == "getAwsMemoryLimit":
990         # Logic for get aws memory limit
991
992     if action == "getAwsRequestId":
993         # Logic for get aws request id
994
995     if action == "getAwsIdentity":
996         # Logic for get aws identity
997
998     if action == "getAwsIdentityArn":
999         # Logic for get aws identity arn
999
999     if action == "getAwsRegion":
999
999     if action == "getAwsLogGroup":
999
999     if action == "getAwsLogStream":
999
999     if action == "getAwsMemoryLimit":
999
```

# **WebSockets**

## **Server Code Demo**