# **CSC 346 - Cloud Computing** 04 - Web Servers, Ports & Sockets

- How do things communicate over the internet? (the simple version)
- This is not a networking class is

THE UNIVERSITY ■ Welcome The class websit assignments. Overviev CSC 346 Class S Instructor: Mark Instructors Ema fall2022@list.ar	Welcome to CSC 346 - Fall 2 Welcome to CSC 346 - Fall 2 e to CSC 346 - Fall 2 te will be used to share files, s te will be used to share files, s N Syllabus Fischer: fischerm@arizona.ed il List: csc346-instructors- izona.edu	022 ≡ Fall 2022 lides, and	
Day	Times		
Monday	9am - 10am GS 934		
	6pm - 7pm Discord/700m		
Tuesday	6pm - 7pm Discord/Zoom		
Wednesday	9am - 10am GS 934		
	6pm - 7pm Discord/Zoom		
Friday	9am - 10am GS 846		



- Some computing resource must *bind* to a specific *port* on its host, and then *listen* for incoming connections
- Listens on a specific *port*
- For a HTTP, this software is our web server
- Since a bind must always precede a listen, we will typically omit the bind in our descriptions
- Most socket libraries will take care of this for you



## **Networking Ports** What's a Port?

- It's basically a door
  - Italian: Porta
  - French: Porte
  - Spanish: Puerta
- I like to think of a port as a door to a building.



# **Networking Ports** What's a Port?

- If we have some device on the internet with an IP address assigned to it, we can think of that as a building.
- A port then can be thought of as a door to the building.
- Doors can let stuff in or out.



# **Networking Ports** What's a Port?

- Each port has a number
  - 16 bit unsigned integers

123

- 0 65535
- Internet Assigned Numbers Authority (IANA) has designated different port ranges for different thing, but there's nothing stopping you from using them for whatever





# Networking Ports Common Ports

Port Number	Application
22	ssh - Secure Shell
23	Telnet (unsecure)
25	SMTP - Simple Mail Transport Protocol (unsecure)
80	HTTP - HyperText Transport Protocol (unsecure)
123	NTP - Network Time Protocol
443	HTTPS - HTTP Secure
587	SMTP Secure
3306	MySQL
25565	Minecraft

- A client then opens a socket to the server
- A socket data stream that sits on top of the network layer provided by the operating system.
- A socket is described by an *IP address*, a port, and a transport protocol
- For our class, we'll use TCP for our protocol
  - **Transmission Control Protocol**



## IP: 192.12.69.186 Port: 80 Protocol: TCP

listen(80)

Web Server



- Both sides must *bind* to a port
- The server binds to the well known port 80, since the clients need to know this
- The client typically uses a random high number available port
- As part of the socket connection, the client tells the server what port it is using

# if you need an invite to the CS department's Discord server, em vising@cs.arizona.edu Exam Schedule Exam 1: September 28t

## bind(46723)

## IP: 192.12.69.186 Port: 80 Protocol: TCP

## bind(80)





 A web server can listen for and accept connections from many clients

Welco	Welcome to CSC 346 - Fall 2	022	
The el	e to CSC 346 - r		
assignments.	site will be used to share files a	all 2022	
		ildes, and	
Overvie	W		
CSC 346 Class	Syllabus		
Instructor: Mar	k Fischer: final		
Insturctors Em	ail List: coo246	lu	
fall2022@list.a	arizona.edu		
Day	Times		
<b>Day</b> Monday	<b>Times</b> 9am - 10am GS 934		
<b>Day</b> Monday	Times 9am - 10am GS 934 6pm - 7pm Discord/Zoom		
<b>Day</b> Monday Tuesday	Times 9am - 10am GS 934 6pm - 7pm Discord/Zoom 6pm - 7pm Discord/Zoom		
Day Monday Tuesday Wednesday	Times 9am - 10am GS 934 6pm - 7pm Discord/Zoom 6pm - 7pm Discord/Zoom 9am - 10am GS 934		
Day Monday Tuesday Wednesday	Times         9am - 10am GS 934         6pm - 7pm Discord/Zoom         6pm - 7pm Discord/Zoom         9am - 10am GS 934         6pm - 7pm Discord/Zoom		
Day Monday Tuesday Wednesday Friday	Times         9am - 10am GS 934         6pm - 7pm Discord/Zoom         6pm - 7pm Discord/Zoom         9am - 10am GS 934         6pm - 7pm Discord/Zoom		
Day Monday Tuesday Wednesday Friday	Times 9am - 10am GS 934 6pm - 7pm Discord/Zoom 6pm - 7pm Discord/Zoom 9am - 10am GS 934 6pm - 7pm Discord/Zoom 9am - 10am GS 846	SCOTT Server amail	
Day Monday Tuesday Wednesday Friday If you need an i advising@cs.at	Times         9am - 10am GS 934         6pm - 7pm Discord/Zoom         6pm - 7pm Discord/Zoom         9am - 10am GS 934         6pm - 7pm Discord/Zoom         9am - 10am GS 934         opm - 7pm Discord/Zoom         opm - 7pm Discord/Zoom         opm - 10am GS 846         onvite to the CS department's Discord.	scord server, email	



- Once a socket is connected, the client and server can exchange data according to whatever protocol the server supports.
- For web servers, this is HTTP

]	Welcom The class websi assignments.	Welcome to CSC 346 - Fall ; e to CSC 346 - ite will be used to share files,	1022 ≡ Fall 2022		
	Overview	A1	uuco, dha		
	CSC 346 Class	VV Svit-1			
	Instructor: Mark	Syllabus			
	Insturctors Ema fall2022@list.al	il List: csc346-instructors- rizona.edu	du		
	Day	Times			
	Monday	9am - 10am GS 934			
		6pm - 7pm Discord/Zoom		C	ΈT
	Tuesday	брт - 7pm Discord/Zoom		1_	
	Wednesday	9am - 10am GS 934		Ω	lOS
		6pm - 7pm Discord/Zoom			
	Friday	9am - 10am GS 846			
	If you need an in advising@cs.ar	nvite to the CS department's D i <b>zona.edu</b> .	iscord server, email		

/index.html HTTP/1.1 : example.com

Web Server listen(80)

## **Echo Server** The world's worst web server

•••					Sei
ζη	EXPLORER	 \$ run.s	h	🗬 server.py 🛛 🗙	
	> OPEN EDITORS	🅏 serv	er.py >		
Q	$\vee$ CS346	1	from so	ocket <i>import</i> *	
/	\$ run.sh	2	import	logging	
90	🍓 server.py	3			
B		4	logging	g.basicConfig(leve	el=logging
~		5			
æ'>		6	logging	g.info("Starting S	Server")
		7			
		8	server_	_socket = socket()	)
10		9	server_	$_addr = ("0.0.0.0]$	", 80)
-0		10	server_	_socket.bind(serve	er_addr)
Ш		11	server_	_socket.listen(5)	
_		12		_	
Д		13	while 1	True:	
		14	(cc	onn, client_addr)	= server_
		15	wit	th conn:	
		16		logging.into(t"	Connection
		17		while True:	
		18		data = conn	.recv(1024
		19		<i>ι</i> j not data	•

erver.py – cs346



•••			se
ŋ	EXPLORER	··· \$ run.sh	🕏 server.py 🛛 🕹
	> OPEN EDITORS	server.py >	•••
$\bigcirc$	$\sim$ CS346	1 from	socket <i>import</i> *
$\sim$	\$ run.sh	2 impor	t logging
00	server.pv	3	
20	••••••••••••	4 loggi	ng.basicConfig(level=logging
		5	
		6 loggi	ng.info("Starting Server")
		7	
		8 serve	r_socket = socket()
09		9 serve	r_addr = ("0.0.0.0", 80)
-0		10 serve	r_socket.bind(server_addr)
Ш		11 serve	r_socket.listen(5)
_		12	
Д		13 while	True:
		14 (	<pre>conn, client_addr) = server_</pre>
		15 W	uth conn:
		16	Logging.into(I"Connection
		10	while frue:
		10	dala = conn.recv(1024
		19	broch
		20	logging info(data)
		22	logging.info("Connection
		DEBUG CONSOL • ~/cs346 \$ ./r + docker run 2022-09-11 01	E TERMINAL PROBLEMS OUT cun.sh -irmname python_socket -p L:45:07,636 INF0 Starting Server
8			

N





```
g.INFO, format="%(asctime)s %(levelname)s %(message)s")
```

```
_socket.accept()
```

```
from {client_addr}")
```

```
Closed")
```

FPUT JUPYTER

```
> bash + \vee \square \square \land \times
```

•••						se
ŋ	EXPLORER	\$	run.sh	🍨 server.py	×	
	<ul> <li>&gt; OPEN EDITORS</li> <li>&gt; CS346</li> <li>\$ run.sh</li> <li>&gt; server.py</li> </ul>		<pre>server.py &gt; 1  from so 2  import 3 4  logging 5 6  logging 7 8  server 9  server 9  server 1  server 1  server 2 3  while 1 4  (cc 5  wit 6 7 8 9  logging 1  logging 1</pre>	<pre>ocket import logging g.basicConfig g.info("Star socket = soc addr = ("0.0 socket.bind socket.liste frue: onn, client_a broket.liste frue: onn, client_a broket.liste if not logging.in: while True data = if not broket logging.in:</pre>	* g(le ting cket 0.0. (ser en(5 addr fo(f : con dat eak g.in fo("	<pre>evel=logging () () () () () () () () () () () () ()</pre>
0		DE • ~/ + 20	BUG CONSOLE cs346 \$ ./rur docker run -i 22-09-11 01:4	TERMINAL n.sh irmname 45:07,636 INFC	PRC pyth ) Sta	DBLEMS OUT non_socket -p arting Server

N



g.INFO, format="%(asctime)s %(levelname)s %(message)s")



\_socket.accept()

from {client\_addr}")

Closed")

FPUT JUPYTER

> bash  $+ \vee$   $\square$   $\square$   $\land$   $\times$ 

•••									se
Ŋ	EXPLORER	\$	run.sł	ı	🇬 server.	ру >	×		
	<ul> <li>&gt; OPEN EDITORS</li> <li>&gt; CS346</li> <li>\$ run.sh</li> <li>? server.py</li> </ul>		<pre>serve 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22</pre>	er.py > from so import logging server_ server_ server_ server_ server_ int server_ server_ server_ server_ server_ server_	<pre>ocket impo logging g.basicCon g.info("St socket = addr = (" socket.bi socket.li socket.li frue: onn, clien ch conn: logging. while Tr data if n logging.</pre>	<pre>rt * fig( arti sock 0.0. nd(s sten t_ad info ue: _ = c ot d brea info ing. info</pre>	<pre>ing ing inf inf inf inf inf inf</pre>	el=lo Serve ) ", 80 er_ad er_ad conne .recv : io(dat	gging r") ) dr) rver_ ction (1024 a) tion
8		D • ~ + 20	EBUG C /cs346 docke 022-09	CONSOLE () \$ ./rur () r run -i () -11 01:4	TERMINAL n.sh irmna 15:07,636 I	me py	PROB ytho Star	LEMS	OUT (et -p Server



•••						se
ŋ	EXPLORER	\$ run.s	h	🇬 server.p	y ×	
	<ul> <li>&gt; OPEN EDITORS</li> <li>~ CS346</li> <li>\$ run.sh</li> <li>@ server.py</li> </ul>	Full.s	er.py > from so import logging server_ server_ server_ server_ uhile 1 (co wit	<pre>cket impor logging g.basicConf g.info("Sta _socket = s _addr = ("0 _socket.bin _socket.lis frue: _socket.lis frue: </pre>	t * ig(lo ig(lo ig(lo ig(lo rting ocke .0.0 d(se ten( ic ten( ic ic ic ic ic ic ic ic ic ic ic ic ic	<pre>evel=logging g Server") t() .0", 80) rver_addr) 5) r) = server_ f"Connectior nn.recv(1024 ta: nfo(data) "Connection</pre>
8		DEBUG • ~/cs34 + dock 2022-00	CONSOLE 6 \$ ./rur er run -i 9-11 01:4	TERMINAL n.sh irmnam 45:07,636 IN	PRO e pyt F0 St	DBLEMS OUT hon_socket -p arting Server



g.INFO, format="%(asctime)s %(levelname)s %(message)s")

## Bind the socket we created to the local server addr we defined

\_socket.accept()

```
from {client_addr}")
```

## Closed")

JUPYTER PUT

> bash + ∨ □ □ □ ∧ ×

				server.py — cs346	
ſ	EXPLORER ····	\$ run.sh	🕏 server.py 🛛 🕹		$\triangleright$ ~ $\square$
G	> OPEN EDITORS	<pre>server.py &gt;</pre>	••		
$\bigcirc$	$\sim$ CS346	1 from s	ocket <i>import</i> *		The world layer's new layer layer and layer and layer and layer and layer to the layer and layer and layer and layer memory and layer and layer and and layer and and layer and and layer and and and and and and and and
	\$ run.sh	2 import	logging		(mm, that, and y, same, same, same, same, so may a same same same same same same same sa
90	erver.py	3			
6		4 loggin	<pre>ig.basicConfig(level=</pre>	logging.INFO, format="%(asctime)s %(levelname)s %(message)s")	
		5			
±>		6 loggin	<pre>ig.info("Starting Ser</pre>	ver")	
		7		LISTEN ON UNS SOCKEL	
		8 server	SOCKET = SOCKET()	90)	
		9 Server	auur = (0.0.0.0,	$\frac{\partial \theta}{\partial t} = \frac{1}{2} \int \frac{d^2 t}{dt} dt = \frac{1}{2} \int d^$	
HD HD		11 server	_socket listen(5)	5 IS the number of backlog	
		12		connections to accept before	ρ
л		13 while	True:		
A		14 (c	:onn, client_addr) =	server_socket.acce the server starts refusing	
		15 wi	th conn:	connections	
		16	logging.info(f"Con	nection from {clie	
		17	while True:		
		18	data = conn.re	cv(1024)	
		19	<i>if</i> not data:		
		20	break		
		21	Logging.into(d	ata)	
			Logging.inio("Conn	ection (losed")	
		DEBUG CONSOLE	TERMINAL PROBLEM	S OUTPUT JUPYTER > bash +	~ 🔟 🛍 🥎
Q		• ~/cs346 \$ ./ru + docker run - 2022-09-11 01:	un.sh -irmname python_so :45:07,636 INFO Starting	ocket -p 8080:80 -v /Users/mark/cs346/:/app python:3.9-alpine python /app/serve: g Server	r.py



•••							se
ŋ	EXPLORER	•••	\$ run.s	h	🕏 server.py	, ×	
	> OPEN EDITORS		🕏 serv	er.py > .	•••		
$\bigcirc$	$\sim$ CS346		1	from s	socket <i>impor</i>	t *	
$\sim$	\$ run.sh		2	import	t logging		
0	Server by		3	-			
Po	Sciver.py		4	loggir	ng.basicConf:	ig(le	evel=logging
			5				
			6	loggir	ng.info("Sta	rtin	g Server")
~			7				
			8	servei	r_socket = so	ocket	t()
Γŵ			9	servei	r_addr = ("0	.0.0	.0", 80)
			10	servei	r_socket. <mark>bin</mark>	d(se:	rver_addr)
Б			11	servei	r_socket.list	ten(!	5)
			12				
Д			13	while	True:		
			14	(0	conn, client	_add:	r) = server_
			15	wi	ith conn:		
			16		logging.i	nfo(:	E"Connectior
			17		while True	e:	
			18		data =	= coi	nn.recv(1024
			19		<i>if</i> not	t dat	ta:
			20		b	reak	
			21		loggiı	ng.i	nfo(data)
			22		logging.i	nfo(	Connection
			DEBUG • ~/cs34 + dock	CONSOLE	TERMINAL un.sh -irmname	PRO	DBLEMS OUT
			2022-0	9-11 01	:45:07,636 INF	-U St	arting Server
8			-				



•••				ser
ŋ	EXPLORER	\$ run.sh	🕏 server.py 🛛 🕹	
	<ul> <li>&gt; OPEN EDITORS</li> <li>~ CS346</li> <li>\$ run.sh</li> <li>? server.py</li> </ul>	<pre>     server.py &gt;     1</pre>	ocket <i>import</i> * logging g.basicConfig(leve g.info("Starting Se	l=logging erver")
		<ul> <li>8 server</li> <li>9 server</li> <li>10 server</li> </ul>	_socket = socket() _addr = ("0.0.0.0" _socket.bind(serve:	, 80) r_addr)
Ш		11 server 12 13 while	_socket.listen(5)	
		13 while 14 (c) 15 wi	onn, client_addr) = th conn:	= server_
		16       17         17       18         19       4         20       4         21       4         22       1	<pre>logging.info(f"Co while True:</pre>	onnection recv(1024 (data) nnection
8		• ~/cs346 \$ ./ru + docker run - 2022-09-11 01:	n.sh irmname python <u></u> 45:07,636 INFO Start:	_socket -p ing Server





••						serve
Ċ	EXPLORER	• • •	\$ run.sl	n 🕹 🛑	•	
	> OPEN EDITORS		🗬 serv	er.py > 🚺 👡 ≰	nc - v lc	ocall
Q	$\sim$ CS346		1	from socket	$nc v \perp$	
	\$ run.sh		2	import logg		
20	erver.pv		3	Hell	Lo Inere	
5			4	logging.bas ^C		
			5	~ \$		
$\sum$			6	logging.inf		
~			7			
			8	server_sock		
=∞			9	server_addr		
			10	server_sock		
Б			11	server_sock		
			12			
Д			13	while True:		
			14	(conn,		
			15	with co		
			16	log		
			17	whi		
			18			
			19			
			20	b	oreak	
			21	loggi	.ng <mark>.info(dat</mark> a	1)
			22	logging.i	.nfo("Connect	ion Cl
			DEBUG (	CONSOLE TERMINAL	PROBLEMS	OUTPL
			0 ~/cs346	5 \$ ./run.sh		

2022-09-11 01:45:07,636 INFO Starting Server 2022-09-11 01:46:08,425 INFO Connection from ('172.17.0.1', 56116) 2022-09-11 01:46:13,737 INFO b'Hello There\n' 2022-09-11 01:46:17,960 INFO Connection Closed 



 $(\mathbf{Q})$ 



## host 8080 ocalhost port 8080 [tcp/http-alt] succeeded!

losed")

JUPYTER JT

> bash + ∨  $\square$   $\square$  ∧ ×

```
+ docker run -i --rm --name python_socket -p 8080:80 -v /Users/mark/cs346/:/app python:3.9-alpine python /app/server.py
```









## JavaScript















HTTP







![](_page_26_Picture_0.jpeg)

![](_page_26_Figure_1.jpeg)

![](_page_27_Picture_0.jpeg)

![](_page_28_Picture_0.jpeg)

## Web Servers **The Datacenter Model**

![](_page_29_Figure_1.jpeg)

![](_page_29_Picture_4.jpeg)

![](_page_29_Picture_5.jpeg)

![](_page_30_Figure_0.jpeg)

# Web Servers Many Different Types

- Apache 2 httpd
- nginx (pronounced "Engine X")
- IIS
- Tomcat
- Jetty
- Gunicorn

# Web Servers Many Different Types

- Apache 2 httpd
- nginx (pronounced "Engine X")
- IIS
- Tomcat
- Jetty
- Gunicorn

## General Purpose HTTP Servers

# Web Servers Many Different Types

- Apache 2 httpd
- nginx (pronounced "Engine X")
- IIS
- Tomcat
- Jetty
- Gunicorn

Language Specific HTTP Servers

We've already used containers to run a web server in Homework 2

Let's look closer at what those port mappings mean

## docker run -it --rm -p 8080:80 hw02:latest

0

It works!

8080

localhost:8080

![](_page_35_Picture_1.jpeg)

• Maps port 8080 on your host to the container's port 80.

![](_page_35_Figure_3.jpeg)

![](_page_35_Picture_4.jpeg)

![](_page_35_Picture_5.jpeg)

![](_page_36_Picture_0.jpeg)

![](_page_37_Picture_0.jpeg)

![](_page_38_Picture_0.jpeg)

![](_page_39_Picture_0.jpeg)

- We can run multiple containers, all with the same internal port.
- We can't map the same port on the host to multiple containers!

![](_page_40_Picture_3.jpeg)

 We need separate ports on the host for each container we want to forward traffic to

![](_page_41_Picture_2.jpeg)

- Not all containers need their ports mapped to the host
- Containers can also talk to each other directly, without having to leave the internal docker network

![](_page_42_Figure_3.jpeg)

Up Next: Javascript!