INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

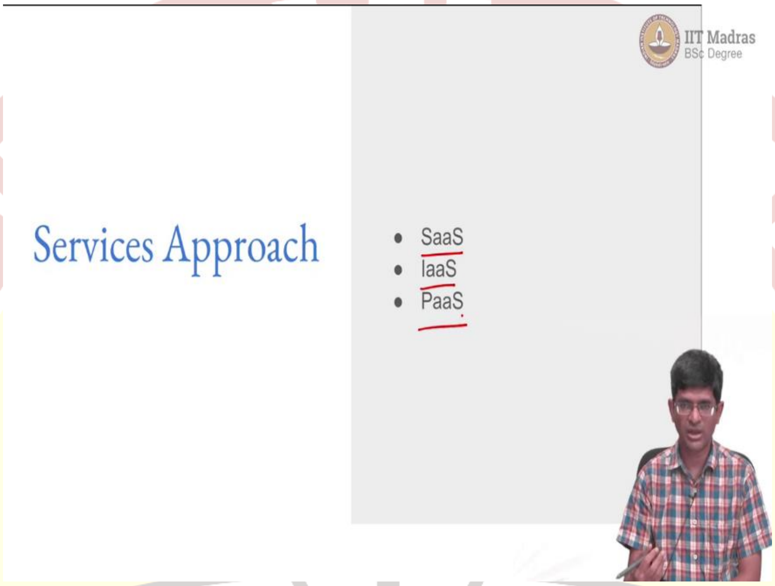
IIT Madras

ONLINE DEGREE

Modern application development-I
Professor Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Service Approach I

Hello everyone and welcome to this course on Modern Application Development.

(Refer Slide Time: 0:16)

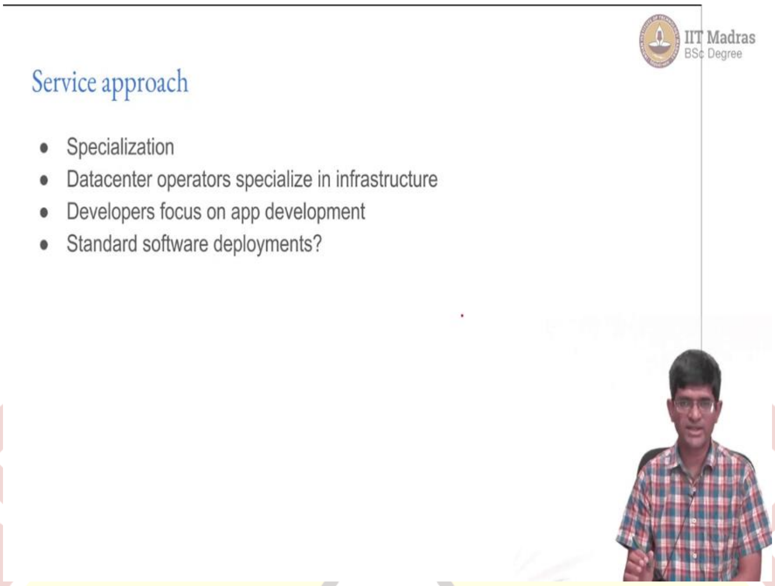


Services Approach

- SaaS
- IaaS
- PaaS

So, now that we have seen the basic components that you need in order to deploy an app. Let us understand a little bit about what I am going to call the services approach. And there are a number of acronyms that you come across here, SaaS, IaaS PaaS. What do these stand for? Why are they useful? Why do you need to know about them? And how can you make use of some of these techniques is what we are going to look at. And I will also be showing a couple of examples.

(Refer Slide Time: 0:43)



Service approach

- Specialization
- Datacenter operators specialize in infrastructure
- Developers focus on app development
- Standard software deployments?

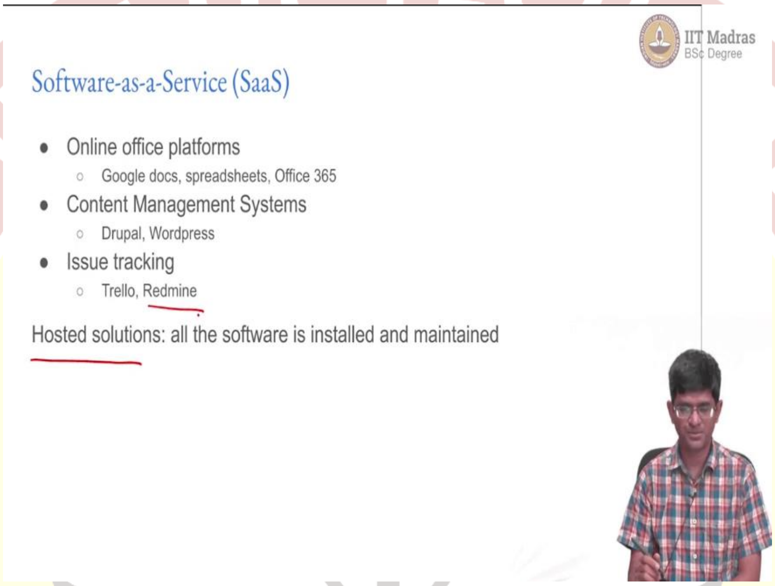
So, the service approach essentially says that rather than one person trying to do everything that is you build your own computer, you install it somewhere, you provide a UPS, you get a network connection, you then monitor it, you run, install the operating system, do everything on it. Let us it makes sense to sort of specialize.

So, what happens over there is datacenter operators then specialize in what is called infrastructure. The actual the physical computers. So, what they do is datacenter specialists who would say I can set up a datacenter. What does that mean? It means that I will have a building. The building will have appropriate cooling and security to make sure people cannot just walk in as they like. The building itself will be monitored properly. There will be all kinds of protections against fire and other hazards and then what do I need to do? I need to provide uninterrupted power and a good network connection.

Once I have that uninterrupted power and a good network connection, it means that I can just put a bunch of computers out there. I do not care what is running on those machines. I do not know what is running on those machines. That is for some else to deal with. So, the datacenter operator specializes in just the infrastructure, which means that the developers can now focus on the app because as far as a developer is concerned, somebody else has taken care of setting up the room, the cooling, the network, the power, and you can assume that you have a server which is always on and has a permanent IP address that can be reached from somewhere and all that you need to do is write your app.

Now, you could go one step further than that and say look not only am I writing my own, I might end up using some kind of standardized systems and maybe there is some standard software that others have written and I want to make it available for other people to use. So, I just install it, make sure that is properly set up and upgraded and so on and others can just use it. So, this whole idea of services that you find out what are all the tasks involved in getting something done and then find out who can specialize in any given part of the task.

(Refer Slide Time: 03:01)



Software-as-a-Service (SaaS)

- Online office platforms
  - Google docs, spreadsheets, Office 365
- Content Management Systems
  - Drupal, Wordpress
- Issue tracking
  - Trello, Redmine

Hosted solutions: all the software is installed and maintained

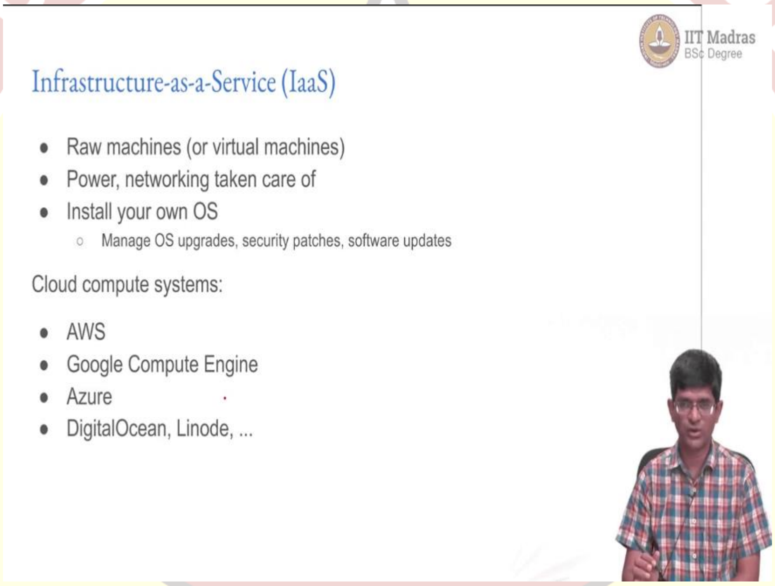
So, one of the first things that came along over there was the so-called Software-as-a-Service and Software-as-a-Service the examples that you are probably most familiar with are things like Google docs, Gmail, Google spreadsheets, Forms, Microsoft Office 365 and you could also have some kind of content management systems, Drupal, WordPress. I mean there are people who actually host those, meaning that they completely install the software for you. All that you need to do is create an account and use it.

So, what does that mean? It means that you want to use certain software. You do not want to install everything or run the code or anything on your local machine. Somebody has already set up the system for you. They have got a datacenter, multiple datacenters. They have installed the software. You just need to create an account and if necessary pay for it.

Now, this essentially is something called hosted solutions. You might also find that something like Redmine which is used for tracking issues or GitLab, which is used for version control. They have versions that you can install on your own. So, you can actually have the entire thing and say I will install it on my own machine and run the entire thing. The alternative is they will run it for you. So, GitLab.com for example is something that you can directly create an account on and use it. It provides a number of services for you. It has the gate interface. It has all these merge requests, pull requests, all of that stuff comes along with the interface and they make sure that it is kept secure and up-to-date.

Now, the alternative would have been that you could have downloaded it from their website and installed it on your own. The point is that by having all the software installed and maintained by someone else, they are taking care of making sure that it is up-to-date and everything else is done for you.

(Refer Slide Time: 4:56)



IIT Madras
BSc Degree

Infrastructure-as-a-Service (IaaS)

- Raw machines (or virtual machines)
- Power, networking taken care of
- Install your own OS
  - Manage OS upgrades, security patches, software updates

Cloud compute systems:

- AWS
- Google Compute Engine
- Azure
- DigitalOcean, Linode, ...

Now, at the other end, you could have what is called Infrastructure-as-a-Service, IaaS. Infrastructure-as-a-Service since we have been using the word infrastructure to refer to what is sometimes called the bare metal or just the raw computers themselves. Essentially what have you viewed over here would be virtual machines or the actual physical machines themselves, set-up and some datacenter and some. Power networking etc. are taken care of but that is it.

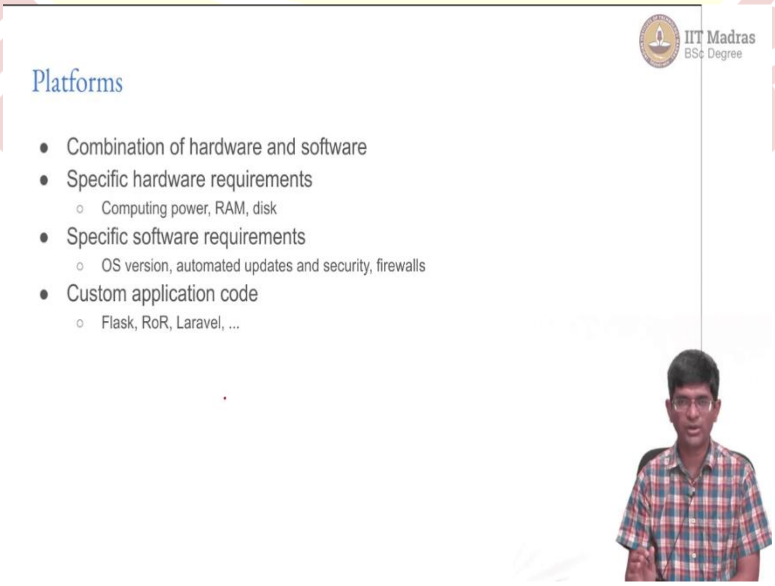
Everything else is your job, from installing an operating system, managing the operating system upgrade, security patches, software updates. All of that stuff is left to you. Pretty much what they say is look rather than buying a computer and putting it under your desk and plugging it in, you can just assume that the computer is taken care of somewhere else but everything else, what you put on it, how you manage it, do you mess it up, make it crash by running bad software on it, all that is your problem. All that the IaaS provider is doing is giving you a machine with some kind of an interface to control it.

There are many cloud compute systems, AWS, Amazon Web Services was start of the more or less the first one to really start this off in a big way but of course Google Compute Engine, Azure, Microsoft, all of these are big players in this area. But apart from that there are also some others, DigitalOcean, Linode and so on and some of them give you actually better options in terms of price to performance and so on.

So, Digital Ocean has some nice options, which can end up being cheaper than some of the Google or AWS offerings. A simpler interface and few other options that actually make it sort of nicer to use in some ways but it depends I mean so Google is not providing those services because they think that DigitalOcean does a better job. Google could also do it. The point is that they are focusing on other aspects. They have a slightly different way of providing these services to you.

So, which one should you choose? What should you do if you are trying to do this? It depends. You need to sort of familiarize yourself with a few of these, look at your options and then decide how to go about it. Now, all of this is assuming that you want to go with the so-called Infrastructure-as-a-Service that is to say you are willing to work with raw machines. Somebody just gives you a machine, you will take care of installing everything that you need on it. So, what is the alternative?

(Refer Slide Time: 7:25)



Platforms

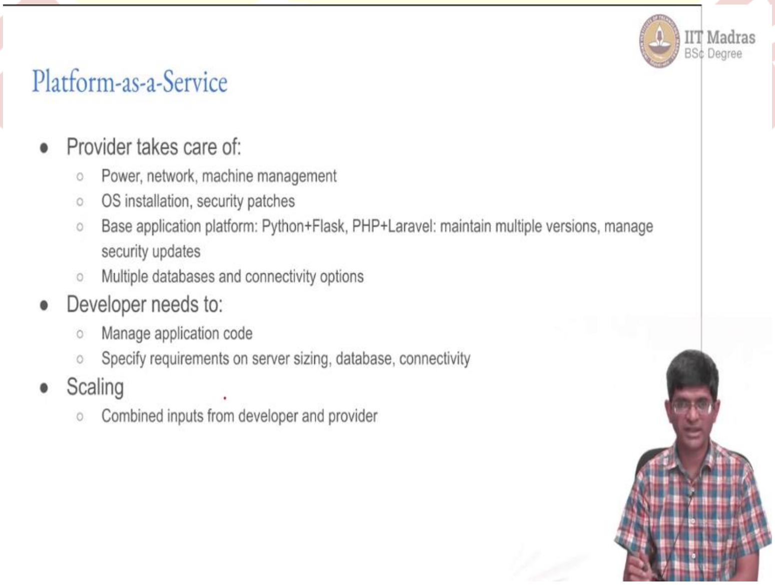
- Combination of hardware and software
- Specific hardware requirements
  - Computing power, RAM, disk
- Specific software requirements
  - OS version, automated updates and security, firewalls
- Custom application code
  - Flask, RoR, Laravel, ...

We can look at something called Platforms and a Platform is a combination of hardware and software. What we are saying is look I need hardware of certain capacity that is I probably need so much memory. I need so much CPU speed, so much disk space and so on but why do I need that because I want to run a specific kind of software. So, this particular operating system maybe Linux, Ubuntu 20.04 or Red Hat Enterprise Linux some version or maybe Microsoft Windows with some particular version. All of those have different options that you might have as your requirement of what is the software to be installed as part of your platform.

In addition to that you might also say that I want a specific development framework. I need python, a specific version of Python or I need Python with this library. So, for example, I want to develop a Flask application. So, I need a specific version of Python. I need a specific version of Python Flask and a specific version of SQLAlchemy and whatever else is required for my app or maybe I am planning to develop in Laravel. So, I need PHP. I need a web server that can interface with PHP. All of those things need to be there.

So, the simple solution of course would be you go with the Infrastructure-as-a-Service approach, get a machine, install whatever you want. The other alternative is to say that somebody will actually set things up for you. All the way up to saying that we will provide you with a web server that has an interface to PHP and this version of PHP this version of Laravel, now write your code, all that you need to do is push your code and it will work. So, that is idea for platform.

(Refer Slide Time: 9:10)



Platform-as-a-Service

- Provider takes care of:
  - Power, network, machine management
  - OS installation, security patches
  - Base application platform: Python+Flask, PHP+Laravel: maintain multiple versions, manage security updates
  - Multiple databases and connectivity options
- Developer needs to:
  - Manage application code
  - Specify requirements on server sizing, database, connectivity
- Scaling
  - Combined inputs from developer and provider

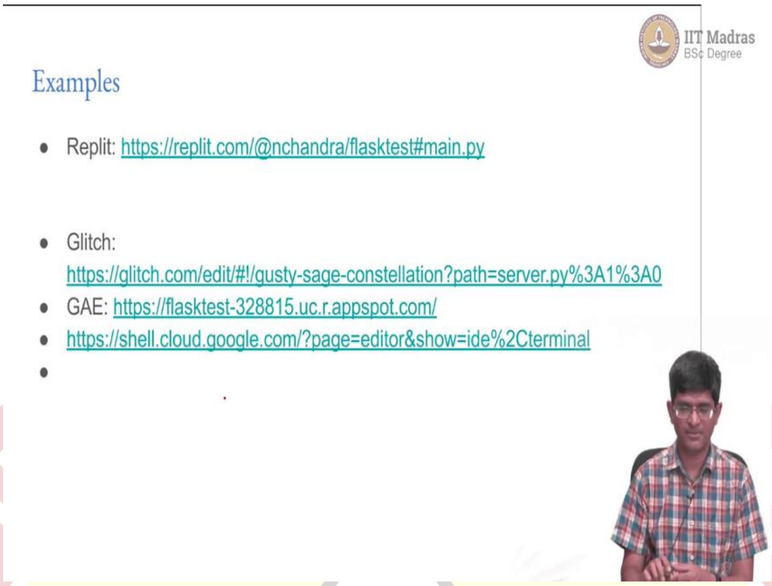
And the Platform-as-a-Service means that the provider takes care of multiple things, not just the power network, machine management, which is infrastructure. They also do OS installation, security patches, which is part of software and they provide you with some kind of a base application platform. It could be Python Flask, PHP Laravel. They will manage multiple versions for you, Python 2, Python 3, Python 3.8, 3.9, various different things whatever is suitable so that you can then decide and say this is the particular option that bests suits my needs.

They also typically provide some web which you can have database connectivity because you do not necessarily want to be installing and managing your own MySQL database or your own PostgreSQL database or you might even want to go with something like a cloud based storage, Redis, MongoDB, Firebase a whole bunch of options that are available for you.

So, what does a developer need to do? Manage the application code. You just concentrate on getting your app correct. And then you need to specify to the Platform-as-a-Service provider what kind of server size you need, what kind of database connectivity you need, and they will give you information about how to actually set things up.

And the nice thing over here is when you have this nice interaction between the developer and the service provider scaling can also be taken care of by sharing inputs between the 2 sides. So, the developer can say when do I want to scale? When do I need a new machine? And the provider can say what are your options? Do you need to deploy a new machine? Can you have a frontend that will take care of part of the processing for you? Can you move to a different kind of database? What are all the options that you have with this setup?

(Refer Slide Time: 10:56)

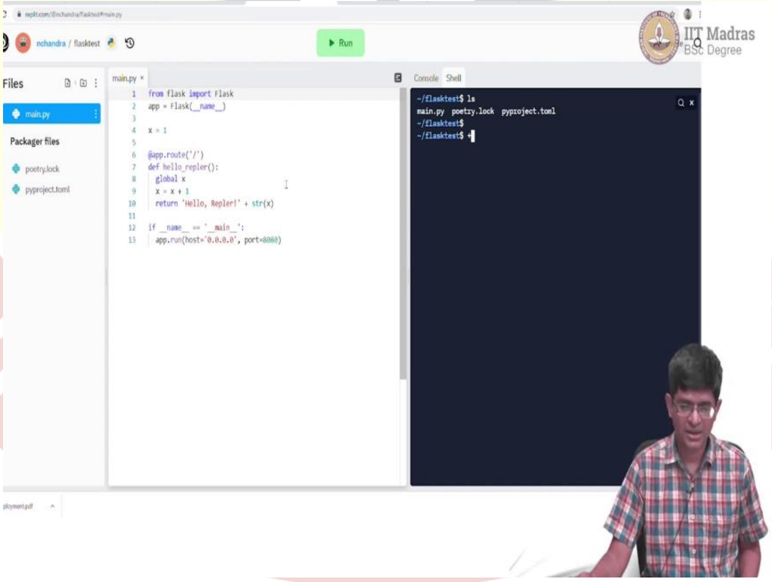


Examples

- Replit: https://replit.com/@nchandra/flasktest#main.py
- Glitch: https://glitch.com/edit/#!/gusty-sage-constellation?path=server.py%3A1%3A0
- GAE: https://flasktest-328815.uc.r.appspot.com/
- https://shell.cloud.google.com/?page=editor&show=ide%2Cterminal
-

So, what I am going to do now is to show a couple of examples of how you could use Replit which you are already familiar with but also a couple of other things and what the interfaces look like and what they imply in terms of deployment.

(Refer Slide Time: 11:15)



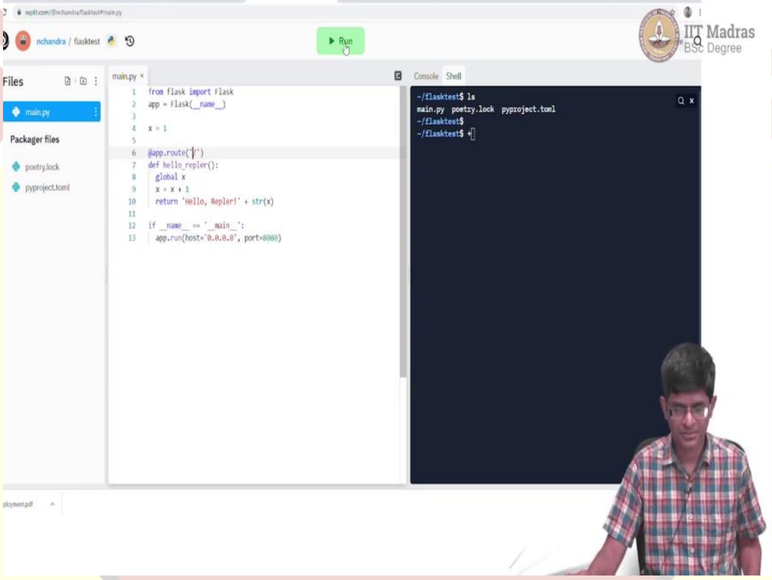
```
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

```
~/flasktest$ ls
main.py  poetry.lock  pyproject.toml
~/flasktest$
~/flasktest$ 
```

So, this is the Replit interface that most of you should be familiar with. All of you hopefully are familiar with at this point because you have been using it throughout this course and possibly for other courses in the diploma program as well. And what we have over here as you know I have created a very trivial flask application. It pretty much consists of just as one file main.py. But there are also a couple of other files poetry.lock. Poetry is a sort of package manager which is used by Replit and what they do is they pretty much just find out what packages you need and thereby figure out what needs to be installed over here.

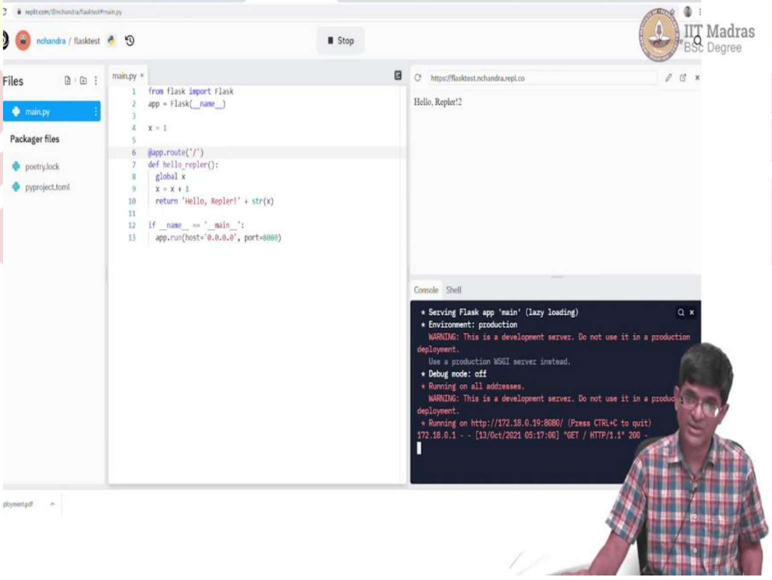
And the code itself is very trivial. I am just giving you sort of dummy example out here. All it says is from Flask, import Flask and I have defined one global variable in this server x and inside my main root, which is to /. All I am doing is that I declare x as a global, do x = x + 1 and then basically it say print out a string. Hello Repler and whatever that string is then I basically go ahead and run it with name = main.

(Refer Slide Time: 12:30)



```
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

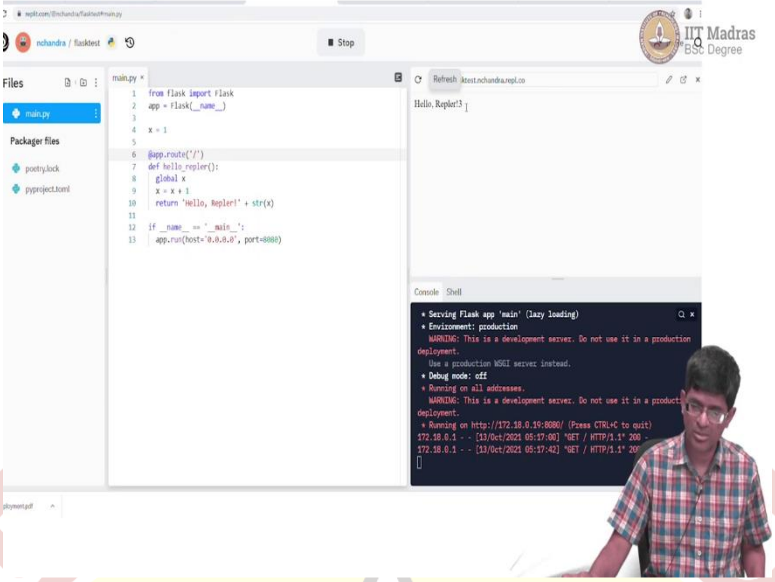
```
~/flasktest$ ls
main.py  poetry.lock  pyproject.toml
~/flasktest$
~/flasktest$ +
```



```
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!2

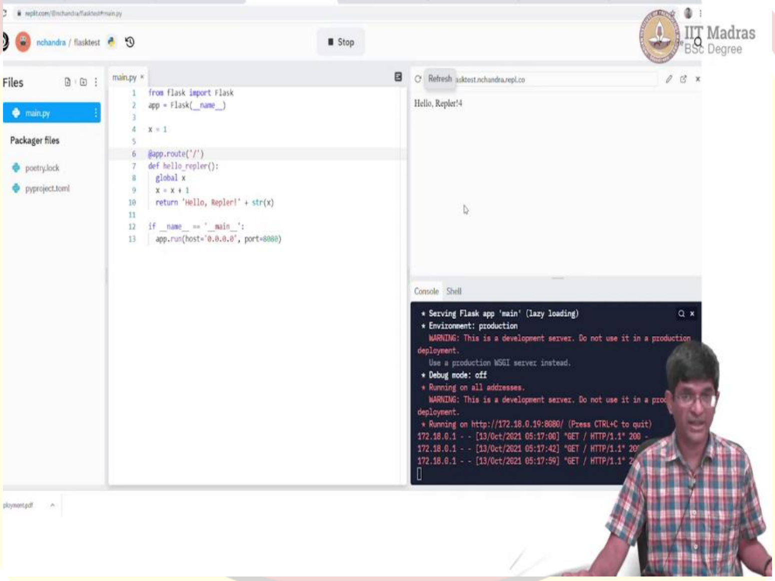
```
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a produc
deployment.
 * Running on http://172.18.0.19:8080/ (Press CTRL+C to quit)
172.18.0.1 - - [13/Oct/2021 05:17:00] "GET / HTTP/1.1" 200 -
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

```
Hello, Repler!3
```

```
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a product
deployment.
 * Running on http://172.18.0.19:8080/ (Press CTRL+C to quit)
172.18.0.1 - - [13/Oct/2021 05:17:00] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [13/Oct/2021 05:17:42] "GET / HTTP/1.1" 20
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

```
Hello, Repler!4
```

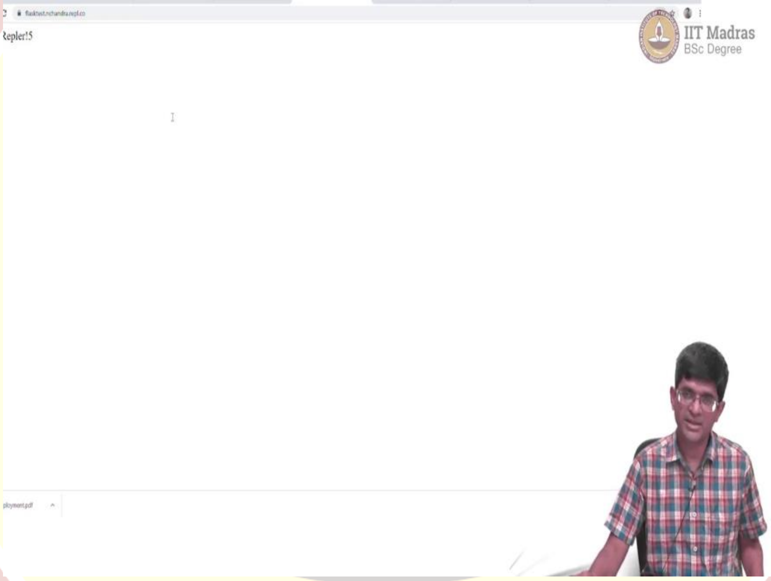
```
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a proc
deployment.
 * Running on http://172.18.0.19:8080/ (Press CTRL+C to quit)
172.18.0.1 - - [13/Oct/2021 05:17:00] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [13/Oct/2021 05:17:42] "GET / HTTP/1.1" 20
172.18.0.1 - - [13/Oct/2021 05:17:59] "GET / HTTP/1.1" 2
```

What happens when I run this code? So, you can see that immediately in the console a few things start happening out there. There are a bunch of messages that pop up saying what is actually happening. It also says this is a development server and so on. I do not use it in a production deployment. All good. And you can see that at the end of the day it has made a request to get / and yes it ended up creating this particular thing. I had called my Repl Flask test. My user name over here as nchandra and therefore flasktest.nchandra.repl.co comes up out here.

So, what does it print? Of course, it prints out hello Repler plus this string, which is x equal to 1. Now, what that in turn means if I refresh this page and send one more request to it, it now shows Hello Repler 3. Why did that happen? Because of course x, which is sitting inside the server at this point is a global variable and is getting incremented. So, every time I hit this page and refresh it, it is basically going to go in there. The server is still alive. I mean the server has once it was started by Replit, it basically retains its state internally. It does not know who is connected to it or where I am connecting from. But it knows that internally this variable x has the value at this point 4. And every time there is a request to the /, it has to display this Hello Repler and basically increment that variable.

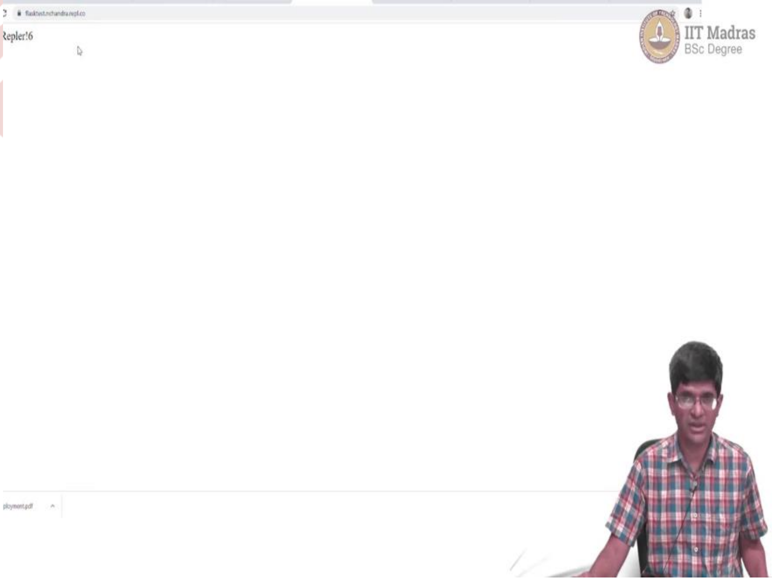
So, that is what is happening out here. If I copied this URL and went and visited it somewhere else in another browser even it would show me exactly the appropriate thing Hello Repler and that number would keep on increasing because ultimately all those requests are going back to the same server. So, I can in fact do that.

(Refer Slide Time: 14:30)



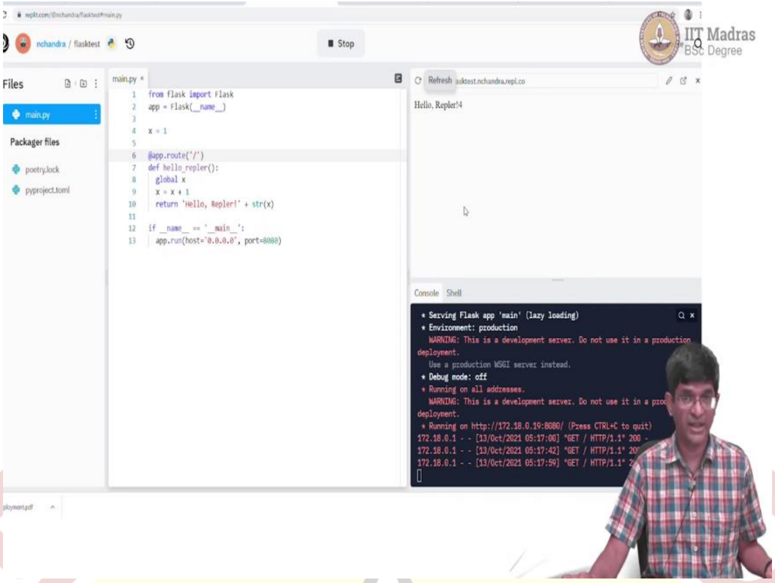
flasktest.nchandra.repl.co

Repler!5



flasktest.nchandra.repl.co

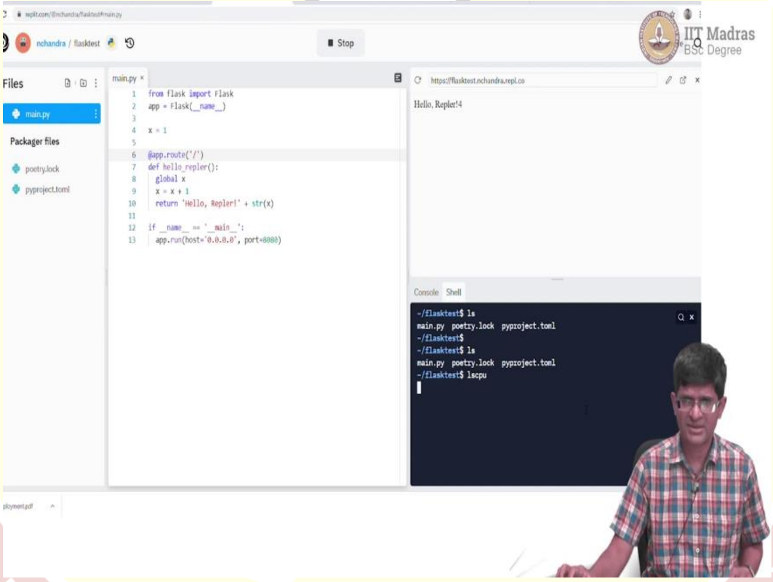
Repler!6



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

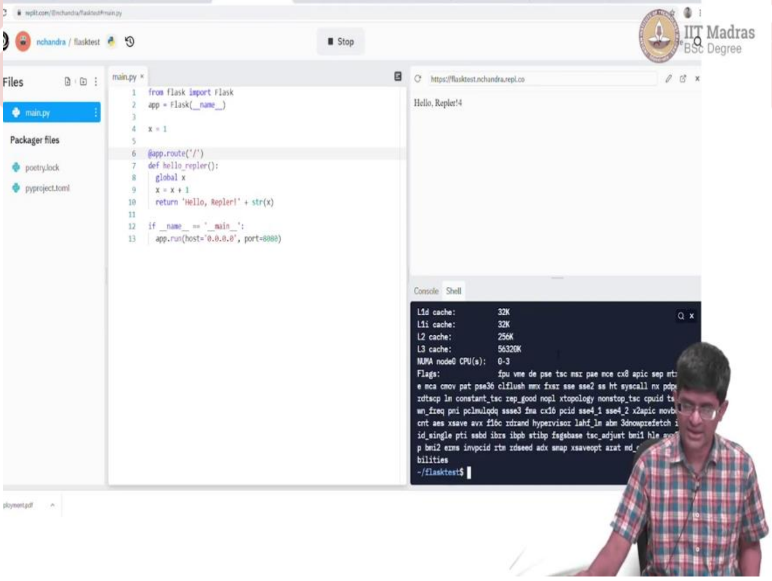
```
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a proc
deployment.
 * Running on http://172.18.0.19:8080/ (Press CTRL+C to quit)
172.18.0.1 - - [13/Oct/2021 05:17:00] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [13/Oct/2021 05:17:42] "GET / HTTP/1.1" 20
172.18.0.1 - - [13/Oct/2021 05:17:59] "GET / HTTP/1.1" 2
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

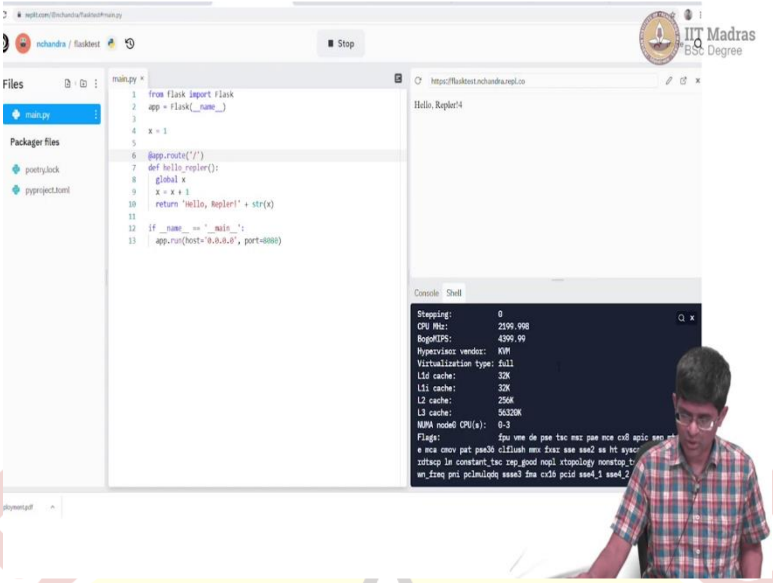
```
~/flasktest$ ls
main.py  poetry.lock  pyproject.toml
~/flasktest$
~/flasktest$ ls
main.py  poetry.lock  pyproject.toml
~/flasktest$ lscpu
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

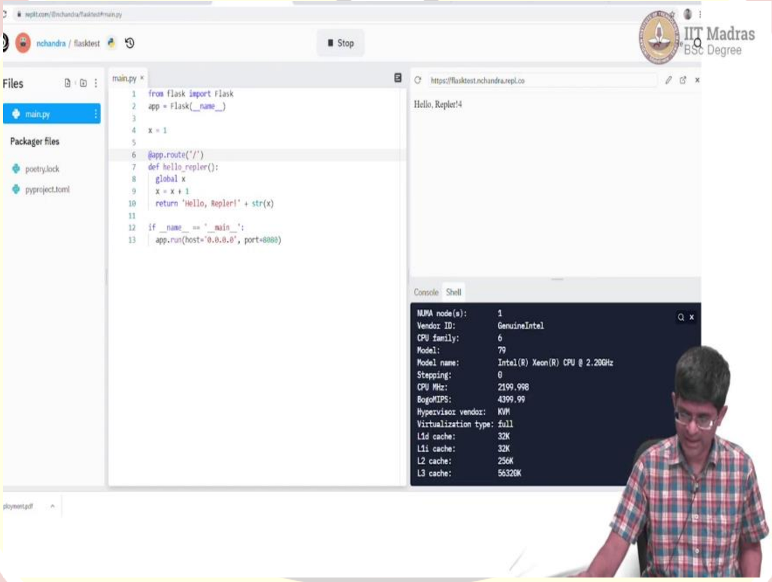
```
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            56320K
NUMA node0 CPU(s):   0-3
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtr
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid ts
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe
cnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch i
id_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 hle avx
p bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat md_c
bilities
~/flasktest$
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

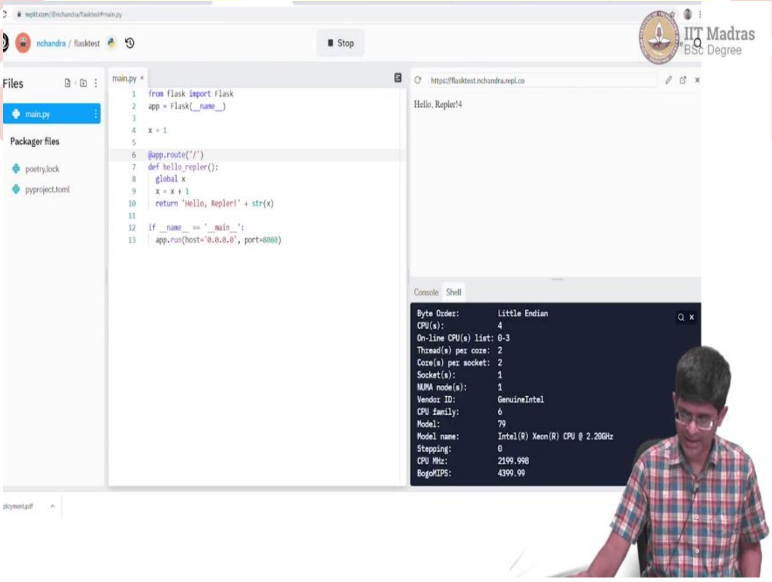
```
Stepping:            0
CPU MHz:             2199.998
BogoMIPS:            4399.99
Hypervisor vendor:   KVM
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            56320K
NUMA node0 CPU(s):   0-3
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mt
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht sysc
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_t
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

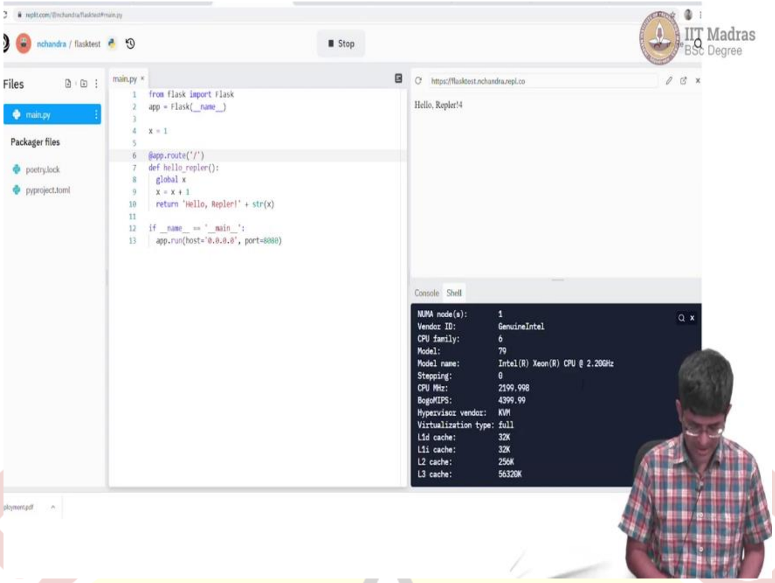
```
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               79
Model name:          Intel(R) Xeon(R) CPU @ 2.20GHz
Stepping:            0
CPU MHz:             2199.998
BogoMIPS:            4399.99
Hypervisor vendor:   KVM
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            56320K
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

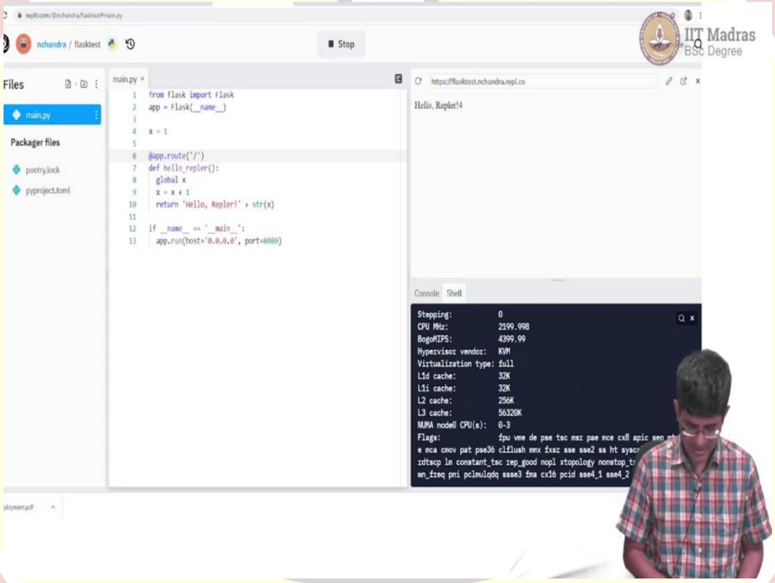
```
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  2
Core(s) per socket:  2
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               79
Model name:          Intel(R) Xeon(R) CPU @ 2.20GHz
Stepping:            0
CPU MHz:             2199.998
BogoMIPS:            4399.99
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

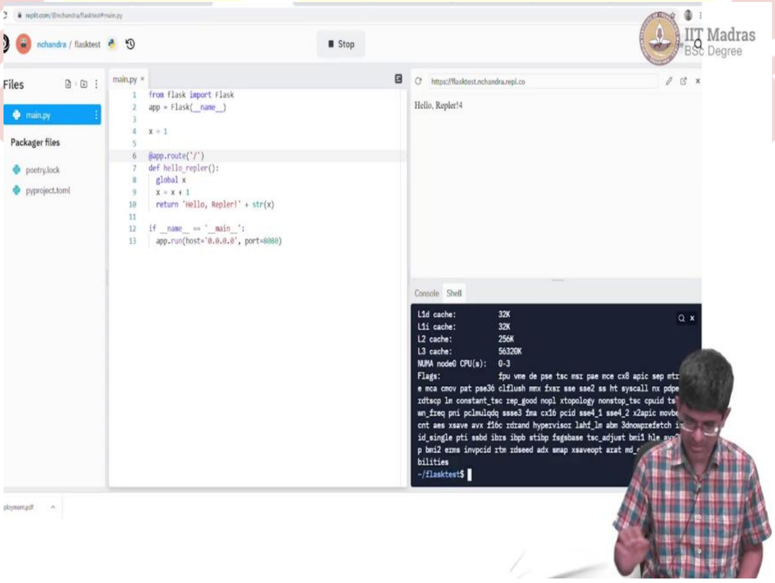
```
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               79
Model name:          Intel(R) Xeon(R) CPU @ 2.20GHz
Stepping:            0
CPU MHz:             2199.998
BogoMIPS:            4399.99
Hypervisor vendor:   KVM
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            56320K
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

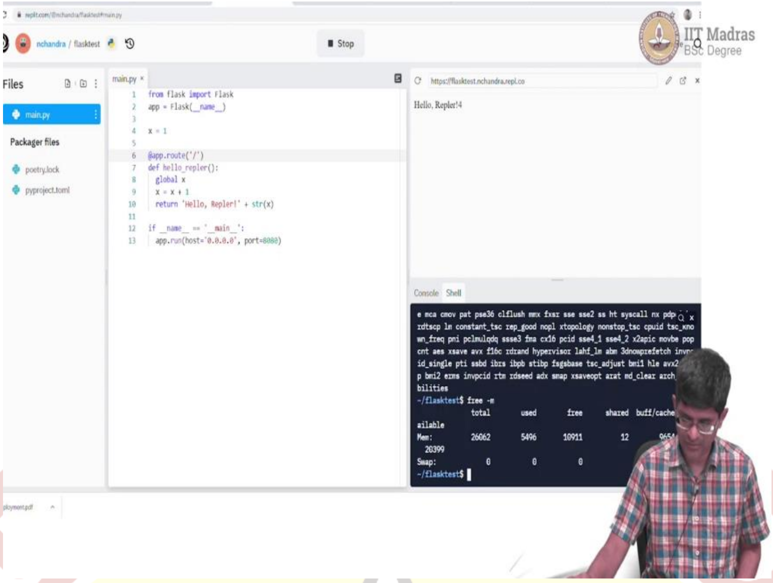
```
Stepping:            0
CPU MHz:             2199.998
BogoMIPS:            4399.99
Hypervisor vendor:   KVM
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            56320K
NUMA node0 CPU(s):   0-3
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mt
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht sysc
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_t
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

```
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            56320K
NUMA node0 CPU(s):   0-3
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtr
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid ts
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe
cnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch i
id_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 hle av
p bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat md_
bilities
~/flasktest$
```



```python
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

```
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdp
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_kno
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe pop
cnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invp
id_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 hle avx2
p bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat md_clear arch_
bilities
~/flasktest$ free -m
              total        used        free      shared  buff/cache
ailable
Mem:          26062        5496       10911          12        9654
  20399
Swap:             0           0           0
~/flasktest$
```

I can just basically open it in another page out here. So, as you can see now this has opened completely in a new Chrome tab. And in the top left corner, you can see there is Hello Repler with the value 5 over there now, so factor 5. Zoom in a little bit, you can sort of see the text a little bit bigger and each time I sort of reload this page, that is going to show me an updated version of the text over there because it is going back and fetching instruction fetching the data from the same server.

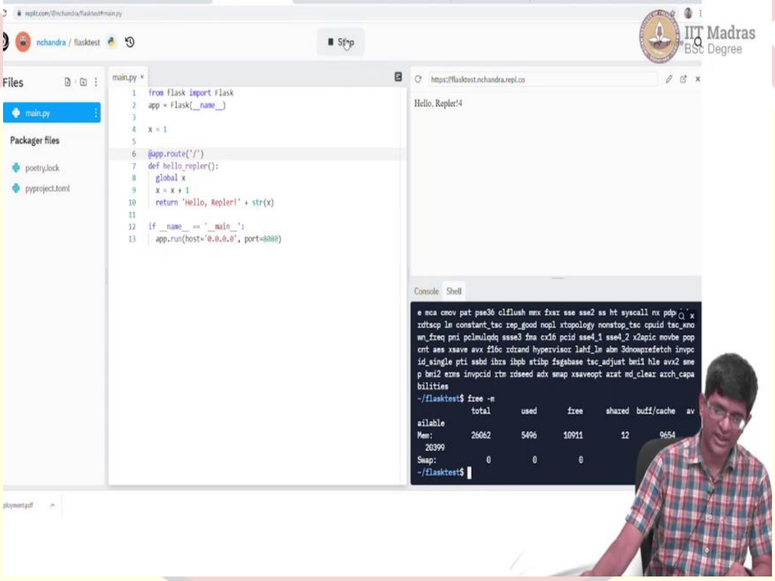
Now, what does this mean? When I go back to Repler, I can see that apart from the console which is showing me all the requests that have come here so far, I also have access to a Shell. And the Shell then I go there and start looking at what is available to me. I realized that this is actually a Linux machine. So, I can type for example LS which tells me that there are 3 files present over here, the same files that are shown on the left hand side in the file manager and not just that I can find out a little bit more information about the server as well.

For example, I can type the command lscpu, which is a Linux system command, which tells me a little bit about what kind of processors are available on the system. And if I scroll up a little bit I basically find that it is running Intel Xeon CPUs and there are 4 CPUs available for the system and it also has some amount of memory. The memory that I have over here by typing in the free command, I find that I basically have 26 GB of RAM, 26,000 megabytes.

So, in other words what this is telling me is that Replit then I tried running this particular code automatically gave me a machine with 4 CPUs and 26 GB of RAM that sounds like it is like extremely generous of them and it is. I mean it is quite big chunk of resource to give you. The point is why they are able to do this that it has shared? So, even though it says 4 CPUs and 26 GB of RAM the way it gets implemented in practice is that this itself is shared among a large number of users.

And on their datacenter the backend of the datacenter what they have is a system with probably much more than 26 GB of RAM but it is partitioned into smaller blocks, which in turn are shared out among multiple users. And not just that, the way Replit works is the main purpose of Replit is not really to run apps. It is more for education, development. I mean to sort of learn how to build such systems, which means that after you have had the app running for a little while and then it goes away you will find that the app shuts down.

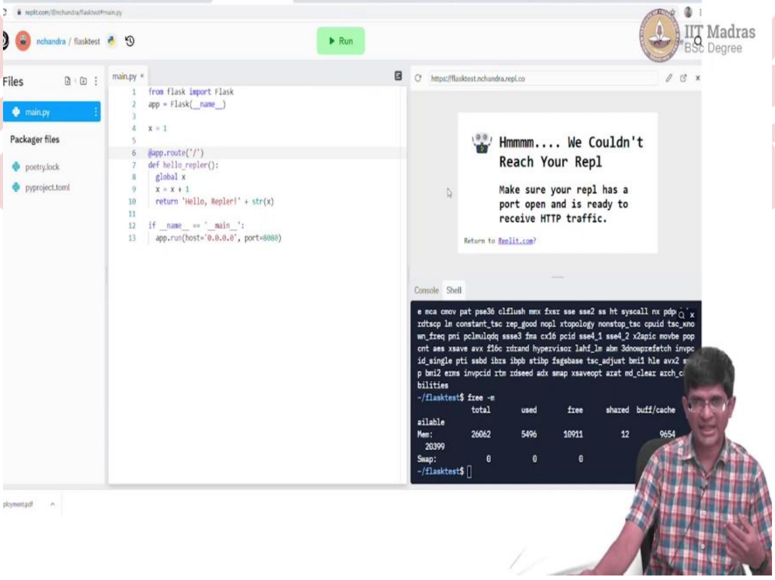
(Refer Slide Time: 17:26)



```
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hello, Repler!4

```
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdp
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_kno
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe pop
cnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpc
id_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 hle avx2 sme
p bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat md_clear arch_capa
bilities
~/flasktest$ free -m
              total        used        free      shared  buff/cache   av
ailable
Mem:          26062        5496       10911          12        9654
  20399
Swap:             0           0           0
~/flasktest$
```



```
from flask import Flask
app = Flask(__name__)

x = 1

@app.route('/')
def hello_repler():
  global x
  x = x + 1
  return 'Hello, Repler!' + str(x)

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=8080)
```

Hmmmm.... We Couldn't Reach Your Repl

Make sure your repl has a port open and is ready to receive HTTP traffic.

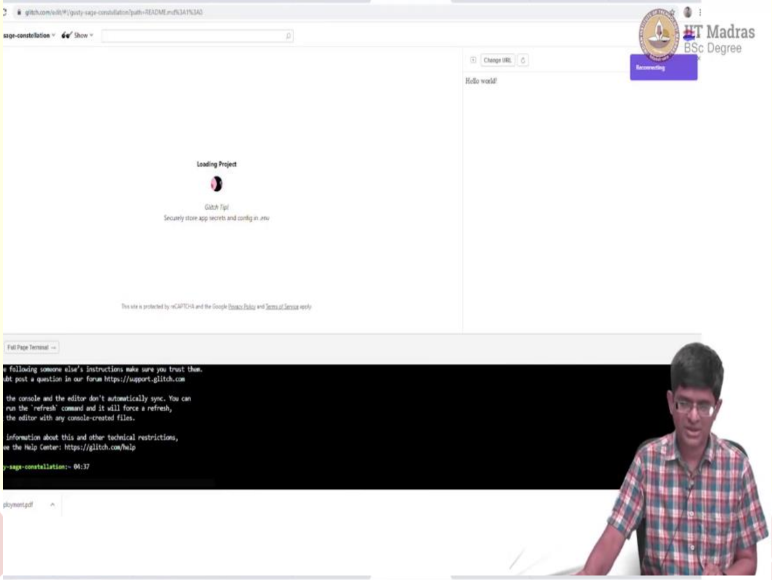
Return to Replit.com?

```
e mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdp
rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_kno
wn_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe pop
cnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpc
id_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 hle avx2 s
p bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat md_clear arch_c
bilities
~/flasktest$ free -m
              total        used        free      shared  buff/cache
ailable
Mem:          26062        5496       10911          12        9654
  20399
Swap:             0           0           0
~/flasktest$
```

So, if I now leave this running without clicking stop, I am going to click stop because I do not necessarily want it, just by using up resources on the Replit server, what will happen is that after a while, Replit will automatically shut it down. And at that point if I then go back to the URL it will once again start up my server, which means my global variable x will be reset back to 1. It will take some amount of time in order to start it up but otherwise I have the entire thing working the way that I expected it to.

So, Replit is one example of a Platform-as-a-Service in the sense that you could actually develop Flask apps. The main difference here is its main purpose is not deployment. So, if you want to actually deploy an app and start scaling it and so on you will find that it does not really have those kinds of resources.

(Refer Slide Time: 18:26)



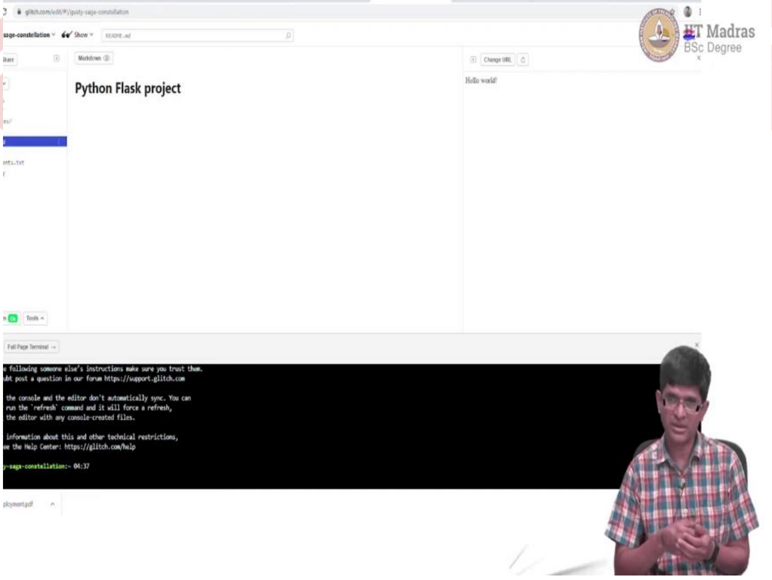
Loading Project

Glitch Tip!
Securely store app secrets and config in .env

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Hello world!

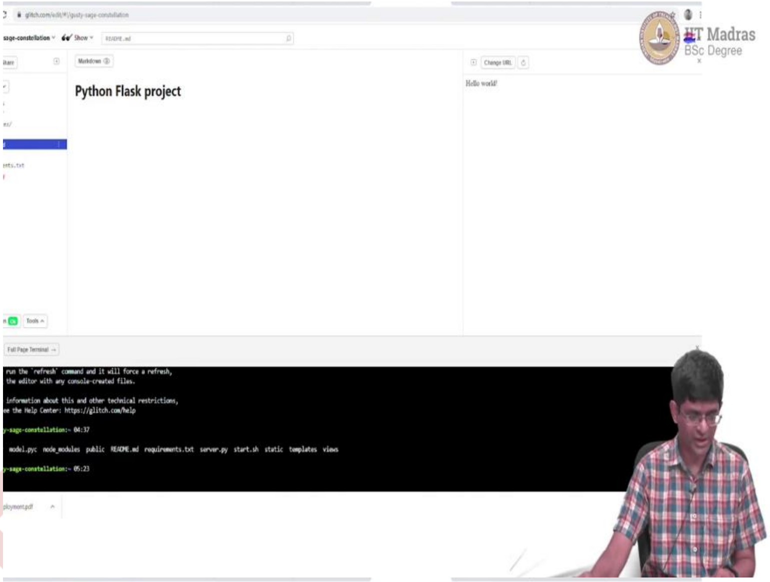
```
e following someone else's instructions make sure you trust them.
ubt post a question in our forum https://support.glitch.com

 the console and the editor don't automatically sync. You can
 run the `refresh` command and it will force a refresh,
 the editor with any console-created files.

 information about this and other technical restrictions,
ee the Help Center: https://glitch.com/help

y-sage-constellation:~ 04:37
```



Python Flask project

Hello world!

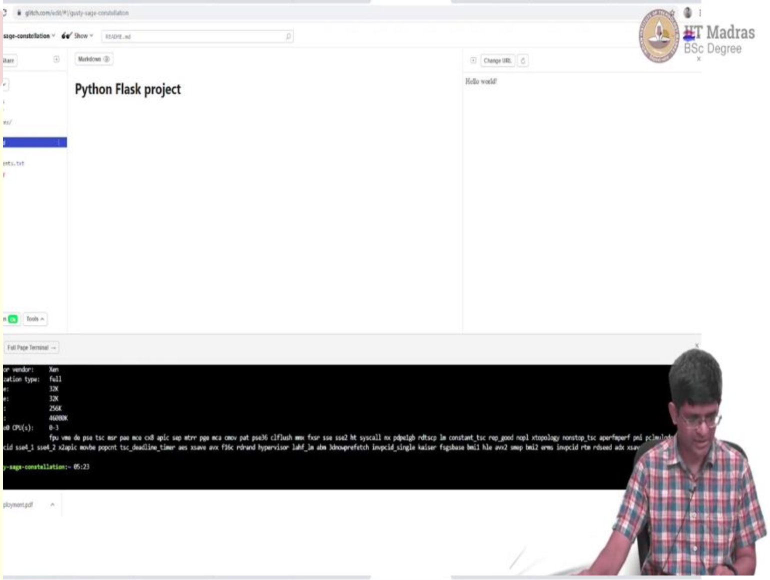
```
e following someone else's instructions make sure you trust them.
ubt post a question in our forum https://support.glitch.com

 the console and the editor don't automatically sync. You can
 run the `refresh` command and it will force a refresh,
 the editor with any console-created files.

 information about this and other technical restrictions,
ee the Help Center: https://glitch.com/help

y-sage-constellation:~ 04:37
```



Python Flask project

Hello world!

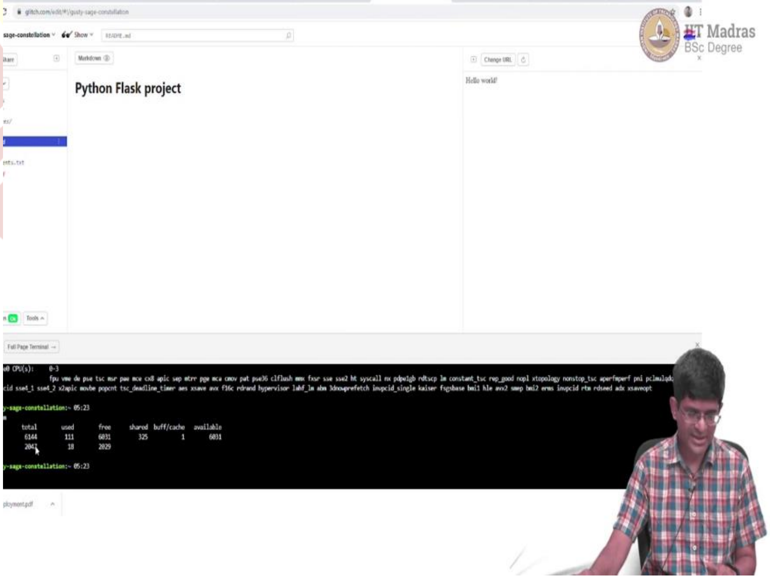
```
run the `refresh` command and it will force a refresh,
the editor with any console-created files.

information about this and other technical restrictions,
ee the Help Center: https://glitch.com/help

y-sage-constellation:~ 04:37

 model.pyc  node_modules  public  README.md  requirements.txt  server.py  start.sh  static  templates  views

y-sage-constellation:~ 05:23
```



Python Flask project

Hello world!

```
or vendor:     Xen
zation type:   full
e:             32K
e:             32K
:              256K
:              46080K
e0 CPU(s):     0-3
               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqd
cid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single kaiser fsgsbase bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx xsav

y-sage-constellation:~ 05:23
```



Python Flask project

Hello world!

```
e0 CPU(s):     0-3
               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
cid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single kaiser fsgsbase bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx xsaveopt

y-sage-constellation:~ 05:23

       total        used        free      shared  buff/cache   available
        6144         111        6031         325           1        6031
        2047          18        2029

y-sage-constellation:~ 05:23
```

Now, there is another option which is called Glitch. And glitch.com is something which once again is they provide a similar kind of interface to Replit. In Glitch also the main sort of core idea that they have is sort of to say let us make the development task easier. So, they provide you once again just like Replit. They have the set-up where you have a file manager. You also have access to the web interface or rather a Shell interface, which as you can see over there it has all of this thing. The LS command from Linux.

Lscpu over here shows that once again this is also a quad that is a 4 core Xeon processor and in this case the amount of memory that this processor has is 6 GB. So, does it mean Replit is better because they have 26 GB? No, I mean these are all meant for small applications in any case. They can scale when required and the point about Glitch is rather than focusing only on the development part of it, they also provide you some capabilities for how you could have an app always on for example and in some sense they are trying to go in a slightly different track than what Replit is doing.

Replit is focusing mostly on share development, education things of that sort. Having said that, they might also start providing facilities where you can actually have it always on and various other kinds of deployment options. Glitch is trying to focus more on developers and how they can share development of an app and actually deploy the basic starting point of an app.

Now, in both of these cases, what is the platform that they are providing? They are taking care of the Linux server for you. They are taking care of the networking, datacenter, all of that part. So, at some level it looks as though what they are doing is infrastructure but they are also providing a platform, in the sense that they have already installed Python. They have installed certain versions of flask. They give you starter templates that can be used in order to get started creating an application. And on the top of that the web server which automatically provides that URL for you that allows you to connect and see what the output is going to be is also taken care of by this same Replit or Glitch.

So, all of that part of the software infrastructure is also taken care by them which is why when you combine things. I mean you look at overall. You have infrastructure which is the hardware. You have some part of the software which is the web server, the Python version, the other dependencies that are required internally. All of those are already installed for you and what you have is a platform that has been provided.

Now, both of these are focusing more on the development aspect. What I mean by that is? They give you a nice interface whereby you can edit files directly in the browser. They also provide some mechanism whereby you can connect to it using either a version control system or directly download all the files, edit them on your local machine and then push it back up. So, that can also be done if necessary on these kinds of platforms.