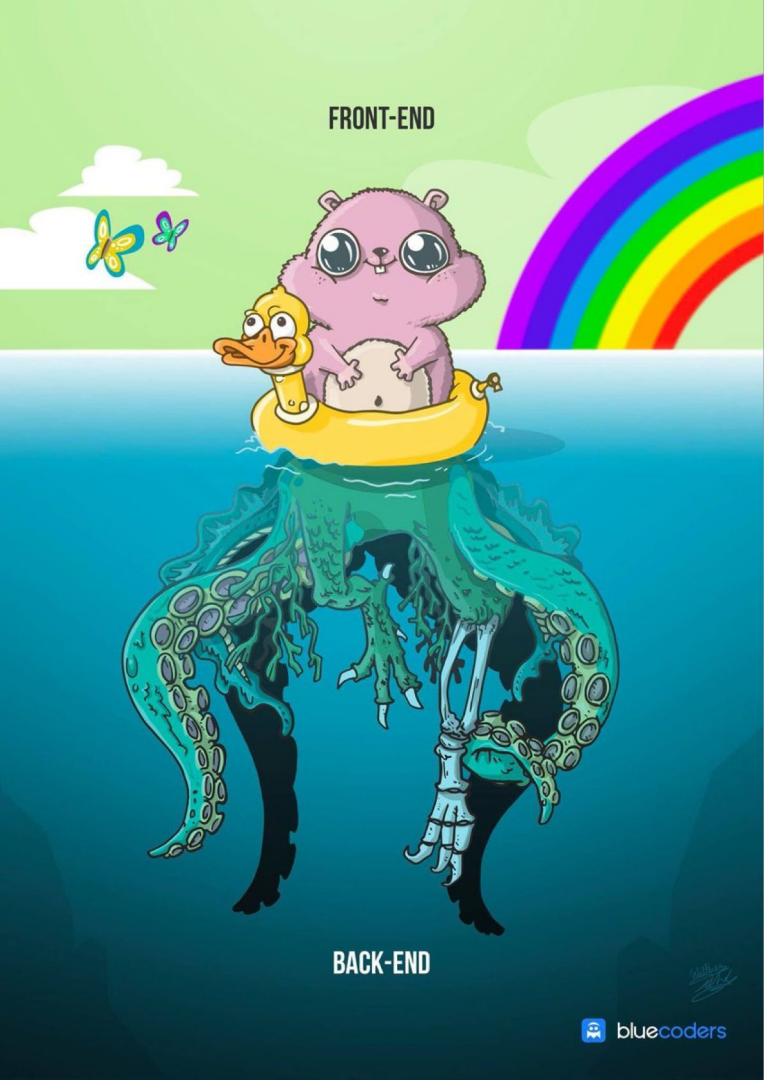


Backend Development

AGENDA

1. Introduction to Backend & Javascript

2. Building a Web Server & Communicating with Frontend

3. Connecting to a database

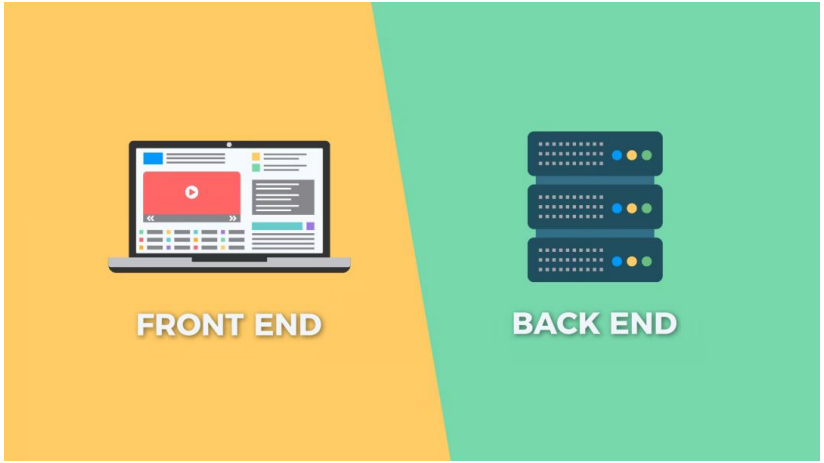
4. Deploying to the cloud

Prerequisites

- Install Node.js -> https://nodejs.org/en/download/
- Install MongoDb -> https://docs.mongodb.com/manual/installation/#mongodb-community-edition-installation-tutorials
- Install CURL -> https://curl.haxx.se/download.html
- Azure account: https://azure.microsoft.com/

What is a backend?

- Usually, applications are split into a frontend and a backend
- Front-end “what the user sees”, backend “what the user does not see”



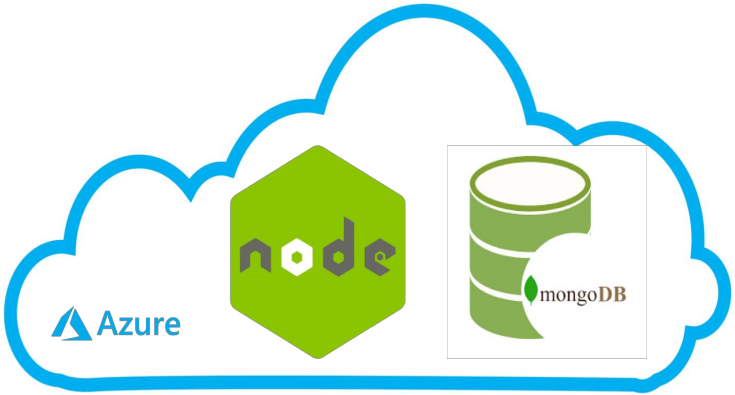
Why do we need a backend?

Frontend

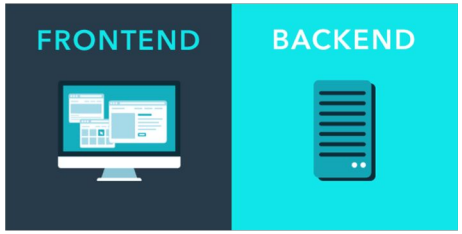
- Presentation (UI)
- Input data validation
- User session tracking:
  - Saving info about current user

Backend

- Heavy computation
- Persistent storage
- User authentication:
  - Verifying identity of user



Where do we put the logic?



Frontend

Pros

- Very responsive (low latency)

Cons

- Security (User can change the code)
- Performance (Dependent on client devices)
- Unable to share between front-ends

Backend

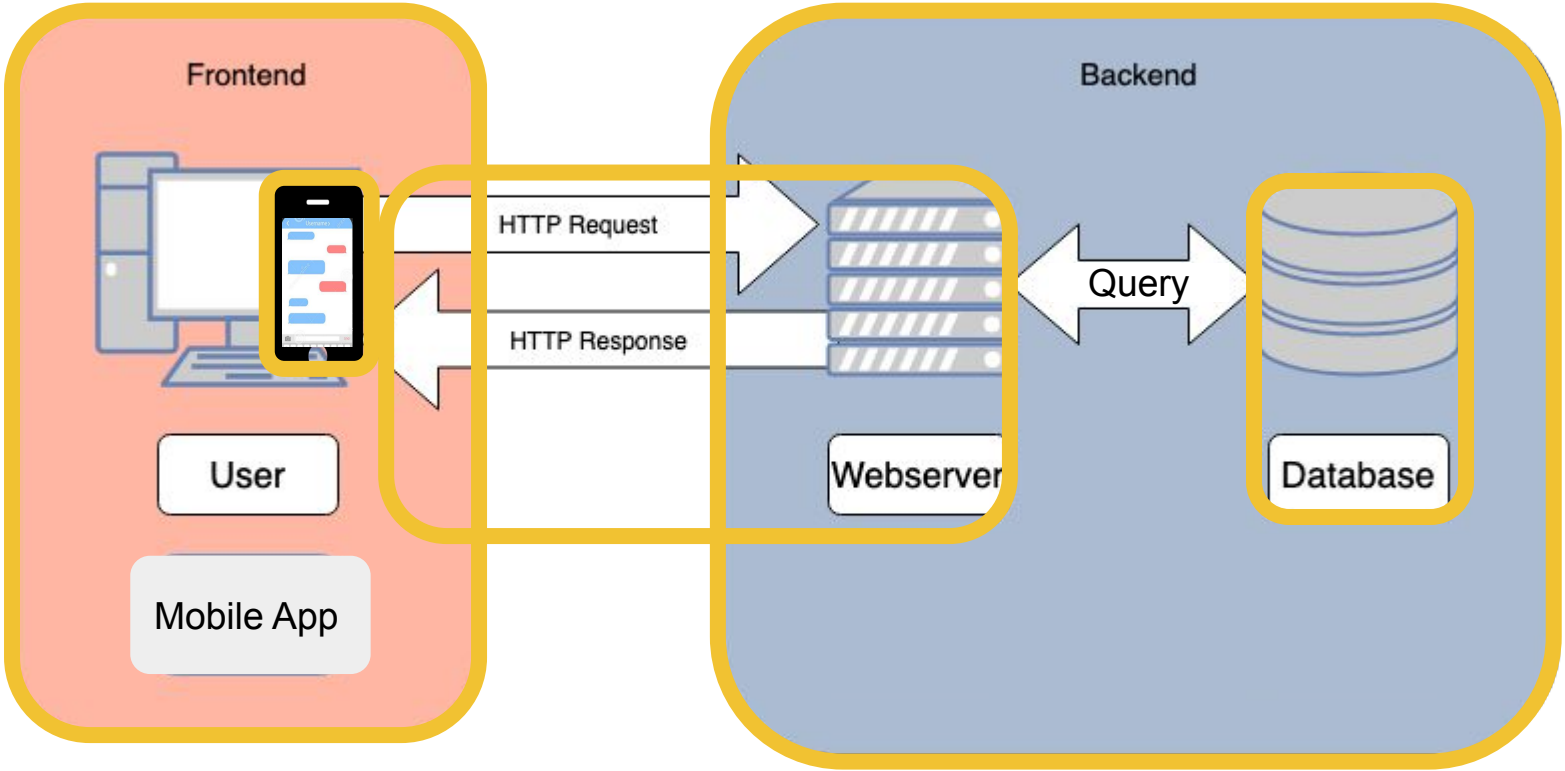
Pros

- Easy to share between multiple clients (Mobile and Web etc)
- Logic hidden from users (good for security, compatibility, and intensive computation)

Cons

- Interactions require a round-trip to server

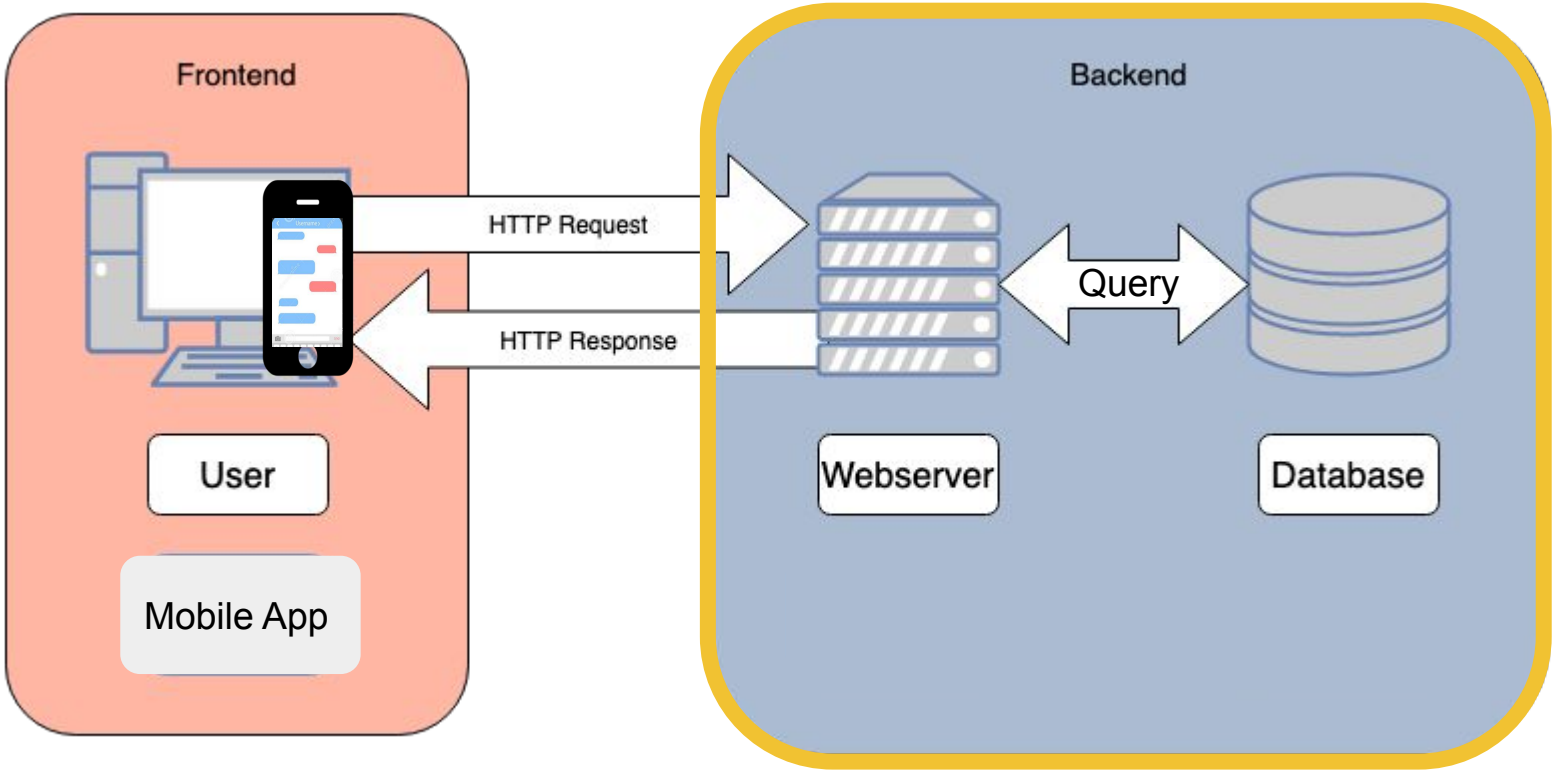
What would the infrastructure be?



Java

Javascript

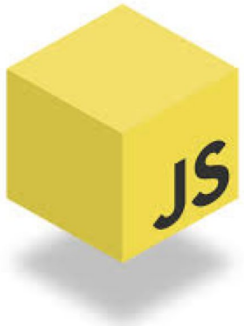
What would the infrastructure be: JS Backend



Java

Javascript

JavaScript: Introduction



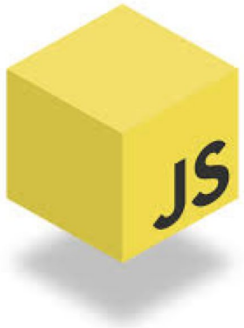
- A scripting language initially designed for web pages
- Key differences with Java
  - Interpreted
    - I.e. Instructions translated at run time
  - No pre-runtime type checking
    - TypeScript, an extension to JavaScript, allows type declarations
    - Be cautious about weird behaviours

```
>> [2] * 2
<- 4
```

```
>> [2] + 2
<- "22"
```

```
>> 3 * "3"
<- 9
```

Javascript: Key Concepts



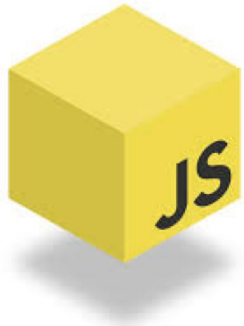
- Everything is an object (OOP)
  - Even functions!
  - Even the “class” of an object is an object!
- Key differences with Java
  - Objects are just associative arrays/maps from properties to values
  - No “class” per se., **object prototype** *

```
// function constructor
function Person(name, job, yearOfBirth){
    this.name= name;
    this.job= job;
    this.yearOfBirth= yearOfBirth;
```

```
1  // Initializing an empty object
2  var empty_object = {};
3
4  // Object with two attributes
5  var name = {
6      firstName: "Karthik",
7      lastName: "Pattabiraman";
8  };
```

*https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object_prototypes

Javascript: Key Concepts



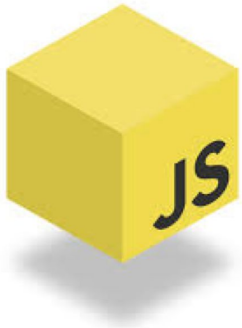
- Everything is an object (OOP)
  - Even functions!
  - Even the “class” of an object is an object!
- Key differences with Java
  - No “class” per se., **object prototype**
  - __proto__ field added when object built with function constructor

```
// function constructor
function Person(name, job, yearOfBirth){
    this.name= name;
    this.job= job;
    this.yearOfBirth= yearOfBirth;
}
Person.prototype.calculateAge= function(){
    console.log('The current age is: '+(2019- this.yearOfBirth));
}
console.log(Person.prototype);
```

Output

```
▼ {calculateAge: ƒ, constructor: ƒ}
  ▶ calculateAge: ƒ ()
  ▶ constructor: ƒ Person(name, job, yearOfBirth)
  ▶ __proto__: Object
```

Javascript: Key Concepts



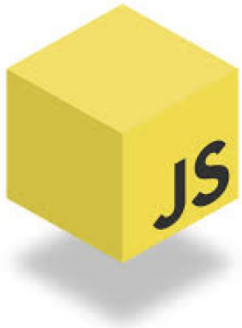
- Callbacks
  - Functions passed as an argument to another function
  - Executed within outer function

```
function greeting(name) {
  alert('Hello ' + name);
}

function processUserInput(callback) {
  var name = prompt('Please enter your name.');
  callback(name);
}

processUserInput(greeting);
```

Output: Hello “Entered Name”

"I will call back later!"

Javascript: Key Concepts



- Asynchronous nature

setTimeout(callback, time) handled asynchronously (non-blocking)

setTimeout(callback, time) executes *callback* after *time* ms

```
function printMe() {
  console.log('print me');
}

function test() {
  console.log('test');
}

setTimeout(printMe, 2000);
test();
```

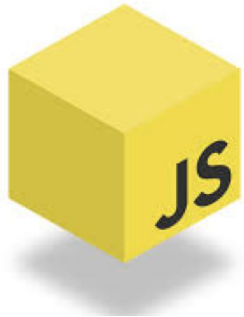
Output

print me
test

test
print me

- Always refer to documentation to check if API asynchronous!

Javascript: Key Concepts



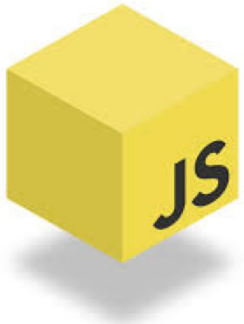
- Two-main types of asynchronous APIs
  - Callback-based
    - Execute callback once operation is complete

setTimeout(callback, time) handled asynchronously (non-blocking)

setTimeout(callback, time) executes *callback* after *time* ms

```
function printMe() {
  console.log('print me');
}

function test() {
  console.log('test');
}

setTimeout(printMe, 2000);
test();
```

Javascript: Key Concepts



- Two-main types of asynchronous APIs
  - Promise-based

```
var promise = new Promise(function(resolve, reject) {
  // do a thing, possibly async, then…

  if (/* everything turned out fine */) {
    resolve("Stuff worked!");
  }
  else {
    reject(Error("It broke"));
  }
});
```

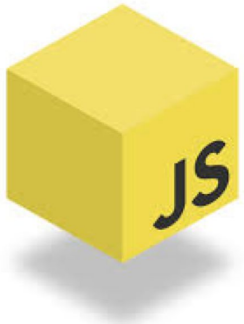
Represent eventual completion (or failure) of an asynchronous operation

returns failure promise

returns success promise

"I Promise a Result!"

Javascript: Key Concepts



- Two-main types of asynchronous APIs
  - Promise-based

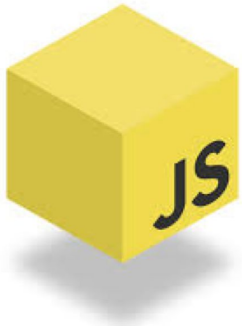
```
var promise = new Promise(function(resolve, reject) {
  // do a thing, possibly async, then…

  if (/* everything turned out fine */) {
    resolve("Stuff worked!");
  }
  else {
    reject(Error("It broke"));
  }
});
```

then(successCb, failureCb) “consumes” result when ready *

```
promise.then(function(result) {
  console.log(result); // "Stuff worked!"
}, function(err) {
  console.log(err); // Error: "It broke"
});
```

* Not the only way (more in database section)

Javascript: Key Concepts



- Two-main types of asynchronous APIs
  - Promise-based

```
1  var promise = new Promise(function(resolve, reject) {
2    // do some long running async thingâ€¦
3
4    if (/* everything turned out fine */) {
5      resolve("Stuff worked!");
6    }
7    else {
8      reject(Error("It broke"));
9    }
10 });
11
12 //usage
13 promise.then(
14   function(result) { /* handle a successful result */ },
15   function(error) { /* handle an error */ }
16 );
```

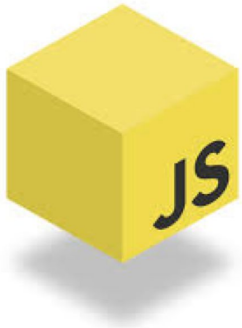
Represent eventual completion (or failure) of an asynchronous operation

returns failure promise

returns success promise

"I Promise a Result!"

Javascript: Key Concepts



- Two-main types of asynchronous APIs
  - Promise-based

```
1  var promise = new Promise(function(resolve, reject) {
2    // do some long running async thingâ€¦
3
4    if (/* everything turned out fine */) {
5      resolve("Stuff worked!");
6    }
7    else {
8      reject(Error("It broke"));
9    }
10 });
11
12 //usage
13 promise.then(
14   function(result) { /* handle a successful result */ },
15   function(error) { /* handle an error */ }
16 );
```

then(successCb, failureCb)
“consumes” result when ready *

Case of success i.e *resolve* was invoked

Case of failure i.e *reject* was invoked

"I Promise a Result!"

* Not the only way (more in database section)

JavaScript: Usages

- JavaScript is widely used for web applications
- Browsers contain JavaScript engines

Chrome has **V8**

Firefox has **SpiderMonkey**

- Due to its versatility, developers wanted to use JavaScript outside of browsers
  - **Node.js** is a such a runtime environment for JavaScript

Node JS

- Javascript Runtime
- Built on Chrome’s V8 javascript engine
- Asynchronous IO model
- Large package ecosystem, NPM!

Download and install Node.js at https://nodejs.org/en/download/

Running Node for the first time (on Linux / Mac)

- Create a new javascript file with .js as extension

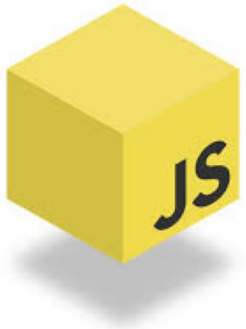
```
$ edit server.js
```

- Add print statement to server.js file

```
$ console.log('May Node be with you')
```

- Execute server.js file

```
$ node server.js
```



Using Node and NPM

- Executing javascript using node

```
$ node <File Name>
```

- Installing Modules using NPM
  - To install any Node.js module, in the command line:

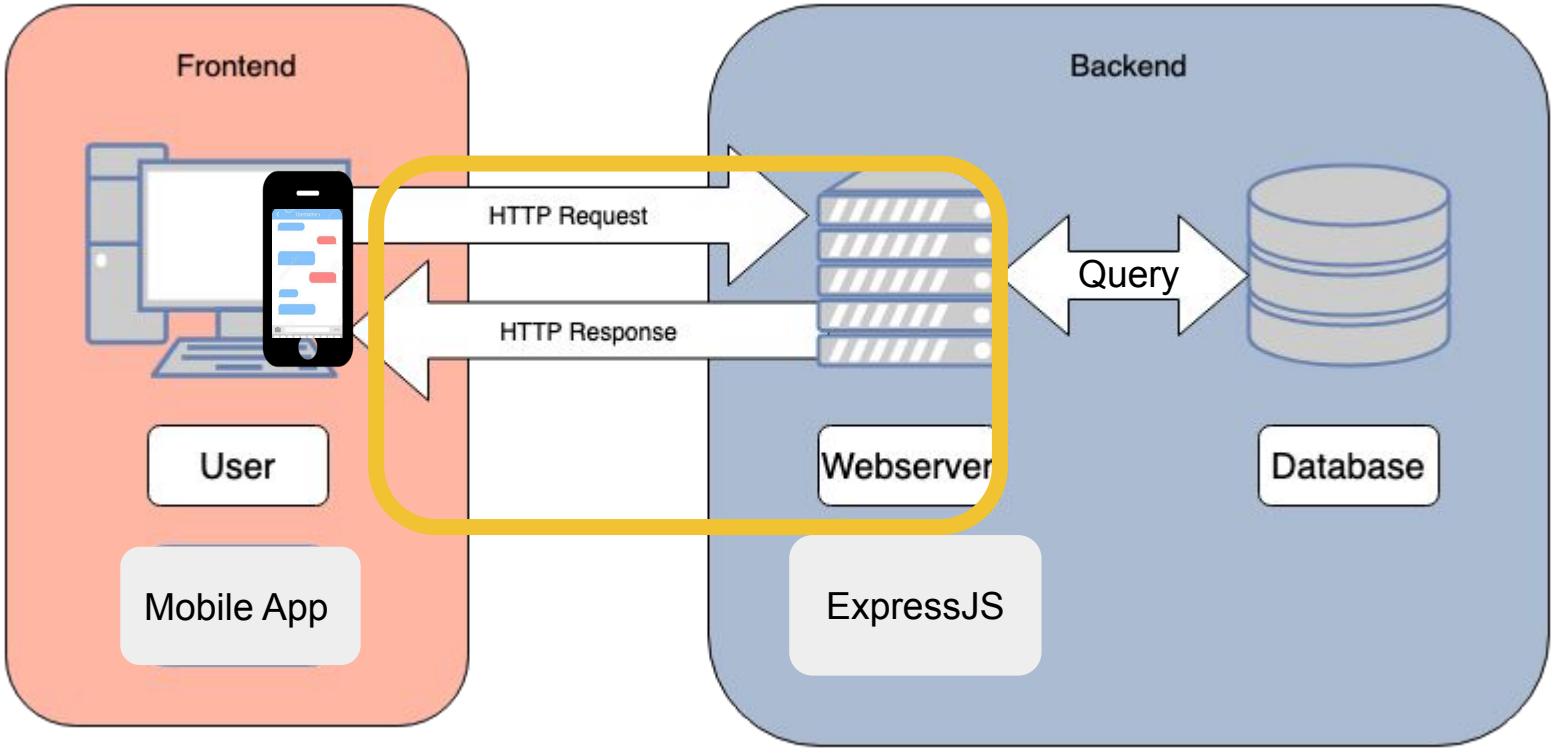
```
$ npm install <Module Name>
```

  - E.g: `$ npm install express`

- Using installed module in JavaScript

```
var express = require('express');
```

What would the infrastructure be: HTTP & Webserver



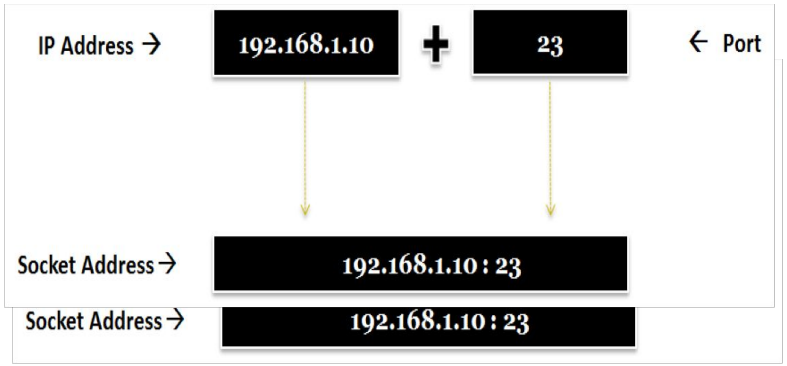
Java

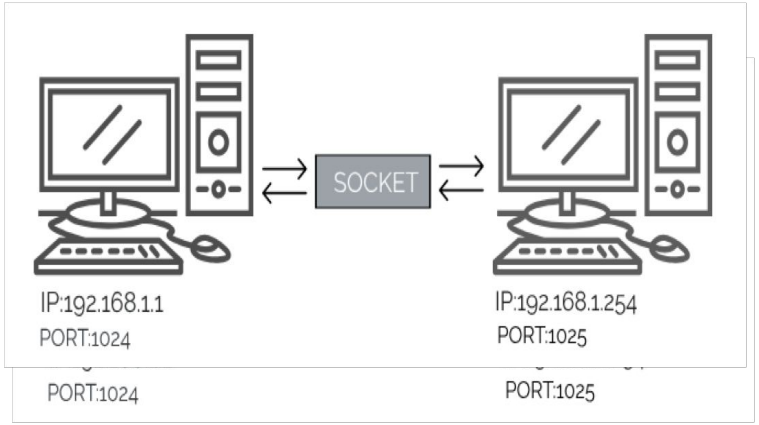
Javascript

Extra slide/Not covered

Frontend and backend communication - Socket

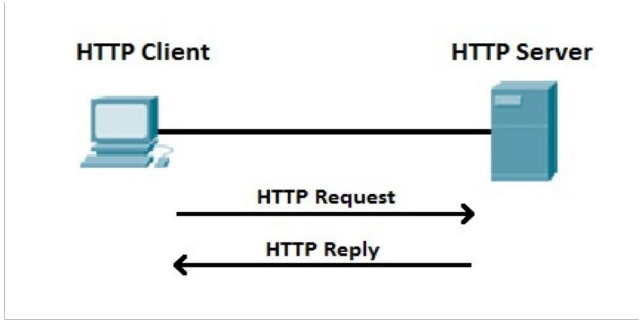
- Backend server is uniquely identified using **socket** (IP:PORT combination)
- **IP address** is a unique number given to a computer
- **PORT** is used to identify a particular program running inside that computer
- **DNS name** is a human readable alias for IP address





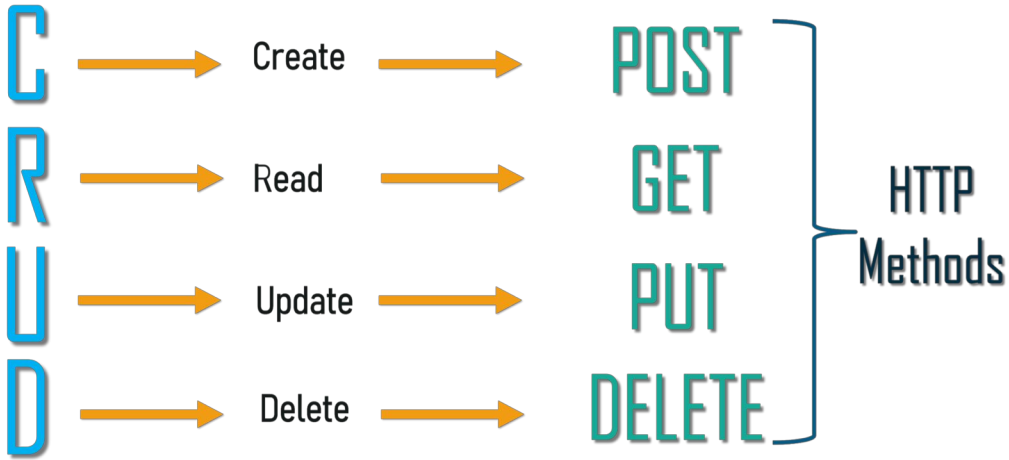
Frontend and backend communication - HTTP

- HTTP is most common network protocol used by systems in web
- A **network protocol** is set of rules two computers use to talk to each other
- HTTP frontend (e.g., browser) makes requests to HTTP backend and gets responses back
- In this tutorial, we will be creating a backend that uses HTTP to communicate with frontend.



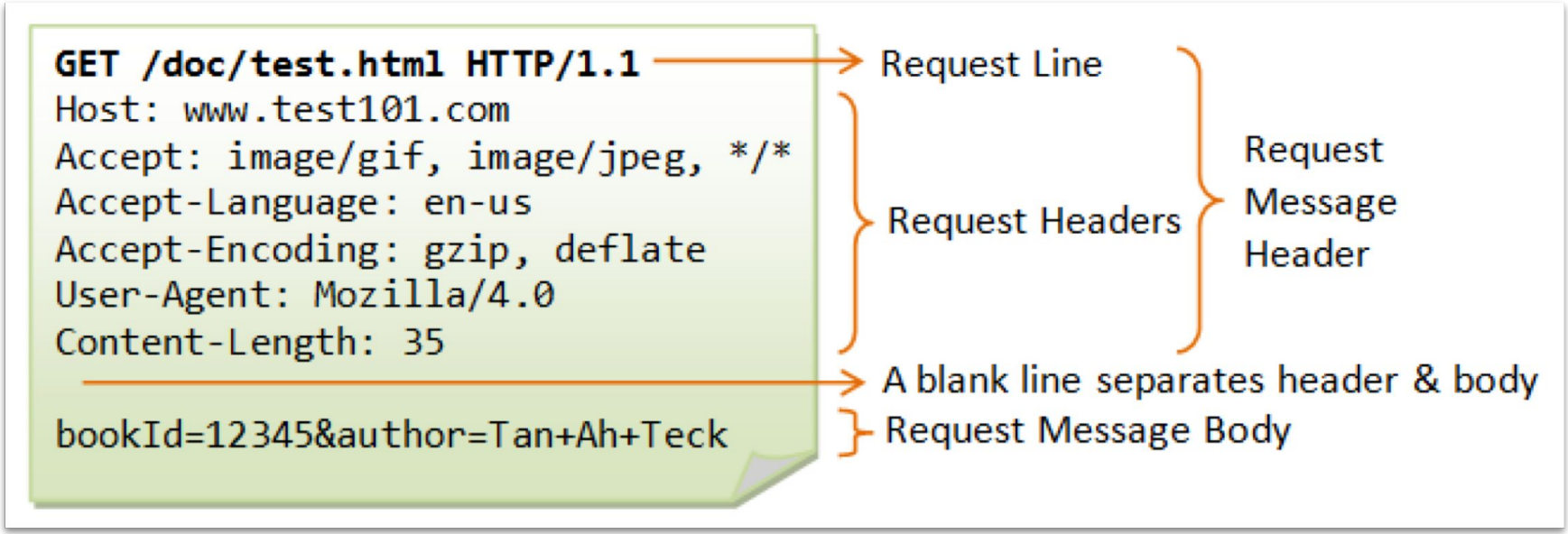
HTTP protocol - Verbs

- Functions that can be executed on a backend server through HTTP request



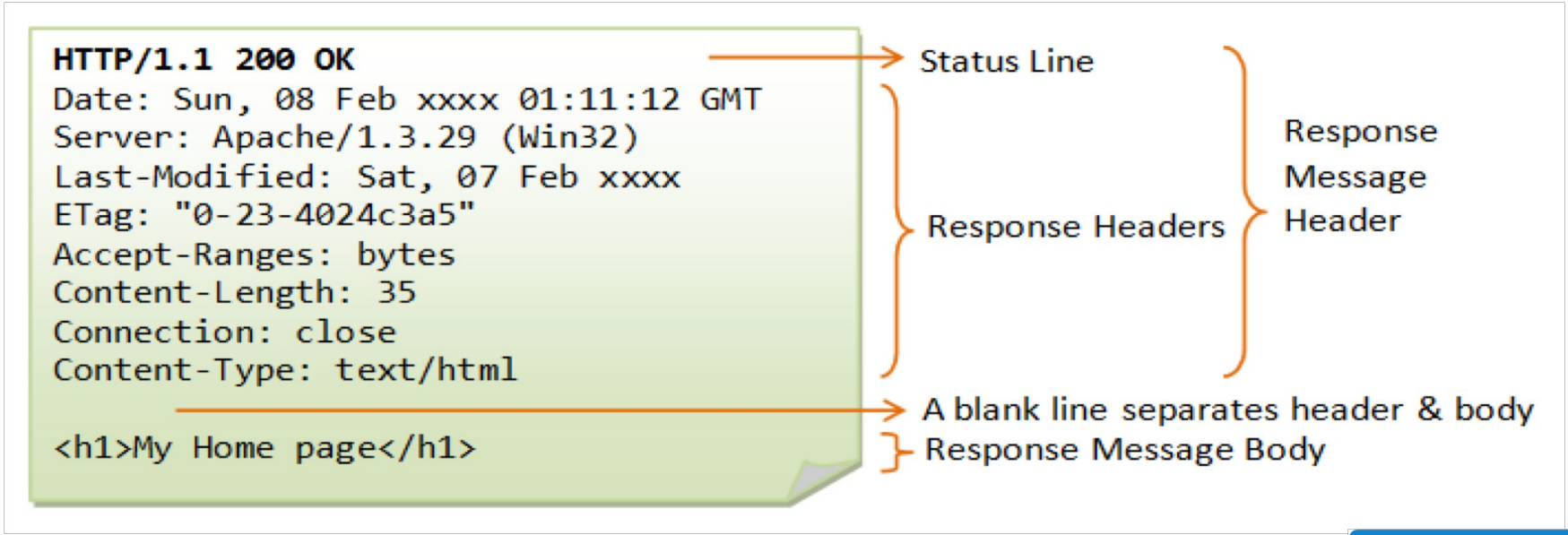
http://

HTTP protocol - Request



http://

HTTP protocol - Response



http://

Data format of HTTP response

- Server response can be any text format: Text, HTML, XML, etc.
- A popular format (used in this tutorial) is **JSON**.

Sample response JSON object

```
{
    "parameter":"value",
    "parameter":number,
    "nested parameter": {
        "parameter":"value",
    }
}
```

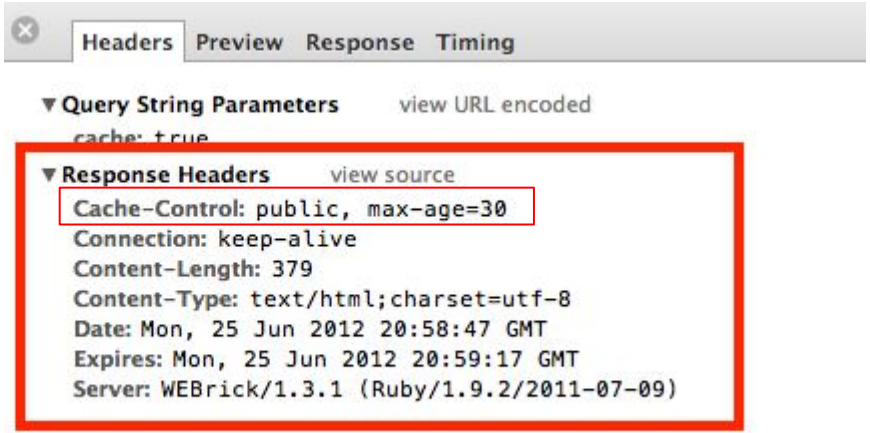
Response codes

- When server detects a error, it will send some standard error codes to client
- These error codes have to be used as they defined in the standard
  - A list of the most important status codes:
    - 2xx = Success
    - 3xx = Redirect
    - 4xx = User error
    - 5xx = Server error

Extra slide/Not covered

Caching HTTP Response

- The network between client and server can have a cache which will store most recent responses for each request
  - Each web browser gets a cache
- In HTTP header, a server can specify whether to store the response in the cache and for how long



- A Web application framework for Node.js (a library)
- Light-weight and minimalist
- Provides boilerplate structure & organization for your web-apps

Hello World in Express

```
var express = require('express');
var app = express();
```

Import Express module

Create a Express object

```
app.get('/', function (req,
res) {
   res.send('Hello
   World');
}
)
```

Handle HTTP GET request for path “/”

```
var server = app.listen(8081,
function () {
   var host =
   server.address().address
   var port =
   server.address().port

   console.log("Example app listening at
   http://%s:%s", host, port)
})
```

Create a HTTP server that listens to port 8081

Save the above code in a file named server.js and run it with the following command.

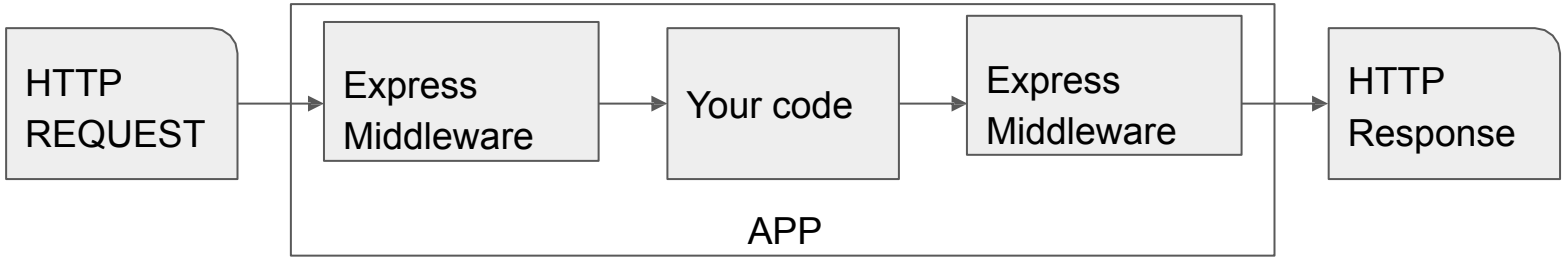
```
$ node server.js
```

Run node js program

```
Curl -X GET server-ip:8081
```

Make a GET request from client using curl

Express middleware



- **Middlewares** can be used and chained to pre-process the HTTP requests (resp.response) before(resp.after) they are treated received by Your code
  - E.g. you can use express.json() middleware to extract json from your request.
  - To enable a middleware (e.g json parsing), you use

```
app.use(express.json())
```

Reading JSON from requests

```
var express = require('express');
var app = express();

app.use(express.jso
n)
app.post('/', function (req,
res) {
   res.send(req.body.text
} );
)
var server = app.listen(8081, function ()
{  var host =
   var port =
   server.address().port
   console.log("Example app listening at http://%s:%s" , host,
   port)})
```

Insert middleware before HTTP handlers

Read json objects in HTTP handlers

Save the above code in a file named server.js and run it with the following command.

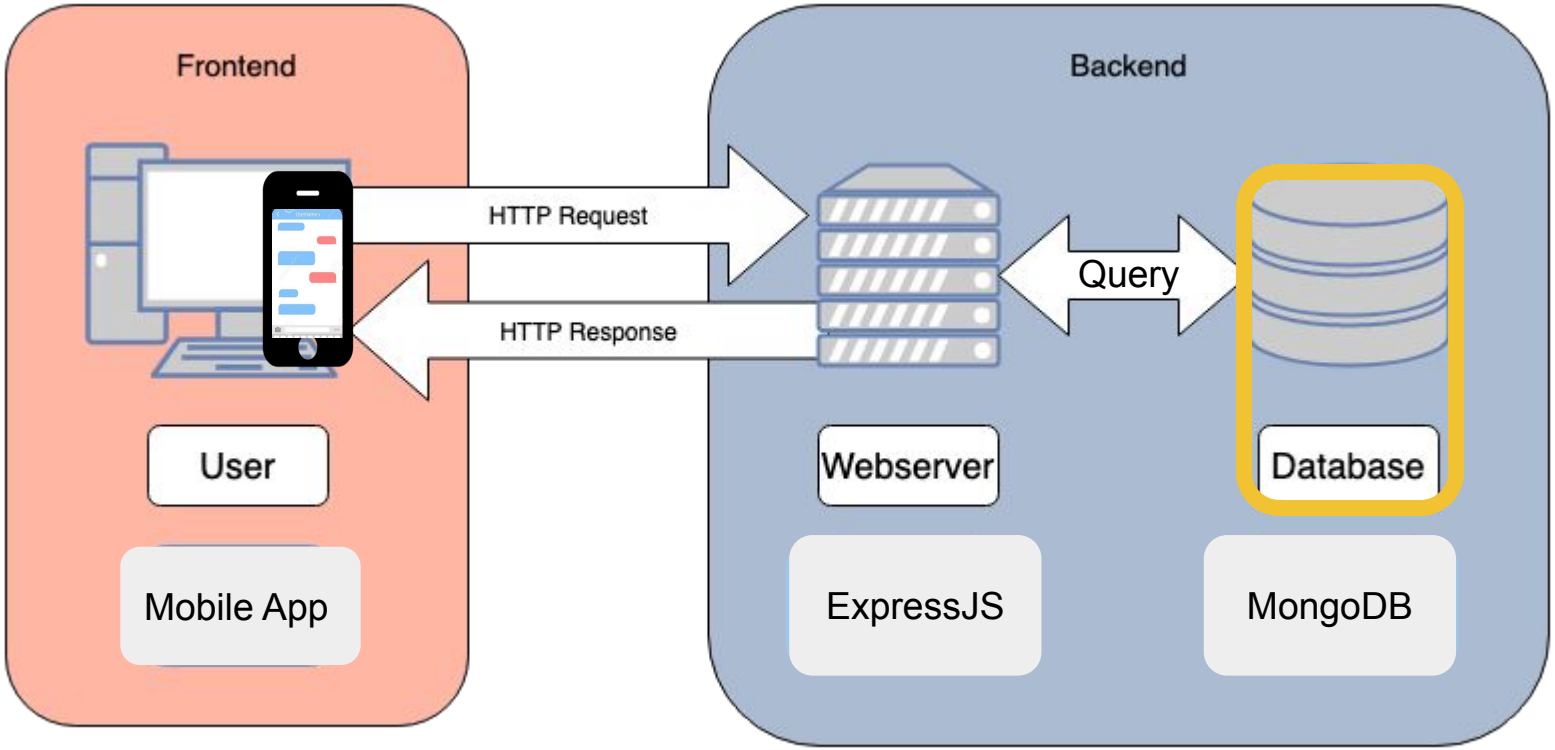
```
$ node server.js
```

Pass JSON data in HTTP requests

You will see the following output −

```
Curl -X POST -H "Content-Type: application/json" -d '{"text":"helloworld"}'
0.0.0.0:8081
```

What would the infrastructure be: Database connection



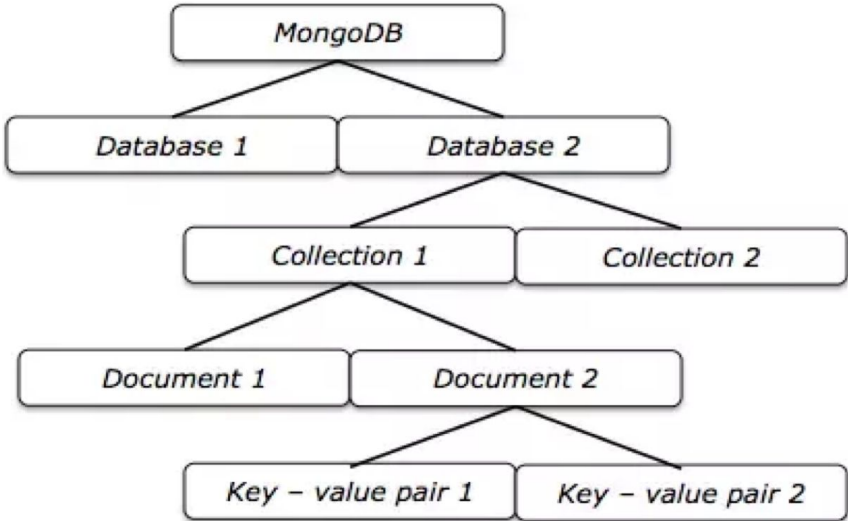
Java

Javascript

mongoDB®

- **Database:** physical container for collections
  - Each database gets its own set of files on the file system
  - A single MongoDB server typically has multiple databases

- **Collection:** a group of MongoDB documents
  - Does not enforce a schema (documents can have different fields)
  - Good practice to keep related documents in same collection



Install Mongodb client module from NPM

```
npm install mongodb --save
```

Install MongoDb server

https://docs.mongodb.com/manual/administration/install-community/

Connecting to the database

```
const {MongoClient} =
require('mongodb')
const uri = "mongodb://localhost:27017 "
const client = new MongoClient(uri)
```

Import mongodb module

```
try{
      await client.connect(uri)
}catch(err){
      await client.close()
}
```

Connect to the database (asynchronous operation)

Closing connection in case of error

mongoDB®

- Create -> InsertOne

```
db.collection("list").insertOne(JSON)
```

- Update->UpdateOne

```
db.collection("list").replaceOne(queryJSON,{$set:UPdateJSON}
```

- Read -> Find

```
db.collection("list").find()
```

- Delete -> DeleteOne

```
db.collection("list").deleteOne(queryJSON)
```

Example Application - TODO list

- Let’s implement what we have learned.
  - Create (POST) - Create an Item
  - Read (GET) - Get an item
  - Update (PUT) - Change an item
  - Delete (DELETE)- Remove an item

First, import required libraries and add initializations

```
var express = require("express")
var app = express()

const {MongoClient} = require("mongodb")
const uri = "mongodb://localhost:27017"
const client = new MongoClient(uri)
```

Create express JS object and attach JSON middleware to it

```
const app = express()
app.use(express.json())
```

Open db connection, and start server on success

We need to deal with asynchronous operations, async/await is one possible way

```
async function run(){
    try{
        await client.connect()
        console.log("Successfully connected to the database")
        var server = app.listen(8081, function (){
            var host = server.address().address
            var port = server.address().port
            console.log("Example app running at http://%s:%s", host, port)
        })
    }
    catch(err){
        console.log(err)
        await client.close()
    }
}

run()
```

Now, server should be up: **node server.js**

```
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ node server.js
Successfully connected to the database
Example app running at http://:::8081
```

Now, behavior is still undefined for POST requests

```
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ curl -X POST localhost:8081/todolist
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot POST /todolist</pre>
</body>
</html>
```

Let’s add a POST request handler to insert new items

```
app.post("/todolist", async (req, res) =>{
    try{
        await client.db("test").collection("todolist").insertOne(req.body)
        res.status(200).send("Todo item added successfully\n")
    }
    catch(err){
        console.log(err)
        res.send(400).send(err)
    }

})
```

We add some error handling and status code for valid/invalid request

We can send a POST request to this endpoint

```
curl -X PUT -H "Content-Type: application/json" -d '{"task": "Finish this tutorial", "status": "Lol"}' localhost:8081/todolist
```

We can also retrieve list elements with GET

```
app.get("/todolist", async (req, res) => {
    try{
        const result = await
client.db("test").collection("todolist").find(req.body).toArray()
        res.send(result)
    }
    catch(err){
        console.log(err)
        res.status(400).send(err)
    }
})
```

Now using curl to retrieve content of the database

```
faridah@DESKTOP-UGM16E7:~/HelloWorld$ curl -X GET -H "Content-Type: application/json" -d '{"task": "Finish this tutorial"}' localhost:8081/todolist
[{"_id":"6282b3c71e111c5ad39ec42e","task":"Finish this tutorial","status":"RIP"}]faridah@DESKTOP-UGM16E7:~/HelloWorld$
```

As well as update elements with PUT

```
app.put("/todolist", async (req, res) => {
    try{
        await
client.db("test").collection("todolist").replaceOne({"task":
"Finish this tutorial"}, req.body)
        res.status(200).send("Todo item modified successfully\n")
    }
    catch(err){
        console.log(err)
        res.status(400).send(err)
    }

})
```

Here replaceOne takes two arguments: the first argument is a query, which will be used to search for task name; the second argument will be used to update task's info.

Lets use CURL to test it

```
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ curl -X PUT -H "Content-Type: application/json" -d '{"task": "Finish this tutorial", "status": "Almost over"}' localhost:8081/todolist
Todo item modified successfully
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ curl -X GET -H "Content-Type: application/json" -d '{"task": "Finish this tutorial"}' localhost:8081/todolist
[{"_id":"6282b3c71e111c5ad39ec42e","task":"Finish this tutorial","status":"Almost over"}]faridah@DESKTOP-UGMI6E7:~/HelloWorld$
```

Finally, we can DELETE items from our todolist

```
app.delete("/todolist", async (req, res) => {
    try{
        await
client.db("test").collection("todolist").deleteOne({"task":
req.body.task})
        res.status(200).send("Todo item deleted successfully\n")
    }
    catch(err){
        console.log(err)
        res.status(400).send(err)
    }
})
```

First variable in deleteOne is query JSON same as replaceOne.
We will query the database based on the task name.

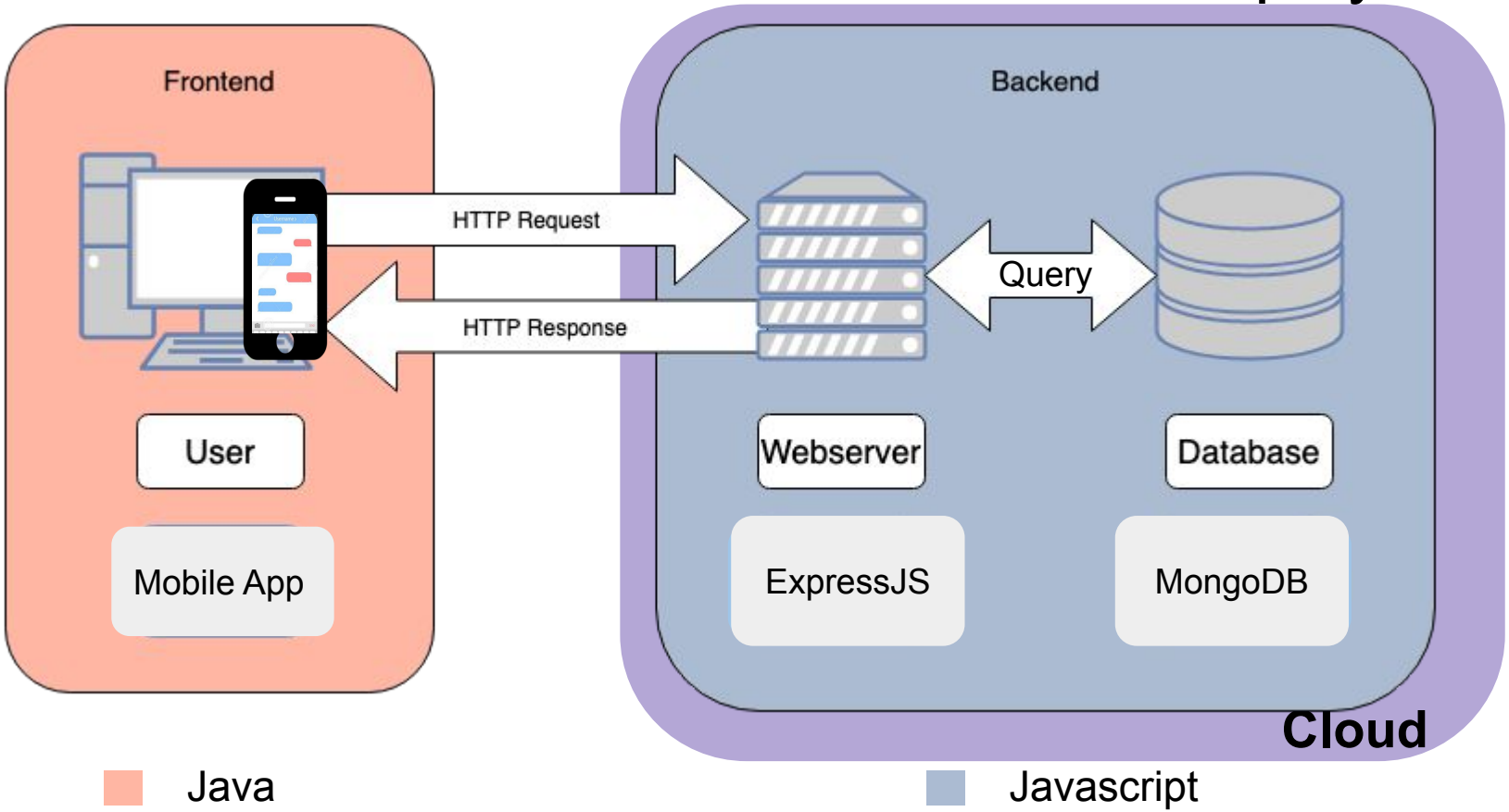
Let's test using CURL

```
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ curl -X DELETE -H "Content-Type: application/json" -d '{"task": "Finish this tutorial"}' localhost:8081/todolist
Todo item deleted successfully
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ curl -X GET -H "Content-Type: application/json" -d '{"task": "Finish this tutorial"}' localhost:8081/todolist
[]faridah@DESKTOP-UGMI6E7:~/HelloWorld$
```

Useful Links/Recommendations

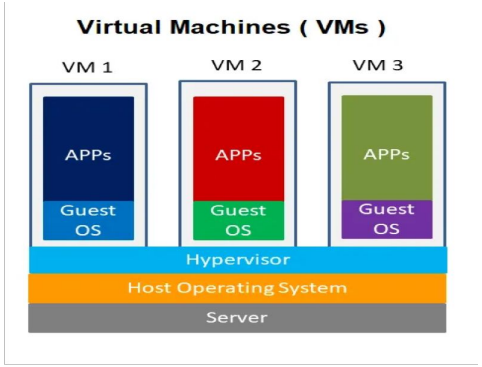
- Modules: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction#importing_and_creating_modules
- Routing: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction#creating_route_handlers
- ORM: https://mongoosejs.com/
- Curl:https://curl.se/download.html, Postman: https://www.postman.com/
- Nodemon

What would the infrastructure be: Cloud deployment



What is cloud and what is a VM ?

- **Cloud** refers to servers accessed through the internet
  - In Cloud Computing, you will rent a server from a cloud provider and pay only for the time server is up.
- For efficiency, cloud providers share a physical machine to multiple clients
- The part of machine each client gets is called a **virtual machine**.



Getting an application on the cloud

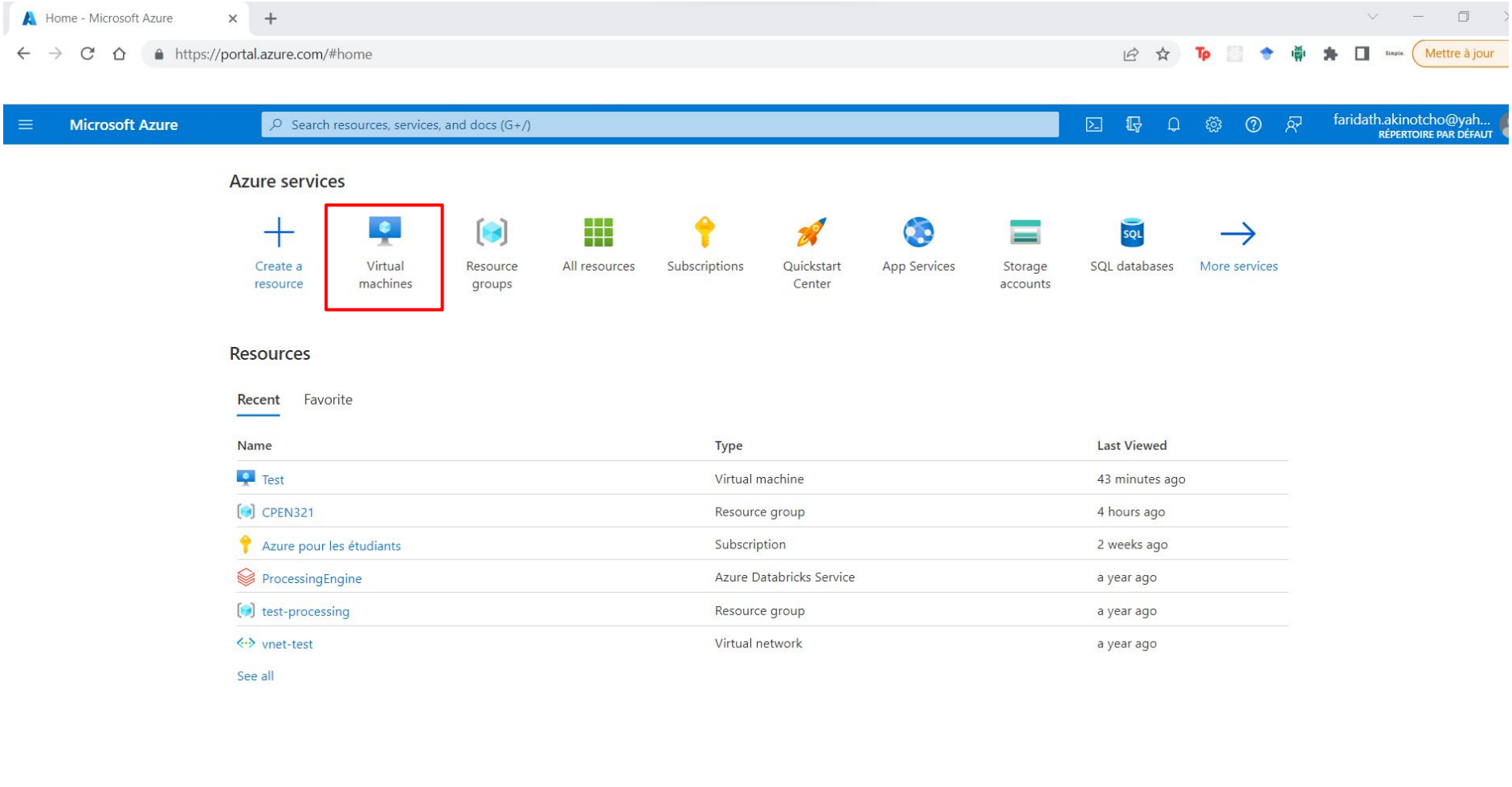
You are free to choose any cloud provider you like.

Most vendors provide free student subscriptions. You can also request Microsoft Azure credits from the course instructor. They are limited and will be given by the professor based on your request.

Important: Cloud providers charge based on VM up time. So, switch off your VM when you are not using it. You will get limited credits to spend on cloud so spend wisely.

Getting an application on the cloud

- Create a VM
- Get IP Address
- SSH into the VM using IP address
- Transfer files from local to VM (e.g using SCP, git clone)
- Install all necessary dependencies on VM
- Allow public access to the port your server runs on


Home - Microsoft Azure
https://portal.azure.com/#home
Mettre à jour
Microsoft Azure
Search resources, services, and docs (G+/)
faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT
Azure services
Create a resource
Virtual machines
Resource groups
All resources
Subscriptions
Quickstart Center
App Services
Storage accounts
SQL databases
More services
Resources
Recent
Favorite
Name
Type
Last Viewed
Test
Virtual machine
43 minutes ago
CPEN321
Resource group
4 hours ago
Azure pour les étudiants
Subscription
2 weeks ago
ProcessingEngine
Azure Databricks Service
a year ago
test-processing
Resource group
a year ago
vnet-test
Virtual network
a year ago
See all

Virtual machines - Microsoft Azu

https://portal.azure.com/#blade/HubsExtension/BrowseResource/resourceType/Microsoft.Compute%2FVirtualMachines

Microsoft Azure

Search resources, services, and docs (G+/)

faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT

Home >

Virtual machines

Répertoire par défaut

Create | Switch to classic | Reservations | Manage view | Refresh | Export to CSV | Open query | Assign tags | Start | Restart | Stop | Delete | Services | Maintenance

Filter for any field... | Subscription == all | Type == all | Resource group == all | Location == all | Add filter

No grouping | List view

Name	Type	Subscription	Resource group	Location	Status	Operating system	Size	Public IP address	Disks

No virtual machines to display

Create a virtual machine that runs Linux or Windows. Select an image from the marketplace or use your own customized image.

Learn more about Windows virtual machines

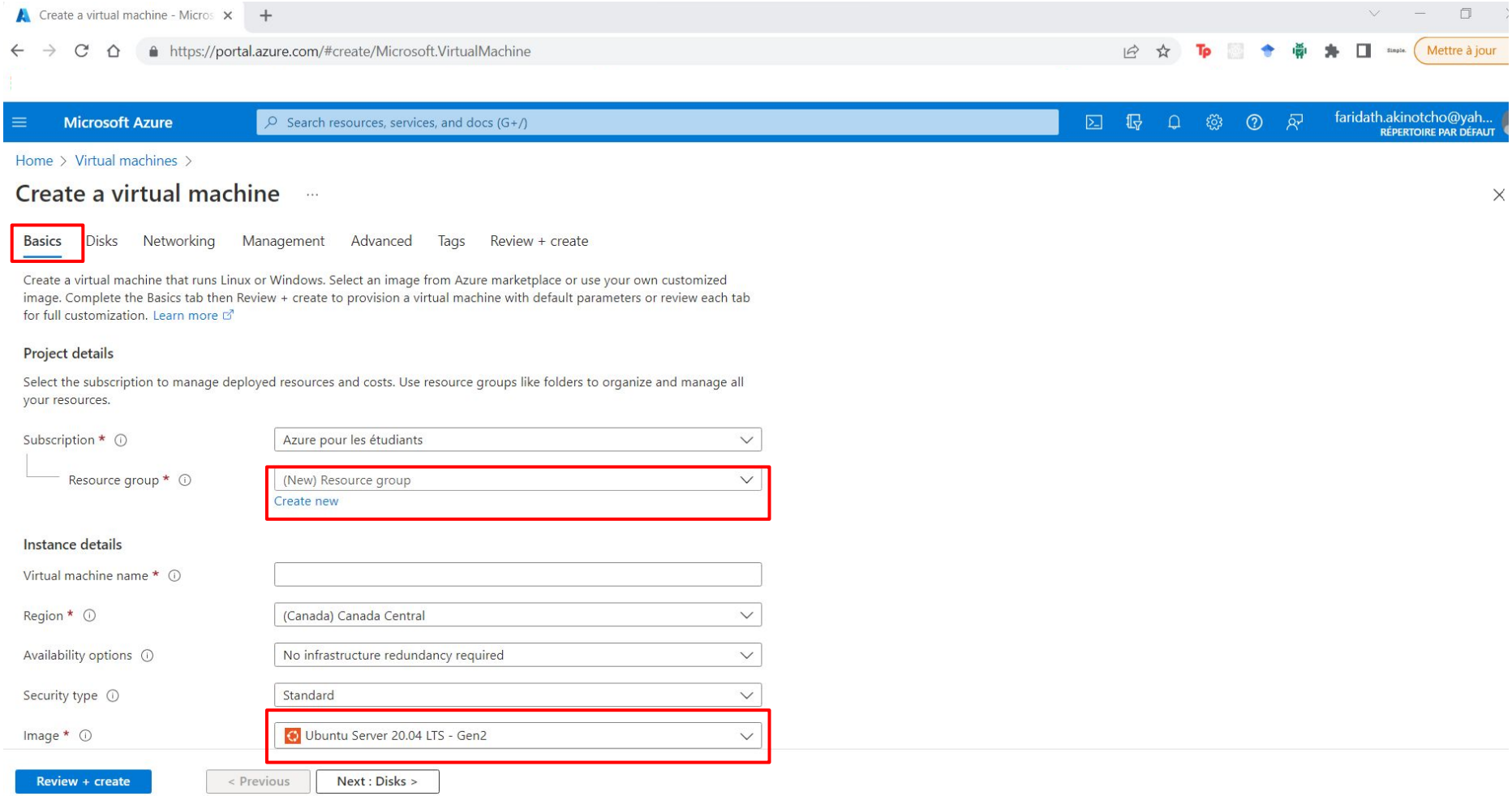
Learn more about Linux virtual machines

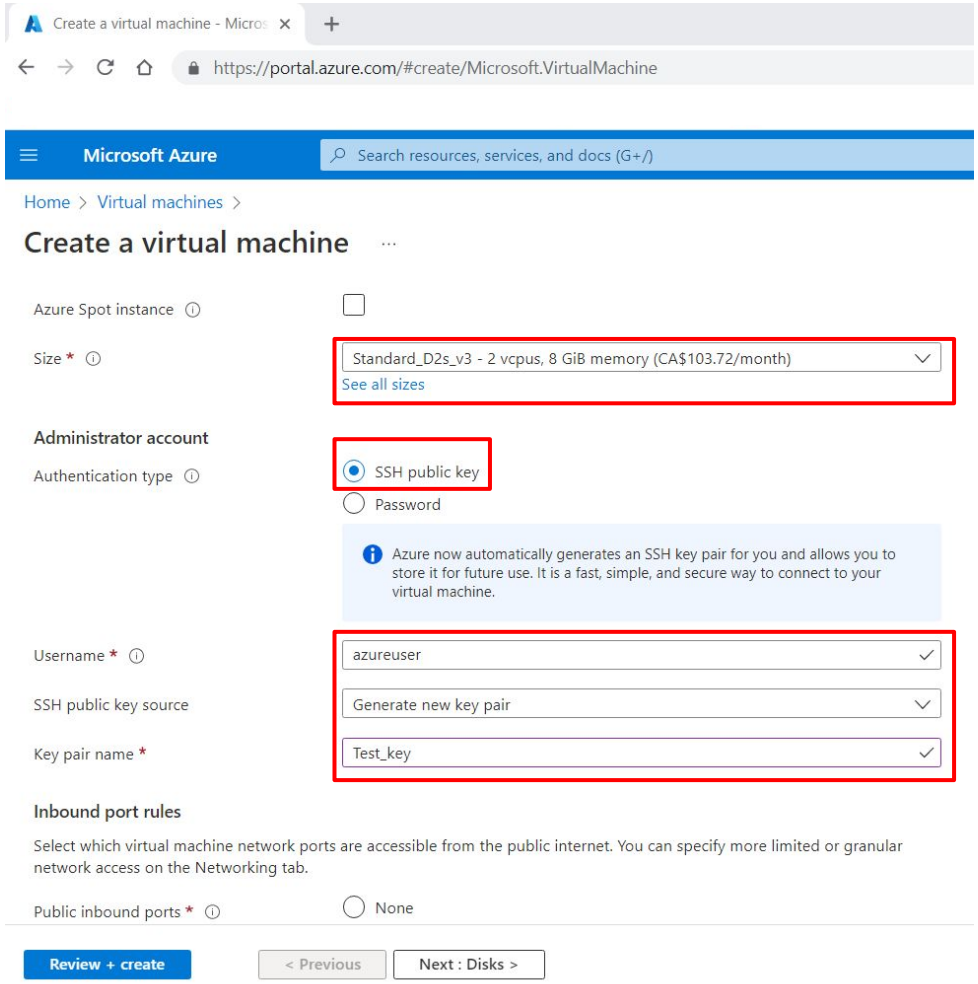
Give feedback

12°C
Nuageux

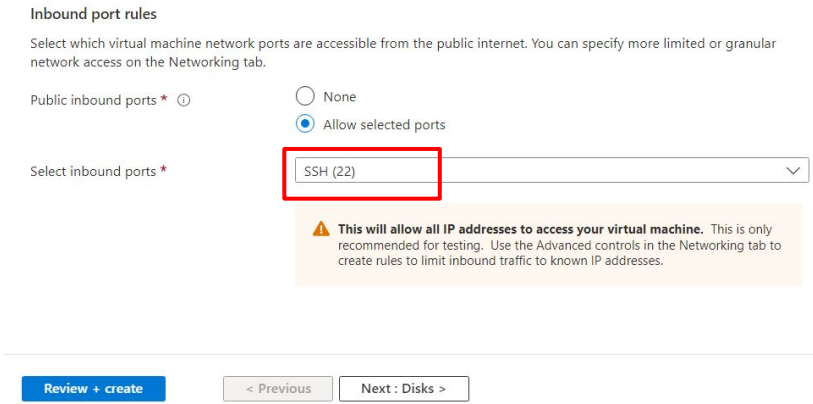
FRA

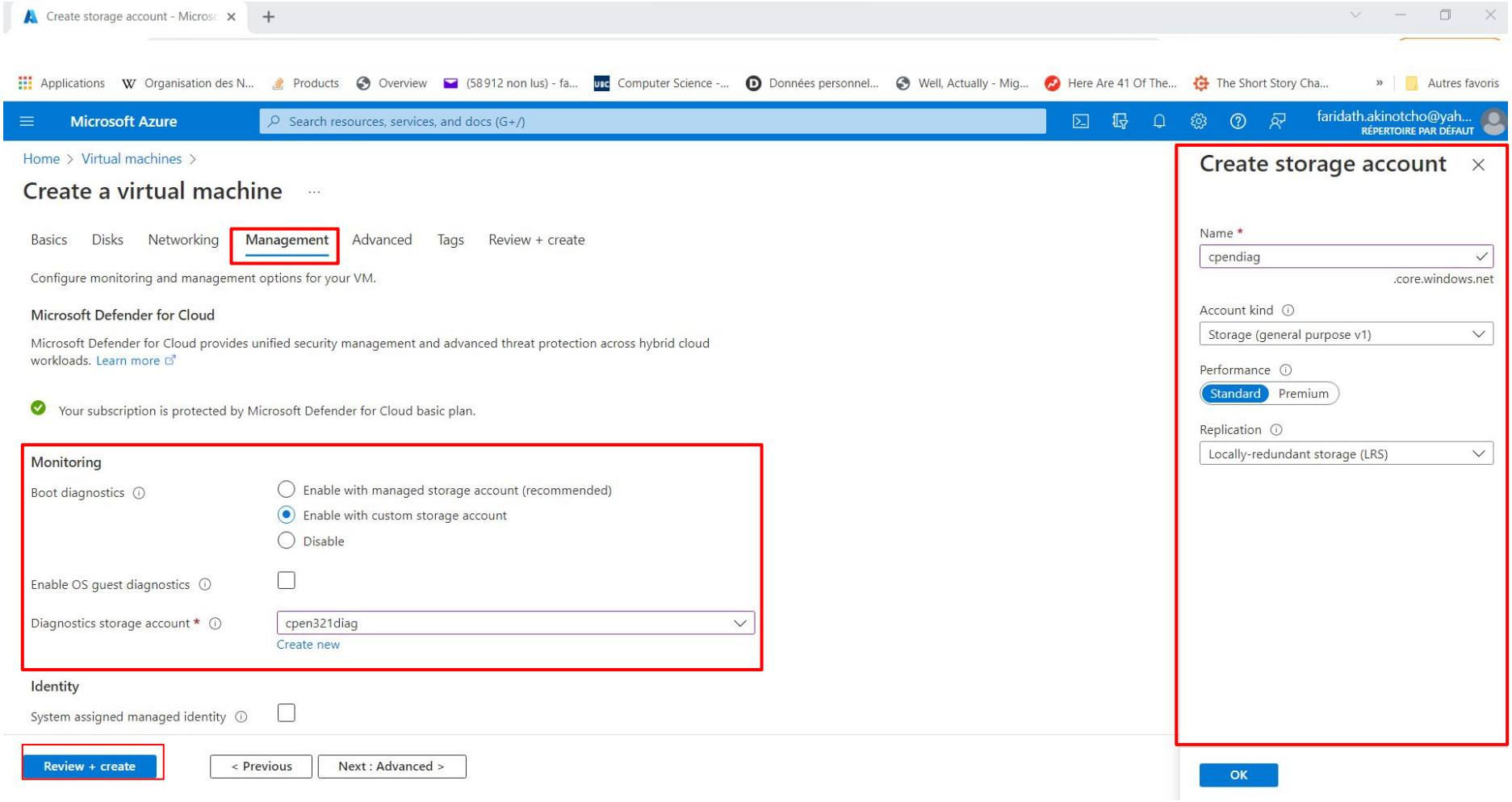
15:13
06/05/2022


Create a virtual machine - Micros
https://portal.azure.com/#create/Microsoft.VirtualMachine
Mettre à jour
Microsoft Azure
Search resources, services, and docs (G+/)
faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT
Home > Virtual machines >
Create a virtual machine
Basics
Disks
Networking
Management
Advanced
Tags
Review + create
Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
Azure pour les étudiants
Resource group *
(New) Resource group
Create new
Instance details
Virtual machine name *
Region *
(Canada) Canada Central
Availability options
No infrastructure redundancy required
Security type
Standard
Image *
Ubuntu Server 20.04 LTS - Gen2
Review + create
< Previous
Next : Disks >



Test your app to see if performs as expected on the VM type/size you select!




Create storage account - Micros
Applications
Organisation des N...
Products
Overview
(58912 non lus) - fa...
Computer Science -...
Données personnel...
Well, Actually - Mig...
Here Are 41 Of The...
The Short Story Cha...
Autres favoris
Microsoft Azure
Search resources, services, and docs (G+/)
faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT
Home > Virtual machines >
Create a virtual machine
Basics
Disks
Networking
Management
Advanced
Tags
Review + create
Configure monitoring and management options for your VM.
Microsoft Defender for Cloud
Microsoft Defender for Cloud provides unified security management and advanced threat protection across hybrid cloud workloads. Learn more
Your subscription is protected by Microsoft Defender for Cloud basic plan.
Monitoring
Boot diagnostics
Enable with managed storage account (recommended)
Enable with custom storage account
Disable
Enable OS guest diagnostics
Diagnostics storage account *
cpen321diag
Create new
Identity
System assigned managed identity
Review + create
< Previous
Next : Advanced >
Create storage account
Name *
cpendiag
.core.windows.net
Account kind
Storage (general purpose v1)
Performance
Standard
Premium
Replication
Locally-redundant storage (LRS)
OK

Test - Microsoft Azure | Use SSH keys to connect to Linux | Microsoft Portal Self Help

https://portal.azure.com/#@faridathakinotchoyahoo.onmicrosoft.com/resource/subscriptions/49463166-2e12-4308-9870-977678075248/resourcegroups/CPE...

Microsoft Azure

Search resources, services, and docs (G+/)

faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT

Home >

Test

Virtual machine

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

Settings

- Networking
- Connect
- Disks
- Size
- Security
- Advisor recommendations
- Extensions + applications
- Continuous delivery
- Availability + scaling
- Configuration
- Identity

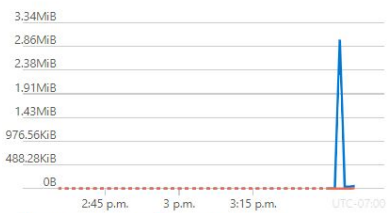
Connect | Start | Restart | Stop | Capture | Delete | Refresh | Open in mobile | CLI / PS | Feedback

Show data for last: 1 hour | 6 hours | 12 hours | 1 day | 7 days | 30 days

CPU (average)

Percentage CPU (Avg)
test
--

Network (total)



Network In Total (Sum)
test
3.12 MiB

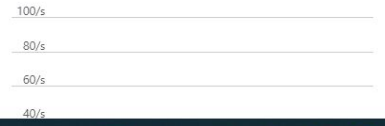
Network Out Total (Sum)
test
7.07 KiB

Disk bytes (total)

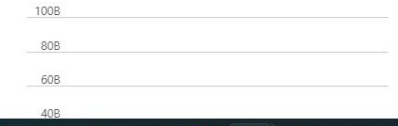
Disk Read Bytes (Sum)
test
--

Disk Write Bytes (Sum)
test
--

Disk operations/sec (average)



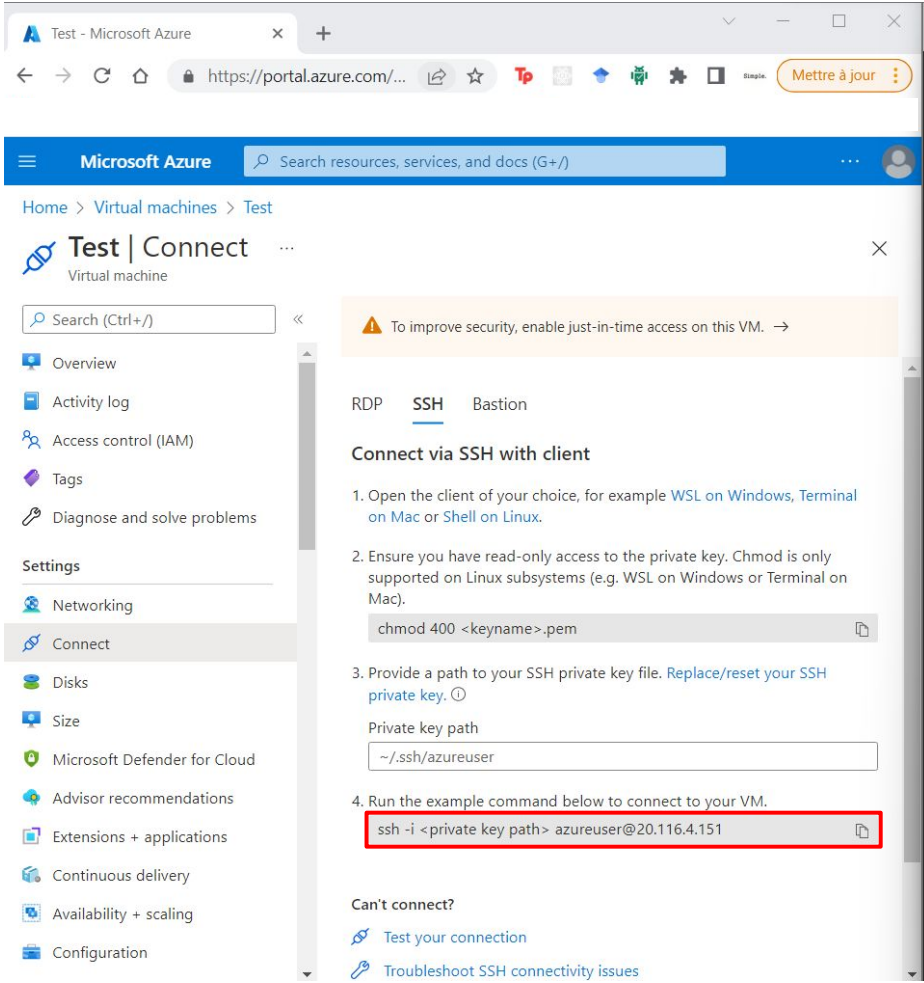
Available Memory Bytes (Preview)

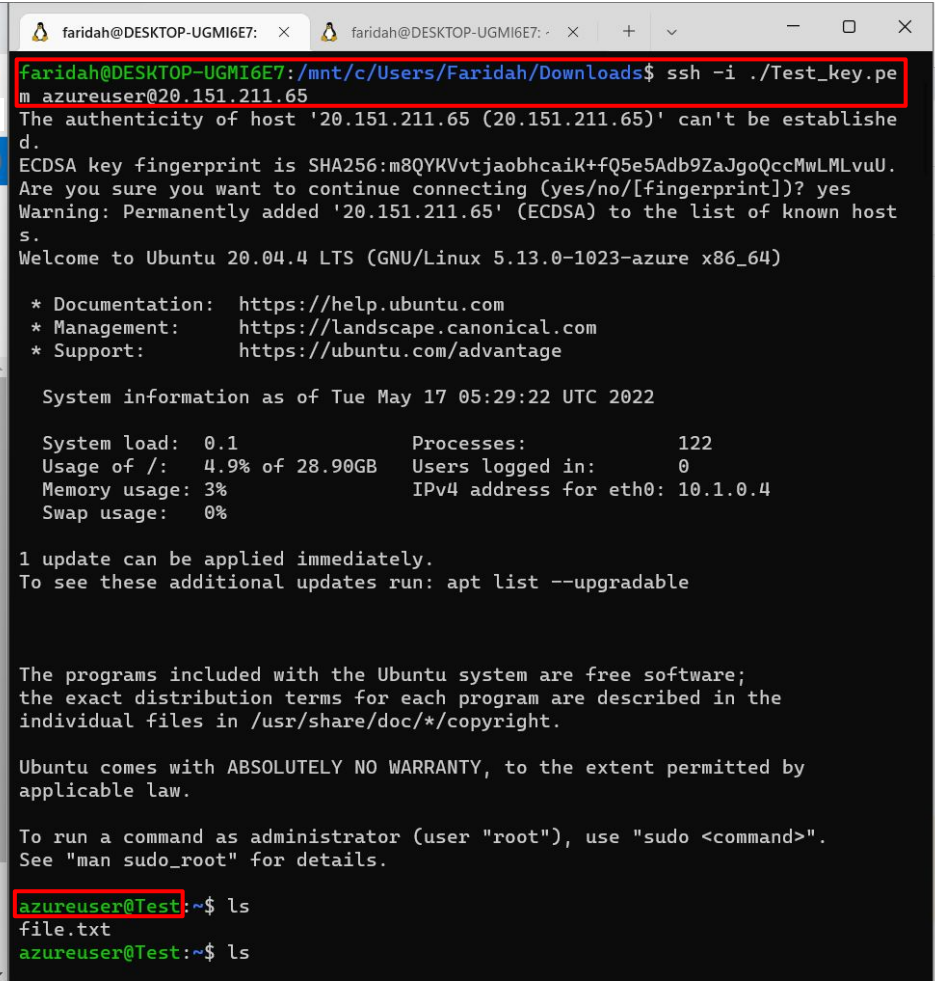


12°C
Nuageux

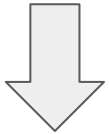
FRA

15:35
06/05/2022


Test - Microsoft Azure
https://portal.azure.com/...
Mettre à jour
Microsoft Azure
Search resources, services, and docs (G+/)
Home > Virtual machines > Test
Test | Connect
Virtual machine
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Microsoft Defender for Cloud
Advisor recommendations
Extensions + applications
Continuous delivery
Availability + scaling
Configuration
To improve security, enable just-in-time access on this VM. →
RDP
SSH
Bastion
Connect via SSH with client
1. Open the client of your choice, for example WSL on Windows, Terminal on Mac or Shell on Linux.
2. Ensure you have read-only access to the private key. Chmod is only supported on Linux subsystems (e.g. WSL on Windows or Terminal on Mac).
chmod 400 <keyname>.pem
3. Provide a path to your SSH private key file. Replace/reset your SSH private key.
Private key path
~/.ssh/azureuser
4. Run the example command below to connect to your VM.
ssh -i <private key path> azureuser@20.116.4.151
Can't connect?
Test your connection
Troubleshoot SSH connectivity issues

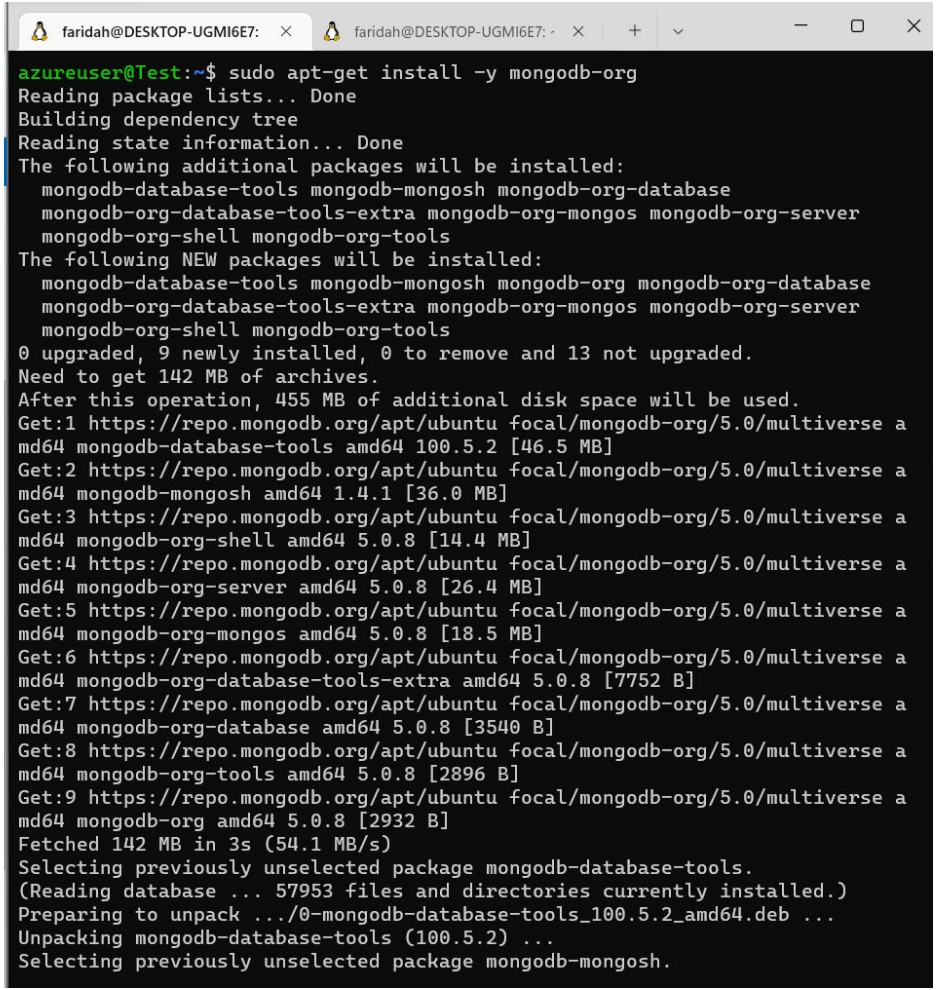

faridah@DESKTOP-UGMI6E7:
faridah@DESKTOP-UGMI6E7:
faridah@DESKTOP-UGMI6E7:/mnt/c/Users/Faridah/Downloads$ ssh -i ./Test_key.pem azureuser@20.151.211.65
The authenticity of host '20.151.211.65 (20.151.211.65)' can't be established.
ECDSA key fingerprint is SHA256:m8QYKVvtjaobhcaiK+fQ5e5Adb9ZaJgoQccMwLMLvuU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.151.211.65' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-1023-azure x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Tue May 17 05:29:22 UTC 2022
System load: 0.1 Processes: 122
Usage of /: 4.9% of 28.90GB Users logged in: 0
Memory usage: 3% IPv4 address for eth0: 10.1.0.4
Swap usage: 0%
1 update can be applied immediately.
To see these additional updates run: apt list --upgradable
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
azureuser@Test:~$ ls
file.txt
azureuser@Test:~$ ls

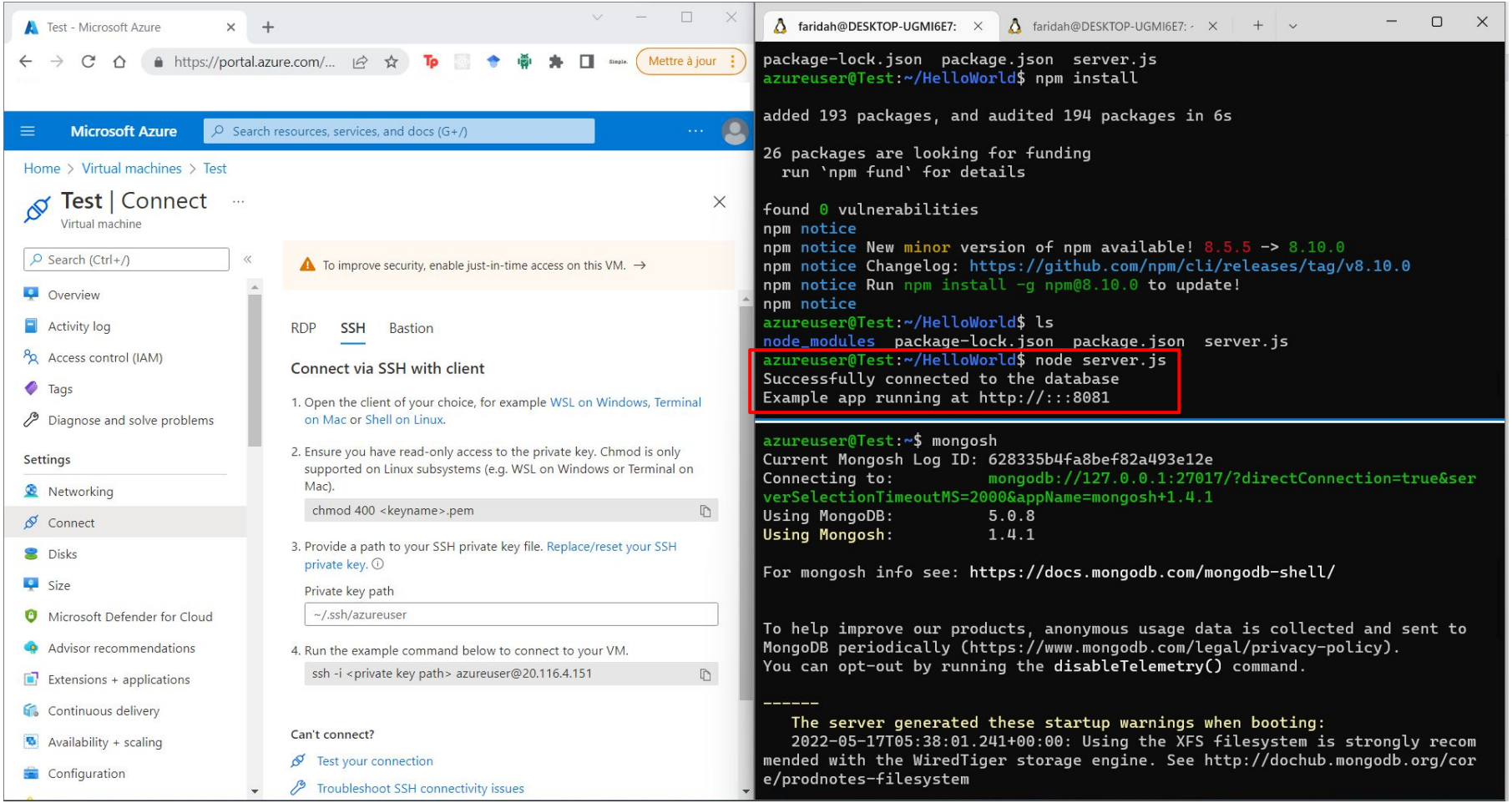
```
faridah@DESKTOP-UGMI6E7:~$ ls
HelloWorld  data  test.txt
faridah@DESKTOP-UGMI6E7:~$ cd HelloWorld/
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ ls
node_modules  package-lock.json  package.json  server.js
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ rm -rf node_modules/
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ ls
package-lock.json  package.json  server.js
faridah@DESKTOP-UGMI6E7:~/HelloWorld$ cd ..
faridah@DESKTOP-UGMI6E7:~$ scp -i /mnt/c/Users/Faridah/Downloads/Test_key.pe
m -r HelloWorld/ azureuser@20.151.211.65:HelloWorld
package-lock.json                             100%  131KB 513.5KB/s   00:00
package.json                                  100%  131     1.8KB/s   00:00
server.js                                     100% 1953    35.5KB/s   00:00
```

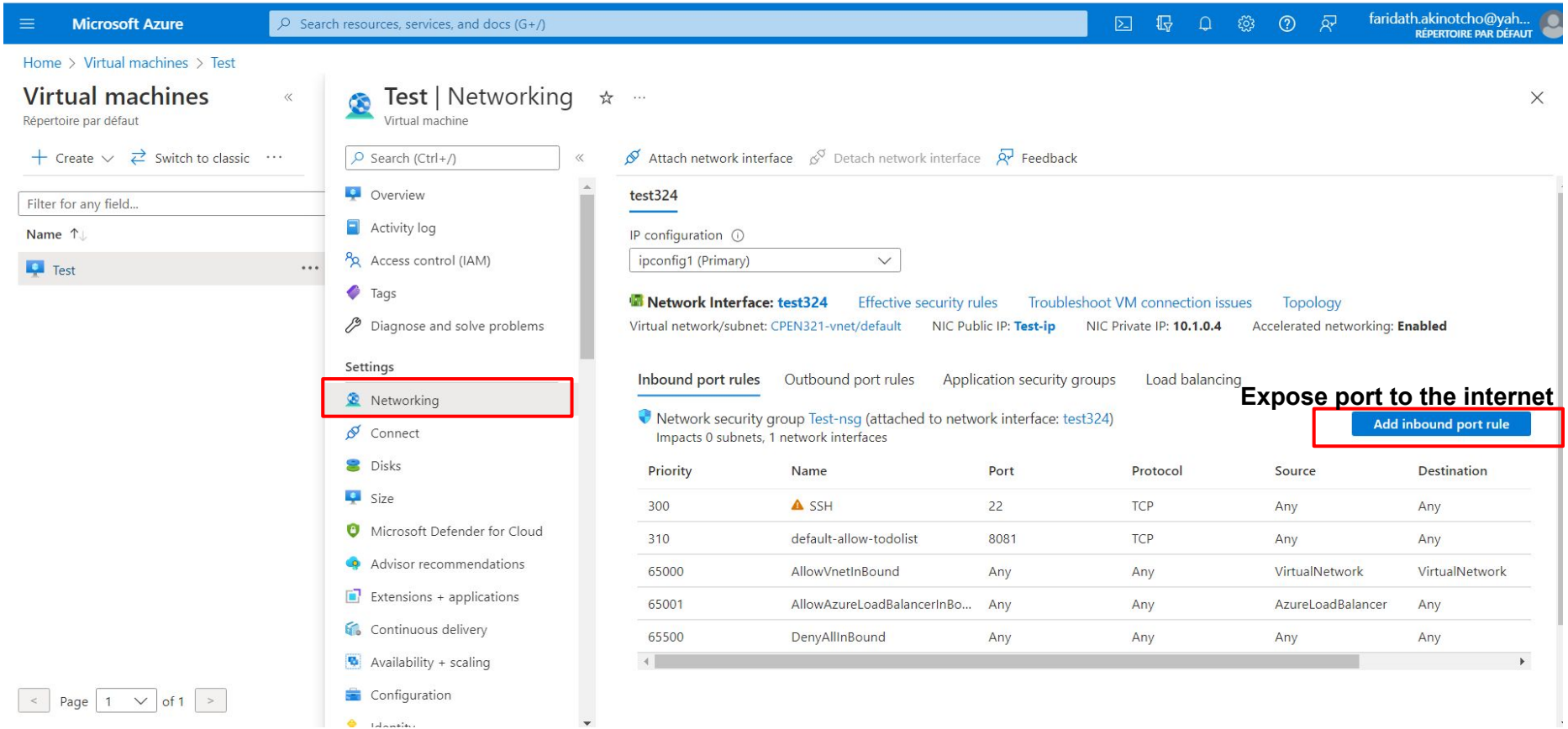


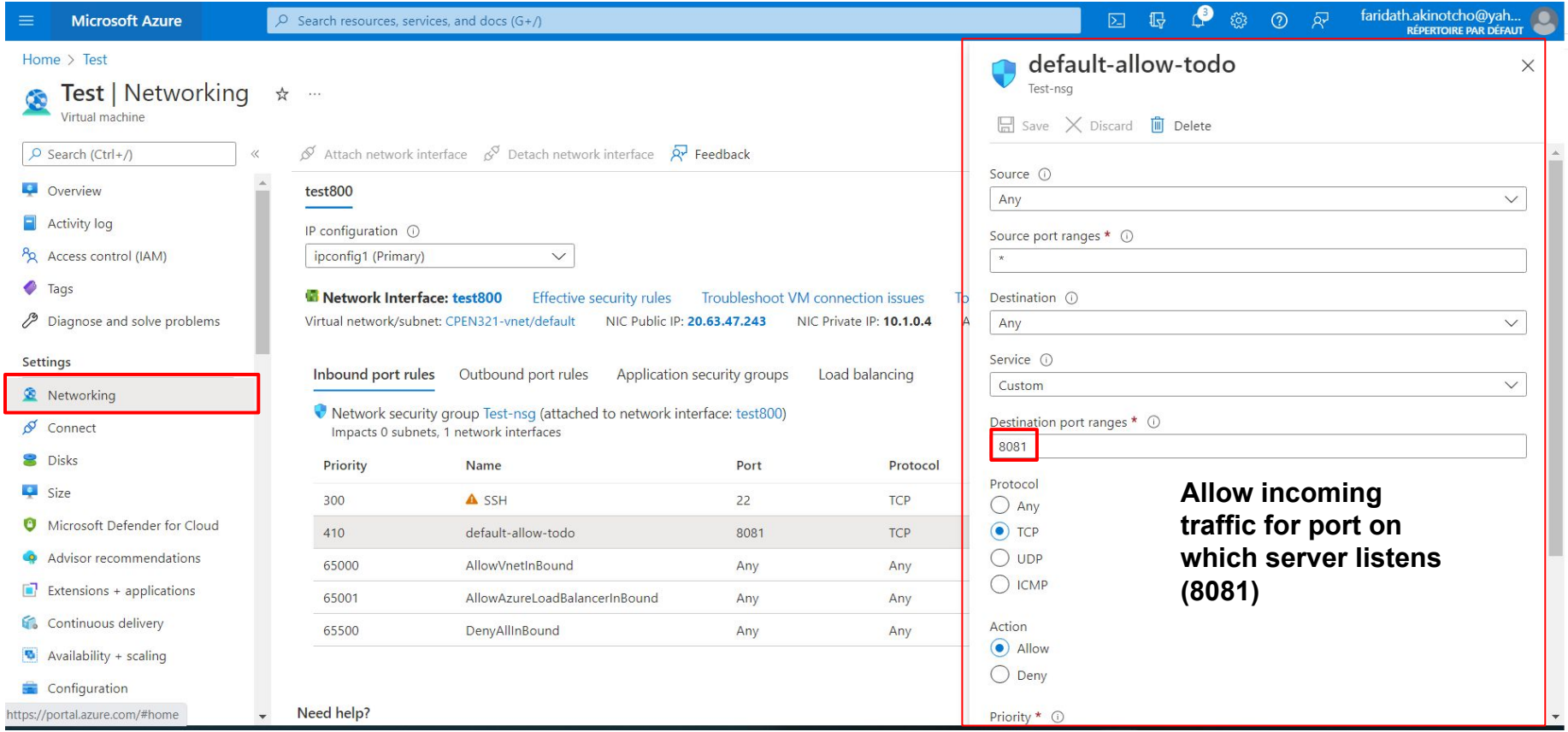
```
azureuser@Test:~$ ls
HelloWorld  file.txt
```

- **Install same dependencies as local (mongodb, nodejs) with same version**
- **Start mongodb server**




Test - Microsoft Azure
https://portal.azure.com/...
Mettre à jour
Microsoft Azure
Search resources, services, and docs (G+/)
Home > Virtual machines > Test
Test | Connect
Virtual machine
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Microsoft Defender for Cloud
Advisor recommendations
Extensions + applications
Continuous delivery
Availability + scaling
Configuration
To improve security, enable just-in-time access on this VM. →
RDP
SSH
Bastion
Connect via SSH with client
1. Open the client of your choice, for example WSL on Windows, Terminal on Mac or Shell on Linux.
2. Ensure you have read-only access to the private key. Chmod is only supported on Linux subsystems (e.g. WSL on Windows or Terminal on Mac).
chmod 400 <keyname>.pem
3. Provide a path to your SSH private key file. Replace/reset your SSH private key.
Private key path
~/.ssh/azureuser
4. Run the example command below to connect to your VM.
ssh -i <private key path> azureuser@20.116.4.151
Can't connect?
Test your connection
Troubleshoot SSH connectivity issues
faridah@DESKTOP-UGMI6E7:
faridah@DESKTOP-UGMI6E7:
package-lock.json package.json server.js
azureuser@Test:~/HelloWorld$ npm install
added 193 packages, and audited 194 packages in 6s
26 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.5.5 -> 8.10.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.10.0
npm notice Run npm install -g npm@8.10.0 to update!
npm notice
azureuser@Test:~/HelloWorld$ ls
node_modules package-lock.json package.json server.js
azureuser@Test:~/HelloWorld$ node server.js
Successfully connected to the database
Example app running at http://:::8081
azureuser@Test:~$ mongosh
Current Mongosh Log ID: 628335b4fa8bef82a493e12e
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.4.1
Using MongoDB: 5.0.8
Using Mongosh: 1.4.1
For mongosh info see: https://docs.mongodb.com/mongodb-shell/
To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.
The server generated these startup warnings when booting:
2022-05-17T05:38:01.241+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem


Microsoft Azure
Search resources, services, and docs (G+/)
faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT
Home > Virtual machines > Test
Virtual machines
Répertoire par défaut
Create
Switch to classic
Filter for any field...
Name
Test
Page 1 of 1
Test | Networking
Virtual machine
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Microsoft Defender for Cloud
Advisor recommendations
Extensions + applications
Continuous delivery
Availability + scaling
Configuration
Attach network interface
Detach network interface
Feedback
test324
IP configuration
ipconfig1 (Primary)
Network Interface: test324
Effective security rules
Troubleshoot VM connection issues
Topology
Virtual network/subnet: CPEN321-vnet/default
NIC Public IP: Test-ip
NIC Private IP: 10.1.0.4
Accelerated networking: Enabled
Inbound port rules
Outbound port rules
Application security groups
Load balancing
Expose port to the internet
Network security group Test-nsg (attached to network interface: test324)
Impacts 0 subnets, 1 network interfaces
Add inbound port rule
Priority
Name
Port
Protocol
Source
Destination
300
SSH
22
TCP
Any
Any
310
default-allow-todolist
8081
TCP
Any
Any
65000
AllowVnetInBound
Any
Any
VirtualNetwork
VirtualNetwork
65001
AllowAzureLoadBalancerInBo...
Any
Any
AzureLoadBalancer
Any
65500
DenyAllInBound
Any
Any
Any
Any


Microsoft Azure
Search resources, services, and docs (G+/)
faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT
Home > Test
Test | Networking
Virtual machine
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Microsoft Defender for Cloud
Advisor recommendations
Extensions + applications
Continuous delivery
Availability + scaling
Configuration
https://portal.azure.com/#home
Attach network interface
Detach network interface
Feedback
test800
IP configuration
ipconfig1 (Primary)
Network Interface: test800
Effective security rules
Troubleshoot VM connection issues
Virtual network/subnet: CPEN321-vnet/default
NIC Public IP: 20.63.47.243
NIC Private IP: 10.1.0.4
Inbound port rules
Outbound port rules
Application security groups
Load balancing
Network security group Test-nsg (attached to network interface: test800)
Impacts 0 subnets, 1 network interfaces
Priority
Name
Port
Protocol
300
SSH
22
TCP
410
default-allow-todo
8081
TCP
65000
AllowVnetInBound
Any
Any
65001
AllowAzureLoadBalancerInBound
Any
Any
65500
DenyAllInBound
Any
Any
Need help?
default-allow-todo
Test-nsg
Save
Discard
Delete
Source
Any
Source port ranges *
*
Destination
Any
Service
Custom
Destination port ranges *
8081
Protocol
Any
TCP
UDP
ICMP
Action
Allow
Deny
Priority *
Allow incoming traffic for port on which server listens (8081)

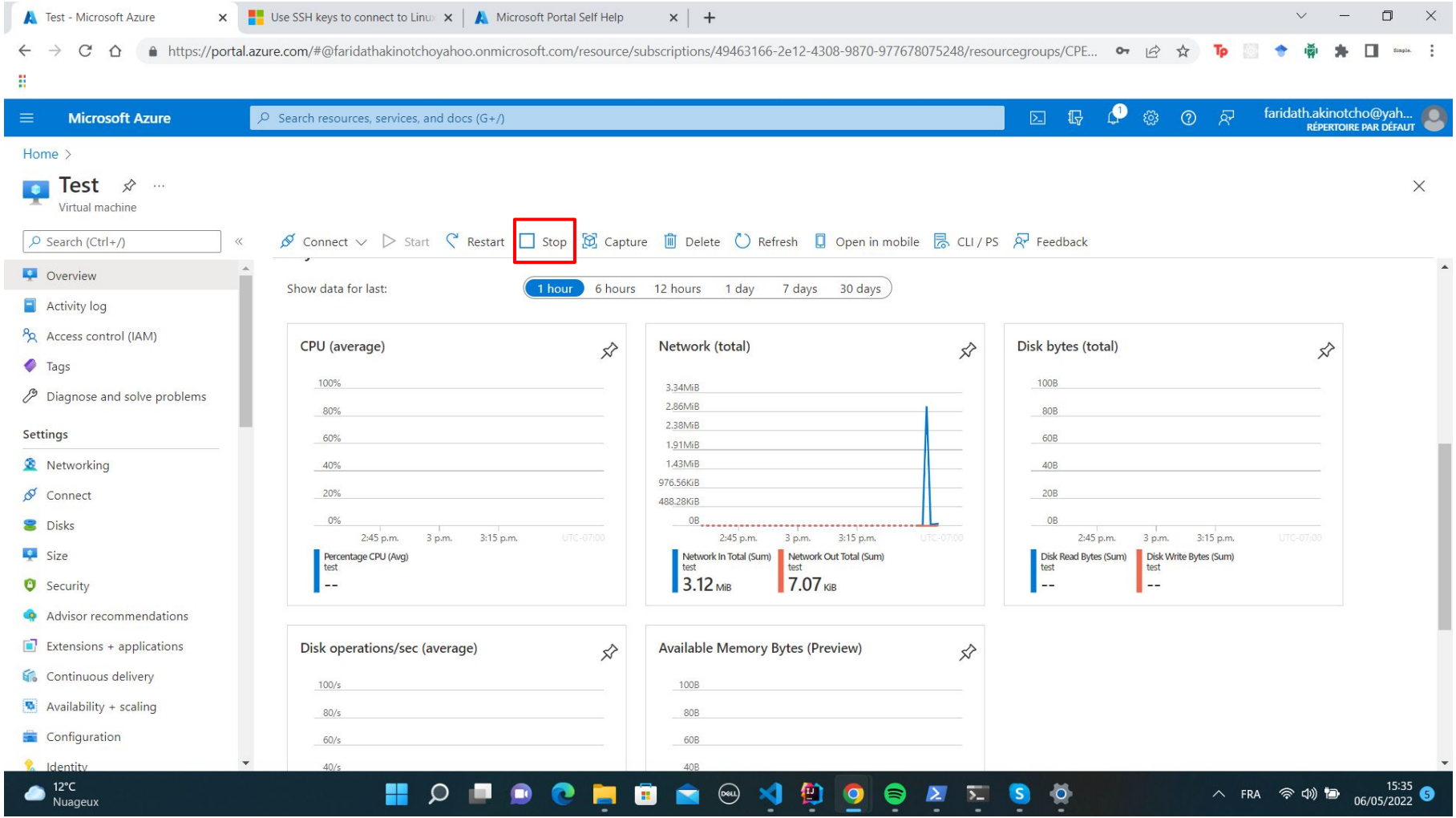
Local machine

```
faridah@DESKTOP-UGMI6E7:~$ curl -X POST -H "Content-Type: application/json"
-d '{"task": "Finish this tutorial", "status": "RIP"}' 20.151.211.65:8081/to
dolist
Todo item added successfully
faridah@DESKTOP-UGMI6E7:~$ curl -X PUT -H "Content-Type: application/json" -
d '{"task": "Finish this tutorial", "status": "jUST FOR FUN"}' 20.151.211.65
:8081/todolist
Todo item modified successfully
```

VM

```
package-lock.json  package.json  server.js
azureuser@Test:~/HelloWorld$ npm install

added 193 packages, and audited 194 packages in 6s

26 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.5.5 -> 8.10.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.10.0
npm notice Run npm install -g npm@8.10.0 to update!
npm notice
azureuser@Test:~/HelloWorld$ ls
node_modules  package-lock.json  package.json  server.js
azureuser@Test:~/HelloWorld$ node server.js
Successfully connected to the database
Example app running at http://:::8081
```

```
test> show collections
todolist
test> db.todolist.findOne()
{
  _id: ObjectId("6283366764cda1b7c0699e41"),
  task: 'Finish this tutorial',
  status: 'RIP'
}
test> db.todolist.find()
[
  {
    _id: ObjectId("6283366764cda1b7c0699e41"),
    task: 'Finish this tutorial',
    status: 'jUST FOR FUN'
  },
  {
    _id: ObjectId("6283368164cda1b7c0699e42"),
    task: 'Finish this tutorial',
    status: 'RIP'
```


Test - Microsoft Azure
Use SSH keys to connect to Linux
Microsoft Portal Self Help
https://portal.azure.com/#@faridathakinotchoyahoo.onmicrosoft.com/resource/subscriptions/49463166-2e12-4308-9870-977678075248/resourcegroups/CPE...
Microsoft Azure
Search resources, services, and docs (G+/)
faridath.akinotcho@yah...
RÉPERTOIRE PAR DÉFAUT
Home
Test
Virtual machine
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Security
Advisor recommendations
Extensions + applications
Continuous delivery
Availability + scaling
Configuration
Identity
Connect
Start
Restart
Stop
Capture
Delete
Refresh
Open in mobile
CLI / PS
Feedback
Show data for last:
1 hour
6 hours
12 hours
1 day
7 days
30 days
CPU (average)
100%
80%
60%
40%
20%
0%
2:45 p.m.
3 p.m.
3:15 p.m.
Percentage CPU (Avg)
test
--
Network (total)
3.34MiB
2.86MiB
2.38MiB
1.91MiB
1.43MiB
976.56KiB
488.28KiB
0B
Network In Total (Sum)
test
3.12 MiB
Network Out Total (Sum)
test
7.07 KiB
Disk bytes (total)
100B
80B
60B
40B
20B
0B
Disk Read Bytes (Sum)
test
--
Disk Write Bytes (Sum)
test
--
Disk operations/sec (average)
100/s
80/s
60/s
40/s
Available Memory Bytes (Preview)
100B
80B
60B
40B
12°C
Nuageux
FRA
15:35
06/05/2022

See you in the Q&A session!