



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Tarea 4. Implementación de un token-ring

PRESENTA
Vladimir Azpeitia Hernández

PROFESOR
Carlos Pineda Guerrero

ASIGNATURA
Desarrollo de sistemas distribuidos

29 de marzo de 2021

Tarea 4. Implementación de un token-ring

Vladimir Azpeitia Hernández

29 de marzo de 2021

1. Introducción

Desarrollar un programa en Java, el cual implementará un token que pasará de un nodo a otro nodo, en una topología lógica de anillo. El anillo constará de tres nodos:

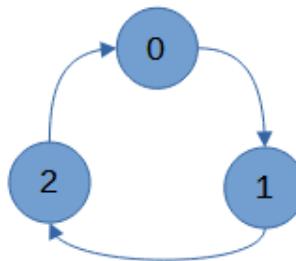


Figura 1: Token-Ring

1. El token será un número entero de 32 bits. El nodo 0 inicializará el token con 1.
2. El nodo 0 enviará el token al nodo 1, entonces el nodo 1 recibirá el token y lo enviará al nodo 2. El nodo 2 recibirá el token y lo enviará al nodo 0.
3. Cada nodo contará y desplegará las veces que recibe el token. Cuando la cuenta en el nodo 0 llegue a 1000, el nodo 0 deberá terminar su ejecución, entonces ¿qué pasará con los otros nodos?

2. Desarrollo

2.1. Código fuente del programa

```
1 import java.net.Socket;
2 import java.net.ServerSocket;
3 import java.io.DataOutputStream;
4 import java.io.DataInputStream;
5
6 class Token {
7     static DataInputStream entrada;
8     static DataOutputStream salida;
9     static boolean primera_vez = true;
10    static String ip;
11    static int nodo;
12    static int token;
13    static int contador = 0;
14
15    static class Worker extends Thread {
16        public void run() {
17            try {
18                ServerSocket servidor = new ServerSocket(5000);
19                Socket conexion = servidor.accept();
20                DataInputStream(conexion.getInputStream())
21                entrada = new DataInputStream(conexion.getInputStream());
22            } catch (Exception e) {
23                e.printStackTrace();
24            }
25        }
26    }
27}
```

```

28 public static void main(String[] args) throws Exception {
29     if (args.length != 2) {
30         System.err.println("usage: java Token <nodo> <ip>");
31         System.exit(1);
32     }
33     nodo = Integer.valueOf(args[0]);
34     ip = args[1];
35
36     Worker w = new Worker();
37     w.start();
38
39     Socket conexion = null;
40     while(true) {
41         try {
42             conexion = new Socket(ip, 5000);
43             break;
44         }catch (Exception e) {
45             Thread.sleep(500);
46         }
47     }
48     salida = new DataOutputStream(conexion.getOutputStream());
49     w.join();
50
51     while(true) {
52         if(nodo == 0) {
53             if(primeravez == true) {
54                 primeravez = false;
55                 token = 1;
56             } else {
57                 token = entrada.readInt();
58                 contador++;
59                 System.out.printf("nodo=%d\t contador=%d\t token=%d\n", nodo, contador, token);
60             }
61         } else {
62             entrada.readInt();
63             token = entrada.readInt();
64             contador++;
65             System.out.printf("nodo = %d \t contador = %d \t token = %d \n", nodo, contador, token);
66         }
67         if(nodo == 0 && contador ==1000) {
68             break;
69         }
70         salida.writeInt(token);
71     }
72 }
73 }
```

2.2. Instalación y configuración de las máquinas virtuales

A continuación se muestran las capturas de la creación de las máquinas virtuales, una para cada nodo, con el servicio de Microsoft Azure. Cada máquina tiene Ubuntu con 1 CPU, 1 GB de RAM y disco HDD estándar.

2.2.1. Nodo 0

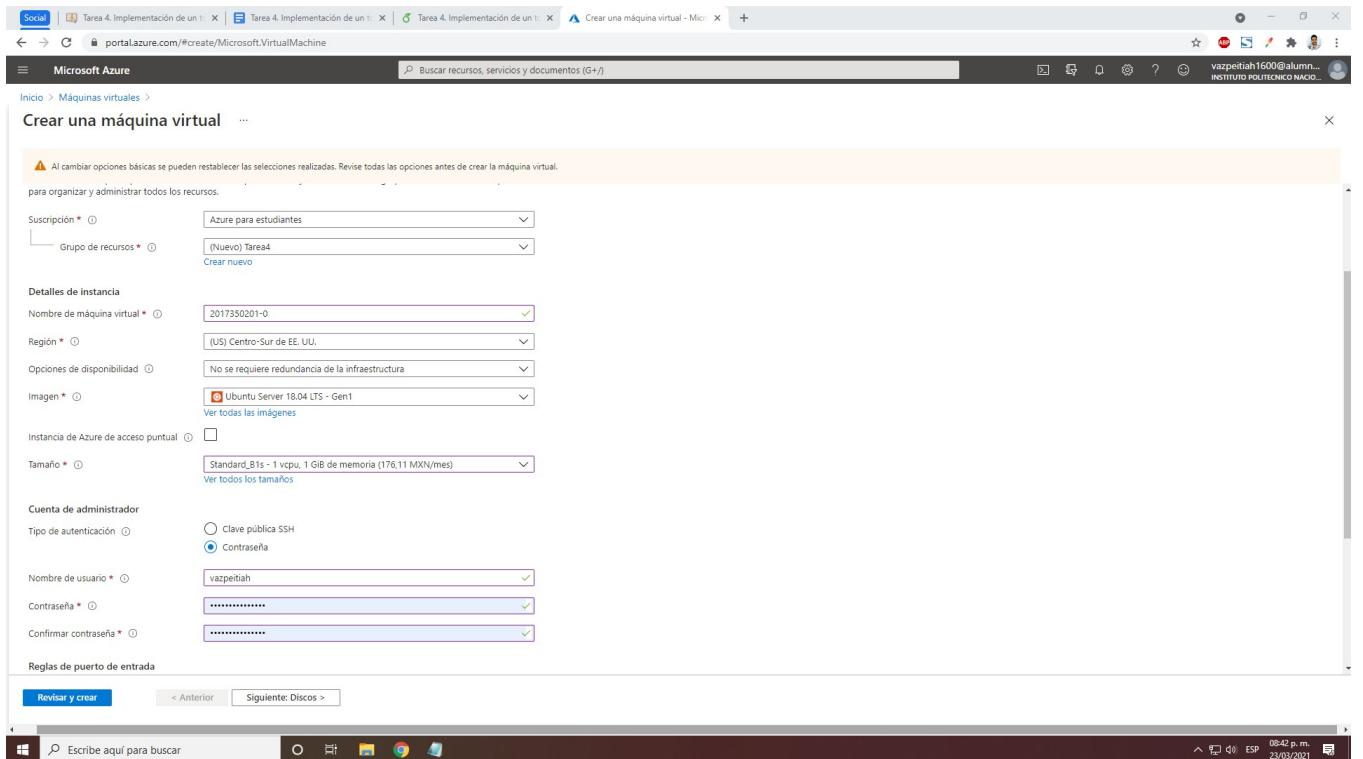


Figura 2: Nodo 0. Información básica de la máquina virutal

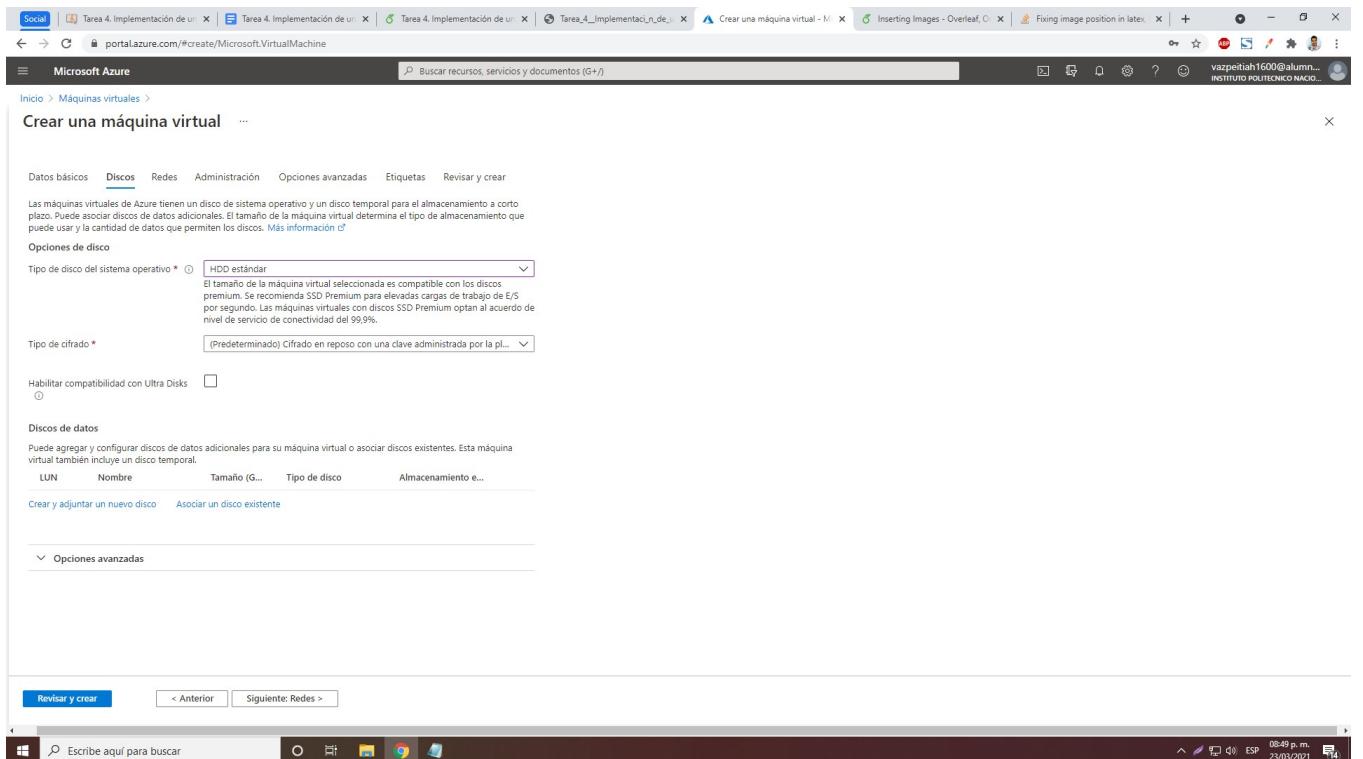


Figura 3: Nodo 0. Información acerca de los discos

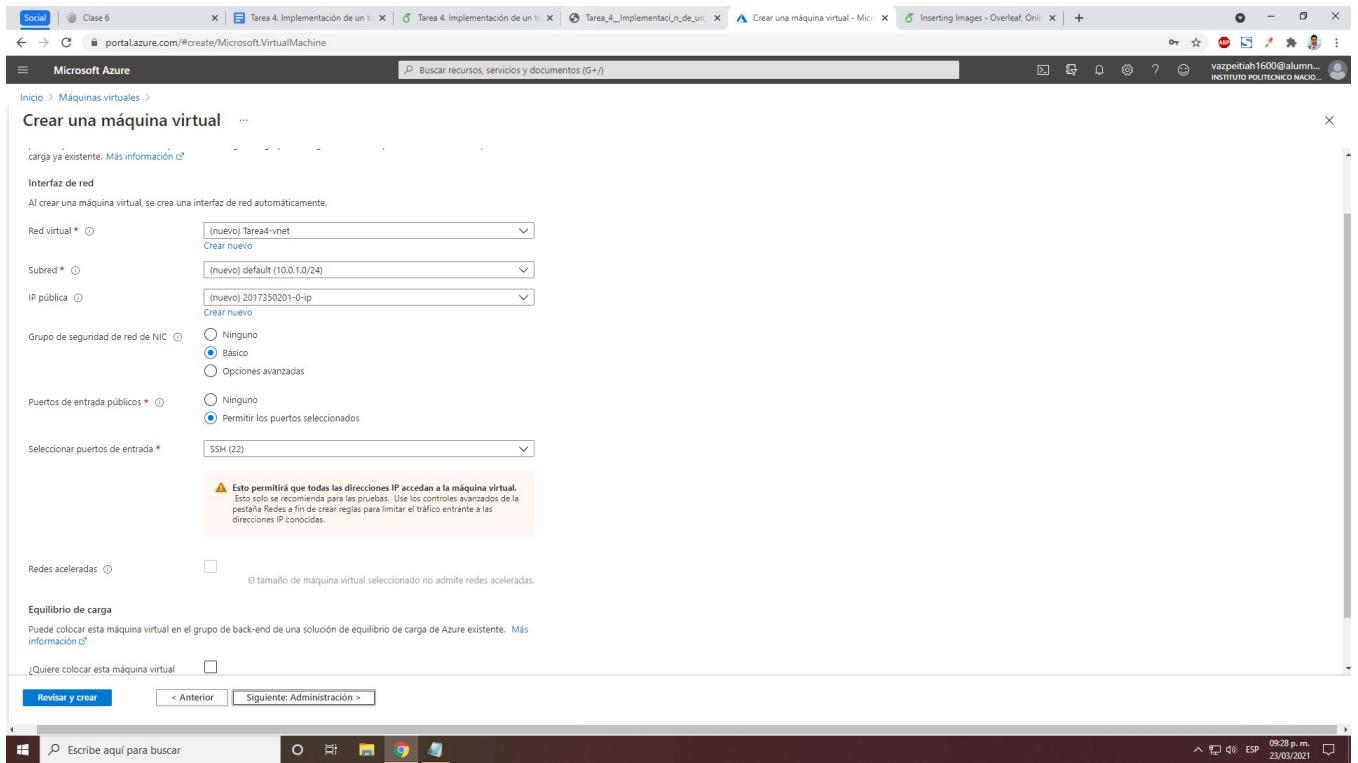


Figura 4: Nodo 0. Información acerca de las redes

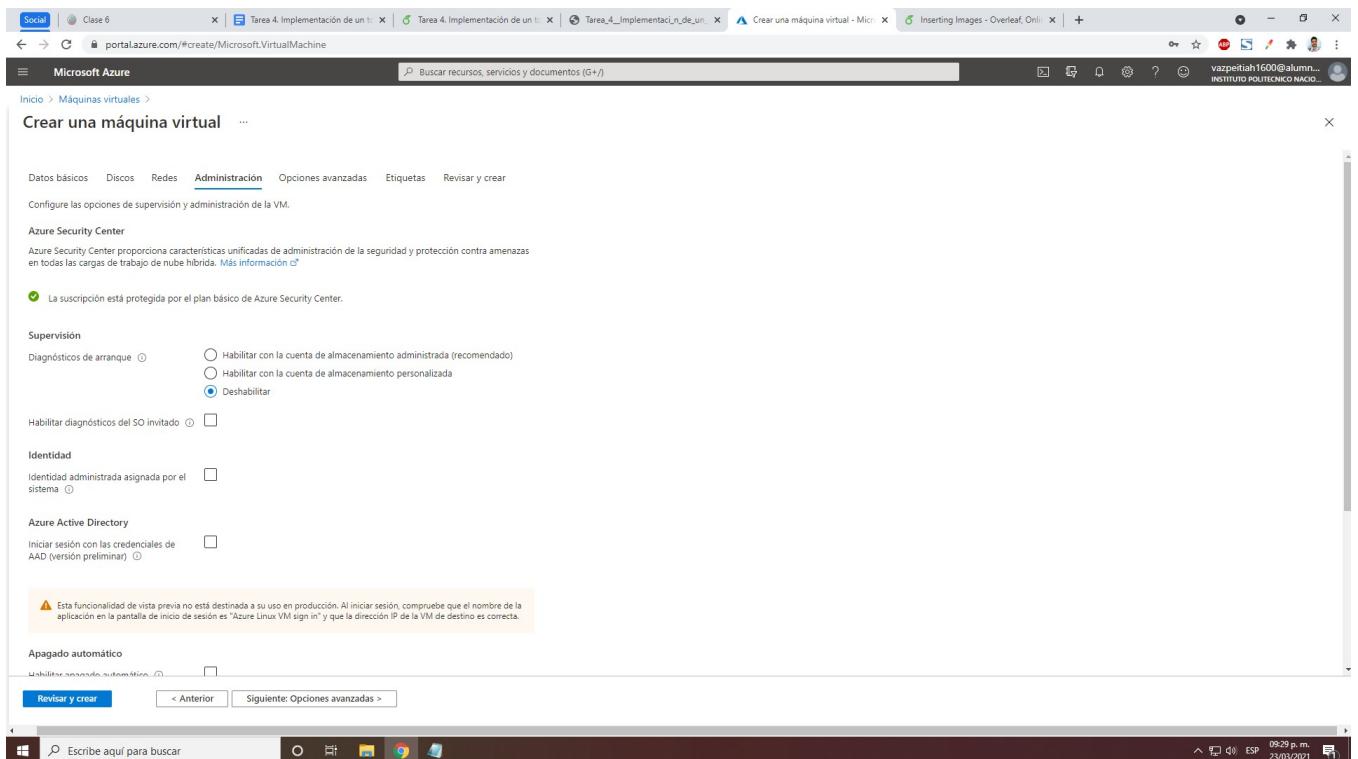


Figura 5: Nodo 0. Información acerca de la administración

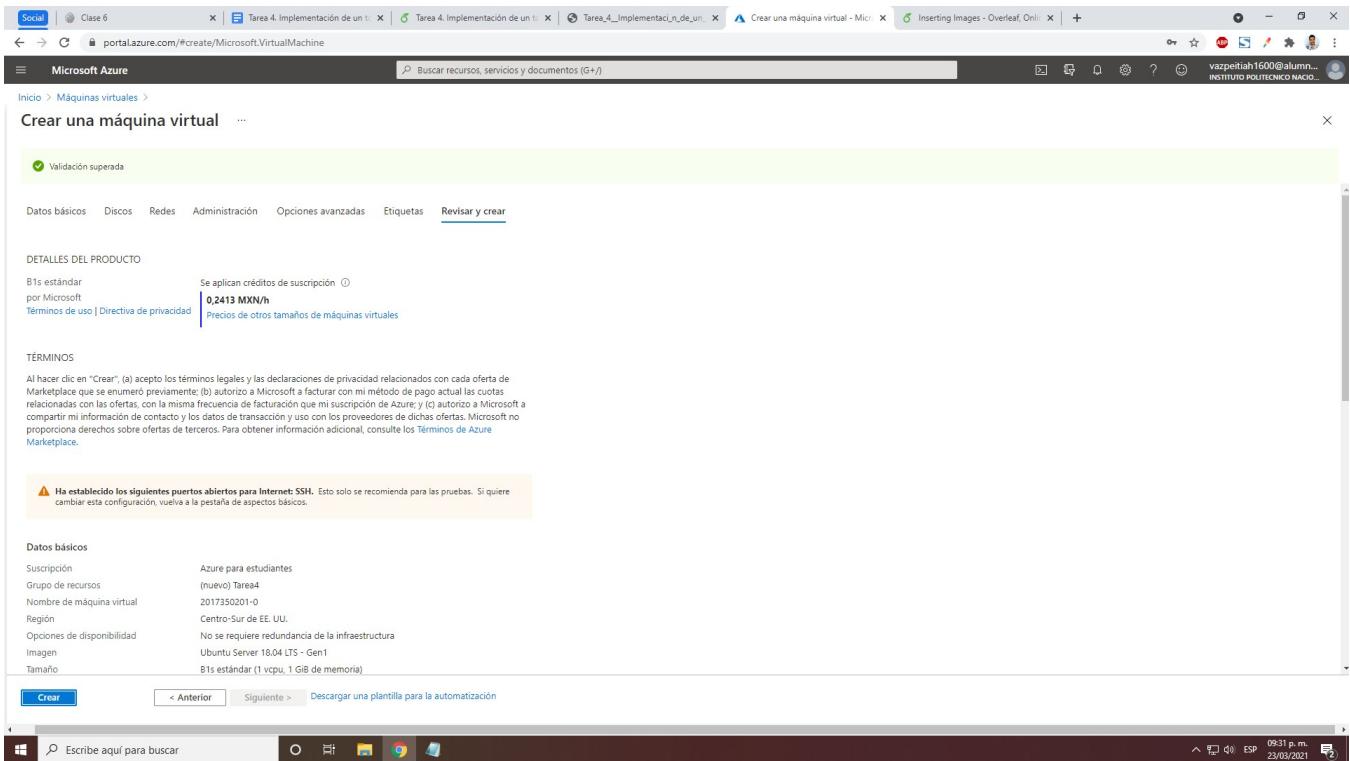


Figura 6: Nodo 0. Validación superada

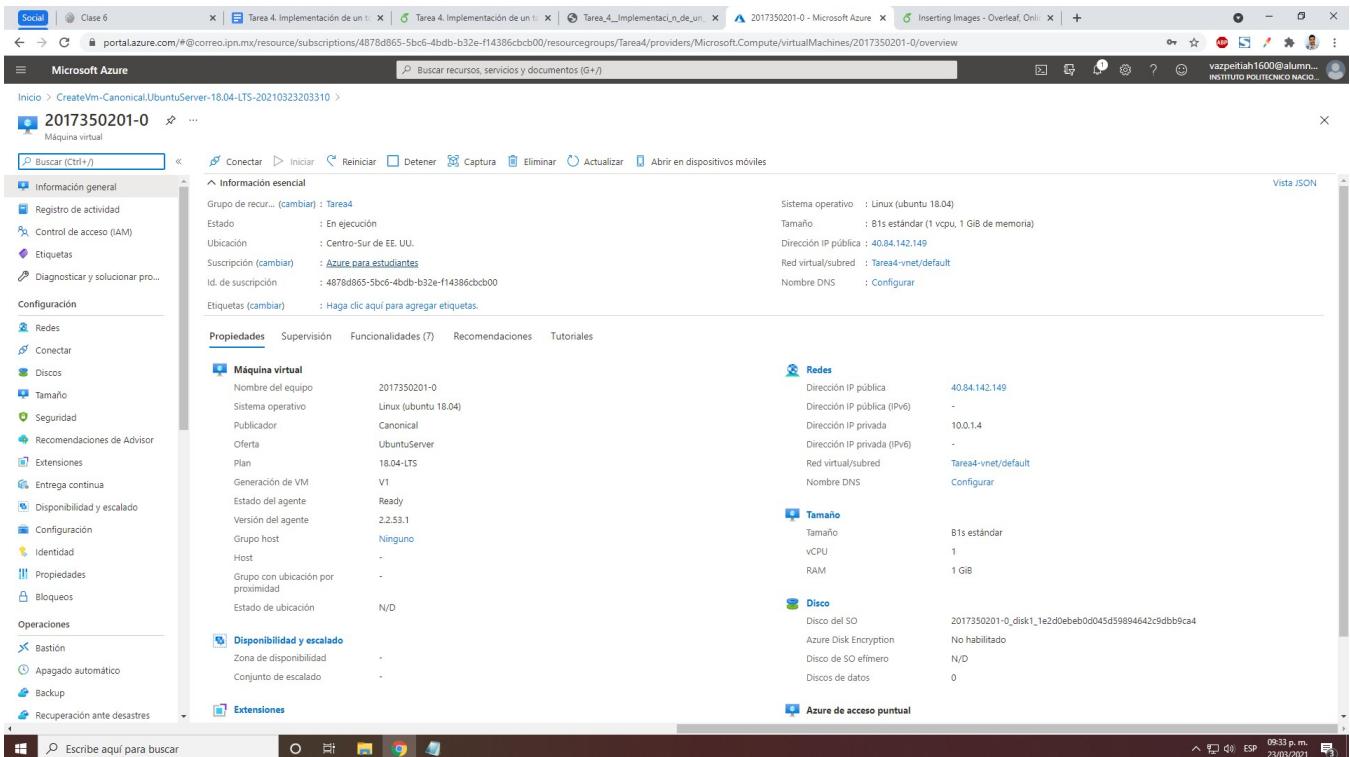


Figura 7: Nodo 0. Panel de control de la máquina virutal

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	Cualquiera	Cualquiera	Permitir
310	Puerto_50000	5000	TCP	Cualquiera	Cualquiera	Permitir
65000	AllowVnetInBound	Cualquier	Cualquier	VirtualNetwork	VirtualNetwork	Permitir
65001	AllowAzureLoadBalancerInBound	Cualquier	Cualquier	AzureLoadBalancer	Cualquier	Permitir
65500	DenyAllInBound	Cualquier	Cualquier	Cualquier	Cualquier	Denegar

Figura 8: Nodo 0. Puerto 5000 abierto

```

vaspeitah@2017350201-0:~$ ssh vaspeitah@40.84.142.149
password: 
Welcome to Ubuntu 18.04 LTS (GNU/Linux 5.4.0-1041-azure x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: http://ubuntu.com/advantage

System information as of Wed Mar 24 04:12:54 UTC 2021

System load: 0.0          Processes: 109
Usage of /: 4.1% of 28.90GB   Users logged in: 0
Memory usage: 20%
Swap usage: 0%
0 packages can be updated,
0 of these updates are security updates.

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.

vaspeitah@2017350201-0:~$ 

```

Figura 9: Nodo 0. Conectarse con la máquina virutal a través de ssh

2.2.2. Nodo 1

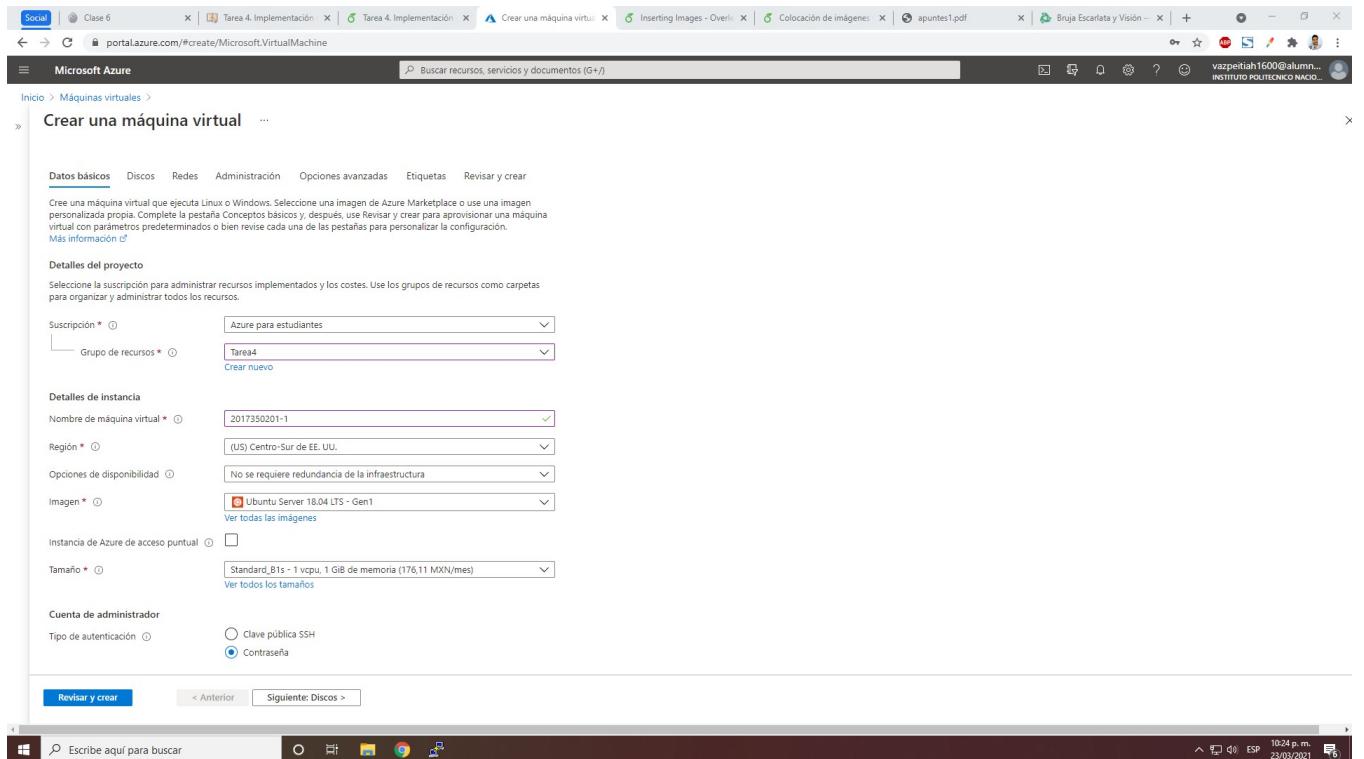


Figura 10: Nodo 1. Información básica de la máquina virutal

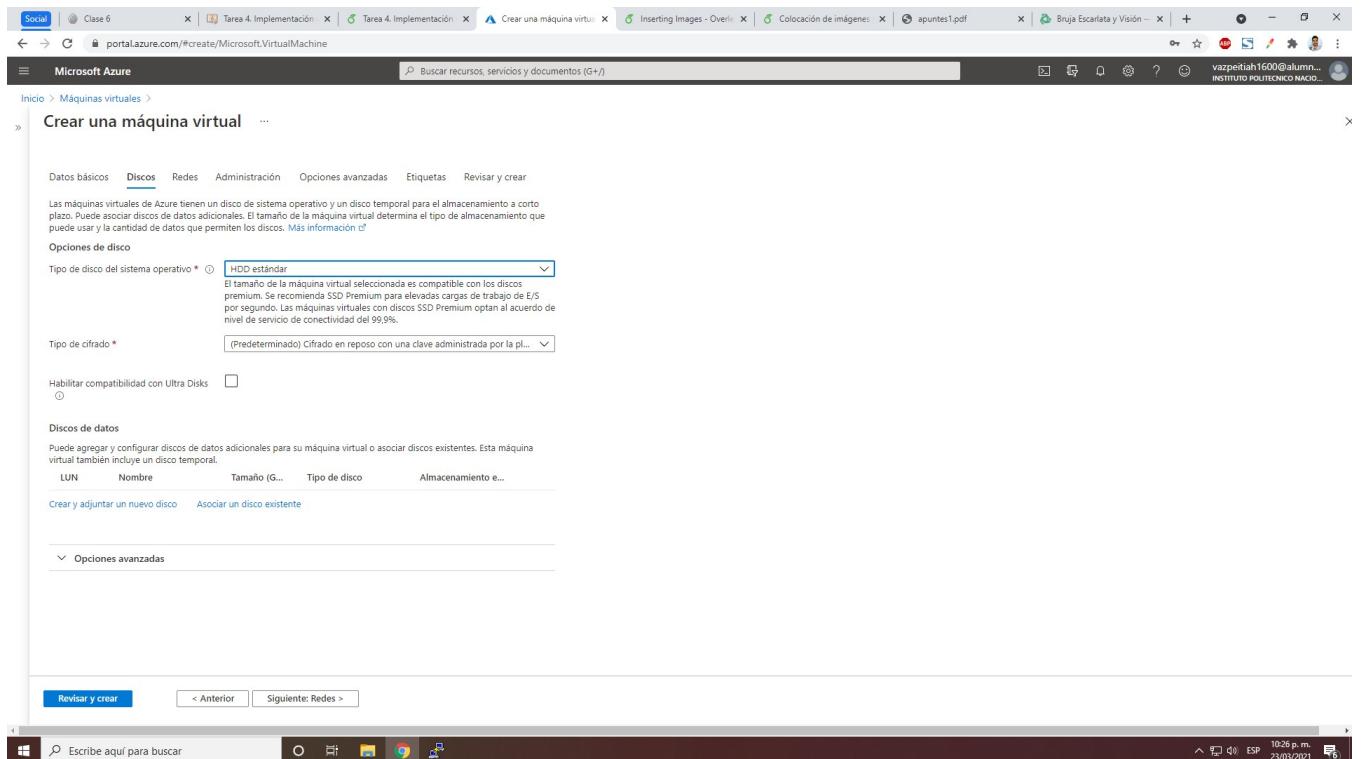


Figura 11: Nodo 1. Información acerca de los discos

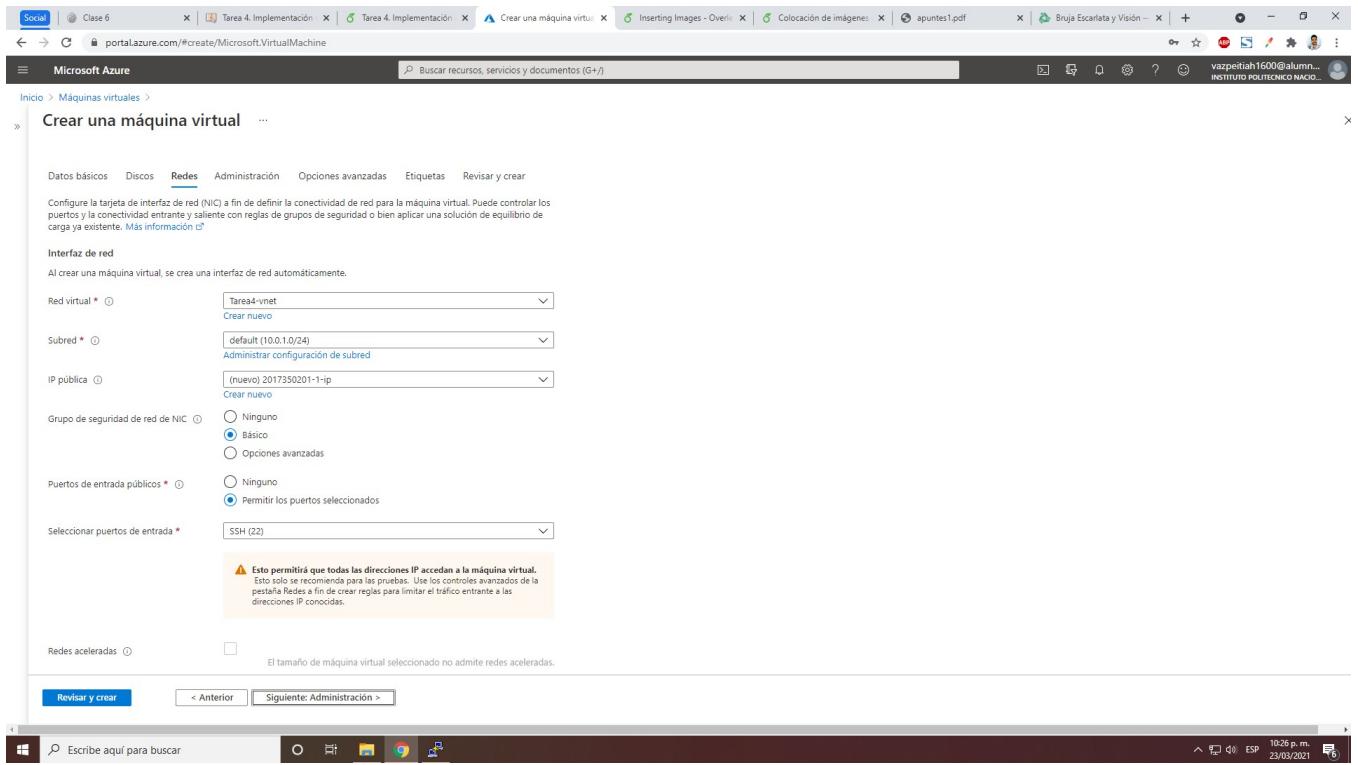


Figura 12: Nodo 1. Información acerca de las redes

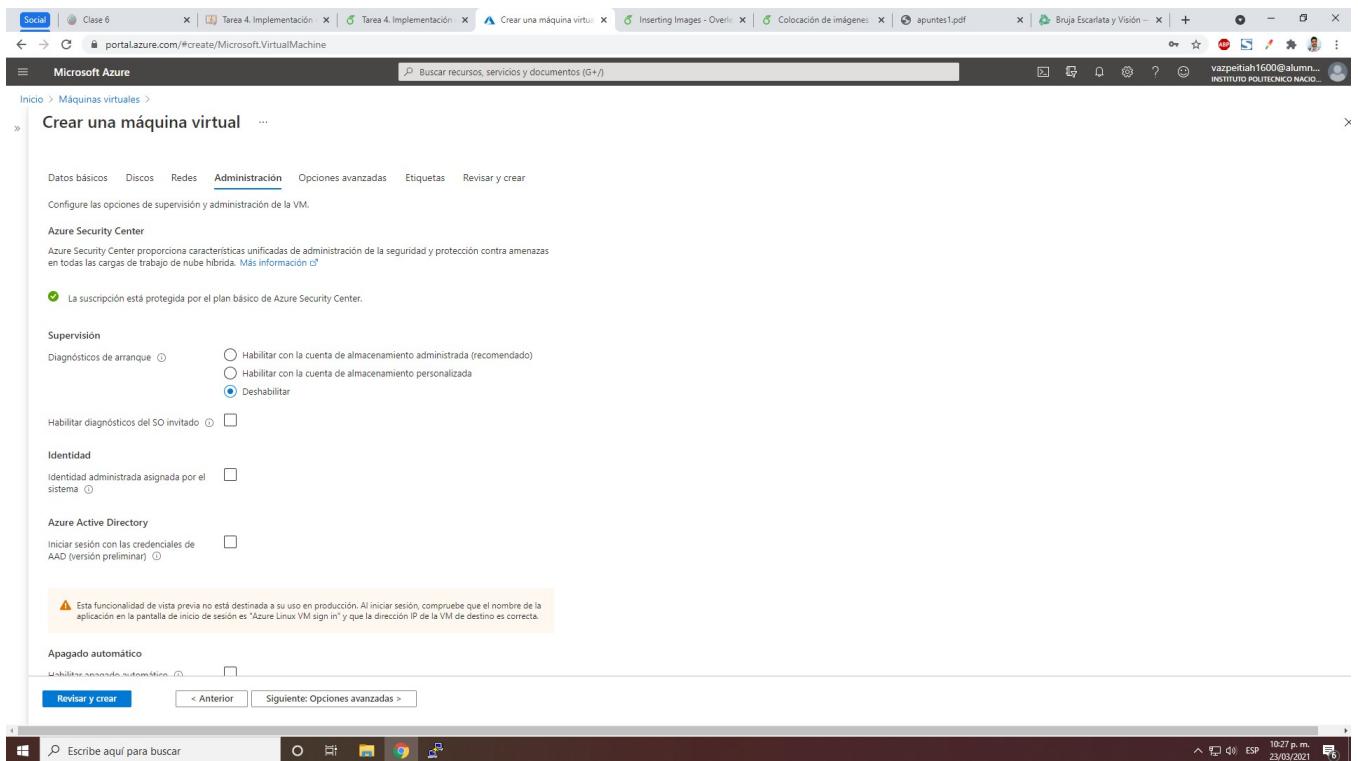


Figura 13: Nodo 1. Información acerca de la administración

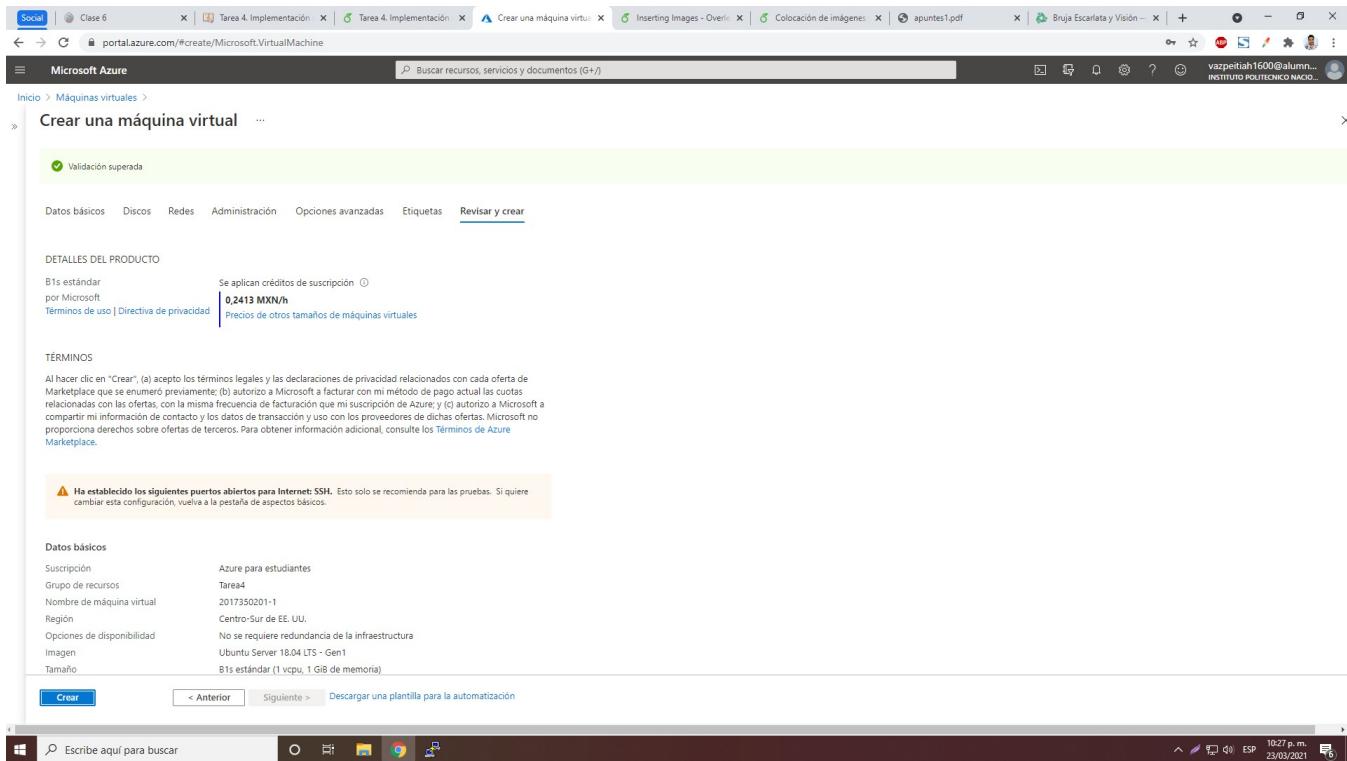


Figura 14: Nodo 1. Validación superada

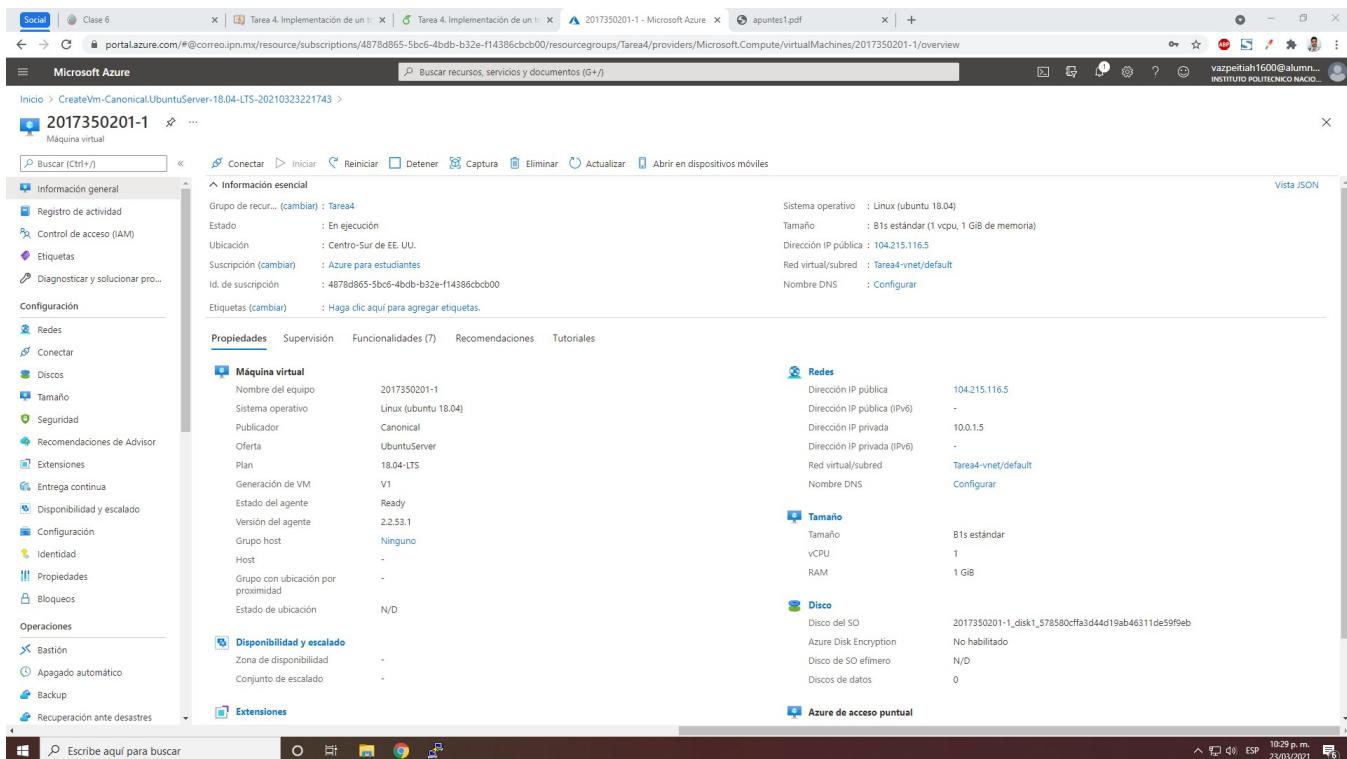


Figura 15: Nodo 1. Panel de control de la máquina virtual

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	Cualquiera	Cualquiera	<input checked="" type="radio"/> Permitir
310	Port_5000	5000	TCP	Cualquiera	Cualquiera	<input checked="" type="radio"/> Permitir
65000	AllowVnetInBound	Cualquier	Cualquier	VirtualNetwork	VirtualNetwork	<input checked="" type="radio"/> Permitir
65001	AllowAzureLoadBalancerInBound	Cualquier	Cualquier	AzureLoadBalancer	Cualquier	<input checked="" type="radio"/> Permitir
65500	DenyAllInBound	Cualquier	Cualquier	Cualquier	Cualquier	<input type="radio"/> Denegar

Figura 16: Nodo 1. Puerto 5000 abierto

```

vazpeitiah@2017350201-1: ~
Login as: vazpeitiah
vazpeitiah@104.215.16.5's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1041-azure x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Wed Mar 24 04:33:20 UTC 2021

System load: 0.13      Processes: 109
Usage of /: 4.5% of 28.90GB  Users logged in: 0
Memory usage: 20%
Swap usage: 0%
0 packages can be updated.
0 of these updates are security updates.

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.

vazpeitiah@2017350201-1:~$ 

```

Figura 17: Nodo 1. Conectarse con la máquina virutal a través de ssh

2.2.3. Nodo 2

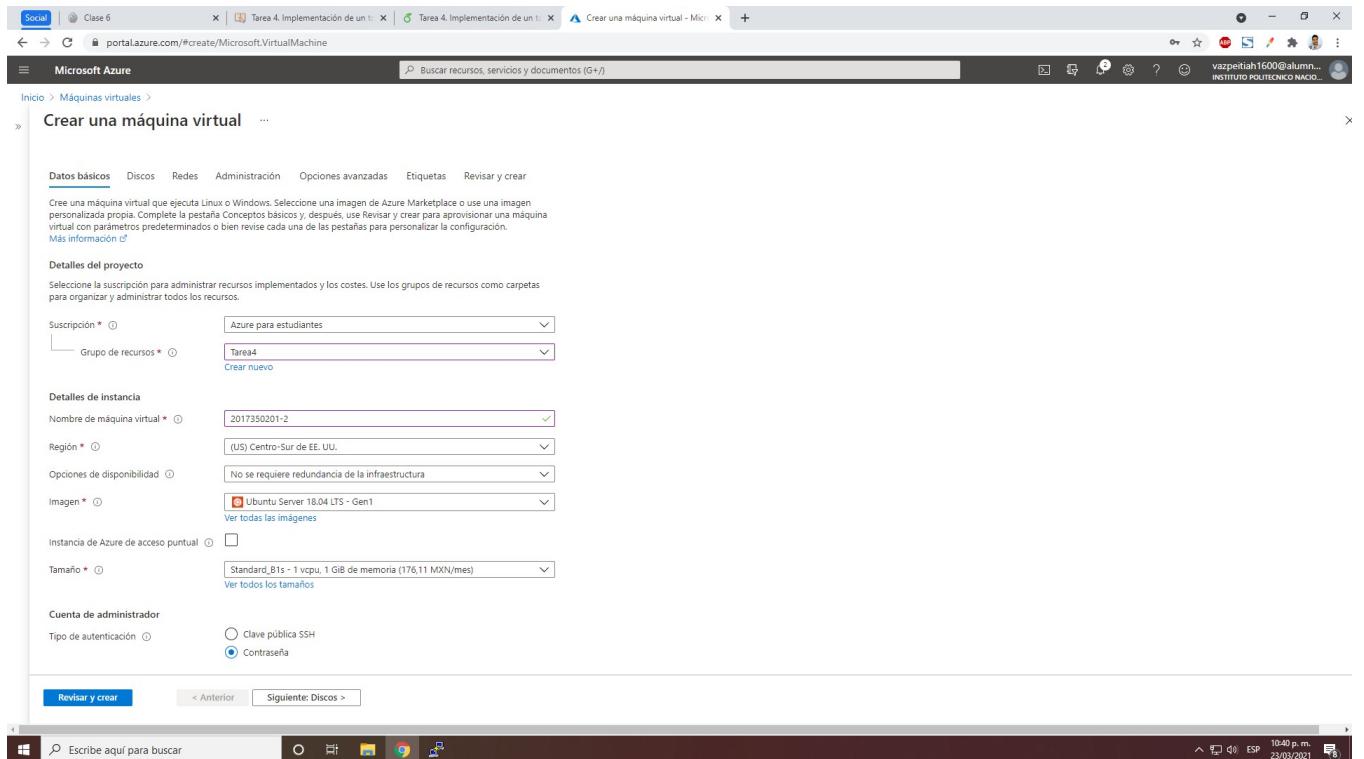


Figura 18: Nodo 2. Información básica de la máquina virutal

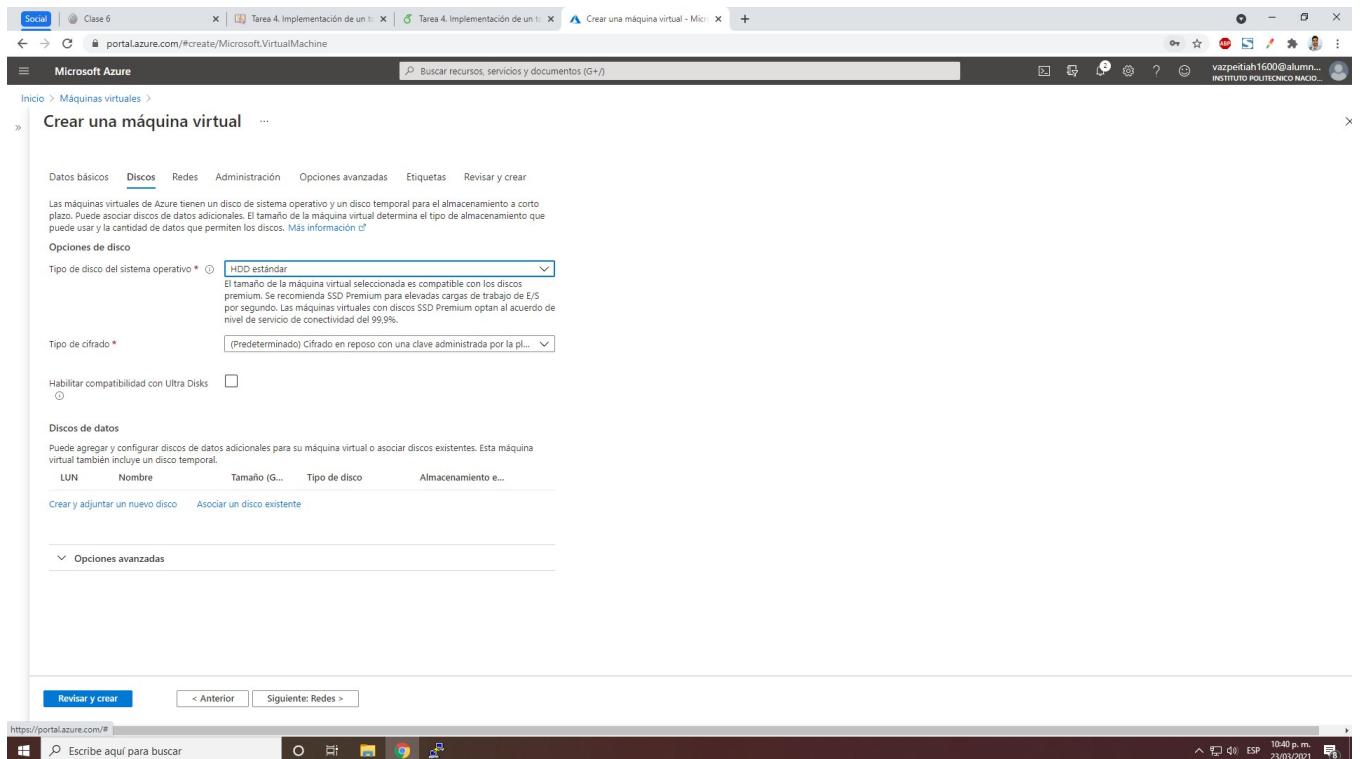


Figura 19: Nodo 2. Información acerca de los discos

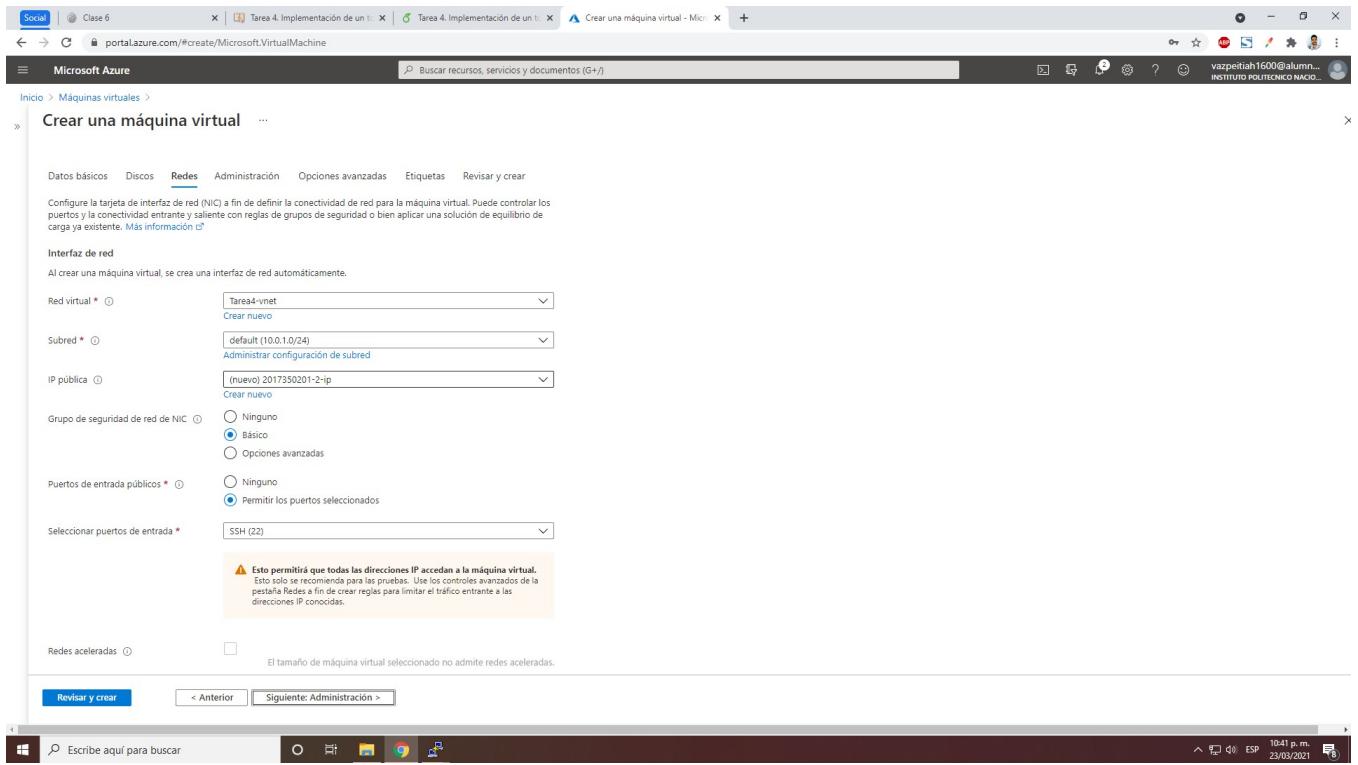


Figura 20: Nodo 2. Información acerca de las redes

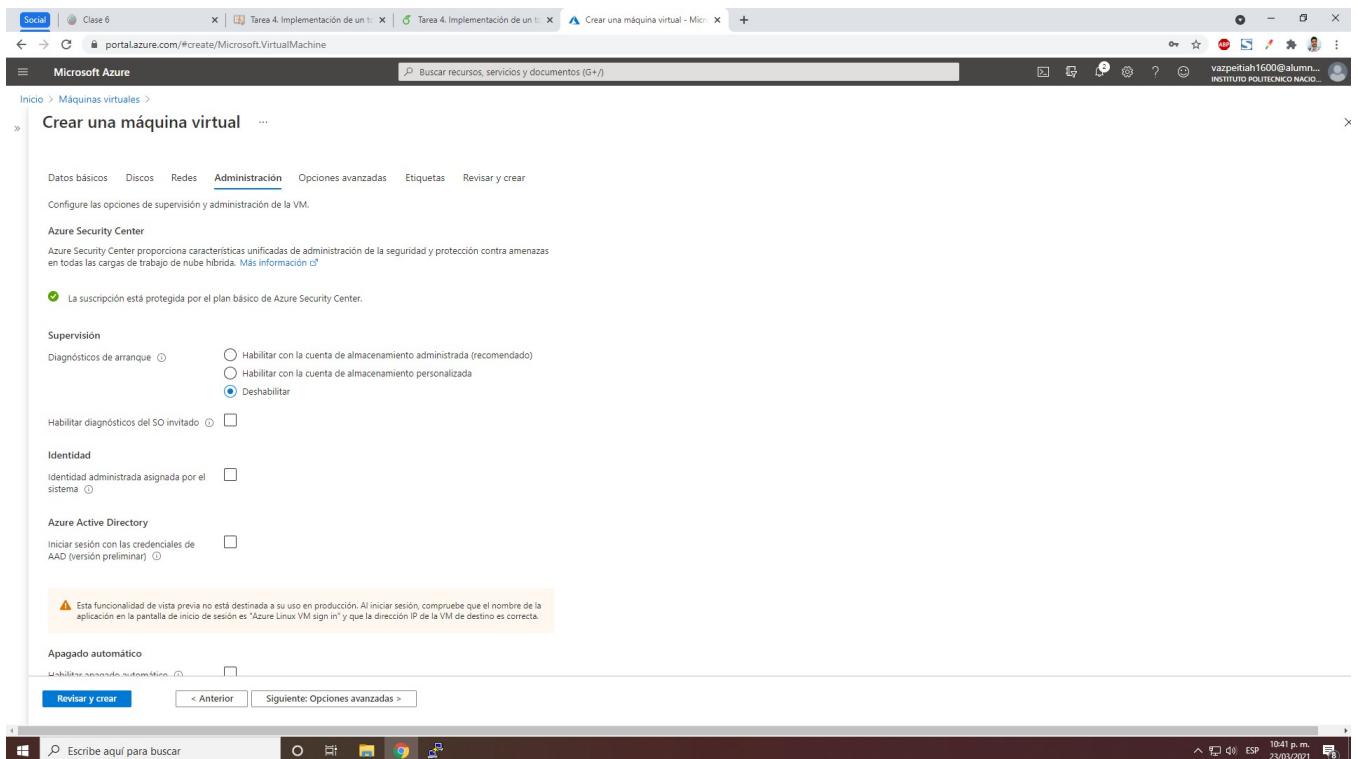


Figura 21: Nodo 2. Información acerca de la administración

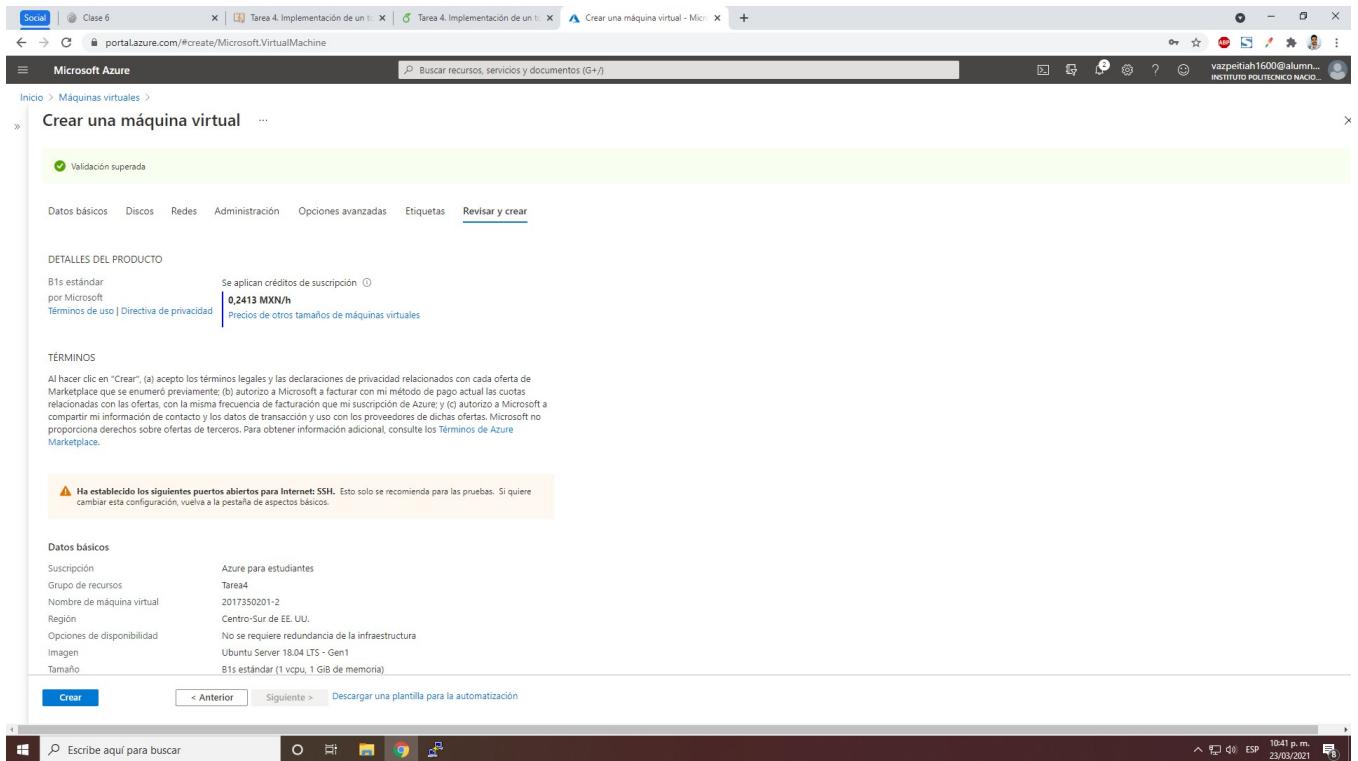


Figura 22: Nodo 2. Validación superada

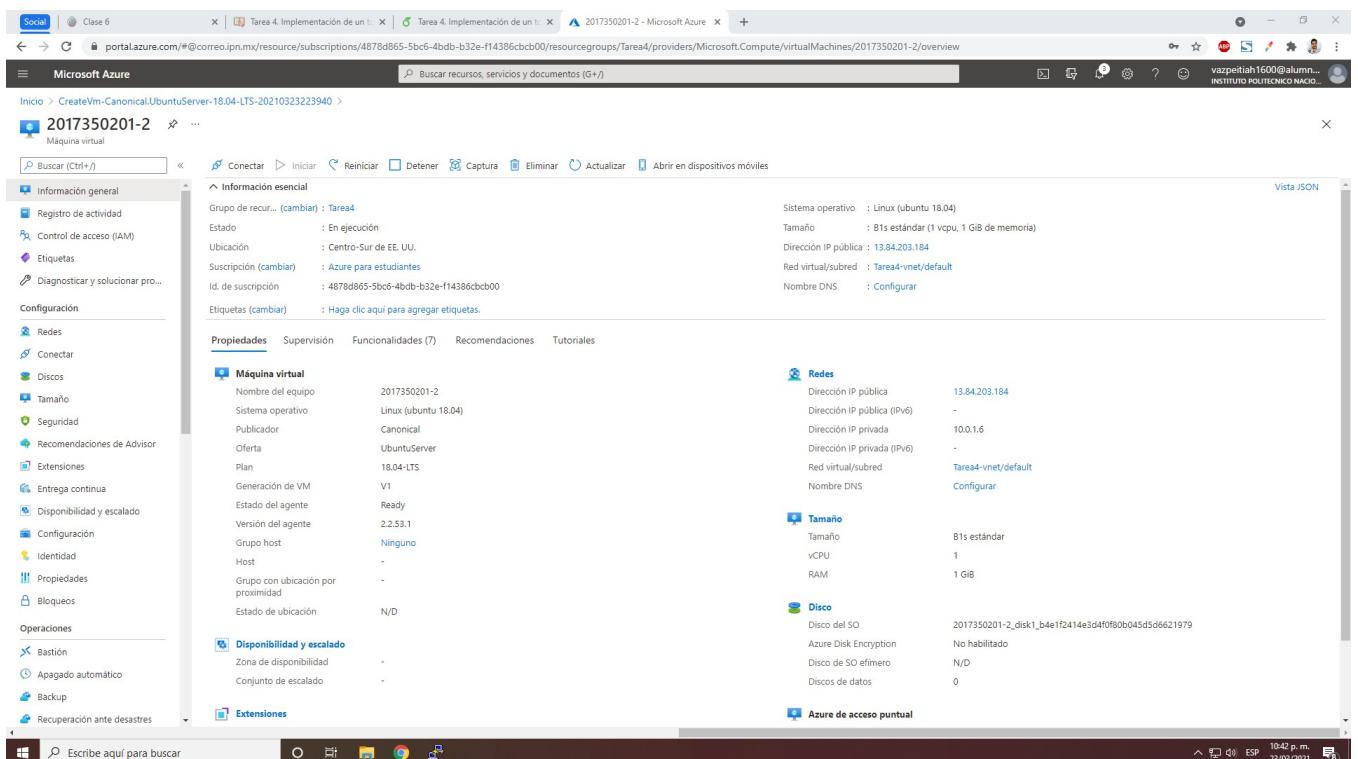


Figura 23: Nodo 2. Panel de control de la máquina virutal

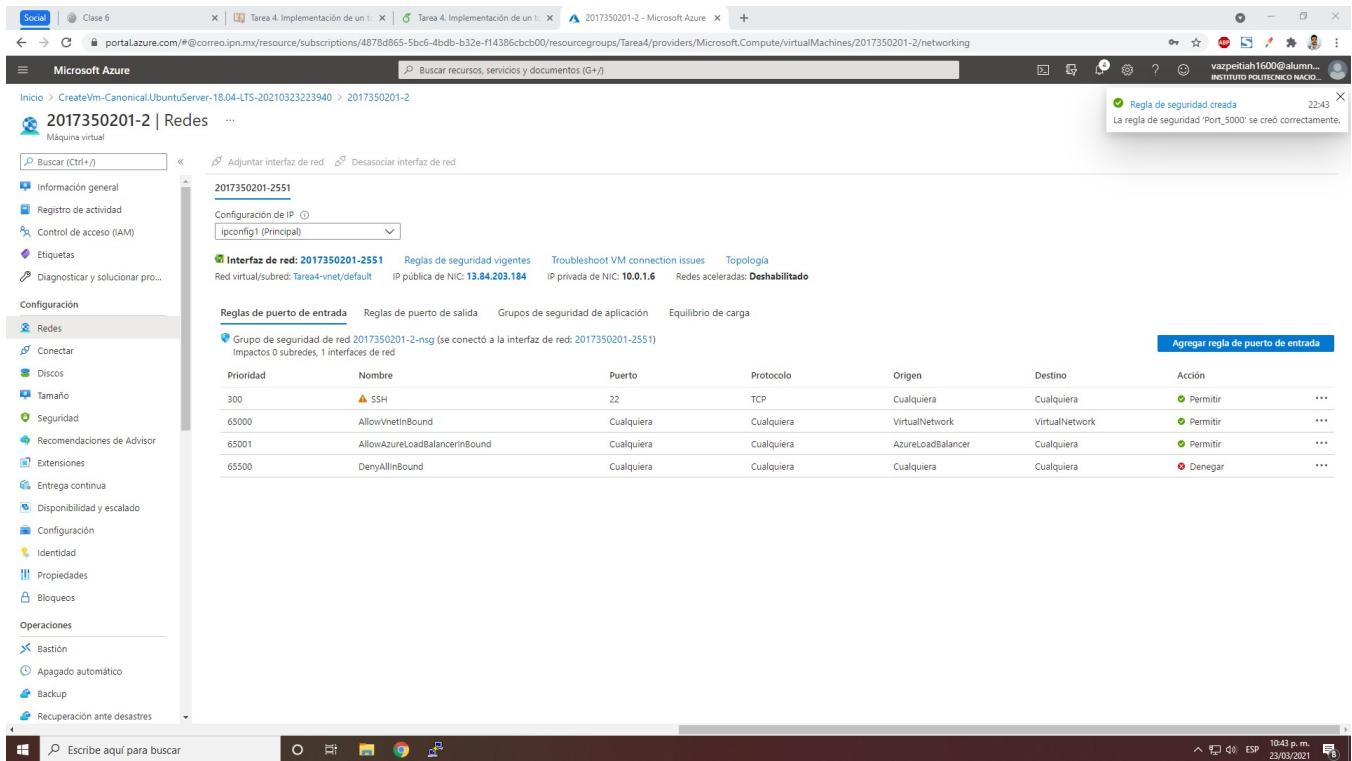


Figura 24: Nodo 2. Puerto 5000 abierto

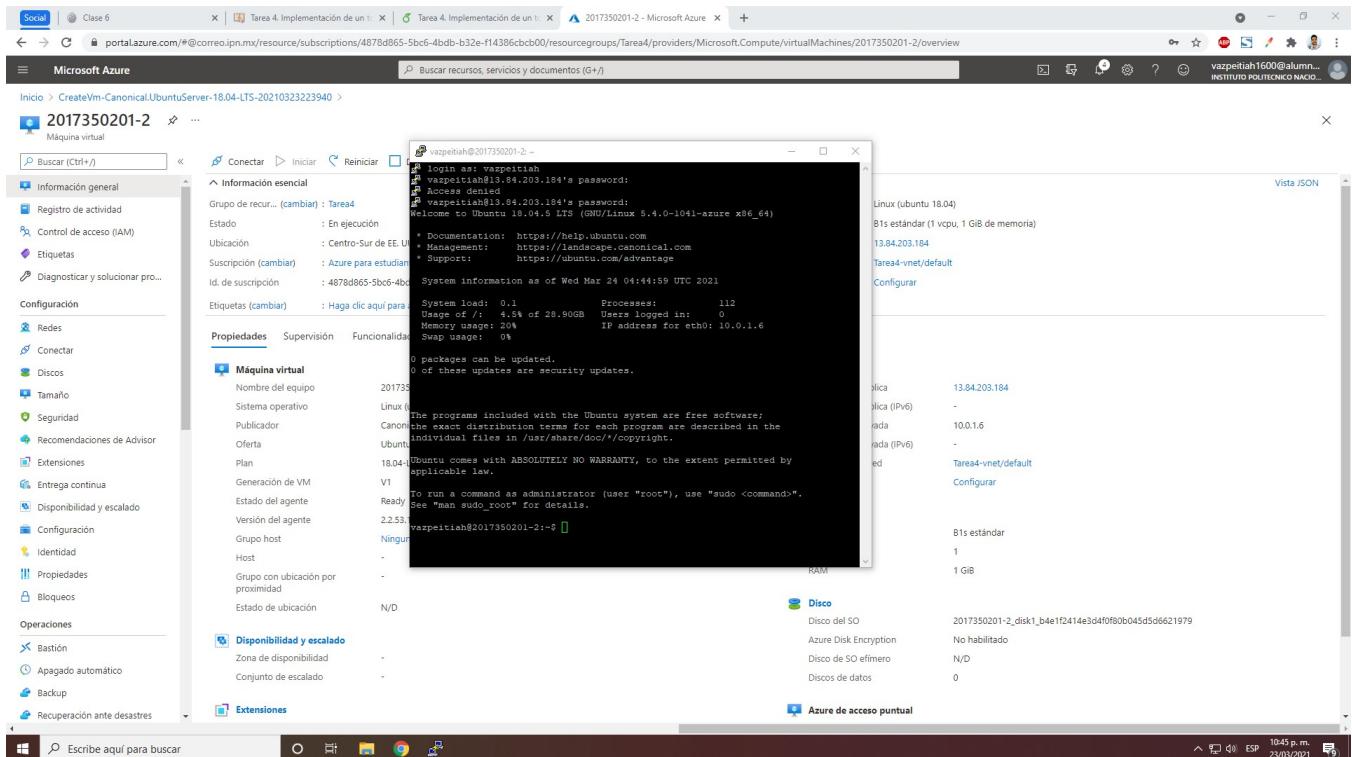


Figura 25: Nodo 2. Conectarse con la máquina virtual a través de ssh

2.3. Compilación y ejecución del programa

Previamente tenemos que tener nuestras tres máquinas virtuales configuradas, y debemos acceder a ellas a través de ssh. En mi caso, subí el código a un repositorio de github, y luego clone este repositorio en cada una de las máquinas virtuales. Una vez que el código fuente del programa se encuentre cargado en las máquinas, procedemos

a compilarlo y ejecutarlo. Debemos pasarle el número de nodo al programa y la dirección IP del siguiente nodo, como parámetros del programa:

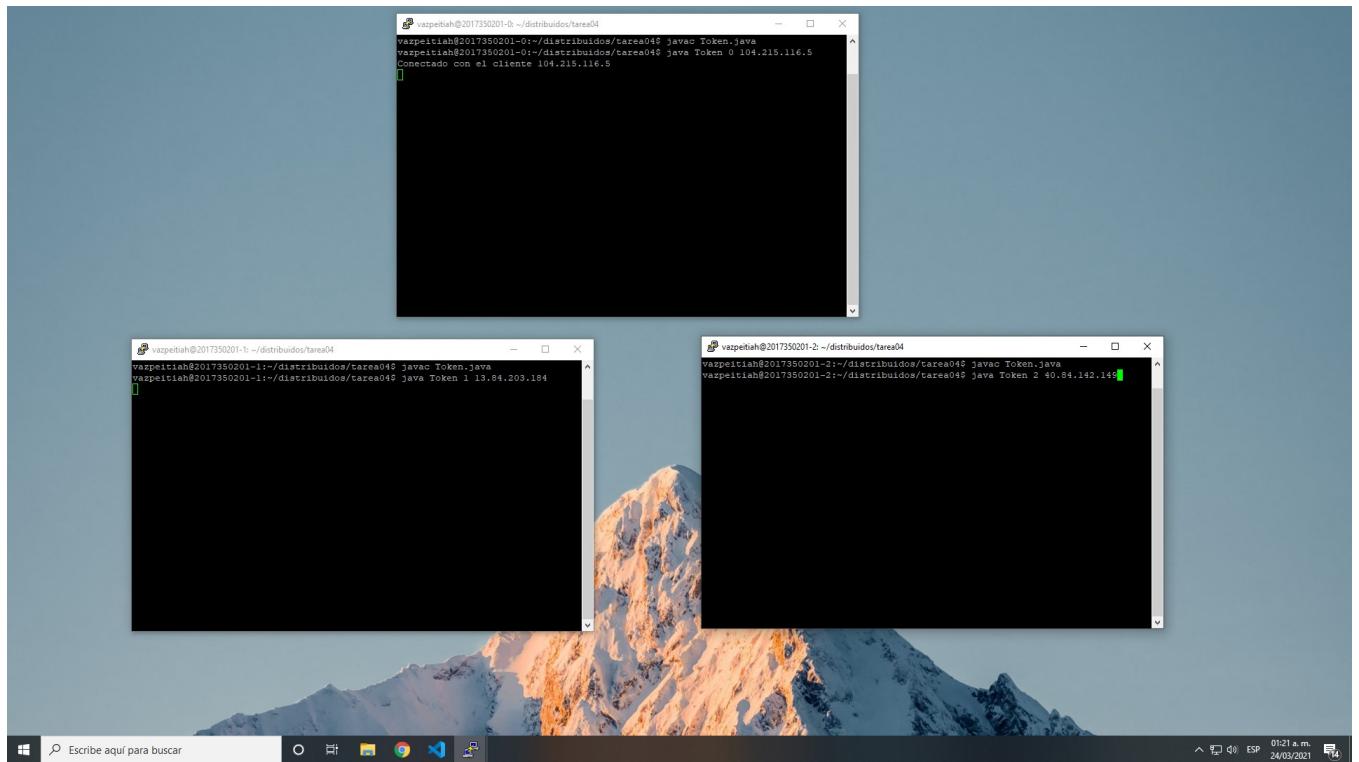


Figura 26: Compilando y ejecutando el programa en todos los nodos

Una vez que hayamos ejecutado los 3 nodos, estos empezaran a mostrar la salida.

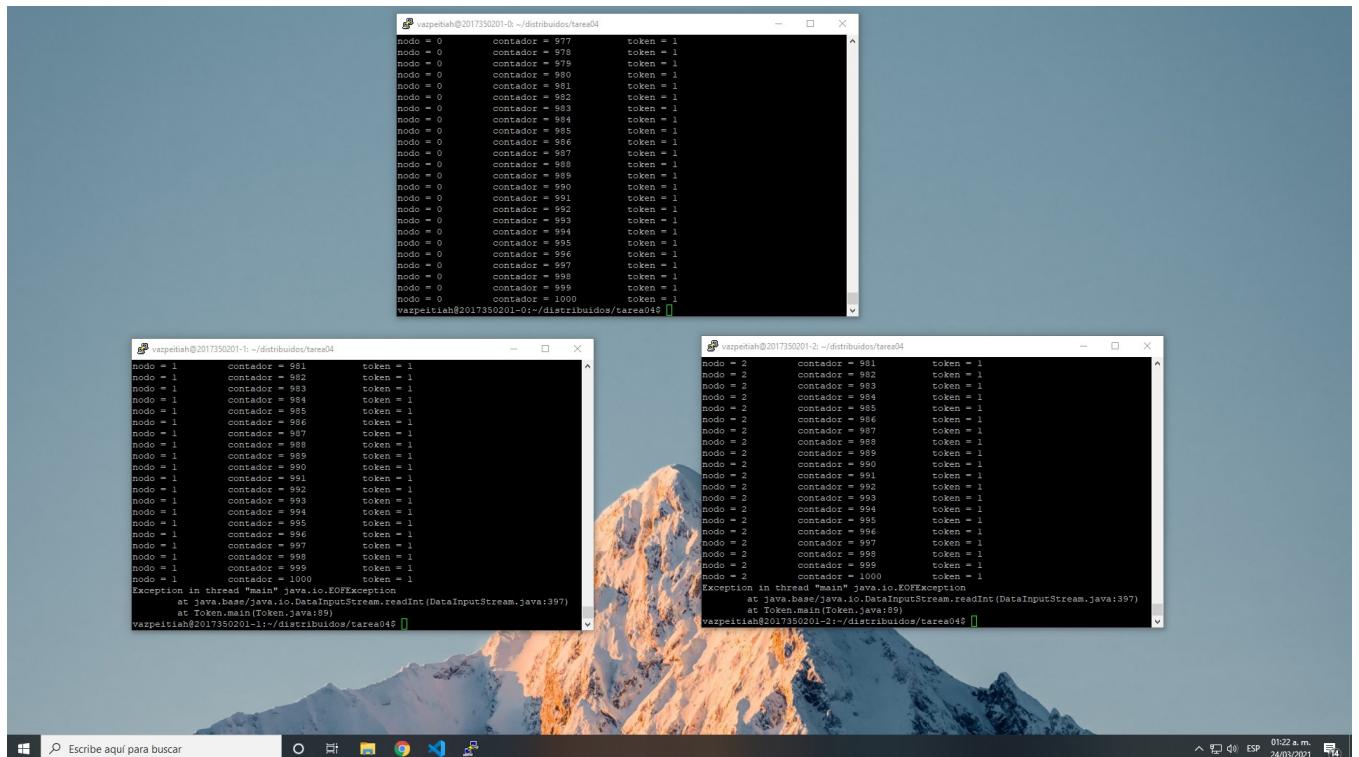


Figura 27: Compilando y ejecutando el programa en todos los nodos 2

3. Conclusiones

El poder usar máquinas virtuales desde Microsoft Azure me parece una excelente opción para nosotros los estudiantes. El tan solo configurarlas y acceder a ellas de manera remota, implica adquirir nuevos conocimientos en diversos temas, y esto me pareció muy bueno. El desarrollo del código de la práctica era muy sencillo, pero me sirvió para entender mejor el concepto de exclusión mutua. Para poder subir el código a las máquinas virtuales, use Github y luego clone el repositorio en cada una de las máquinas. Una vez cargado el código en las máquinas, instale el JDK de java para poder compilar el código fuente. Después ejecute cada uno de los nodos. Los nodos se quedaban en espera, hasta que se ejecutarán los 3. Los 3 muestran, como salida, el número de nodo, el valor del token = 1 y el contador; que cuenta las veces que se pasan el token. Al finalizar el nodo 0, el resto de nodos terminan de forma abrupta, se detienen en la parte en la que reciben el token a través del socket TCP.