Application deployment the hard way

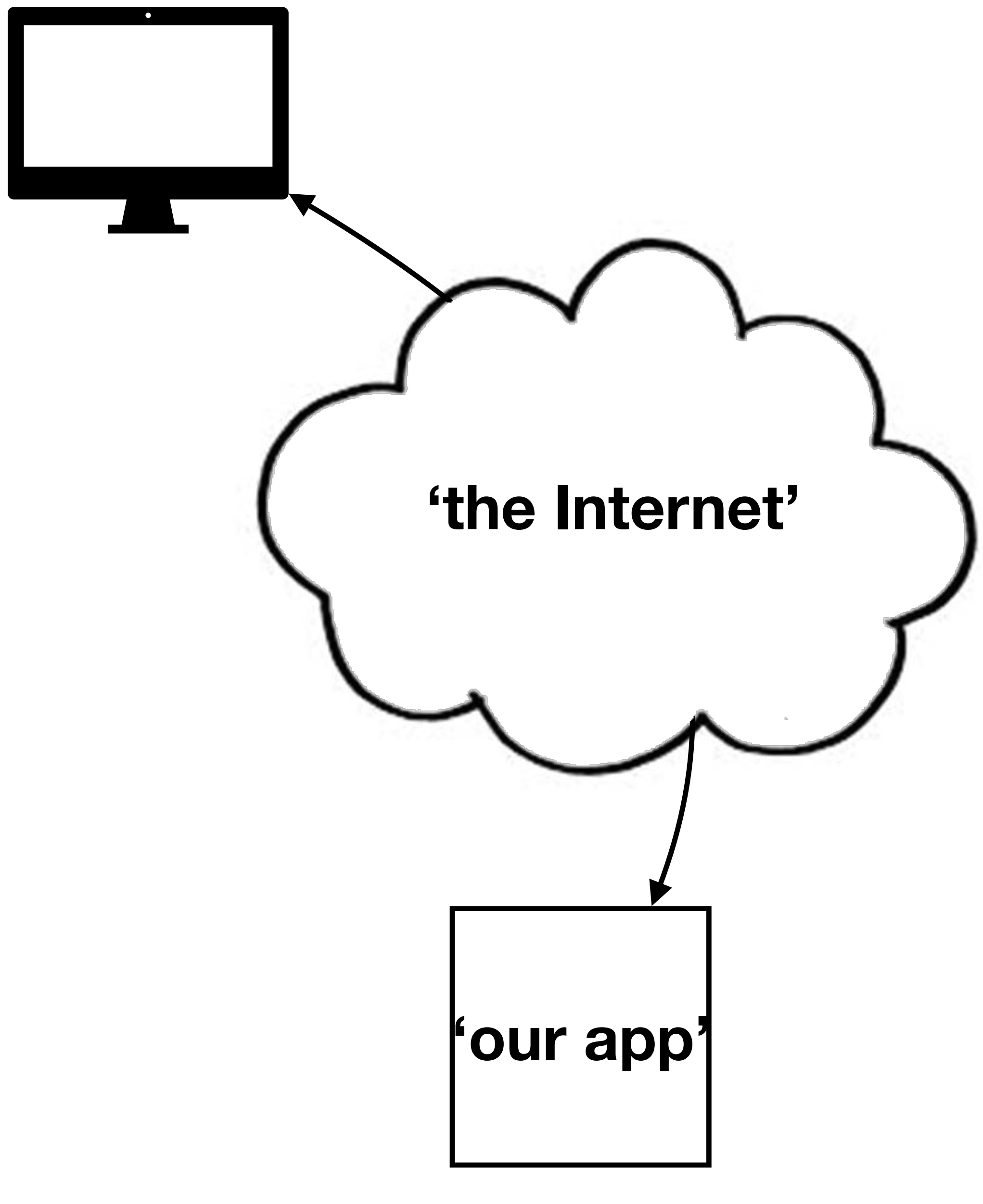
Slack:
https://tinyurl.com/uob-cloud-unit-slack

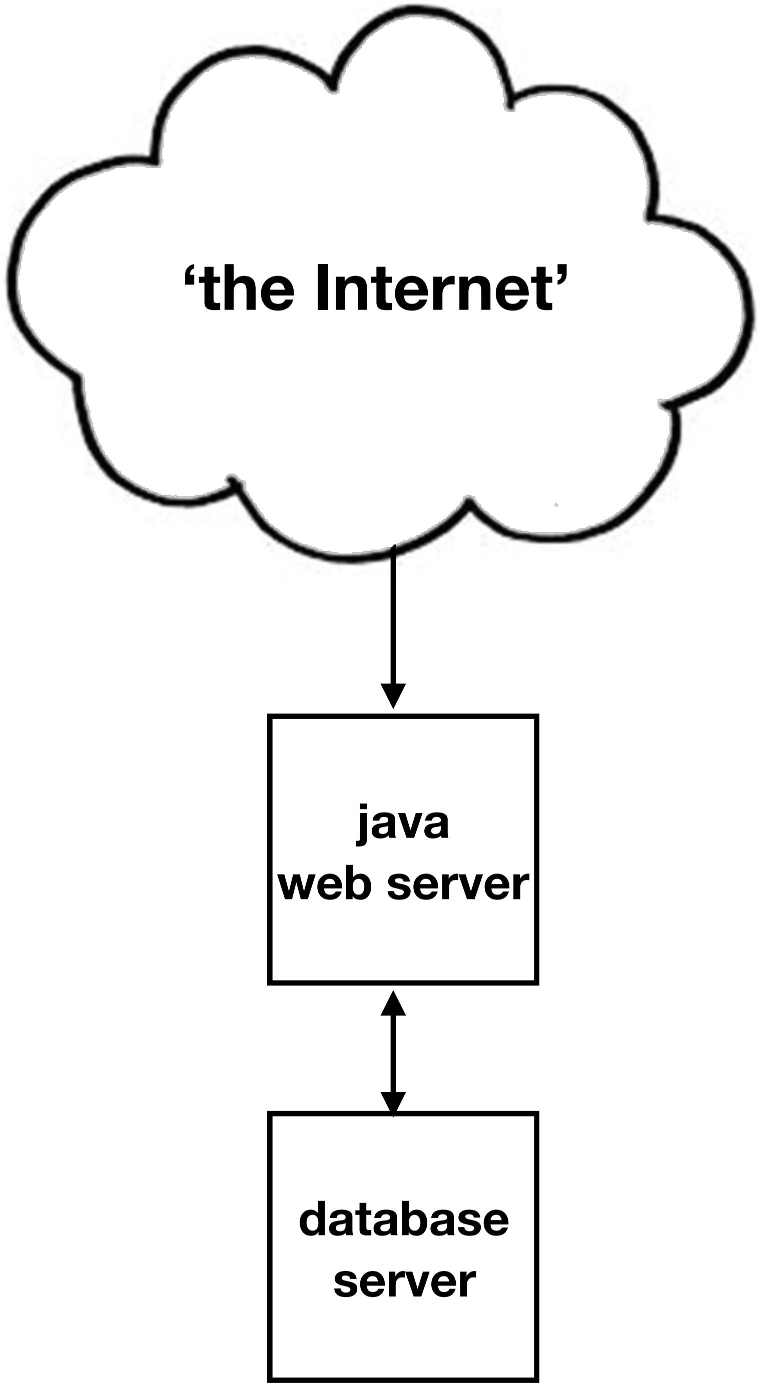
The Goal

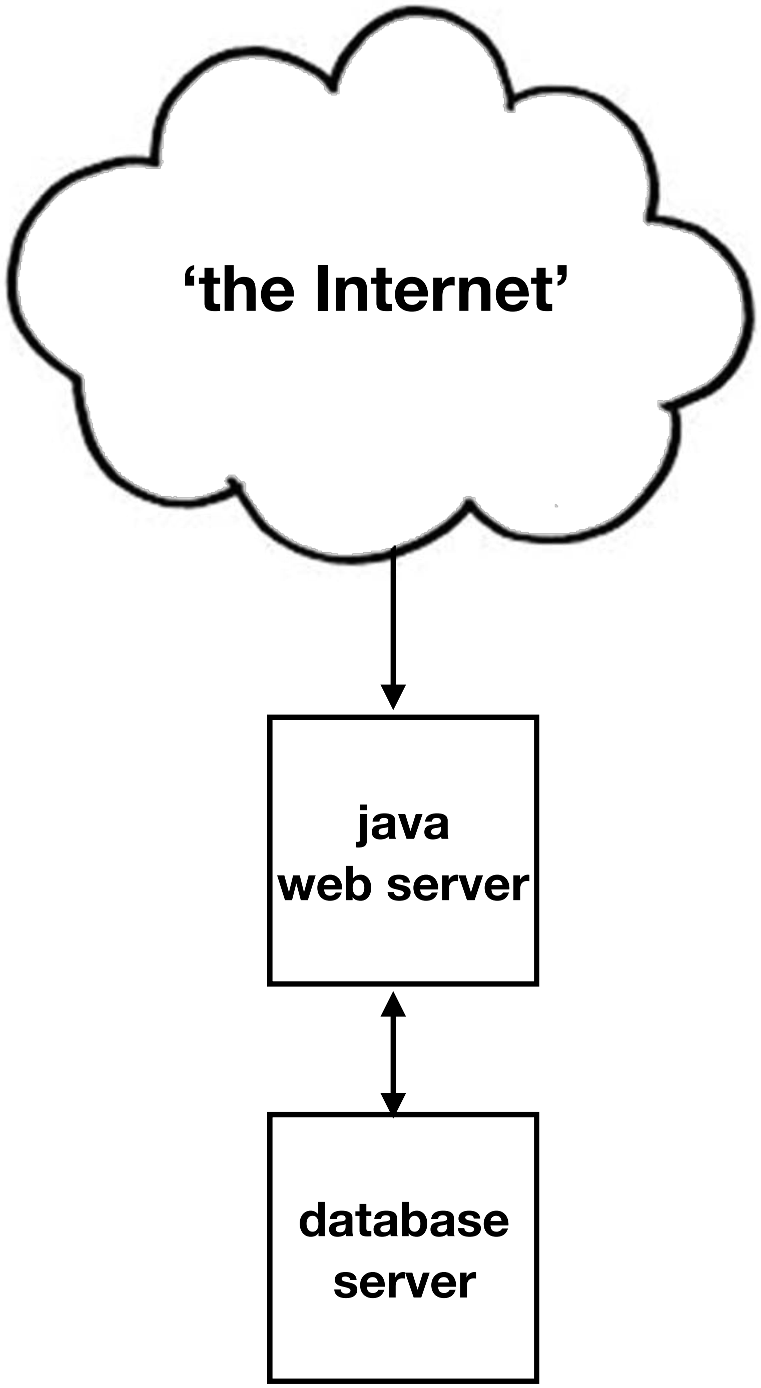
We’d like users to be able to talk to our application over the Internet

The Goal

We’d like users to be able to talk to our application over the Internet




‘the Internet’
java
web server
database
server



‘the Internet’

java web server

database server

Each of these will be hosted on a VM

“hosted on a VM”

- Each VM is a close analogue to a real host
    - It’s got a whole operating system stack, plus our software
- We can potentially stand up more of these than we could ever get our hands on physical hardware!
    - … great! Now I need to manage all these virtual hosts
- We’ll start by looking at doing this “manually”, with a very simple architecture

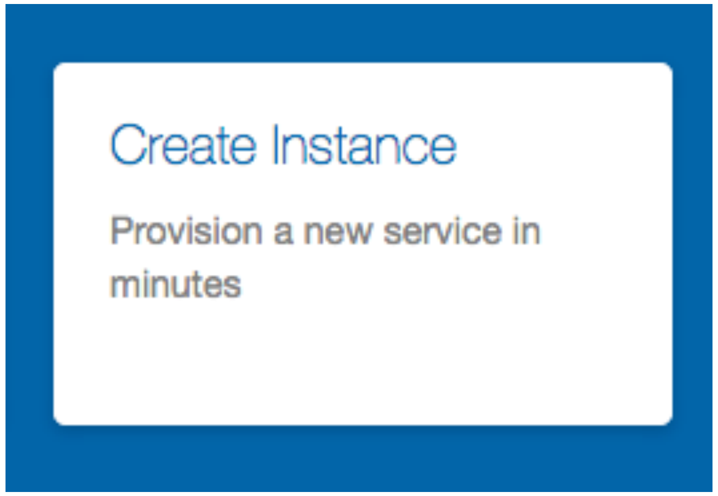
Remote management

- Via a command-line
- We’ll use *ssh* to talk to the hosts interactively
  - (Use *putty* or some other client on Windows; for MacOS and Linux desktops you should have an ssh client already installed)
- We boot hosts with an ssh public key installed
  - You supply this; it’s an alternative to a password

ssh key generation

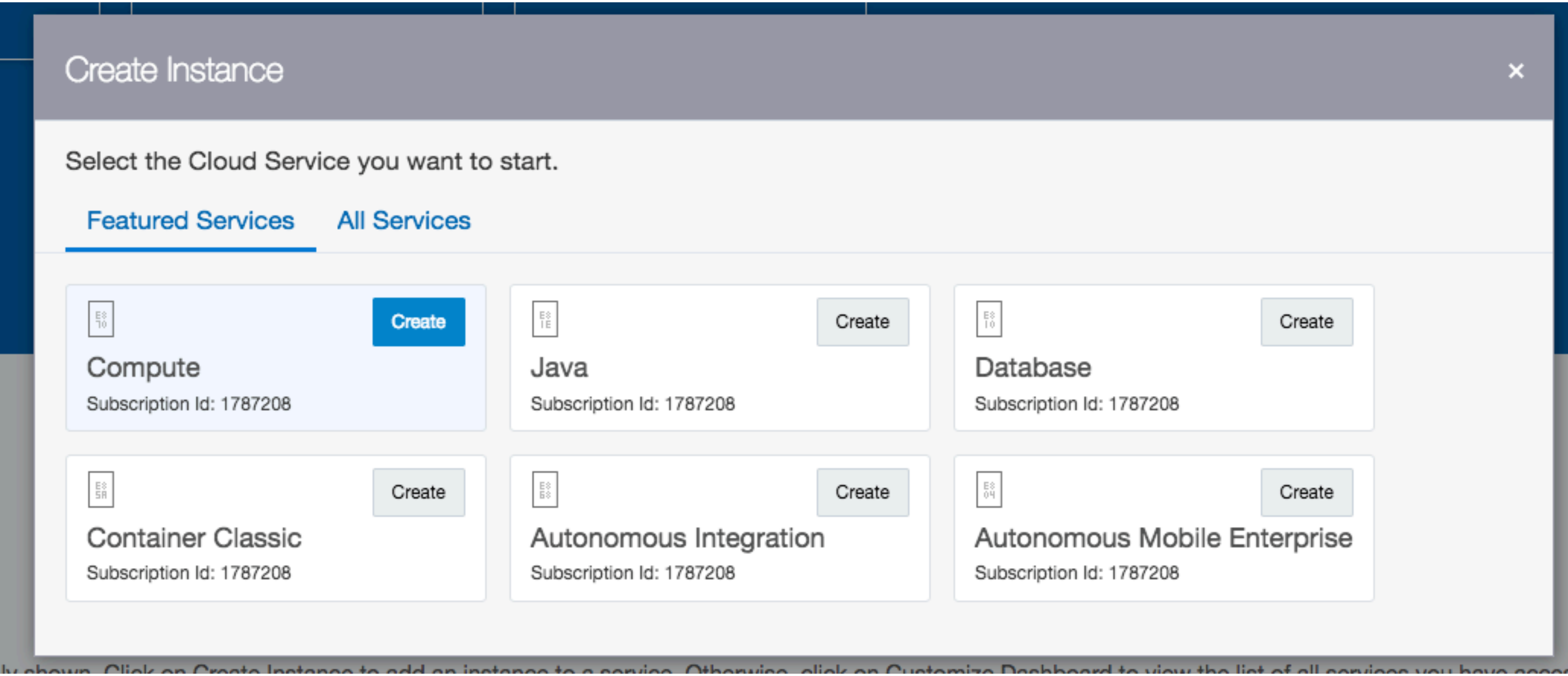
```
% mkdir ~/.ssh
% ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/jan/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/jan/.ssh/id_rsa.
Your public key has been saved in /Users/jan/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:FcVhmf24Lahrpcf56YT65djtKodK4kxTWOFFOo1uyIc jan@jan-Mac
The key's randomart image is:
+---[RSA 2048]----+
|          o+*=   |
|         . O+ .  |
|          B .  o |
|       . B .  . .|
|        E =  . o |
|         +  o.o .|
|        + .=.oo. |
|       + ++.**.o |
|        o.+=o=Boo|
+----[SHA256]-----+
% cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDM+UIk50jMoHOEPReFcO+hTEPe3mfq8ow1ObCF4CM29OjixwWH5UJr08+CbkS
Zgs11LgYPu5QiK17sETSaWW4ZXQC88j5KzsxrgApRb84a+q9gPgGE0nmLAb2ZjGP13dX5Pu41b6vsapglci5/lALFq/by5G6fzq
Qtrh0m3d0mr3hRu1aE1vY1K6igy3Mj8/tyZxcN4OJkFbV4wzavmdpPPgh0LXT41bWfQDzQRlSs/nLPGIuUOlNpSInfSSvNvSz8Z
tsWPQZtt1zuVMIhCwUdzF01urWw4ATkghk9GKNtze9ocGIrIcbNhSSQQiqkYnS8UdHUdfzr+MejiuefsMI1 jan@jan-Mac
```

- We’ll start with the web console



Create Instance

Provision a new service in minutes



Create Instance

Select the Cloud Service you want to start.

Featured Services | All Services

Compute
Subscription Id: 1787208

Java
Subscription Id: 1787208

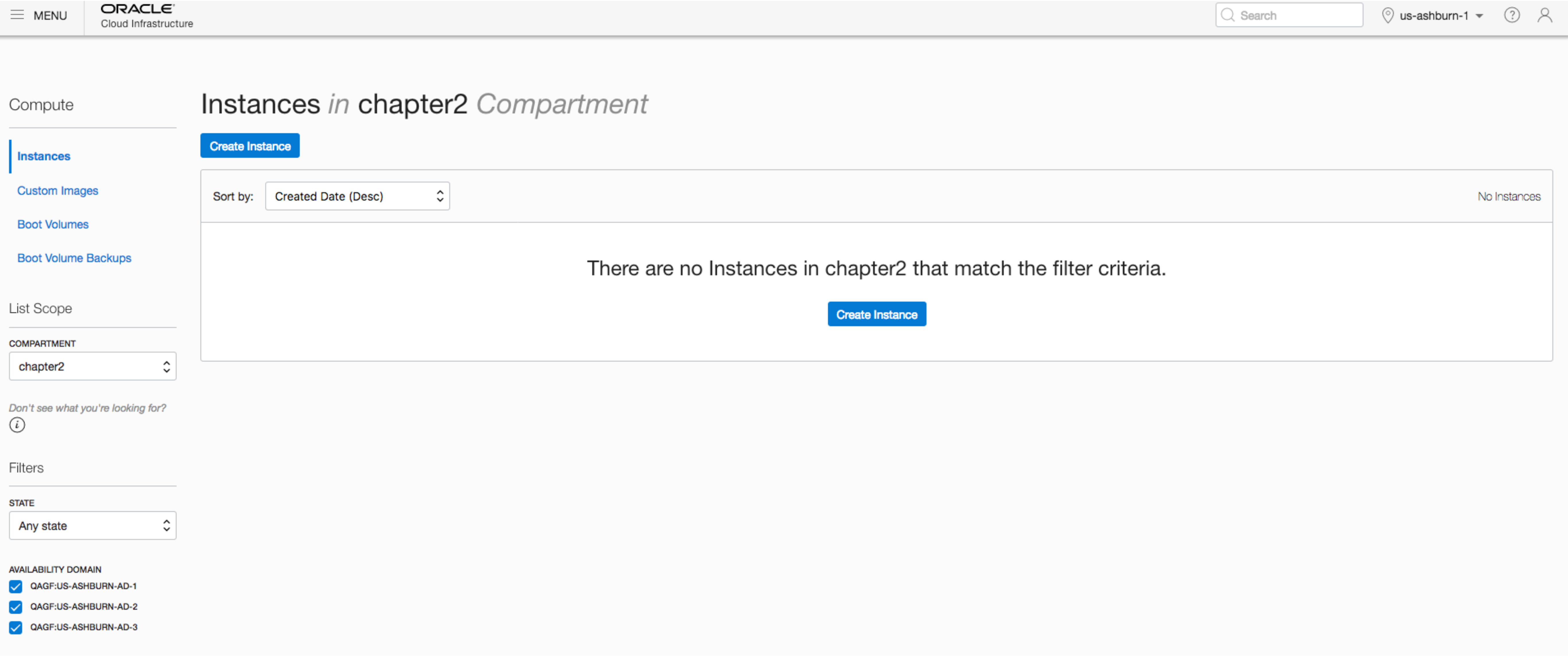
Database
Subscription Id: 1787208

Container Classic
Subscription Id: 1787208

Autonomous Integration
Subscription Id: 1787208

Autonomous Mobile Enterprise
Subscription Id: 1787208

ly shown. Click on Create Instance to add an instance to a service. Otherwise, click on Customize Dashboard to view the list of all services you have acce



Compute

List Scope

COMPARTMENT: chapter2

Don't see what you're looking for?

Filters

STATE: Any state

AVAILABILITY DOMAIN

☑ QAGF:US-ASHBURN-AD-1

☑ QAGF:US-ASHBURN-AD-2

☑ QAGF:US-ASHBURN-AD-3

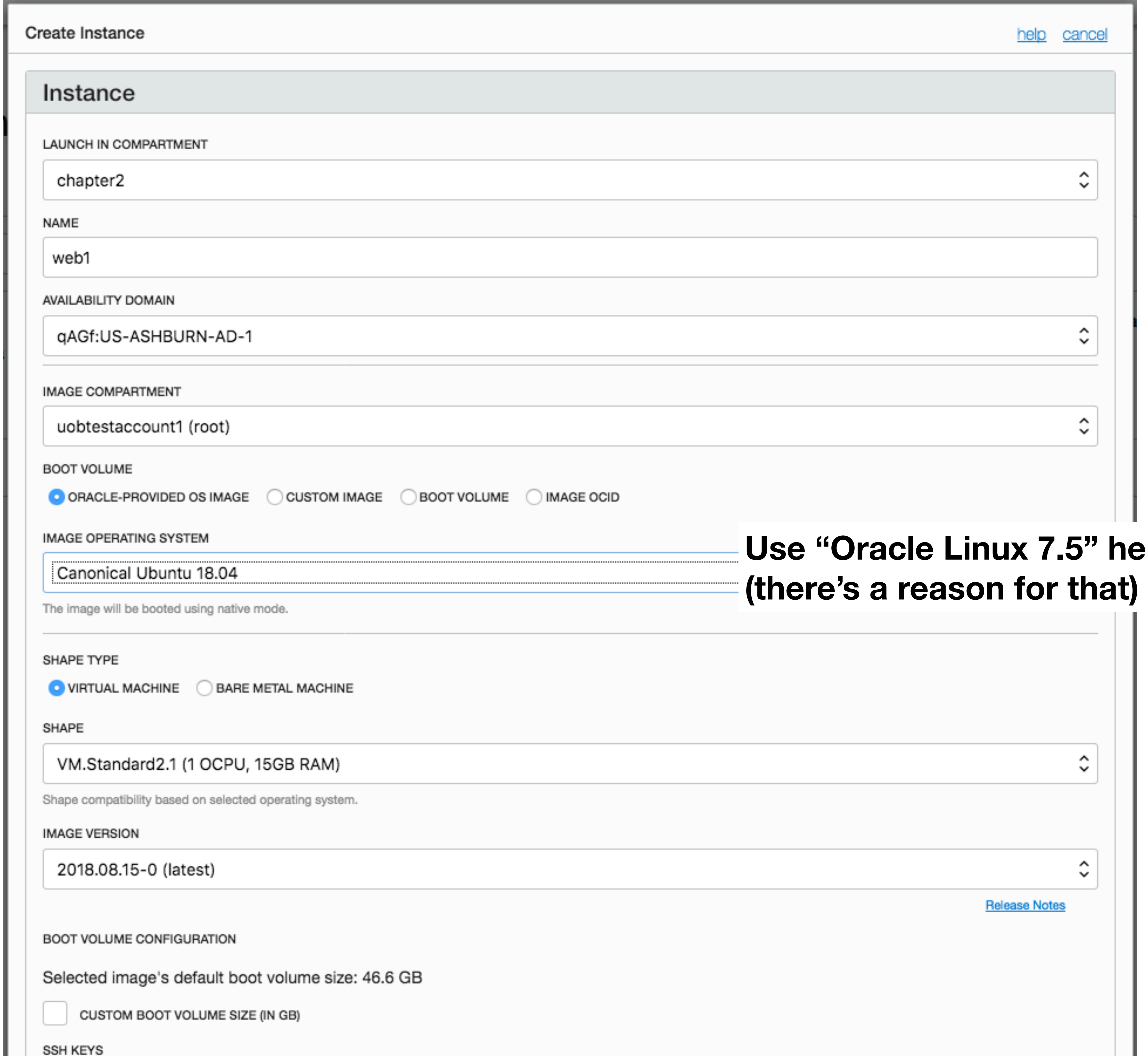
Instances *in* chapter2 *Compartment*

Sort by: Created Date (Desc)

No Instances

There are no Instances in chapter2 that match the filter criteria.

- We’ll begin with a VM to host the Java web server
- We’ll call it “web1”
  - We might want more of these later
- We’ll auto-create the network components at the same time



Create Instance

Instance

LAUNCH IN COMPARTMENT

chapter2

NAME

web1

AVAILABILITY DOMAIN

qAGf:US-ASHBURN-AD-1

IMAGE COMPARTMENT

uobtestaccount1 (root)

BOOT VOLUME

● ORACLE-PROVIDED OS IMAGE ○ CUSTOM IMAGE ○ BOOT VOLUME ○ IMAGE OCID

IMAGE OPERATING SYSTEM

Canonical Ubuntu 18.04

The image will be booted using native mode.

SHAPE TYPE

● VIRTUAL MACHINE ○ BARE METAL MACHINE

SHAPE

VM.Standard2.1 (1 OCPU, 15GB RAM)

Shape compatibility based on selected operating system.

IMAGE VERSION

2018.08.15-0 (latest)

BOOT VOLUME CONFIGURATION

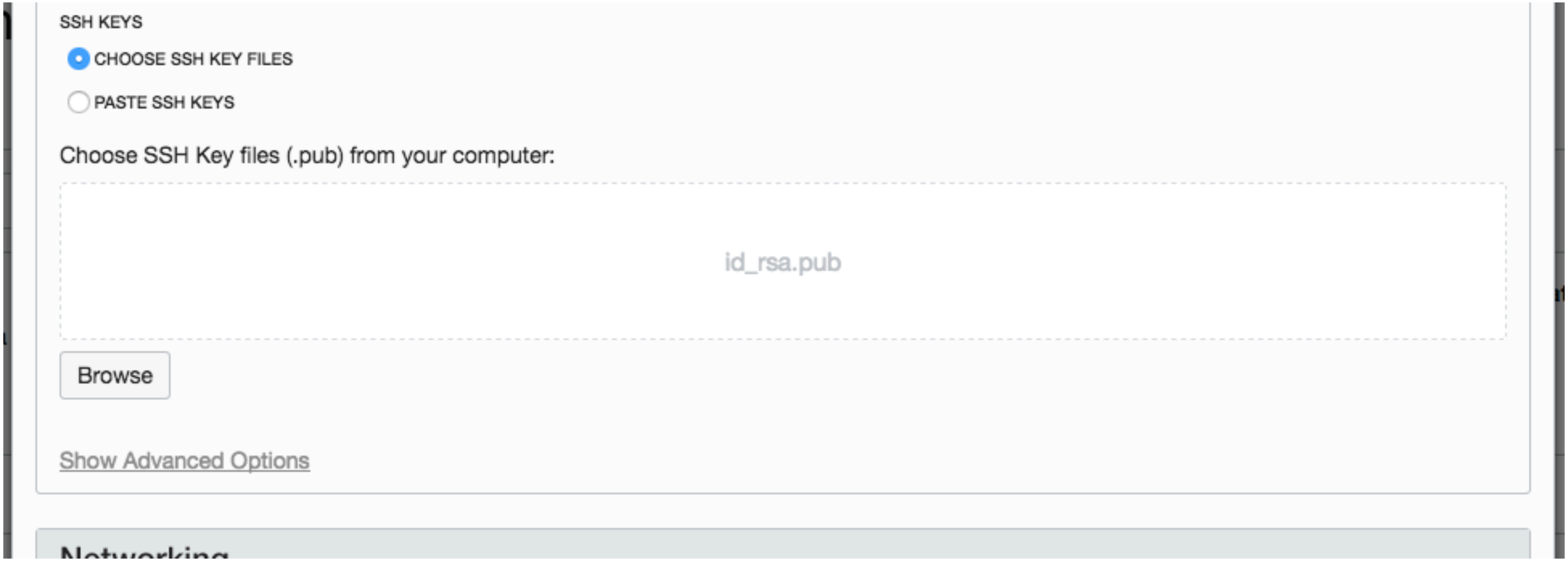
Selected image's default boot volume size: 46.6 GB

□ CUSTOM BOOT VOLUME SIZE (IN GB)

SSH KEYS

Use “Oracle Linux 7.5” here (there’s a reason for that)

Find the public half of your ssh key
(This content is passed to the VM on its first boot)



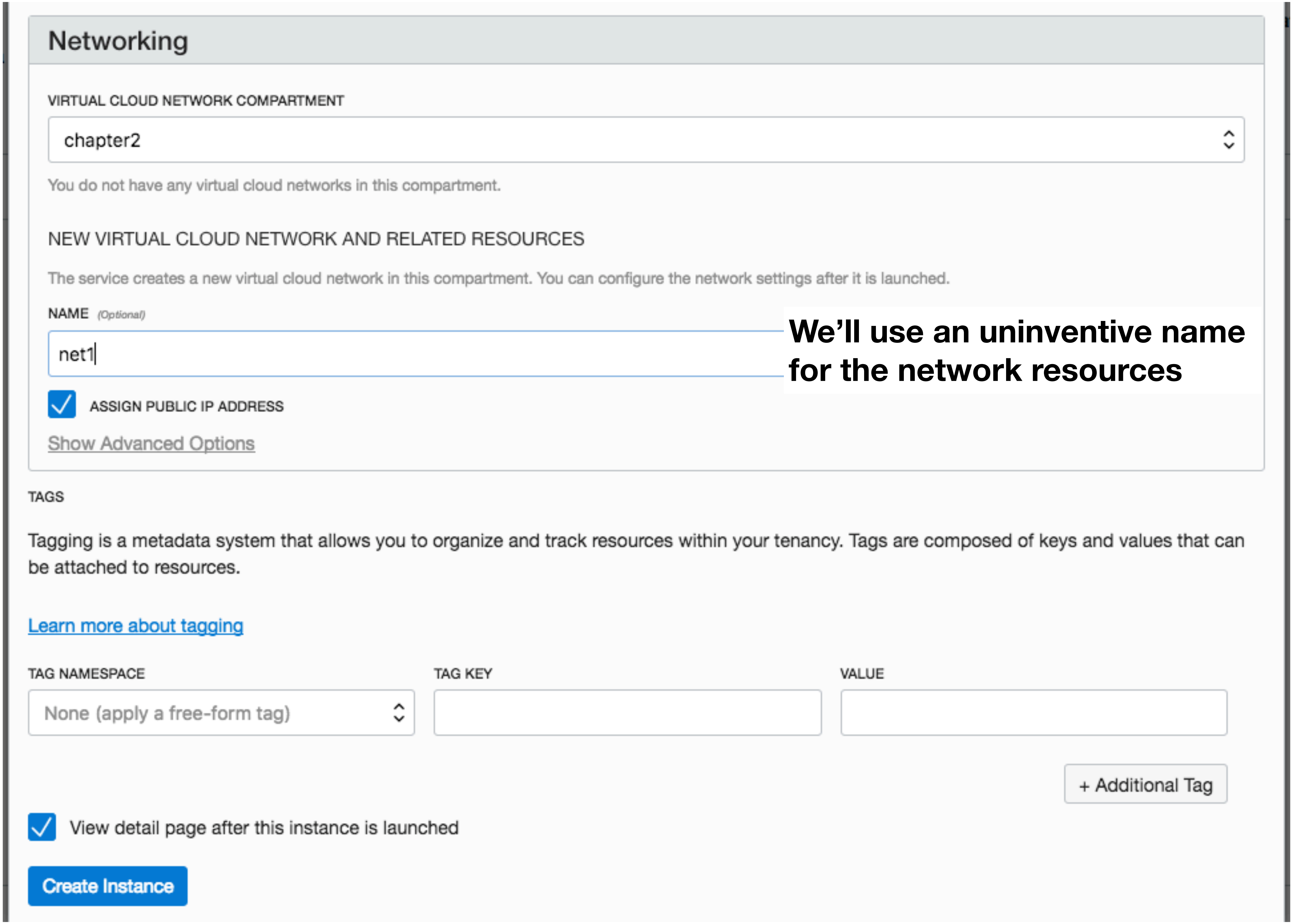
SSH KEYS

● CHOOSE SSH KEY FILES

○ PASTE SSH KEYS

Choose SSH Key files (.pub) from your computer:

id_rsa.pub



Networking

VIRTUAL CLOUD NETWORK COMPARTMENT

chapter2

You do not have any virtual cloud networks in this compartment.

NEW VIRTUAL CLOUD NETWORK AND RELATED RESOURCES

The service creates a new virtual cloud network in this compartment. You can configure the network settings after it is launched.

NAME *(Optional)*

net1

☑ ASSIGN PUBLIC IP ADDRESS

Show Advanced Options

TAGS

Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

Learn more about tagging

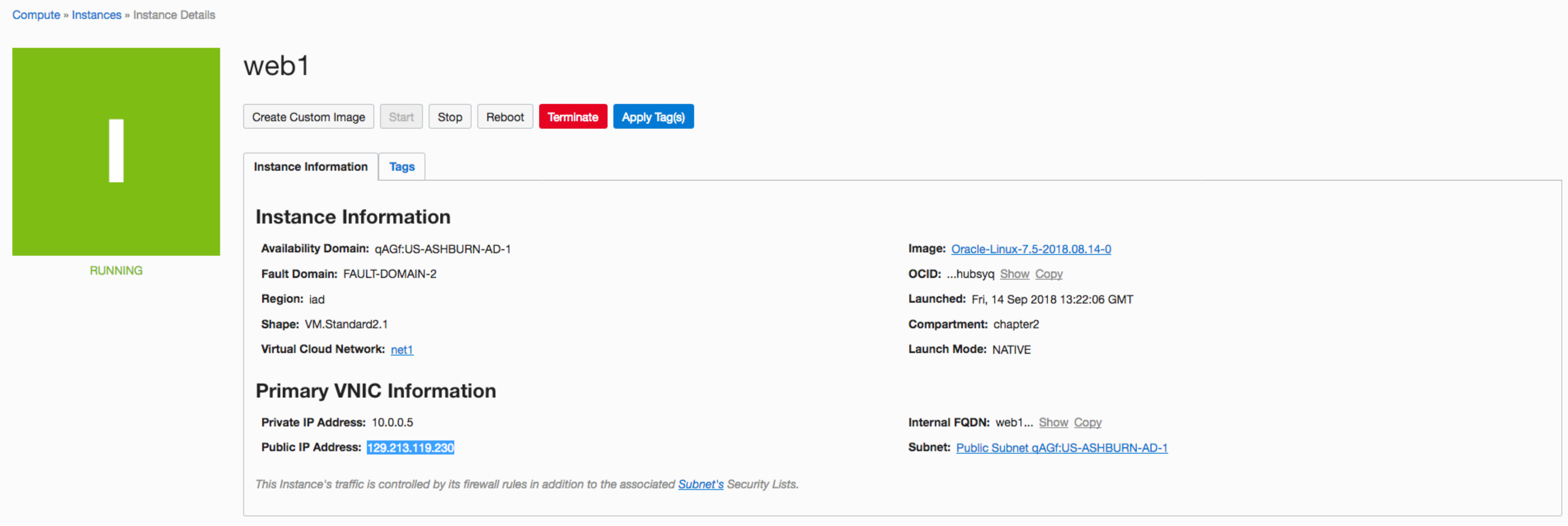
TAG NAMESPACE	TAG KEY	VALUE
None (apply a free-form tag)		

+ Additional Tag

☑ View detail page after this instance is launched

Create Instance

We’ll use an uninventive name for the network resources



web1

Instance Information

Availability Domain: qAGf:US-ASHBURN-AD-1

Fault Domain: FAULT-DOMAIN-2

Region: iad

Shape: VM.Standard2.1

Virtual Cloud Network: net1

Image: Oracle-Linux-7.5-2018.08.14-0

OCID: ...hubsyq Show Copy

Launched: Fri, 14 Sep 2018 13:22:06 GMT

Compartment: chapter2

Launch Mode: NATIVE

Primary VNIC Information

Private IP Address: 10.0.0.5

Public IP Address: 129.213.119.230

Internal FQDN: web1... Show Copy

Subnet: Public Subnet qAGf:US-ASHBURN-AD-1

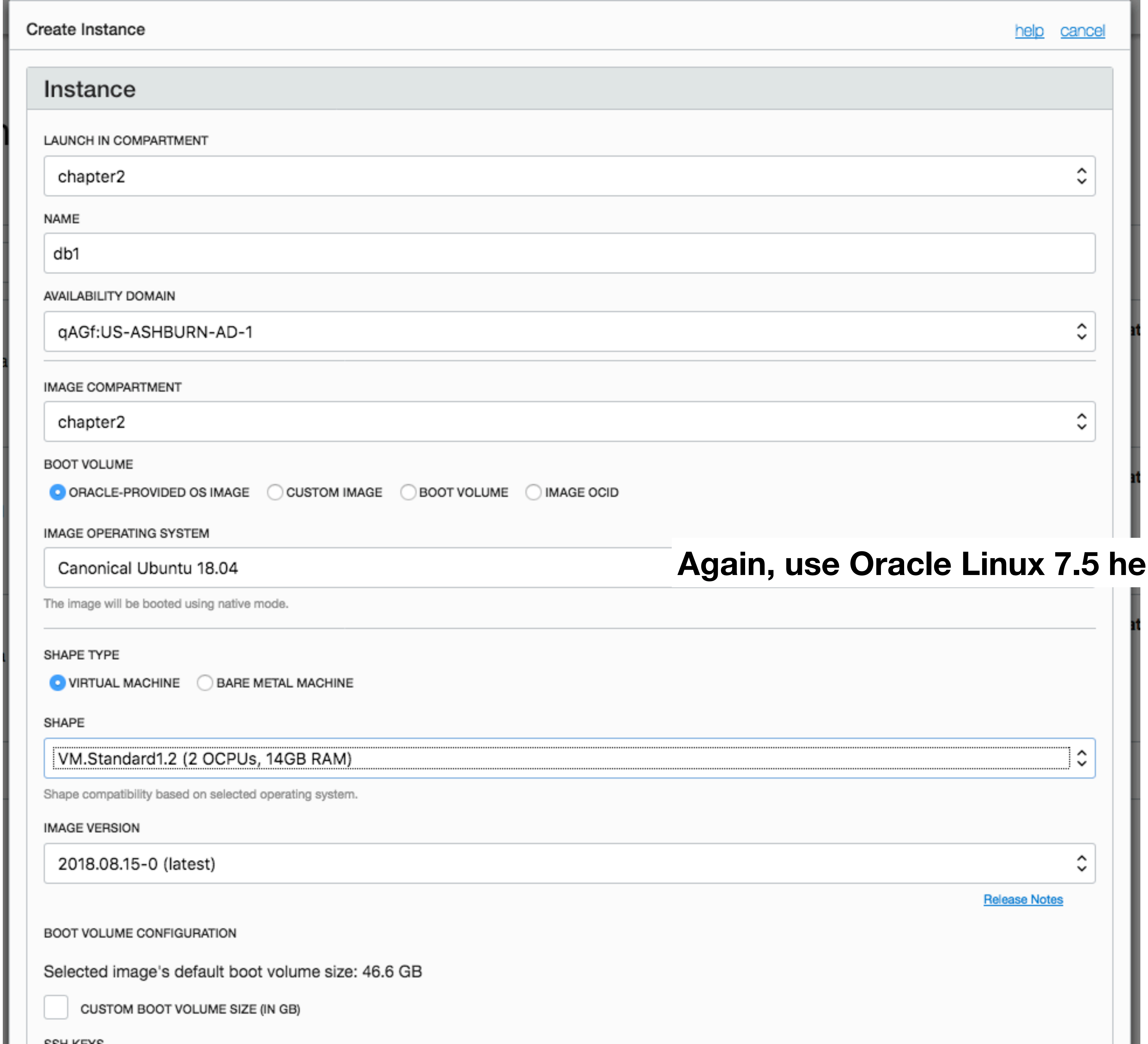
This Instance's traffic is controlled by its firewall rules in addition to the associated Subnet's Security Lists.

You should see this once the instance is booted.
Things to note:

- **there are two IP addresses listed!**
- **we'll use the *public* one to talk to it over the internet**

Booting the second instance

- We will call it “db1”
- We’ll use the same network resources created for the first instance
  - (We’ll look at those in a minute)
- Use “create instance” again…



Create Instance

Instance

LAUNCH IN COMPARTMENT: chapter2

NAME: db1

AVAILABILITY DOMAIN: qAGf:US-ASHBURN-AD-1

IMAGE COMPARTMENT: chapter2

BOOT VOLUME: ● ORACLE-PROVIDED OS IMAGE ○ CUSTOM IMAGE ○ BOOT VOLUME ○ IMAGE OCID

IMAGE OPERATING SYSTEM: Canonical Ubuntu 18.04

The image will be booted using native mode.

SHAPE TYPE: ● VIRTUAL MACHINE ○ BARE METAL MACHINE

SHAPE: VM.Standard1.2 (2 OCPUs, 14GB RAM)

Shape compatibility based on selected operating system.

IMAGE VERSION: 2018.08.15-0 (latest)

BOOT VOLUME CONFIGURATION

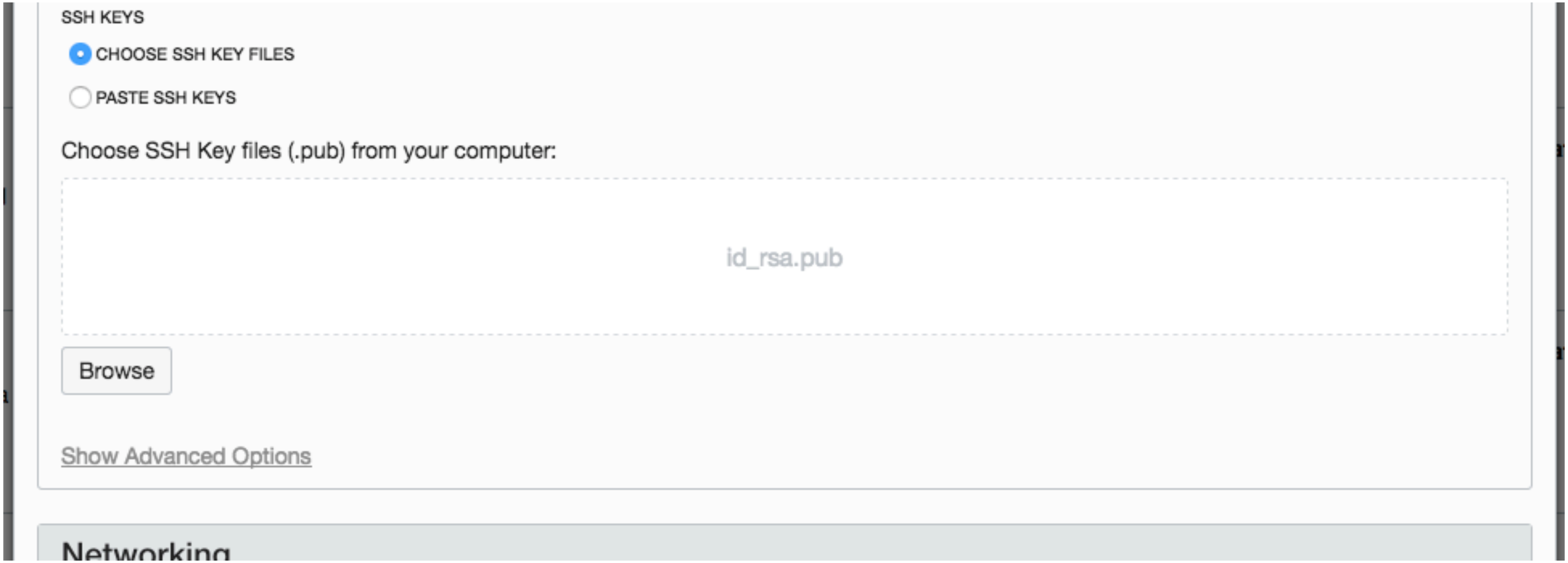
Selected image's default boot volume size: 46.6 GB

□ CUSTOM BOOT VOLUME SIZE (IN GB)

SSH KEYS

Again, use Oracle Linux 7.5 here!

We’ll pick the same SSH key



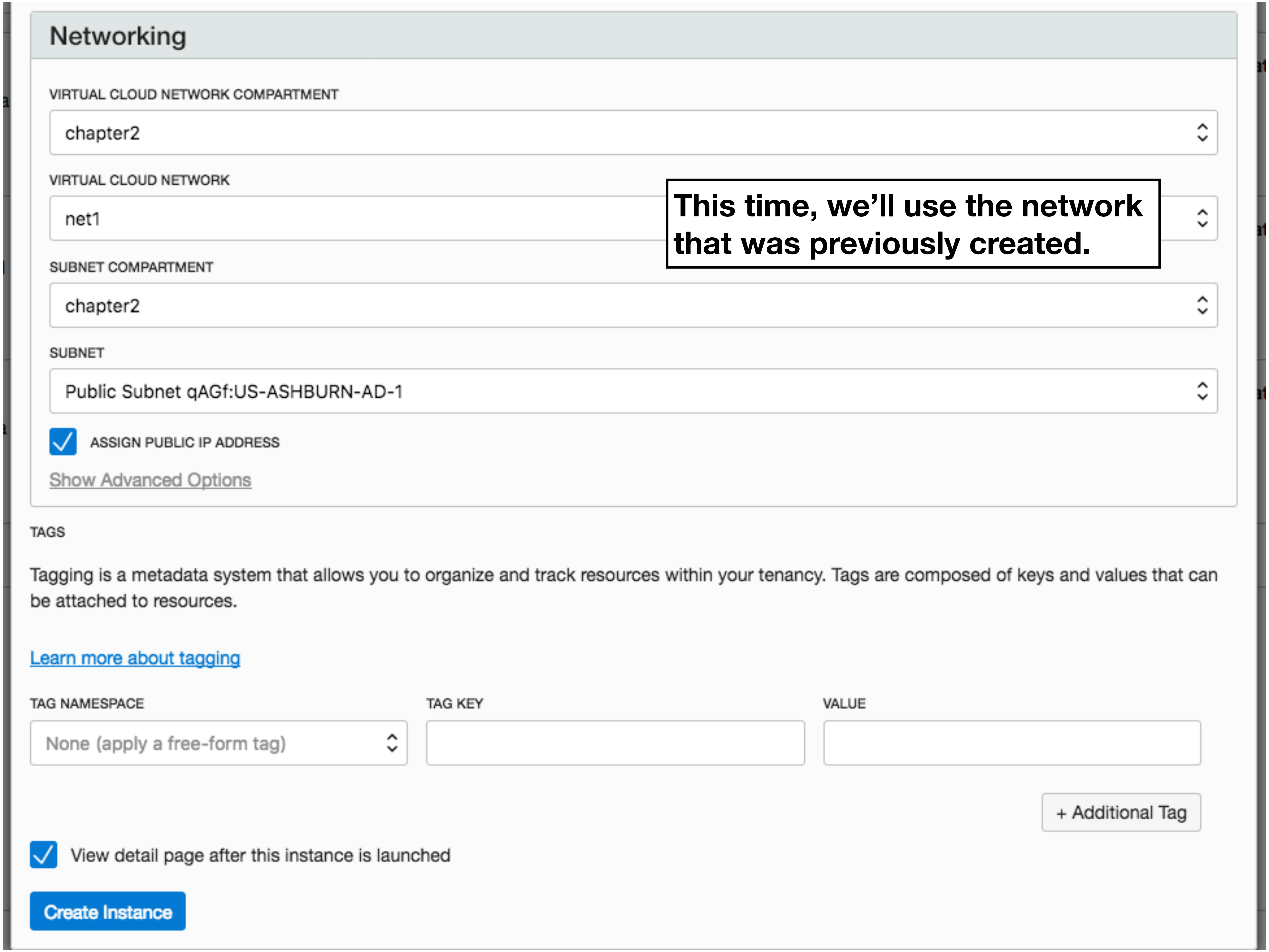
SSH KEYS

● CHOOSE SSH KEY FILES

○ PASTE SSH KEYS

Choose SSH Key files (.pub) from your computer:

id_rsa.pub



Networking

VIRTUAL CLOUD NETWORK COMPARTMENT

chapter2

VIRTUAL CLOUD NETWORK

net1

This time, we'll use the network that was previously created.

SUBNET COMPARTMENT

chapter2

SUBNET

Public Subnet qAGf:US-ASHBURN-AD-1

☑ ASSIGN PUBLIC IP ADDRESS

Show Advanced Options

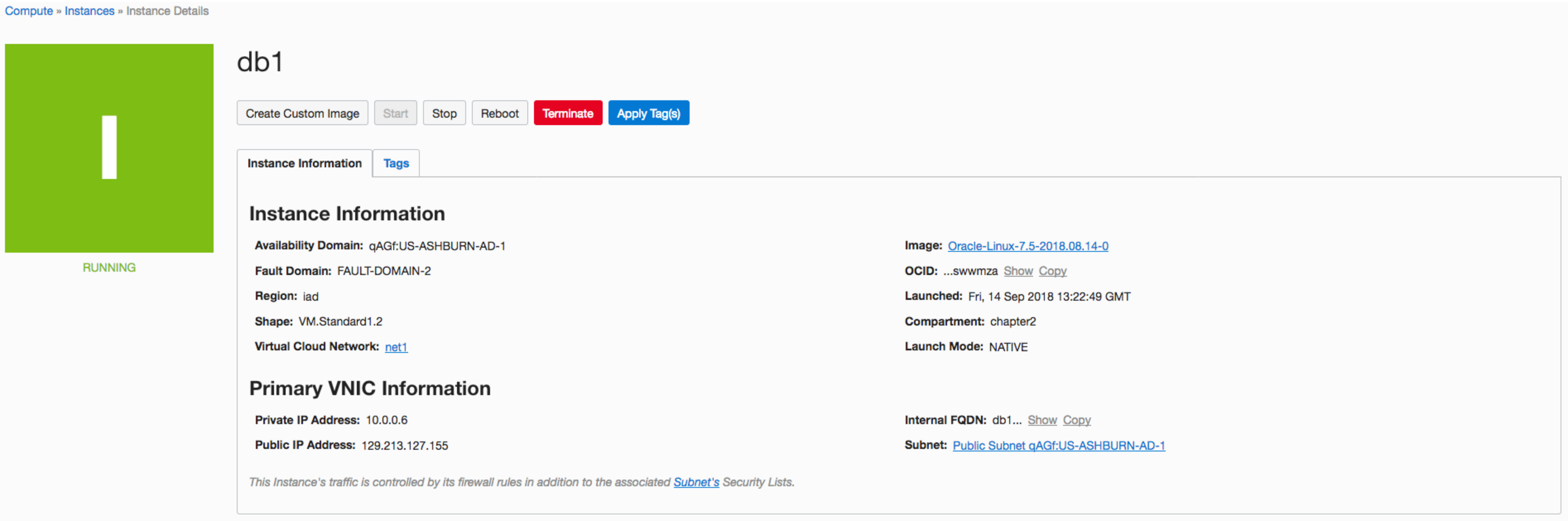
TAGS

Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

Learn more about tagging

TAG NAMESPACE	TAG KEY	VALUE
None (apply a free-form tag)		

☑ View detail page after this instance is launched



db1

Instance Information

Availability Domain: qAGf:US-ASHBURN-AD-1
Fault Domain: FAULT-DOMAIN-2
Region: iad
Shape: VM.Standard1.2
Virtual Cloud Network: net1

Image: Oracle-Linux-7.5-2018.08.14-0
OCID: ...swwmza Show Copy
Launched: Fri, 14 Sep 2018 13:22:49 GMT
Compartment: chapter2
Launch Mode: NATIVE

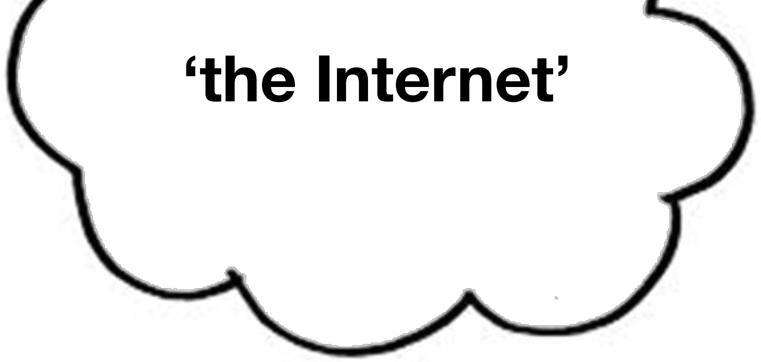
Primary VNIC Information

Private IP Address: 10.0.0.6
Public IP Address: 129.213.127.155

Internal FQDN: db1... Show Copy
Subnet: Public Subnet qAGf:US-ASHBURN-AD-1

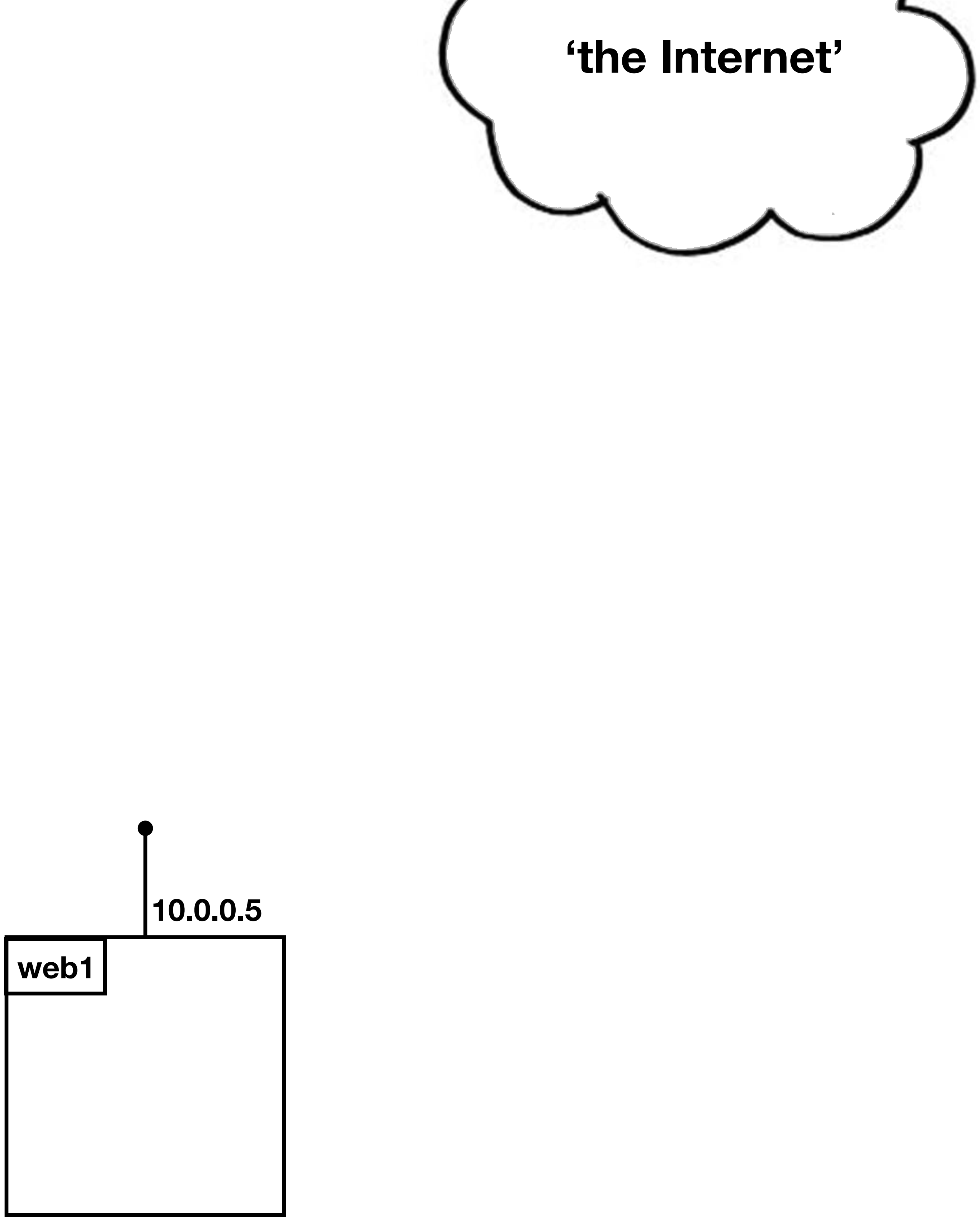
This Instance's traffic is controlled by its firewall rules in addition to the associated Subnet's Security Lists.

The result should look very similar.
Again, we have both an internal and external IP address.



‘the Internet’

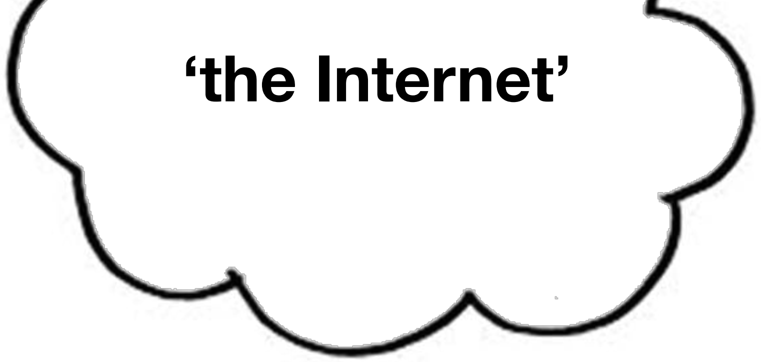
What does this look like?



‘the Internet’

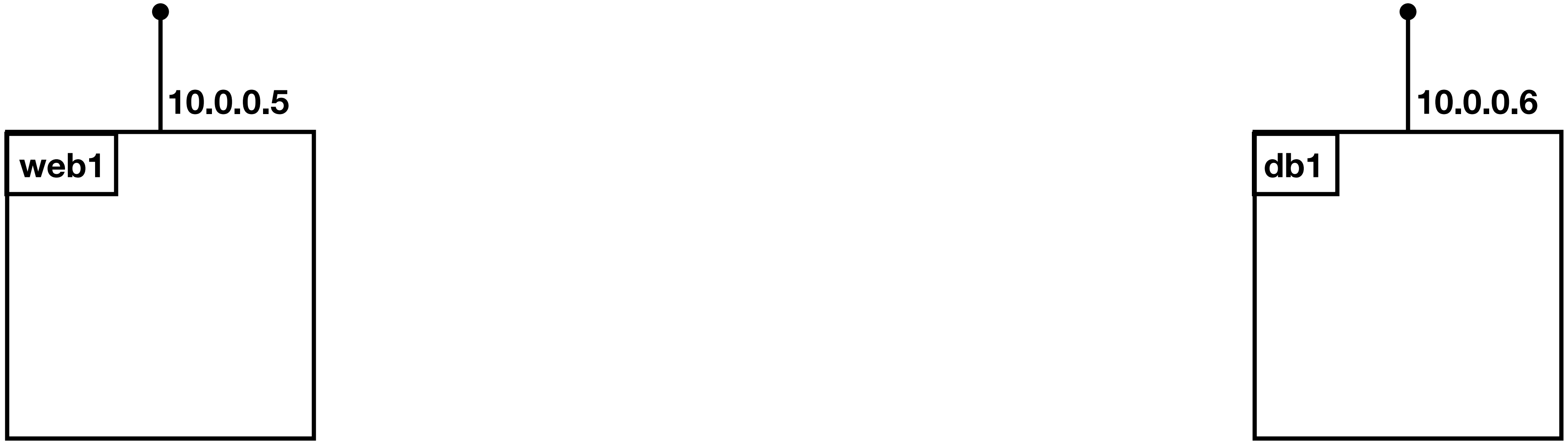
10.0.0.5

web1



‘the Internet’

We’ve two new VMs

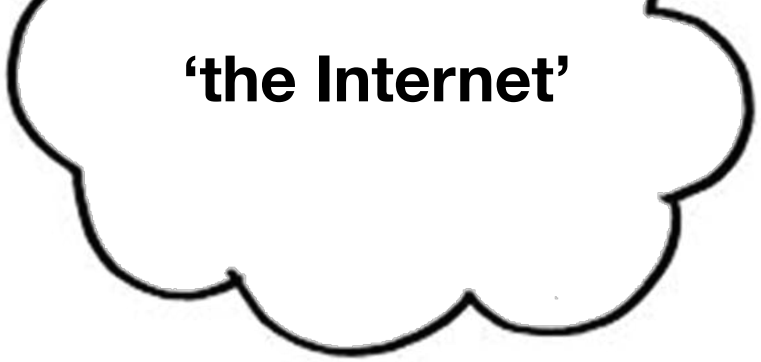


10.0.0.5

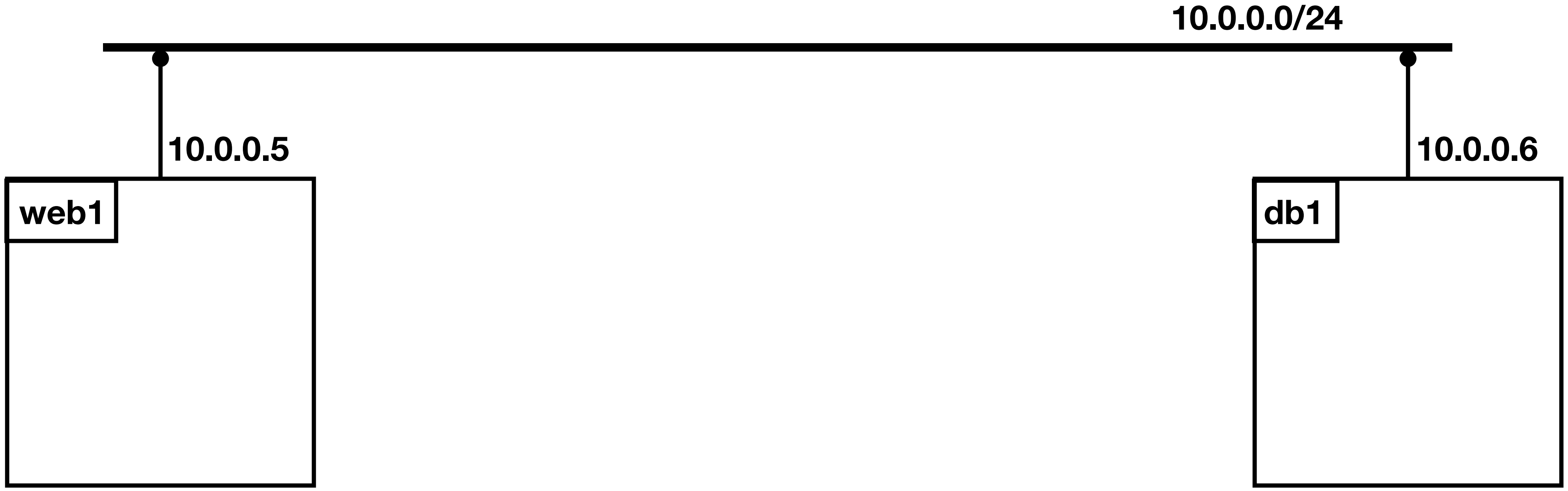
web1

10.0.0.6

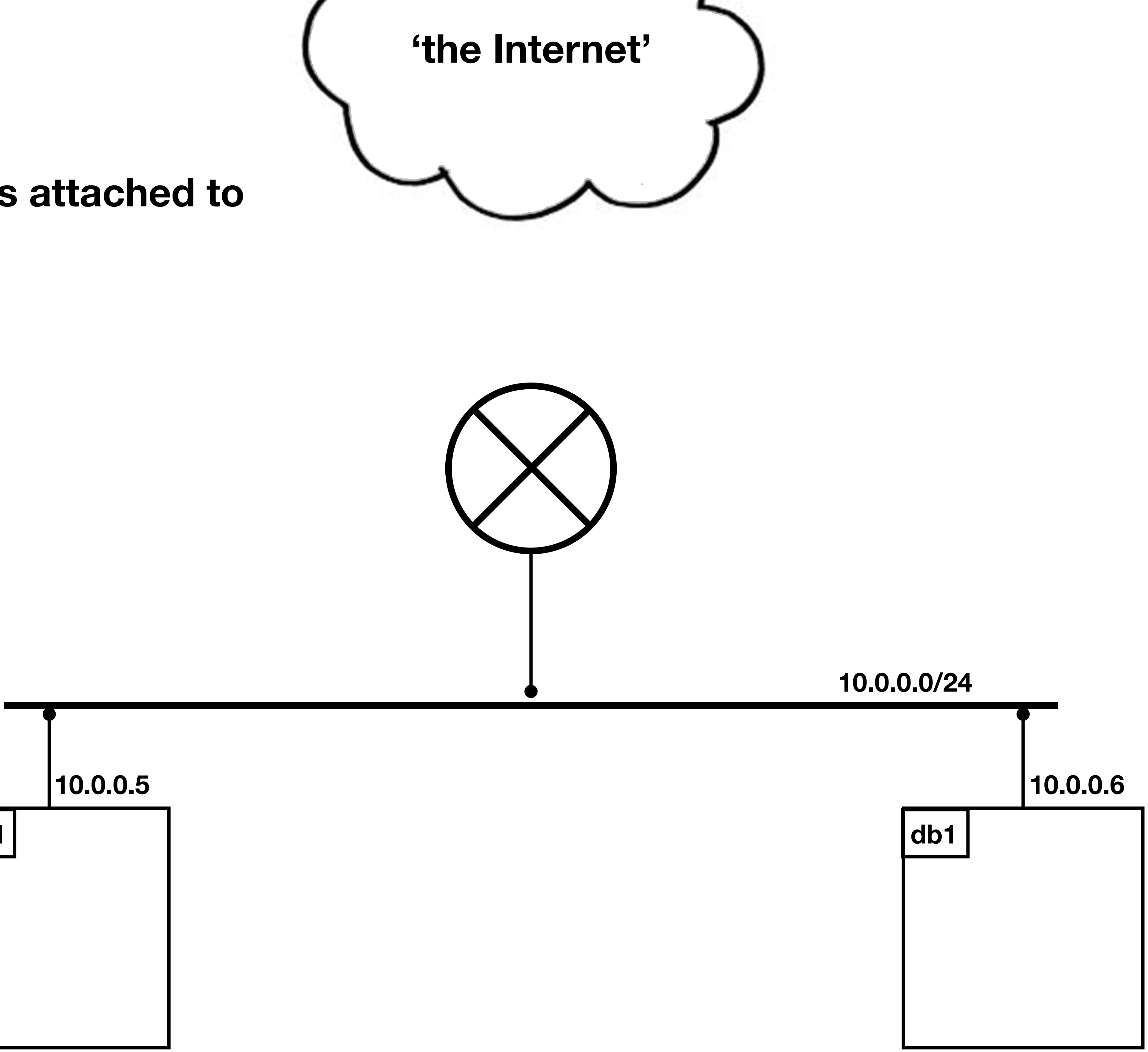
db1



They sit on the same *subnet*



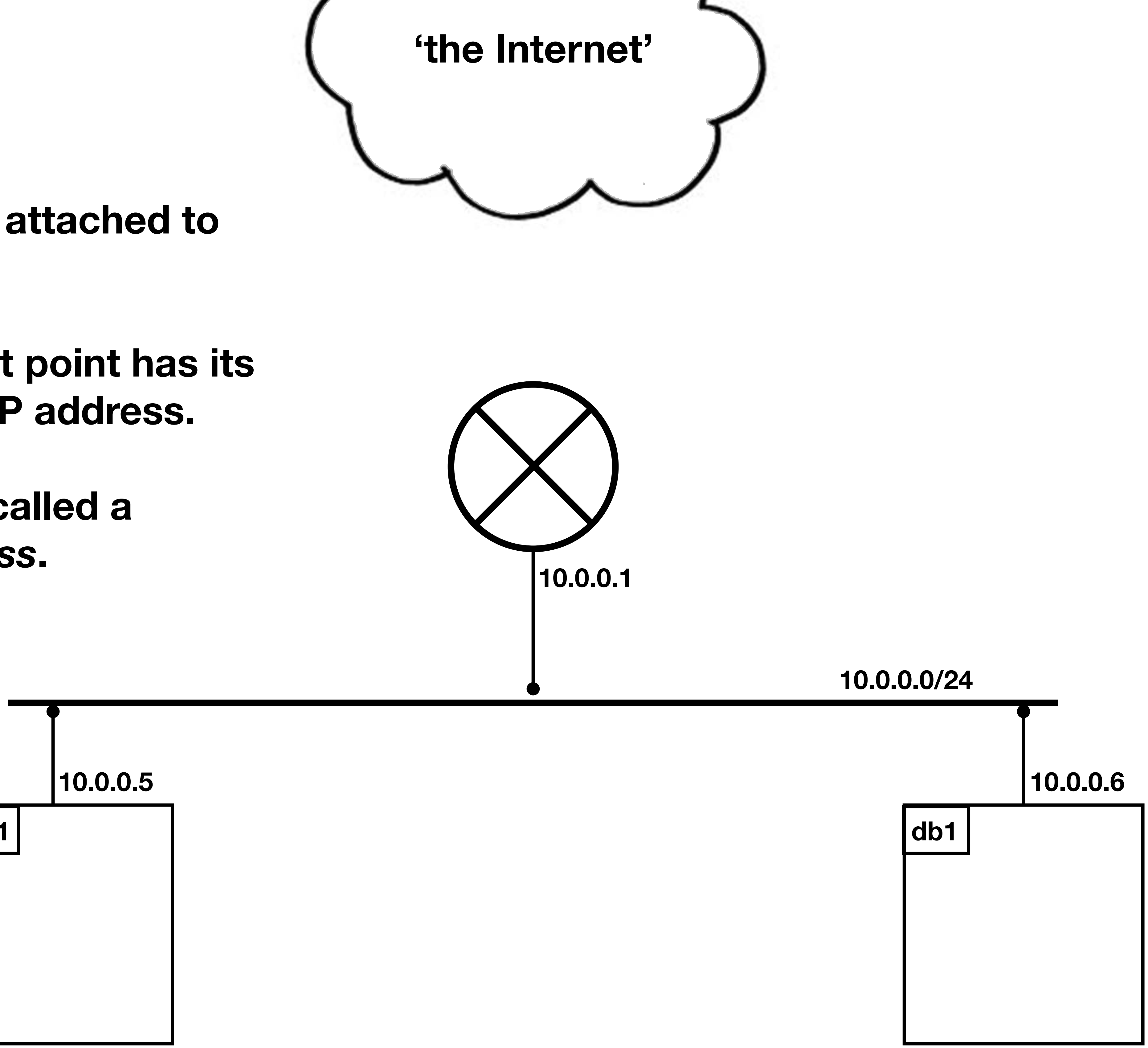
Each subnet is attached to a *router*.



Each subnet is attached to a *router*.

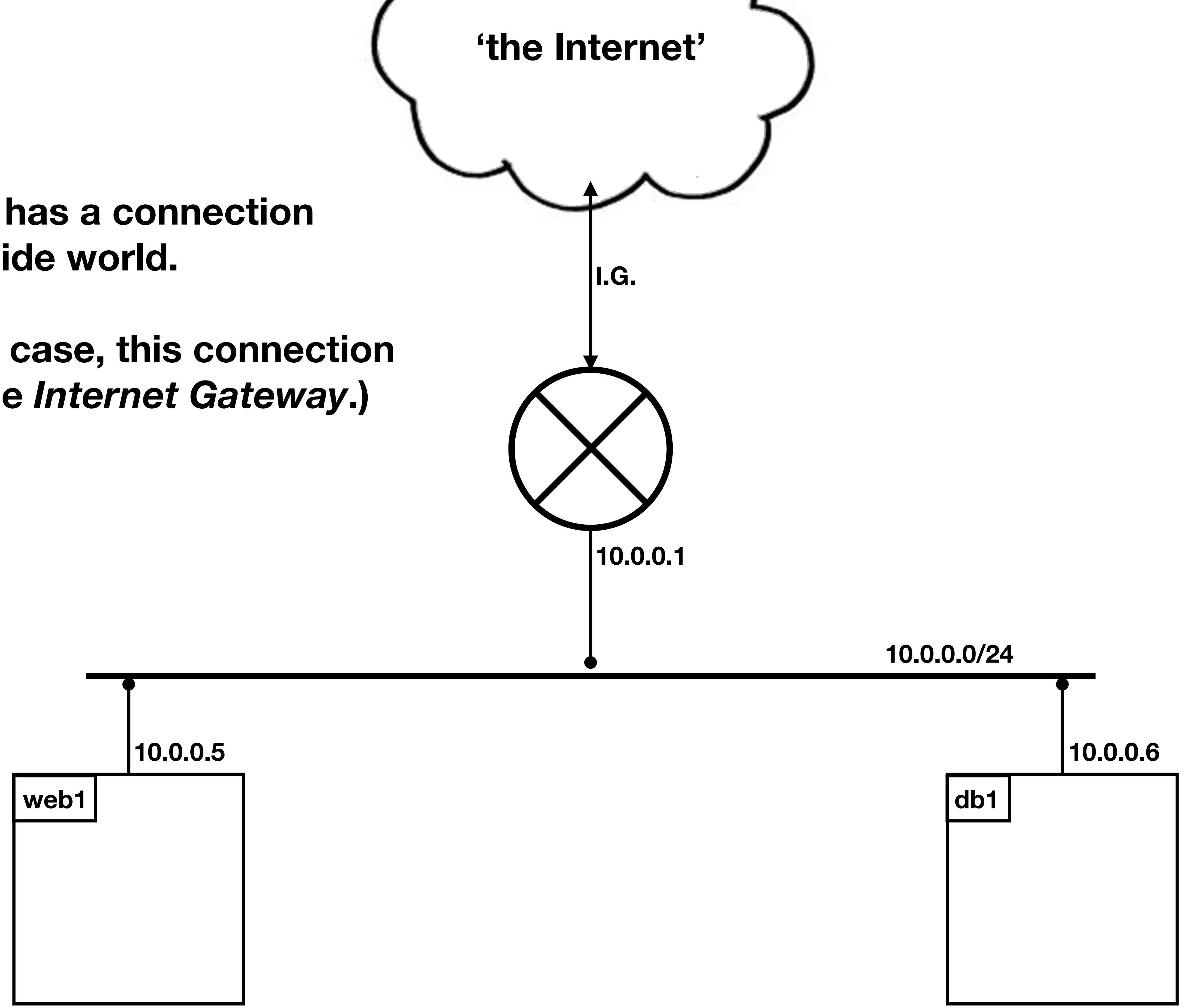
The attachment point has its own (internal) IP address.

You’ll see this called a *gateway address*.



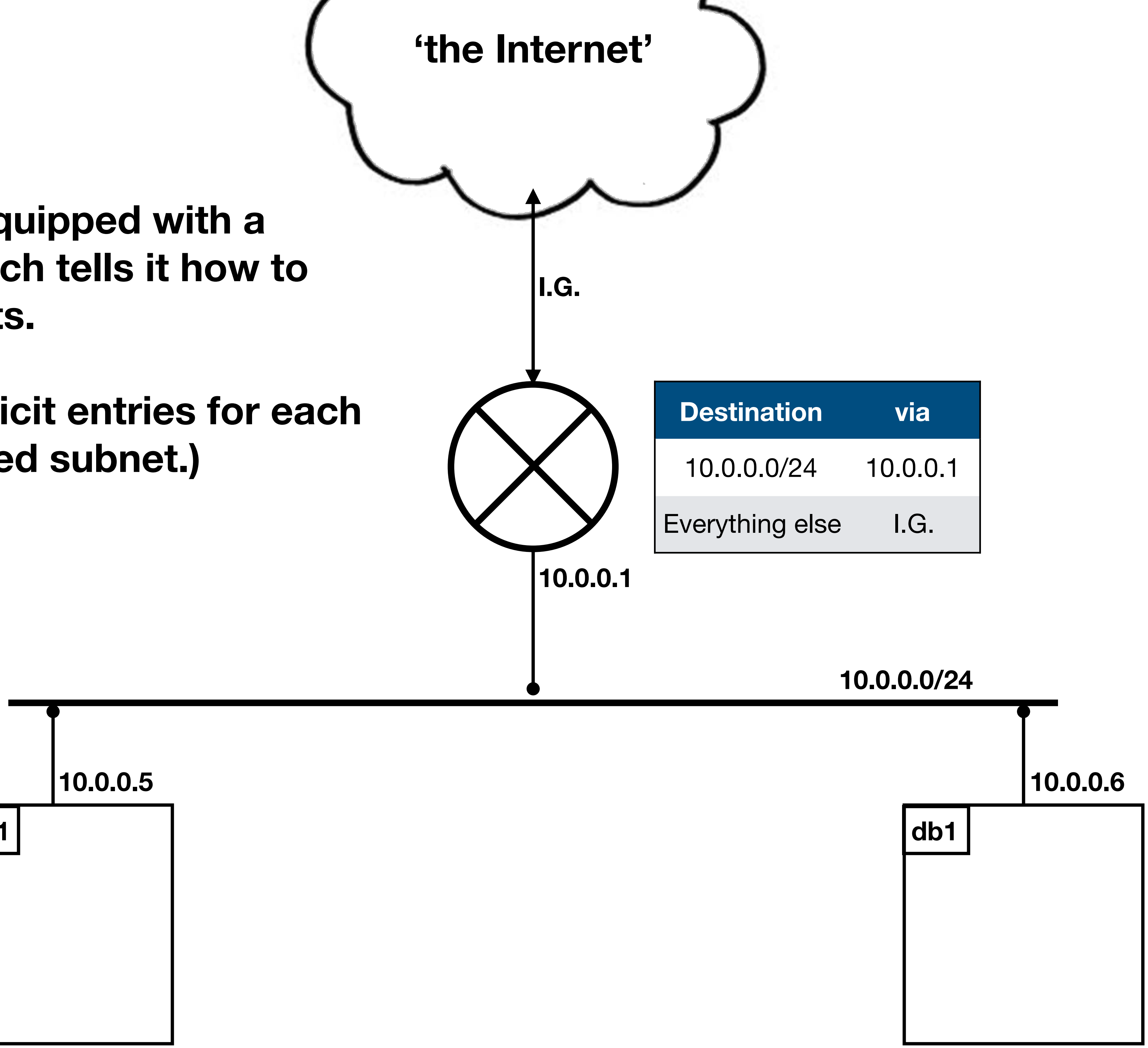
The router has a connection to the outside world.

(In the OCI case, this connection is called the *Internet Gateway*.)



The router is equipped with a *route table* which tells it how to forward packets.

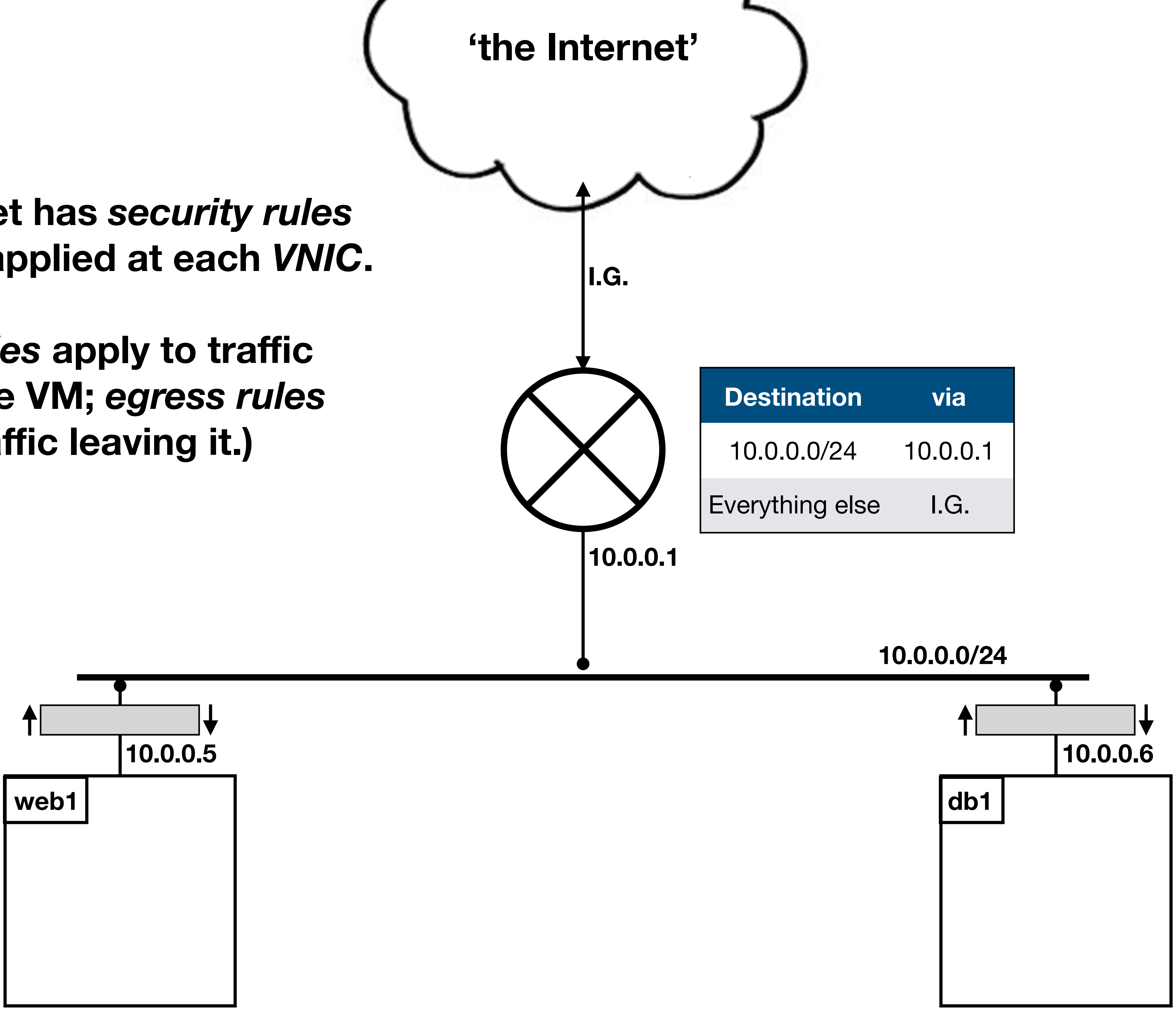
(There are implicit entries for each directly-attached subnet.)



Destination	via
10.0.0.0/24	10.0.0.1
Everything else	I.G.

Each subnet has *security rules* which are applied at each *VNIC*.

(*Ingress rules* apply to traffic entering the VM; *egress rules* apply to traffic leaving it.)



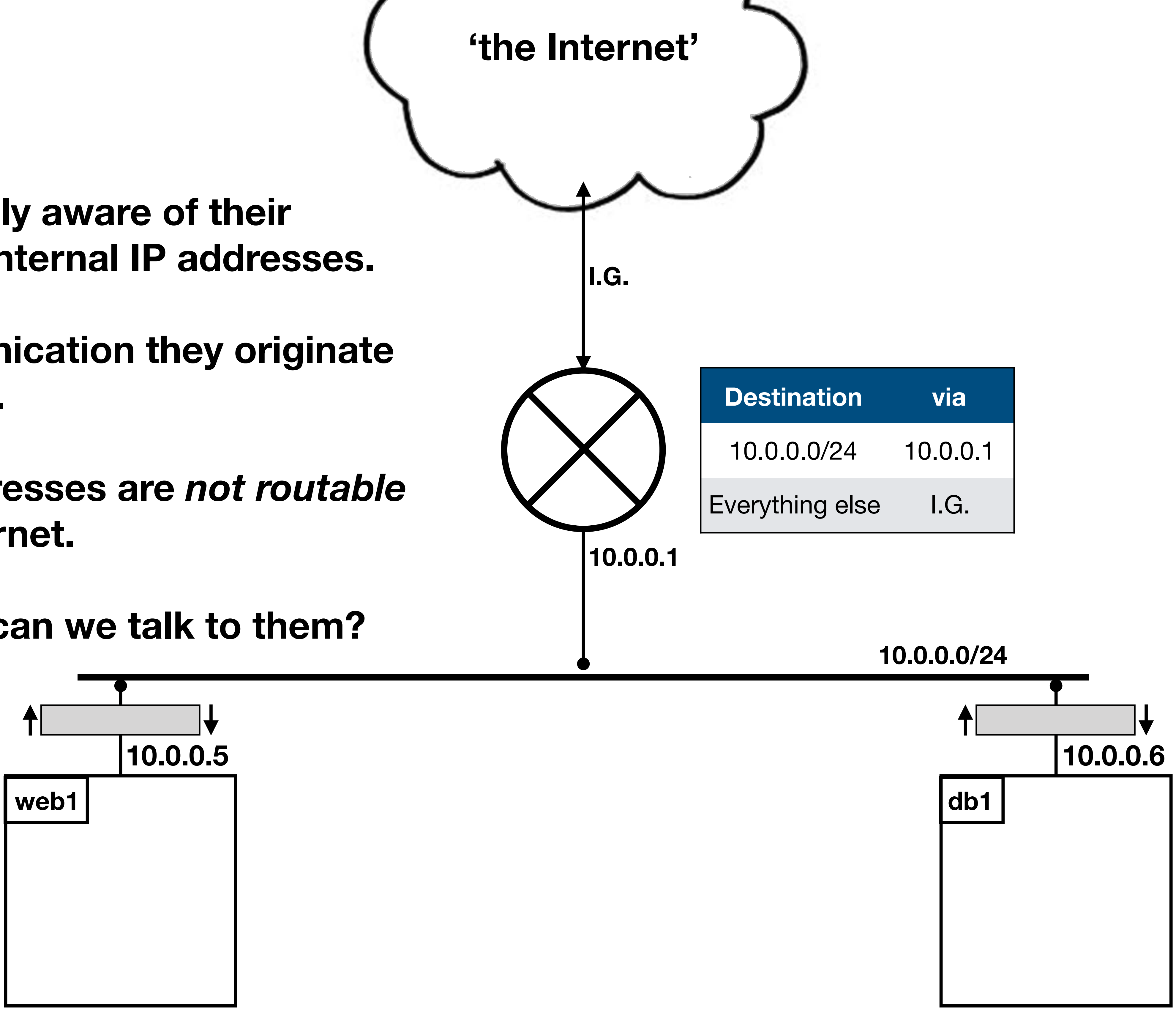
Destination	via
10.0.0.0/24	10.0.0.1
Everything else	I.G.

VMs are only aware of their *private* or internal IP addresses.

All communication they originate uses those.

Those addresses are *not routable* on the Internet.

… so how can we talk to them?



Destination	via
10.0.0.0/24	10.0.0.1
Everything else	I.G.

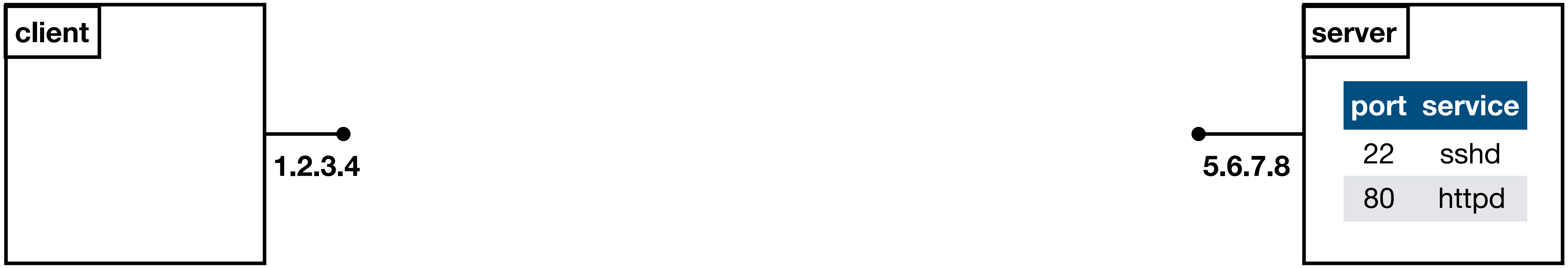
Anatomy of a TCP connection

- Each host involved has (at least) one address

client
1.2.3.4
server
5.6.7.8

Anatomy of a TCP connection

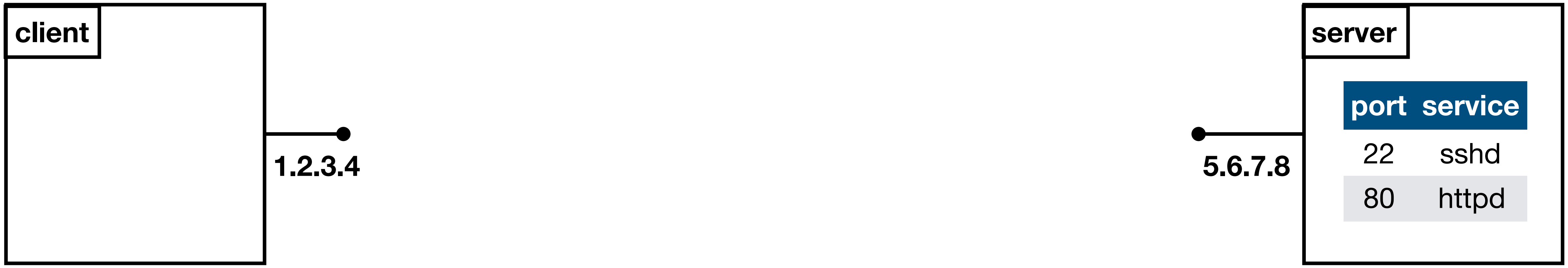
- A *server* may have several services on it. Each listens on a unique *port.* Typically, well-known services have port numbers assigned to them.



port	service
22	sshd
80	httpd

Anatomy of a TCP connection

- A *client* might want to make several outgoing calls to the same service at the same time. To distinguish them, the client also allocates an *ephemeral port* (typically from higher in the available range).



client

1.2.3.4

5.6.7.8

server

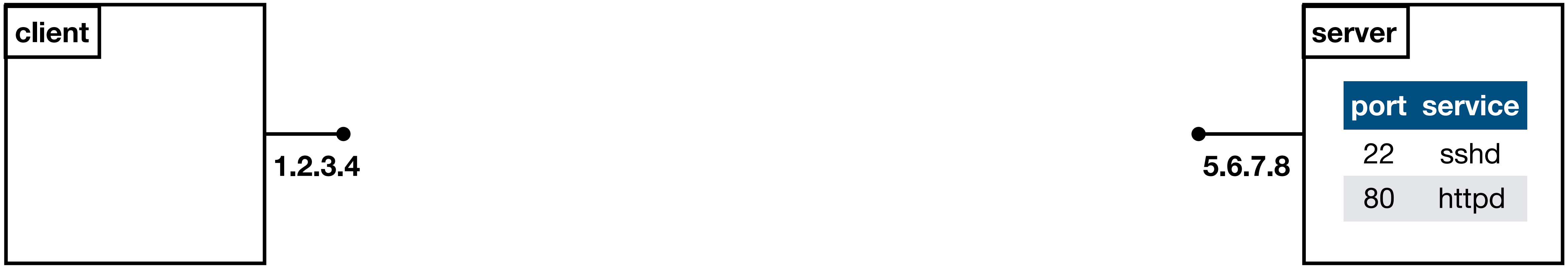
port	service
22	sshd
80	httpd

```
ssh user@5.6.7.8
```

(port 32769 allocated)

Anatomy of a TCP connection

- A connection is identified by a 4-tuple:
  (local address, local port, remote address, remote port)
  All TCP packets carry these addresses.



port	service
22	sshd
80	httpd

```
ssh user@5.6.7.8
```

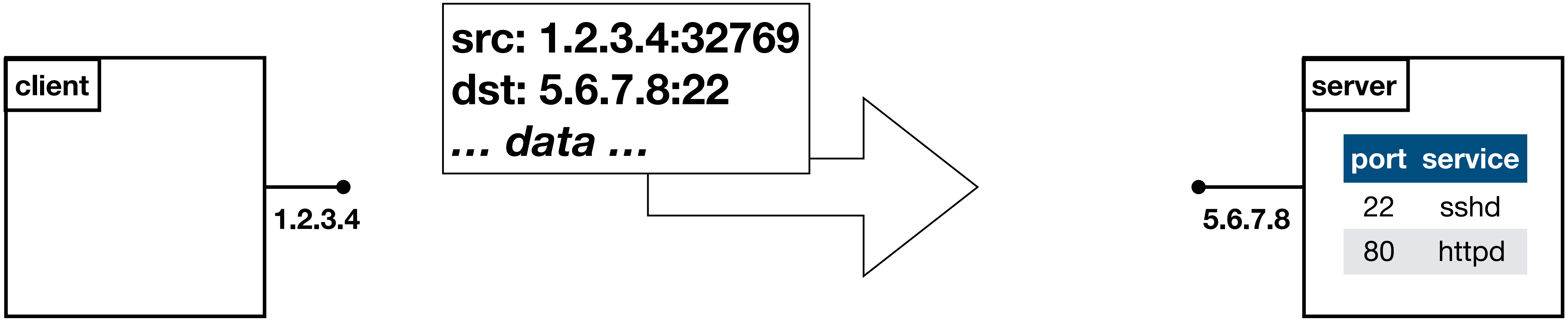
(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	5.6.7.8	22	ssh

local addr	local port	remote addr	remote port	process
5.6.7.8	22	1.2.3.4	32769	sshd

Anatomy of a TCP connection

- A connection is identified by a 4-tuple:
  (local address, local port, remote address, remote port)
  All TCP packets carry these addresses.



src: 1.2.3.4:32769
dst: 5.6.7.8:22
… data …

port	service
22	sshd
80	httpd

```
ssh user@5.6.7.8
```

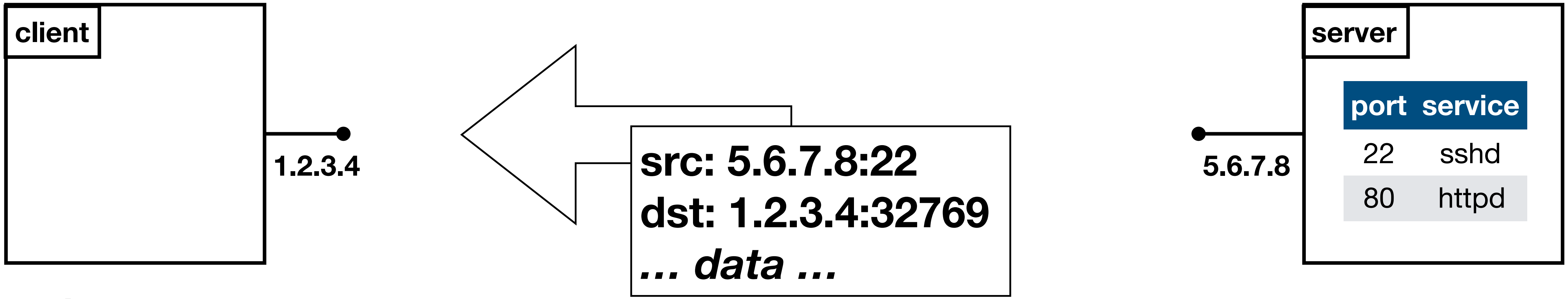
(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	5.6.7.8	22	ssh

local addr	local port	remote addr	remote port	process
5.6.7.8	22	1.2.3.4	32769	sshd

Anatomy of a TCP connection

- A connection is identified by a 4-tuple:
  (local address, local port, remote address, remote port)
  All TCP packets carry these addresses.



src: 5.6.7.8:22
dst: 1.2.3.4:32769
... data ...

port	service
22	sshd
80	httpd

```
ssh user@5.6.7.8
```

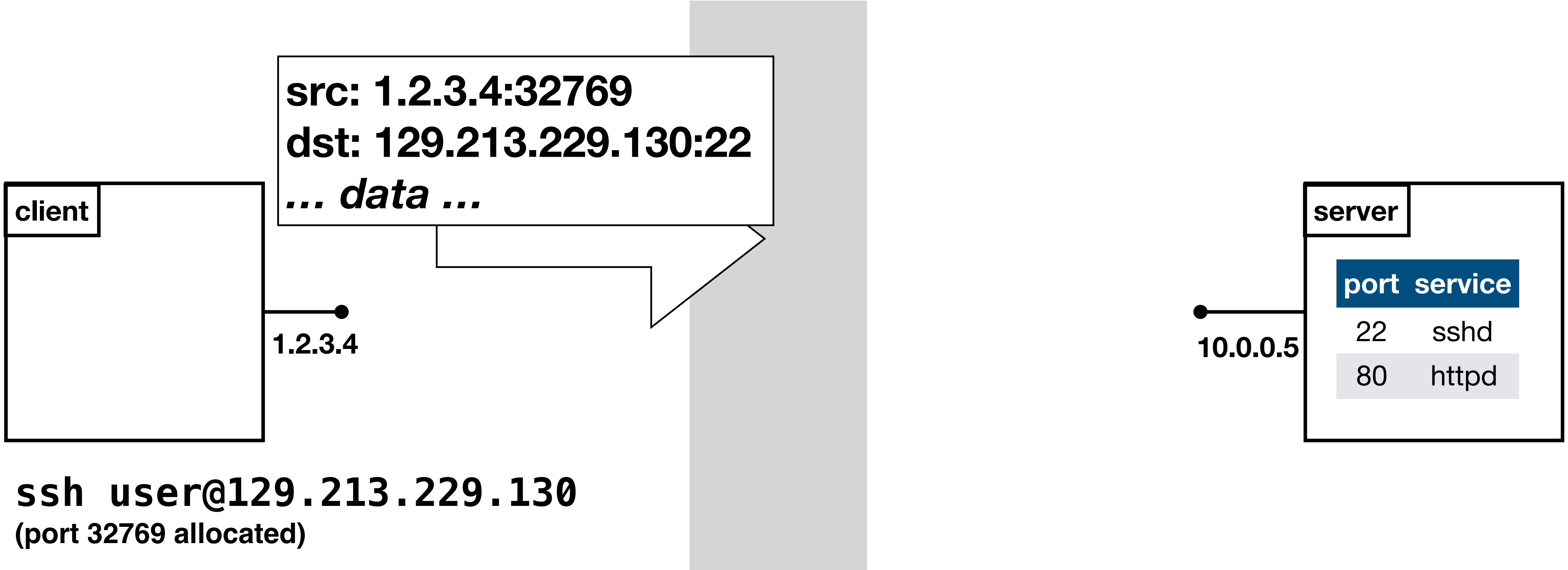
(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	5.6.7.8	22	ssh

local addr	local port	remote addr	remote port	process
5.6.7.8	22	1.2.3.4	32769	sshd

Anatomy of a TCP connection

- The *Internet Gateway* rewrites addresses on inbound and outbound packets. The server sees a local address; the client sees the public one.



port	service
22	sshd
80	httpd

```
ssh user@129.213.229.130
```

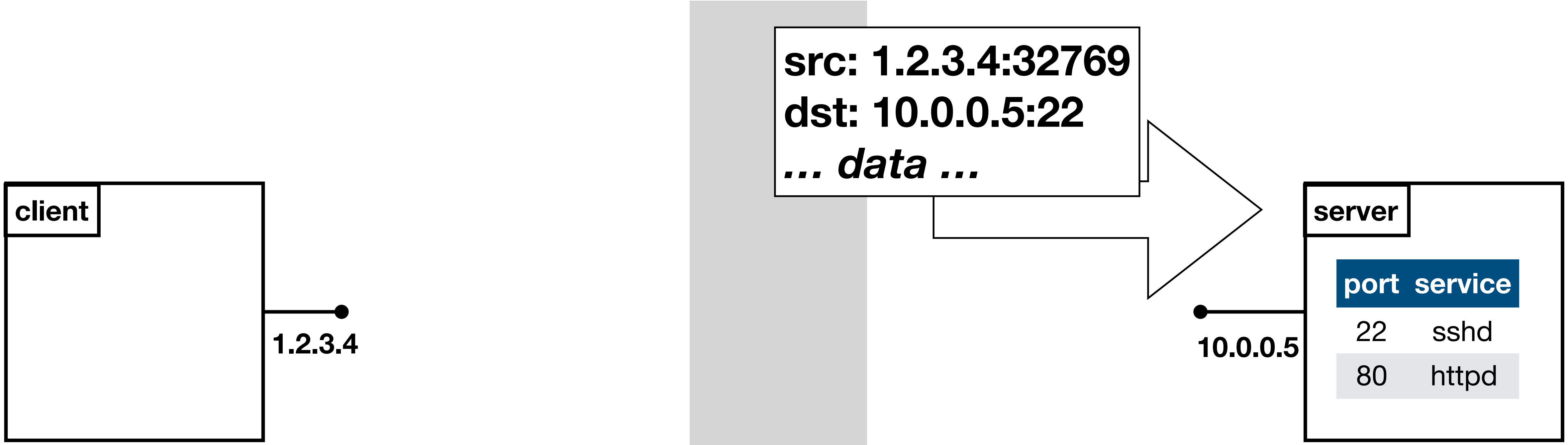
(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	129.213.229.130	22	ssh

local addr	local port	remote addr	remote port	process
10.0.0.5	22	1.2.3.4	32769	sshd

Anatomy of a TCP connection

- The *Internet Gateway* rewrites addresses on inbound and outbound packets. The server sees a local address; the client sees the public one.



port	service
22	sshd
80	httpd

```
ssh user@129.213.229.130
```

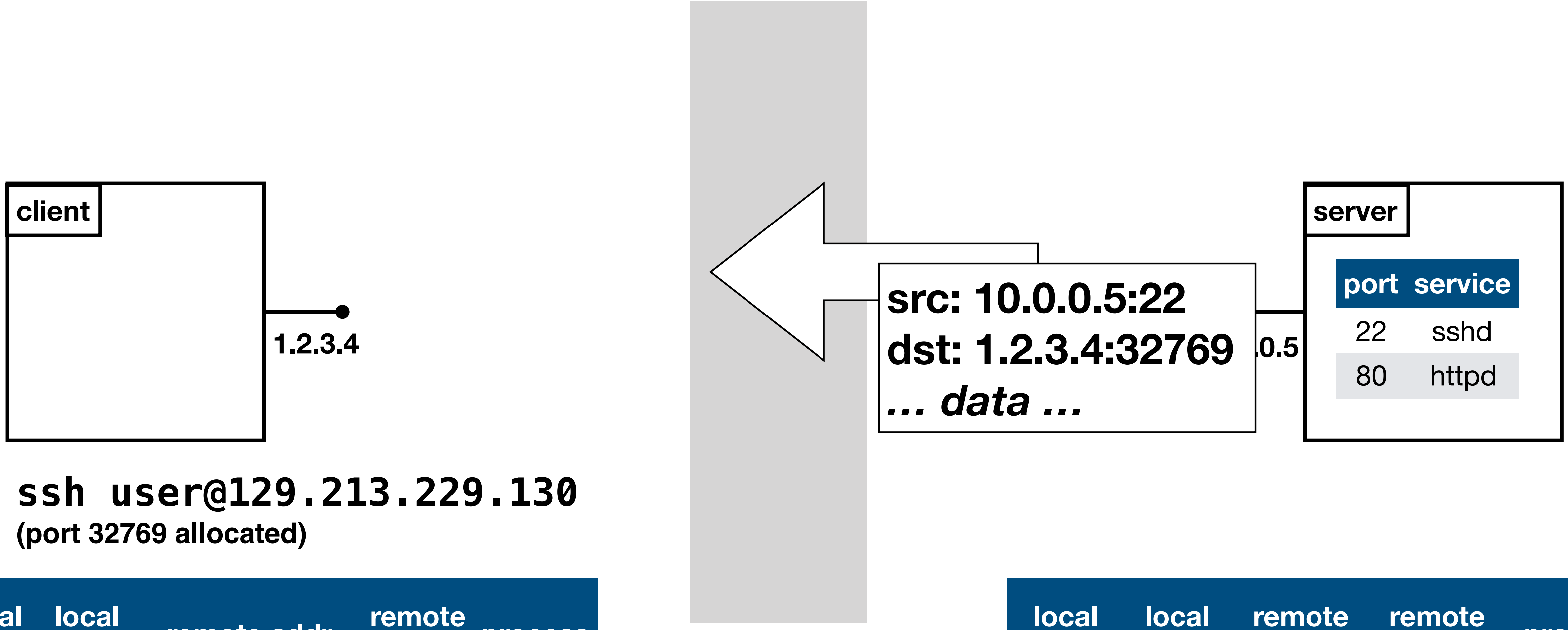
(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	129.213.229.130	22	ssh

local addr	local port	remote addr	remote port	process
10.0.0.5	22	1.2.3.4	32769	sshd

Anatomy of a TCP connection

- The *Internet Gateway* rewrites addresses on inbound and outbound packets. The server sees a local address; the client sees the public one.



port	service
22	sshd
80	httpd

```
ssh user@129.213.229.130
```

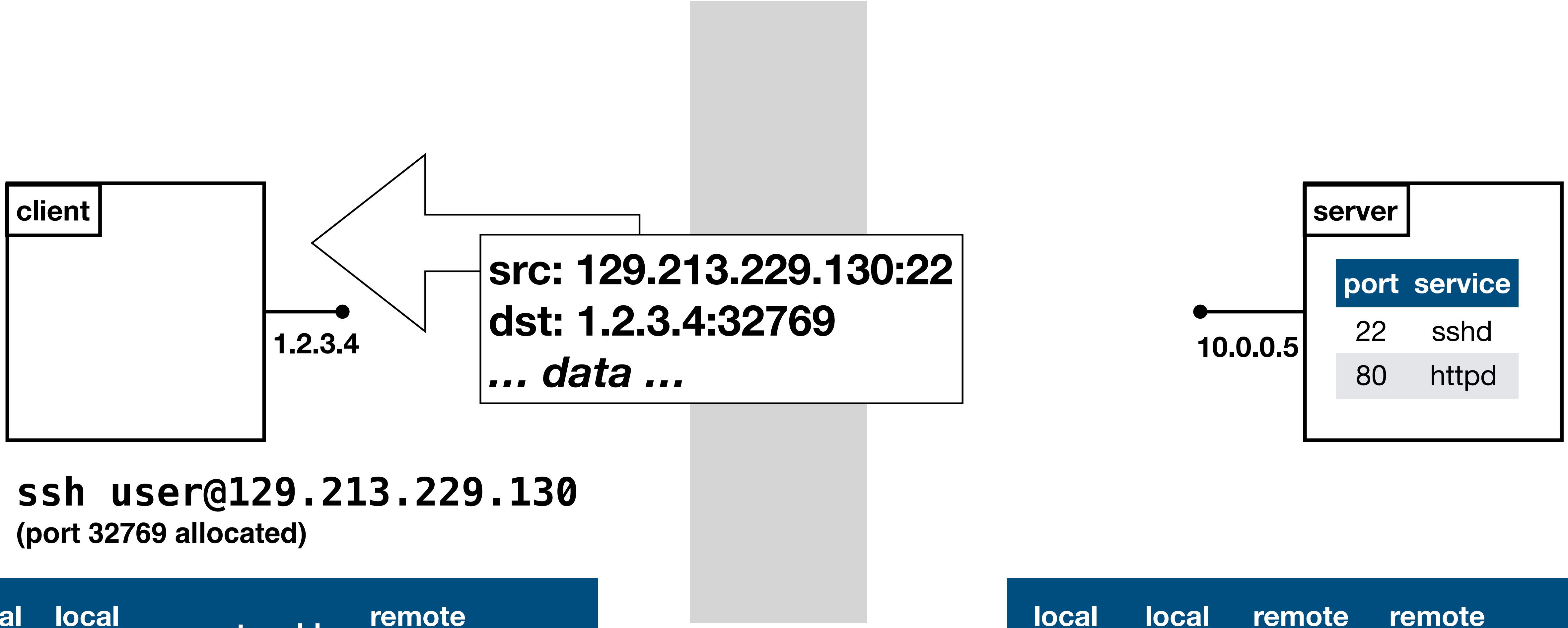
(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	129.213.229.130	22	ssh

local addr	local port	remote addr	remote port	process
10.0.0.5	22	1.2.3.4	32769	sshd

Anatomy of a TCP connection

- The *Internet Gateway* rewrites addresses on inbound and outbound packets. The server sees a local address; the client sees the public one.



src: 129.213.229.130:22
dst: 1.2.3.4:32769
… data …

port	service
22	sshd
80	httpd

```
ssh user@129.213.229.130
```

(port 32769 allocated)

local addr	local port	remote addr	remote port	process
1.2.3.4	32769	129.213.229.130	22	ssh

local addr	local port	remote addr	remote port	process
10.0.0.5	22	1.2.3.4	32769	sshd

Armed with that, you can inspect these connection tables using *netstat*

(There are other commands that’ll give you this information)

Command-line options vary on a Mac: use
`netstat -anf inet`

```
% netstat -an46
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:57439           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 192.168.99.4:44112      192.168.99.2:22         ESTABLISHED
```

How does the VM configure itself?

- It has an IP address, routing table, ssh key, etc.

  There's a low-level and a higher-level configuration

dhcp

(dynamic host configuration protocol)

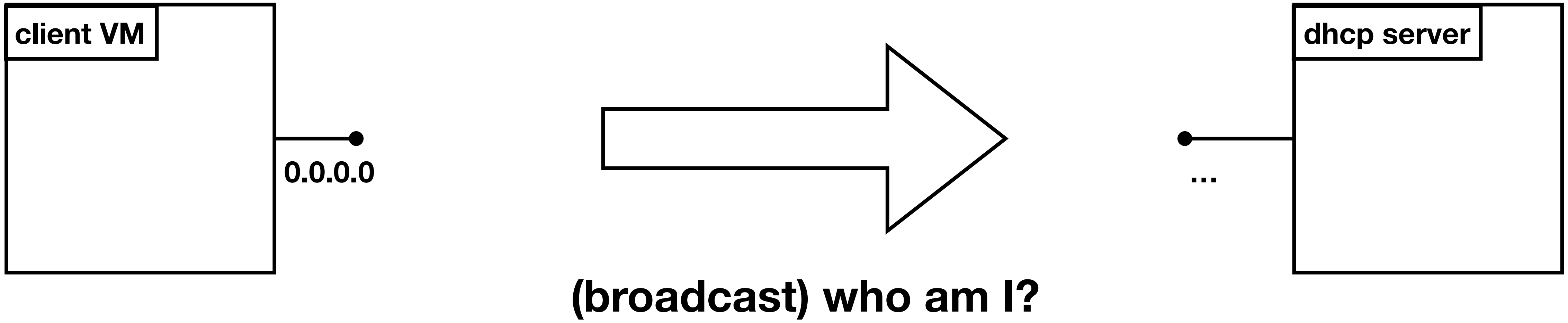
How does the instance get its IP address?

client VM
0.0.0.0
...
dhcp server

dhcp

(dynamic host configuration protocol)

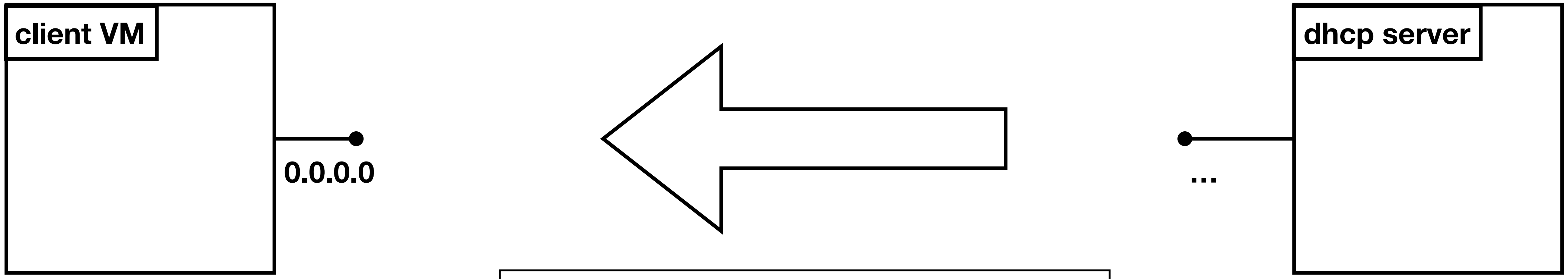
How does the instance get its IP address?



dhcp

(dynamic host configuration protocol)

How does the instance get its IP address?



fixed-ip: 10.0.0.5
subnet-mask: /24
MTU: 9000 bytes

routing table: …

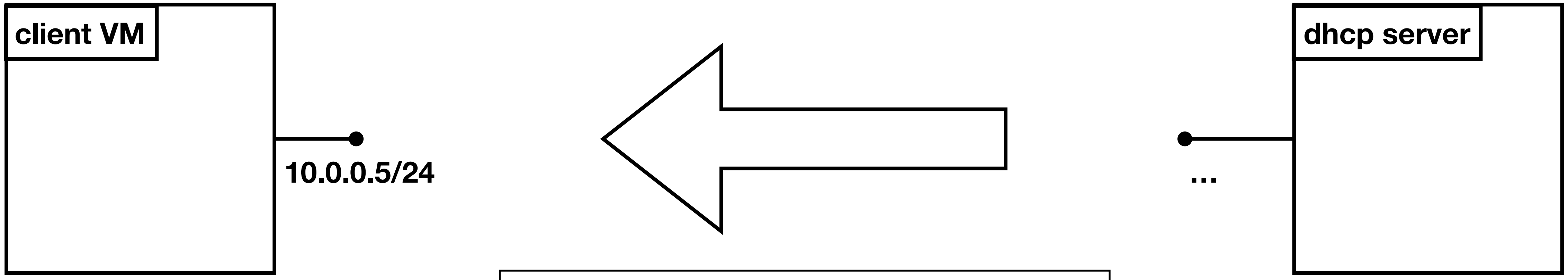
dns domain: …
dns server: 169.254.169.254

lease time: 1 day

dhcp

(dynamic host configuration protocol)

How does the instance get its IP address?



fixed-ip: 10.0.0.5
subnet-mask: /24
MTU: 9000 bytes

routing table: …

dns domain: …
dns server: 169.254.169.254

lease time: 1 day

Other host metadata

(this includes your ssh key)

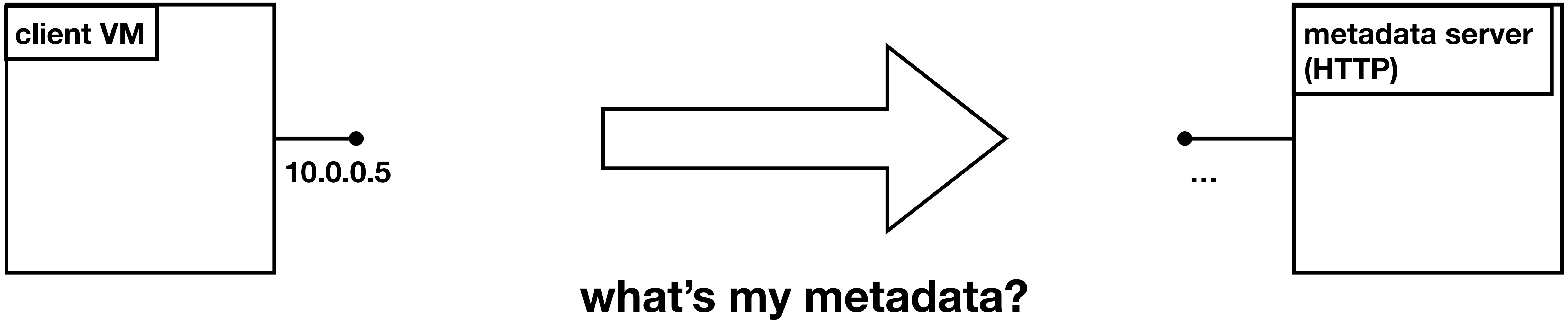
On first boot, the OS image is configured to pull additional configuration from a metadata server.

(You can search for *cloud-init* for more details)

first boot configuration

(cloud-init or equivalent)

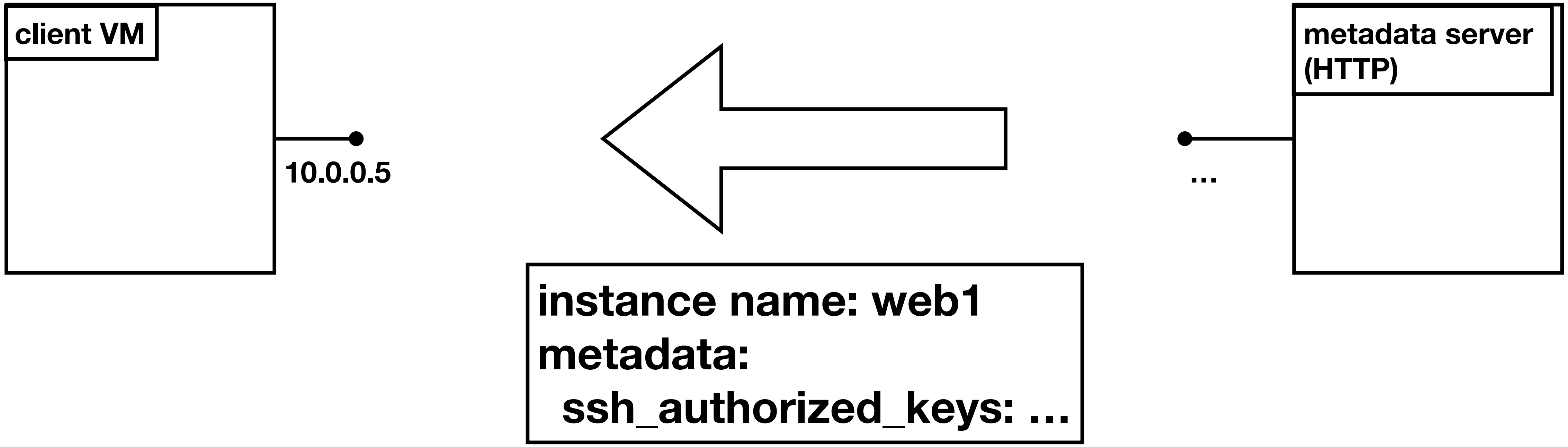
How does the instance get its remaining configuration?



first boot configuration

(cloud-init or equivalent)

How does the instance get its remaining configuration?



```
instance name: web1
metadata:
  ssh_authorized_keys: ...
```

Review of the deployment plan

- Put a database on *db1*
- Put the java application on *web1*

Next steps: log in

In two separate shell windows, launch a connection to each VM.

If you’ve picked Oracle Linux, the username to log in with is *opc. (For ubuntu, it’s “ubuntu”.)*

The VMs should have been configured to accept your ssh key.

(Troubleshooting: add the *-v* flag to the ssh command for debugging output.)

```
% ssh opc@129.213.119.230
The authenticity of host '129.213.119.230 (129.213.119.230)' can't be established.
ECDSA key fingerprint is SHA256:j01Kp0TAJTJgcKdlhecIH7b3KfghdMDoIA5viaMuNWY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '129.213.119.230' (ECDSA) to the list of known hosts.
[opc@web1 ~]$
```

Next: deploy and configure the database

By way of warning: we’re likely to repeatedly find “it doesn’t work”.

Small amounts of progress punctuated by systematic troubleshooting (or head-scratching) is *normal*.

When things don’t work, there’s usually a good reason - we just need to establish a process for narrowing down where the problem lies.

Deploying the database

- Plan: we'll use the pre-packaged software to do this:
  - Install the software
  - Make sure it's turned on
  - Change the *root* password
  - Add credentials and an empty database for the application to use

What’s a *package*?

- Metadata:
    - Name, version, description, …
    - Dependencies and conflicts
    - “Provides” - both concrete and abstract
- pre-, post-installation scripts
- pre-, post-removal scripts
- File contents

What’s a *package repository*?

- Usually just a web-server with a predictable layout
- The package metadata is collected into an index
- Package archives individually downloadable
- Typically protected using cryptographic signing
  (keys are preconfigured on the host)

Database: installation

```
[opc@db1 ~]$ sudo yum-config-manager --enable ol7_MySQL57
[opc@db1 ~]$ sudo yum install mysql-server
[opc@db1 ~]$ systemctl status mysqld.service
[opc@db1 ~]$ sudo systemctl enable mysqld.service
[opc@db1 ~]$ sudo systemctl start mysqld.service
```

This ought to set up a database server.

It’ll be empty!

Check that it’s running: look for it with

```
ps -ef
```

(look for *mysqld*)

Check that it’s listening for connections:

```
netstat -an46
```

(you might see :::3306)

Database: change the root password

```
[opc@db1 ~]$ sudo grep password /var/log/mysqld.log
2018-09-14T14:11:29.382824Z 1 [Note] A temporary password is generated for root@localhost: b;vRuNWWq8yK

[opc@db1 ~]$ mysql -u root -p
Enter password: b;vRuNWWq8yK                        <- you won’t see this
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.23

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> alter user 'root'@'localhost' identified by 'P8%aIjUxIh8:P4Wv';

Query OK, 0 rows affected (0.00 sec)

mysql> ^DBye                                        <- this is a Control-D
[opc@db1 ~]$
```

Check it worked!

```
[opc@db1 ~]$ mysql -u root -p
Enter password: P8%aIjUxIh8:P4Wv
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

mysql comes with some built-in schemas

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

Create some credentials for our application to use

We would like to limit the privileges available to the app to a bare minimum.

(They could be reduced further than shown here.)

Create some credentials for our application to use

```
mysql> create user if not exists 'app'@'%' identified by 'DxIHXE%6d7sD:EXI';
Query OK, 0 rows affected (0.00 sec)
```

These are the credentials we'll use from the application

Create a blank database schema for the app

```
mysql> create database app;
Query OK, 1 row affected (0.00 sec)

mysql> grant all privileges on app.* to 'app'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> ^DBye
```

Check!!

- If you confirm that things are working at each step, then unpicking problems can be less daunting.

Check!!

```
[opc@db1 ~]$ mysql -u app app -p
Enter password: DxIHXE%6d7sD:EXI
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.7.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
Empty set (0.00 sec)

mysql>
```

Check!!

```
mysql> create table first (a integer);
Query OK, 0 rows affected (0.02 sec)

mysql> describe first;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.01 sec)

mysql>
```

Summary

- We’ve installed a database server
- We’ve added credentials for a new user to it (and the password isn’t just “secret” :-))
- We’ve created a database
- We’ve confirmed that it works

Deployment plan

- We want to be able to talk to the database from our web host.
  - This will involve some additional configuration
  - At the moment our VMs can’t talk to each other!

Inter-VM communication

```
[opc@web1 ~]$ ping db1
PING db1.sub09141050190.vcn0914105019.oraclevcn.com (10.0.0.6) 56(84) bytes of data.
^C
--- db1.sub09141050190.vcn0914105019.oraclevcn.com ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 4999ms
```

Inter-VM communication

We can see two things from this:

- The hosts are configured to correctly find each other’s *private* IP addresses by name.

  (This uses a private, internal DNS server.)

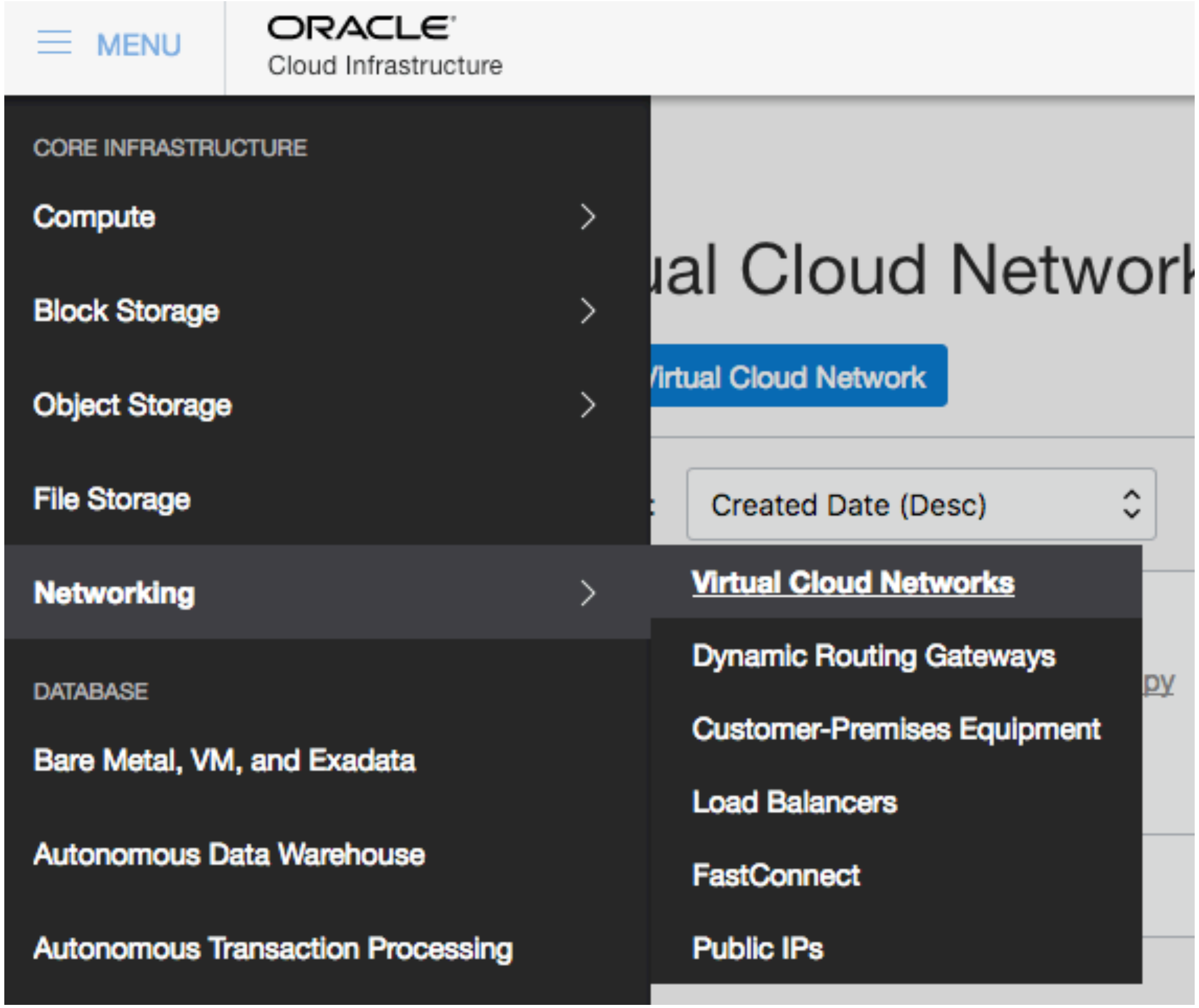
- Traffic between VMs is blocked.

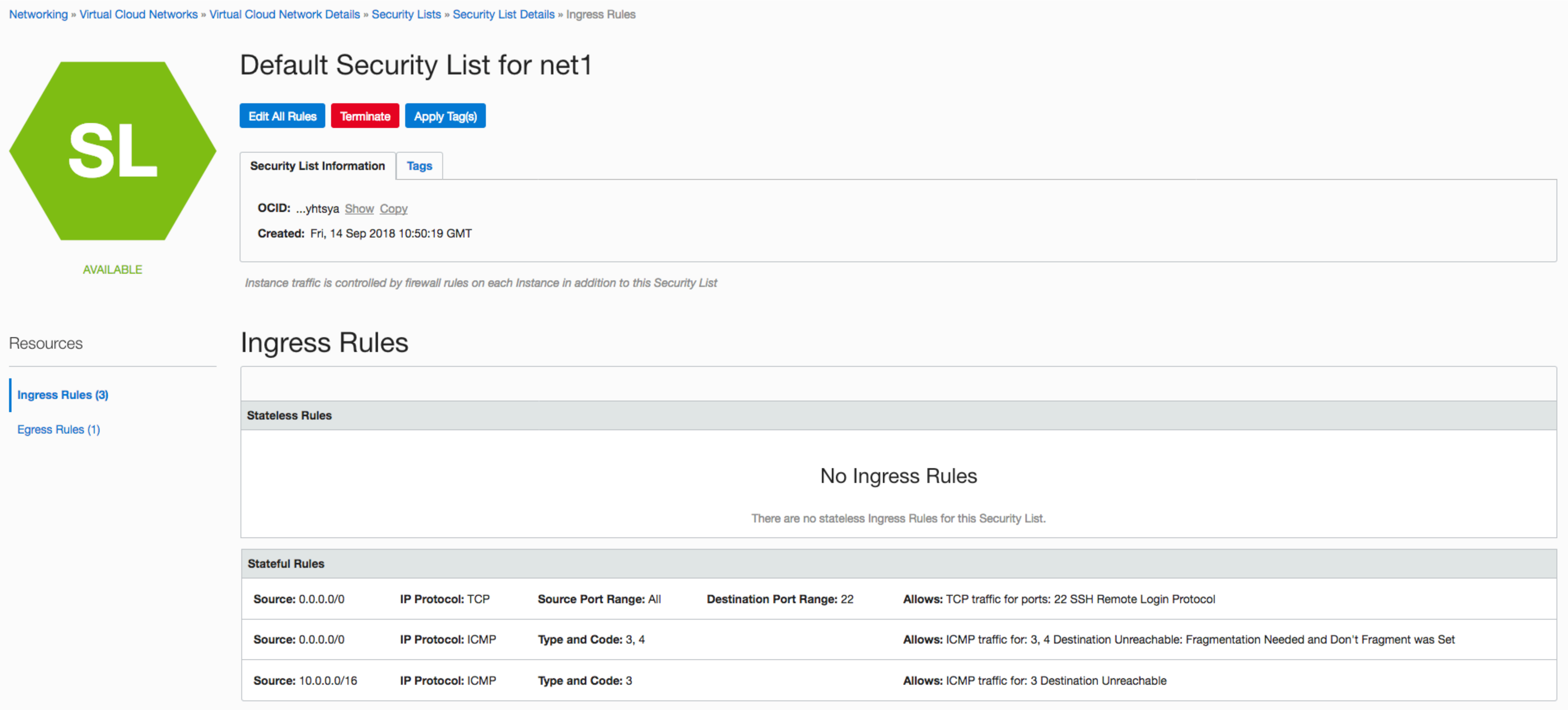
The next hurdle: network security

- By default, you can get ssh traffic in to hosts from the internet at large…
- And arbitrary connections from hosts out to the internet at large…
  - (this is useful for software installation!)
- … but that's pretty much it.
- Network traffic between hosts on a subnet is limited!
- The principle here is: *deny by default*

Network security part one: subnet security rules

- These are applied *outside* of the VMs
- They control all traffic on the subnet
- That includes *between* VMs, *to* them from the outside, and *from* them.
    - To begin with, we want our Java application to be able to talk to the database.
    - We'll use mysql for the database; by default, that listens on port 3306, so we'll want to permit traffic *across our subnet* that's destined to that port.


MENU
ORACLE
Cloud Infrastructure
CORE INFRASTRUCTURE
Compute
Block Storage
Object Storage
File Storage
Networking
DATABASE
Bare Metal, VM, and Exadata
Autonomous Data Warehouse
Autonomous Transaction Processing
Virtual Cloud Networks
Dynamic Routing Gateways
Customer-Premises Equipment
Load Balancers
FastConnect
Public IPs
ual Cloud Network
/irtual Cloud Network
Created Date (Desc)


Networking » Virtual Cloud Networks » Virtual Cloud Network Details » Security Lists » Security List Details » Ingress Rules
SL
AVAILABLE
Edit All Rules
Terminate
Apply Tag(s)
Security List Information
Tags
Show
Copy
Resources
Ingress Rules (3)
Egress Rules (1)

Default Security List for net1

OCID: ...yhtsya

Created: Fri, 14 Sep 2018 10:50:19 GMT

Instance traffic is controlled by firewall rules on each Instance in addition to this Security List

Ingress Rules

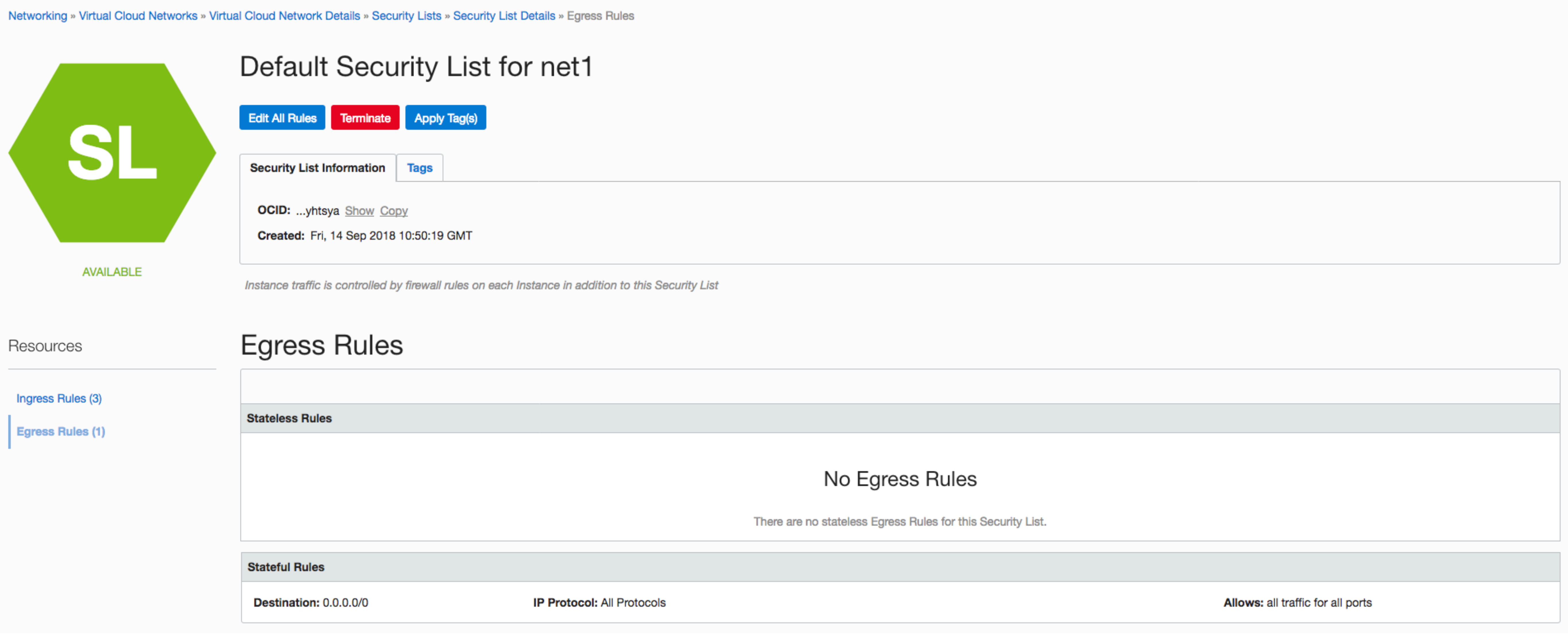
Stateless Rules

No Ingress Rules

There are no stateless Ingress Rules for this Security List.

Stateful Rules

Source: 0.0.0.0/0	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 22	**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol
Source: 0.0.0.0/0	**IP Protocol:** ICMP	**Type and Code:** 3, 4		**Allows:** ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set
Source: 10.0.0.0/16	**IP Protocol:** ICMP	**Type and Code:** 3		**Allows:** ICMP traffic for: 3 Destination Unreachable


Networking » Virtual Cloud Networks » Virtual Cloud Network Details » Security Lists » Security List Details » Egress Rules
SL
AVAILABLE
Edit All Rules
Terminate
Apply Tag(s)
Security List Information
Tags
Show
Copy
Resources
Ingress Rules (3)
Egress Rules (1)

Default Security List for net1

OCID: ...yhtsya

Created: Fri, 14 Sep 2018 10:50:19 GMT

Instance traffic is controlled by firewall rules on each Instance in addition to this Security List

Egress Rules

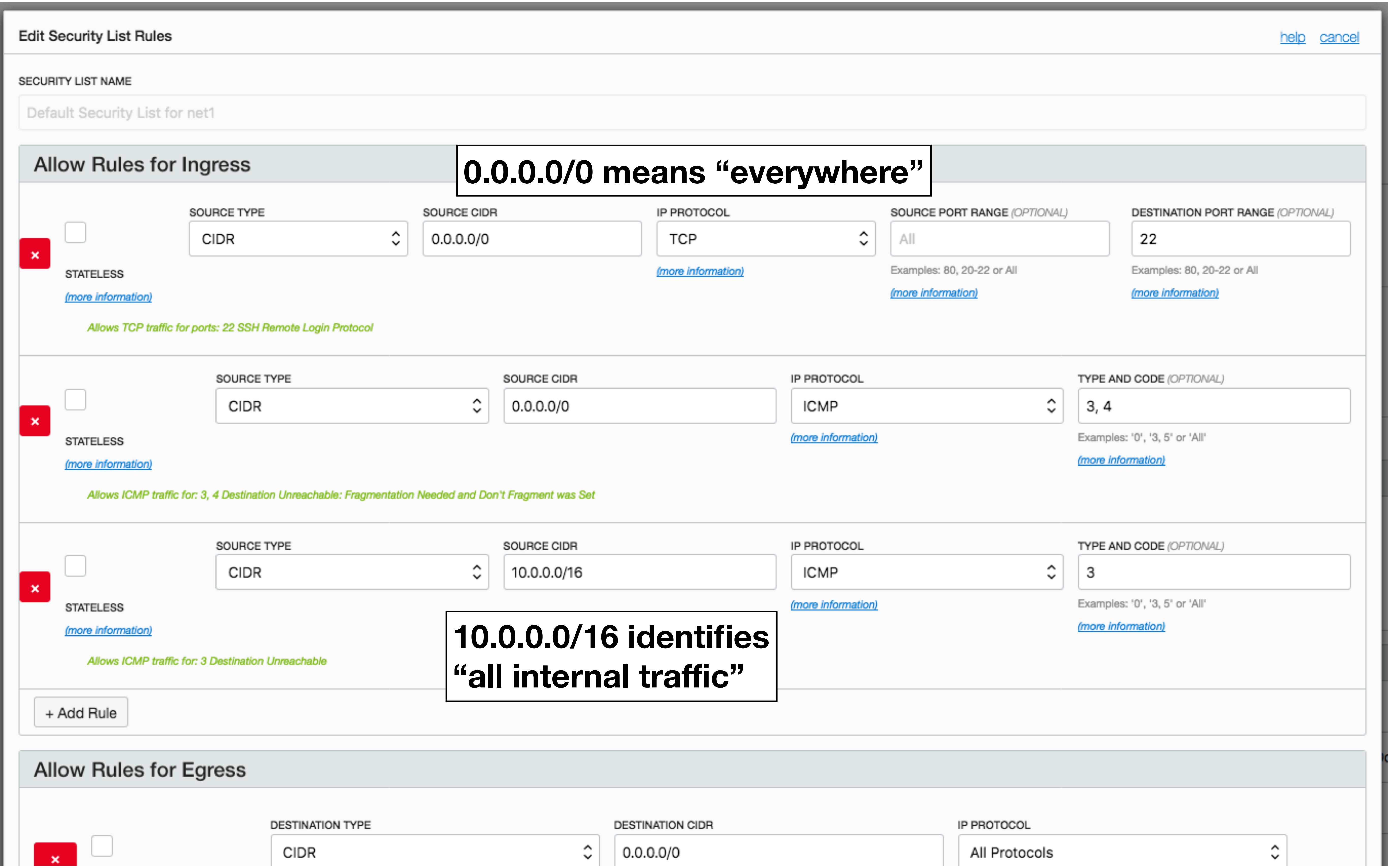
Stateless Rules

No Egress Rules

There are no stateless Egress Rules for this Security List.

Stateful Rules

Destination: 0.0.0.0/0 **IP Protocol:** All Protocols **Allows:** all traffic for all ports



Edit Security List Rules

help cancel

SECURITY LIST NAME

Default Security List for net1

Allow Rules for Ingress

0.0.0.0/0 means “everywhere”

□ STATELESS *(more information)*	SOURCE TYPE CIDR	SOURCE CIDR 0.0.0.0/0	IP PROTOCOL TCP *(more information)*	SOURCE PORT RANGE *(OPTIONAL)* All Examples: 80, 20-22 or All *(more information)*	DESTINATION PORT RANGE *(OPTIONAL)* 22 Examples: 80, 20-22 or All *(more information)*

Allows TCP traffic for ports: 22 SSH Remote Login Protocol

□ STATELESS *(more information)*	SOURCE TYPE CIDR	SOURCE CIDR 0.0.0.0/0	IP PROTOCOL ICMP *(more information)*	TYPE AND CODE *(OPTIONAL)* 3, 4 Examples: '0', '3, 5' or 'All' *(more information)*

Allows ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set

□ STATELESS *(more information)*	SOURCE TYPE CIDR	SOURCE CIDR 10.0.0.0/16	IP PROTOCOL ICMP *(more information)*	TYPE AND CODE *(OPTIONAL)* 3 Examples: '0', '3, 5' or 'All' *(more information)*

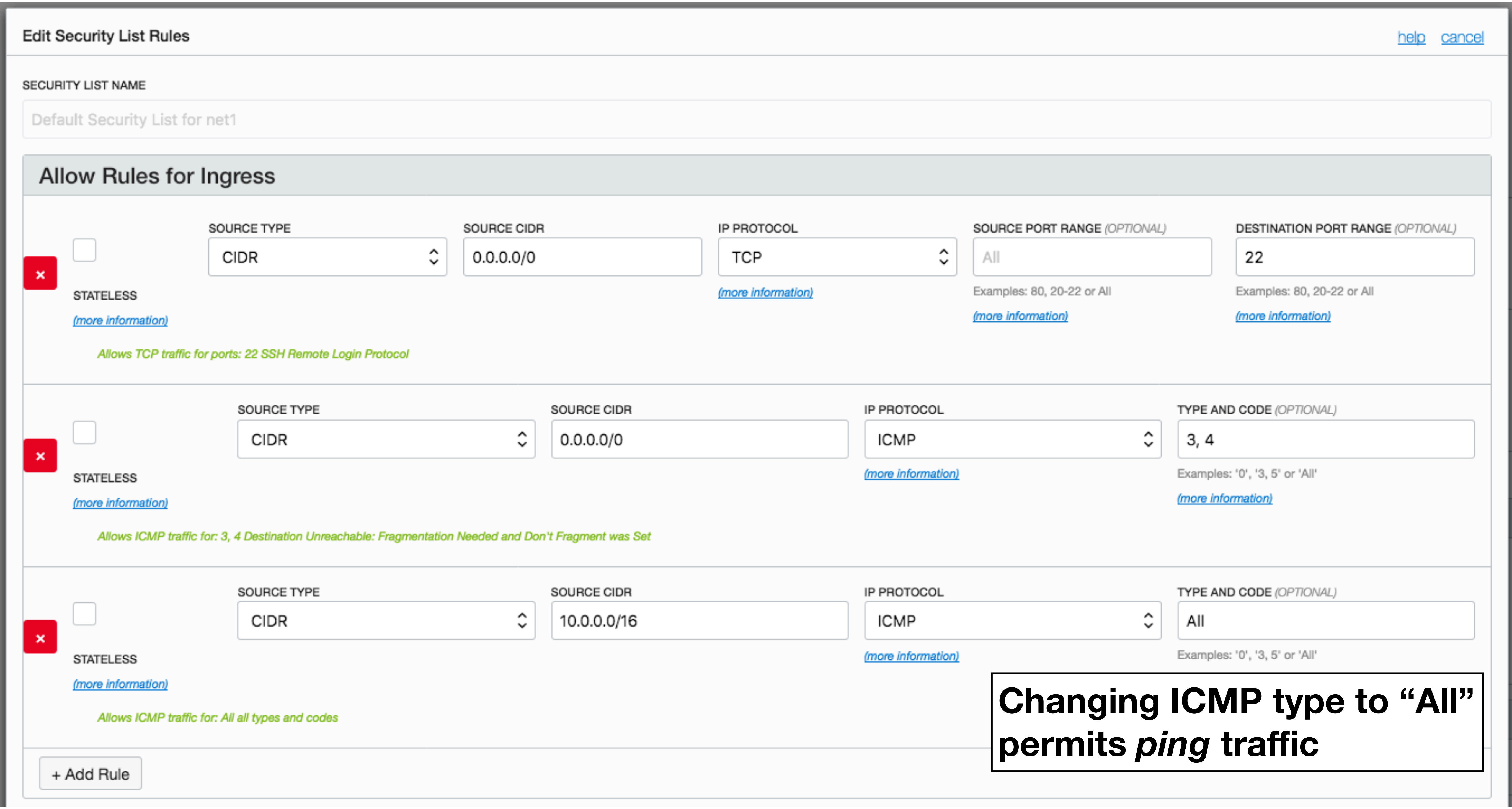
Allows ICMP traffic for: 3 Destination Unreachable

10.0.0.0/16 identifies “all internal traffic”

+ Add Rule

Allow Rules for Egress

□	DESTINATION TYPE CIDR	DESTINATION CIDR 0.0.0.0/0	IP PROTOCOL All Protocols



Edit Security List Rules

SECURITY LIST NAME

Default Security List for net1

Allow Rules for Ingress

☒	□ STATELESS (more information)	SOURCE TYPE: CIDR	SOURCE CIDR: 0.0.0.0/0	IP PROTOCOL: TCP (more information)	SOURCE PORT RANGE (OPTIONAL): All Examples: 80, 20-22 or All (more information)
					DESTINATION PORT RANGE (OPTIONAL): 22 Examples: 80, 20-22 or All (more information)

Allows TCP traffic for ports: 22 SSH Remote Login Protocol

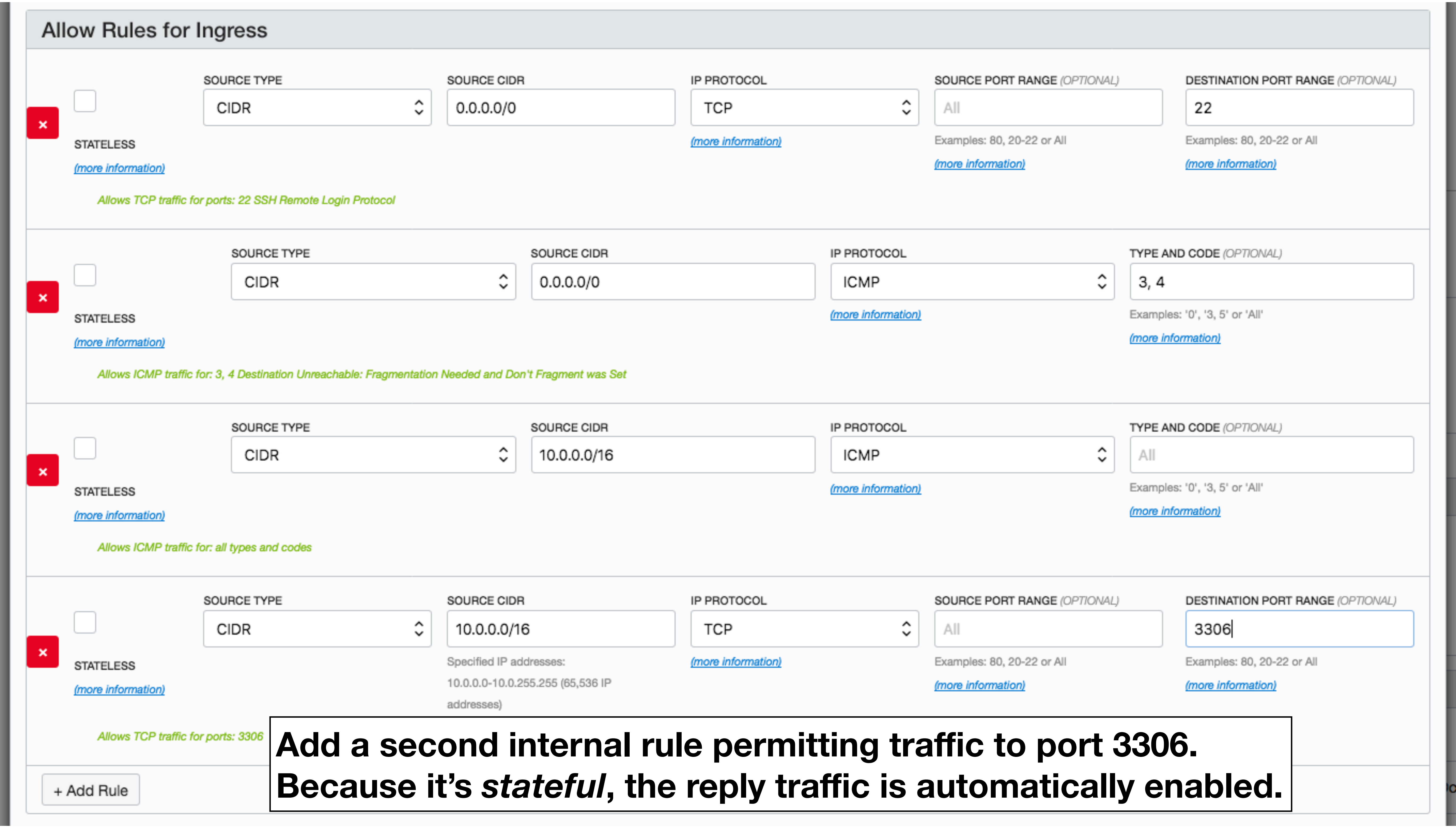
☒	□ STATELESS (more information)	SOURCE TYPE: CIDR	SOURCE CIDR: 0.0.0.0/0	IP PROTOCOL: ICMP (more information)	TYPE AND CODE (OPTIONAL): 3, 4 Examples: '0', '3, 5' or 'All' (more information)

Allows ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set

☒	□ STATELESS (more information)	SOURCE TYPE: CIDR	SOURCE CIDR: 10.0.0.0/16	IP PROTOCOL: ICMP (more information)	TYPE AND CODE (OPTIONAL): All Examples: '0', '3, 5' or 'All'

Allows ICMP traffic for: All all types and codes

Changing ICMP type to "All" permits *ping* traffic



Allow Rules for Ingress

□ STATELESS (more information)	SOURCE TYPE CIDR	SOURCE CIDR 0.0.0.0/0	IP PROTOCOL TCP (more information)	SOURCE PORT RANGE (OPTIONAL) All Examples: 80, 20-22 or All (more information)	DESTINATION PORT RANGE (OPTIONAL) 22 Examples: 80, 20-22 or All (more information)

Allows TCP traffic for ports: 22 SSH Remote Login Protocol

□ STATELESS (more information)	SOURCE TYPE CIDR	SOURCE CIDR 0.0.0.0/0	IP PROTOCOL ICMP (more information)	TYPE AND CODE (OPTIONAL) 3, 4 Examples: '0', '3, 5' or 'All' (more information)

Allows ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set

□ STATELESS (more information)	SOURCE TYPE CIDR	SOURCE CIDR 10.0.0.0/16	IP PROTOCOL ICMP (more information)	TYPE AND CODE (OPTIONAL) All Examples: '0', '3, 5' or 'All' (more information)

Allows ICMP traffic for: all types and codes

□ STATELESS (more information)	SOURCE TYPE CIDR	SOURCE CIDR 10.0.0.0/16 Specified IP addresses: 10.0.0.0-10.0.255.255 (65,536 IP addresses)	IP PROTOCOL TCP (more information)	SOURCE PORT RANGE (OPTIONAL) All Examples: 80, 20-22 or All (more information)	DESTINATION PORT RANGE (OPTIONAL) 3306 Examples: 80, 20-22 or All (more information)

Allows TCP traffic for ports: 3306

Add a second internal rule permitting traffic to port 3306. Because it's *stateful*, the reply traffic is automatically enabled.

Try the *ping* check again

```
[opc@web1 ~]$ ping db1
PING db1.sub09141050190.vcn0914105019.oraclevcn.com (10.0.0.6) 56(84) bytes of data.
64 bytes from db1.sub09141050190.vcn0914105019.oraclevcn.com (10.0.0.6): icmp_seq=1
ttl=64 time=0.252 ms
64 bytes from db1.sub09141050190.vcn0914105019.oraclevcn.com (10.0.0.6): icmp_seq=2
ttl=64 time=0.217 ms
64 bytes from db1.sub09141050190.vcn0914105019.oraclevcn.com (10.0.0.6): icmp_seq=3
ttl=64 time=0.238 ms
^C
--- db1.sub09141050190.vcn0914105019.oraclevcn.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.217/0.235/0.252/0.022 ms
[opc@web1 ~]$
```

What about traffic to port 3306?

- We’ll use a low-level tool called *netcat,* or *nc*, to test this.
    - The goal is simply to see if we get any traffic.

- Begin on the *db1* host - can we get traffic locally to the database?

```
[opc@db1 ~]$ sudo yum install -y nc
[opc@db1 ~]$ nc localhost 3306 < /dev/null
J
5.7.2vWHk#U4???26       *MM"mysql_native_password[opc@db1 ~]$


[opc@db1 ~]$ nc db1 3306 < /dev/null
J
5.7.23  XG}a\#x???BG\L:9;Gysql_native_password[opc@db1 ~]$
```

- It might look like gibberish, but there's *some* traffic passing there.

Let’s try the same test from the *web1* host.

```
[opc@web1 ~]$ sudo yum install -y nc
[opc@web1 ~]$ nc db1 3306
(hangs)


enable security rule, then...


[opc@web1 ~]$ nc db1 3306
Ncat: No route to host.
```

Without the security rule enabled, *netcat* simply hangs.

Once the security rule is enabled, however, *netcat* still fails. Why?

Host-based firewalls

- These apply traffic rules within a VM
- The principle here is *defence in depth*
- We need to explicitly permit traffic to the mysql service on the VM *db1*.

VM *db1*: enable access to the mysql service

```
[opc@db1 ~]$ sudo firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-
rpc bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb
dhcp dhcpv6 dhcpv6-client dns docker-registry dropbox-lansync elasticsearch freeipa-
ldap freeipa-ldaps freeipa-replication freeipa-trust ftp ganglia-client ganglia-
master high-availability http https imap imaps ipp ipp-client ipsec iscsi-target
kadmin kerberos kibana klogin kpasswd kshell ldap ldaps libvirt libvirt-tls
managesieve mdns mosh mountd ms-wbt mssql mysql nfs nfs3 nrpe ntp openvpn ovirt-
imageio ovirt-storageconsole ovirt-vmconsole pmcd pmproxy pmwebapi pmwebapis pop3
pop3s postgresql privoxy proxy-dhcp ptp pulseaudio puppetmaster quassel radius rpc-
bind rsh rsyncd samba samba-client sane sip sips smtp smtp-submission smtps snmp
snmptrap spideroak-lansync squid ssh synergy syslog syslog-tls telnet tftp tftp-
client tinc tor-socks transmission-client vdsm vnc-server wbem-https xmpp-bosh xmpp-
client xmpp-local xmpp-server
[opc@db1 ~]$
```

This gives a list of everything it knows about.
We can see the *mysql* service in there.

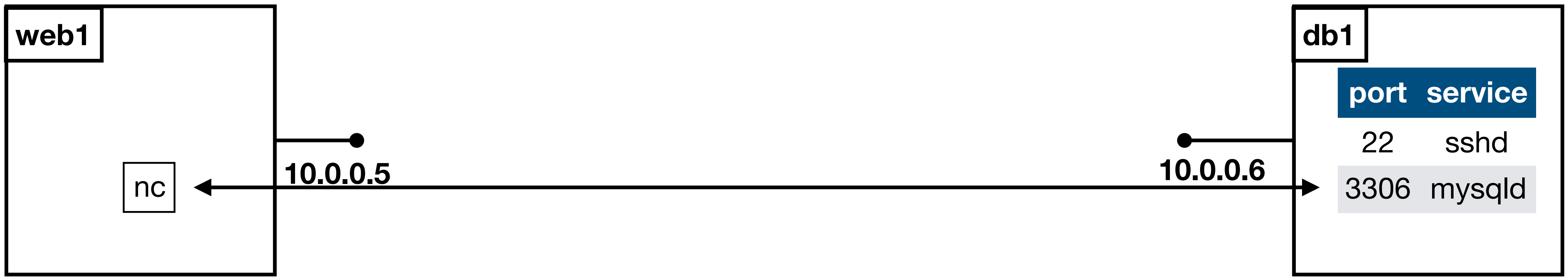
VM *db1*: enable access to the mysql service

```
[opc@db1 ~]$ sudo firewall-cmd --add-service mysql
success
[opc@db1 ~]$ sudo firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client mysql
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

[opc@db1 ~]$ sudo firewall-cmd —-runtime-to-permanent
```

Check!!

```
[opc@web1 ~]$ nc db1 3306 < /dev/null
J
5.7.23
OCiQu&???
         NE1yd}pmysql_native_password[opc@web1 ~]$
```



port	service
22	sshd
3306	mysqld

Check!!

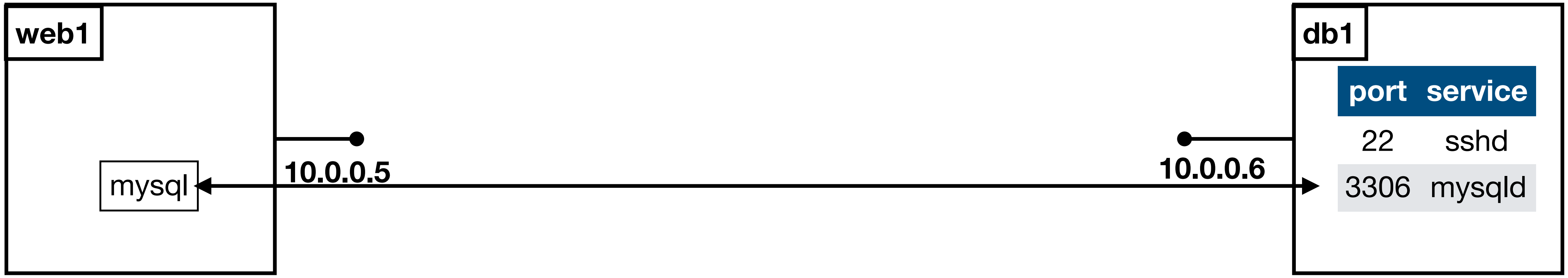
- We’ll install the *mysql* command-line client for a more sophisticated check.

```
[opc@web1 ~]$ sudo yum install -y mysql
[opc@web1 ~]$ mysql -u app -h db1 app -p
Enter password:

…

MySQL [app]> describe first;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)

MySQL [app]>
```



web1

mysql

10.0.0.5

10.0.0.6

db1

port	service
22	sshd
3306	mysqld

Summary

- We’ve permitted connectivity from the VM *web1* to the VM *db1.*

- We’ve demonstrated that we can talk to the database service from the host where we’ll be running our application.

  (So, our application should be able to talk to it also.)

Deployment plan: install the Java application

- We’ll build the application somewhere else, and copy the resulting *jar* file onto the VM.

- We’ll need to install a JRE to run it.

- We’ll begin by launching the application directly from the command-line, then look at how we can get it to automatically restart (like mysql does)

Build and copy the .jar file

- As a pre-requisite to this stage, you should have built the application locally.

- Copy the jar file up to *web1*:

```
% ./gradlew build
% scp build/libs/uob-todo-app-0.1.0.jar opc@129.213.119.230:
uob-todo-app-0.1.0.jar                                              100%   33MB
588.2KB/s   00:58
```

Install a JRE on *web1*

- Another yum invocation:

```
[opc@web1 ~]$ sudo yum install -y java-1.8.0-openjdk-headless
…
Complete!
[opc@web1 ~]$ java -version
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (build 1.8.0_181-b13)
OpenJDK 64-Bit Server VM (build 25.181-b13, mixed mode)
[opc@web1 ~]$
```

Try running the Java app direct from the command-line

- We can launch the application directly - although it'll only run until we press *Control-C* or close the ssh session.
  (The \ at the end of a line tells the shell you've not finished typing yet)

```
[opc@web1 ~]$ java \
>     -Dspring.datasource.url=jdbc:mysql://db1:3306/app \
>     -Dspring.datasource.username=app \
>     -Dspring.datasource.password='DxIHXE%6d7sD:EXI' \
>     -jar uob-todo-app-0.1.0.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.3.RELEASE)
```

… a lot of output elided here …

```
2018-09-28 16:36:38.271  INFO 4980 --- [           main] uob_todo.Application
: Started Application in 12.263 seconds (JVM running for 13.23)
```

Check!!

- use *curl* to talk to the application locally.
  In a *second* session to *web1*:

```
[opc@web1 ~]$ sudo yum install -y curl
…
[opc@web1 ~]$ curl http://localhost:8080
<!doctype html>
<html>
<head>
    <title>My page</title>
    <link rel="stylesheet" href="/styles.css">
    <link rel="stylesheet" href="/main.css">
</head>
<body>
<div id="app"></div>
<script src="/js/app.js"></script>
</body>
</html>
[opc@web1 ~]$
```

Check the database

- Open a session to *db1* and attach to mysql as the *app* user, as before -

```
mysql> show tables;
+--------------------+
| Tables_in_app      |
+--------------------+
| first              |
| hibernate_sequence |
| todo_item          |
+--------------------+
3 rows in set (0.00 sec)

mysql> describe todo_item;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | bigint(20)   | NO   | PRI | NULL    |       |
| completed | bit(1)       | YES  |     | NULL    |       |
| title     | varchar(255) | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Set the Java app up to run as a daemon

- We’ll use a *systemd* unit file
- We’ll read the password from a file
- We’ll run as the *opc* user
- The process still listens on port 8080

- Put the password into a file

```
[opc@web1 ~]$ cat <<‘EOF’ > ~/app.password
> APP_PASSWORD=DxIHXE%6d7sD:EXI
> EOF
```

- Construct a systemd unit file

```
[opc@web1 ~]$ cat <<'EOF' | sudo tee /etc/systemd/system/app.service
> [Unit]
> Description=Sample Java application
> After=network.service
>
> [Service]
> Type=simple
> EnvironmentFile=/home/opc/app.password
> ExecStart=/usr/bin/java \
>   -Dspring.datasource.url=jdbc:mysql://db1:3306/app \
>   -Dspring.datasource.username=app \
>   -Dspring.datasource.password=${APP_PASSWORD} \
>   -jar /home/opc/uob-todo-app-0.1.0.jar
> Restart=never
> StandardOutput=journal
> StandardError=journal
> TimeoutStartSec=300
> User=opc
> Group=opc
>
> [Install]
> WantedBy=multi-user.target
> EOF
```

- Ensure the unit activates on reboot, then start it

```
[opc@web1 ~]$ sudo systemctl daemon-reload
[opc@web1 ~]$ sudo systemctl enable app
[opc@web1 ~]$ sudo systemctl start app
```

- Check its status!

```
[opc@web1 ~]$ systemctl status app
● app.service - Sample Java application
   Loaded: loaded (/etc/systemd/system/app.service; enabled; vendor preset: 
disabled)
   Active: active (running) since Fri 2018-09-28 17:18:00 GMT; 14s ago
 Main PID: 6420 (java)
   CGroup: /system.slice/app.service
           └─6420 /usr/bin/java -Dspring.datasource.url=jdbc:mysql://db1:3306/ap...

[           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat st...ext path ''
Sep 28 17:18:13 web1 java[6420]: 2018-09-28 17:18:13.971  INFO 6420 ---
[           main] uob_todo.Application                     : Started A...for 13.014)
Hint: Some lines were ellipsized, use -l to show in full.
[opc@web1 ~]$
```

- Check again using *curl*

Permit inbound traffic to the HTTP port

- Two-stage process: add a firewall rule on *web1* permitting that traffic

- Add a security rule to port 8080 from 0.0.0.0/0 (all source ports)

- Check! - Can you point your browser at that IP address?

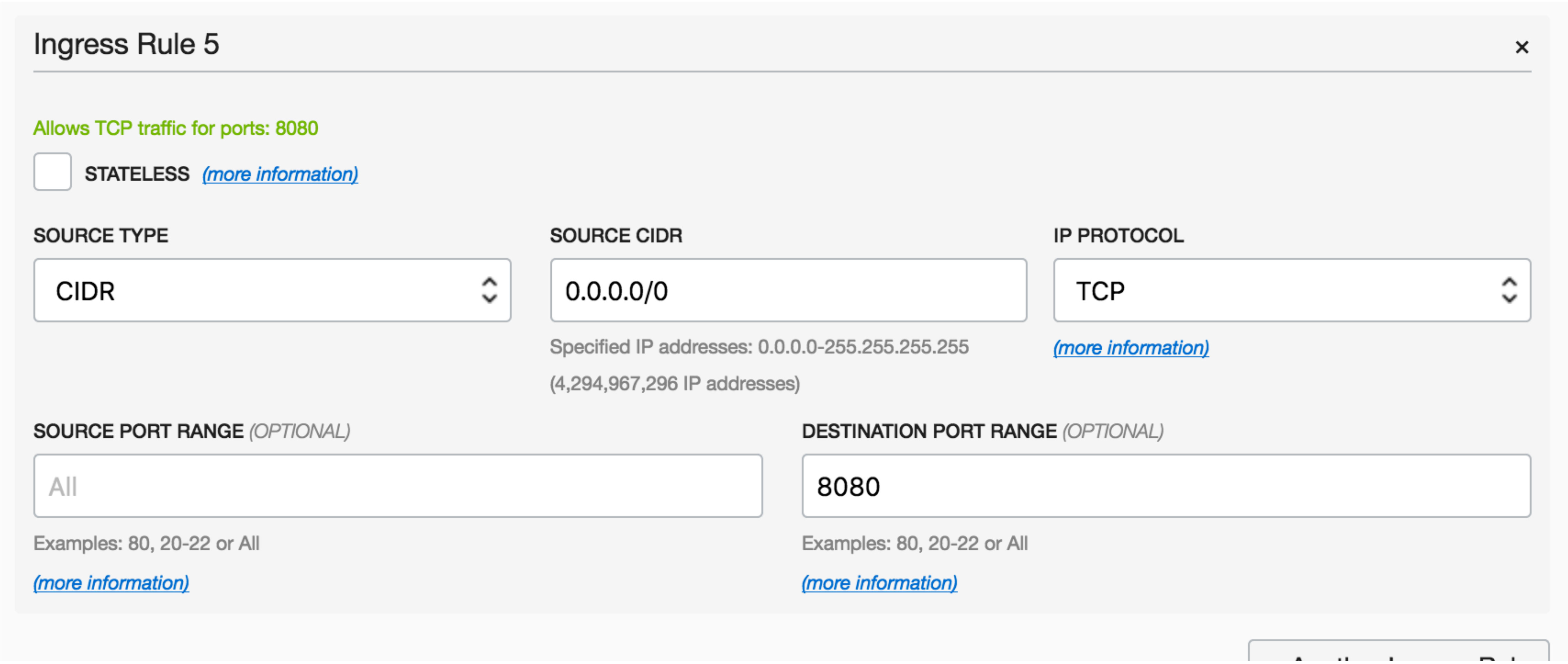
```
[opc@web1 ~]$ sudo firewall-cmd --add-port 8080/tcp
success
[opc@web1 ~]$ sudo firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client
  ports: 8080/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

[opc@web1 ~]$ sudo firewall-cmd --runtime-to-permanent
success
```

Adding the security rule

- “Menu” / Networking > Virtual Cloud Networks
- Select “net1”
- Select “Default security list for net1”
- “Edit all rules”
- Add an ***ingress*** rule as follows…

Adding the security rule



Ingress Rule 5

Allows TCP traffic for ports: 8080

□ STATELESS *(more information)*

SOURCE TYPE: CIDR

SOURCE CIDR: 0.0.0.0/0

Specified IP addresses: 0.0.0.0-255.255.255.255 (4,294,967,296 IP addresses)

IP PROTOCOL: TCP

(more information)

SOURCE PORT RANGE *(OPTIONAL)*: All

Examples: 80, 20-22 or All

(more information)

DESTINATION PORT RANGE *(OPTIONAL)*: 8080

Examples: 80, 20-22 or All

(more information)

(Don’t forget to save the change)

Check!!

- From a laptop / desktop:

```
% curl http://129.213.119.230:8080
<!doctype html>
<html>
<head>
    <title>My page</title>
    <link rel="stylesheet" href="/styles.css">
    <link rel="stylesheet" href="/main.css">
</head>
<body>
<div id="app"></div>
<script src="/js/app.js"></script>
</body>
</html>
```

- Try pointing a browser at http://129.213.119.230:8080

Troubleshooting:

- “It doesn’t work!”
  - Is the process started? Is it still running? *(systemctl status* or *ps)*
    - Did it crash? Check logs (look in */var/log* or use *journalctl*)
  - Is it listening? *(netstat or lsof)*
    - Can I talk to it locally? (*nc* or a specific protocol client, like *mysql* or *curl*)
    - Can I talk to it from another VM?
      - Check host firewall
      - Check security rules
    - Can I talk to it from across the internet? (Should I be able to?)
  - Can it talk to its dependencies?
    - This is application-specific. Are any embedded credentials correct? Can you make a connection to the same service from the same VM?

Locating the application on the net

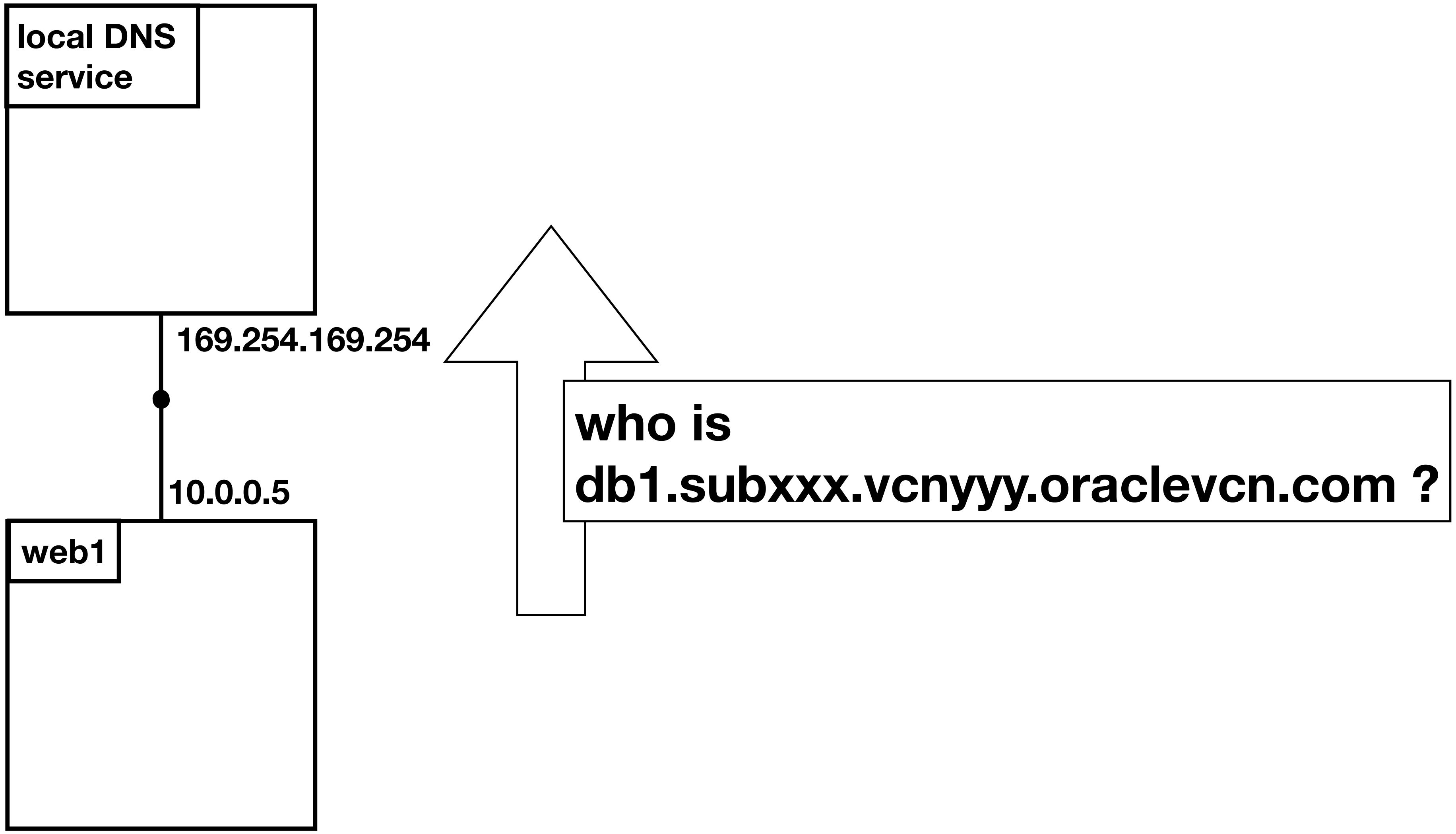
IP addresses aren’t very friendly.

We mentioned the use of an internal DNS to let VMs locate each other by name.

The world at large uses DNS to do the same thing.

A typical DNS request

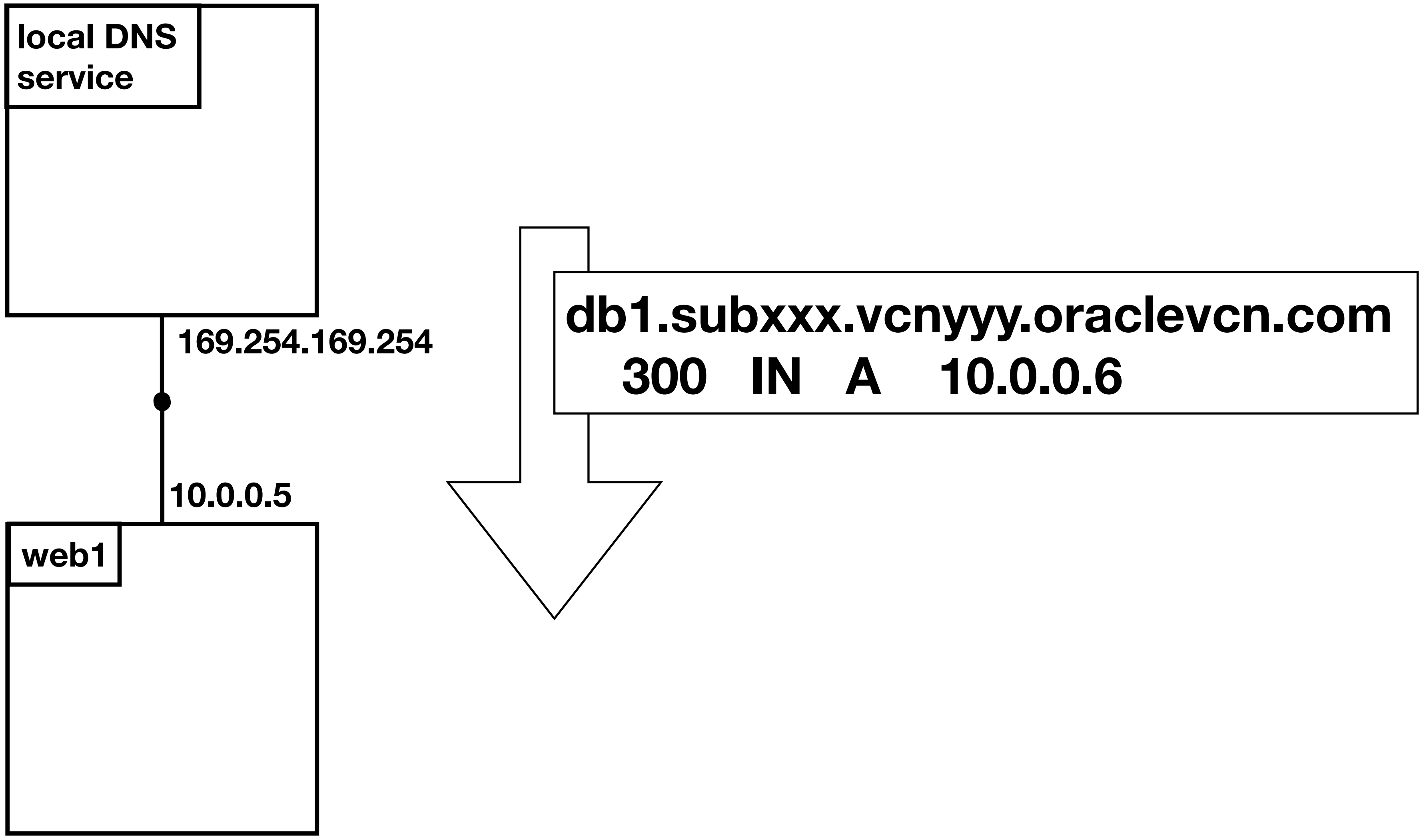
(local names)



who is
db1.subxxx.vcnyyy.oraclevcn.com ?

A typical DNS request

(local names)



local DNS service

169.254.169.254

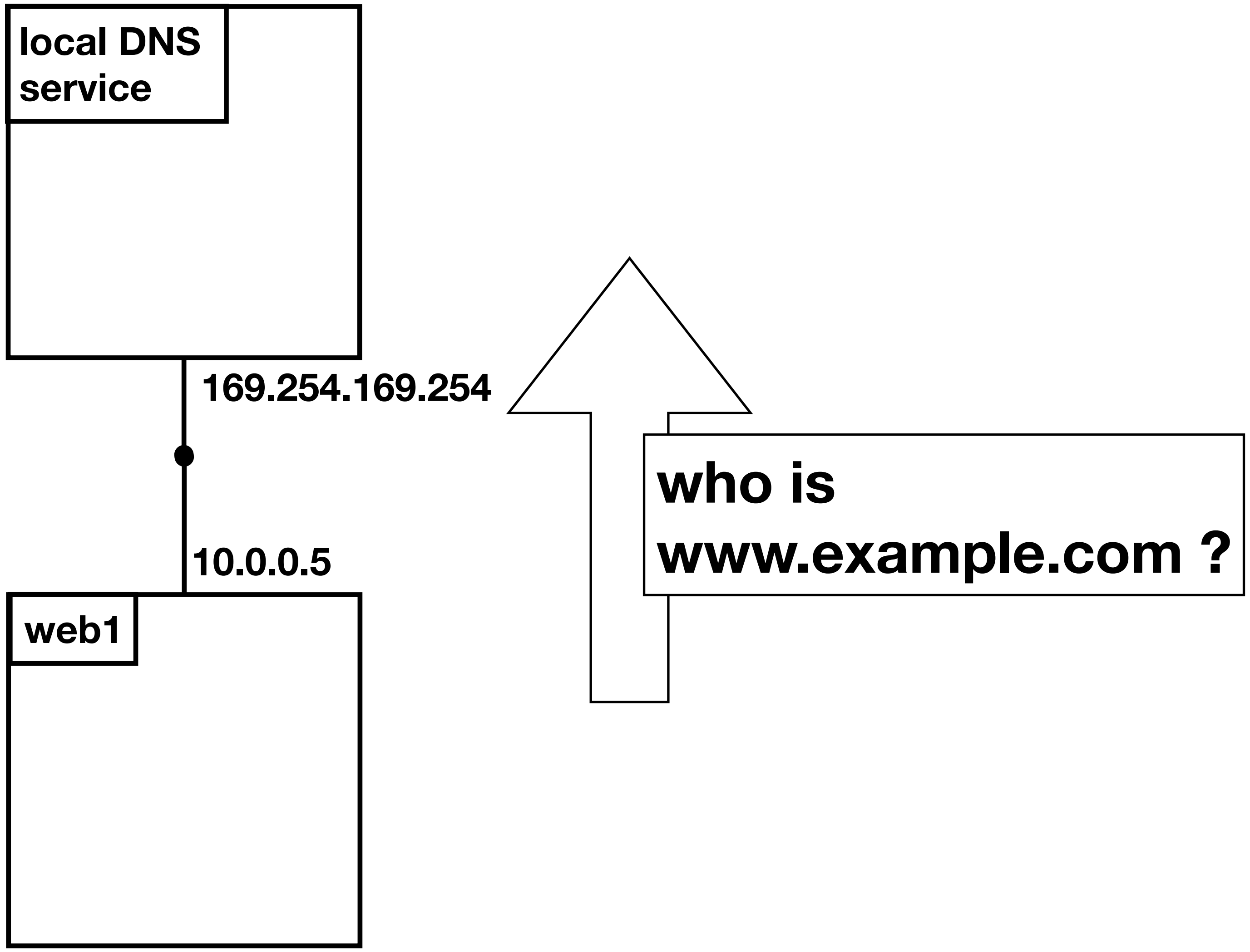
10.0.0.5

web1

```
db1.subxxx.vcnyyy.oraclevcn.com
   300   IN   A    10.0.0.6
```

A typical DNS request

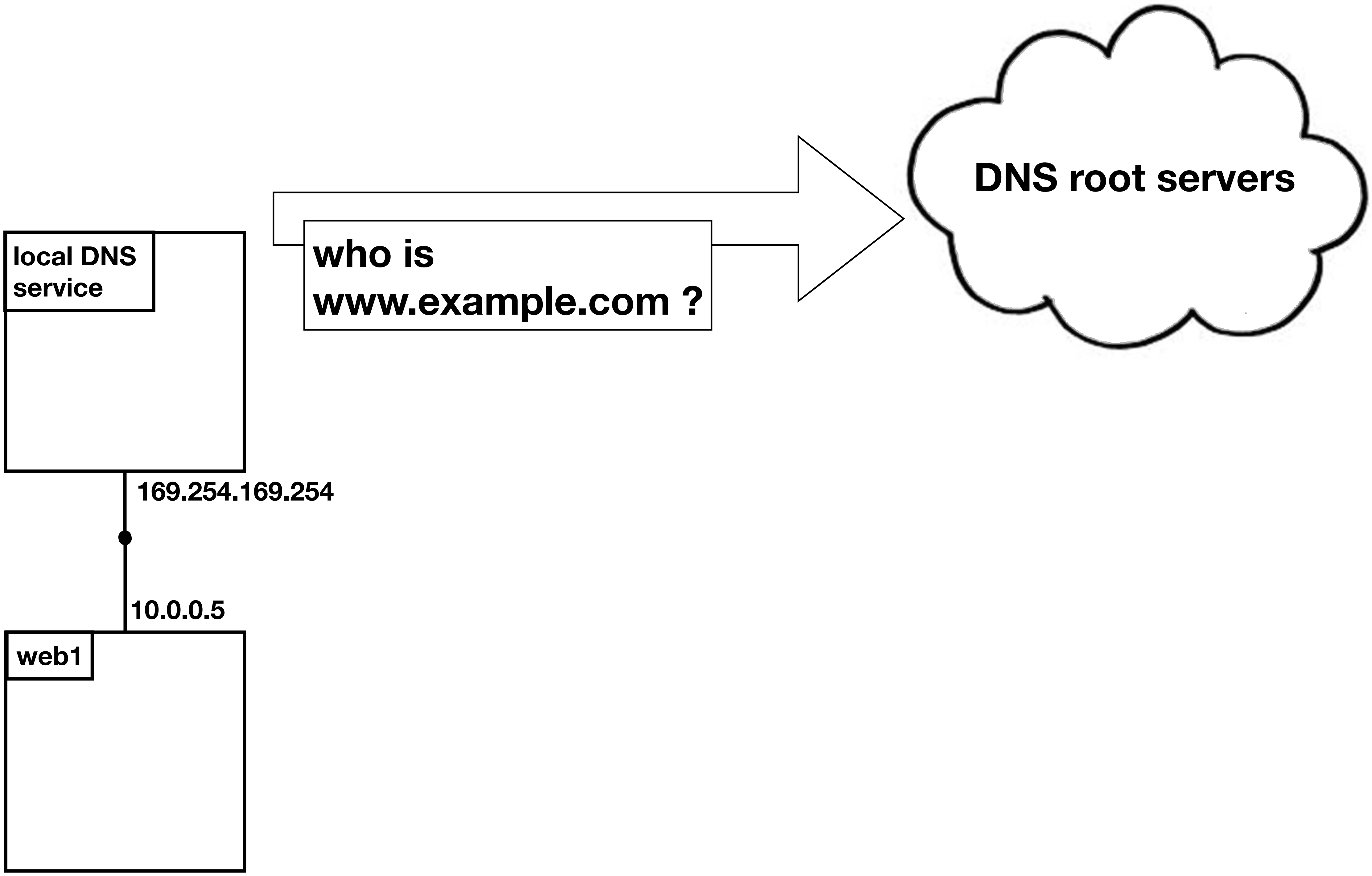
(global names)



who is www.example.com ?

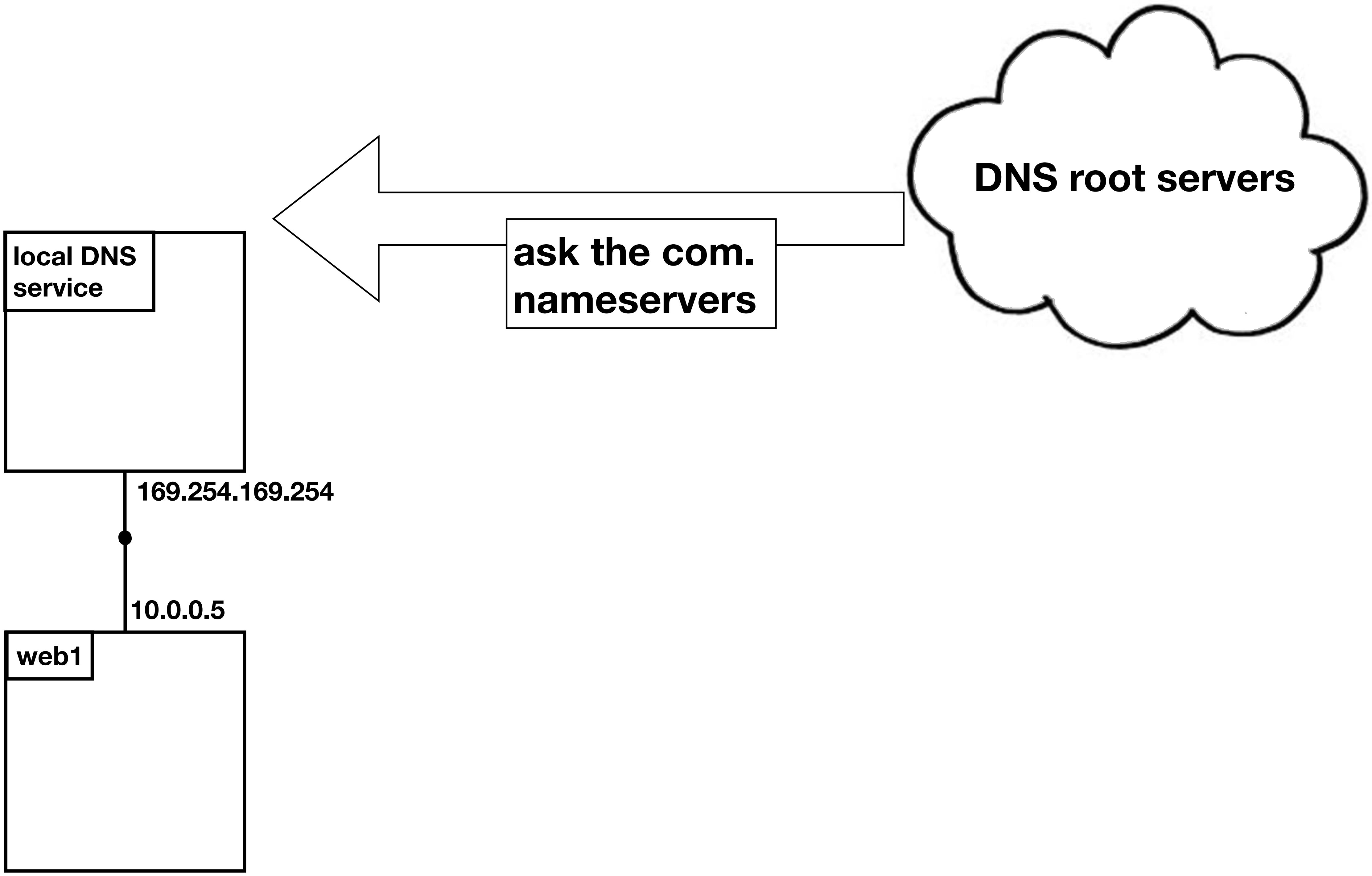
A typical DNS request

(global names)



A typical DNS request

(global names)



DNS root servers

ask the com. nameservers

local DNS service

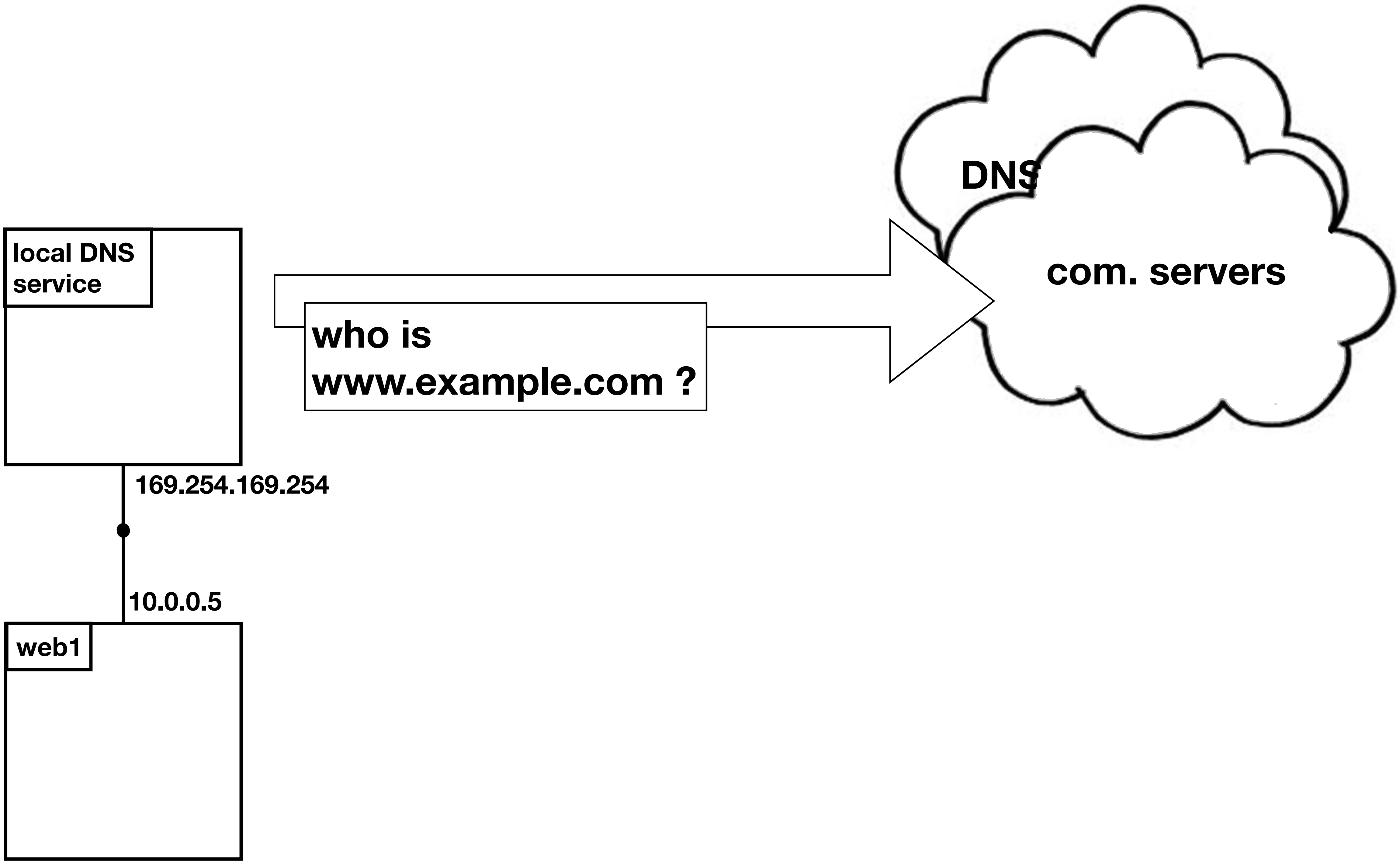
169.254.169.254

10.0.0.5

web1

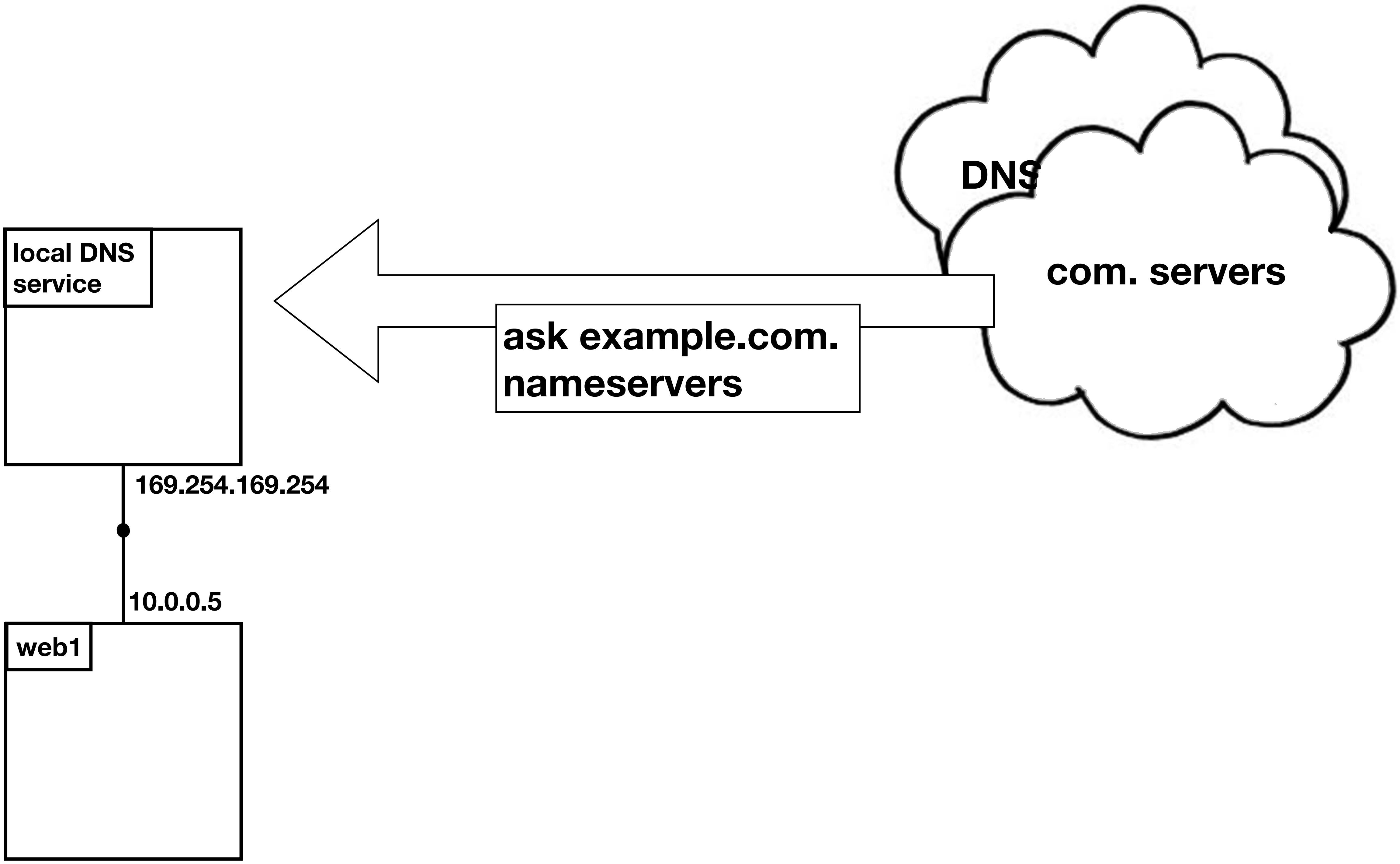
A typical DNS request

(global names)



A typical DNS request

(global names)

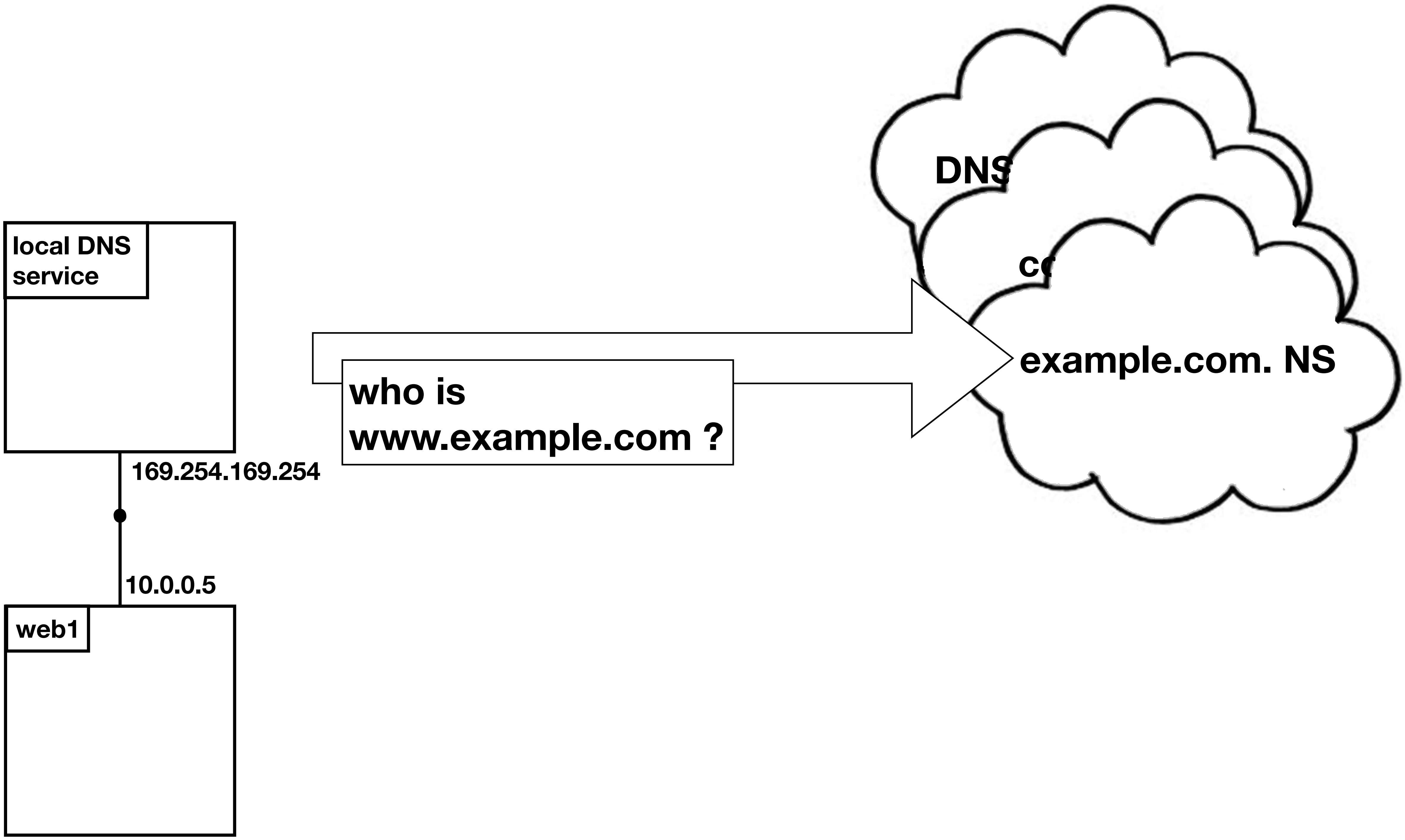


local DNS service

ask example.com. nameservers

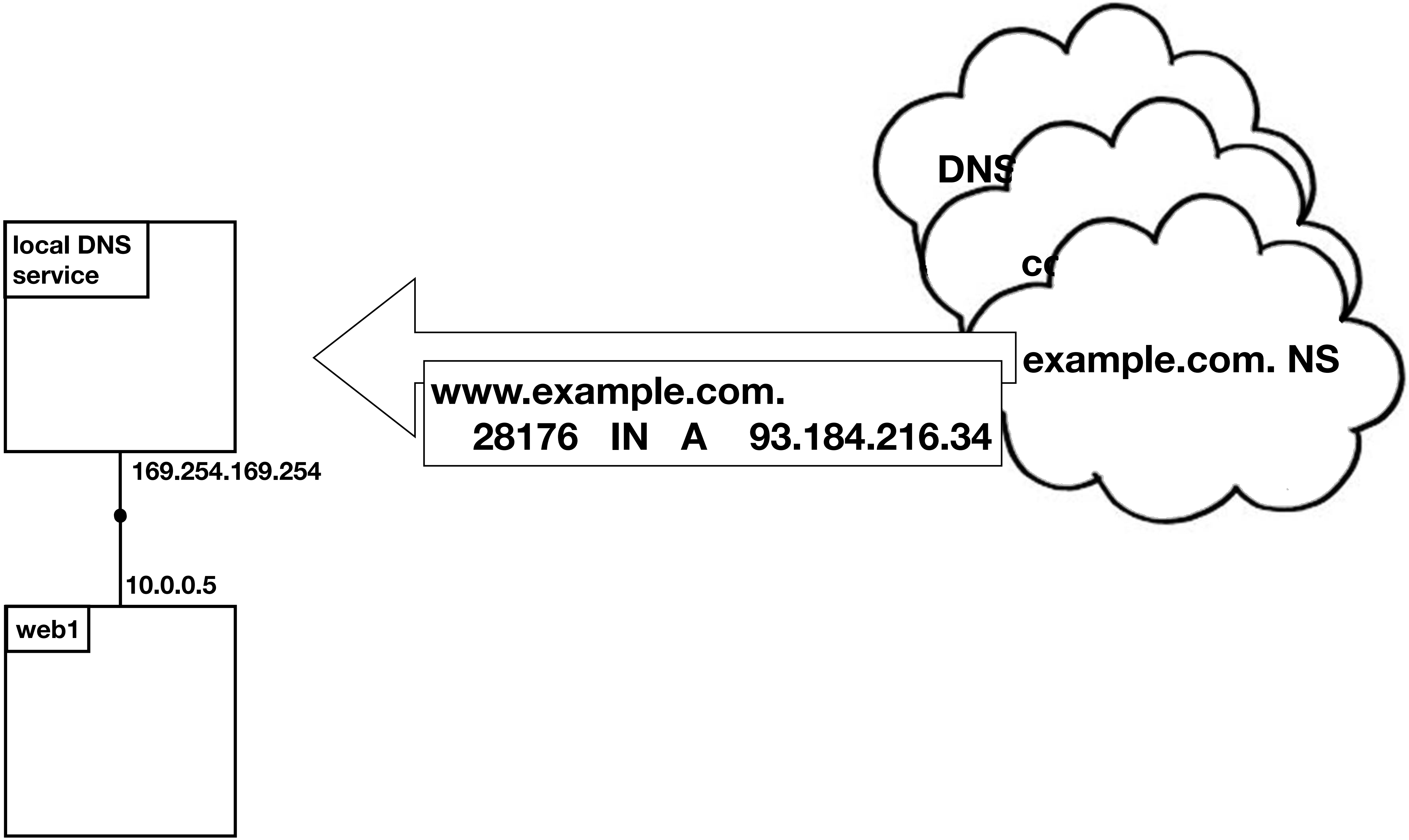
A typical DNS request

(global names)



A typical DNS request

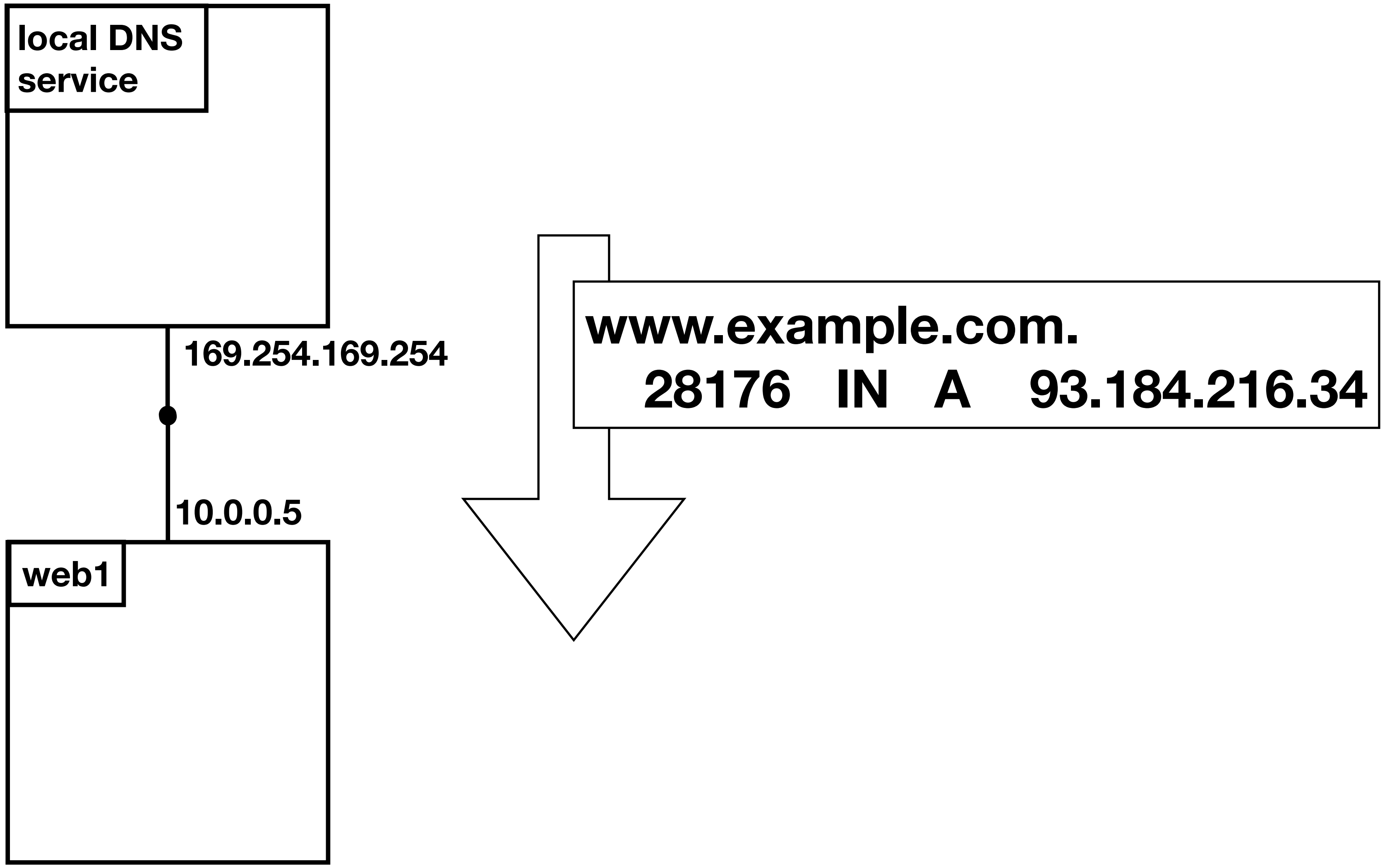
(global names)



```
www.example.com.
  28176   IN   A   93.184.216.34
```

A typical DNS request

(global names)



www.example.com.
  28176   IN   A   93.184.216.34

If you buy a domain:

These are relatively cheap; there are many providers who will also *host the domain* (that is, run a DNS server) for you.

(You won’t need any of the other options such as web hosting, email hosting, etc.; just the ability to edit the *resource records* for the domain.)

You’ll probably need to use a web-console, command-line tool or an API to configure these.

DNS

Suppose you have a domain, *example.com*.

You want to point web traffic to your application, so that a browser, given the link http://example.com/*,* will contact your application.

- look for an "*A record"*
- put the public IP address of the host (or load-balancer front end) as the content of that *A record*.

Updates may not be instant.

(this is an example of something called the ***PACELC theorem*** in practice)

DNS

- The *public IP* of a VM is ephemeral (it lives as long as the VM).
    - It’s possible to pre-allocate an IP address and attach it to the *VNIC* of a VM. This means your IP address remains stable if you recreate your infrastructure.
    - It might be better to target a load-balancer instead.

DNS, checking

How name-resolution works is out-of-scope, but: you can use the *dig* tool to explore.

```
[opc@web1 ~]$ sudo yum install -y bind-utils

[opc@web1 ~]$ dig example.org.

; <<>> DiG 9.9.4-RedHat-9.9.4-61.el7_5.1 <<>> example.org.

…

;; ANSWER SECTION:
example.org.  86400  IN  A  93.184.216.34

…

[opc@web1 ~]$
```

Summary

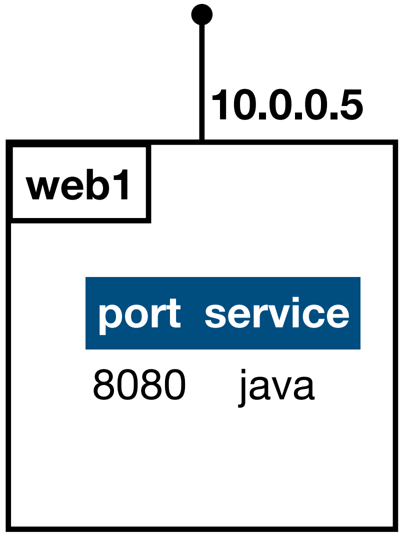
- We’ve installed the application using two VMs
- We can connect to it from the outside world
- We know how to plumb it into DNS if required.

Are we done?

What’s missing here?

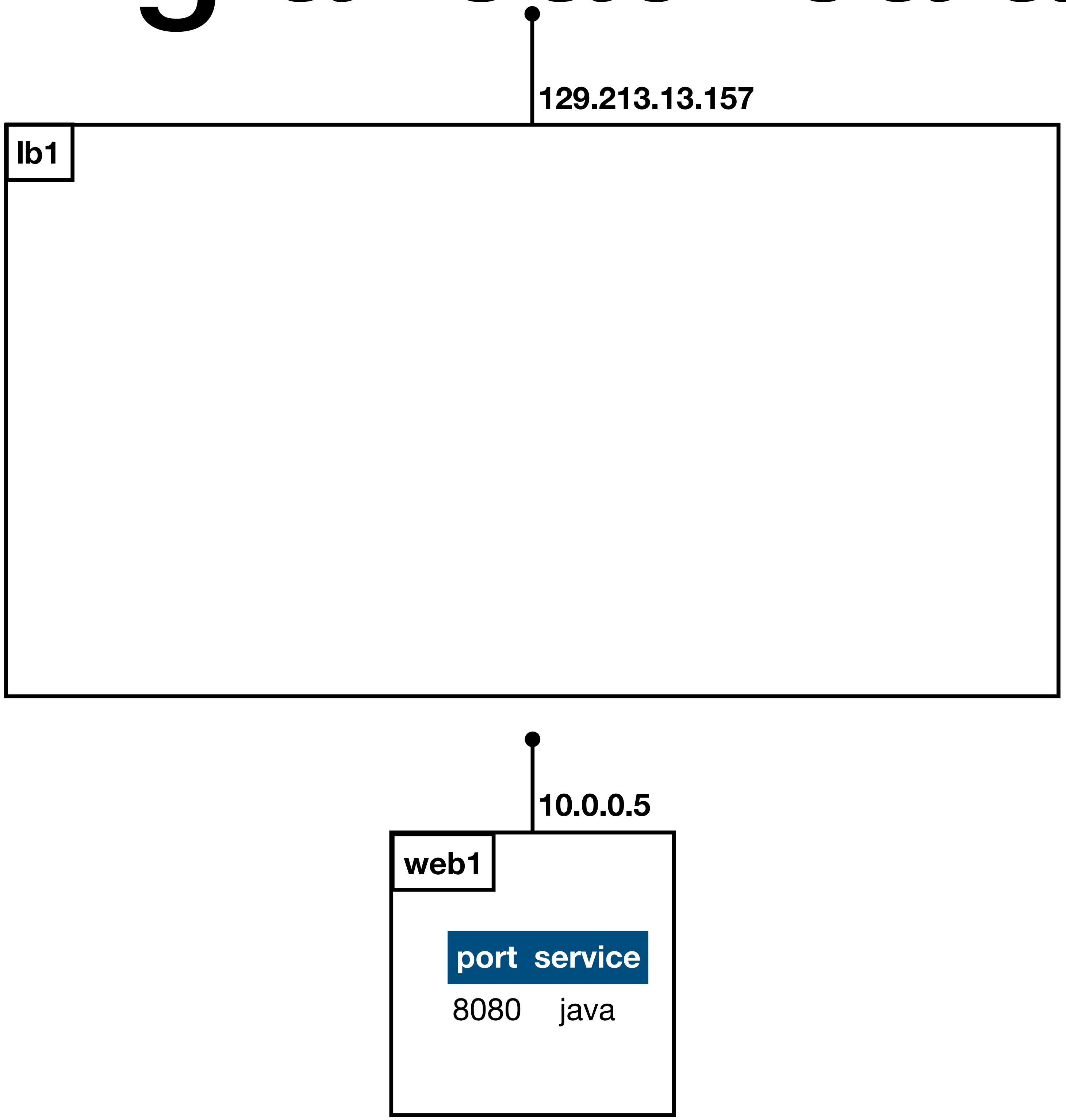
- Scalability! How do we deal with additional load?
    - Scaling the *front end* is a different task to scaling the *back end*
    - There are implications on my application architecture. If multiple requests from the same client might be served by different VMs, how do I ensure that those VMs are in sync?
        - We need to consider *shared state*

Adding a load-balancer



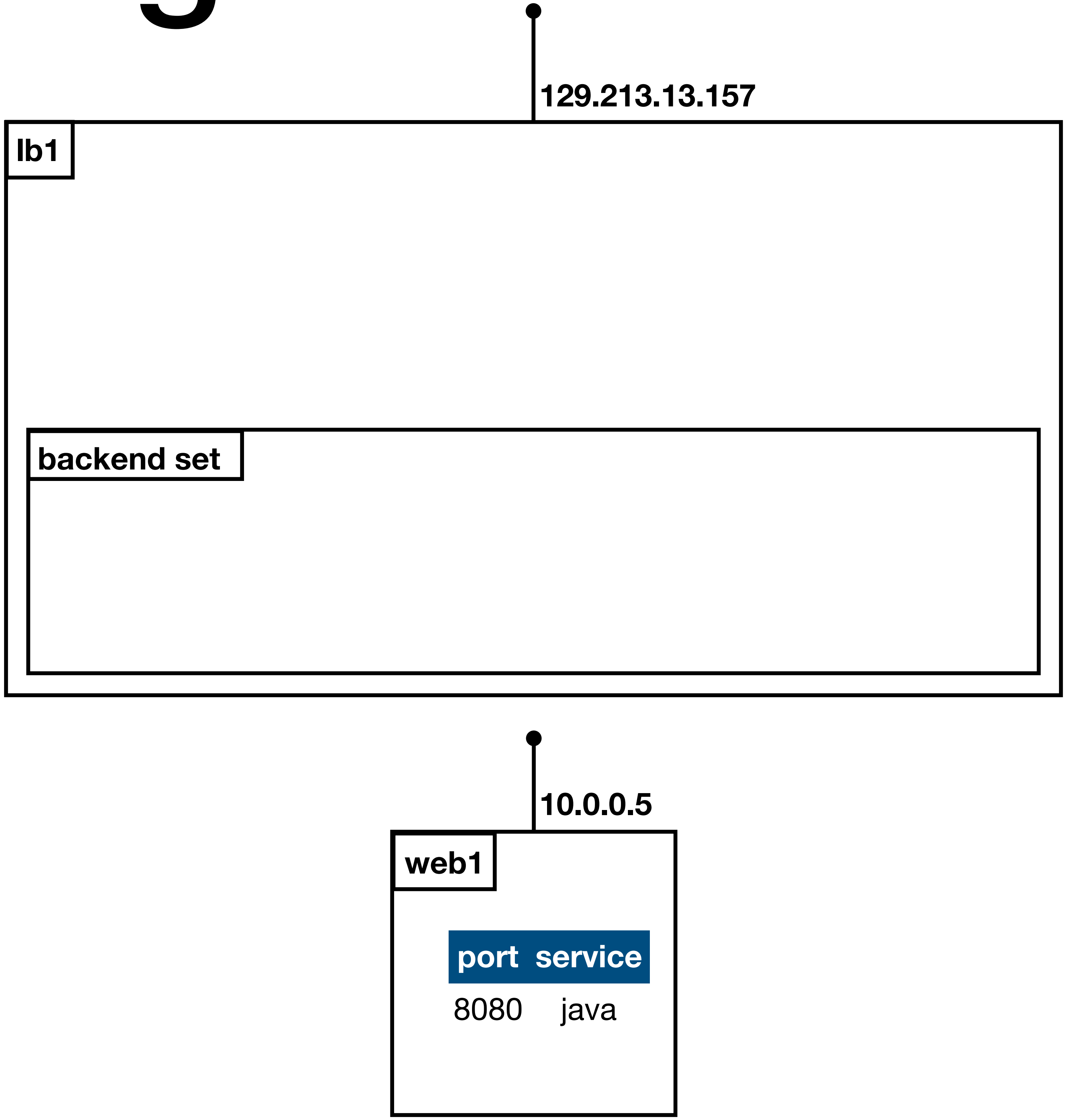
port	service
8080	java

Adding a load-balancer



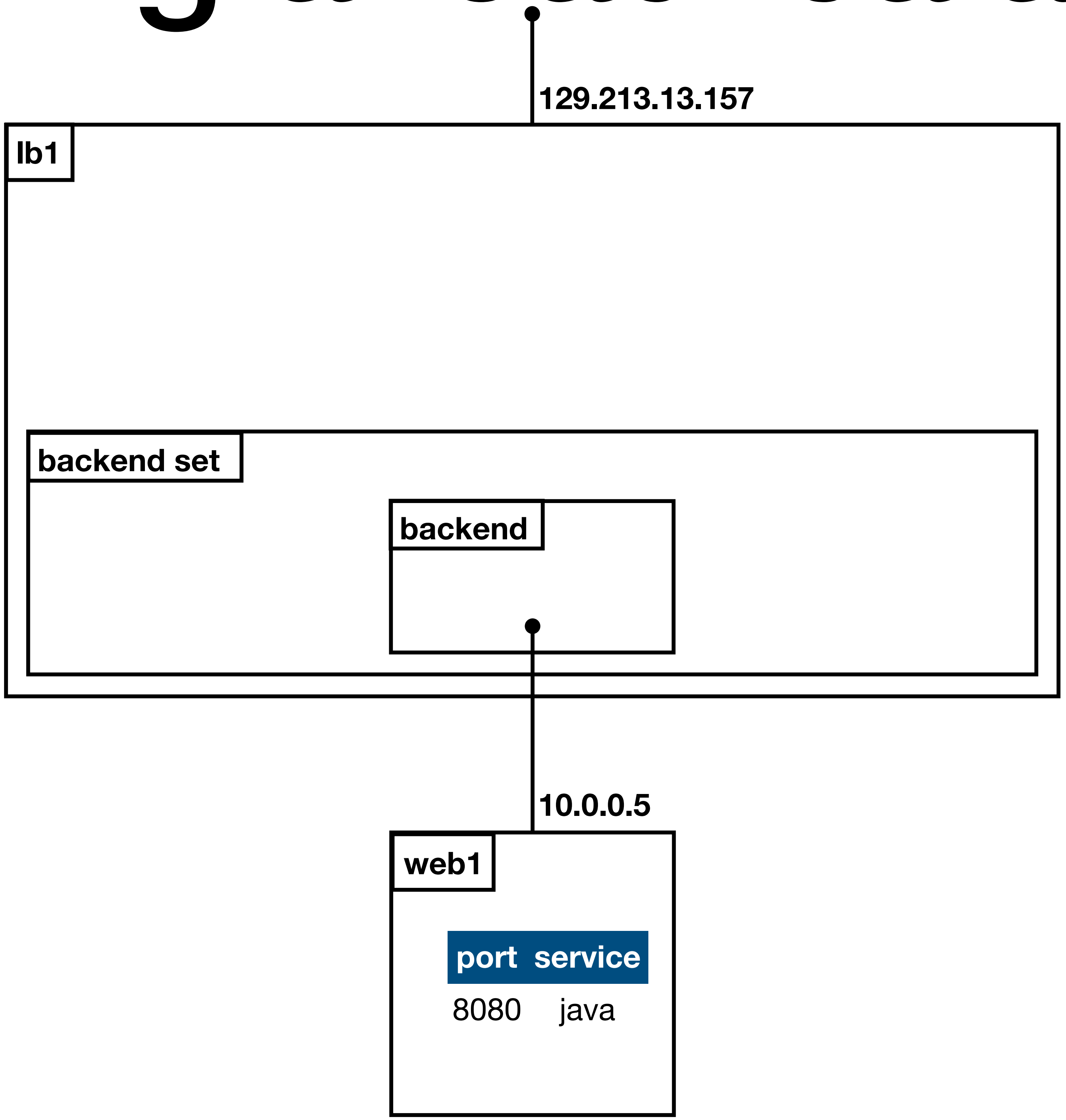
port	service
8080	java

Adding a load-balancer



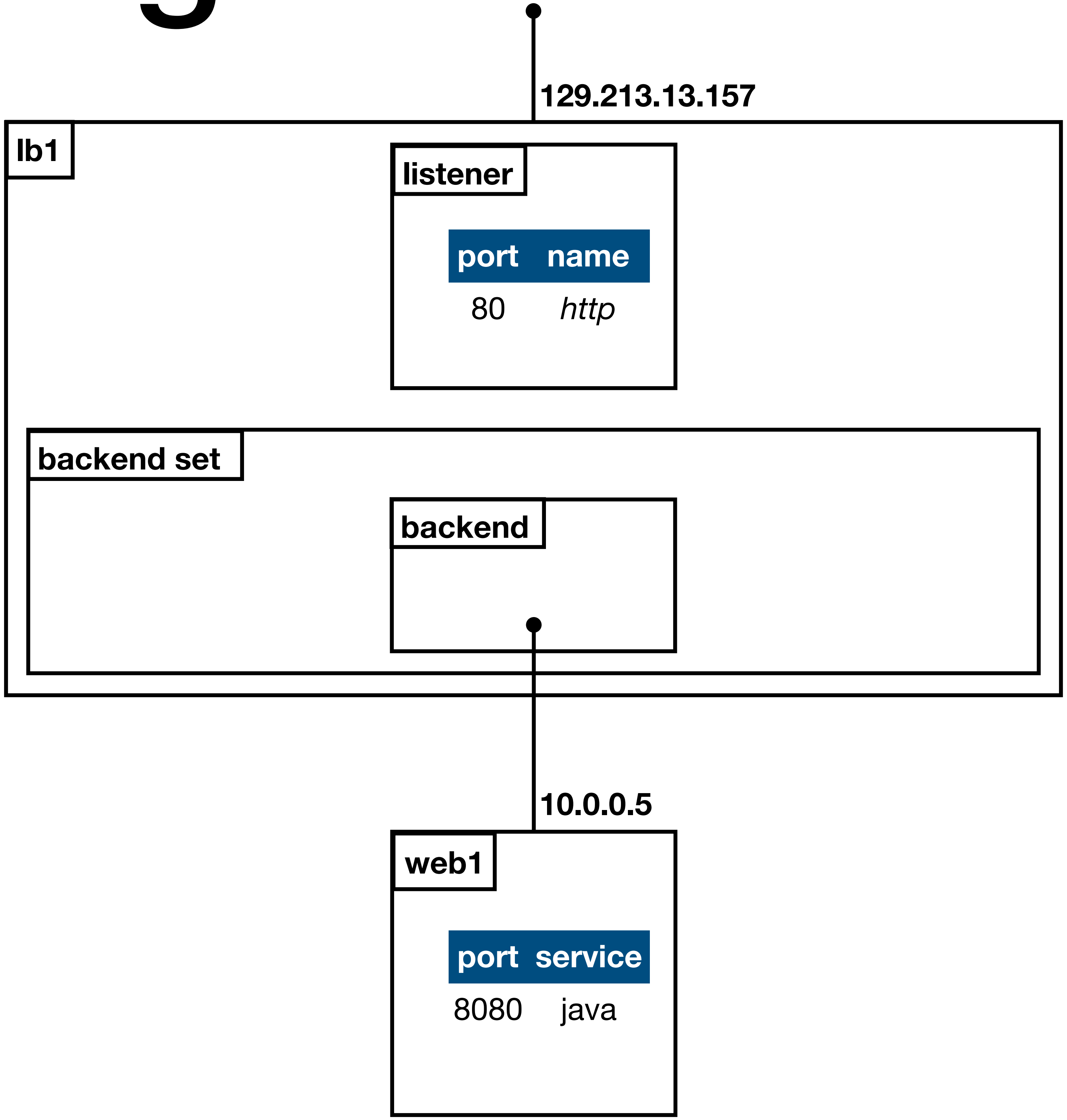
port	service
8080	java

Adding a load-balancer



port	service
8080	java

Adding a load-balancer



129.213.13.157

lb1

listener

port	name
80	*http*

backend set

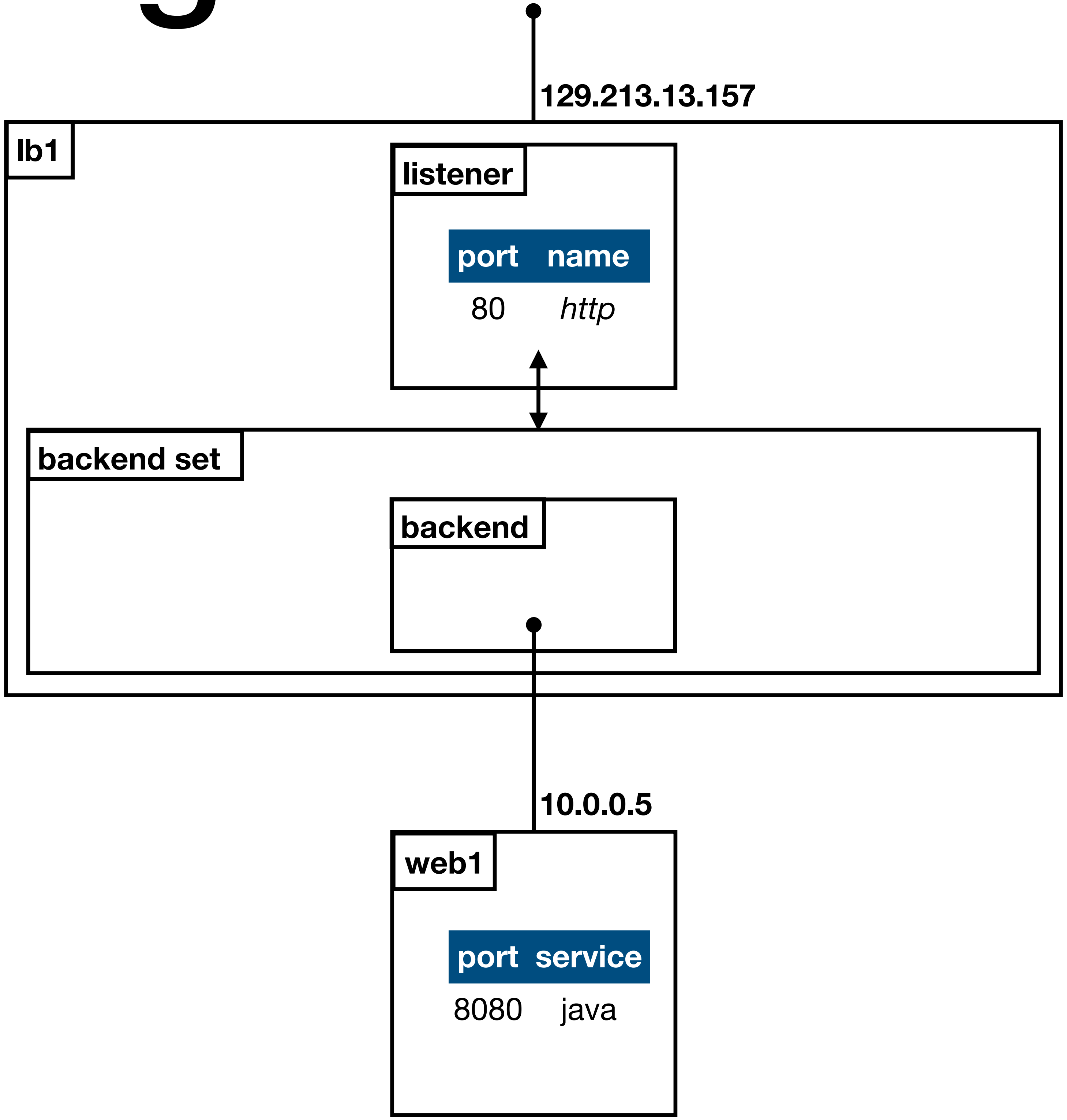
backend

10.0.0.5

web1

port	service
8080	java

Adding a load-balancer



129.213.13.157

lb1

listener

port	name
80	*http*

backend set

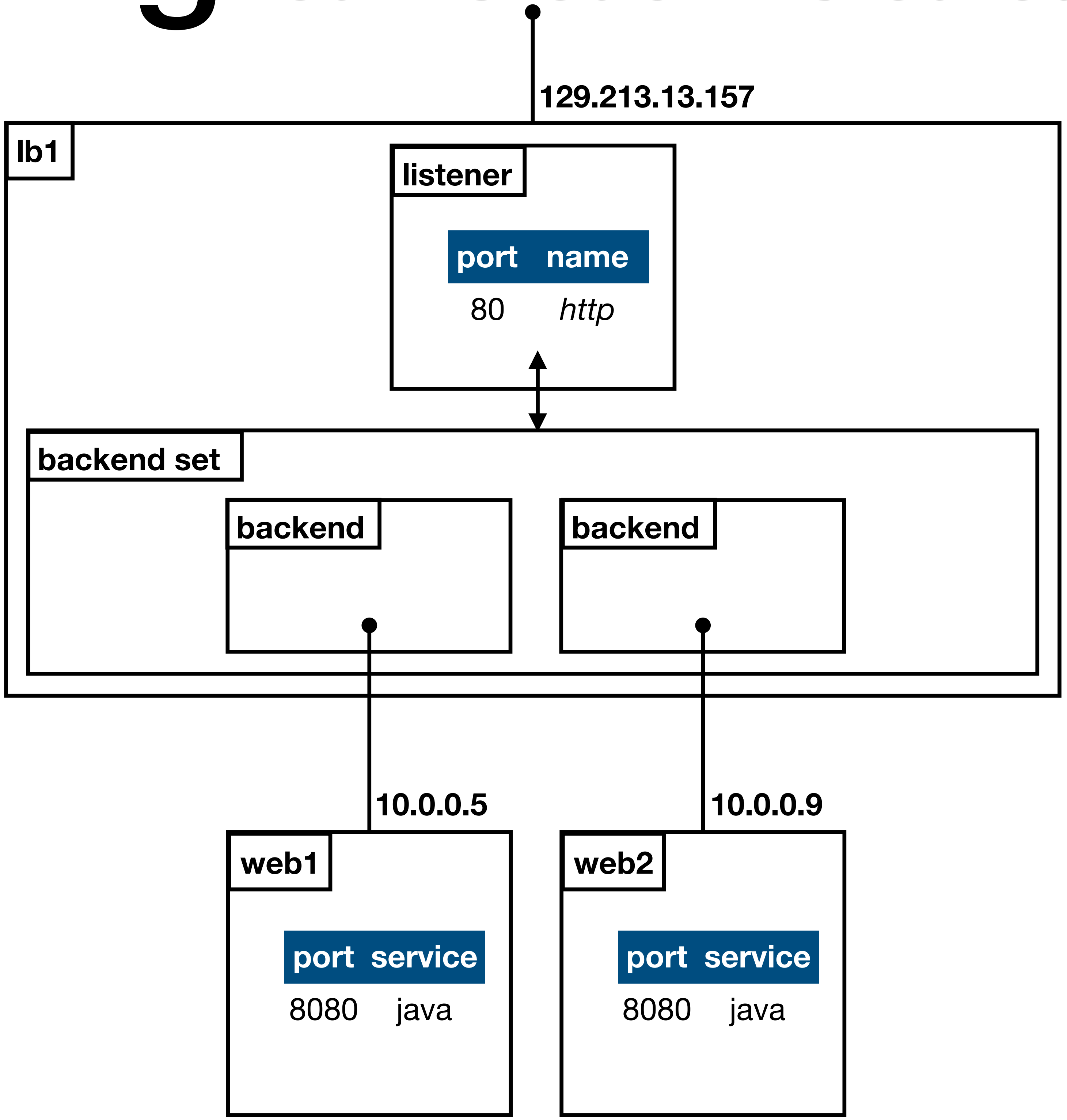
backend

10.0.0.5

web1

port	service
8080	java

Adding a load-balancer

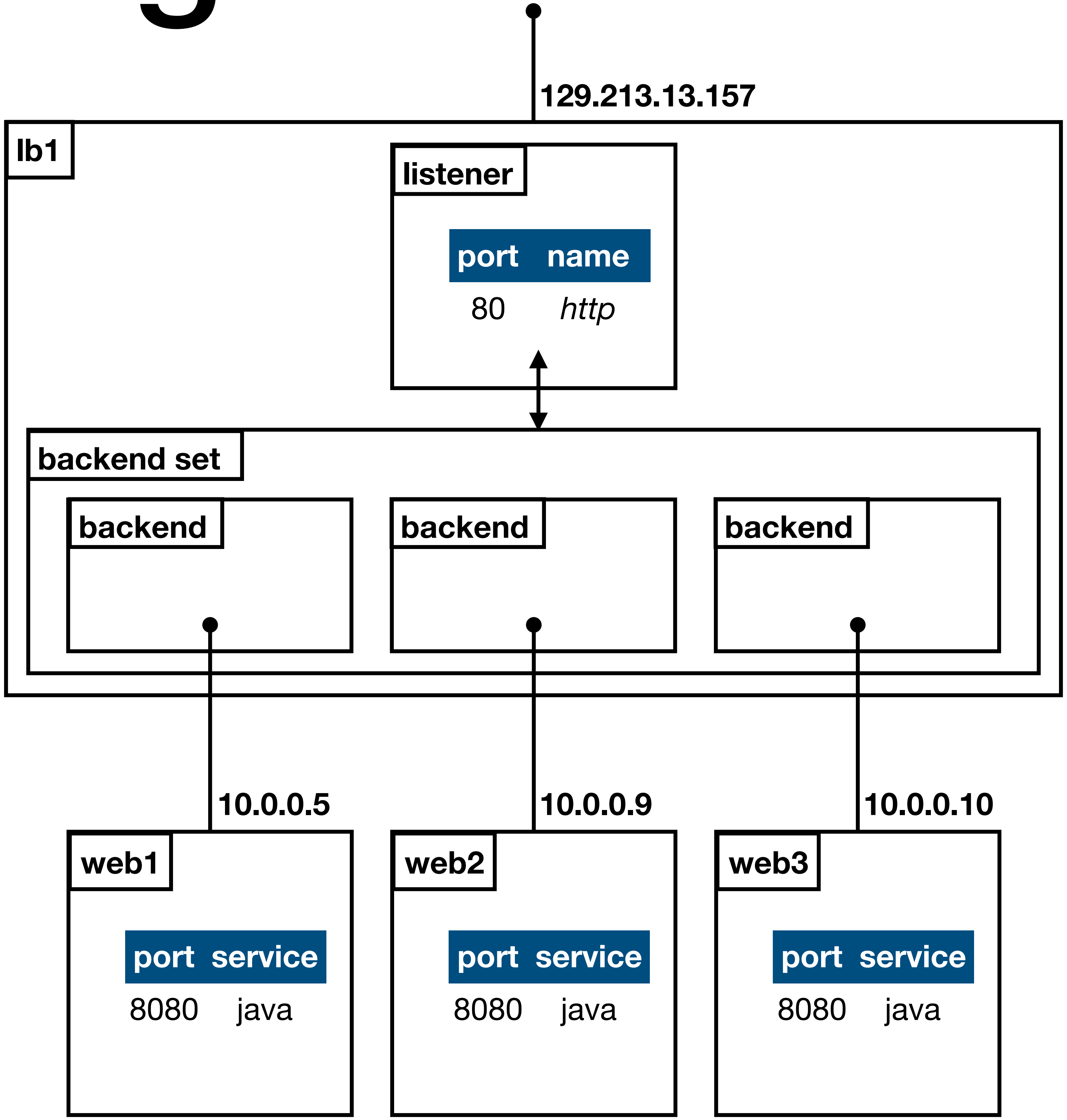


port	name
80	*http*

port	service
8080	java

port	service
8080	java

Adding a load-balancer



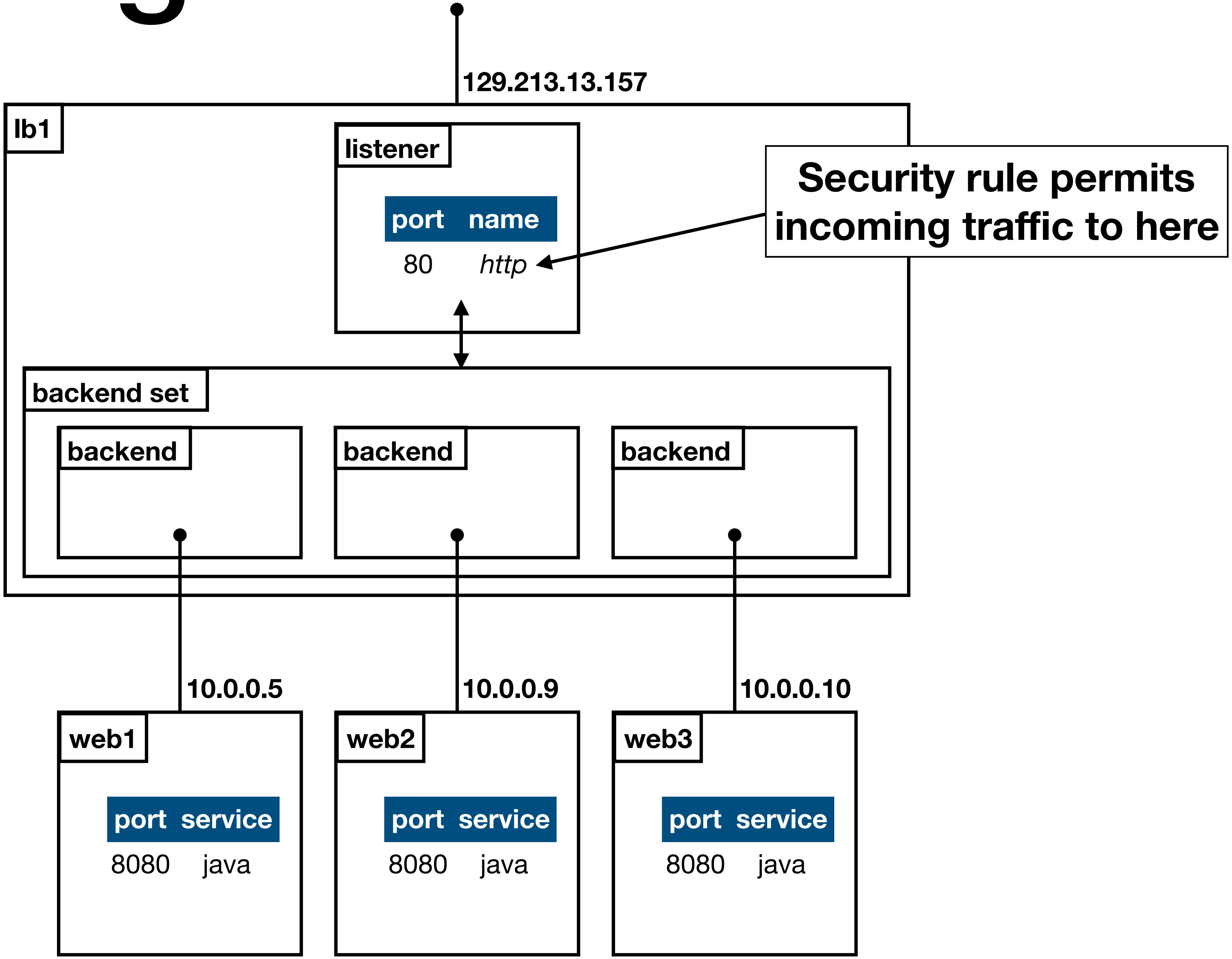
port	name
80	*http*

port	service
8080	java

port	service
8080	java

port	service
8080	java

Adding a load-balancer



Adding a load-balancer through the console

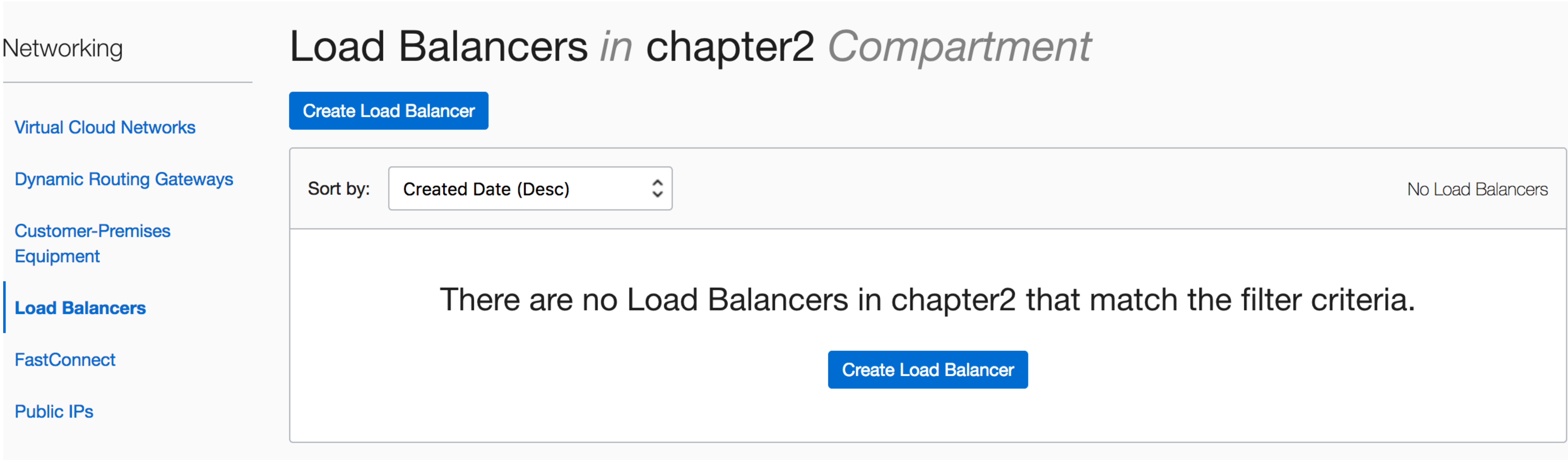
- This is fiddly because the console reflects the structure of the underlying API,

  and the underlying API reflects the different entities just mentioned:

  - load balancer
  - backend set
  - backend
  - listener

- We’ll need to update security rules too,
  although the console automates this somewhat.

- “Menu” / Networking > Load Balancers

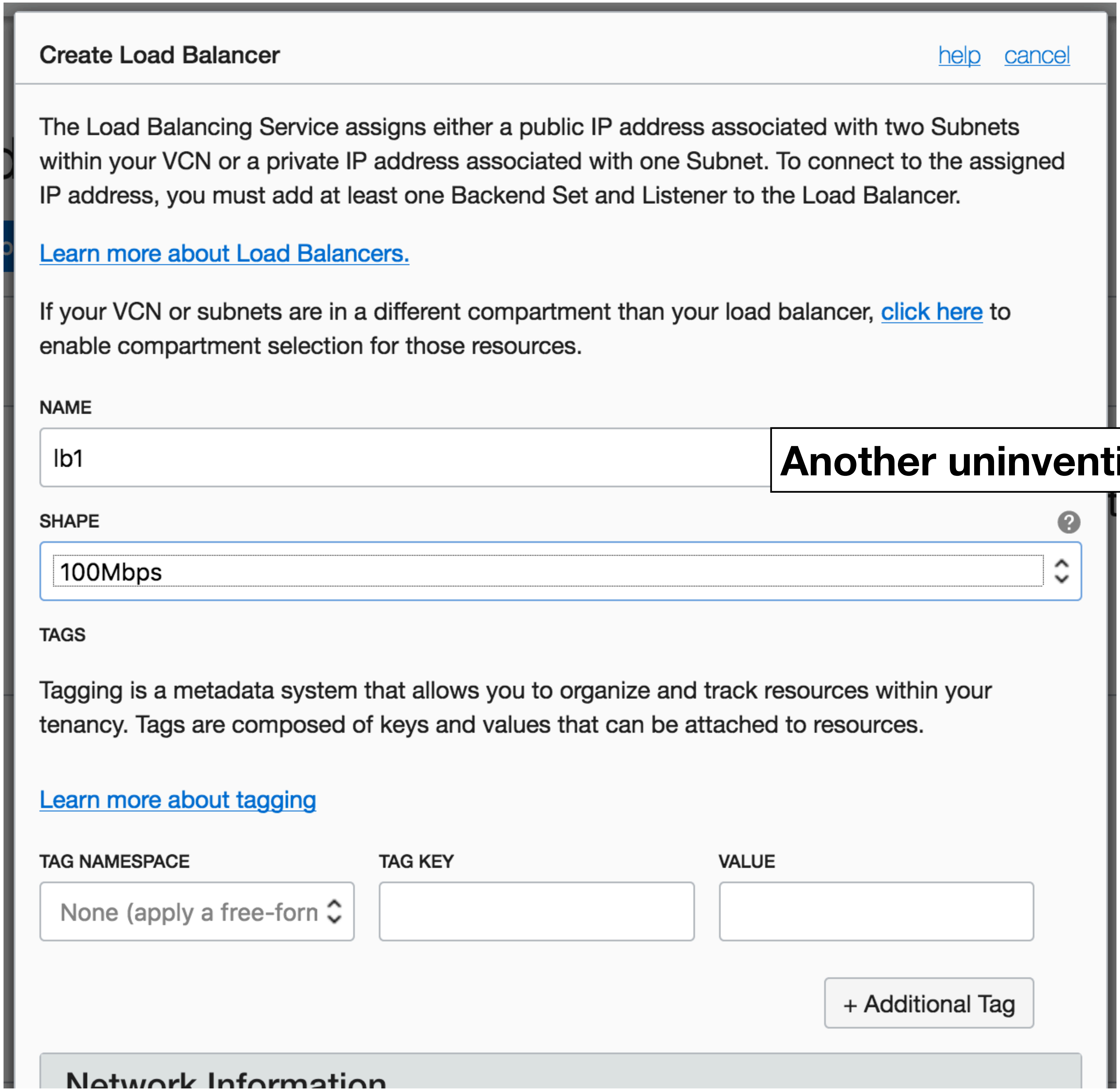


Load Balancers *in* chapter2 *Compartment*

Sort by: Created Date (Desc)

No Load Balancers

There are no Load Balancers in chapter2 that match the filter criteria.



Create Load Balancer

The Load Balancing Service assigns either a public IP address associated with two Subnets within your VCN or a private IP address associated with one Subnet. To connect to the assigned IP address, you must add at least one Backend Set and Listener to the Load Balancer.

Learn more about Load Balancers.

If your VCN or subnets are in a different compartment than your load balancer, click here to enable compartment selection for those resources.

NAME: lb1

SHAPE: 100Mbps

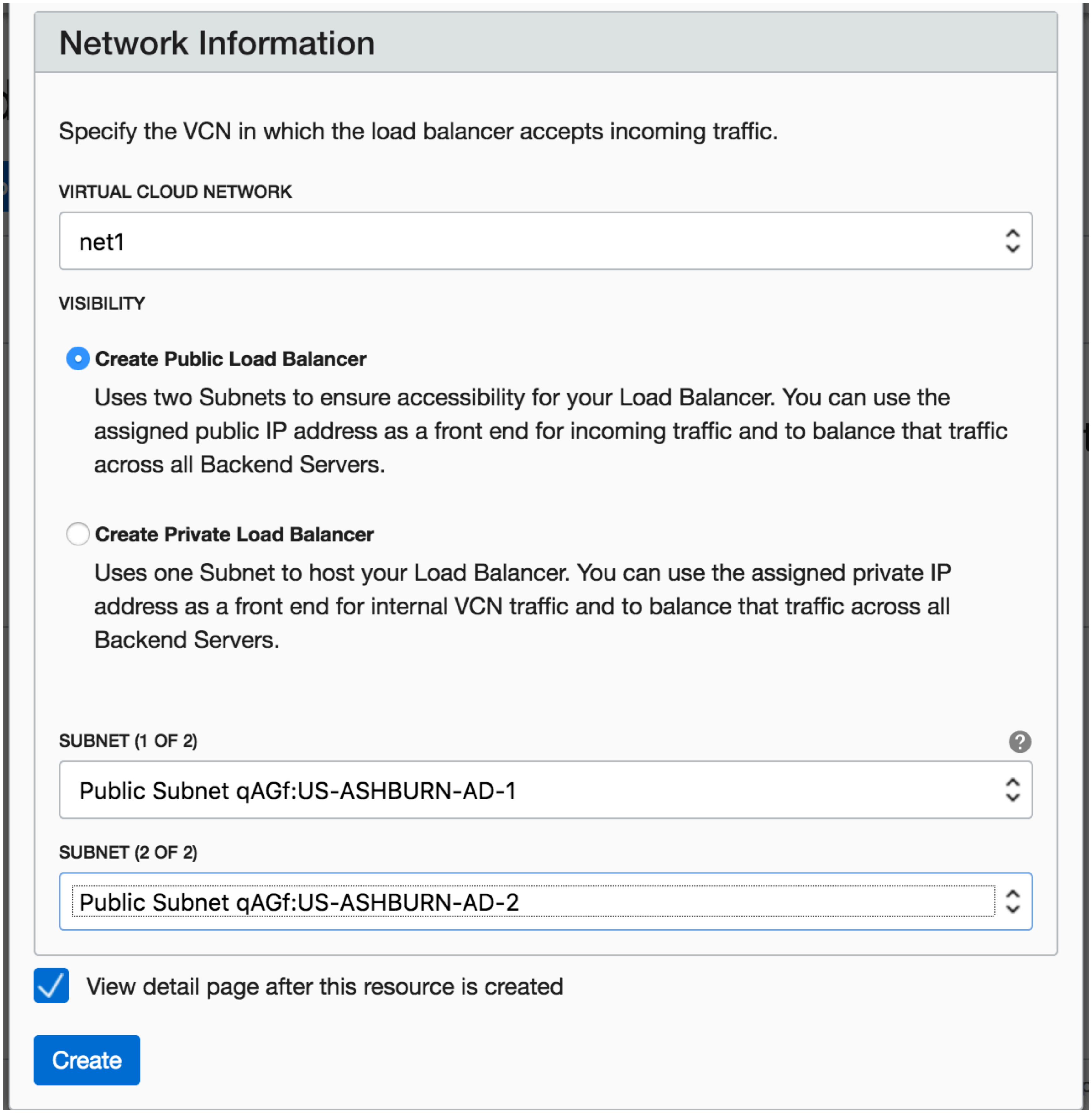
TAGS

Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

Learn more about tagging

TAG NAMESPACE	TAG KEY	VALUE
None (apply a free-forn		

Another uninventive name!



Network Information

Specify the VCN in which the load balancer accepts incoming traffic.

VIRTUAL CLOUD NETWORK

net1

VISIBILITY

● **Create Public Load Balancer**

Uses two Subnets to ensure accessibility for your Load Balancer. You can use the assigned public IP address as a front end for incoming traffic and to balance that traffic across all Backend Servers.

○ **Create Private Load Balancer**

Uses one Subnet to host your Load Balancer. You can use the assigned private IP address as a front end for internal VCN traffic and to balance that traffic across all Backend Servers.

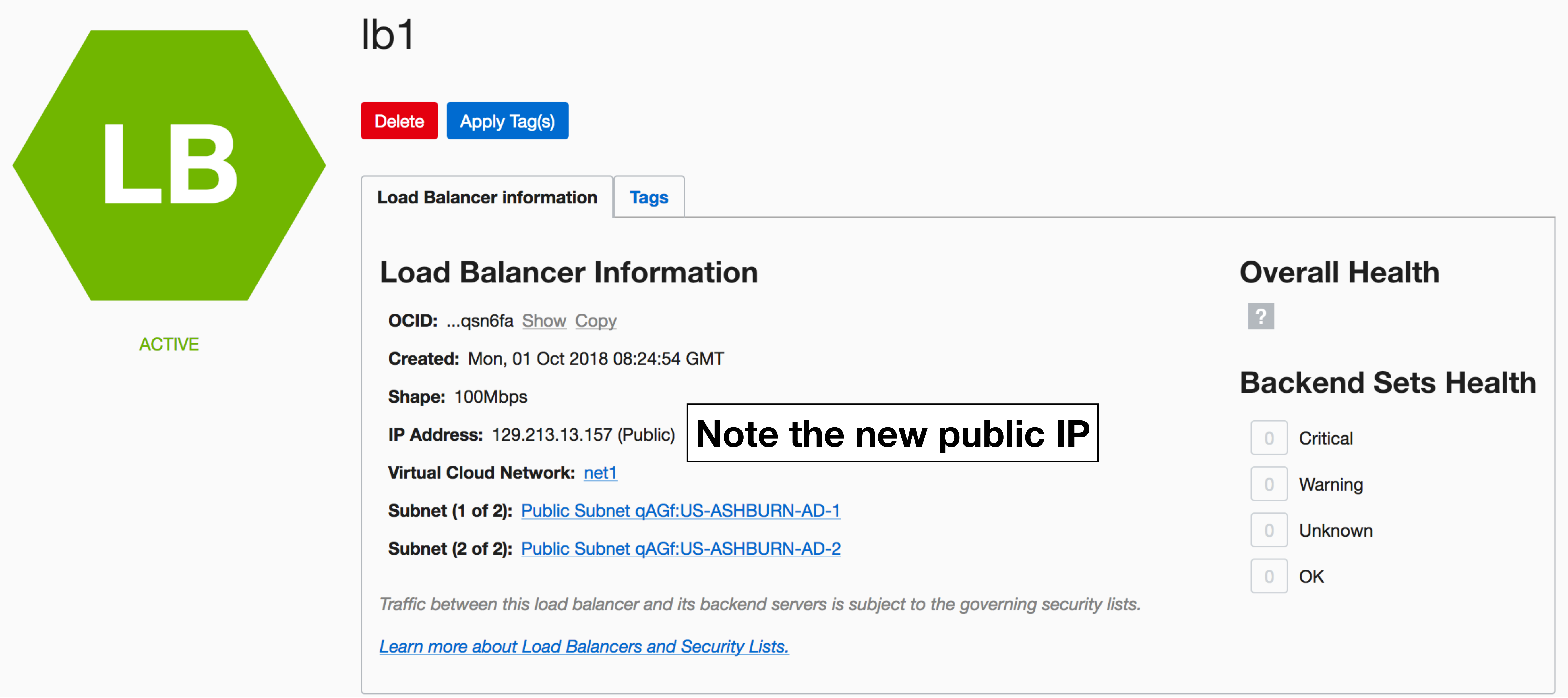
SUBNET (1 OF 2)

Public Subnet qAGf:US-ASHBURN-AD-1

SUBNET (2 OF 2)

Public Subnet qAGf:US-ASHBURN-AD-2

☑ View detail page after this resource is created



lb1

Load Balancer Information

OCID: ...qsn6fa Show Copy

Created: Mon, 01 Oct 2018 08:24:54 GMT

Shape: 100Mbps

IP Address: 129.213.13.157 (Public)

Virtual Cloud Network: net1

Subnet (1 of 2): Public Subnet qAGf:US-ASHBURN-AD-1

Subnet (2 of 2): Public Subnet qAGf:US-ASHBURN-AD-2

Traffic between this load balancer and its backend servers is subject to the governing security lists.

Learn more about Load Balancers and Security Lists.

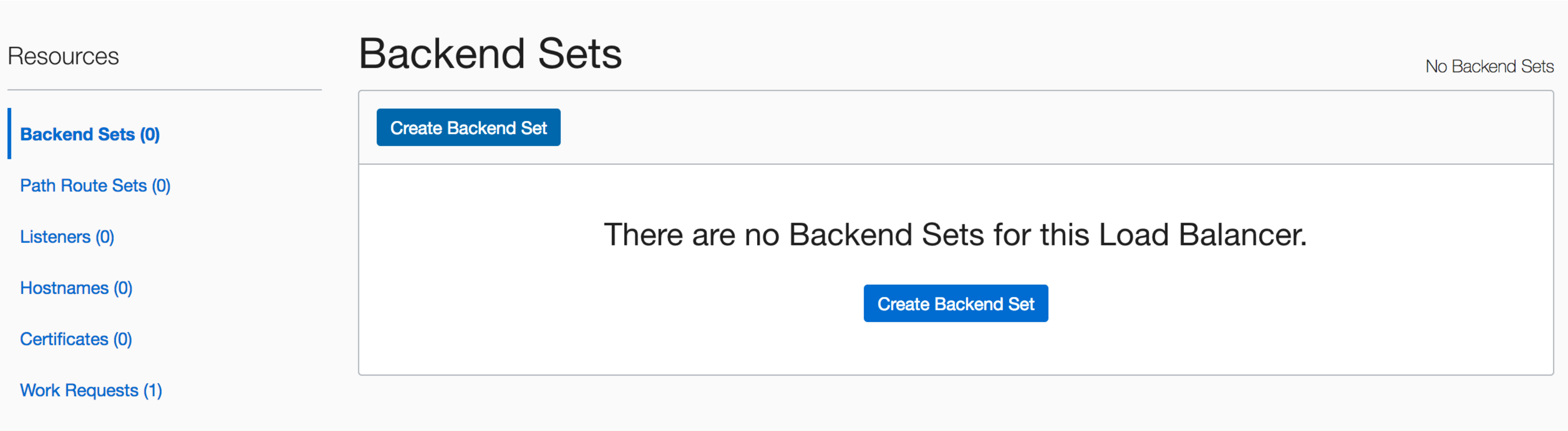
Overall Health

?

Backend Sets Health

0	Critical
0	Warning
0	Unknown
0	OK

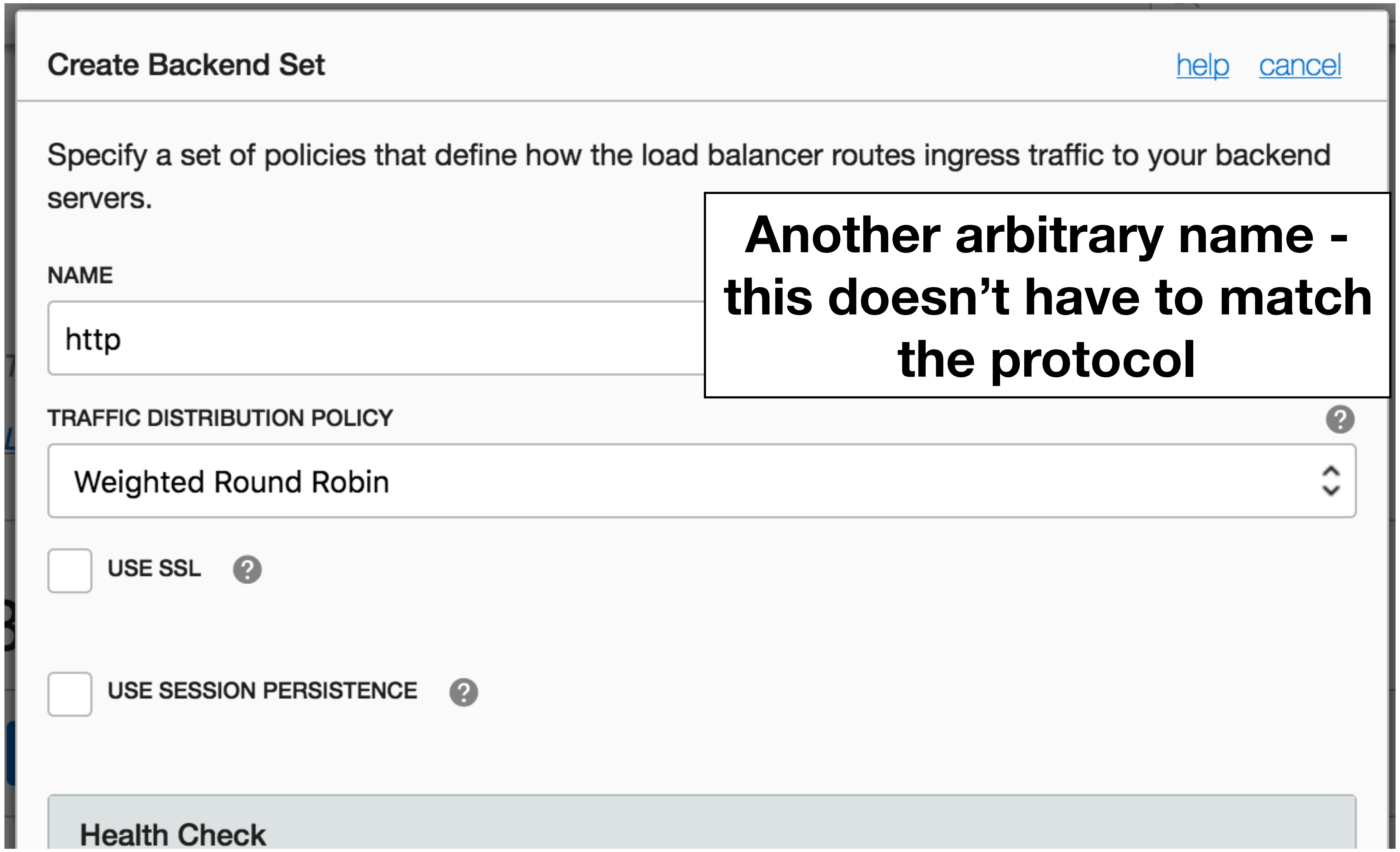
Note the new public IP



Backend Sets

No Backend Sets

There are no Backend Sets for this Load Balancer.



Create Backend Set

Specify a set of policies that define how the load balancer routes ingress traffic to your backend servers.

NAME

http

TRAFFIC DISTRIBUTION POLICY

Weighted Round Robin

□ USE SSL

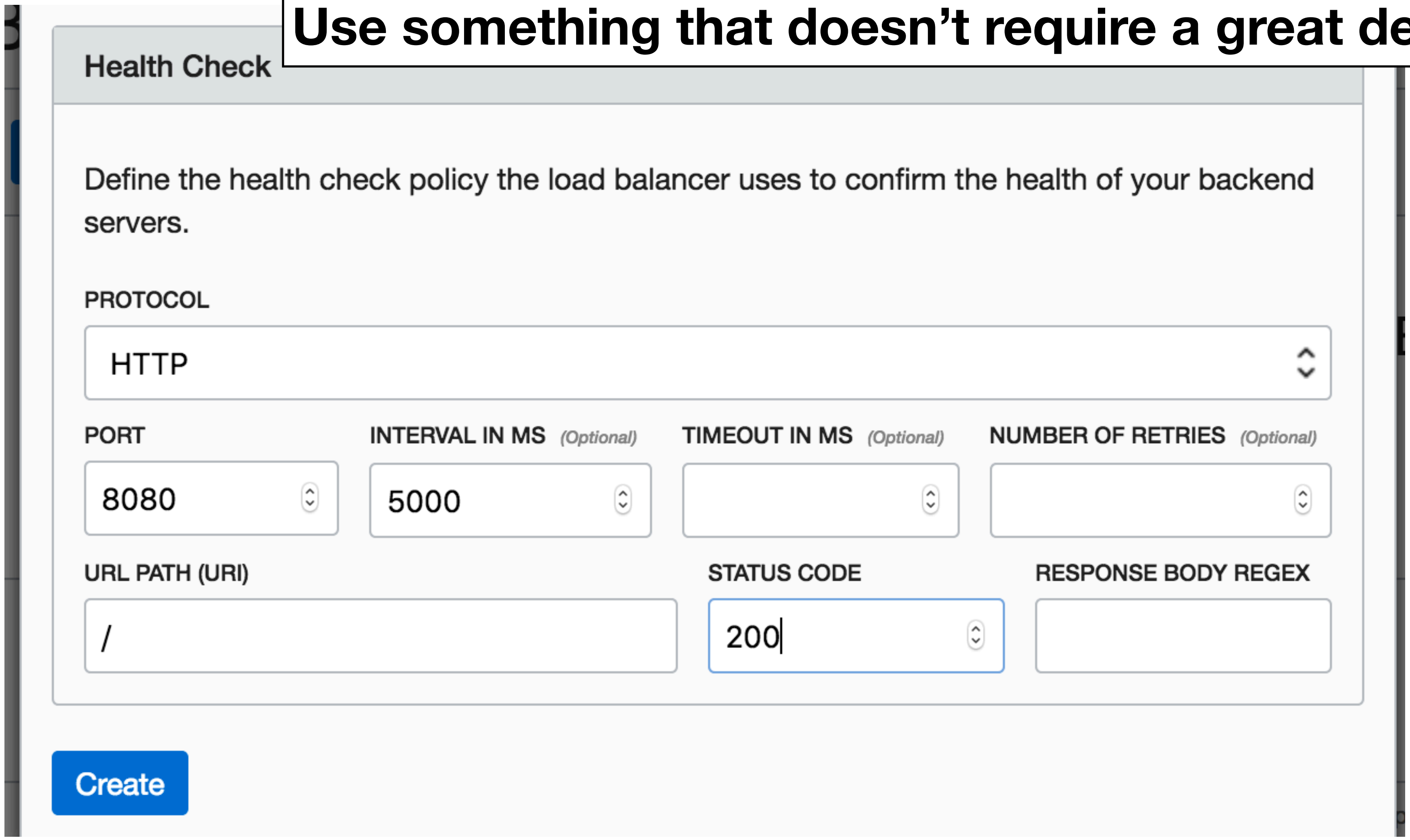
□ USE SESSION PERSISTENCE

Health Check

Another arbitrary name - this doesn't have to match the protocol

The *health check* lets the LB maintain a set of targets that are responding to requests.

Use something that doesn't require a great deal of resource.



Health Check

Define the health check policy the load balancer uses to confirm the health of your backend servers.

PROTOCOL: HTTP

PORT: 8080

INTERVAL IN MS (Optional): 5000

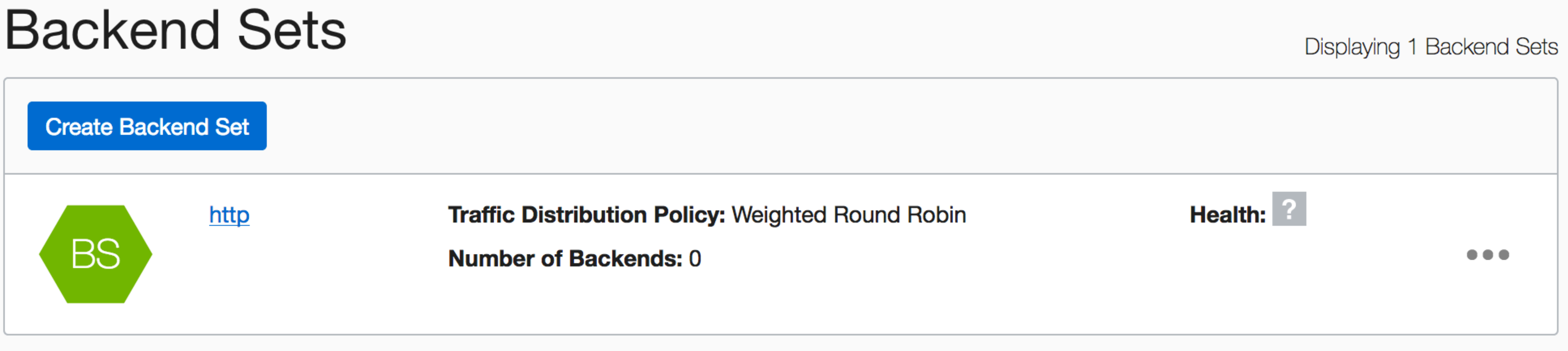
TIMEOUT IN MS (Optional):

NUMBER OF RETRIES (Optional):

URL PATH (URI): /

STATUS CODE: 200

RESPONSE BODY REGEX:



Backend Sets

Displaying 1 Backend Sets

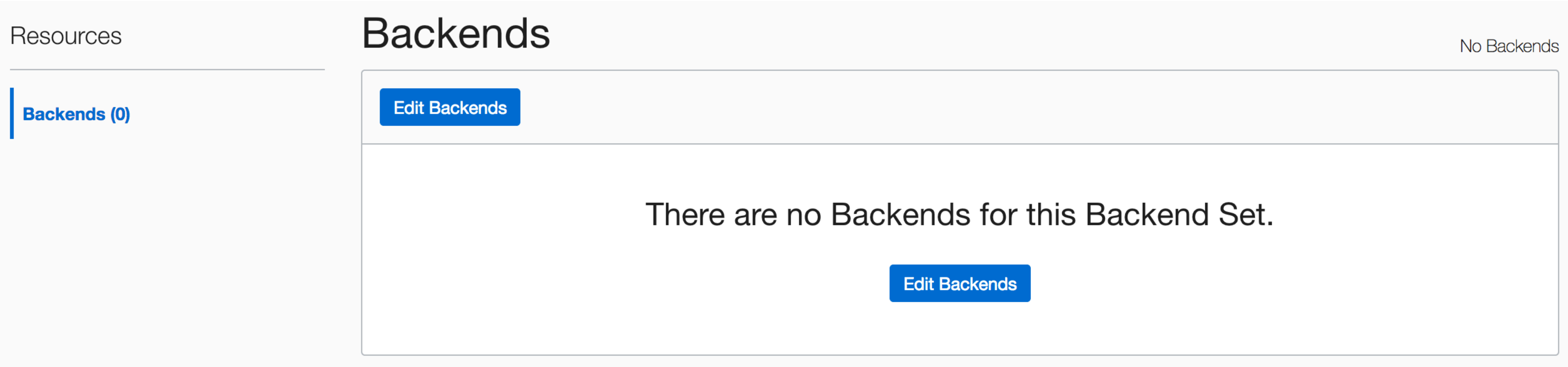
http

Traffic Distribution Policy: Weighted Round Robin

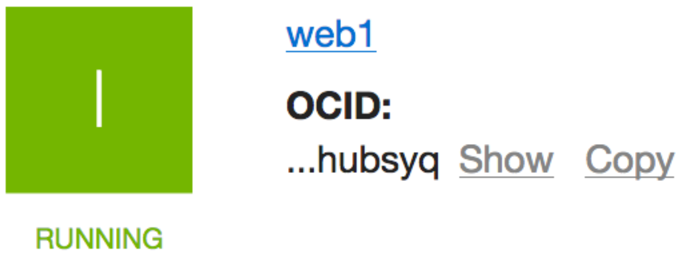
Number of Backends: 0

Health: ?

The result is an empty backend set



There are no Backends for this Backend Set.



web1

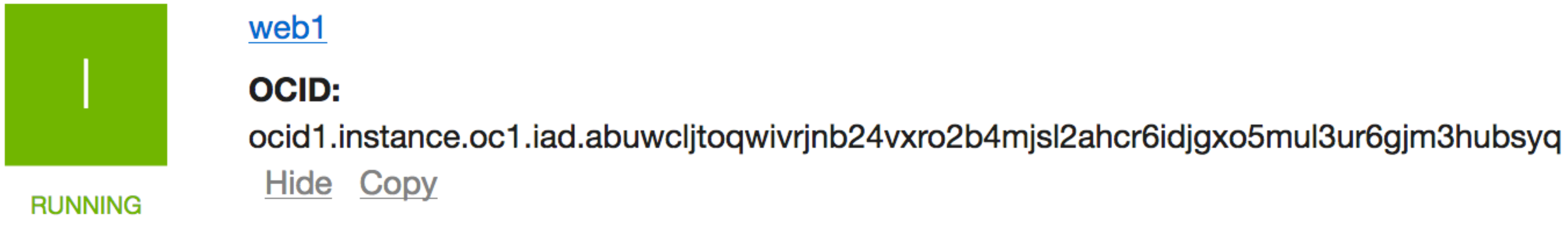
OCID:
...hubsyq

Backends are associated with individual VMs.

These are identified by the *OCID*. It's a long, random identifier.

Navigate to the *Instances* view and copy the OCID of *web1* to the clipboard.

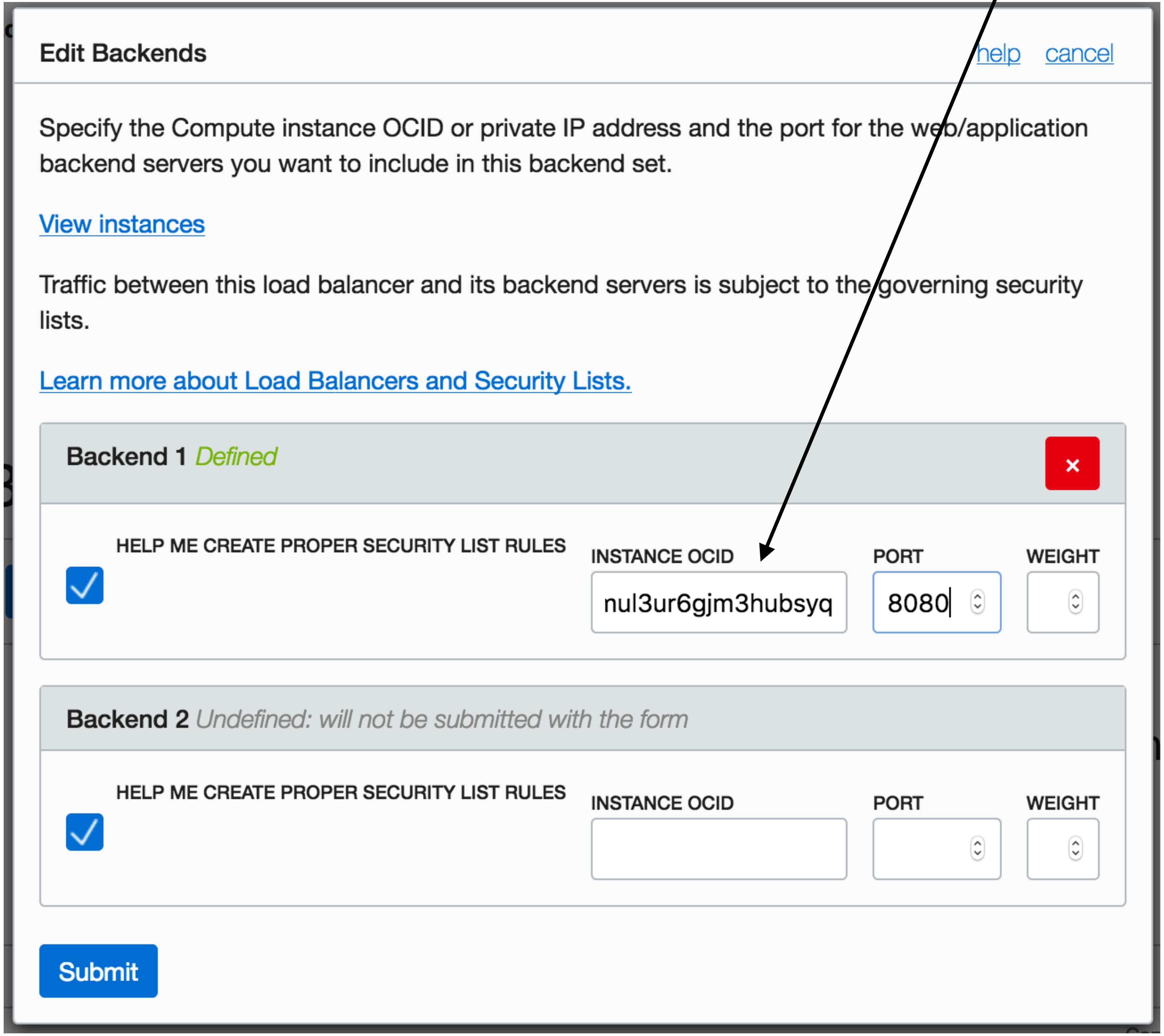
Then find the backend set we just created.



web1

OCID:
ocid1.instance.oc1.iad.abuwcljtoqwivrjnb24vxro2b4mjsl2ahcr6idjgxo5mul3ur6gjm3hubsyq

Paste the OCID in here



Edit Backends

Specify the Compute instance OCID or private IP address and the port for the web/application backend servers you want to include in this backend set.

View instances

Traffic between this load balancer and its backend servers is subject to the governing security lists.

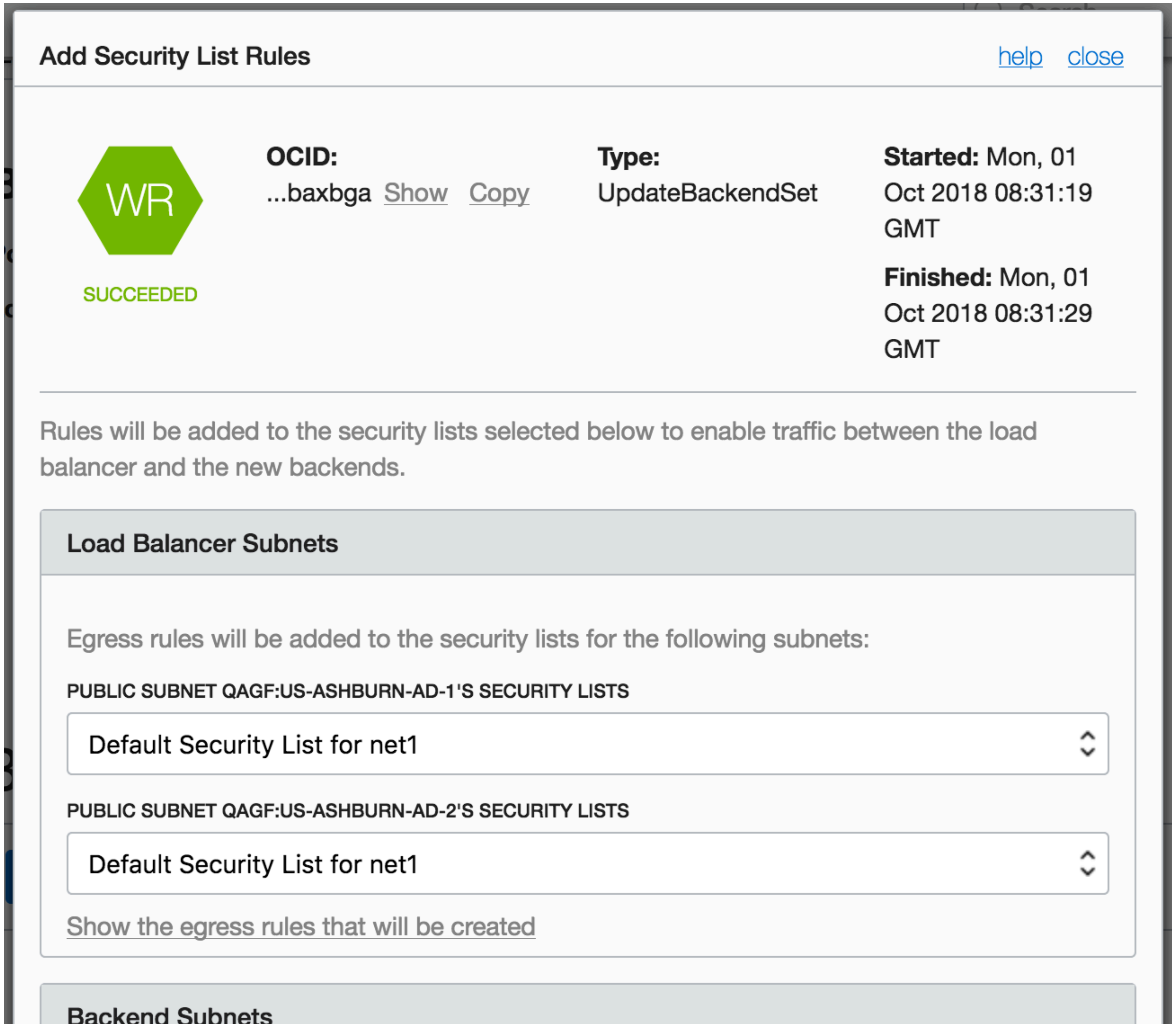
Learn more about Load Balancers and Security Lists.

Backend 1 *Defined*

HELP ME CREATE PROPER SECURITY LIST RULES	INSTANCE OCID	PORT	WEIGHT
☑	nul3ur6gjm3hubsyq	8080	

Backend 2 *Undefined: will not be submitted with the form*

HELP ME CREATE PROPER SECURITY LIST RULES	INSTANCE OCID	PORT	WEIGHT
☑			



Add Security List Rules

OCID: ...baxbga

Type: UpdateBackendSet

Started: Mon, 01 Oct 2018 08:31:19 GMT

Finished: Mon, 01 Oct 2018 08:31:29 GMT

Rules will be added to the security lists selected below to enable traffic between the load balancer and the new backends.

Load Balancer Subnets

Egress rules will be added to the security lists for the following subnets:

PUBLIC SUBNET QAGF:US-ASHBURN-AD-1'S SECURITY LISTS

Default Security List for net1

PUBLIC SUBNET QAGF:US-ASHBURN-AD-2'S SECURITY LISTS

Default Security List for net1

Show the egress rules that will be created

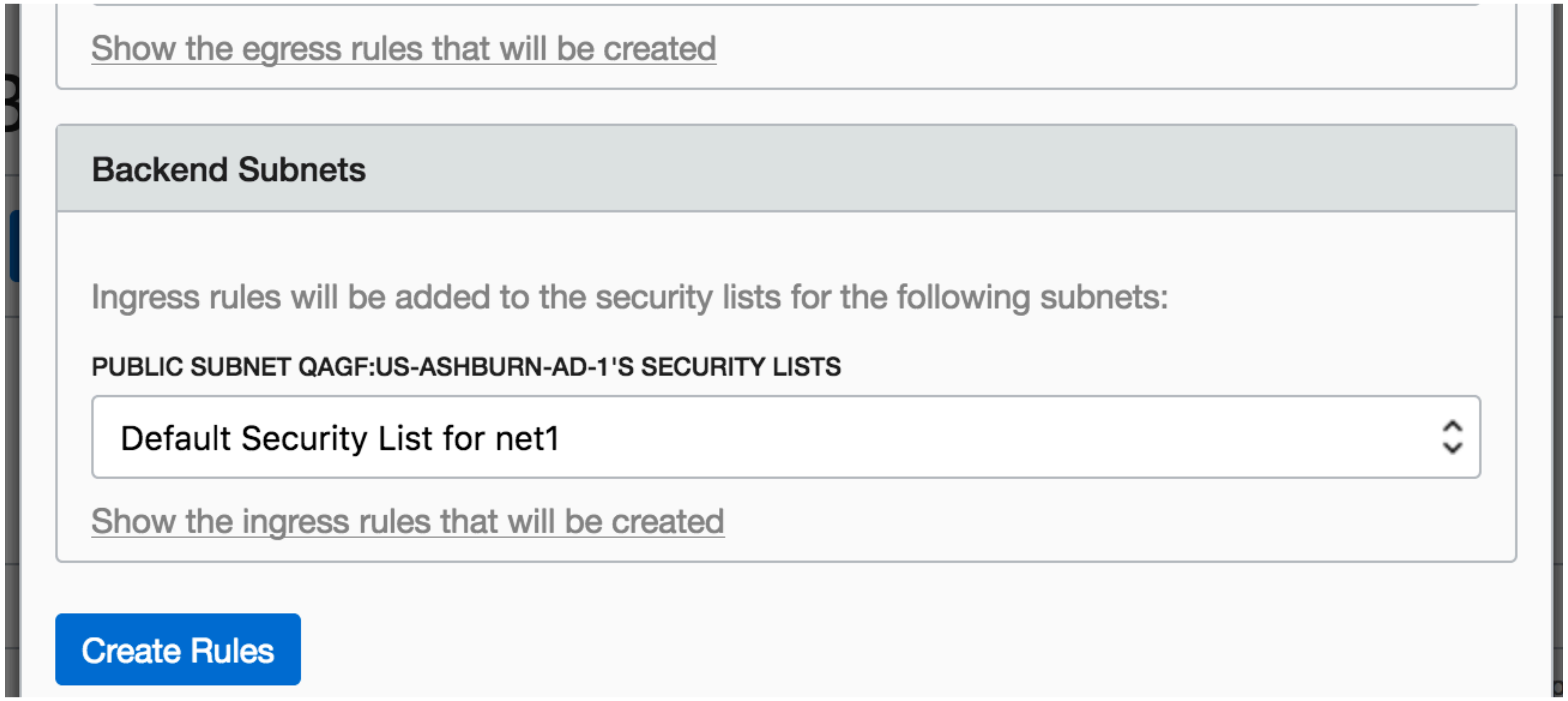
Backend Subnets

You'll be prompted to automatically create additional security rules.

These permit the loadbalancer to reach the backends…

… and these permit traffic from the loadbalancer to the target VMs.

(We need both since the loadbalancer can be attached to different subnets to the worker VMs.)



Show the egress rules that will be created

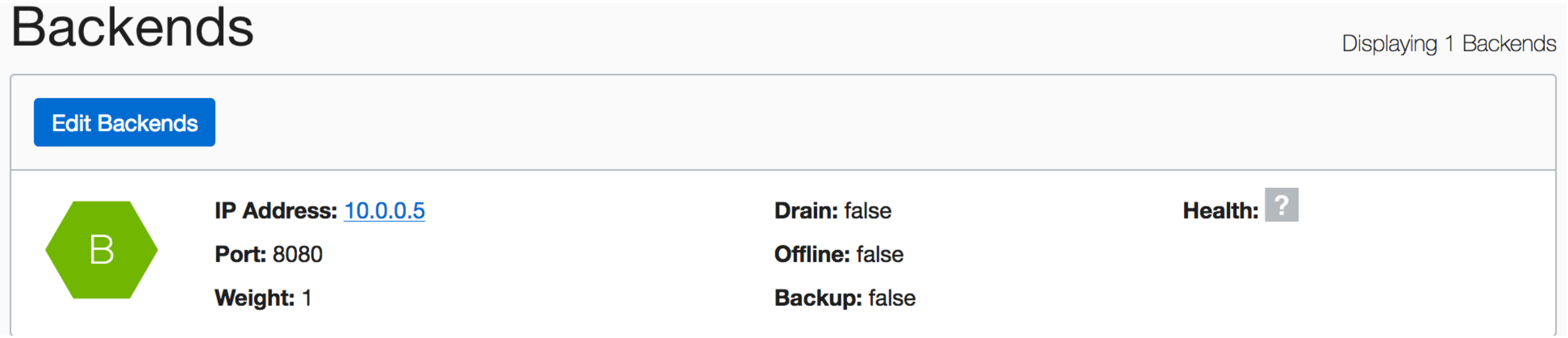
Backend Subnets

Ingress rules will be added to the security lists for the following subnets:

PUBLIC SUBNET QAGF:US-ASHBURN-AD-1'S SECURITY LISTS

Default Security List for net1

Show the ingress rules that will be created



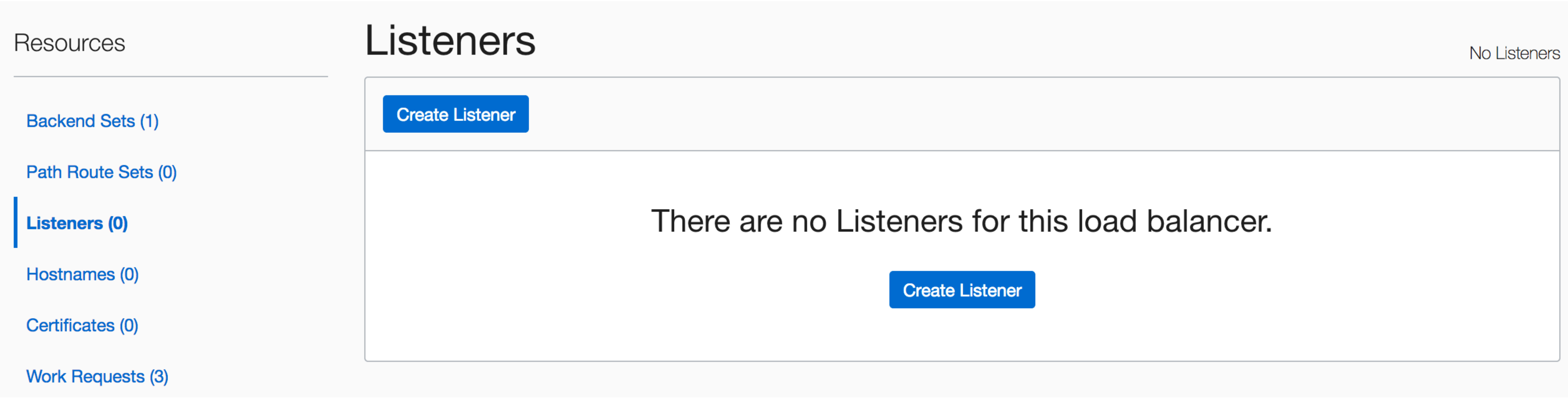
IP Address: 10.0.0.5
Port: 8080
Weight: 1

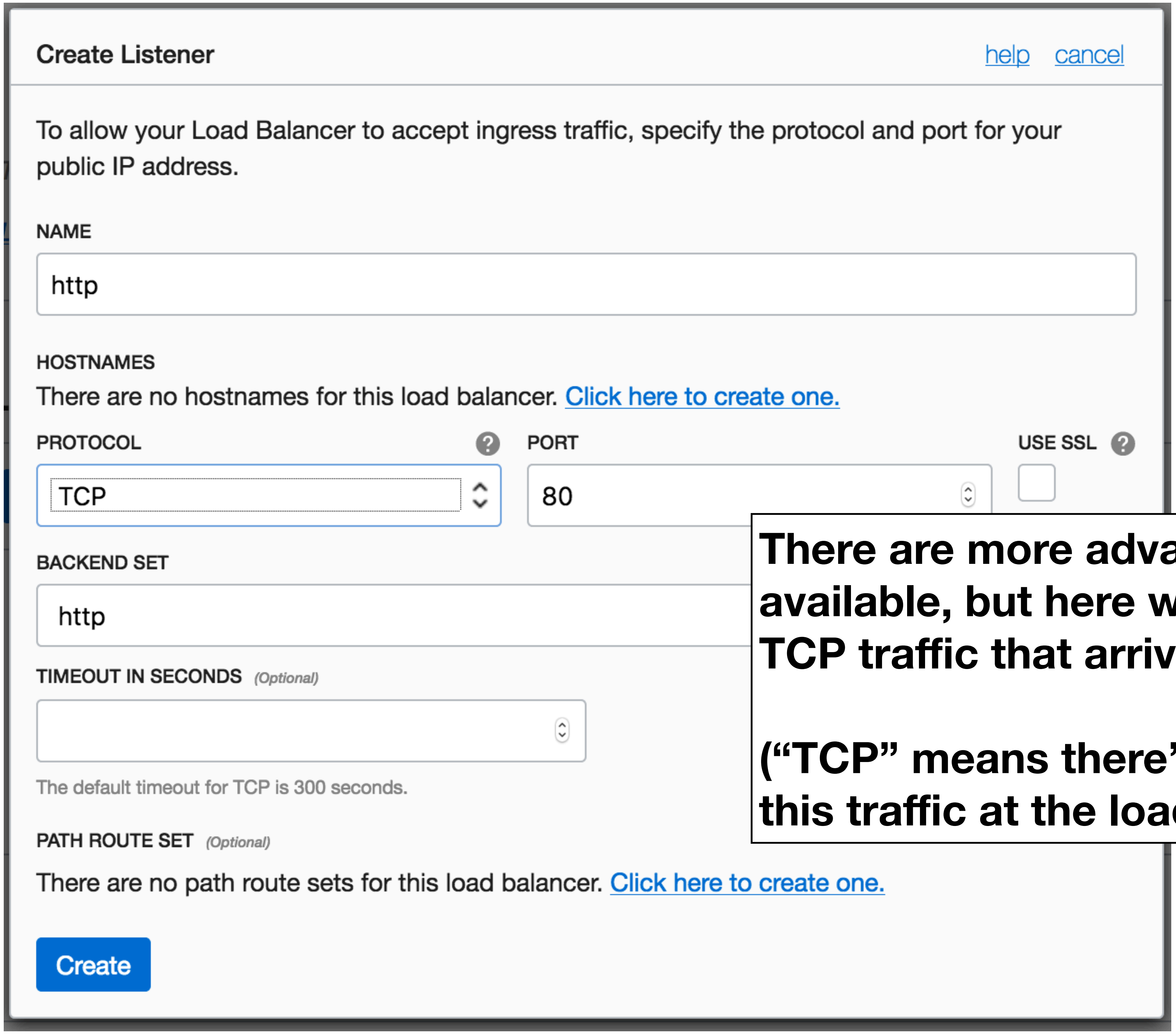
Drain: false
Offline: false
Backup: false

Health: ?

The result: a backend set with a single member.

Finally, create the listener.





Create Listener

To allow your Load Balancer to accept ingress traffic, specify the protocol and port for your public IP address.

NAME

http

HOSTNAMES

There are no hostnames for this load balancer. Click here to create one.

PROTOCOL

TCP

PORT

80

USE SSL

□

BACKEND SET

http

TIMEOUT IN SECONDS *(Optional)*

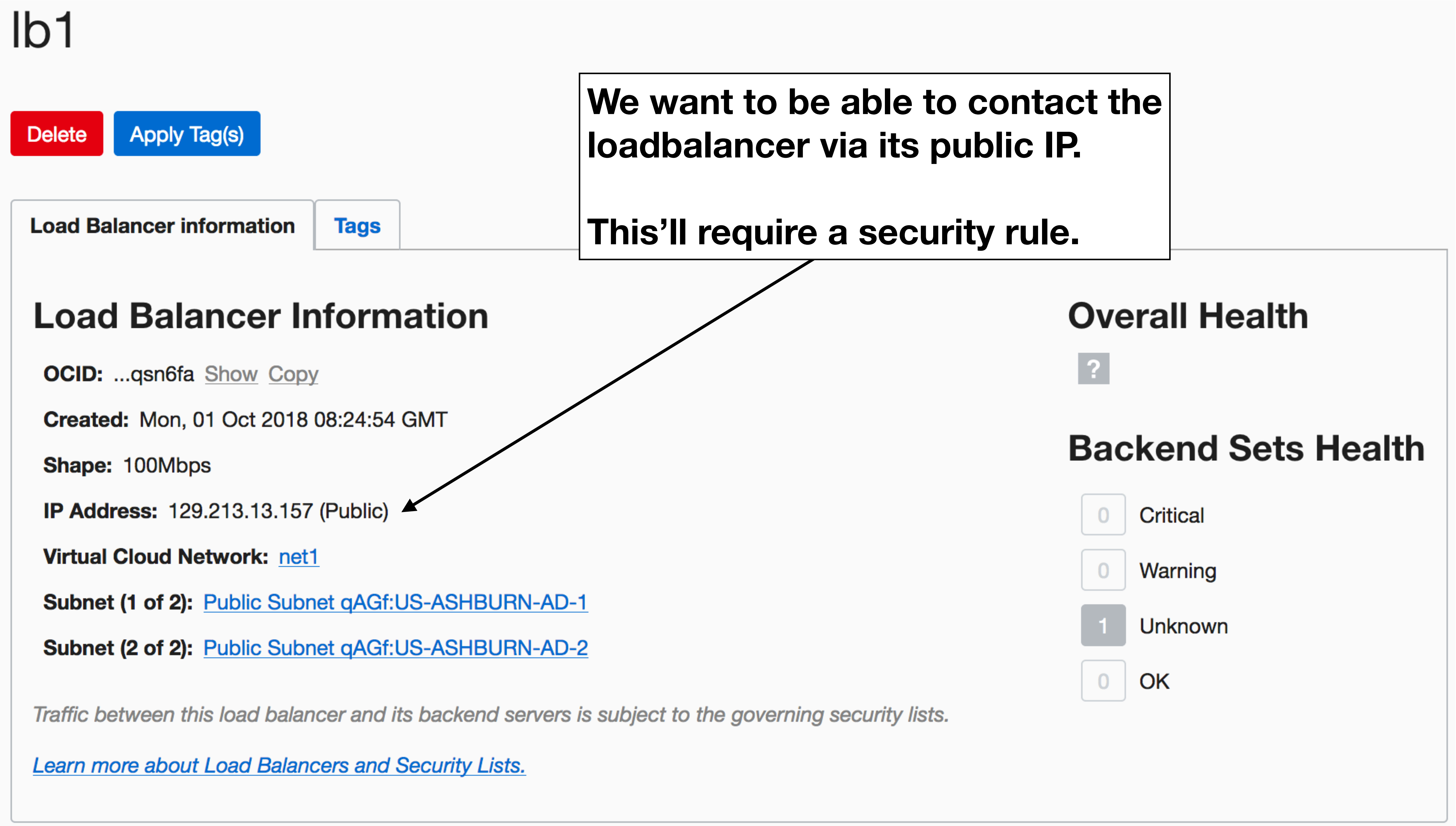
The default timeout for TCP is 300 seconds.

PATH ROUTE SET *(Optional)*

There are no path route sets for this load balancer. Click here to create one.

There are more advanced options available, but here we'll simply forward any TCP traffic that arrives on port 80.

("TCP" means there's no attempt to interpret this traffic at the loadbalancer.)



lb1

Load Balancer Information

OCID: ...qsn6fa

Created: Mon, 01 Oct 2018 08:24:54 GMT

Shape: 100Mbps

IP Address: 129.213.13.157 (Public)

Virtual Cloud Network: net1

Subnet (1 of 2): Public Subnet qAGf:US-ASHBURN-AD-1

Subnet (2 of 2): Public Subnet qAGf:US-ASHBURN-AD-2

Traffic between this load balancer and its backend servers is subject to the governing security lists.

Learn more about Load Balancers and Security Lists.

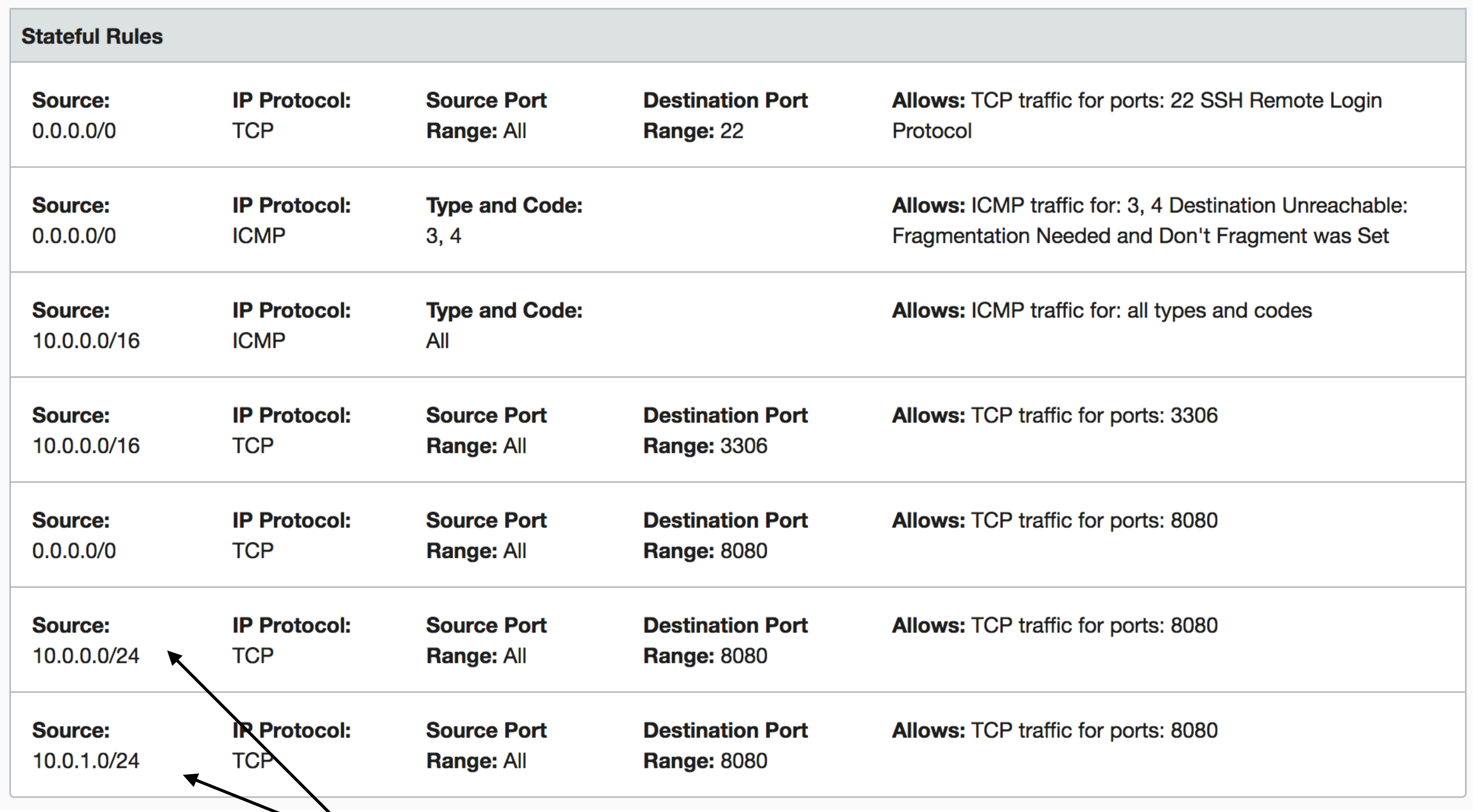
Overall Health

Backend Sets Health

0	Critical
0	Warning
1	Unknown
0	OK

We want to be able to contact the loadbalancer via its public IP.

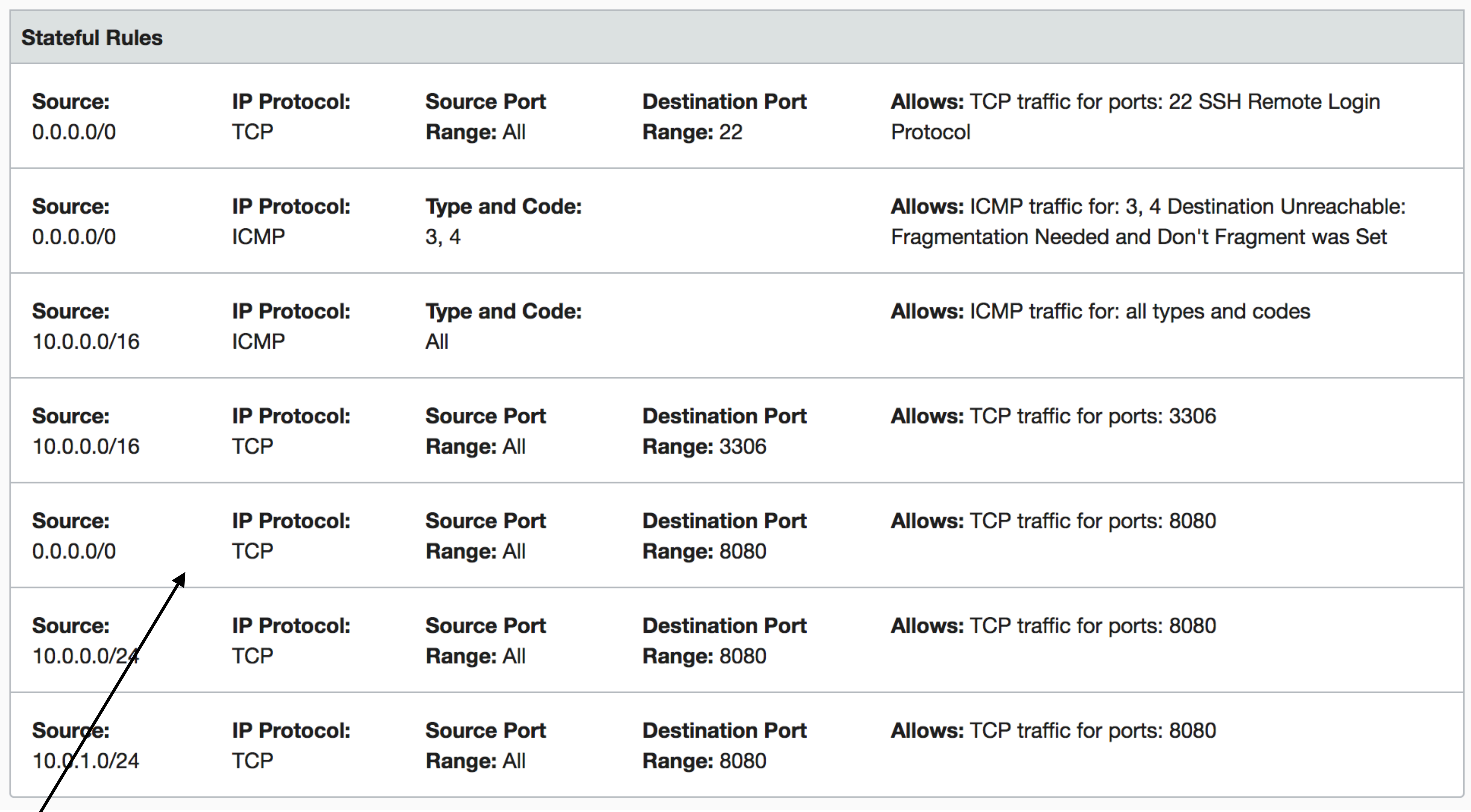
This'll require a security rule.



Stateful Rules

Source: 0.0.0.0/0	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 22	**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol
Source: 0.0.0.0/0	**IP Protocol:** ICMP	**Type and Code:** 3, 4		**Allows:** ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set
Source: 10.0.0.0/16	**IP Protocol:** ICMP	**Type and Code:** All		**Allows:** ICMP traffic for: all types and codes
Source: 10.0.0.0/16	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 3306	**Allows:** TCP traffic for ports: 3306
Source: 0.0.0.0/0	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 8080	**Allows:** TCP traffic for ports: 8080
Source: 10.0.0.0/24	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 8080	**Allows:** TCP traffic for ports: 8080
Source: 10.0.1.0/24	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 8080	**Allows:** TCP traffic for ports: 8080

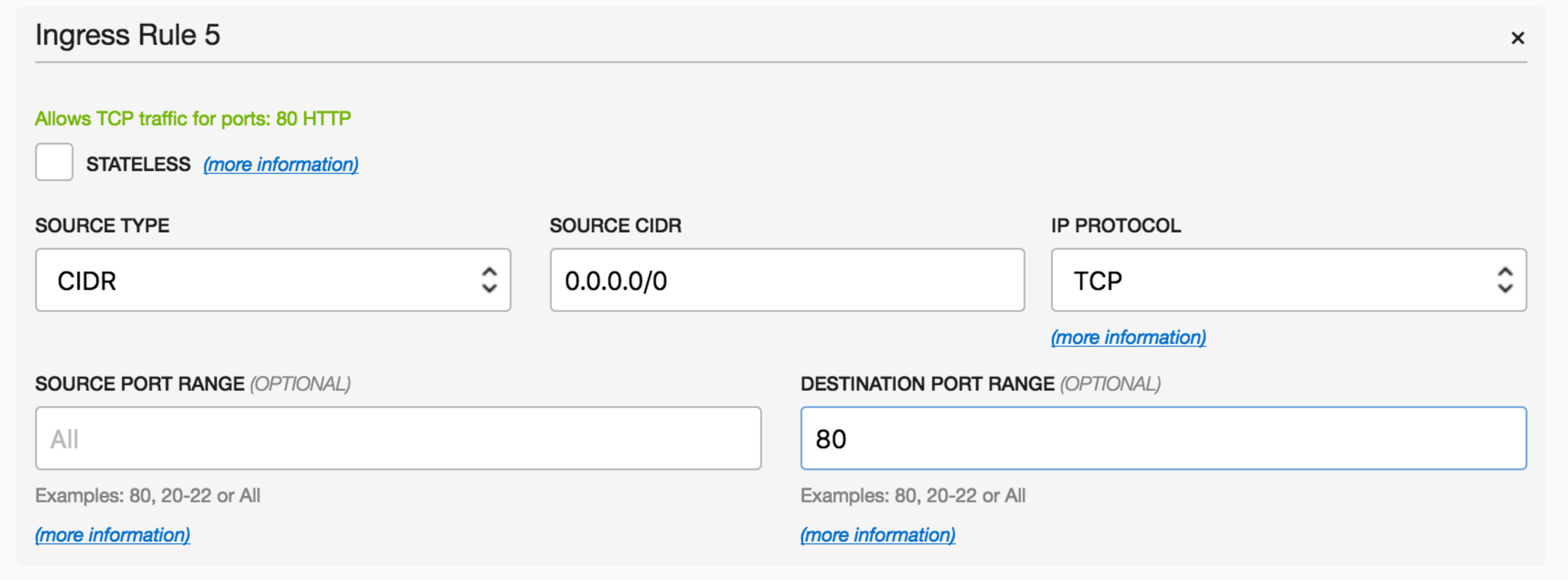
These ingress rules were added automatically



Stateful Rules

Source: 0.0.0.0/0	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 22	**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol
Source: 0.0.0.0/0	**IP Protocol:** ICMP	**Type and Code:** 3, 4		**Allows:** ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set
Source: 10.0.0.0/16	**IP Protocol:** ICMP	**Type and Code:** All		**Allows:** ICMP traffic for: all types and codes
Source: 10.0.0.0/16	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 3306	**Allows:** TCP traffic for ports: 3306
Source: 0.0.0.0/0	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 8080	**Allows:** TCP traffic for ports: 8080
Source: 10.0.0.0/24	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 8080	**Allows:** TCP traffic for ports: 8080
Source: 10.0.1.0/24	**IP Protocol:** TCP	**Source Port Range:** All	**Destination Port Range:** 8080	**Allows:** TCP traffic for ports: 8080

This one is the one we added earlier. We'll edit it to turn off direct access from the Internet to our VM on port 8080, and instead to permit access to the listener.



Ingress Rule 5

Allows TCP traffic for ports: 80 HTTP

□ STATELESS *(more information)*

SOURCE TYPE: CIDR

SOURCE CIDR: 0.0.0.0/0

IP PROTOCOL: TCP

(more information)

SOURCE PORT RANDE *(OPTIONAL)*: All

Examples: 80, 20-22 or All

(more information)

DESTINATION PORT RANGE *(OPTIONAL)*: 80

Examples: 80, 20-22 or All

(more information)

Check!!

```
% curl http://129.213.13.157
<!doctype html>
<html>
<head>
    <title>My page</title>
    <link rel="stylesheet" href="/styles.css">
    <link rel="stylesheet" href="/main.css">
</head>
<body>
<div id="app"></div>
<script src="/js/app.js"></script>
</body>
</html>
```

Check!!

```
% curl http://129.213.13.157
<!doctype html>
<html>
<head>
    <title>My page</title>
    <link rel="stylesheet" href="/styles.css">
    <link rel="stylesheet" href="/main.css">
</head>
<body>
<div id="app"></div>
<script src="/js/app.js"></script>
</body>
</html>
```

- If you can see this, try pointing your browser at
  http://129.213.13.157
  (use your loadbalancer’s public IP here).

DNS, again

- If you’ve a domain, update the *A record* to point to the loadbalancer’s public IP address.

What’s missing here?

- Backups! What happens if one of our VMs is destroyed?
  - There are various mysql tools that can dump the state of the database to a file.
  - We might upload that file to *object storage*
- A backup plan is not complete without a recovery plan
  - A recovery plan *doesn’t work* unless you’ve tested it

but…

Backups are not enough

- Focus on the question of *service availability*
- What does my data represent?
- How important is it that it’s fresh (consistent, versus available)?
- How long an outage can I tolerate?
- Of what fraction of data?
- How secure are copies?
- Do I need a transactional history?

What's missing here?

- Regional Scalability! Does all traffic need to cross the Atlantic?
  - Approaches like *GeoDNS* give different results to client depending on where in the world they are
  - Different *A records* means that I might be directed to a data centre in London rather than the US.
    - What are the implications on my application/data architecture?
    - Can I rely on asynchronous updates?

What’s missing here?

- Observability!
  - How can I tell if my app is working?
  - As my application architecture becomes more complex, what can I do to locate bottlenecks?

What’s missing here?

- Security!
    - How can I prevent the interception of traffic from/to a client?
    - “Let’s Encrypt” offers a straightforward way to install TLS certificates.
    - The certificate should identify the *public name* of the service.
- There’s a lot more to this. Should we encrypt internal traffic? (Yes!)

What’s missing here?

- Automation!
  - Standing up these VMs was a really tedious, error-prone process
  - There are better tools
    - OCI offers a command-line tool (and a REST API). (Documentation is available via the console.)
    - Terraform lets you describe your infrastructure (more-or-less) declaratively
    - Ansible, Puppet, Salt, Chef, … are all tools for automating the configuration of hosts

What’s missing here?

- Updates: probably the most important aspect.
    - “What about this critical OS update?”
        - Are you going to patch, or blow away and redeploy?
    - “How do I change my application?”
        - What about database schemas?
        - Do I need to take everything down to bring up a new version?
        - What about the REST API? Is it versioned? Will old clients continue to work? Does that matter?
        - How can I manage multi-region updates?
- Thinking about this needs to be done early in a design.

Good luck!

- Good news: there are better ways to do this.

- Deployment architecture, how does that map onto: VMs, networks, loadbalancers, etc.
- - java
- - - making it run
- - - making it run on reboot
- - - configuring in the same credentials
- - - schema management, different creds.
- - - What about db migrations?
- - - backups?
- - - - well, the state is stored in the DB. what about the configuration?
- - - robustness: clustering
- - implications for the "Google login" - can't guarantee that the callback hits the same host that the subsequent user request will.

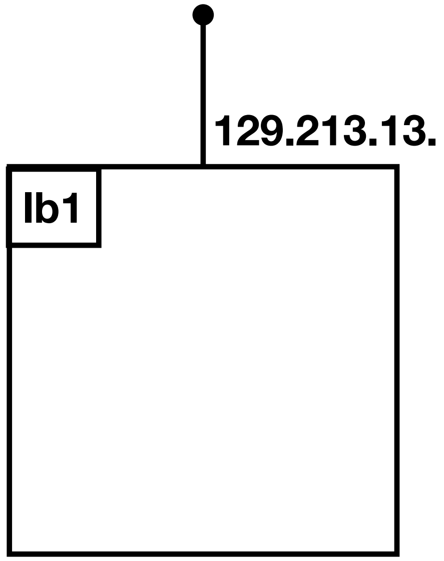
- Locating the app on the web
- - Securing it
- - - security certs
- - - - self-signed?
- - - - let's encrypt?
- - - - the certificate needs to identify the service *as the client sees it*
- - - what about securing internal traffic?
- - - - It's a good idea.

- - LBaaS at the front end
- - - This becomes the IP address you want DNS to point to
- - - you’ll need security rules for this traffic, too.

- Repeatability
- - from the web console to command-line tools, scripting
- - targeted scripting tools (eg terraform, others..?)
- - - some tools target host configuration, some target infrastructure layout, some try to do a bit of both.

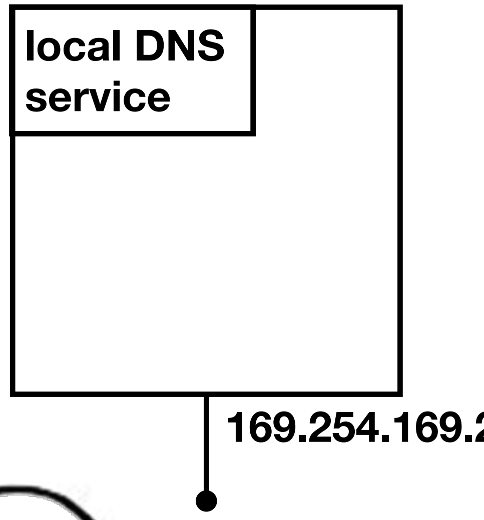
- What about updates?
- - Ideal: I can deliver updates with no user interruption. I can roll updates out piecemeal and cut over to them.

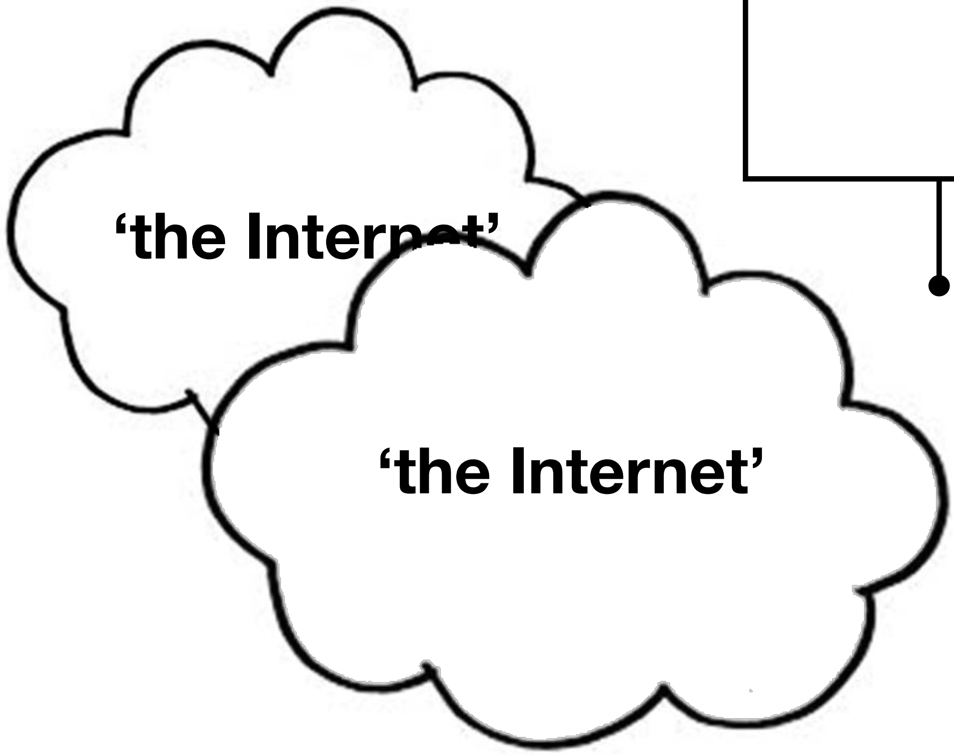
- - What kind of implications does this have? Schema, object versioning. Writing software that's forwards- and

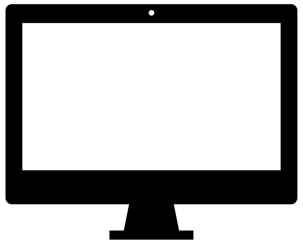


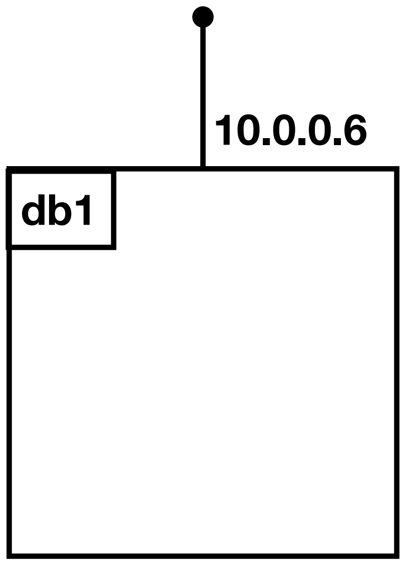
129.213.229.130

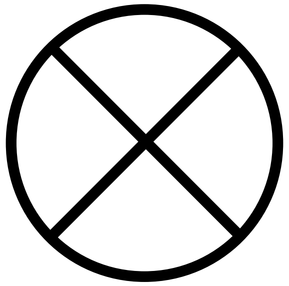
port	service
22	sshd
80	httpd

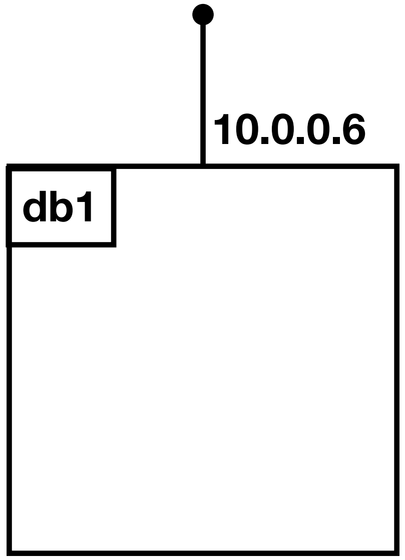


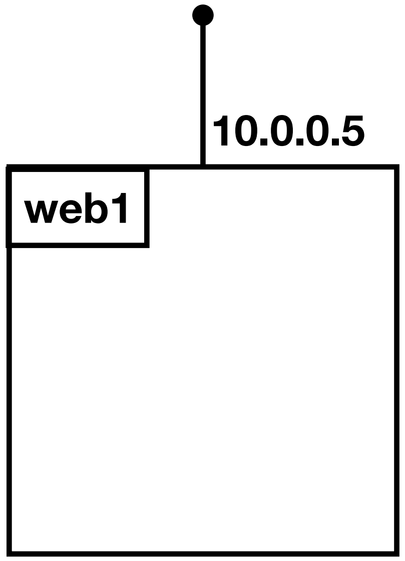


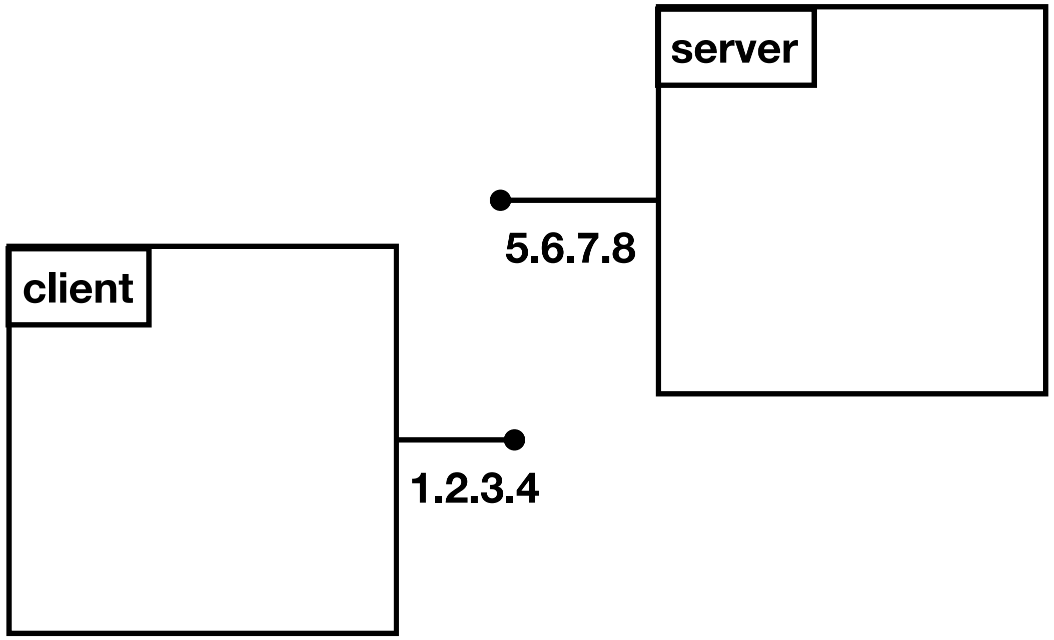











server
5.6.7.8
client
1.2.3.4