Reverse Proxy Implementation using Amazon EC2

Shanto Roy
University of Houston
Houston, Texas
sroy10@uh.edu

1 INTRODUCTION

This report represents the reverse proxy set up in Amazon AWS[1] EC2 instance based on the previous code[2] from HW1.

2 BACKGROUND

2.1 Amazon AWS EC2

Amazon Web Service (AWS) provides on-demand cloud computing platforms for computation, application hosting, storages, and numerous other services. AWS is now one of the leading cloud provider and it is better to understand how the EC2 instances work (security group, networking, public access, etc.).

2.2 Domain Names

Domain names are typical identifiers of different entities within the Internet. To have public access, AWS provides public API to developers. However, the public access domain names are long, and that's why we use custom Domain Names to replace that.

3 PROCEDURE

3.1 Creating EC2 Instances

- Three EC2 instances are launched using Amazon Linux 2 AMI 64 bit.
- Instance type is **t2.micro**
- Created two security groups
  - One for the Reverse Proxy
  - Another for the servers
- Storage Selection is the default choice
- Used one key pair to access all three EC2 instances

3.2 EC2 Local and Global Networking

Added Security Group inbound rules

- Reverse Proxy port accessible from anywhere
- All ports accessible locally (inbound between the two security groups)

3.3 Connecting to Amazon EC2

```
ssh -i ~/aws-keys/reverse-proxy-aws.pem\
            ec2-user@reverseproxy.ddns.net
```

3.4 Setting Up Domain Name

3.4.1 Create a Domain Name. NOIP[3] offers three free domain names. I registered- **reverseproxy.ddns.net**.

3.4.2 DDNS Setup in Amazon EC2 Instance. Setting up DDNS using noip[3]. The DDNS setup [1] steps are as follows:

```
    $ sudo yum-config-manager --enable epel
    $ sudo yum install -y noip
    $ sudo noip2 -C
    $ sudo chkconfig noip on
    $ sudo service noip start
```

3.5 Copying Scripts to AWS EC2

For the reverse proxy

```
$ scp -i ~/aws-keys/reverse-proxy-aws.pem\
            aws_rproxy.py\
            ec2-user@reverseproxy.ddns.net:~
```

For other server instances

```
$ scp -i ~/aws-keys/reverse-proxy-aws.pem\
        aws_server.py\
        ec2-user@reverseproxy.ddns.net:~
```

3.6 Create Python3 Virtual Environment

```
$ sudo yum install python3 -y
$ python3 -m venv rproxy/env
$ source ~/rproxy/env/bin/activate
(env) $ pip install pip --upgrade
(env) $ pip install pandas
```

3.7 Running the Codes

3.7.1 Reverse Proxy. Running Reverse Proxy on port 8000

```
$ python reverse_proxy.py -port 8000
```

[1] https://aws.amazon.com/

[2] https://github.com/shantoroy/reverse_proxy

[3] https://www.noip.com/

3.7.2 Servers. Running two servers (id 100 & 200) on port 5000

```
$ python server.py -id 100 -pp 111 \
            -listen 5000 -revproc 8000
```

```
$ python server.py -id 200 -pp 111 \
            -listen 5000 -revproc 8000
```

3.7.3 Client. Sending multiple packets using the following command:

```
$ python client.py -id 1 -revproc 8000 \
                     -pkt pktfiles/1.json
```

4 RESULT

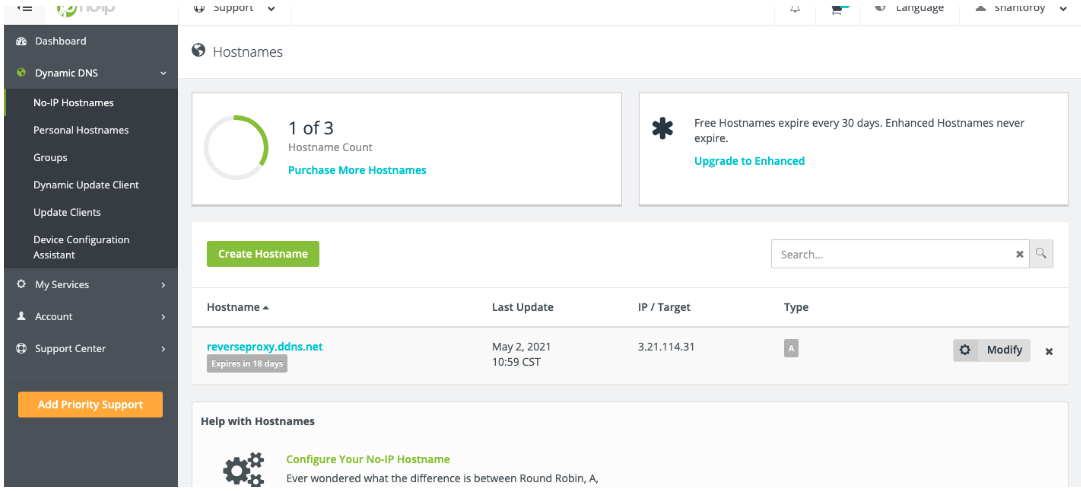
The demo is shown in the Appendix and a video demo can be found in the Github repository^2.

REFERENCES

[1] [n.d.]. Set up dynamic DNS on Your Amazon Linux instance - Amazon Elastic Compute Cloud. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dynamic-dns.html. (Accessed on 05/02/2021).

A APPENDIX

A.1 Creating a Temporary Domain Name



1 of 3
Hostname Count
Purchase More Hostnames

Free Hostnames expire every 30 days. Enhanced Hostnames never expire.
Upgrade to Enhanced

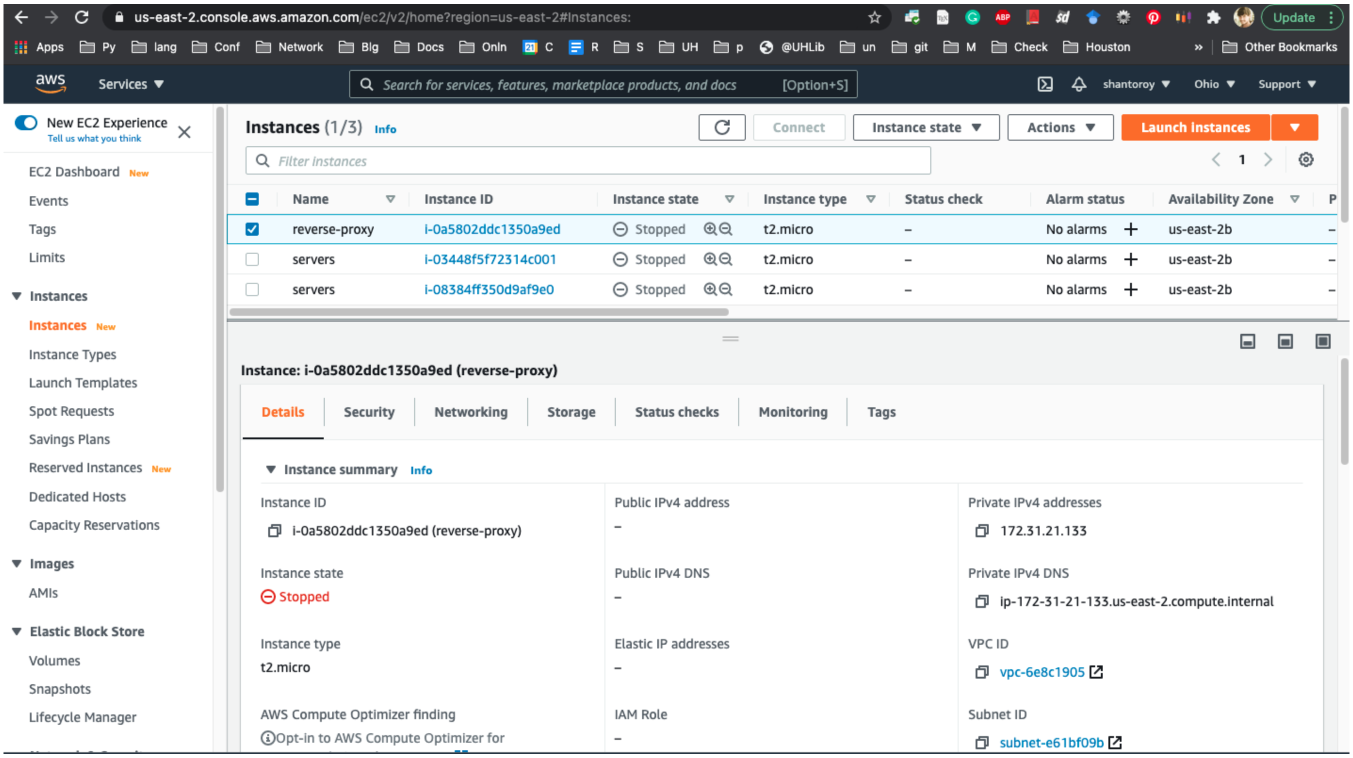
Hostname ▲	Last Update	IP / Target	Type
reverseproxy.ddns.net Expires in 18 days	May 2, 2021 10:59 CST	3.21.114.31	A

Help with Hostnames

Configure Your No-IP Hostname
Ever wondered what the difference is between Round Robin, A,

Figure 1: Free Hostname in NoIP

A.2 Amazon EC2 User Dashboard



Instances (1/3) Info

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑	reverse-proxy	i-0a5802ddc1350a9ed	Stopped	t2.micro	–	No alarms	us-east-2b
□	servers	i-03448f5f72314c001	Stopped	t2.micro	–	No alarms	us-east-2b
□	servers	i-08384ff350d9af9e0	Stopped	t2.micro	–	No alarms	us-east-2b

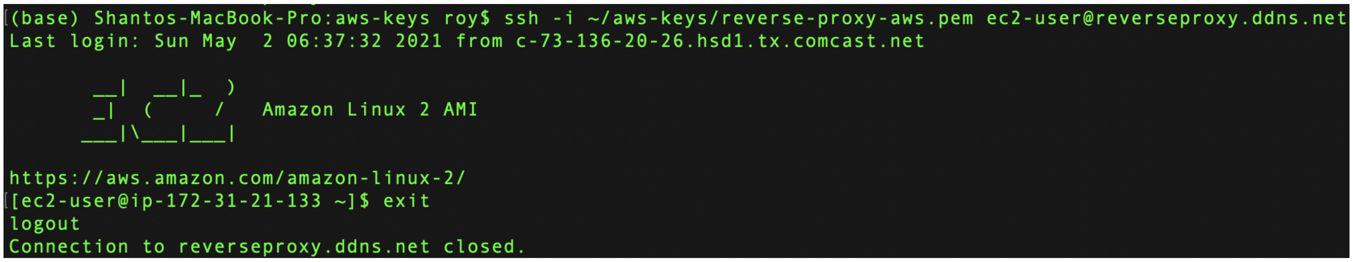
Instance: i-0a5802ddc1350a9ed (reverse-proxy)

▼ Instance summary Info

Instance ID i-0a5802ddc1350a9ed (reverse-proxy)	Public IPv4 address –	Private IPv4 addresses 172.31.21.133
Instance state Stopped	Public IPv4 DNS –	Private IPv4 DNS ip-172-31-21-133.us-east-2.compute.internal
Instance type t2.micro	Elastic IP addresses –	VPC ID vpc-6e8c1905
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for	IAM Role –	Subnet ID subnet-e61bf09b

Figure 2: Amazon AWS EC2 User Dashboard

A.3 SSH to an Amazon EC2 Instance from Terminal



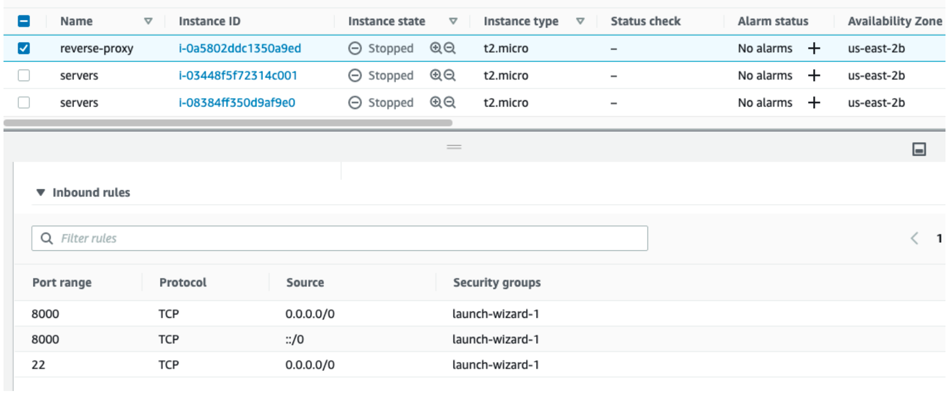
```
(base) Shantos-MacBook-Pro:aws-keys roy$ ssh -i ~/aws-keys/reverse-proxy-aws.pem ec2-user@reverseproxy.ddns.net
Last login: Sun May  2 06:37:32 2021 from c-73-136-20-26.hsd1.tx.comcast.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-21-133 ~]$ exit
logout
Connection to reverseproxy.ddns.net closed.
```

Figure 3: SSH to AWS EC2

A.4 Add Inbound Rules for Local and Global Access



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑	reverse-proxy	i-0a5802ddc1350a9ed	Stopped	t2.micro	–	No alarms	us-east-2b
☐	servers	i-03448f5f72314c001	Stopped	t2.micro	–	No alarms	us-east-2b
☐	servers	i-08384ff350d9af9e0	Stopped	t2.micro	–	No alarms	us-east-2b

▼ Inbound rules

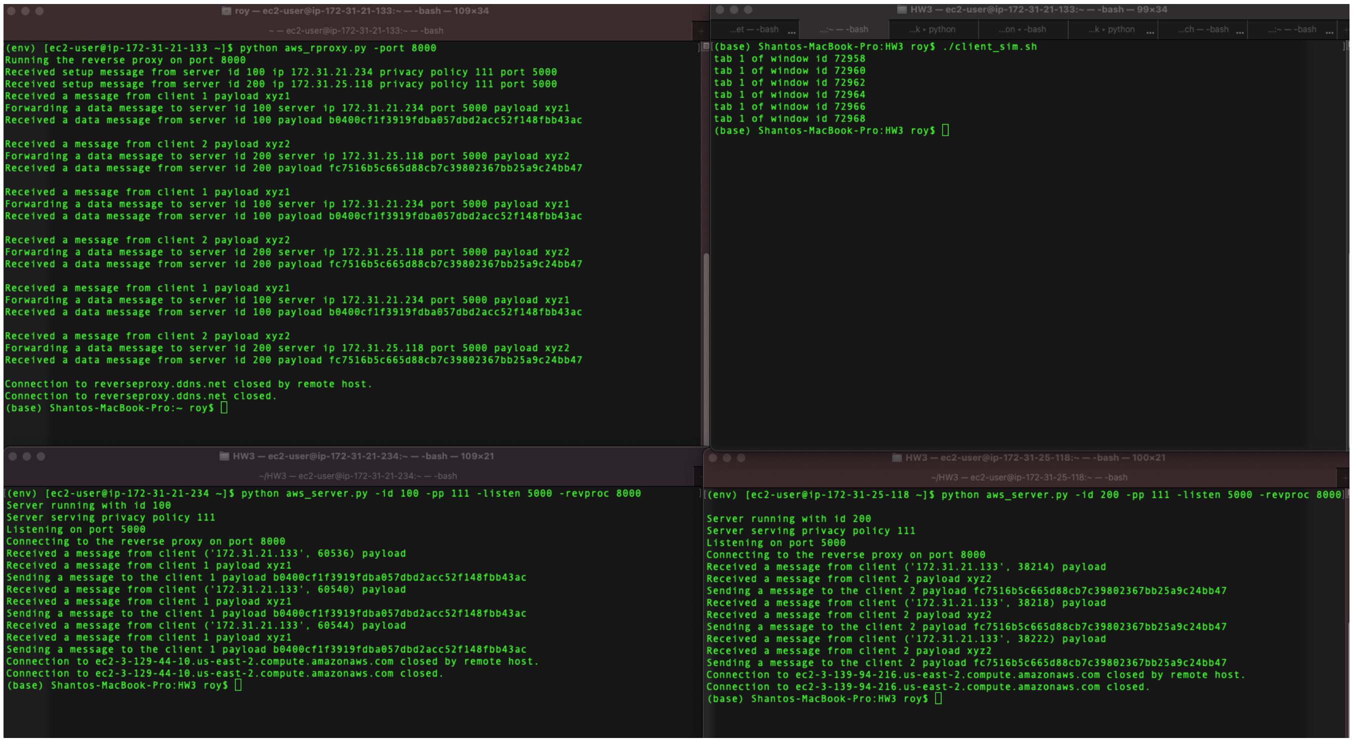
Port range	Protocol	Source	Security groups
8000	TCP	0.0.0.0/0	launch-wizard-1
8000	TCP	::/0	launch-wizard-1
22	TCP	0.0.0.0/0	launch-wizard-1

Figure 4: Reverse Proxy Inbound Port

A.5 Final Output

The upper left terminal represents the reverse proxy server, the bottom two are two servers, and the upper right represents the client side. Video demonstration is available at the Github repository[4].

[4] https://github.com/shantoroy/reverse_proxy



roy — ec2-user@ip-172-31-21-133:~ — -bash — 109×34
~ — ec2-user@ip-172-31-21-133:~ — -bash
HW3 — ec2-user@ip-172-31-21-133:~ — -bash — 99×34
...et — -bash
...:~ — -bash
...k ▸ python
...on ◂ -bash
...k ▸ python
...ch — -bash
...:~ — -bash
HW3 — ec2-user@ip-172-31-21-234:~ — -bash — 109×21
~/HW3 — ec2-user@ip-172-31-21-234:~ — -bash
HW3 — ec2-user@ip-172-31-25-118:~ — -bash — 100×21
~/HW3 — ec2-user@ip-172-31-25-118:~ — -bash

```
(env) [ec2-user@ip-172-31-21-133 ~]$ python aws_rproxy.py -port 8000
Running the reverse proxy on port 8000
Received setup message from server id 100 ip 172.31.21.234 privacy policy 111 port 5000
Received setup message from server id 200 ip 172.31.25.118 privacy policy 111 port 5000
Received a message from client 1 payload xyz1
Forwarding a data message to server id 100 server ip 172.31.21.234 port 5000 payload xyz1
Received a data message from server id 100 payload b0400cf1f3919fdba057dbd2acc52f148fbb43ac

Received a message from client 2 payload xyz2
Forwarding a data message to server id 200 server ip 172.31.25.118 port 5000 payload xyz2
Received a data message from server id 200 payload fc7516b5c665d88cb7c39802367bb25a9c24bb47

Received a message from client 1 payload xyz1
Forwarding a data message to server id 100 server ip 172.31.21.234 port 5000 payload xyz1
Received a data message from server id 100 payload b0400cf1f3919fdba057dbd2acc52f148fbb43ac

Received a message from client 2 payload xyz2
Forwarding a data message to server id 200 server ip 172.31.25.118 port 5000 payload xyz2
Received a data message from server id 200 payload fc7516b5c665d88cb7c39802367bb25a9c24bb47

Received a message from client 1 payload xyz1
Forwarding a data message to server id 100 server ip 172.31.21.234 port 5000 payload xyz1
Received a data message from server id 100 payload b0400cf1f3919fdba057dbd2acc52f148fbb43ac

Received a message from client 2 payload xyz2
Forwarding a data message to server id 200 server ip 172.31.25.118 port 5000 payload xyz2
Received a data message from server id 200 payload fc7516b5c665d88cb7c39802367bb25a9c24bb47

Connection to reverseproxy.ddns.net closed by remote host.
Connection to reverseproxy.ddns.net closed.
(base) Shantos-MacBook-Pro:~ roy$ 
```

```
(base) Shantos-MacBook-Pro:HW3 roy$ ./client_sim.sh
tab 1 of window id 72958
tab 1 of window id 72960
tab 1 of window id 72962
tab 1 of window id 72964
tab 1 of window id 72966
tab 1 of window id 72968
(base) Shantos-MacBook-Pro:HW3 roy$ 
```

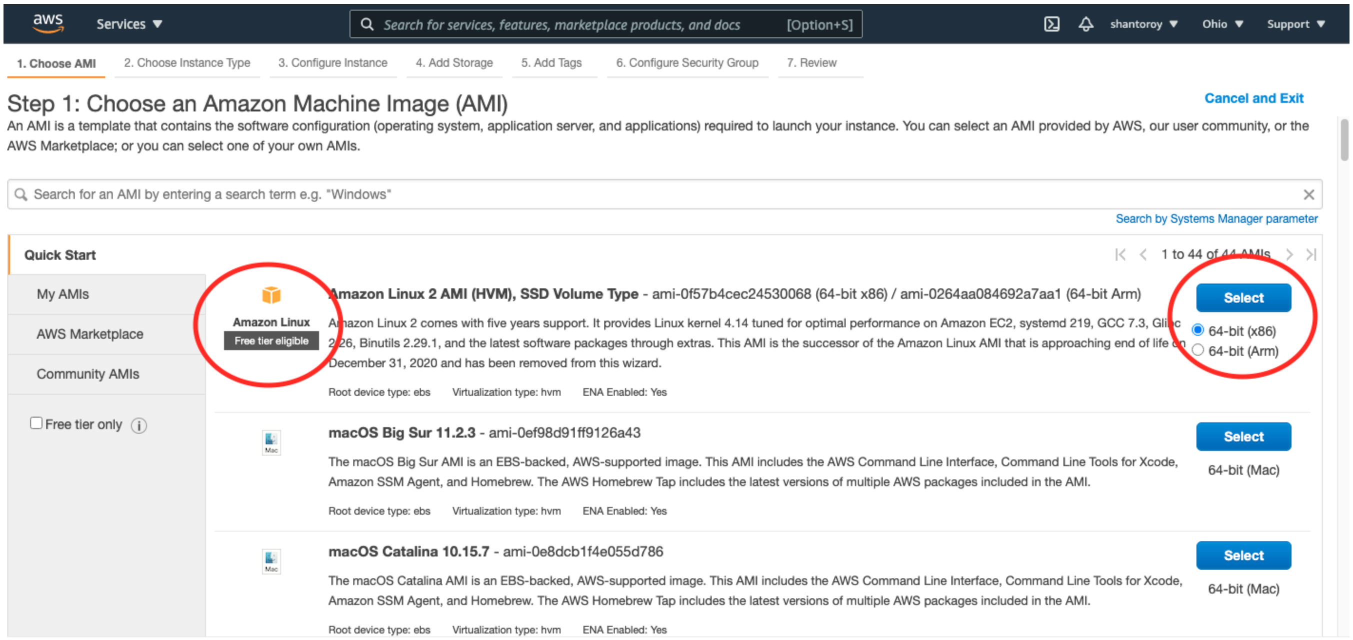
```
(env) [ec2-user@ip-172-31-21-234 ~]$ python aws_server.py -id 100 -pp 111 -listen 5000 -revproc 8000
Server running with id 100
Server serving privacy policy 111
Listening on port 5000
Connecting to the reverse proxy on port 8000
Received a message from client ('172.31.21.133', 60536) payload
Received a message from client 1 payload xyz1
Sending a message to the client 1 payload b0400cf1f3919fdba057dbd2acc52f148fbb43ac
Received a message from client ('172.31.21.133', 60540) payload
Received a message from client 1 payload xyz1
Sending a message to the client 1 payload b0400cf1f3919fdba057dbd2acc52f148fbb43ac
Received a message from client ('172.31.21.133', 60544) payload
Received a message from client 1 payload xyz1
Sending a message to the client 1 payload b0400cf1f3919fdba057dbd2acc52f148fbb43ac
Connection to ec2-3-129-44-10.us-east-2.compute.amazonaws.com closed by remote host.
Connection to ec2-3-129-44-10.us-east-2.compute.amazonaws.com closed.
(base) Shantos-MacBook-Pro:HW3 roy$ 
```

```
(env) [ec2-user@ip-172-31-25-118 ~]$ python aws_server.py -id 200 -pp 111 -listen 5000 -revproc 8000

Server running with id 200
Server serving privacy policy 111
Listening on port 5000
Connecting to the reverse proxy on port 8000
Received a message from client ('172.31.21.133', 38214) payload
Received a message from client 2 payload xyz2
Sending a message to the client 2 payload fc7516b5c665d88cb7c39802367bb25a9c24bb47
Received a message from client ('172.31.21.133', 38218) payload
Received a message from client 2 payload xyz2
Sending a message to the client 2 payload fc7516b5c665d88cb7c39802367bb25a9c24bb47
Received a message from client ('172.31.21.133', 38222) payload
Received a message from client 2 payload xyz2
Sending a message to the client 2 payload fc7516b5c665d88cb7c39802367bb25a9c24bb47
Connection to ec2-3-139-94-216.us-east-2.compute.amazonaws.com closed by remote host.
Connection to ec2-3-139-94-216.us-east-2.compute.amazonaws.com closed.
(base) Shantos-MacBook-Pro:HW3 roy$ 
```

Figure 5: Reverse Proxy Simulation Output

A.6 Launching an AWS EC2 Instance



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Amazon Linux
Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0f57b4cec24530068 (64-bit x86) / ami-0264aa084692a7aa1 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select
● 64-bit (x86)
○ 64-bit (Arm)

macOS Big Sur 11.2.3 - ami-0ef98d91ff9126a43

The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select
64-bit (Mac)

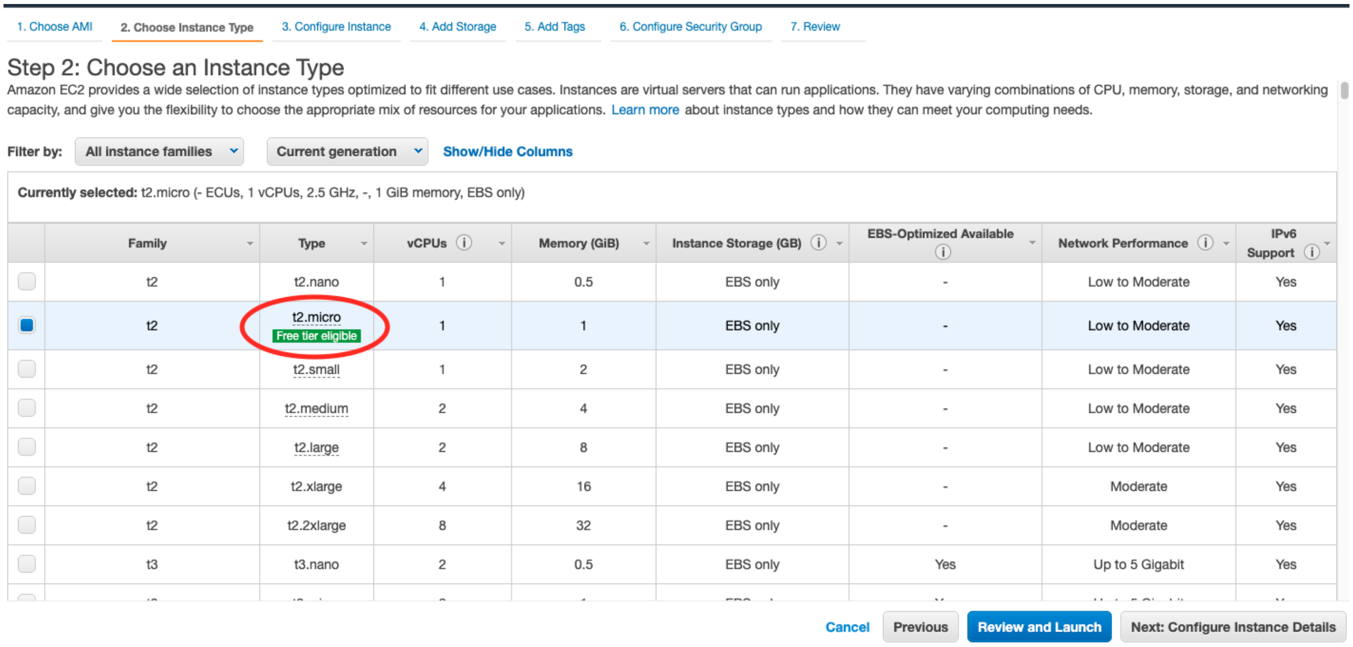
macOS Catalina 10.15.7 - ami-0e8dcb1f4e055d786

The macOS Catalina AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select
64-bit (Mac)

Figure 6: Step 1: Choose Machine Image (Free Tier)



Step 2: Choose an Instance Type

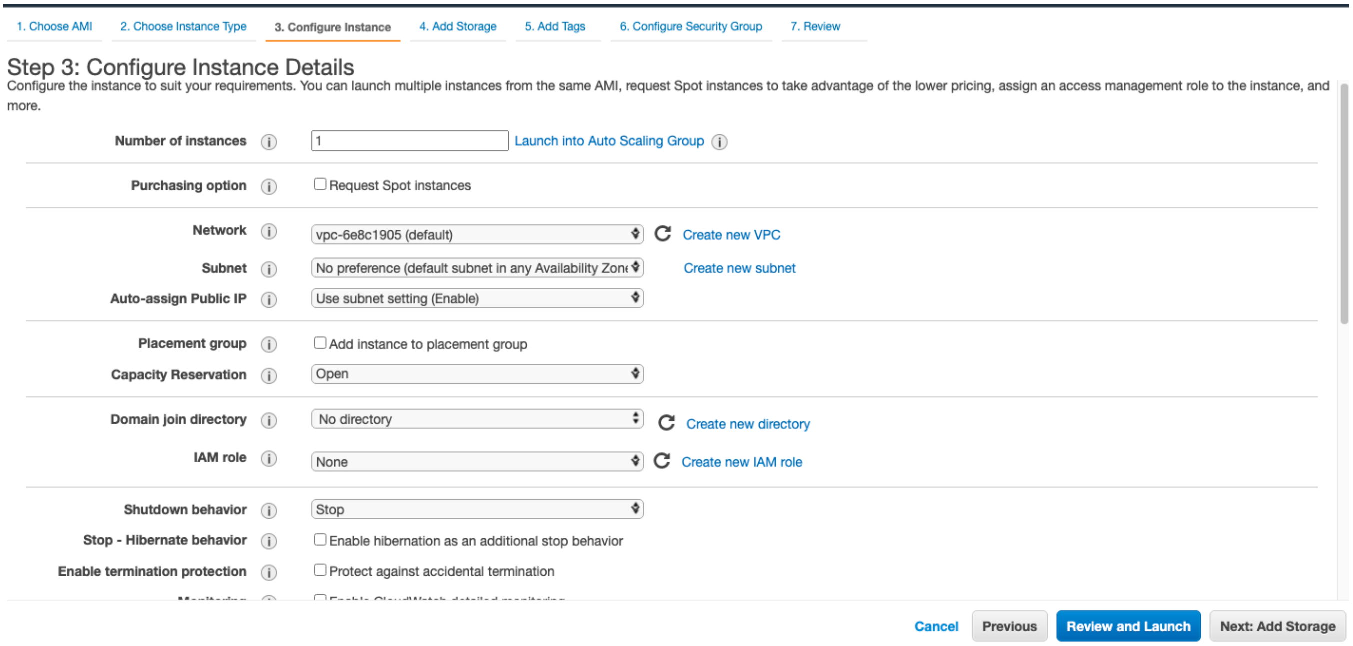
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance families Current generation Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
□	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
□	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
□	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
□	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Figure 7: Step 2: Choose Instance Type (Free Tier)



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1

Purchasing option: □ Request Spot instances

Network: vpc-6e8c1905 (default)

Subnet: No preference (default subnet in any Availability Zon

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: □ Add instance to placement group

Capacity Reservation: Open

Domain join directory: No directory

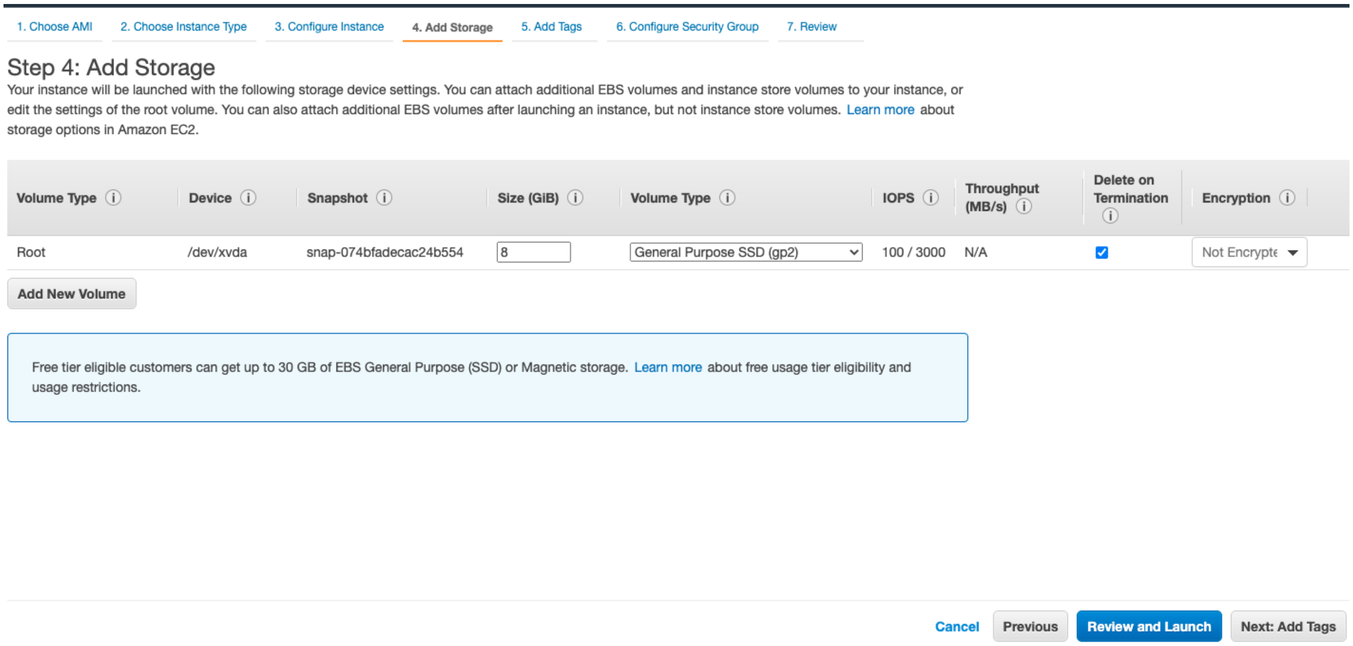
IAM role: None

Shutdown behavior: Stop

Stop - Hibernate behavior: □ Enable hibernation as an additional stop behavior

Enable termination protection: □ Protect against accidental termination

Figure 8: Step 3: Configure Instance Details



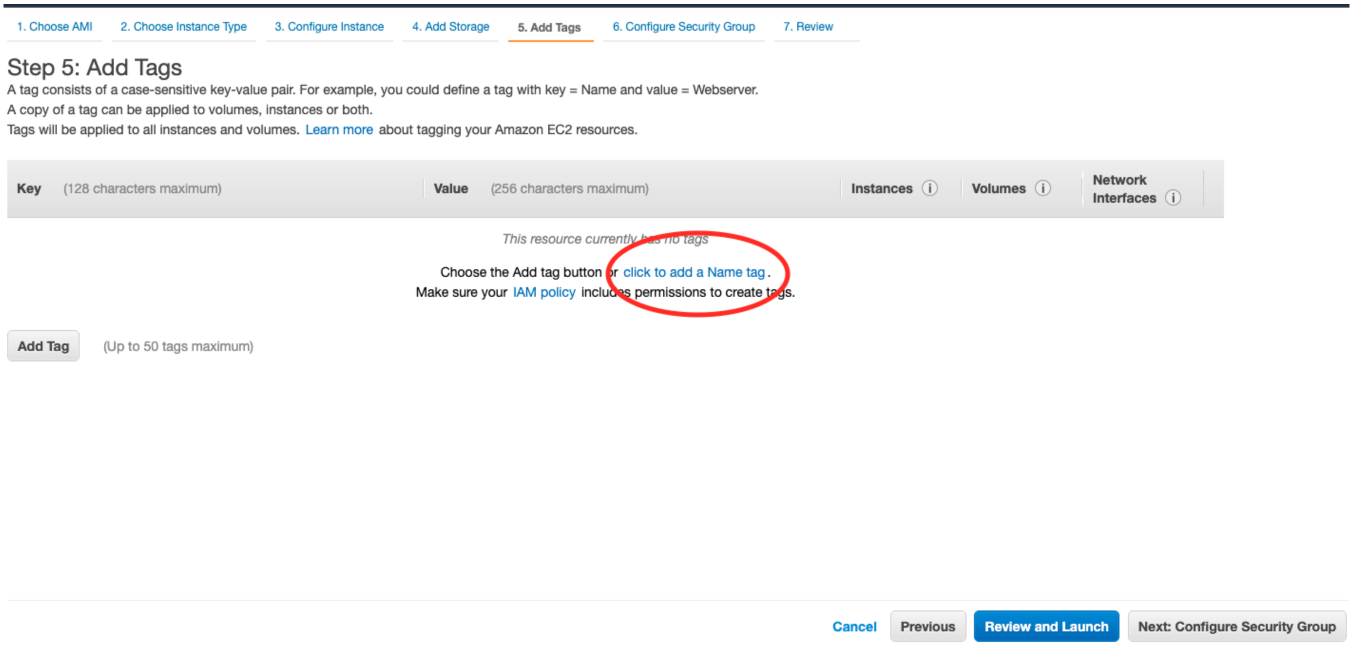
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-074bfadecac24b554	8	General Purpose SSD (gp2)	100 / 3000	N/A	☑	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

Figure 9: Step 4: Add Storage (used default)



Step 5: Add Tags

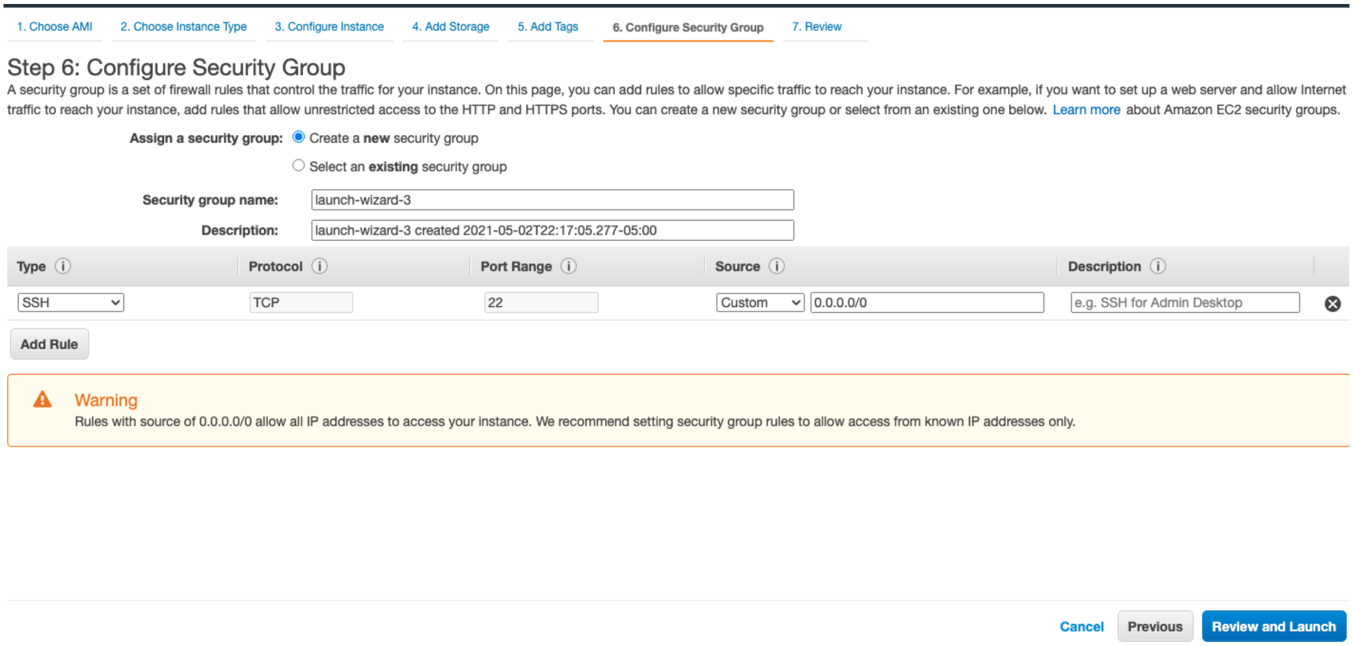
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes	Network Interfaces

This resource currently has no tags

Choose the Add tag button or click to add a Name tag.
Make sure your IAM policy includes permissions to create tags.

Figure 10: Step 5: Add a name tag



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

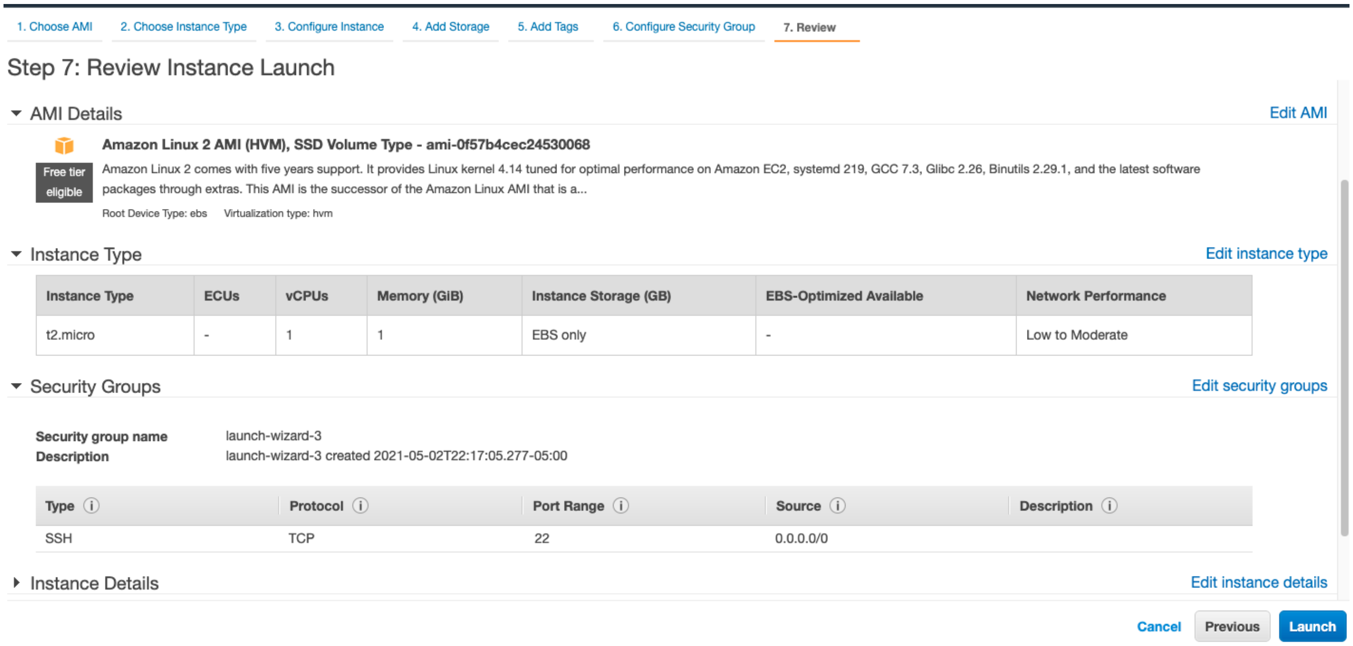
Assign a security group: ● Create a **new** security group
○ Select an **existing** security group

Security group name: launch-wizard-3
Description: launch-wizard-3 created 2021-05-02T22:17:05.277-05:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Figure 11: Step 6: Configure New or Existing Security Group



Step 7: Review Instance Launch

AMI Details — Edit AMI

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0f57b4cec24530068

Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a...

Root Device Type: ebs Virtualization type: hvm

Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

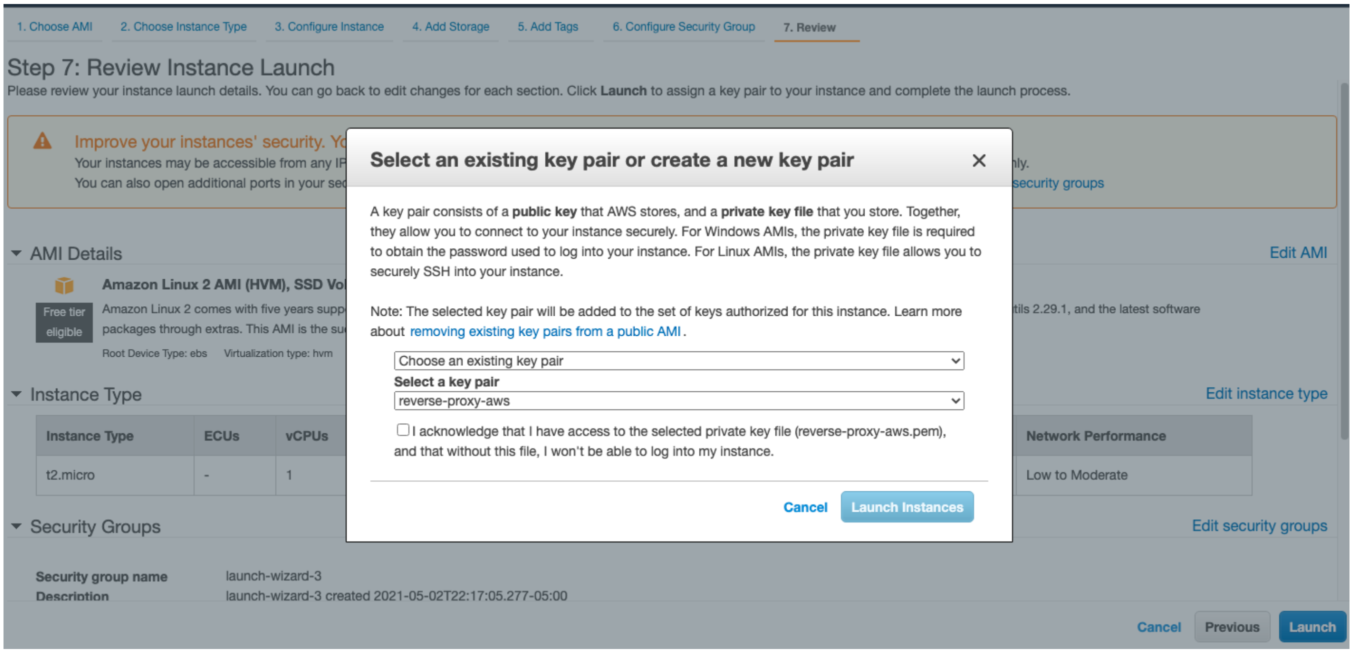
Security Groups — Edit security groups

Security group name launch-wizard-3
Description launch-wizard-3 created 2021-05-02T22:17:05.277-05:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	

Instance Details — Edit instance details

Figure 12: Step 7: Review Before Instance Launch



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Yo... ly.
Your instances may be accessible from any IP...
You can also open additional ports in your sec... security groups

Select an existing key pair or create a new key pair ✕

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Choose an existing key pair

Select a key pair
reverse-proxy-aws

☐ I acknowledge that I have access to the selected private key file (reverse-proxy-aws.pem), and that without this file, I won't be able to log into my instance.

AMI Details
Amazon Linux 2 AMI (HVM), SSD Vo...
Free tier eligible
Amazon Linux 2 comes with five years supp... ...tils 2.29.1, and the latest software
packages through extras. This AMI is the su...
Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Network Performance
t2.micro	-	1	Low to Moderate

Security Groups

Security group name launch-wizard-3
Description launch-wizard-3 created 2021-05-02T22:17:05.277-05:00

Figure 13: Step 8: Create new Key Pair or Use Existing one