

## SUMMARY:

This Lab covers EC2: SSH vs Instance Connect, Snapshots & Instance Stores, Manual WP on EC2, creating an AMI

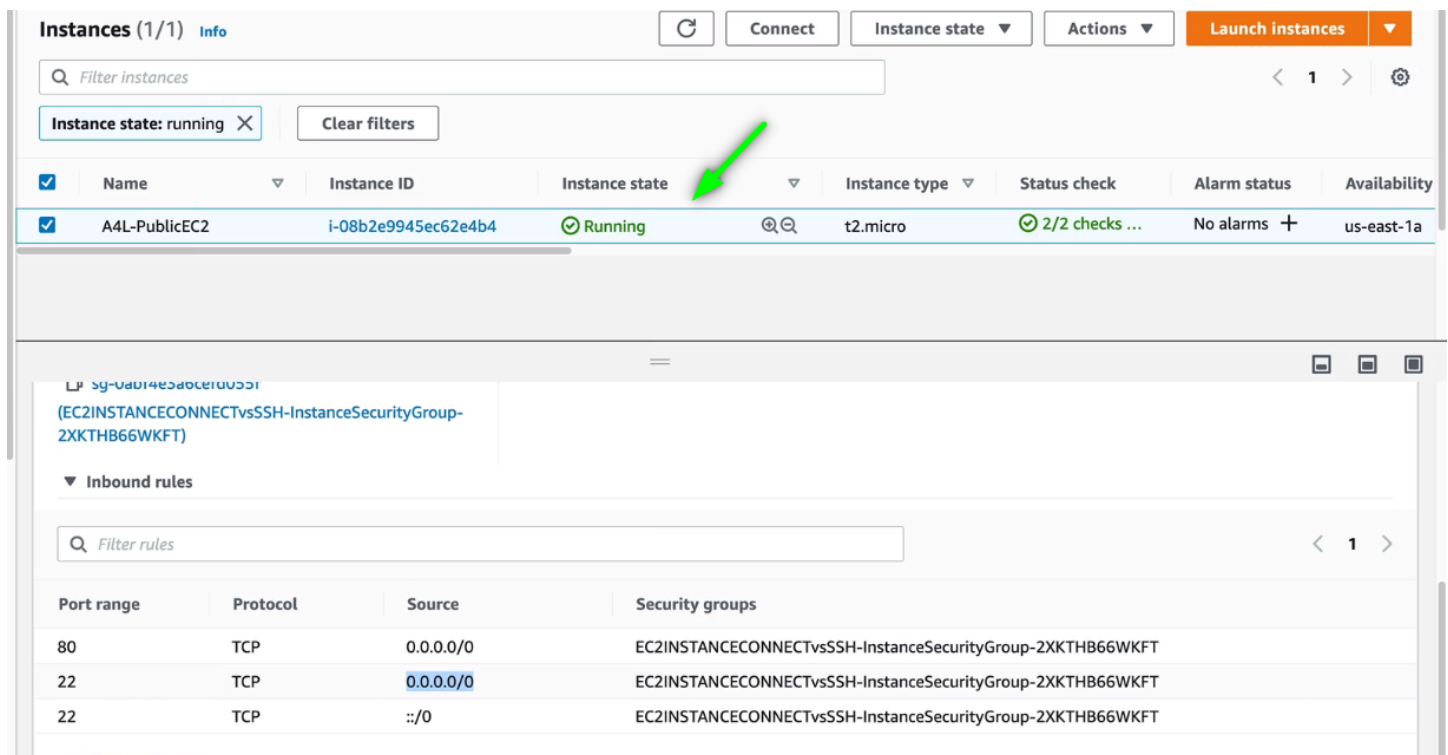
### Contents

1. EC2 SSH vs EC2 Instance Connect .....	1
2. EBS, Snapshots and Instance Store Volumes .....	4
3. Manual Install of Wordpress on EC2.....	10
4. Creating an Animals4life AMI.....	14

## 1. EC2 SSH vs EC2 Instance Connect

Amazon EC2 Instance Connect provides a simple and secure way to connect to your Linux instances using Secure Shell (SSH). With EC2 Instance Connect, you use AWS Identity and Access Management (IAM) policies and principals to control SSH access to your instances, removing the need to share and manage SSH keys.

Create instance:



The screenshot displays the AWS Management Console interface for EC2 instances. At the top, there's a header with 'Instances (1/1)' and buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below this is a search bar and a filter for 'Instance state: running'. The main table lists one instance: 'A4L-PublicEC2' with ID 'i-08b2e9945ec62e4b4', state 'Running' (highlighted with a green arrow), type 't2.micro', and status '2/2 checks...'. Below the table, the 'Inbound rules' for the security group 'EC2INSTANCECONNECTvsSSH-InstanceSecurityGroup-2XKTHB66WKFT' are shown. The rules table has columns for Port range, Protocol, Source, and Security groups. It lists three rules: port 80 (TCP, 0.0.0.0/0), port 22 (TCP, 0.0.0.0/0), and port 22 (TCP, ::/0).

Port range	Protocol	Source	Security groups
80	TCP	0.0.0.0/0	EC2INSTANCECONNECTvsSSH-InstanceSecurityGroup-2XKTHB66WKFT
22	TCP	0.0.0.0/0	EC2INSTANCECONNECTvsSSH-InstanceSecurityGroup-2XKTHB66WKFT
22	TCP	::/0	EC2INSTANCECONNECTvsSSH-InstanceSecurityGroup-2XKTHB66WKFT

Connect via SSH:

EC2 > Instances > i-08b2e9945ec62e4b4 > Connect to instance

### Connect to instance [Info](#)

Connect to your instance i-08b2e9945ec62e4b4 (A4L-PublicEC2) using any of these options

EC2 Instance Connect | Session Manager | **SSH client**

Instance ID  
i-08b2e9945ec62e4b4 (A4L-PublicEC2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is A4L.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
`chmod 400 A4L.pem`
4. Connect to your instance using its Public DNS:  
`ec2-3-235-51-2.compute-1.amazonaws.com`

Example:

```
ssh -i "A4L.pem" ec2-user@ec2-3-235-51-2.compute-1.amazonaws.com
```

**Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

```
> cd Downloads
> ssh -i "A4L.pem" ec2-user@ec2-3-235-51-2.compute-1.amazonaws.com
The authenticity of host 'ec2-3-235-51-2.compute-1.amazonaws.com (3.235.51.2)' can't be established.
ECDSA key fingerprint is SHA256:Jx/Eajrzqre+onuNJD3lykZaSFnKTzw7eG8uQuoLyUA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-235-51-2.compute-1.amazonaws.com,3.235.51.2' (ECDSA) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'A4L.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "A4L.pem": bad permissions
ec2-user@ec2-3-235-51-2.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
> chmod 400 A4L.pem
> ssh -i "A4L.pem" ec2-user@ec2-3-235-51-2.compute-1.amazonaws.com


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

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 5 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-16-55-5 ~]$
```

Connect via instance connect:

### Connect to instance [Info](#)


Connect to your instance i-08b2e9945ec62e4b4 (A4L-PublicEC2) using any of these options

EC2 Instance Connect


Session Manager

SSH client

Instance ID

 i-08b2e9945ec62e4b4 (A4L-PublicEC2)

Public IP address

 3.235.51.2

User name

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

**Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

```
 _ _ | ( _ _ )
 _ _ | ( _ _ /   Amazon Linux 2 AMI
 _ _ |\ _ _ | _ _ |

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 5 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-16-55-5 ~]$
```

Configure instance inbound rules to allow for interaction with SSH:

Inbound rules control the incoming traffic that's allowed to reach the instance.

### Inbound rules Info

Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>	
HTTP	TCP	80	Custom <input type="text" value="0.0.0.0/0"/>	Allow HTTP IPv4 IN	Delete
SSH	TCP	22	Custom <input type="text" value="18.206.107.24/29"/>	Allow SSH IPv4 IN	Delete
SSH	TCP	22	Custom <input type="text" value="::/0"/>	Allow SSH IPv6 IN	Delete

[Add rule](#)

**NOTE:** Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

[Cancel](#) [Preview changes](#) [Save rules](#)

```

--|  --|  )
_| (    /   Amazon Linux 2 AMI
---|\---|---|

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 5 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-16-55-5 ~]$
[ec2-user@ip-10-16-55-5 ~]$
[ec2-user@ip-10-16-55-5 ~]$
[ec2-user@ip-10-16-55-5 ~]$ exit
logout
Connection to ec2-3-235-51-2.compute-1.amazonaws.com closed.
> ssh -i "A4L.pem" ec2-user@ec2-3-235-51-2.compute-1.amazonaws.com

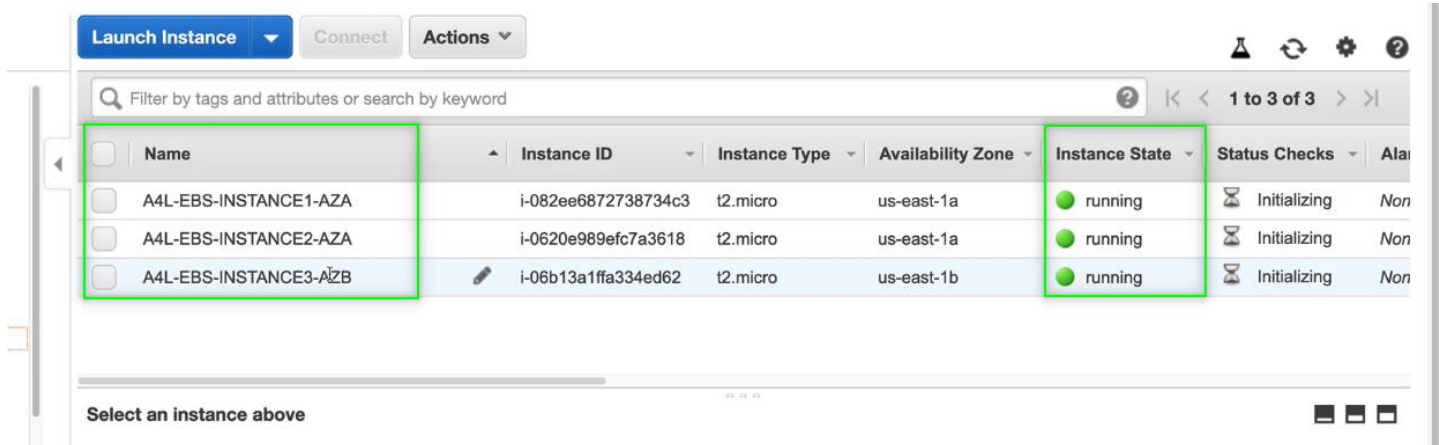
```

## 2. EBS, Snapshots and Instance Store Volumes

1. Create an EBS Volume
2. Mount it to an EC2 instance
3. Create and Mount a file system
4. Generate a test file
5. Migrate the volume to another EC2 instance in the same AZ
6. Verify the file system and file are intact
7. Create a EBS SNApshot from the volume
8. Create a new EBS Volume in AZ-B
9. Verify the filesystem and file are intact

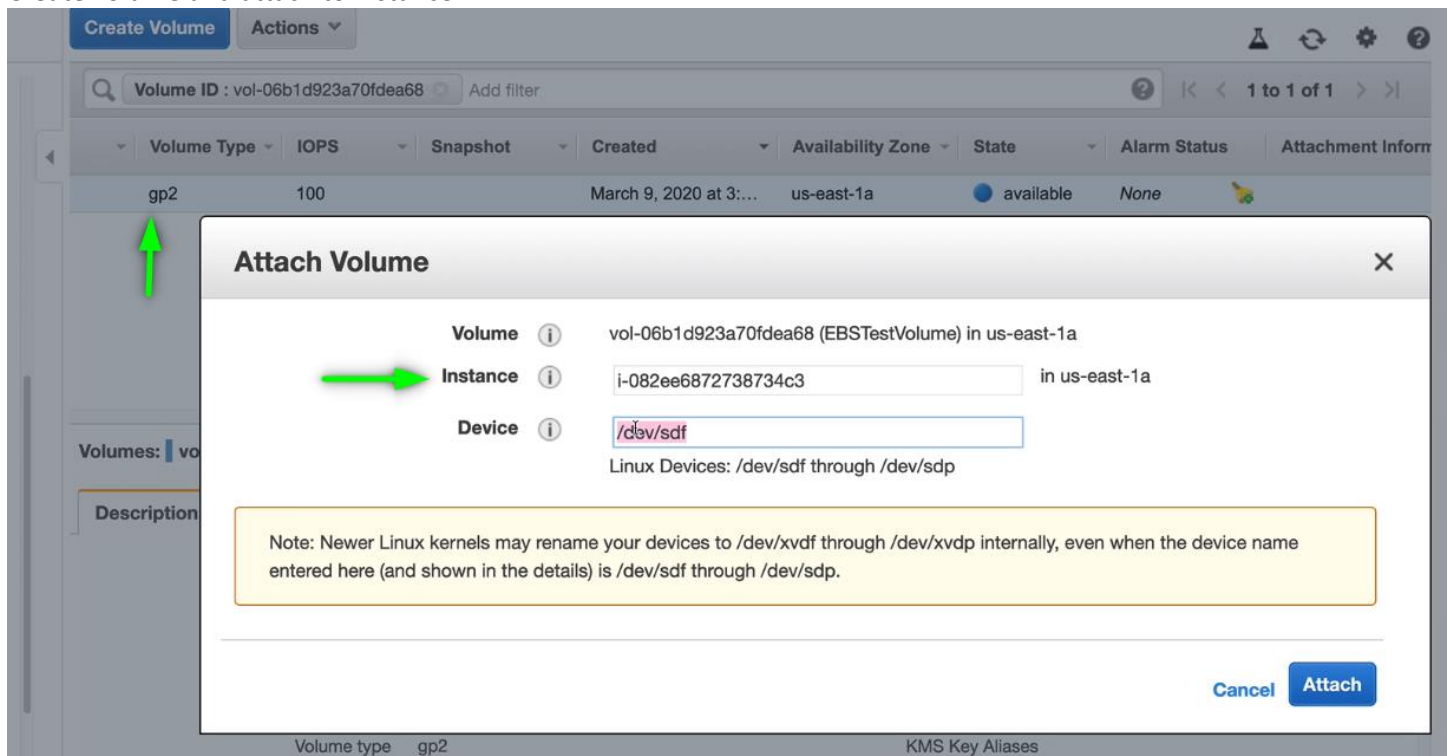
10. Copy the snapshot to another region
11. Create an EC2 instance with instance store volumes
12. Create a filesystem and test file
13. Restart instance and verify the file system is intact
14. Stop and Start the instance
15. Verify the file system is no longer present - new EC2 Host.

Create EC2 instances:



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm
A4L-EBS-INSTANCE1-AZA	i-082ee6872738734c3	t2.micro	us-east-1a	running	Initializing	Non
A4L-EBS-INSTANCE2-AZA	i-0620e989efc7a3618	t2.micro	us-east-1a	running	Initializing	Non
A4L-EBS-INSTANCE3-AZB	i-06b13a1ffa334ed62	t2.micro	us-east-1b	running	Initializing	Non

Create volume and attach to instance:



Volume ID : vol-06b1d923a70fdea68

Volume Type	IOPS	Snapshot	Created	Availability Zone	State	Alarm Status	Attachment Inform
gp2	100		March 9, 2020 at 3:...	us-east-1a	available	None	

**Attach Volume**

Volume *i* vol-06b1d923a70fdea68 (EBSTestVolume) in us-east-1a

Instance *i* i-082ee6872738734c3 in us-east-1a

Device *i* /dev/sdf

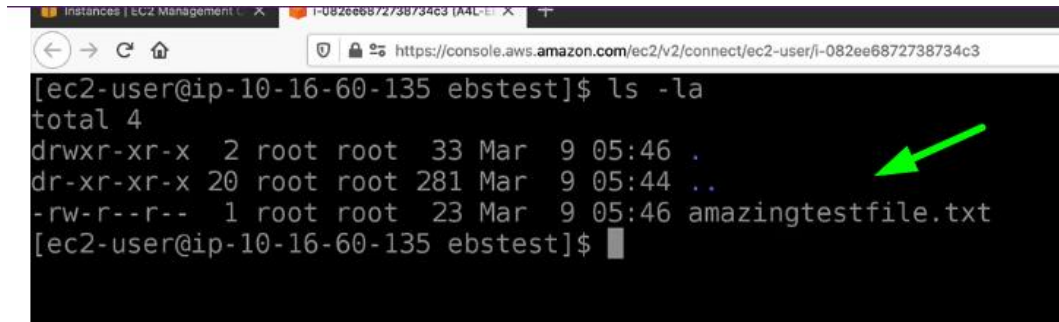
Linux Devices: /dev/sdf through /dev/sdp

Note: Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdp internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.

Cancel Attach

Create test file in instance connect:

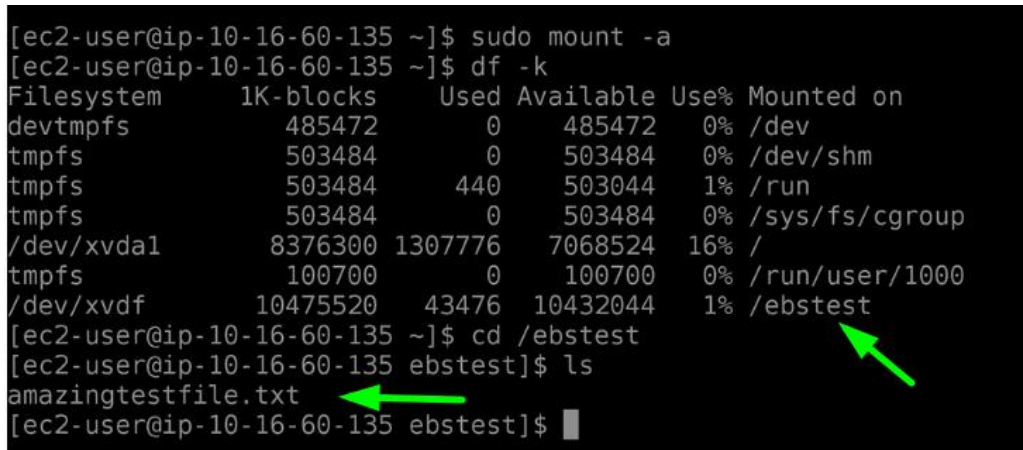




```
[ec2-user@ip-10-16-60-135 ebstest]$ ls -la
total 4
drwxr-xr-x  2 root root  33 Mar  9 05:46 .
dr-xr-xr-x 20 root root 281 Mar  9 05:44 ..
-rw-r--r--  1 root root 23 Mar  9 05:46 amazingtestfile.txt
[ec2-user@ip-10-16-60-135 ebstest]$
```

A green arrow points to the file listing output.

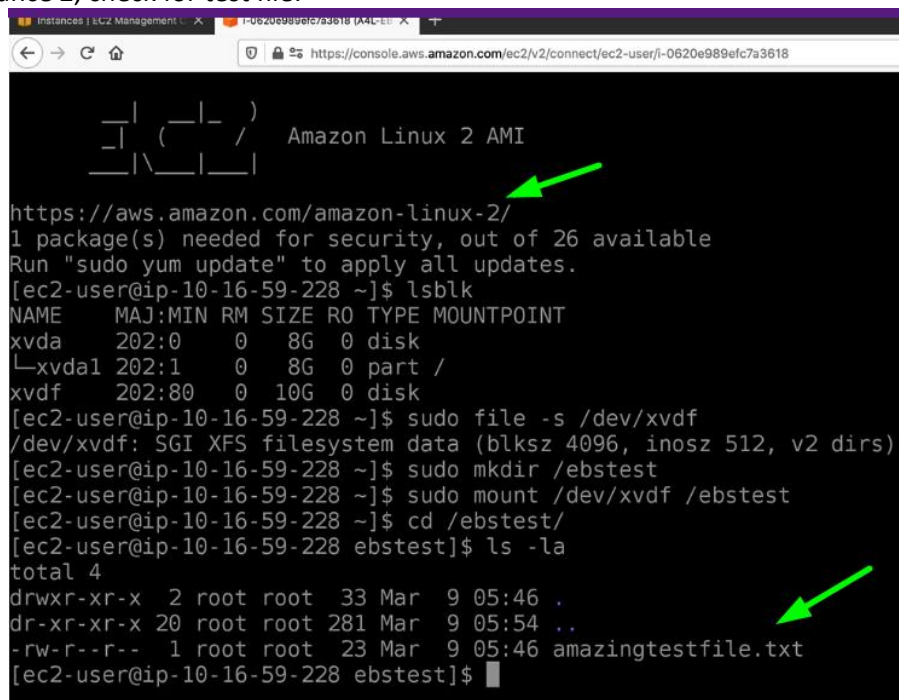
Confirm file in EBS:



```
[ec2-user@ip-10-16-60-135 ~]$ sudo mount -a
[ec2-user@ip-10-16-60-135 ~]$ df -k
Filesystem      1K-blocks    Used Available Use% Mounted on
devtmpfs        485472         0   485472    0% /dev
tmpfs           503484         0   503484    0% /dev/shm
tmpfs           503484       440   503044    1% /run
tmpfs           503484         0   503484    0% /sys/fs/cgroup
/dev/xvda1      8376300 1307776  7068524   16% /
tmpfs          100700         0   100700    0% /run/user/1000
/dev/xvdf       10475520   43476 10432044    1% /ebstest
[ec2-user@ip-10-16-60-135 ~]$ cd /ebstest
[ec2-user@ip-10-16-60-135 ebstest]$ ls
amazingtestfile.txt
[ec2-user@ip-10-16-60-135 ebstest]$
```

Two green arrows point to the output of the 'df -k' command and the file listing in the /ebstest directory.

Attach volume to instance 2, check for test file:



```

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

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 26 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-16-59-228 ~]$ lsblk
NAME        MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda        202:0    0  8G  0 disk
└─xvda1     202:1    0  8G  0 part /
xvdf        202:80    0 10G  0 disk
[ec2-user@ip-10-16-59-228 ~]$ sudo file -s /dev/xvdf
/dev/xvdf: SGI XFS filesystem data (blkisz 4096, inosz 512, v2 dirs)
[ec2-user@ip-10-16-59-228 ~]$ sudo mkdir /ebstest
[ec2-user@ip-10-16-59-228 ~]$ sudo mount /dev/xvdf /ebstest
[ec2-user@ip-10-16-59-228 ~]$ cd /ebstest/
[ec2-user@ip-10-16-59-228 ebstest]$ ls -la
total 4
drwxr-xr-x  2 root root  33 Mar  9 05:46 .
dr-xr-xr-x 20 root root 281 Mar  9 05:54 ..
-rw-r--r--  1 root root 23 Mar  9 05:46 amazingtestfile.txt
[ec2-user@ip-10-16-59-228 ebstest]$
```

Two green arrows point to the Amazon Linux 2 AMI logo and the file listing output.

Create instance store:

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

## 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-03a475f314b035167	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted
ephemeral0	/dev/nvme0n1	N/A	75	NVMe SSD	N/A	N/A	N/A	Hardware Encrypted

Add New Volume

Launch Instance

Connect

Actions

search : i-0924d6e8ce11d7a91 Add filter

1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm
InstanceStoreTest	i-0924d6e8ce11d7a91	m5dn.large	us-east-1a	running	Initializing	Non

Instance: i-0924d6e8ce11d7a91 (InstanceStoreTest) Public DNS: ec2-52-206-106-102.compute-1.amazonaws.com

Description

Status Checks

Monitoring

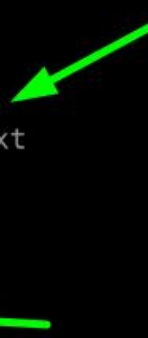
Tags

Instance ID	i-0924d6e8ce11d7a91
Instance state	running
Instance type	m5dn.large
Finding	Opt-in to AWS Compute Optimizer for recommendations. <a href="#">Learn more</a>
Private DNS	ip-10-16-53-43.ec2.internal
Private IPs	10.16.53.43

Public DNS (IPv4)	ec2-52-206-106-102.compute-1.amazonaws.com
IPv4 Public IP	52.206.106.102
IPv6 IPs	2600:1f18:dfc:ed03:5929:bc41:2836:d591
Elastic IPs	
Availability zone	us-east-1a
Security groups	<a href="#">A4LEBS-InstanceSecurityGroup-1NE5M1U5NJ84V</a> , <a href="#">view inbound</a>

Create test file in instance store connect:

```
nvme0n1p1 259:2 0 8G 0 part /
nvme0n1p128 259:3 0 1M 0 part
[ec2-user@ip-10-16-53-43 ~]$ sudo file -s /dev/nvme1n1
/dev/nvme1n1: data
[ec2-user@ip-10-16-53-43 ~]$ sudo mkfs -t xfs /dev/nvme1n1
meta-data=/dev/nvme1n1 isize=512 agcount=4, agsize=4577637 blks
          = sectsz=512 attr=2, projid32bit=1
          = crc=1 finobt=1, sparse=0
data      = bsize=4096 blocks=18310546, imaxpct=25
          = sunit=0 swidth=0 blks
naming    =version 2 bsize=4096 ascii-ci=0 ftype=1
log       =internal log bsize=4096 blocks=8940, version=2
          = sectsz=512 sunit=0 blks, lazy-count=1
realtime  =none extsz=4096 blocks=0, rtextents=0
[ec2-user@ip-10-16-53-43 ~]$ sudo file -s /dev/nvme1n1
/dev/nvme1n1: SGI XFS filesystem data (blksz 4096, inosz 512, v2 dirs)
[ec2-user@ip-10-16-53-43 ~]$ sudo mkdir /instancestore
[ec2-user@ip-10-16-53-43 ~]$ sudo mount /dev/nvme1n1 /instancestore
[ec2-user@ip-10-16-53-43 ~]$ cd /instancestore/
[ec2-user@ip-10-16-53-43 instancestore]$ ls -la
total 0
drwxr-xr-x  2 root root  6 Mar  9 06:13 .
dr-xr-xr-x 20 root root 287 Mar  9 06:14 ..
[ec2-user@ip-10-16-53-43 instancestore]$ sudo touch instancestore.txt
[ec2-user@ip-10-16-53-43 instancestore]$ ls -la
total 0
drwxr-xr-x  2 root root 31 Mar  9 06:14 .
dr-xr-xr-x 20 root root 287 Mar  9 06:14 ..
-rw-r--r--  1 root root  0 Mar  9 06:14 instancestore.txt
[ec2-user@ip-10-16-53-43 instancestore]$
```



Reboot instance – then check for file, file should still be there. Stopping the instance store however will clear everything, instance stores are ephemeral storage – very fast, but not durable.



Launch Instance Connect Actions

search : i-0924d6e8ce11d7a91 Add filter

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm
InstanceStore	i-0924d6e8ce11d7a91	m5dn.large	us-east-1a	running	Initializing	Non

Instance: i-0924d6e8ce11d7a91

52-206-106-102.compute-1.amazonaws.com

Instance ID i-0924d6e8ce11d7a91

Instance state running

Instance type m5dn.large

Finding Opt-in to AWS Compute Optimizer for recommendations. [Learn more](#)

Private DNS ip-10-16-53-43.ec2.internal

Private IPs 10.16.53.43

Public DNS (IPv4) ec2-52-206-106-102.compute-1.amazonaws.com

IPv4 Public IP 52.206.106.102

IPv6 IPs 2600:1f18:dfc:ed03:5929:bc41:2836:d591

Elastic IPs

Availability zone us-east-1a

Security groups [A4LEBS-InstanceSecurityGroup-1NE5M1U5NJ84V](#). [view inbound](#)

```

  _ | ( _ | )
  _ | ( _ | /
  _ | \ _ | _ |

Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 26 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-16-53-43 ~]$ df -k
Filesystem      1K-blocks      Used Available Use% Mounted on
devtmpfs        3957512         0   3957512   0% /dev
tmpfs           3957524         0   3957524   0% /dev/shm
tmpfs           3957524        424   3975100   1% /run
tmpfs           3957524         0   3975524   0% /sys/fs/cgroup
/dev/nvme0n1p1  8376300 1307688   7068612  16% /
tmpfs           795108         0    795108   0% /run/user/1000
[ec2-user@ip-10-16-53-43 ~]$ sudo mount /dev/nvme1n1 /instancestore
[ec2-user@ip-10-16-53-43 ~]$ cd /instancestore/
[ec2-user@ip-10-16-53-43 instancestore]$ ls
instancestore.txt
[ec2-user@ip-10-16-53-43 instancestore]$
  
```

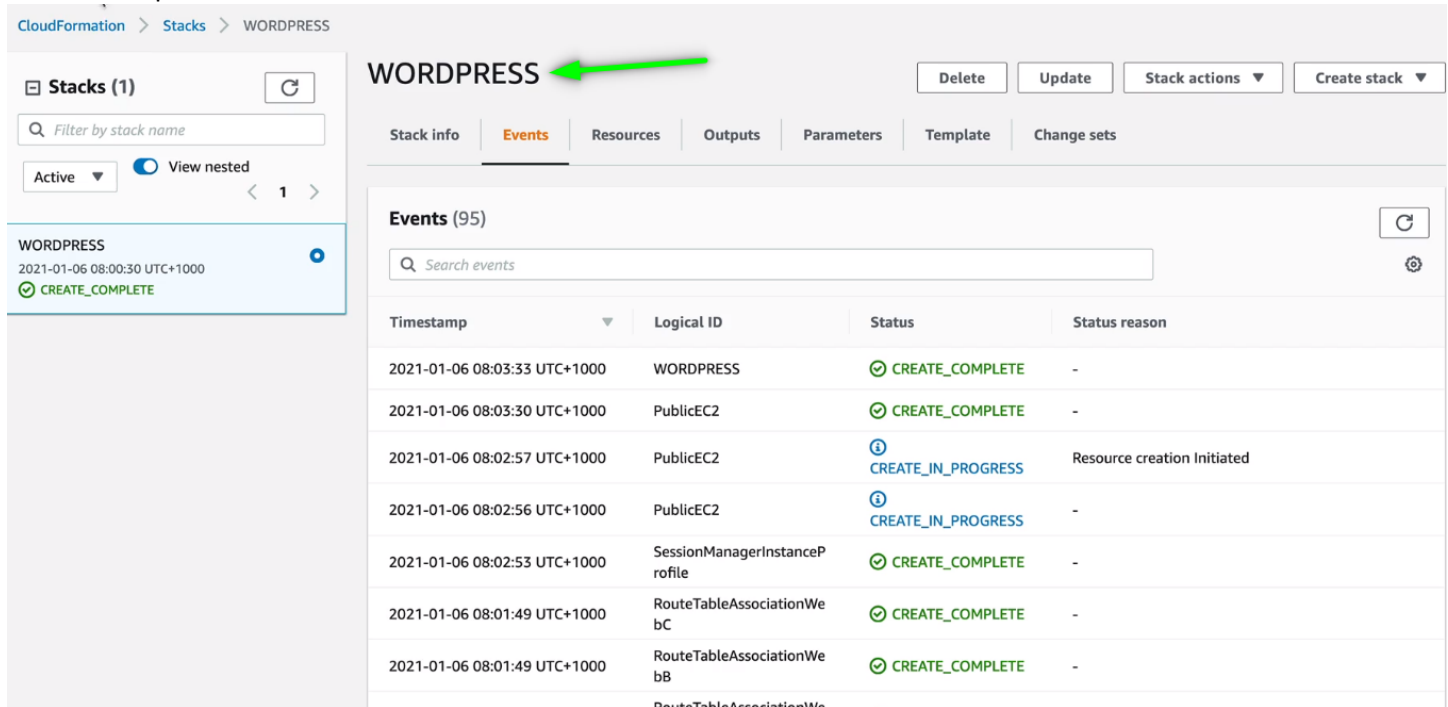
### 3. Manual Install of Wordpress on EC2

- In this [DEMO] lesson you will install wordpress on an EC2 instance.
- To appreciate the automation and efficiency which can be achieved within AWS you first need to experience the process manually.
- We will use EC2, install MariaDB, Apache & libraries and then download and install wordpress.

Commands for Manual WP install:

```
# DBName=database name for wordpress
# DBUser=mariadb user for wordpress
# DBPassword=password for the mariadb user for wordpress
# DBRootPassword = root password for mariadb
# STEP 1 - Configure Authentication Variables which are used below
DBName='a4lwordpress'
DBUser='a4lwordpress'
DBPassword='REPLACEME'
DBRootPassword='REPLACEME'
# STEP 2 - Install system software - including Web and DB
sudo yum install -y mariadb-server httpd wget
sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
# STEP 3 - Web and DB Servers Online - and set to startup
sudo systemctl enable httpd
sudo systemctl enable mariadb
sudo systemctl start httpd
sudo systemctl start mariadb
# STEP 4 - Set Mariadb Root Password
mysqladmin -u root password $DBRootPassword
# STEP 5 - Install Wordpress
sudo wget http://wordpress.org/latest.tar.gz -P /var/www/html
cd /var/www/html
sudo tar -zxvf latest.tar.gz
sudo cp -rvf wordpress/* .
sudo rm -R wordpress
sudo rm latest.tar.gz
# STEP 6 - Configure Wordpress
sudo cp ./wp-config-sample.php ./wp-config.php
sudo sed -i "s/'database_name_here'/'$DBName'/g" wp-config.php
sudo sed -i "s/'username_here'/'$DBUser'/g" wp-config.php
sudo sed -i "s/'password_here'/'$DBPassword'/g" wp-config.php
sudo chown apache:apache * -R
# STEP 7 Create Wordpress DB
echo "CREATE DATABASE $DBName;" >> /tmp/db.setup
echo "CREATE USER '$DBUser'@'localhost' IDENTIFIED BY '$DBPassword';" >> /tmp/db.setup
echo "GRANT ALL ON $DBName.* TO '$DBUser'@'localhost';" >> /tmp/db.setup
echo "FLUSH PRIVILEGES;" >> /tmp/db.setup
mysql -u root --password=$DBRootPassword < /tmp/db.setup
sudo rm /tmp/db.setup
# STEP 8 - Browse to http://your\_instance\_public\_ipv4\_ip
```

## Create wordpress stack:



CloudFormation > Stacks > WORDPRESS

**Stacks (1)**

Filter by stack name

Active View nested

WORDPRESS

2021-01-06 08:00:30 UTC+1000

CREATE\_COMPLETE

**WORDPRESS**

Delete Update Stack actions Create stack

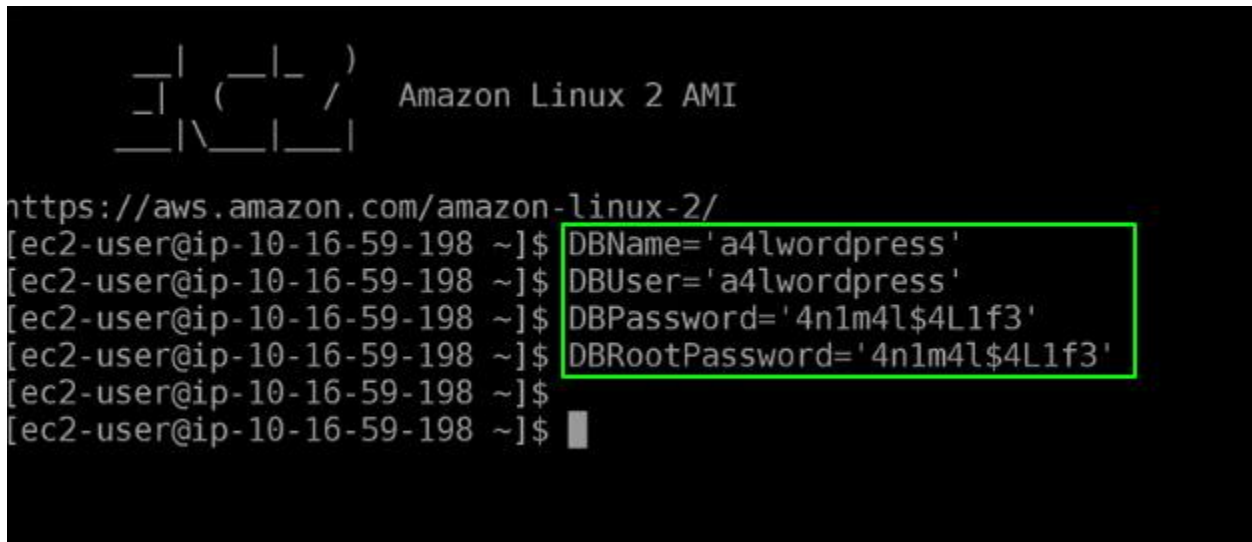
Stack info Events Resources Outputs Parameters Template Change sets

**Events (95)**

Search events

Timestamp	Logical ID	Status	Status reason
2021-01-06 08:03:33 UTC+1000	WORDPRESS	CREATE_COMPLETE	-
2021-01-06 08:03:30 UTC+1000	PublicEC2	CREATE_COMPLETE	-
2021-01-06 08:02:57 UTC+1000	PublicEC2	CREATE_IN_PROGRESS	Resource creation Initiated
2021-01-06 08:02:56 UTC+1000	PublicEC2	CREATE_IN_PROGRESS	-
2021-01-06 08:02:53 UTC+1000	SessionManagerInstanceProfile	CREATE_COMPLETE	-
2021-01-06 08:01:49 UTC+1000	RouteTableAssociationWebC	CREATE_COMPLETE	-
2021-01-06 08:01:49 UTC+1000	RouteTableAssociationWebB	CREATE_COMPLETE	-
2021-01-06 08:01:49 UTC+1000	RouteTableAssociationWebA	CREATE_COMPLETE	-

## Set variables in instance:

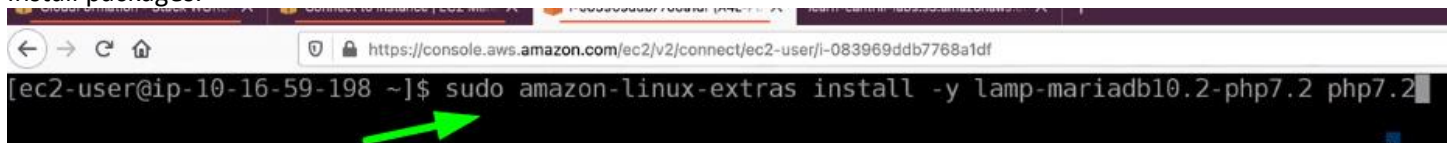


```
Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/

[ec2-user@ip-10-16-59-198 ~]$ DBName='a4lwordpress'
[ec2-user@ip-10-16-59-198 ~]$ DBUser='a4lwordpress'
[ec2-user@ip-10-16-59-198 ~]$ DBPassword='4n1m4l$4L1f3'
[ec2-user@ip-10-16-59-198 ~]$ DBRootPassword='4n1m4l$4L1f3'
[ec2-user@ip-10-16-59-198 ~]$
```

## Install packages:



```
[ec2-user@ip-10-16-59-198 ~]$ sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
```

## Tar the wordpress .gz files

```
[ec2-user@ip-10-16-59-198 ~]$ mysqladmin -u root password $DBRootPassword
[ec2-user@ip-10-16-59-198 ~]$ sudo wget http://wordpress.org/latest.tar.gz -P /var/www/html
--2021-01-05 22:18:45-- http://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://wordpress.org/latest.tar.gz [following]
--2021-01-05 22:18:45-- https://wordpress.org/latest.tar.gz
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15422346 (15M) [application/octet-stream]
Saving to: '/var/www/html/latest.tar.gz'

100%[=====>] 15,422,346 18.8MB/s in 0.8s

2021-01-05 22:18:46 (18.8 MB/s) - '/var/www/html/latest.tar.gz' saved [15422346/15422346]

[ec2-user@ip-10-16-59-198 ~]$ cd /var/www/html
[ec2-user@ip-10-16-59-198 html]$ sudo tar -zxvf latest.tar.gz
```

Confirm files in instance:

```
[ec2-user@ip-10-16-59-198 html]$ ls -la
total 216
drwxr-xr-x  5 root root  4096 Jan  5 22:21 .
drwxr-xr-x  4 root root    33 Jan  5 22:12 ..
-rw-r--r--  1 root root   405 Jan  5 22:20 index.php
-rw-r--r--  1 root root 19915 Jan  5 22:20 license.txt
-rw-r--r--  1 root root  7278 Jan  5 22:20 readme.html
-rw-r--r--  1 root root  7101 Jan  5 22:20 wp-activate.php
drwxr-xr-x  9 root root  4096 Jan  5 22:20 wp-admin
-rw-r--r--  1 root root   351 Jan  5 22:20 wp-blog-header.php
-rw-r--r--  1 root root  2328 Jan  5 22:20 wp-comments-post.php
-rw-r--r--  1 root root  2913 Jan  5 22:20 wp-config-sample.php
drwxr-xr-x  4 root root    52 Jan  5 22:20 wp-content
-rw-r--r--  1 root root  3939 Jan  5 22:20 wp-cron.php
drwxr-xr-x 25 root root  8192 Jan  5 22:20 wp-includes
-rw-r--r--  1 root root  2496 Jan  5 22:20 wp-links-opml.php
-rw-r--r--  1 root root  3300 Jan  5 22:20 wp-load.php
-rw-r--r--  1 root root 49831 Jan  5 22:20 wp-login.php
-rw-r--r--  1 root root  8509 Jan  5 22:20 wp-mail.php
-rw-r--r--  1 root root 20975 Jan  5 22:20 wp-settings.php
-rw-r--r--  1 root root 31337 Jan  5 22:20 wp-signup.php
-rw-r--r--  1 root root  4747 Jan  5 22:20 wp-trackback.php
-rw-r--r--  1 root root  3236 Jan  5 22:20 xmlrpc.php
[ec2-user@ip-10-16-59-198 html]$
```

Finish wordpress setup:

```
[ec2-user@ip-10-16-59-198 html]$ echo "CREATE DATABASE $DBName;" >> /tmp/db.setup
[ec2-user@ip-10-16-59-198 html]$ echo "CREATE USER '$DBUser'@'localhost' IDENTIFIED BY '$DBPassword';" >> /tmp/db.setup
[ec2-user@ip-10-16-59-198 html]$ echo "GRANT ALL ON $DBName.* TO '$DBUser'@'localhost';" >> /tmp/db.setup
[ec2-user@ip-10-16-59-198 html]$ echo "FLUSH PRIVILEGES;" >> /tmp/db.setup
[ec2-user@ip-10-16-59-198 html]$ cat /tmp/db.setup
CREATE DATABASE a4lwordpress;
CREATE USER 'a4lwordpress'@'localhost' IDENTIFIED BY '4n1m4l$4L1f3';
GRANT ALL ON a4lwordpress.* TO 'a4lwordpress'@'localhost';
FLUSH PRIVILEGES;
[ec2-user@ip-10-16-59-198 html]$ mysql -u root --password=$DBRootPassword < /tmp/db.setup
[ec2-user@ip-10-16-59-198 html]$ sudo rm /tmp/db.setup
```

Go test instance, use IP address in browser:

The screenshot shows the AWS Management Console interface for an EC2 instance. At the top, there's a navigation bar with 'Instances (1/1)' and a search bar. Below this is a table listing instances. The instance 'A4L-PublicEC2' with ID 'i-083969ddb7768a1df' is highlighted. A green arrow points to this instance. Below the table, the 'Details' tab is selected for the instance. The 'Instance summary' section shows the instance ID, state (Running), and type (t2.micro). To the right, the 'Public IPv4 address' is shown as '3.235.130.76' with a green arrow pointing to it. Other details include 'Public IPv4 DNS', 'Elastic IP addresses', 'Private IPv4 addresses', 'Private IPv4 DNS', and 'VPC ID'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publ
A4L-PublicEC2	i-083969ddb7768a1df	Running	t2.micro	2/2 checks ...	No alarms +	us-east-1a	ec2-

Instance: i-083969ddb7768a1df (A4L-PublicEC2)

Details | Security | Networking | Storage | Status Checks | Monitoring | Tags

▼ Instance summary Info

Instance ID  
i-083969ddb7768a1df (A4L-PublicEC2)

Instance state  
Running

Instance type  
t2.micro

Public IPv4 address  
3.235.130.76 | open address

Public IPv4 DNS  
ec2-3-235-130-76.compute-1.amazonaws.com | open address

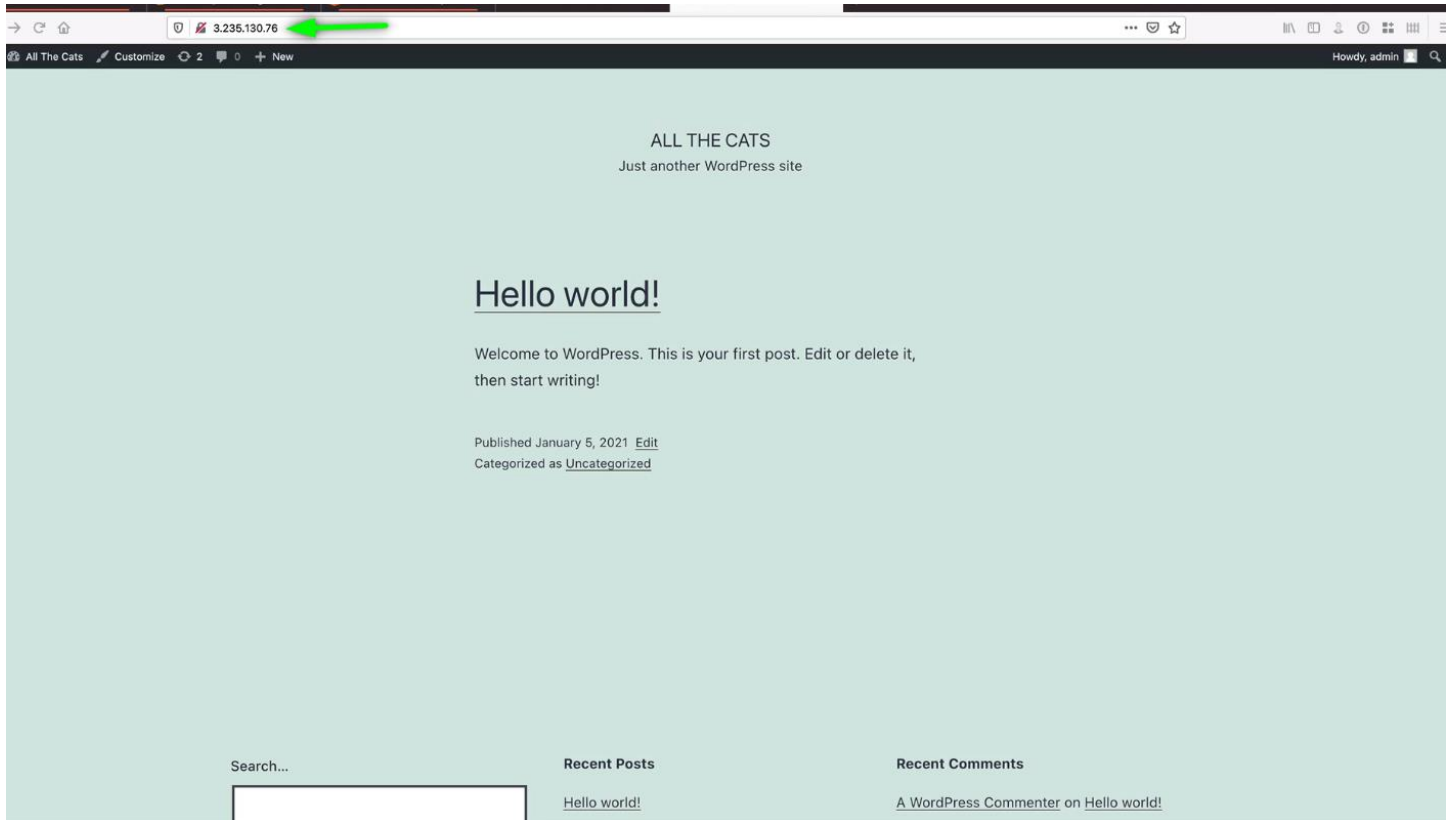
Elastic IP addresses  
-

Private IPv4 addresses  
10.16.59.198

Private IPv4 DNS  
ip-10-16-59-198.ec2.internal

VPC ID  
vpc-0d7152035817375dc (a4l-vpc1)





#### 4. Creating an Animals4life AMI

- In this [DEMO] after recovering from the announcement that you will AGAIN have to install wordpress manually on EC2 .. you create a customized EC2 instance which has wordpress installed and configured right up to the 'create site' stage.
- Additionally you improve the EC2 login screen by replacing the usual banned, with one provided by `cowsay` (It's animal themed !!)
- Once the EC2 instance is ready - you will create an AMI from the customized source instance and use this to deploy a custom EC2 instance from this AMI.
- Its a simple example - but mirrors real world usage of AMI Baking.

Pause WP instance, create image, snapshot, and AMI:

Image name

Animals4lifeTemplateWordpress

Maximum 127 characters. Can't be modified after creation.

Image description - optional

Animals4lifeTemplateWordpress

Maximum 255 characters

No reboot

☐ Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/x...	Create new snapshot fr...	8	EBS General Purpose SS...	100		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

Add volume

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

☒ Tag image and snapshots together

Tag the image and the snapshots with the same tag.

☐ Tag image and snapshots separately

Tag the image and the snapshots with different tags.

Create Snapshot

Actions

Owned By Me

Filter by tags and attributes or search by keyword

	Name	Snapshot ID	Size	Description	Status
<input checked="" type="checkbox"/>		snap-0778d5de87e...	8 GiB	Created by CreateImage(i-0fe4835fbab21eacd) for ami-0f85b8...	complet

Snapshot: snap-0778d5de87ea72968

Description

Permissions

Tags

Snapshot ID

snap-0778d5de87ea72968

Progress

100%

Status

completed

Capacity

8 GiB

Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts New

Scheduled Instances

Capacity Reservations

▼ Images

AMIs

▼ Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Launch

EC2 Image Builder

Actions ▼

Owned by me ▼

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name	AMI Name	AMI ID	Source	Owner	Visibility	Status
<input checked="" type="checkbox"/>		Animals4lifeTe...	ami-0f85b801cf2405633	945690336440/...	945690336440	Private	available