

School: CLOUDBOOSTA	Course: Cloud Computing & DevOps
Name: AKOSA SAMUEL ONYEJEKWE	Module: Networking on Cloud
	Class 2: Assignment 2b_ Installing Jenkins Server on an EC2 Instance
Student No: CBALC0168	Issue Date: March 1, 2026

Required tasks:

1. Set up a custom VPC (Virtual Private Cloud) and its components (Subnets, NACL (Network Access Control List), Security Group, Route Table, and IGW (Internet Gateway)).
2. Provision an Amazon EC2 Instance on the VPC.
3. Launch the EC2 Instance.
4. Install Jenkins Server on the EC2 instance.

Result:

Figure A0 presents screengrabs confirming that the Jenkins tool has successfully been installed on the EC2 instance.

Figure A0: Panel showing Jenkins has been installed on EC2 instance





Getting Started

Create First Admin User

Username

Password

Confirm password

Jenkins 2.541.2

[Skip and continue as admin](#) [Save and Continue](#)

Information and Explanation for this demonstration:

- Include an additional Security group rule (Open port 8080¹).
- The Availability Region is London, eu-west-2, in Europe.
- Two subnets (1 private, and 1 public) are in 1 Availability Zone.
- 1 Internet Gateway (IGW)
- 1 EC2 instance will be provisioned on the Public Subnet.

¹ Port 8080 is also known as Alternative HTTP/Web App/Proxy/Application server port.

- The VPC has IPv4 and IPv6 CIDR blocks².
- The VPC is assigned IPv4 CIDR block, 10.0.0.0/16, and an IPv6 CIDR ³ provided by Amazon.
- The following are CIDR block sizes assigned to the Private Subnet, and Public Subnet.

Table 1: CIDR Block sizes assigned to the Private Subnet and Public Subnet

Subnet	CIDR Block size
Private	10.0.128.0/20
Public	10.0.0.0/20

- EC2 instance in Public Subnet has IPv4 and IPv6 addresses (both public).
- EC2 instance accesses the internet via IPv6.
- Every resource deployed within the Public Subnet would have a Public IP⁴.
- Our case study allows no egress-only IGW (Internet gateway)⁵.
- Zero EIGW was enabled in this exercise. (IPv4 always allows zero EIGW; IPv6 allows zero EIGW sometimes).
- In this demonstration, this IPv6 allows no EIGW because the IPv6 is public.
- Both outbound, and inbound internet traffic (if Security group allows), are allowed on EC2 instance.
- No NAT gateway is required since EC2 instance has an IPv6 address.
- As a result, EC2 instance is directly routable to internet, **without the instance's address translation** to public IP.
- The NACL and Security group protect EC2 instance from inbound internet traffic.
- EC2 instance would access the Amazon S3 privately without using: (i) internet; (ii) NAT gateway; (iii) going through the IGW⁶.

² This VPC is a dual-stack VPC.

³ In this demonstration, no NAT Gateway was required, hence the enabling of IPv6 in the EC2 instance.

⁴ Auto-assign public IPv4 address, and IPv6 address was enabled.

⁵ No egress-only Internet gateway is a regular Internet gateway.

⁶ By the S3 Gateway VPC Endpoint, we keep traffic inside AWS's private network. We require secure internal access to Amazon S3. This ultimately saves cost.

- EC2 instance should turn its domain name into its IP address. DNS resolution is needed for S3 access⁷.
- The following Security group rules were created (except SSH which is default) and defined for:
 - SSH (already set at default) from anywhere to custom at source 0.0.0.0/0⁸
 - HTTP and HTTPS, each at source 0.0.0.0/0

Practical:

Action 1: Setting up the VPC and its components

- I logged in to the AWS console as IAM Admin user.
- The Availability Region, Europe (London), eu-west-2, was selected.
- On the VPC settings, **VPC and more** button was selected after negotiating to the VPC dashboard. This is done to automatically create NACL, Route Table, and Internet Gateway.
- This name is named: “*cba-akosa*”
- An IPv4 CIDR block, 10.0.0.0/16, was assigned as the size of the VPC.
- Amazon-provided IPv6 CIDR block was selected, from the console.
- Availability Zone is set to 1.
- Both Private Subnet and Public Subnet are each set to 1.
- CIDR block sizes of 10.0.0.0/20 and 10.0.128.0/20 were assigned to the public subnet and private Subnet, respectively. Please see Table 1.
- The S3 gateway VPC Endpoint was selected. See Figure 1 which presents our selections on the console.

Figure 1: Annotated panel showing a preview of the network that was created

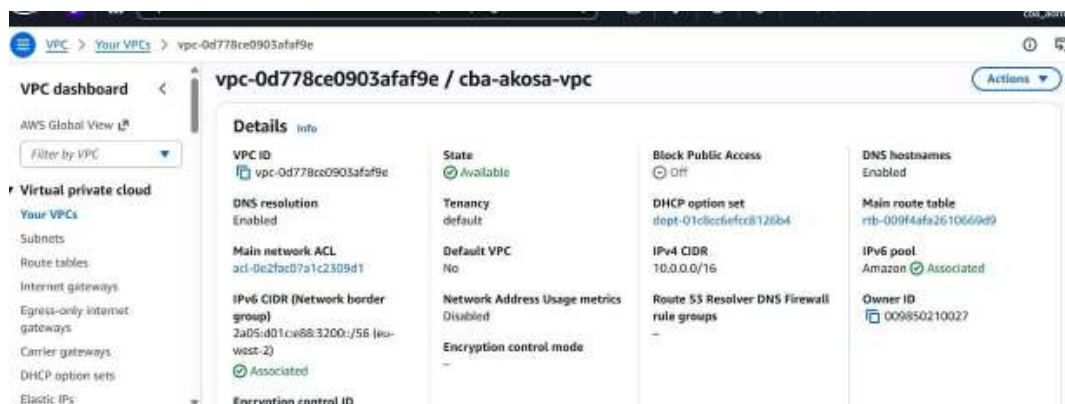
⁷ If both DNS Resolution and Hostnames are enabled, then public the EC2 instance gets a public DNS name. The instance can resolve S3, RDS, API endpoints.

⁸ The change from anywhere to custom at 0.0.0.0/0 (which technically effect the same thing) is not common professional practice. This is because 0.0.0.0/0 will have the SSH (or Port 22) exposed to the entire internet, thereby increasing brute-force risk. This is only done for temporary testing, non-production environment, and learning labs (as is our case).



- DNS Resolution is enabled so that our VPC can resolve domain names to IP addresses. Also, DNS Hostnames was enabled to assign public DNS name to our EC2 instance.
- The **Create VPC** button was clicked after previewing the network that I intended to create. Figure 2 shows the VPC that has been created.

Figure 2: Panel showing details of created VPC



From Figure 3, we can see that 2 subnets (1 private, and 1 public) have been created. Figure 4 also confirms the creation of 2 Route tables. From Figure 5, we can see that we have 1 IGW, newly created. Figure 6 reveals that 1 NACL has been created.

Figure 3: Panel showing our 2 Subnets



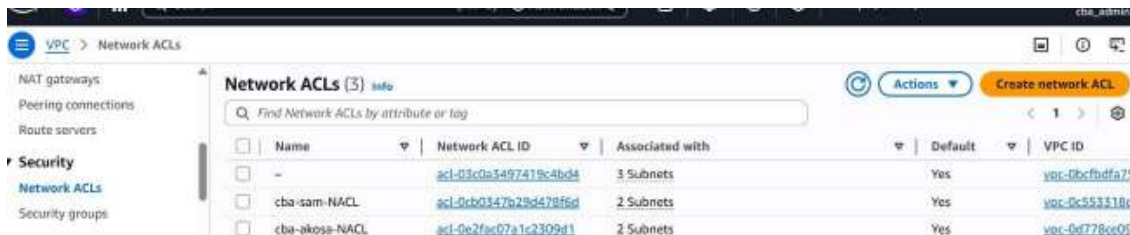
Figure 4: Panel showing our 2 Route tables



Figure 5: Panel showing 1 Internet Gateway created



Figure 6: Panel showing 1 NACL (named: “cba-akosa-NACL”) created



The Auto-assign for IPv6 and public IPv4 addresses was enabled. Please see Figure 7.

Figure 7: Panel showed enabled auto-assign setting for IPv6 and public IPv4

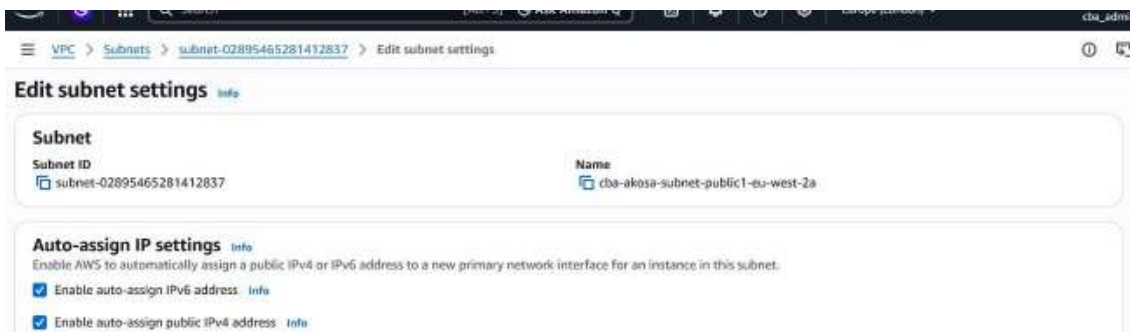


Figure 8: Panel showing successfully changed subnet settings



- The Subnet settings was saved.
- I confirmed that each of the 2 Subnets (1 Private, and 1 Public) has been explicitly associated with a Route Table. Please see Figure 9.

Figure 9: Panel showing each Subnet associated with a Route Table

Route tables	☐	cba-sam-rtb-public	rtb-0c8c27c455b0bf2fe	subnet-036bc011c90470...	-	No
Internet gateways	☐	cba-sam-rtb-private1-eu-west-2a	rtb-0e3b5ce8c33335886	subnet-04bae4ee40959b...	-	No
Egress-only internet gateways	☐	cba-akosa-rtb-public	rtb-0d8adc6680d453a71	subnet-02895465281412...	-	No
	☐	cba-akosa-rtb-private1-eu-west-2a	rtb-02be38a676c2c31d5	subnet-0d388688881234...	-	No

I confirmed that the Public Route Table (*cba-akosa-rtb-public*) has access to the Internet. Please see Figure 10.

Figure 10: Panel confirms that both IPv4 and IPv6 are connected to IGW

Q 0.0.0.0/0	X	Internet Gateway	Active	No	CreateRoute	Remove
Q :::/0	X	Internet Gateway	Active	No	CreateRoute	Remove

Action 2: Creating and Launching EC2 instance

- The **EC2 Instances** field was accessed from the EC2 dashboard on the console.
- The **Launch instance** button was clicked.
- The EC2 instance is named “*cba-akosa_instance*”
- Amazon Linux AMI was used as template for the operating system I needed to preload the EC2 instance.
- t3 micro (free tier) was used as EC2 instance type.
- A keypair was created to enable me to connect to the EC2 instance, remotely, using SSH for Linux.
- Under the **Network settings**, I changed the VPC option to *cba-akosa-vpc* to enable the EC2 instance inside the custom VPC and the Subnet, which I created. See Figure 11.

Figure 11: Panel showing the change to the newly created VPC: *cba-akosa-vpc*

▼ Network settings Info

VPC - required Info

vpc-0d778ce0903afaf9e (cba-akosa-vpc)	▼
10.0.0.0/16	2a05:d01c:e88:3200::/56

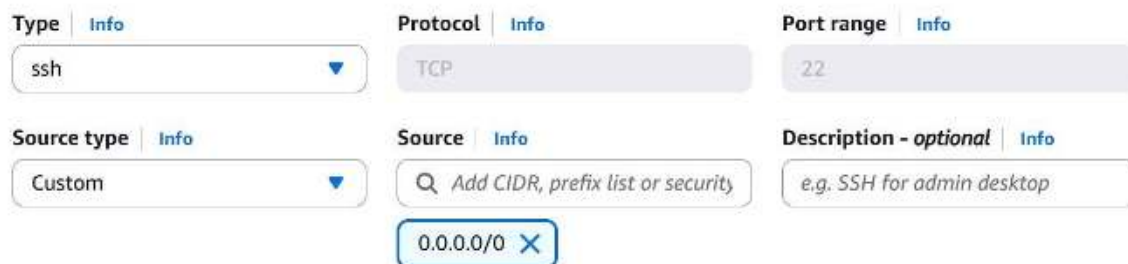
- We changed to Public Subnet so that the EC2 instance can be launched⁹. Please see Figure 12.

Figure 12: Panel shows the select Public Subnet on which the EC2 instance will be launched.



- The Auto-assign public IP was enabled.
- A Security group were created named “*cba_akosa_security group*”¹⁰
- I proceeded to create (except SSH at default) and define 3 Security group rules. See Figure 13:
 - For SSH (at default), I changed the **Source** type from anywhere to custom defined at 0.0.0.0/0
 - For HTTP and HTTPS, each changed to Source 0.0.0.0/0

Figure 13: Panel showing custom definition of source for SSH [Port range: 22], HTTP [Port range: 80], HTTPS [Port range: 443], and custom TCP [Port range: 8080]



⁹ To be able to remotely connect to our EC2 instance after its launch, it must have a Public IP. This means that the EC2 instance needs to be launched in a Public Subnet.

¹⁰ The *cba_akosa_security group* is given this description: “**allow internet access to automation server.**”

The image shows two identical security group rule configuration forms for HTTP and HTTPS, and a summary view of a security group rule for TCP on port 8080.

HTTP Rule Configuration:

- Type: HTTP
- Protocol: TCP
- Port range: 80
- Source type: Custom
- Source: 0.0.0.0/0
- Description - optional: e.g. SSH for admin desktop

HTTPS Rule Configuration:

- Type: HTTPS
- Protocol: TCP
- Port range: 443
- Source type: Custom
- Source: 0.0.0.0/0
- Description - optional: e.g. SSH for admin desktop

Security group rule 4 (TCP, 8080, 0.0.0.0/0):

- Type: Custom TCP
- Protocol: TCP
- Port range: 8080
- Source type: Custom
- Source: 0.0.0.0/0
- Description - optional: e.g. SSH for admin desktop

- After this, I launched the EC2 instance. Figure 14 confirms the successful launch.

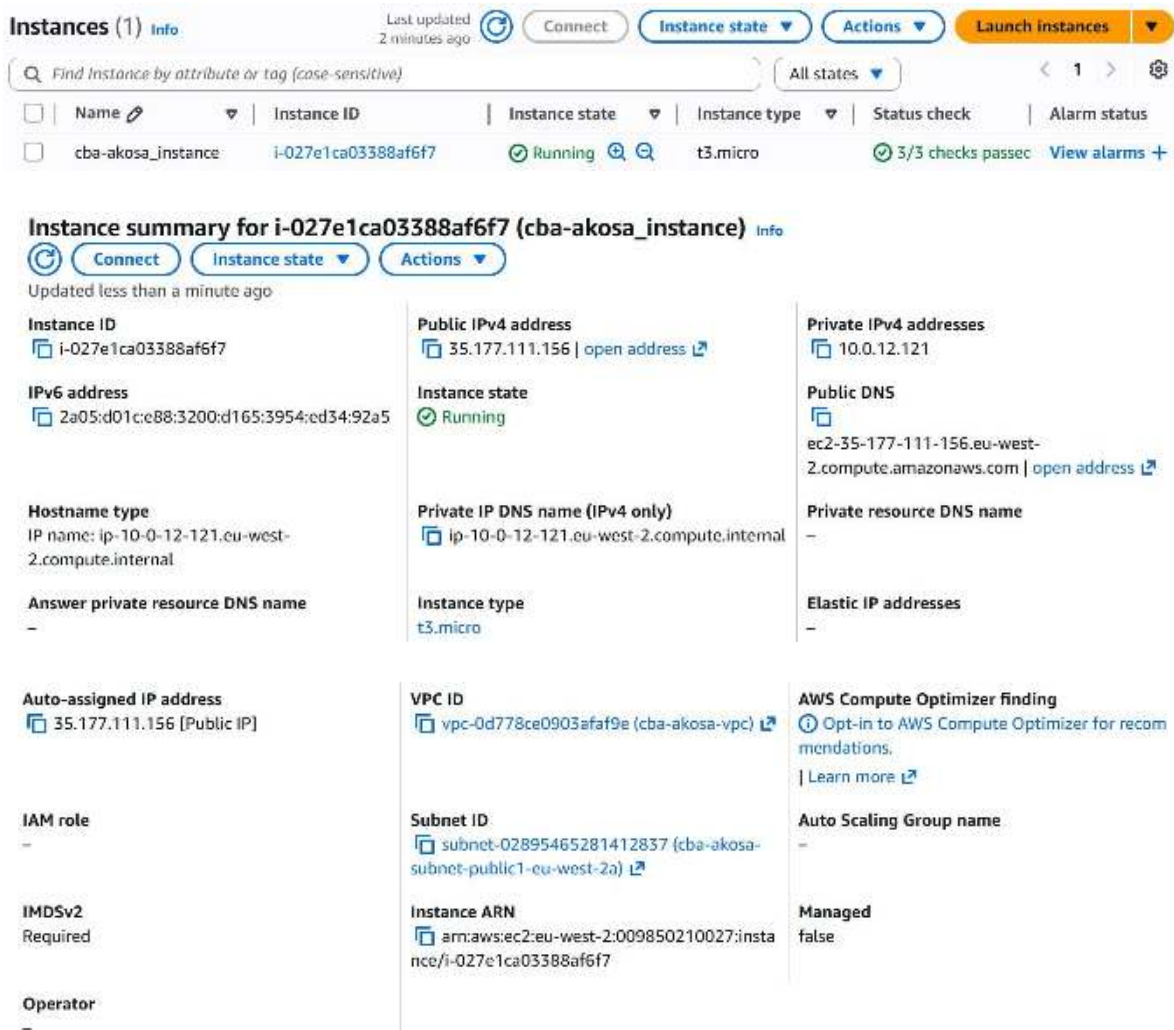
Figure 14: Panel showing that EC2 instance was successfully launched



- The EC2 instance in the Public Subnet was connected after ensuring the status check passed successfully¹¹. Figure 15 reveals that the EC2 instance is running.

¹¹ The connection was made using Public IP address (public IPv4 or IPv6 address).

Figure 15: Panel showing that the EC2 instance is running



Instances (1) Info Last updated 2 minutes ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states < 1 > ⚙

☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
☐	cba-akosa_instance	i-027e1ca03388af6f7	Running	t3.micro	3/3 checks passed	View alarms

Instance summary for i-027e1ca03388af6f7 (cba-akosa_instance)

 Info
Updated less than a minute ago
[Connect](#) [Instance state](#) [Actions](#)

- EC2 instance was connected using a Public IPv4 address. Please see Figure 16.

Figure 16: Panel showing EC2 instance connection initiated



EC2 Instance Connect SSM Session Manager SSH client EC2 serial console

Instance ID
i-027e1ca03388af6f7 (cba-akosa_instance)

Connection type

☒ Connect using a Public IP
Connect using a public IPv4 or IPv6 address

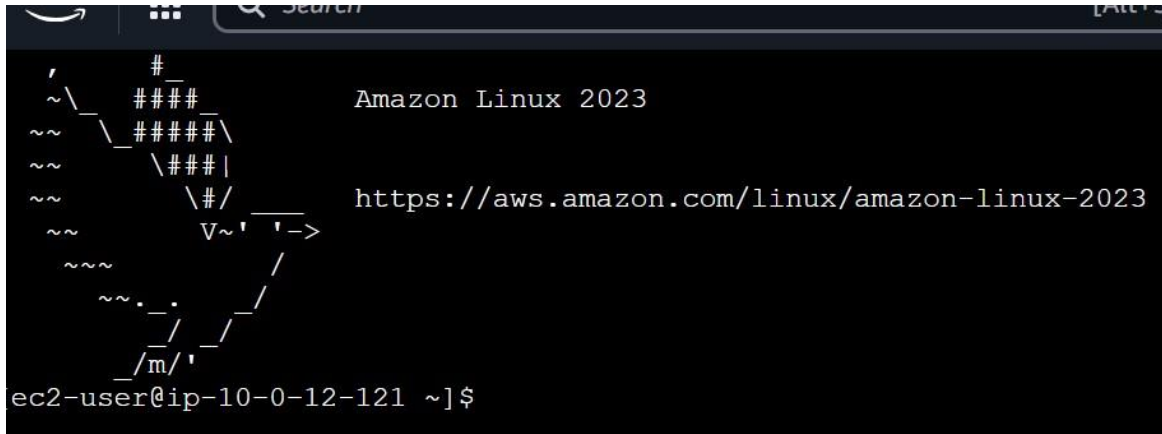
☐ Connect using a Private IP
Connect using a private IP address and a VPC endpoint

☒ Public IPv4 address
35.177.111.156

☐ IPv6 address
2a05:d01ce88:3200:d165:3954:ed34:92a5

- The SSH terminal opens in a new tab. Please see Figure 17.

Figure 17: Panel shows SSH terminal opened in a new tab



Action 3: Installing Jenkins automation server on the EC2 instance

Screenshots of each step are attached as appendix to this report (please see pages 16 to 18).

The link provided in the task advice is suited for Ubuntu AMI.

<https://github.com/cloudboostauk/lab-links/blob/main/EC2/userdata.sh>

The following is the EC2 User Data script implemented to install and run Jenkins tool on the provisioned EC2 instance. This includes Java, the Jenkins repo, starting the device, printing the admin password path. This User data script (holding commands and comments) is tailored to the AWS SSH terminal (Amazon Linux).

User Data Script tailored to AWS SSH Terminal (Amazon Linux)

```
#!/bin/bash

# 1) Identify OS (optional, but useful)
cat /etc/os-release

# 2) Update packages (dnf for AL2023, yum for AL2)
if command -v dnf >/dev/null 2>&1; then
  sudo dnf -y update
else
  sudo yum -y update
fi

# 3) Install Java 17
if command -v dnf >/dev/null 2>&1; then
  sudo dnf -y install java-17-amazon-corretto
else
  sudo yum -y install java-17-amazon-corretto
fi

# 4) Add Jenkins repo (Amazon Linux)
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

# 5) Import Jenkins key
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

# 6) Install Jenkins
if command -v dnf >/dev/null 2>&1; then
  sudo dnf -y install jenkins
else
  sudo yum -y install jenkins
fi

# 7) Enable + start Jenkins
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins --no-pager

# 8) Get initial admin password
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Action 4: Validating that Jenkins is working fine

- On the AWS console, I clicked on the EC2 instance-ID. Please see Figure 18.

Figure 18: Panel shows that EC2 instance ID from the AWS console



- Copied the Public IPv4 URL and added :8080 to the URL as in 35.177.111.156:8080
- Ran it on a new tab of my browser. The following webpage appeared. Please see Figure 19.

Figure 19: Panel showing the Jenkins webpage





Getting Started

Create First Admin User

Username

Password

Confirm password

Jenkins 2.541.2

[Skip and continue as admin](#) [Save and Continue](#)

- The following appears after putting the admin password retrieved from */var/lib/jenkins/secrets/initialAdminPassword*
The password is **65cd934c481a4dc2a89e68fdd96fc0cd**
- The EC2 instance was later terminated after ensuring that everything was fine. Please see Figure 20.

Figure 20: Panel confirming that EC2 instance has been terminated

⚠ On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.

Are you sure you want to terminate these instances?

Instance ID	Termination protection
☒ i-027e1ca03388af6f7 (cba-akosa_instance)	☒ Disabled

To confirm that you want to delete the instances, choose the terminate button below. Instances with termination protection enabled will not be terminated. Terminating the instance cannot be undone.

Skip OS shutdown
This option skips the graceful OS shutdown process. Use only when your instance must be stopped immediately, such as during an emergency or failover.

☐ Skip OS shutdown

[Cancel](#) [Terminate \(delete\)](#)

✔ Successfully initiated termination (deletion) of i-027e1ca03388af6f7

Instances (1/1) [Info](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

☒ Find instance by attribute or tag (case-sensitive) [All states](#) [1](#)

☒	Name	Instance ID	Instance state	Instance type	Status check	Alarm state
☒	cba-akosa_instance	i-027e1ca03388af6f7	Shutting-down	t3.micro	3/3 checks passed	View alarm

Summary:

- The public subnet is now connected to internet.
- The Public Subnet now has 1 EC2 instance provisioned on it.
- Three inbound security group rules allowing SSH, HTTP, HTTPS, and HTTP-Alt were configured for the EC2 instance.


```

else
    sudo yum -y update
fi
Last metadata expiration check: 0:55:30 ago on Mon Mar  2 15:33:06 2026.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-12-121 ec2-user]# if command -v dnf >/dev/null 2>&1; then
    sudo dnf -y install java-17-amazon-corretto
else
    sudo yum -y install java-17-amazon-corretto
fi
Last metadata expiration check: 0:57:57 ago on Mon Mar  2 15:33:06 2026.
Package java-17-amazon-corretto-1:17.0.18+9-1.amzn2023.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-12-121 ec2-user]#

```

```

Complete!
[root@ip-10-0-12-121 ec2-user]# sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2026-03-02 16:32:12-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 2a04:4e42:2d::645, 151.101.190.133
Connecting to pkg.jenkins.io (pkg.jenkins.io)[2a04:4e42:2d::645]:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://pkg.jenkins.io/rpm-stable/jenkins.repo [following]
--2026-03-02 16:32:12-- https://pkg.jenkins.io/rpm-stable/jenkins.repo
Reusing existing connection to [pkg.jenkins.io]:443.
HTTP request sent, awaiting response... 200 OK
Length: 267 [application/octet-stream]
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.repo  100%[=====>]  267  --KB/s  in 0s

2026-03-02 16:32:12 (13.4 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [267/267]

[root@ip-10-0-12-121 ec2-user]#

```

```

[root@ip-10-0-12-121 ec2-user]# sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@ip-10-0-12-121 ec2-user]#

```

```

Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.repo  100%[=====>]  267  --KB/s  in 0s

2026-03-02 16:32:12 (13.4 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [267/267]

[root@ip-10-0-12-121 ec2-user]# sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@ip-10-0-12-121 ec2-user]# if command -v dnf >/dev/null 2>&1; then
    sudo dnf -y install jenkins
else
    sudo yum -y install jenkins
fi
Last metadata expiration check: 1:00:52 ago on Mon Mar  2 15:33:06 2026.
Package jenkins-2.541.2-1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-12-121 ec2-user]#

```

```

[root@ip-10-0-12-121 ec2-user]# sudo systemctl enable jenkins
[root@ip-10-0-12-121 ec2-user]#

```

```

[root@ip-10-0-12-121 ec2-user]# sudo systemctl start jenkins
[root@ip-10-0-12-121 ec2-user]#

```

```

Tasks: 38 (limit: 1009)
Memory: 353.6M
CPU: 3min 91ms
CGroup: /system.slice/jenkins.service
└─28641 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=80...
Mar 02 15:09:36 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: [LF]>
Mar 02 15:09:36 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: [LF]> *****
Mar 02 15:09:36 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: [LF]> *****
Mar 02 15:09:36 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: [LF]> *****
Mar 02 15:09:42 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: 2026-03-02 15:09:42.024+0000 [id=30] INFO jen_ation
Mar 02 15:09:42 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: 2026-03-02 15:09:42.046+0000 [id=23] INFO hud_anning
Mar 02 15:09:42 ip-10-0-12-121.eu-west-2.compute.internal systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Mar 02 15:09:42 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: 2026-03-02 15:09:42.656+0000 [id=49] INFO h.m.taller
Mar 02 15:09:42 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: 2026-03-02 15:09:42.657+0000 [id=49] INFO hud_mpt #1
Mar 02 15:09:47 ip-10-0-12-121.eu-west-2.compute.internal jenkins[28641]: 2026-03-02 15:09:47.115+0000 [id=67] WARNING ... space
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-10-0-12-121 ec2-user]#

[root@ip-10-0-12-121 ec2-user]# sudo cat /var/lib/jenkins/secrets/initialAdminPassword
65cd934c481a4dc2a89e68fdd96fc0cd
[root@ip-10-0-12-121 ec2-user]#

```