

## Deployment 5 Documentation

### Before Deployment

Create a VPC called kura-vpc with the IPV4 CIDR of 192.168.0.0/16.

Create 4 subnets. 2 private and 2 public with the following ranges.

192.168.0.0 - 192.168.63.255

192.168.64.0 - 192.168.127.255

192.168.128.0 - 192.168.191.255

192.168.192.0 - 192.168.255.255

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR
<input type="checkbox"/>	public01	subnet-033073c2283225dd8	Available	vpc-0923116d7924ad7d0   ku...	192.168.0.0/18
<input type="checkbox"/>	private01	subnet-0119f0ebc4a0b7c31	Available	vpc-0923116d7924ad7d0   ku...	192.168.128.0/18
<input type="checkbox"/>	public02	subnet-0660ccbd79e22236e	Available	vpc-0923116d7924ad7d0   ku...	192.168.64.0/18
<input type="checkbox"/>	private02	subnet-041003a6e672f1133	Available	vpc-0923116d7924ad7d0   ku...	192.168.192.0/18

Create an internet gateway to allow our devices to communicate outside of the network. Once created, attach it to the VPC

VPC > Internet gateways > Attach to VPC (igw-04946cd1f6561bf70)

### Attach to VPC (igw-04946cd1f6561bf70) [Info](#)

**VPC**

Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs

Attach the internet gateway to this VPC.

▶ AWS Command Line Interface command

Cancel Attach internet gateway

Once attached, create a private and public route table

## Create route table [Info](#)

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

### Route table settings

#### Name - *optional*

Create a tag with a key of 'Name' and a value that you specify.

#### VPC

The VPC to use for this route table.

### Tags

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.

#### Key



#### Value - *optional*



You can add 49 more tags.

Private route tables route traffic to vpc but not outside the internet.

Attach the Internet gateway to the publicRT routes

### Edit routes

#### Destination

#### Target



Go into the subnet association of publicRT and attach the two public ips to it.

<b>Explicit subnet associations (0)</b>	
<input type="text" value="Find subnet association"/>	
Subnet ID	IPv4 CIDR
<b>Subnets without explicit associations (2)</b>	
The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:	
<input type="text" value="Find subnet association"/>	
Subnet ID	IPv4 CIDR
subnet-033073c2283225dd8 / public01	192.168.0.0/18
subnet-0660ccbd79e22236e / public02	192.168.64.0/18

Go into the subnet association of privateRTand attach the two private ips to it.

<b>Explicit subnet associations (2)</b>	
<input type="text" value="Find subnet association"/>	
Subnet ID	IPv4 CIDR
subnet-0119f0ebc4a0b7c31 / private01	192.168.128.0/18
subnet-041003a6e672f1133 / private02	192.168.192.0/18
<b>Subnets without explicit associations (2)</b>	
The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:	
<input type="text" value="Find subnet association"/>	
Subnet ID	IPv4 CIDR
subnet-033073c2283225dd8 / public01	192.168.0.0/18
subnet-0660ccbd79e22236e / public02	192.168.64.0/18

Inside of the VPC subnet, select subnets. And actions enable auto sign ip address

Actions ▲

View details

Create flow log

Modify auto-assign IP settings

## Create a public EC2 JumpHost

Network ⓘ	vpc-0923116d7924ad7d0   kura-vpc	<a href="#">Create new VPC</a>
Subnet ⓘ	subnet-033073c2283225dd8   public01   us-east-1a	<a href="#">Create new subnet</a>
	16378 IP Addresses available	
Auto-assign Public IP ⓘ	Use subnet setting (Enable)	

Assign a security group: ☒ Create a **new** security group  
☐ Select an **existing** security group

Security group name:

Description:

Type ⓘ

SSH ▼

Protocol ⓘ

TCP

Port Range ⓘ

22

**Source** ⓘ

Anywhere ▾


0.0.0.0/0, ::/0

Launch and select the keypair that you have downloaded

Create a private01 EC2 instance

**Network** ⓘ

vpc-0923116d7924ad7d0 | kura-vpc ▾

 [Create new VPC](#)

**Subnet** ⓘ

subnet-0119f0ebc4a0b7c31 | private01 | us-east-1a ▾

[Create new subnet](#)

16379 IP Addresses available

**Auto-assign Public IP** ⓘ

Use subnet setting (Disable) ▾

**Assign a security group:** ☒ Create a **new** security group

☐ Select an **existing** security group

**Security group name:**

only\_jumphost

**Description:**

Only\_jumphost should have access only

**Type** ⓘ

SSH ▾

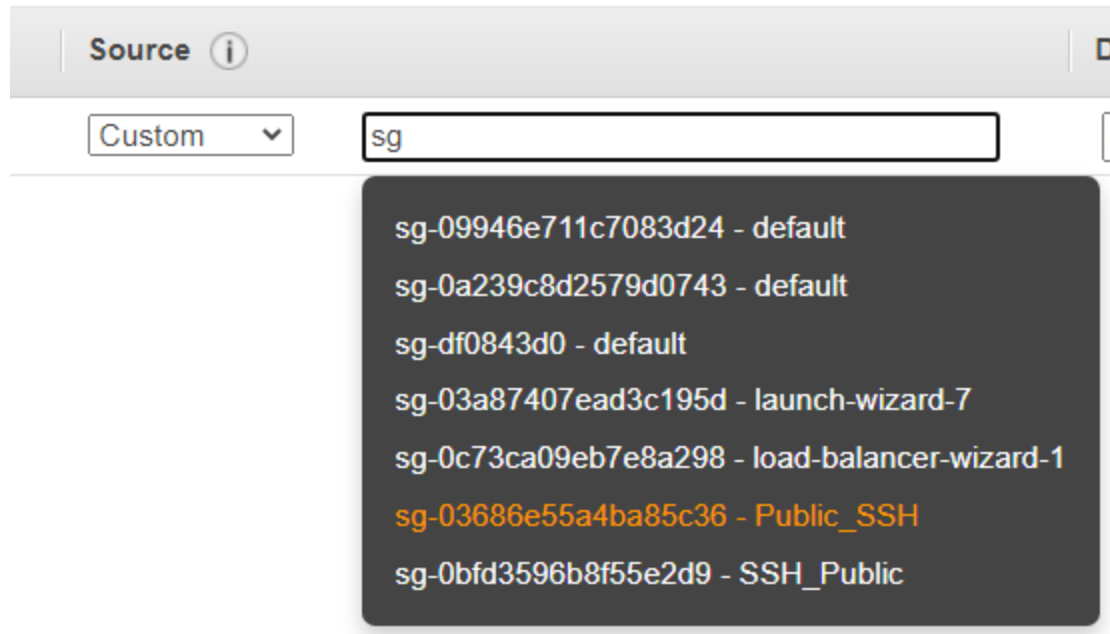
**Protocol** ⓘ

TCP

**Port Range** ⓘ

22

Select the security group you created for JumpHost (public ec2) which is Public\_SSH



Launch and select the keypair that you have downloaded

Connect to the Public SSH named JumpHost  
`ssh -i .key.pem ec2-user@Public IPv4 address`

Run `ping 8.8.8.8`  
You should get some requests.

`nano linux.pem`

Now get the information from your pem keypair that you had downaloded and paste it in  
linux.pem on the ec2 instance

Save it. Control + O to save and Control + X to exit

`chmod 400 linux.pem`

The Private IPv4 addresses is from the private01 ec2 instance.

`ssh -i linux.pem ec2-user@Private IPv4 addresses`

`ping 8.8.8.8` and you should have 100% packet loss

```
--- 8.8.8.8 ping statistics ---  
10 packets transmitted, 0 received, 100% packet loss, time 9209ms
```

NAT gateway, allows us to restrict inbound but allow us to do outbound.  
Go into aws VPC  
Create a NAT Gateway on the left side

### NAT gateway settings

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

**Subnet**  
Select a subnet in which to create the NAT gateway.

subnet-033073c2283225dd8 (public01) ▼

**Connectivity type**  
Select a connectivity type for the NAT gateway.

☒ Public  
☐ Private

**Elastic IP allocation ID** [Info](#)  
Assign an Elastic IP address to the NAT gateway.

eipalloc-0674755f1ad7583f2 ▼

Allocate Elastic IP

Go to Routing table

Select the private-RT  
Edit the Routes

Add Routes

Q 0.0.0.0/0 X

Q nat-0e4b97ad35b94b71e

Now go back into terminal and inside the private ec2 that you SSH into, run ping 8.8.8.8  
You should see some responses.

```
[ec2-user@ip-192-168-189-227 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=1.84 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=0.997 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=51 time=1.04 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=51 time=1.02 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=51 time=1.03 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=51 time=0.973 ms
```

## DEPLOYMENT 5

### Step 1

```
sudo amazon-linux-extras install java-openjdk11

sudo amazon-linux-extras install epel

sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum upgrade

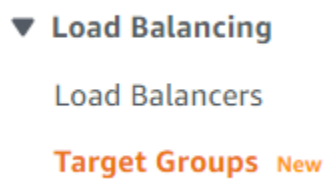
sudo yum install epel-release java-11-openjdk-devel

sudo yum install jenkins

sudo systemctl start jenkins
```

### Step 2

Select Target Group inside of AWS EC2



Then create a target group.

**Create target group**

Then select instances



Choose a target type

- ☒ Instances
  - Supports load balancing to instances within a specific VPC.

Create a target group name

Target group name

Jenkins

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Select protocol HTTP and put port 8080

Protocol

Port

HTTP



:

8080

Select your VPC

VPC

Select the VPC with the instances that you want to include in the target group.

kura-vpc

vpc-0923116d7924ad7d0

IPv4: 192.168.0.0/16



Select HTTP1

Protocol version

- ☒ HTTP1
  - Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
- ☐ HTTP2
  - Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
- ☐ gRPC
  - Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Select HTTP and enter /login

## Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

### Health check protocol

HTTP ▼

### Health check path

Use the default path of "/" to ping the root, or specify a custom path if preferred.

/login

Up to 1024 characters allowed.

Select Advanced Health check settings and select override and enter 8080

### ▼ Advanced health check settings

#### Port

The port the load balancer uses when performing health checks. The default is the traffic port of the load balancer, but you can specify a different port.

☐ Traffic port

☒ Override

8080

1-65535



Select next page

Click on your instance

# Register targets

## Available instances (1/2)

 *Filter resources by property or value*

	Instance ID		Name
<input type="checkbox"/>	i-02445fb34d9528bdc		JumpHost
<input checked="" type="checkbox"/>	i-04b513102c81868bf		Private01

Click include as pending below

**1 selected**

### Ports for the selected instances

Ports for routing traffic to the selected instances (separate multiple ports with commas):

8080


**Include as pending below**

Select create group

**Create target group**

Now create your ALB:

Select Load balancers

 **Load Balancing**

**Load Balancers**

Select create Load Balancer

Create Load Balancer

Select ALB

**Application Load Balancer**

HTTP  
HTTPS

Create

Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

[Learn more >](#)

Name the load balancer

Load balancer name

Name must be unique withi

ALB

A maximum of 32 alphanun

Select Internet facing

### Scheme [Info](#)

Scheme cannot be changed after the load balancer is created.

☒ **Internet-facing**

An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#) 

☐ **Internal**

An internal load balancer routes requests from clients to targets using private IP addresses.

### Select IPv4

#### IP address type [Info](#)

Select the type of IP addresses that your subnets use.

☒ **IPv4**

Recommended for internal load balancers.

☐ **Dualstack**

Includes IPv4 and IPv6 addresses.

### Select your VPC

#### Network mapping [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

#### VPC [Info](#)

Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed. To confirm the VPC for your targets, view your [target groups](#) .

kura-vpc

vpc-0923116d7924ad7d0

IPv4: 192.168.0.0/16





### Select two AZ's and two public subnets

☒ **us-east-1a**

Subnet

subnet-033073c2283225dd8

public01 ▼

 The subnet for your internet-facing load balancer must have a route to an internet gateway. You can update the subnet's route table in the [VPC Console](#) .

**IPv4 settings**



Assigned by AWS

☒ **us-east-1b**

Subnet

subnet-0660ccbd79e22236e

public02 ▼

 The subnet for your internet-facing load balancer must have a route to an internet gateway. You can update the subnet's route table in the [VPC Console](#) .

Select the security group for the ALB

## Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer.

### Security groups

Select security groups

[Create new security group](#)

Allow\_all sg-06e5357bdb155c1db X  
VPC: vpc-0923116d7924ad7d0

### Select HTTP and your target group

#### Listeners and routing [Info](#)

A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specifications. You can specify multiple rules and multiple certificates per listener after the load balancer is created.

##### ▼ Listener HTTP:80

Protocol

HTTP ▼

Port

80

1-65535

Default action [Info](#)

Forward to

Jenkins

Target type: Instance

HTTP ▼

[Create target group](#)

### Finally select create load balancer

Create load balancer

Takes a few minutes to set up. You will see the status is active and the target group health is healthy in the target group section

Edit only\_jumpshot

Custom TCP port 8080

Source 0.0.0.0

Type Info

Custom TCP ▼

Protocol Info

TCP

Port range Info

8080

Source Info

Custom ▼

Q

0.0.0.0/0 X

Go into Load Balancer and copy the DNS name and paste it in url

DNS name

ALB-935408808.us-east-1.elb.amazonaws.com

If your target group says unhealthy then try these steps...

First Edit inbound rule of your ec2 instance with jenkins to allow port 8080 inbound from the security group of the public ec2.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-05690b9a7a8953b0e	SSH ▼	TCP	22	Custom ▼	<div>Q</div> <div>649474668035/sg-0f2465e31829468e3 X</div>	<div></div> <div>Delete</div>
sgr-0d6c50f2d560b07db	Custom TCP ▼	TCP	8080	Custom ▼	<div>Q</div> <div>649474668035/sg-07324f6574ffc45b1 X</div>	<div></div> <div>Delete</div>

Add rule

Allow the security group of your load balancer to accept port 80 inbound.



Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
sgr-0440ff1315aa0bd07	HTTP	TCP	80	Custom	<input type="text" value="Q"/> <input type="text" value="0.0.0.0/0"/> <input type="button" value="X"/>

-----

Create another EC2 inside the same private subnet of the Jenkins master (This will be the agent)

**Network** [i](#) vpc-0923116d7924ad7d0 | kura-vpc  [Create new VPC](#)

**Subnet** [i](#) subnet-0119f0ebc4a0b7c31 | private01 | us-east-1a  [Create new subnet](#)  
 16378 IP Addresses available

**Auto-assign Public IP** [i](#) Use subnet setting (Disable)

Tag  
Name - "Private01 - Child"

Create a new security group

Assign a security group: ☒ Create a **new** security group  
☐ Select an **existing** security group

**Security group name:**

**Description:**

Type <a href="#">i</a>	Protocol <a href="#">i</a>	Port Range <a href="#">i</a>	Source <a href="#">i</a>
SSH <input type="button" value="v"/>	TCP	22	Custom <input type="button" value="v"/> sg-043377a0f24abb656
Custom TCP F <input type="button" value="v"/>	TCP	8080	Custom <input type="button" value="v"/> 0.0.0.0/0, ::/0

For SSH Source. Put the security group that has jenkins on it. This is only\_jumphost

SSH into JumpHost (Public01),

Once inside, then SSH into private01.

Create a new key and put the RSA information into it

nano linux.pem

chmod 400 linux.pem

Once inside there, SSH once again into private01-child

ssh -i linux.pem ec2-user@Private IPv4 addresses

```
PS C:\Users\robin\.ssh> ssh -i .\rixardo.pem ec2-user@54.82.69.201
Last login: Sat Sep 11 20:07:07 2021 from cpe-24-193-149-222.nyc.res.rr.com

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-45-222 ~]$ ls
linux.pem
[ec2-user@ip-192-168-45-222 ~]$ ssh -i linux.pem ec2-user@192.168.189.227
Last login: Sat Sep 11 20:07:24 2021 from 192.168.45.222

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-189-227 ~]$ ls
linux.pem
[ec2-user@ip-192-168-189-227 ~]$ ssh -i linux.pem ec2-user@192.168.185.83
The authenticity of host '192.168.185.83 (192.168.185.83)' can't be established.
ECDSA key fingerprint is SHA256:FRLjRzRdmKbTibyPX/fO/eriCzKU80nSoQ4jSPOxMME.
ECDSA key fingerprint is MD5:58:86:69:35:5b:3d:d7:f5:42:13:7a:f3:ff:2e:5c:3c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.185.83' (ECDSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-185-83 ~]$
```

Once on Jenkin's page

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

## Install suggested plugins

Install plugins the Jenkins community finds most useful.

Once that is done, put in information

-----

Configure the Jenkins master to SSH into the agent

Once logged into Jenkins, go to Manage Jenkins



**Manage Jenkins**

Select manage nodes



### **Manage Nodes and Clouds**

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Select new node in the left



**New Node**

Create a name for the node and select permanent agent.

Node name

Test

☒ **Permanent Agent**

Adds a plain, permanent agent to .  
Select this type if no other agent t

OK

Create a name and description

Name

Test

Description

test

Enter 2 for executors


Number of executors

2

Enter {/home/ec2-user/jenkins} for remote root directory

Remote root directory

{/home/ec2-user/jenkins}

 **Are you sure you want to use  
current working directory. Us**

Create a label - agent-linux

Labels

agent-linux

Select use this node as much as possible

Usage

Use this node as much as possible

Select launch agent via SSH

Launch method

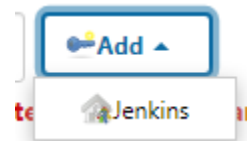
Launch agents via SSH

Enter the private IP address of the agent for Host - 192.168.185.83

Host

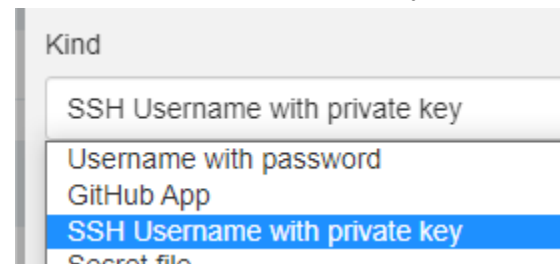
192.168.185.83

Add SSH credentials (username: ec2-user | key: the private key you used to ssh into agent)



For Kind Select

SSH Username with private key



Enter ID - **worker-ssh** and enter description - **ssh into agent**

ID

worker-ssh

Description

ssh into agent

Enter username **ec2-user**

Username

ec2-user

For the private key, enter your RSA key information directly into the box.

Private Key

☒ Enter directly

Key

```
JGSWQZKCgYEA14MPCBa1KECqGCqSVLYL1GY22n6p0V0FVZKSay0WzHGLyOKpyG8Y
x0IGlPuMntrwi5Re3oJCM1vtA1fHQPRF2K8nqWs2zpNGwzemHwNjFHMtJxmBIpaQ
4SAgMxNrvhLdyIbakrs2sg7XRMAYRK3F2xXoif5LewPnC8WhlvYt9Qs=
-----END RSA PRIVATE KEY-----
```

No passphrase for the key. Press Add

Select your credentials

Credentials

ec2-user (ssh into agent) ▼

ec2-user (ssh into agent)

Select non verifying verification strategy

Host Key Verification Strategy

Non verifying Verification Strategy

Save and then look at the logs to see if the setup was successful.

Save

Look at the logs to see if your setup was successful

Click on agent



Test



Log

Click on Logs on the left side.

## Errors - Java

```
Checking Java version in the PATH
bash: java: command not found
Java is not in the PATH nor configured with the javaPath setting, Jenkins will try to guess where is Java, this guess will be removed in the future. :Launch
agents via SSH
[09/11/21 20:34:22] [SSH] Checking java version of {/home/ec2-user/jenkins}/jdk/bin/java
Couldn't figure out the Java version of {/home/ec2-user/jenkins}/jdk/bin/java
bash: {/home/ec2-user/jenkins}/jdk/bin/java: No such file or directory

[09/11/21 20:34:22] [SSH] Checking java version of java
Couldn't figure out the Java version of java
bash: java: command not found

[09/11/21 20:34:22] [SSH] Checking java version of /usr/bin/java
Couldn't figure out the Java version of /usr/bin/java
bash: /usr/bin/java: No such file or directory

[09/11/21 20:34:22] [SSH] Checking java version of /usr/java/default/bin/java
Couldn't figure out the Java version of /usr/java/default/bin/java
bash: /usr/java/default/bin/java: No such file or directory

[09/11/21 20:34:22] [SSH] Checking java version of /usr/java/latest/bin/java
Couldn't figure out the Java version of /usr/java/latest/bin/java
bash: /usr/java/latest/bin/java: No such file or directory

[09/11/21 20:34:22] [SSH] Checking java version of /usr/local/bin/java
Couldn't figure out the Java version of /usr/local/bin/java
bash: /usr/local/bin/java: No such file or directory





[09/11/21 20:34:22] [SSH] Checking java version of /usr/local/java/bin/java
Couldn't figure out the Java version of /usr/local/java/bin/java
bash: /usr/local/java/bin/java: No such file or directory

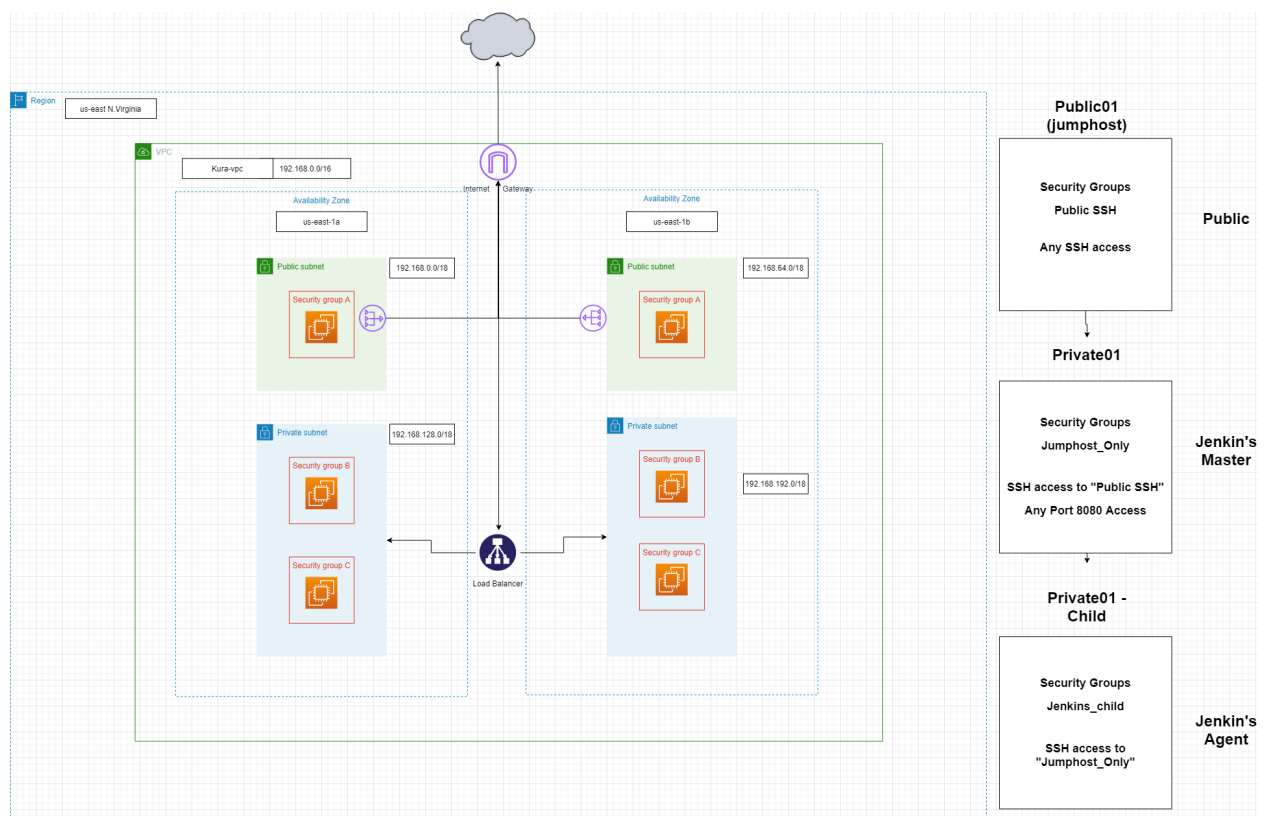
java.io.IOException: Java not found on hudson.slaves.SlaveComputer@df16141. Install Java 8 or Java 11 on the Agent.
    at hudson.plugins.sshslaves.JavaVersionChecker.resolveJava(JavaVersionChecker.java:84)
    at hudson.plugins.sshslaves.SSHLauncher$1.call(SSHLauncher.java:453)
    at hudson.plugins.sshslaves.SSHLauncher$1.call(SSHLauncher.java:421)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)
    at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
    at java.base/java.lang.Thread.run(Thread.java:829)
[09/11/21 20:34:22] Launch failed - cleaning up connection
[09/11/21 20:34:22] [SSH] Connection closed.
```

Fix: Install the following in the agent ec2 terminal

```
sudo yum install maven
sudo yum install git
```

Success:

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.90 GB	0 B	5.90 GB	0ms 
	Test	Linux (amd64)	In sync	5.92 GB	0 B	5.92 GB	81ms 
Data obtained		1 min 41 sec	1 min 41 sec	1 min 41 sec	1 min 41 sec	1 min 41 sec	1 min 41 sec



So we created a VPC (virtual private cloud) Then we created 4 different subnets with different IP ranges using subnetting. Once we had that we created an internet gateway which basically allows our vpc/devices to communicate outside of the network. In other words it's like a router. Then we created two routing tables for publicRT and privateRT. The routing table is for traffic inside the VPC. For the public subnets, we associated the public routing tables to them and vice versa.



Then we created a public EC2 called jumphost which is the public EC2. This ec2 instance has a public subnet and allows auto-assign public IP.

We then created a private ec2 instance that has the private01 subnet attached. For the security group we assigned the SSH access to the public01 security group. So that basically means that you can only SSH into private01 if you are inside of the public01 (jumphost) instance. Then we created a pem keypair and pasted the information into the file and chmod it. Once inside the private01 instance we pinged and got no response.

We then created a NAT gateway to allow us to restrict inbound but allow us to do outbound for updating. We attached the NAT gateway to the publicRT which basically gave us a response when we pinged

Then we created another EC2 private instance in the same private01 subnet and named it child. For the security group, the SSH's value is the only\_jumphost security group. So basically the only\_jumphost(private01) can ssh into the child ec2 instance.

User -> Internet/Cloud -> Internet Gateway -> EC2 with attached Nat Gateway -> private01 jenkins

**So for this topology there are two AZ zones. They both have a public subnet and a private subnet. Both of them can ssh into the public subnet and the private subnet has a master jenkins program on it. When someone accesses the application load balancer, traffic is either gone to us-east-1a or us-east-1b. This is in case one zone is overloaded and for redundancy. Once inside, the target group redirects traffic from the public subnet to the private subnet that has jenkins installed. It uses port forwarding to direct the traffic.**

**Delete the following in this order so you don't get charged...**

**Nat Gateway**

**Application Load Balancer**

**Stop the Ec2's**

**Release your Elastic IP**