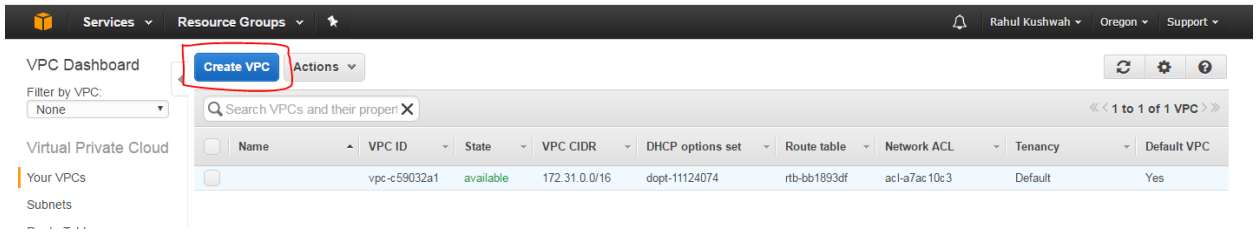
Assignment 3:

Consider a monolithic java application stack.
Apache Web Server, Apache Tomcat application server with Active MQ and Oracle and MongoDB backend.

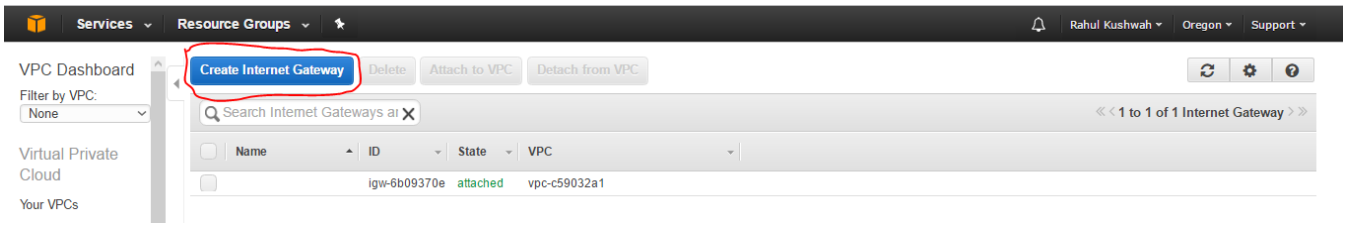
Propose a solution to migrate this application stack to AWS. Mention all the AWS services you would use and how you would maintain HA and Load Balancing (consider app to be stateless).

Solution 1: Using independent Service of AWS

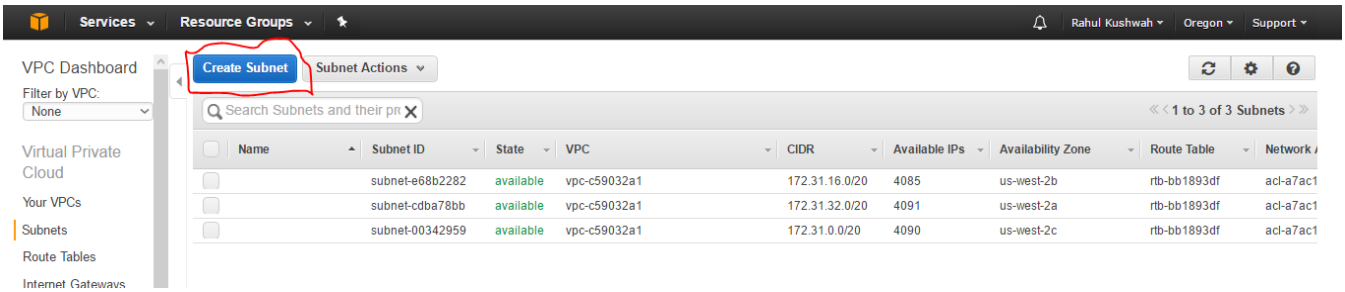
- First we will create our own private Cloud that is **VPC.**
- Using Networking Service of AWS create a VPC



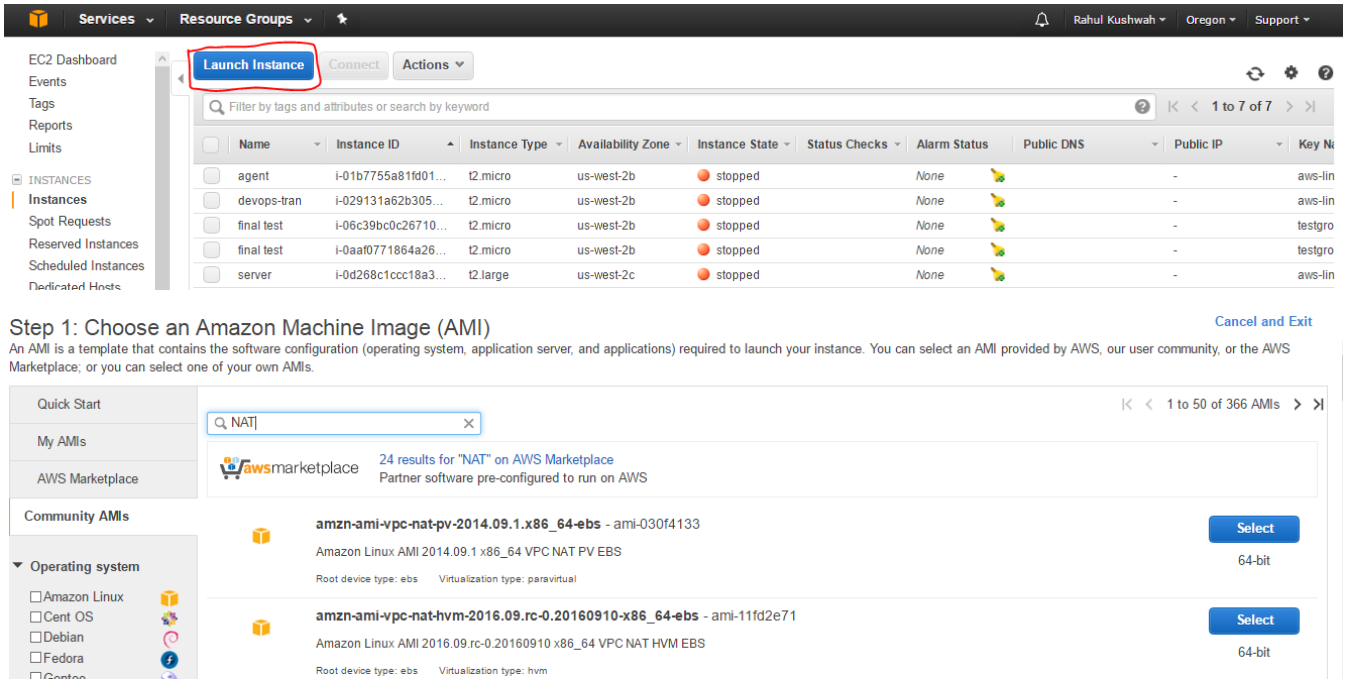
- Then give the VPC name and CIDR block range i.e. how many IP address you need in your VPC.
- Then create **Internet Gateway** and Attach to you VPC.



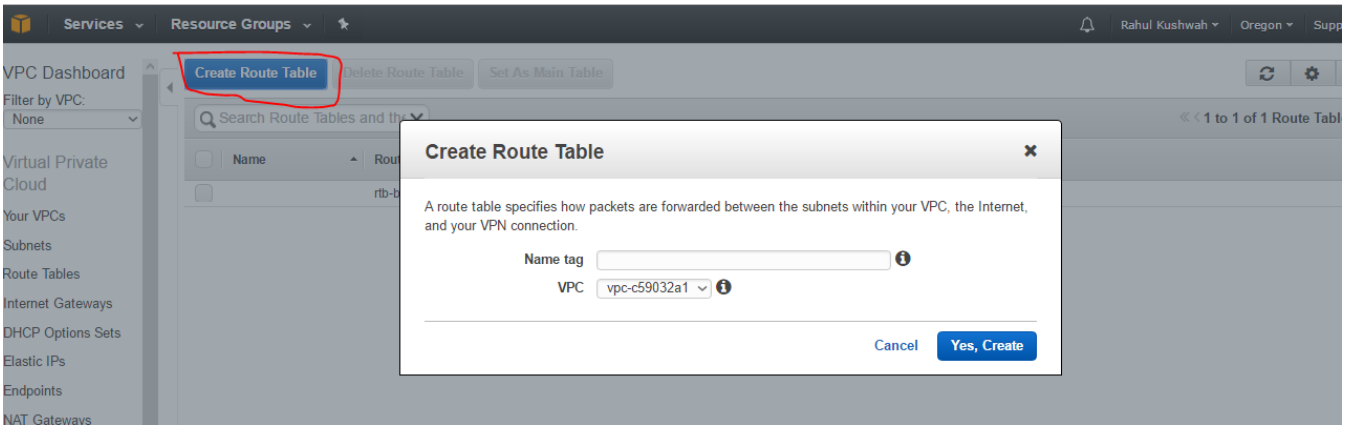
- Then **Create Subnets** in that VPC.



- In this case we can create 3 Subnets.
    - 1 Public and 2 private.
- Public subnet will have Web Servers and 1 private subnet will have Application servers and 1 private network will have DB instances.
- Now Create a **NAT** instance in side public subnet by using EC2 service.(This instance is created to provide internet connectivity for the instance without exposing there IP and connecting them to internet)
- Disable the source/destination check for that NAT instance.



- Now Create Route Table for public subnet to provide internet for public instances.
    a. Create new route table under VPC section
    b. In destination add 0.0.0.0/0
    c. In Target provide the gateway name which we created.
- Now Create Route Table for Private subnet to provide internet through NAT for private instance.
    d. Create new route table under VPC section
    e. In destination add 0.0.0.0/0
    f. In Target provide the NAT instance name which we created.



---------- By Above Steps we have created basic skeleton of our environment ----------

- Now using **EC2 service** launch number of web server in public subnet with desired AMI.
- In Security group you can open port which is required for e.g.
    a. Port 80 for HTTP
    b. Port 22 for SSH
    c. Port 443 for HTTPS

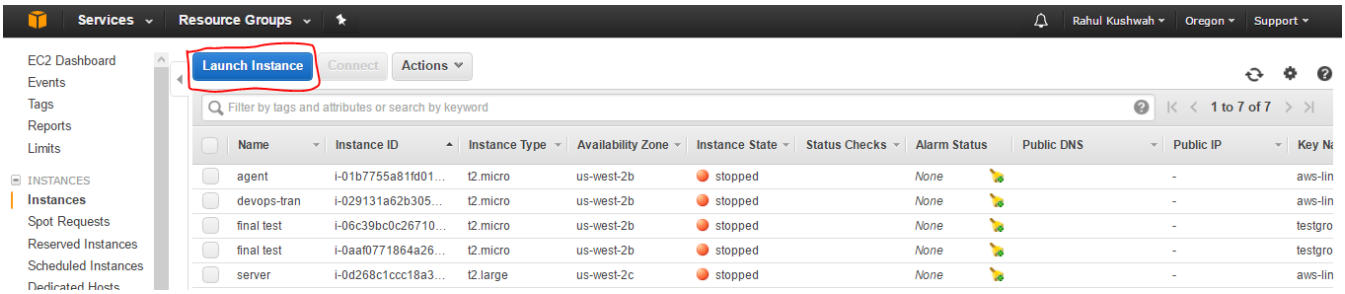
- During this process you can assign Role to those EC2 so that they can communicate to other service without storing password in it.
- You can give a boot script which will run as soon as instance is boot up for e.g.

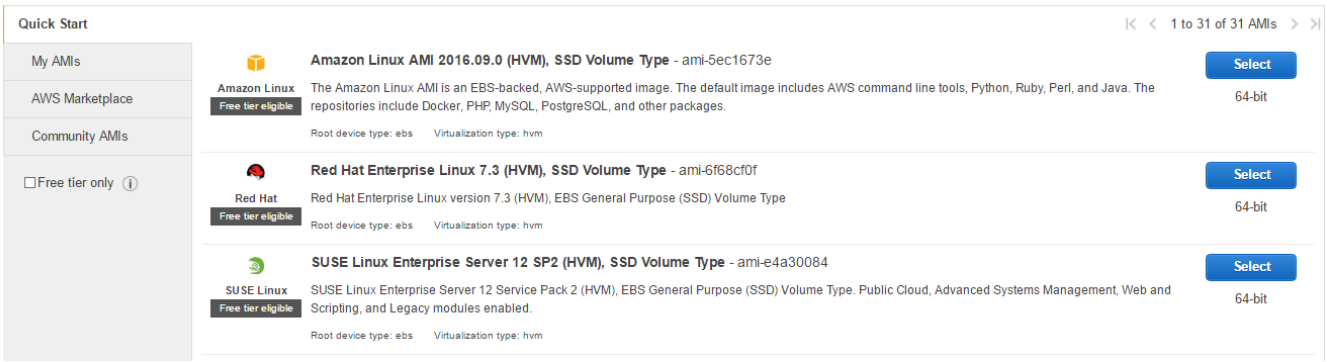
  #!/bin/bash

  Yum install httpd –y

  Service httpd start

- Create **Elastic IP** and attach them to web servers which are in public subnet.





---------- By this changes our public subnet is ready----------

- Now using **EC2 service** launch number of application server in Private subnet with desired AMI.
- In Security group you can open port which is required for e.g.
  d. Port 80 for HTTP
  e. Port 22 for SSH
- During this process you can assign Role to those EC2 so that they can talk to other service without storing password in it.
- You can give a boot script which will run as soon as instance is boot up for e.g.

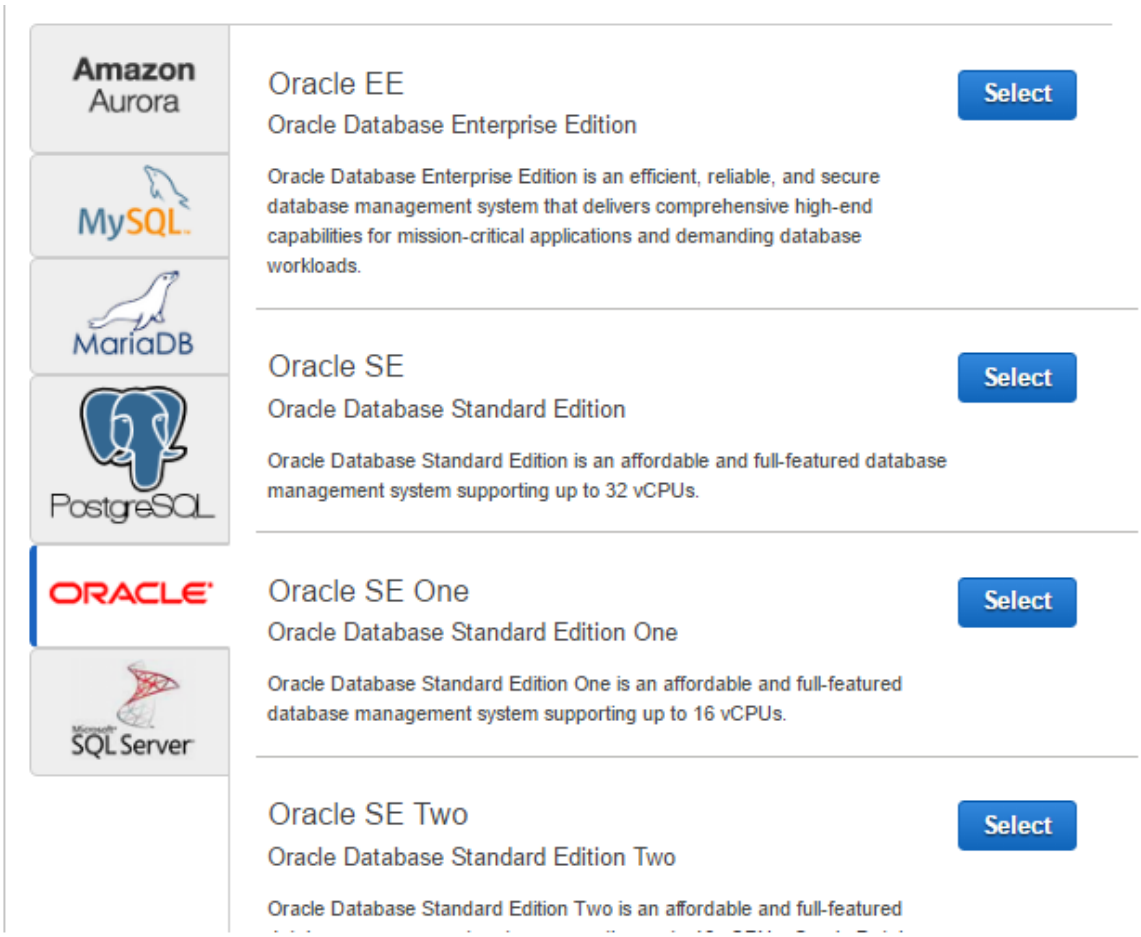
  #!/bin/bash

  Yum install tomcat –y

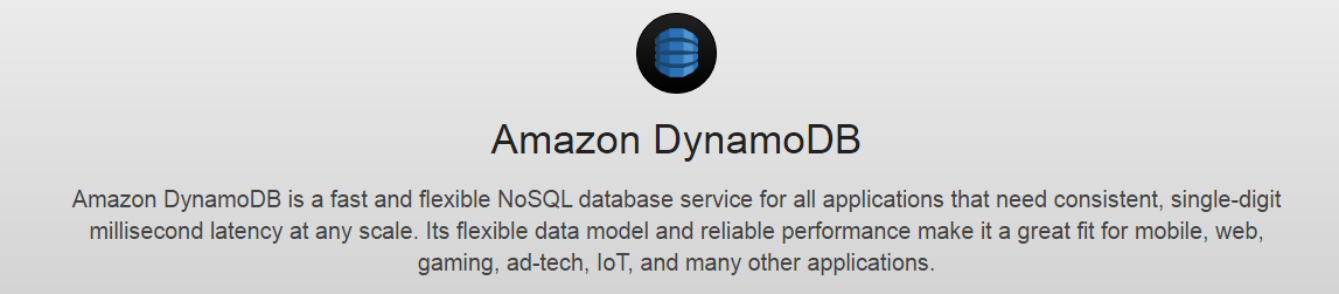
  Service tomcat start

----------By this changes our Application private subnet is ready----------

- Using **DATABASE** service we can create database in third private subnet
- For **ORACLE** we can use **RDS** service and can launch the oracle DB with any configuration



- Instead of **MongoDb** we can use **DynamoDB** service which also a **NoSQL** database which use **JSON**



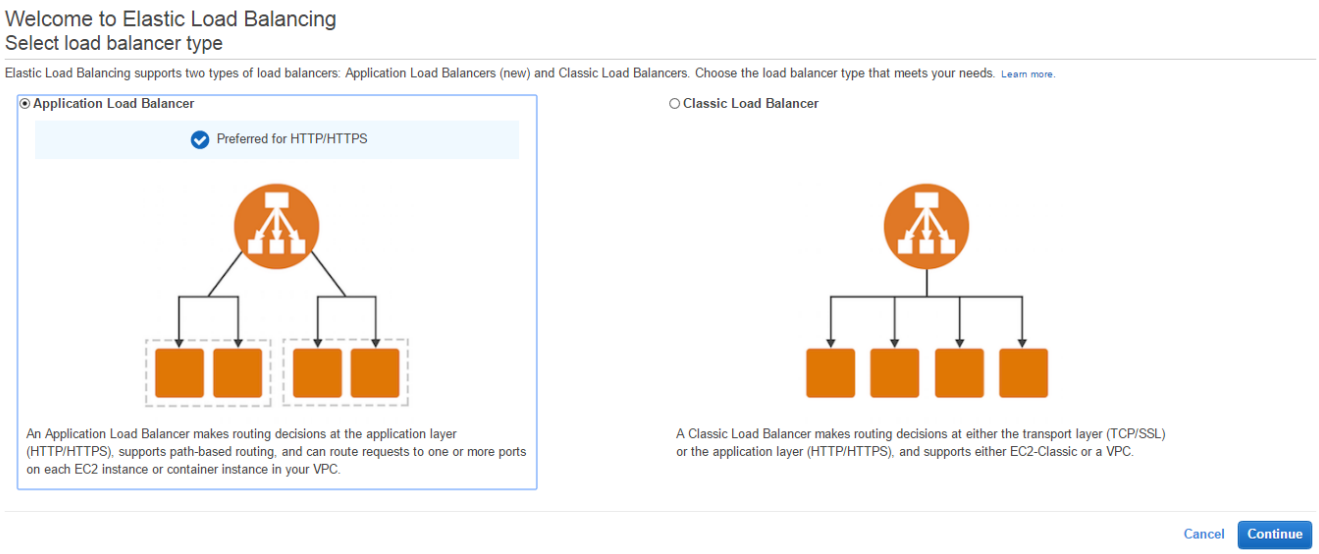
- For High Availability we can use Multi-AZ deployment so that if any failure occurs then another DB instance is there for backup.

----------By this changes our Data Base private subnet is ready----------

<u>FOR LOAD BALANCING</u>

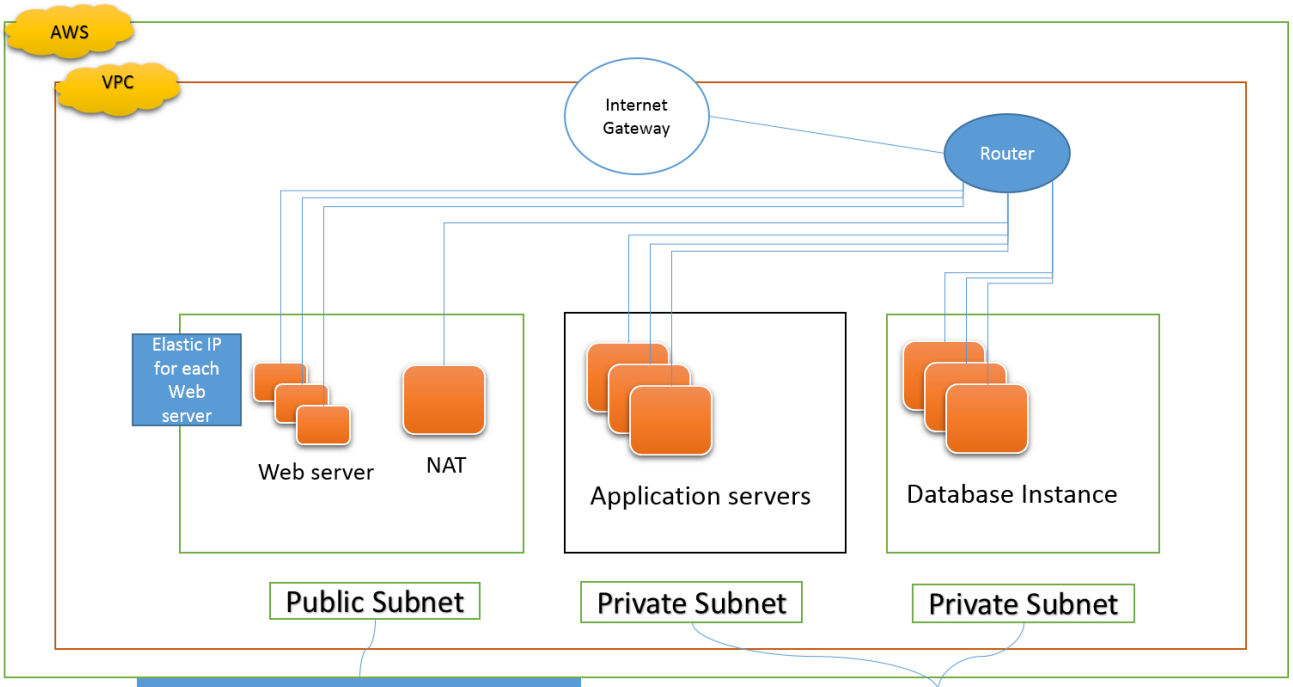
- By using **Elastic load balancer** service we can maintain the load across all web server instance.
- By load balance we can do heath checks of all servers and can create alarm also.
- In load balancer service we can create scaling policies for HIGH AVAILABILITY.

- This scaling policy can launch or terminate any instance in any subnet except Database instance.
- We can create internal load balancer for maintaining the load on app servers



- ---

<u>After all these Steps the AWS structure will look like this:</u>



Route Table for Public Subnet	
Destination	Target
Public Subnet IP	local
0.0.0.0/0	Internet Gateway

Route Table for Private Subnet	
Destination	Target
Private Subnet IP	local
0.0.0.0/0	NAT

Solution 2: Using Cloud Formation Service in AWS

1. Using JSON script you can create all the instance in one run
2. In JSON Script we have to configure all the services we need in our environment. Then it launch all the required thing and we will get a basic structure of our environment.
3. We can use Diagram method also to create basic skeleton of our VPC

Solution 3: Using Elastic Beanstalk Service in AWS

- By providing our java or any other language code it will create the infrastructure.