Create one RedHat Linux Management node

Login to the node install kops

kops helps you create, destroy, upgrade and maintain production-grade, highly available, Kubernetes clusters from the command line. AWS (Amazon Web Services) is currently officially supported, with GCE and VMware vSphere in alpha and other platforms planned.

Install Kops

wget -O kops https://github.com/kubernetes/kops/releases/download/1.7.0/kops-linux-amd64

chmod +x ./kops

sudo mv ./kops /usr/local/bin/

Install kubectl

wget -O kubectl https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

Install AWS CLI

sudo yum -y install wget

wget <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

sudo yum -y install epel-release-latest-7.noarch.rpm

sudo yum -y install python-pip

sudo pip install awscli

Configure Aws

aws configure

ACCESS KEY:

SECRET KEY:

REGION:

OUTTYPE:

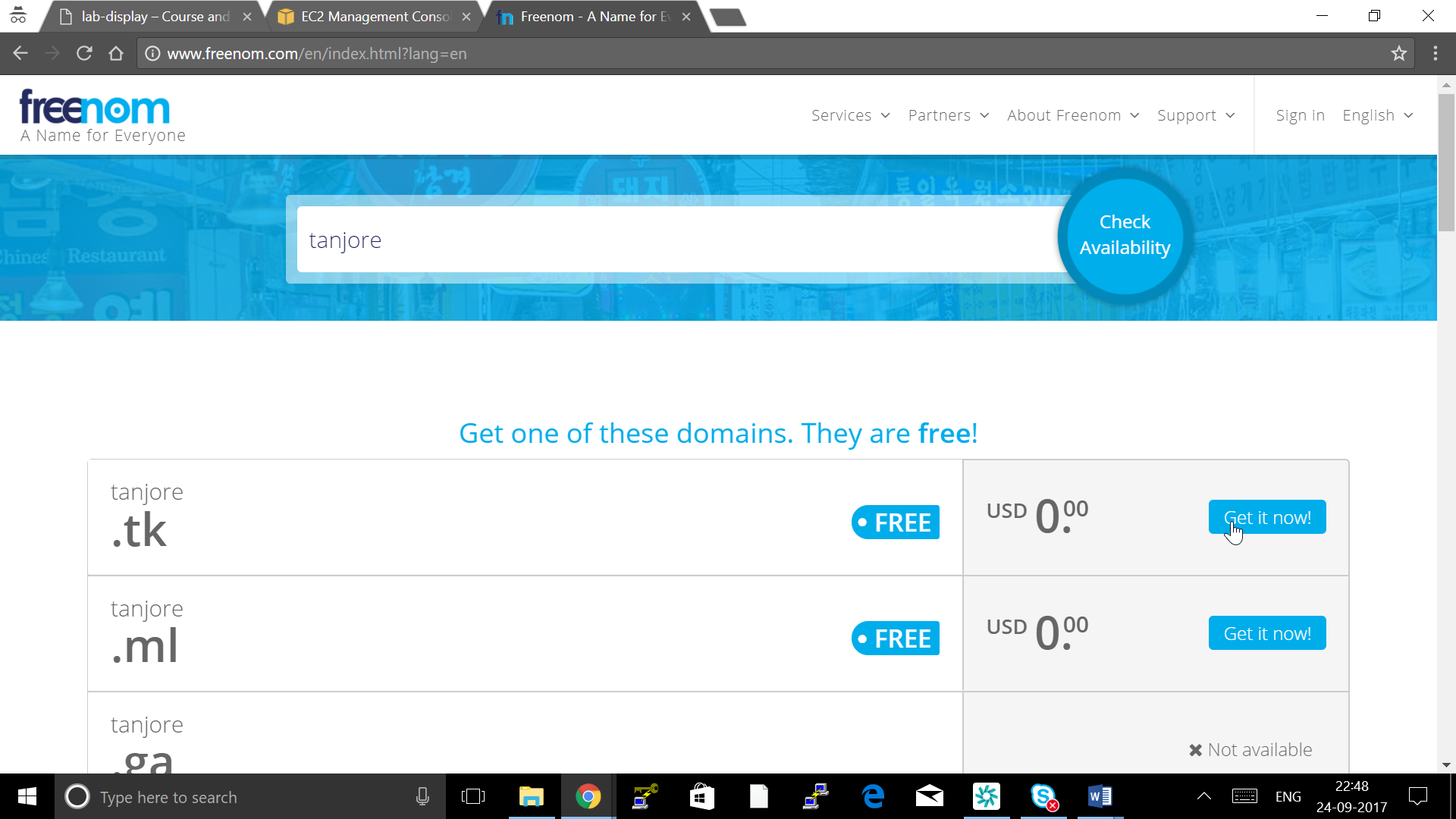
Create a domain name

Go to <http://www.freenom.com/en/index.html?lang=en>

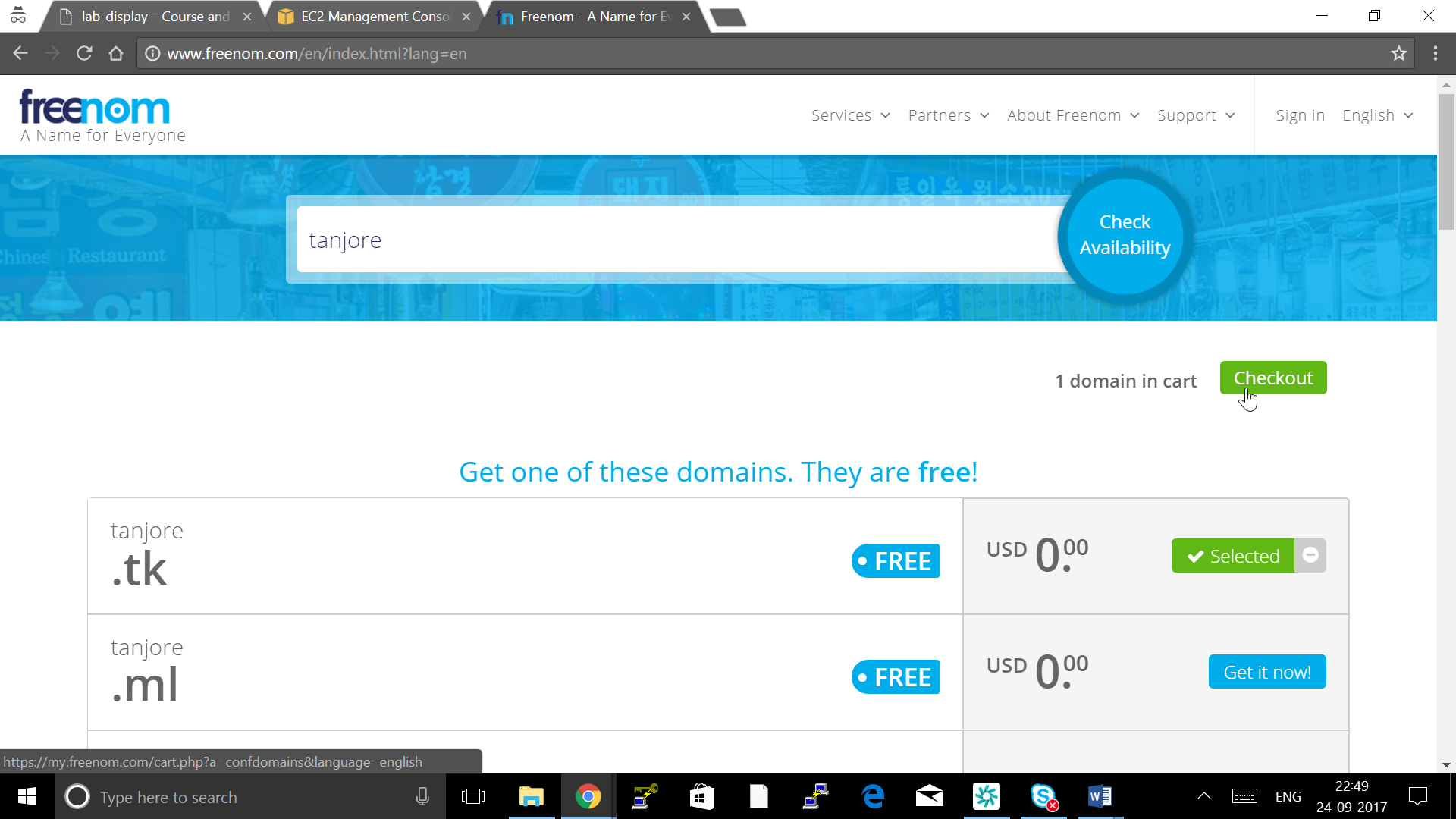
Type name of your domain and check for availability

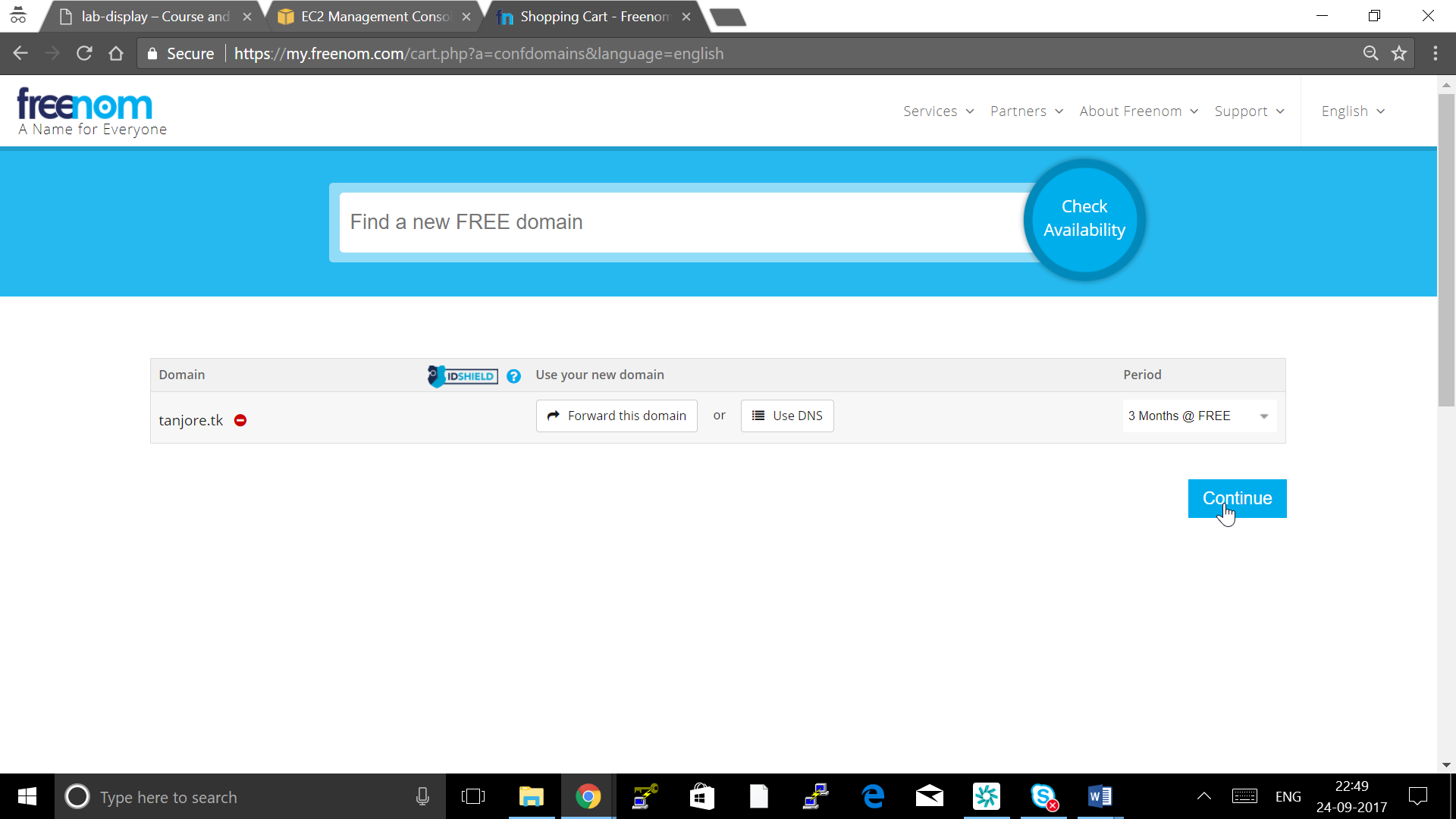


Click on Get it now

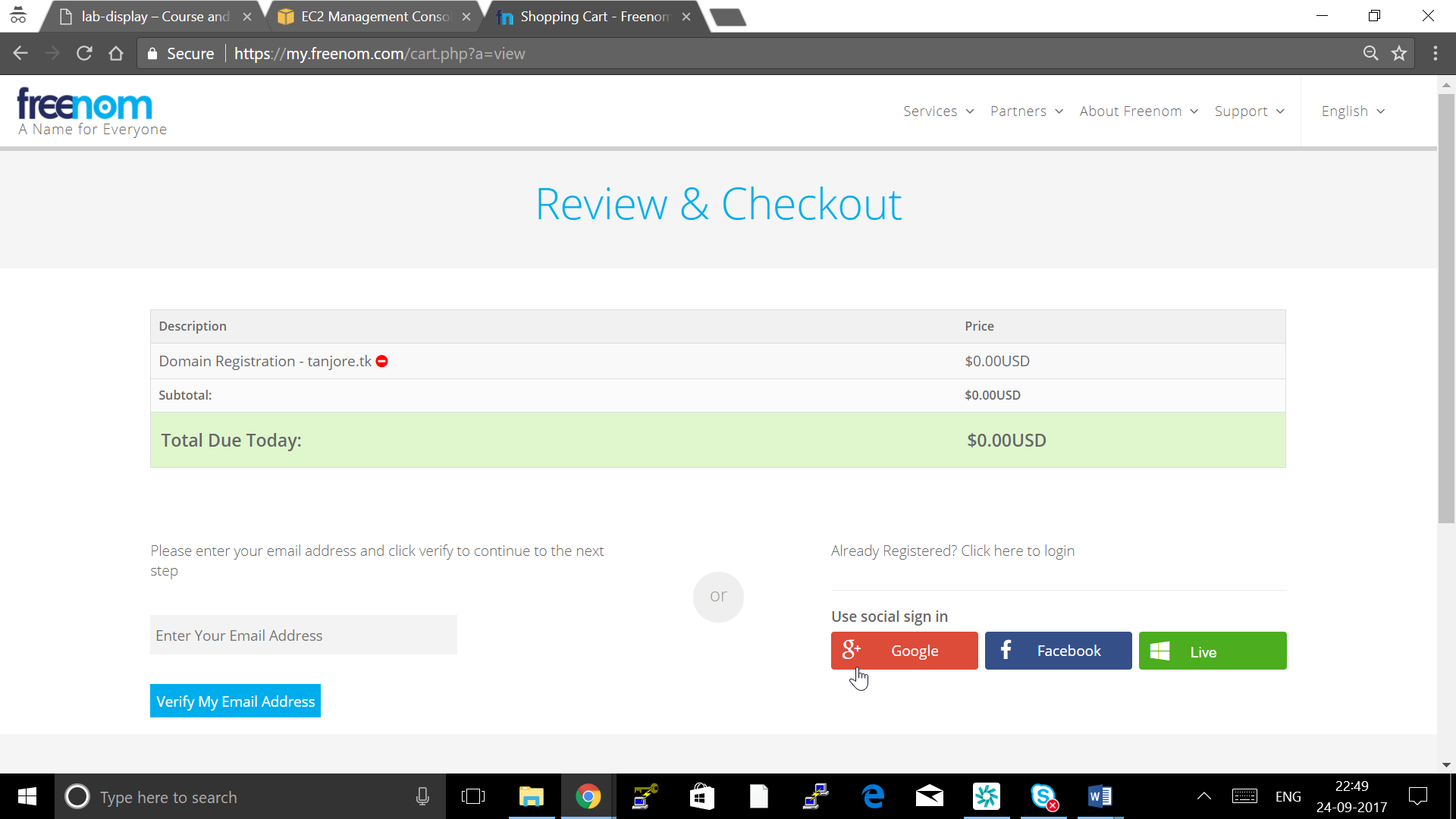


Click on Checkout

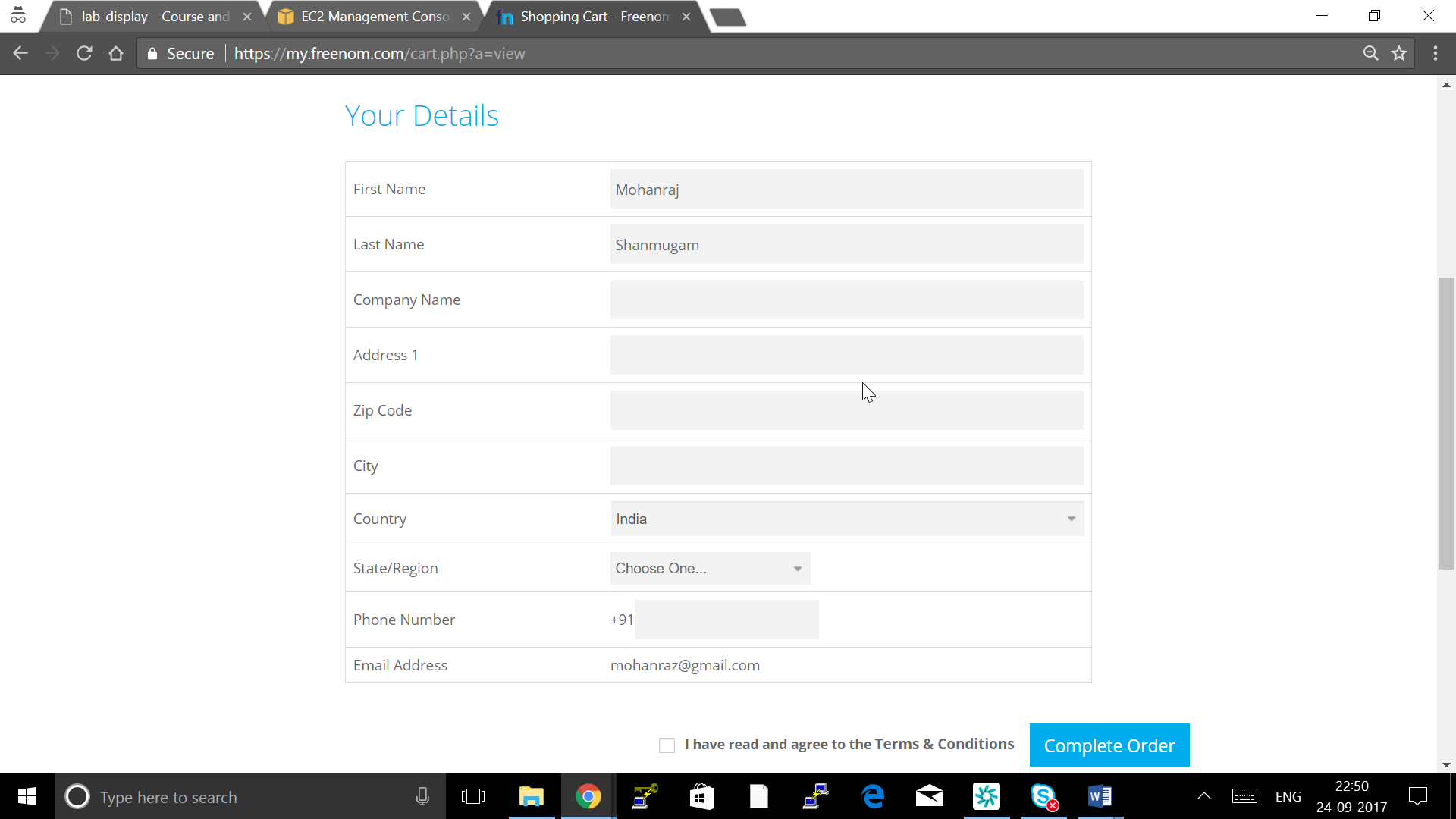
Click Continue

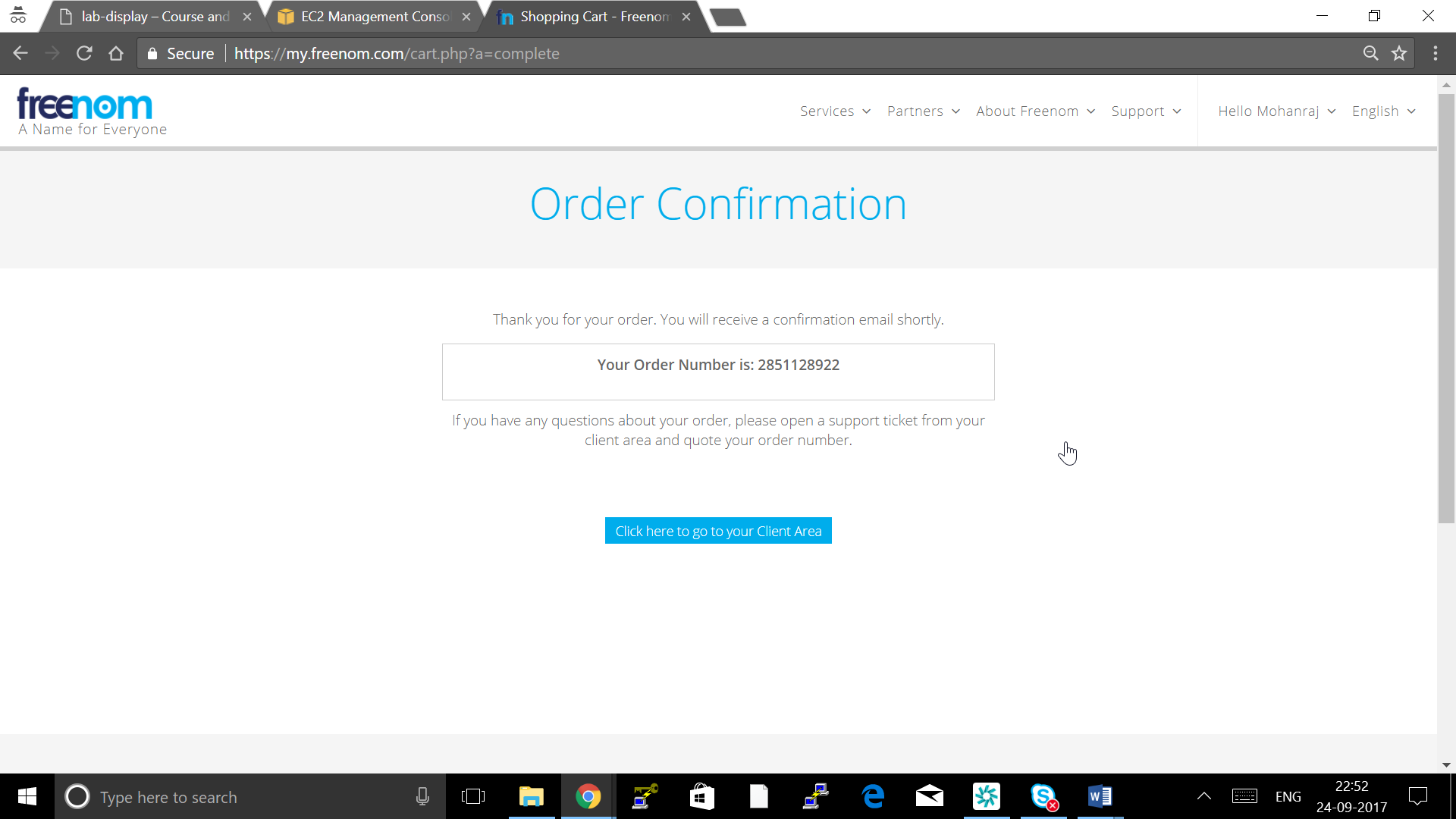


Click google login with your gmail id

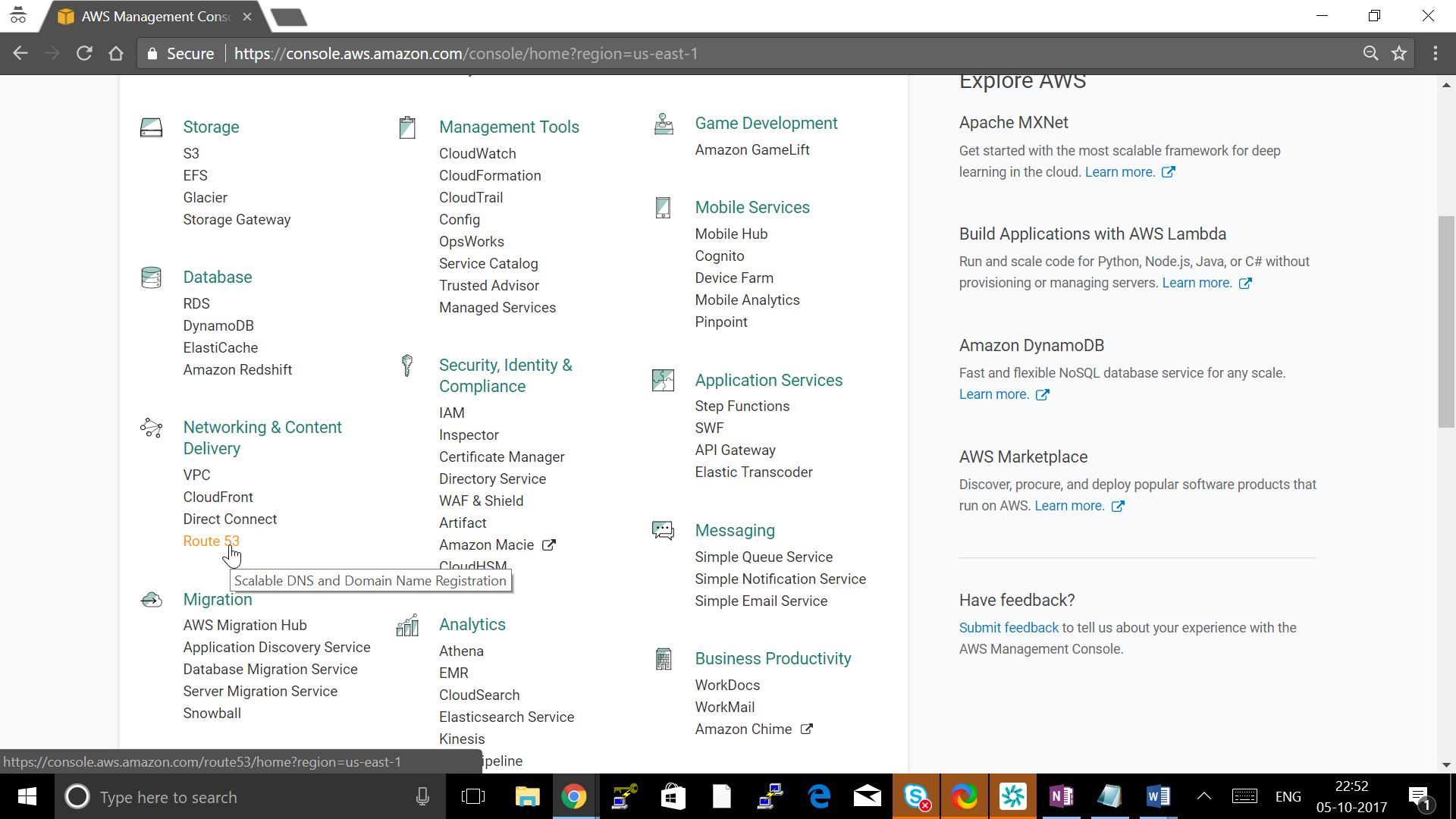


Fill all your details ( This can be dummy details ) and click Complete Order





Go to AWS Route-53



Click Hosted zone



Create Hosted Zone

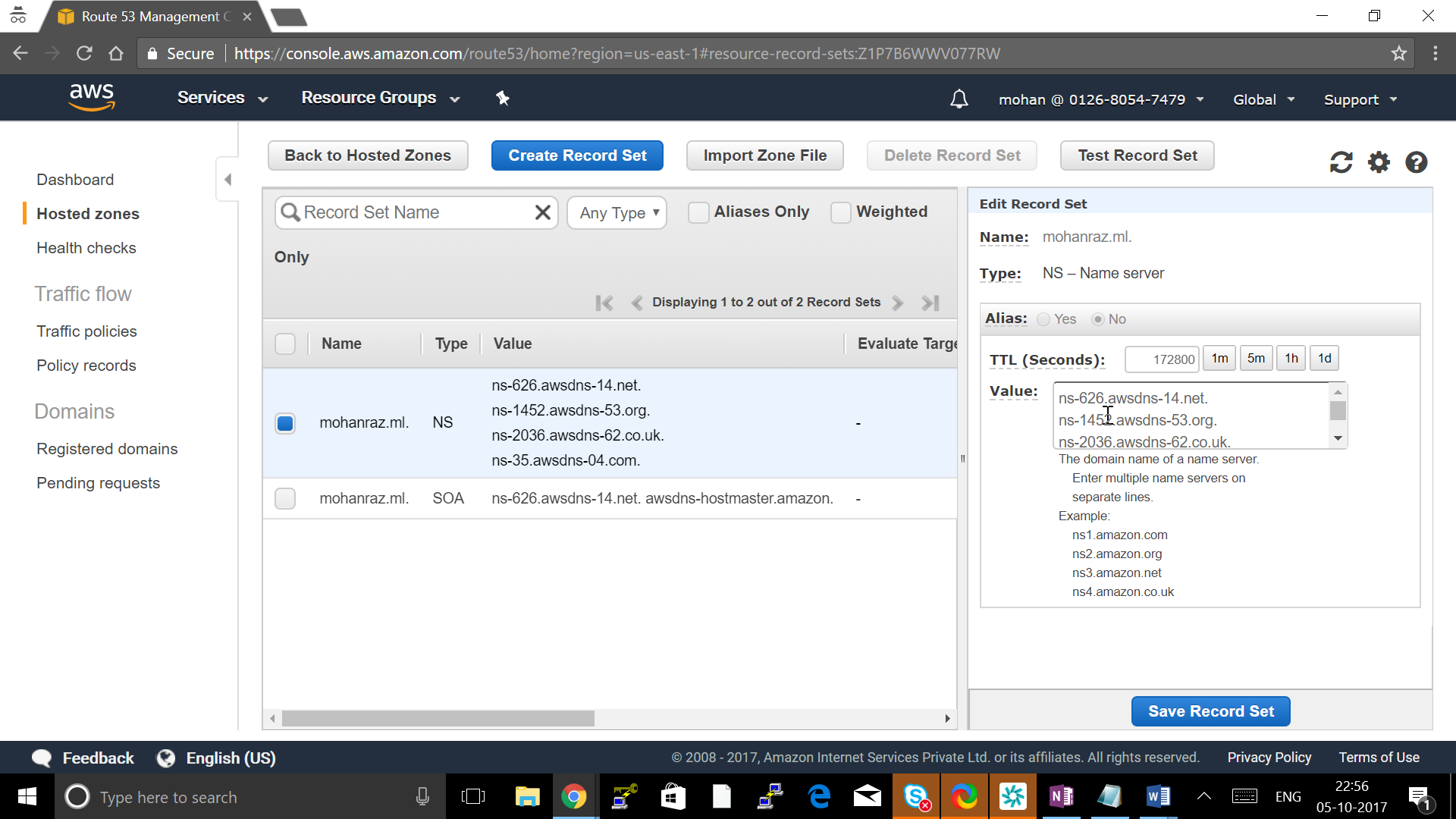
Give : Domain Name created

Type: Public hosted zone

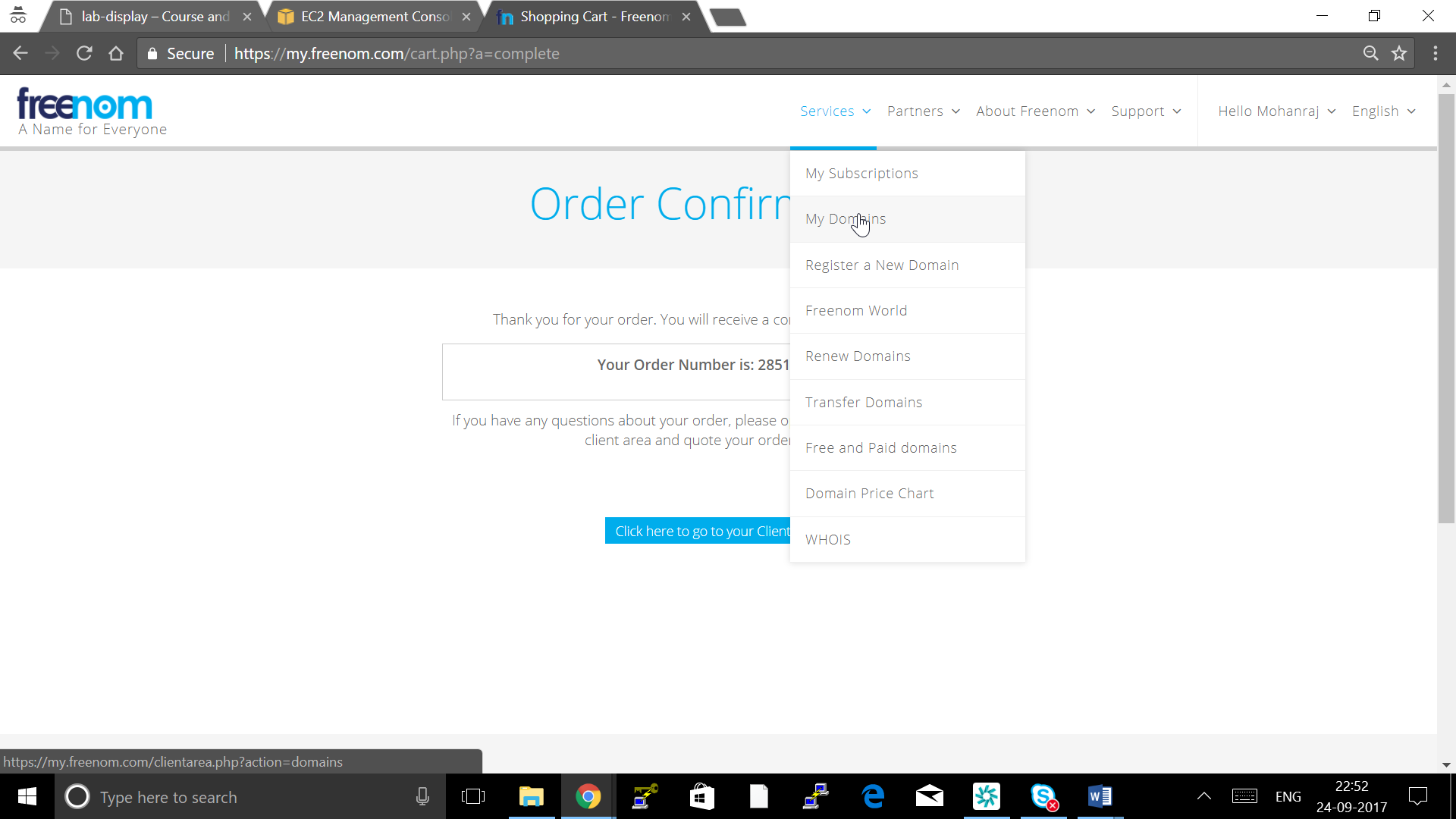
Click create



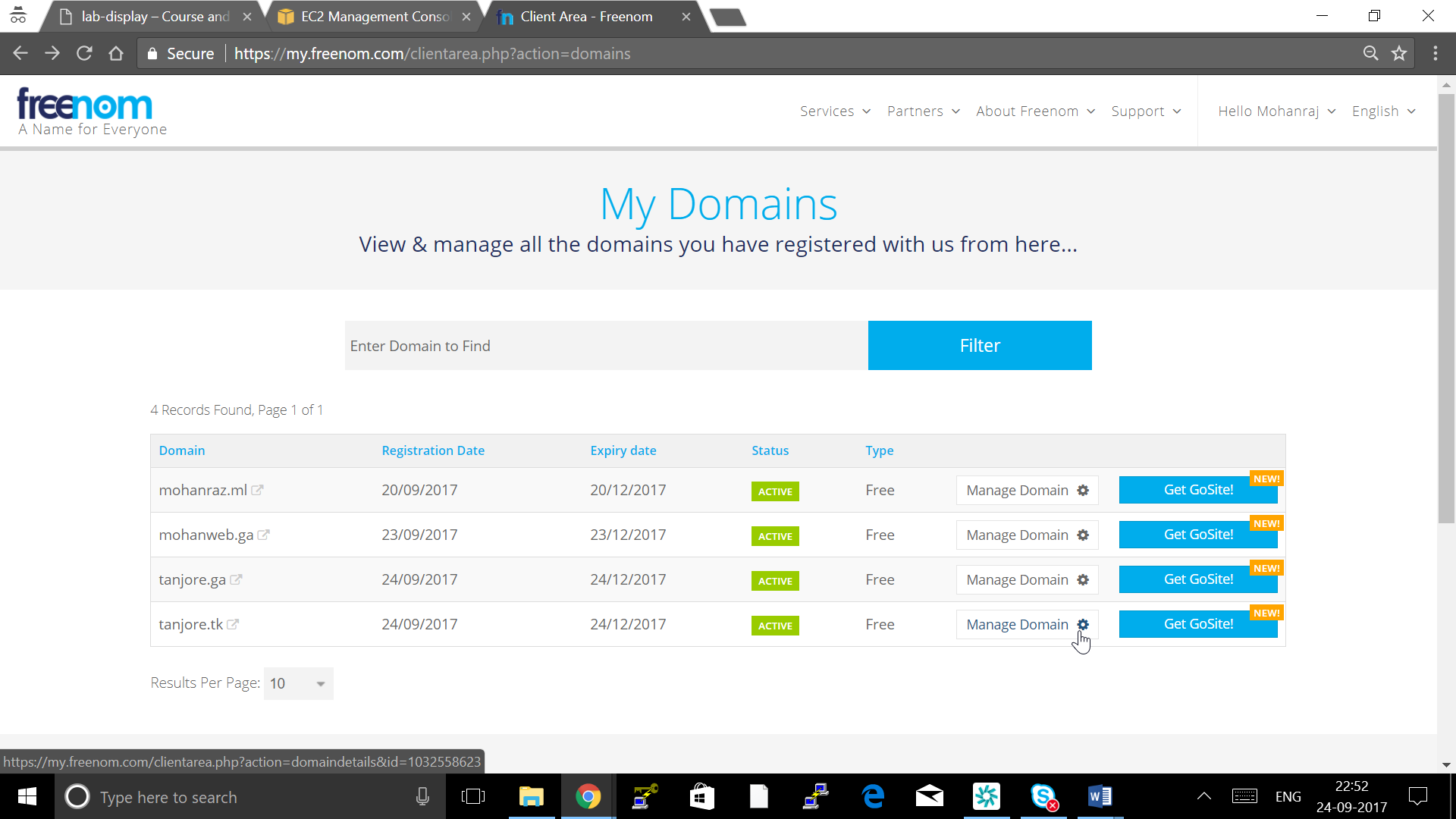
Click on NS Record and note down all 4 NS DNS



Go to Freenom-> services -> My Domain

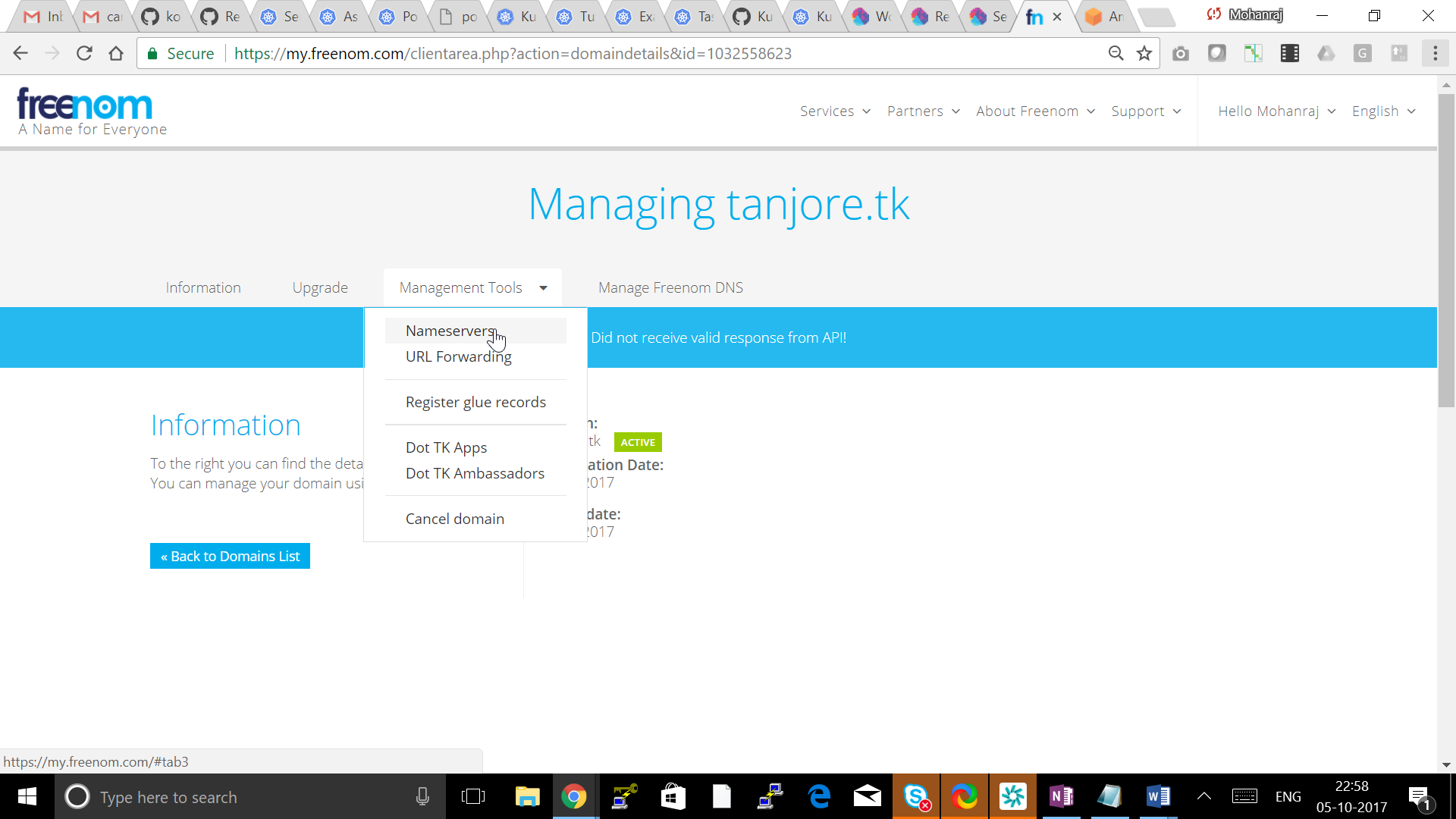


Click on Manage Domain

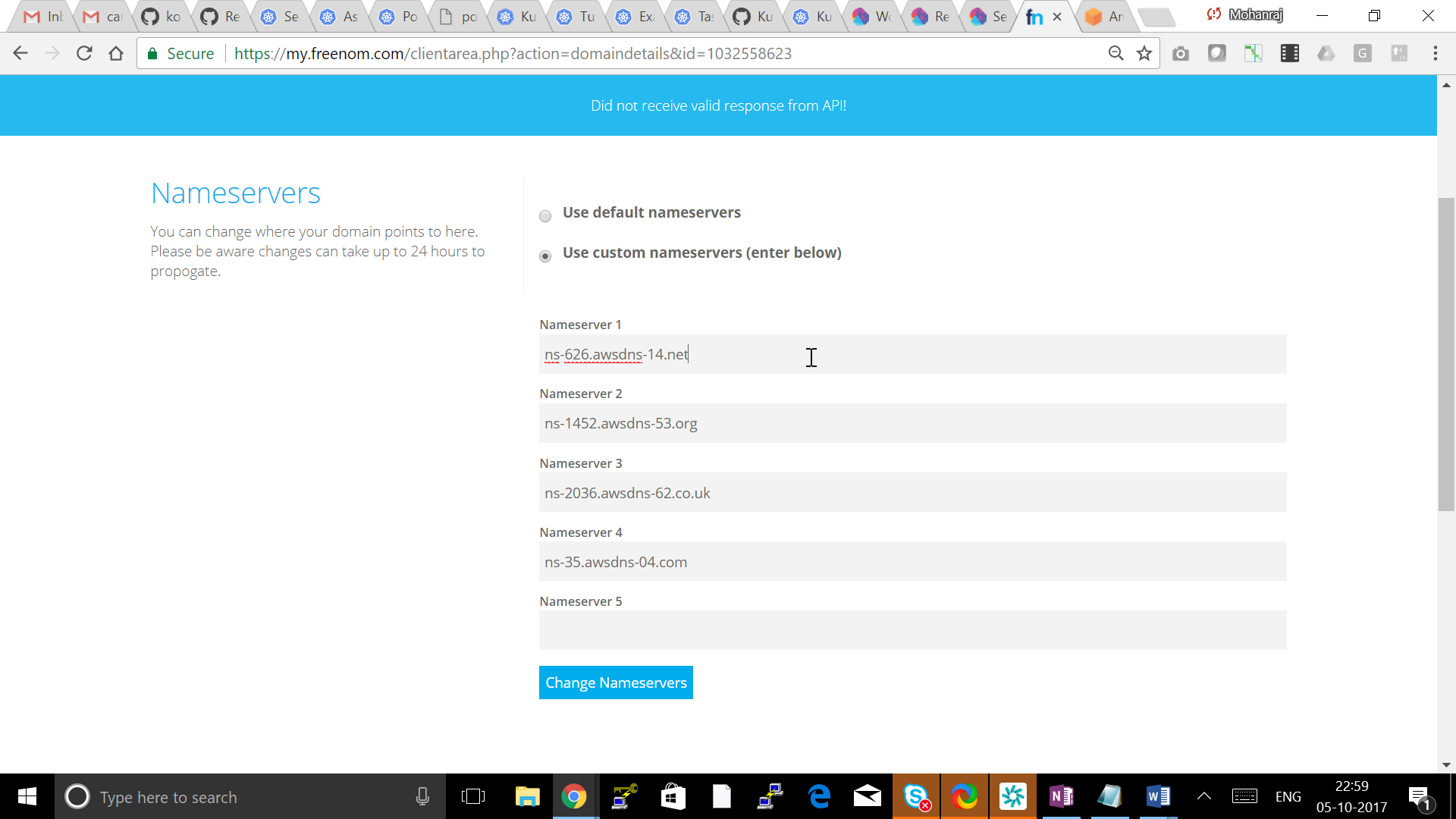




Go to Manage Tools -> Nameserver



Put all 4 Name server shown in Route53 NS record and Click Change name Server



**Testing your DNS setup**

This section is not be required if a gossip-based cluster is created.

You should now able to dig your domain (or subdomain) and see the AWS Name Servers on the other end.

## [ec2-user@ip-10-53-3-250 ~]$ dig ns mohanraz.ml

## ; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.62.rc1.56.amzn1 <<>> ns mohanraz.ml

## ;; global options: +cmd

## ;; Got answer:

## ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16447

## ;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0

## ;; QUESTION SECTION:

## ;mohanraz.ml. IN NS

## ;; ANSWER SECTION:

## mohanraz.ml. 57 IN NS ns-1452.awsdns-53.org.

## mohanraz.ml. 57 IN NS ns-2036.awsdns-62.co.uk.

## mohanraz.ml. 57 IN NS ns-35.awsdns-04.com.

## mohanraz.ml. 57 IN NS ns-626.awsdns-14.net.

## ;; Query time: 0 msec

## ;; SERVER: 10.53.0.2#53(10.53.0.2)

## ;; WHEN: Fri Oct 6 01:23:37 2017

## ;; MSG SIZE rcvd: 168

## Cluster State storage

In order to store the state of your cluster, and the representation of your cluster, we need to create a dedicated S3 bucket for kops to use. This bucket will become the source of truth for our cluster configuration. In this guide we'll call this bucket example-com-state-store, but you should add a custom prefix as bucket names need to be unique.

[ec2-user@ip-10-53-3-250 ~]$ aws s3api create-bucket --bucket kube-mohanraz-ml-state-store --region us-east-1

{

"Location": "/kube-mohanraz-ml-state-store"

}

# [ec2-user@ip-10-53-3-250 ~]$ aws s3api put-bucket-versioning --bucket kube-mohanraz-ml-state-store --versioning-configuration Status=Enabled

# Creating your first cluster

## [ec2-user@ip-10-53-3-250 ~]$ export NAME=kubecluster.mohananz.tk

## [ec2-user@ip-10-53-3-250 ~]$ export KOPS\_STATE\_STORE=s3://kube-mohanraz-ml-state-store

## Create cluster configuration

kops create cluster \

--zones=us-east-1a,us-east-1b,us-east-1c \

--master-zones=us-east-1a,us-east-1b,us-east-1c \

--node-count=4 \

--node-size=t2.small \

--master-size=t2.small \

--name ${NAME}

## Customize Cluster Configuration

Now we have a cluster configuration, we can look at every aspect that defines our cluster by editing the description.

kops edit cluster ${NAME}

This opens your editor (as defined by $EDITOR) and allows you to edit the configuration. The configuration is loaded from the S3 bucket we created earlier, and automatically updated when we save and exit the editor.

We'll leave everything set to the defaults for now, but the rest of the kops documentation covers additional settings and configuration you can enable.

## Build the Cluster

Now we take the final step of actually building the cluster. This'll take a while. Once it finishes you'll have to wait longer while the booted instances finish downloading Kubernetes components and reach a "ready" state.

kops update cluster ${NAME} --yes

## Validate and Wait until cluster is running

## [ec2-user@ip-10-53-3-250 ~]$ kops validate cluster

## Using cluster from kubectl context: testcluster.mohanraz.ml

## Validating cluster testcluster.mohanraz.ml

## INSTANCE GROUPS

## NAME ROLE MACHINETYPE MIN MAX SUBNETS

## master-us-east-1a Master t2.medium 1 1 us-east-1a

## master-us-east-1b Master t2.medium 1 1 us-east-1b

## master-us-east-1c Master t2.medium 1 1 us-east-1c

## nodes Node t2.medium 4 4 us-east-1a,us-east-1b,us-east-1c

## NODE STATUS

## NAME ROLE READY

## ip-172-20-107-129.ec2.internal master True

## ip-172-20-125-104.ec2.internal node True

## ip-172-20-54-42.ec2.internal master True

## ip-172-20-59-193.ec2.internal node True

## ip-172-20-82-35.ec2.internal node True

## ip-172-20-89-128.ec2.internal master True

## ip-172-20-90-6.ec2.internal node True

## Your cluster testcluster.mohanraz.ml is ready

## Use the Cluster

Remember when you installed kubectl earlier? The configuration for your cluster was automatically generated and written to ~/.kube/config for you!

A simple Kubernetes API call can be used to check if the API is online and listening.

Let's use kubectl to check the nodes.

kubectl get nodes

You will see a list of nodes that should match the --zones flag defined earlier. This is a great sign that your Kubernetes cluster is online and working.

Also kops ships with a handy validation tool that can be ran to ensure your cluster is working as expected.

kops validate cluster

You can look at all the system components with the following command.

kubectl -n kube-system get pods