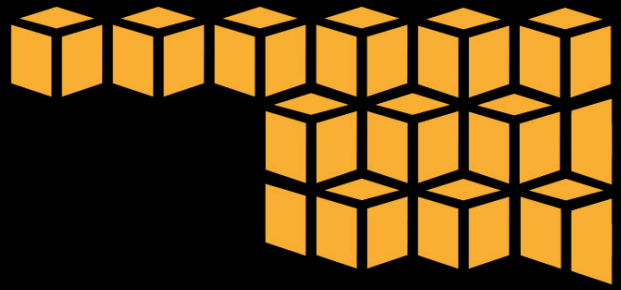


aws okc

Lab: Build a Redundant WordPress Stack in AWS Part 2

Nathan Aker
Director of IT @ SYMPHONY TALENT

Overview

Lab:

Build a Redundant WordPress Stack in AWS

Intent:

- Get hands on with AWS services with a realistic use case
- Gain confidence to jump in and play with AWS services
- Stay on track to get through the lab (Exposure ≠ Training)

= Warning – Cost Associated!!!

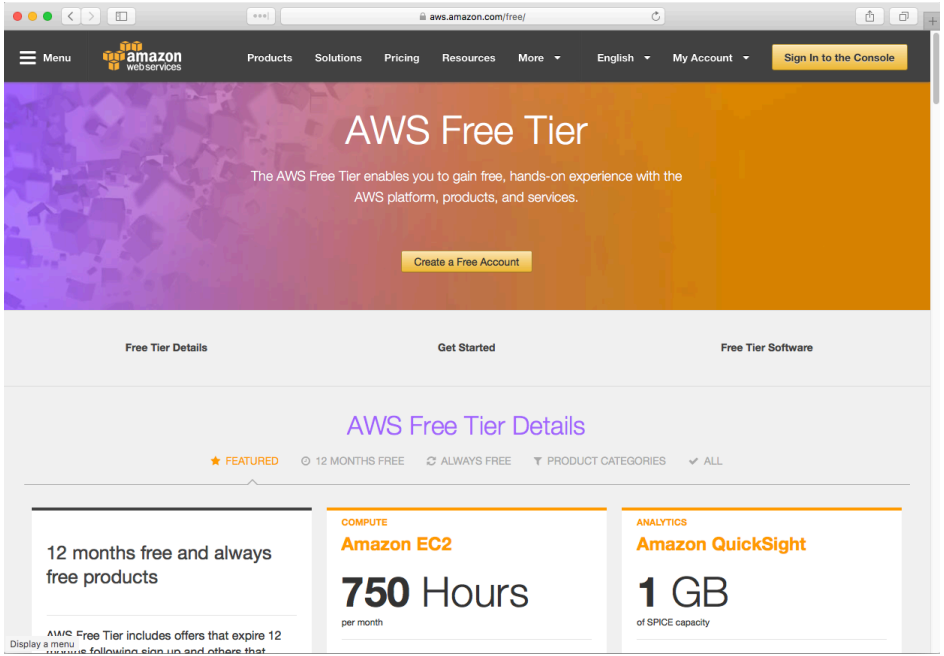
Acknowledgement

Lab Taken from:
A Cloud Guru's "AWS Certified Solutions Architect Associate" course

https://acloud.guru/learn/aws-certified-solutions-architect-associate

Prerequisites

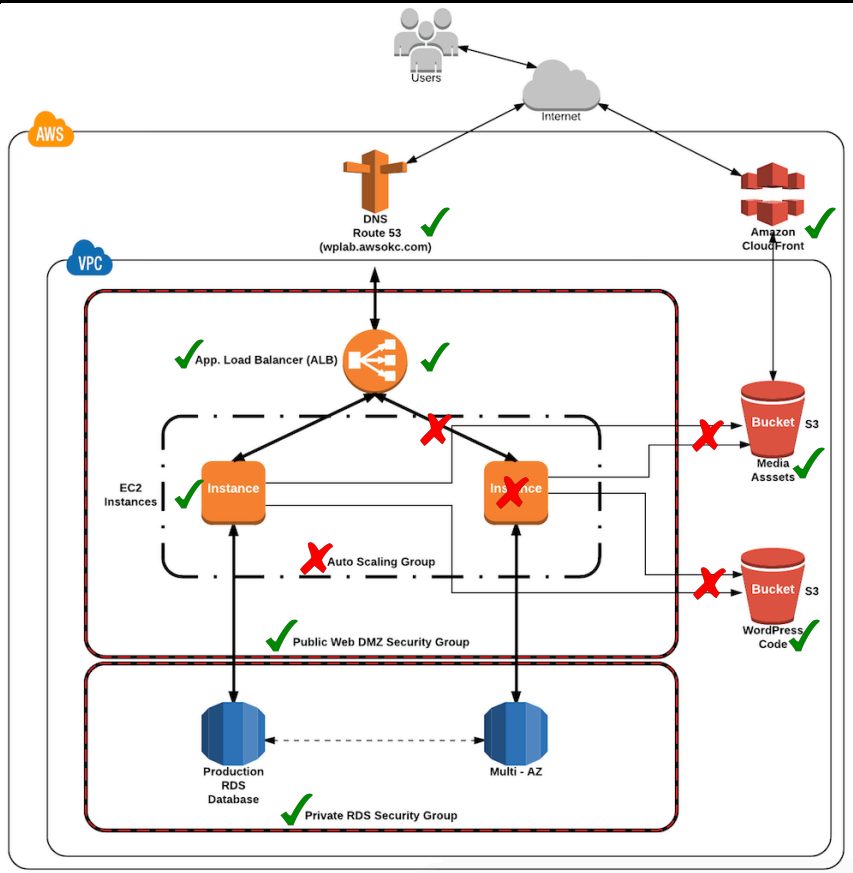
- AWS Free Tier Account: aws.amazon.com/free



Prerequisites

- Download Resources from:
  - https://github.com/awsokc/wplab

Recap

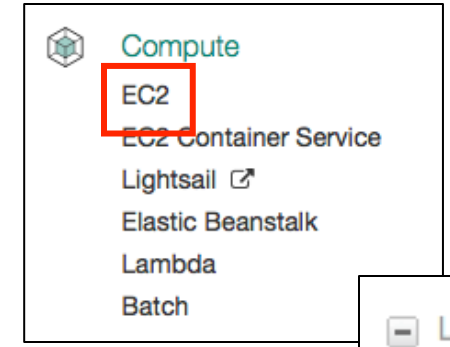

Users
Internet
AWS
DNS
Route 53
(wplab.awsokc.com)
Amazon
CloudFront
VPC
App. Load Balancer (ALB)
Bucket
S3
Media
Asssets
EC2
Instances
Instance
Instance
Auto Scaling Group
Bucket
S3
WordPress
Code
Public Web DMZ Security Group
Production
RDS
Database
Multi - AZ
Private RDS Security Group

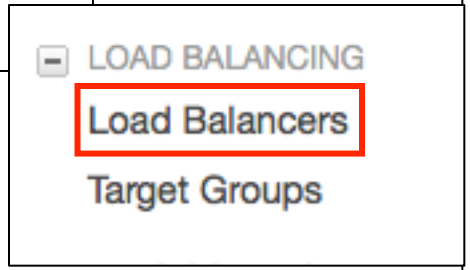
Now the Fun Stuff

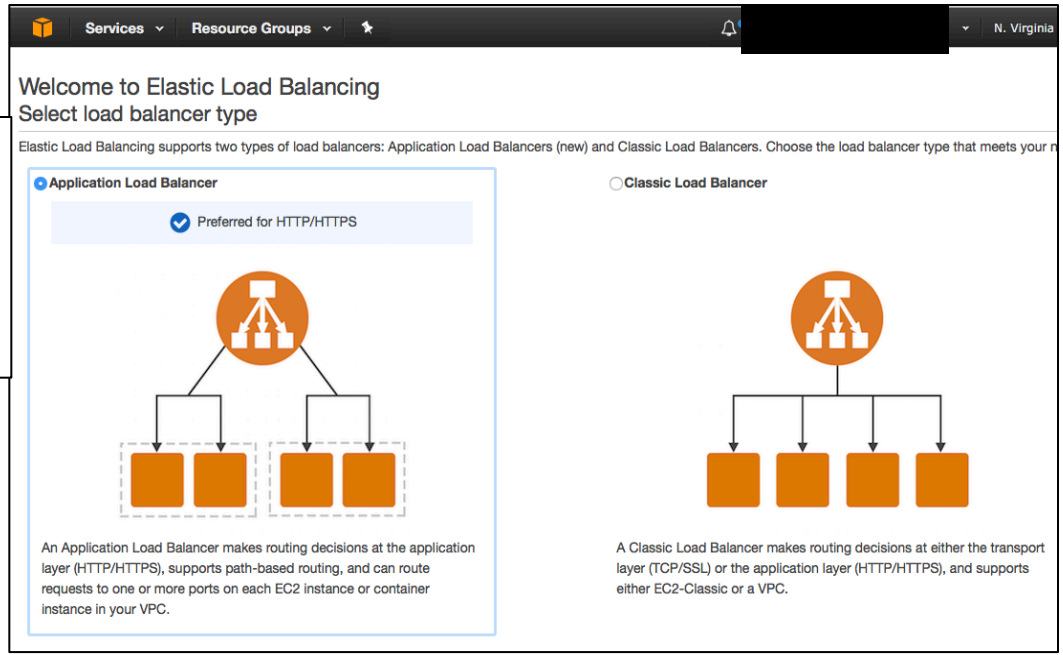
We previously setup a functioning WordPress instance, but one that only had one single web server and lacked any kind of resilience or self-healing capabilities.

Now, we'll add Resilience via Auto-Scaling and CloudFront

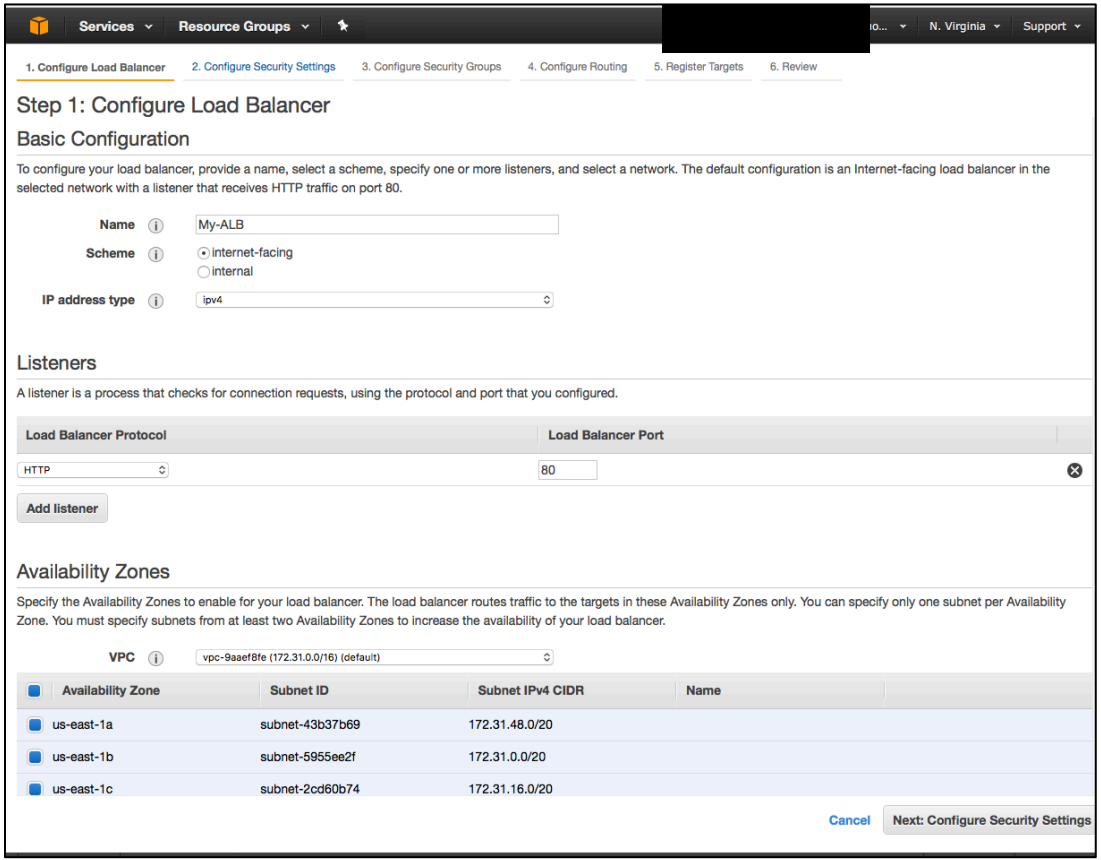
Re-Create ALB (ELB)





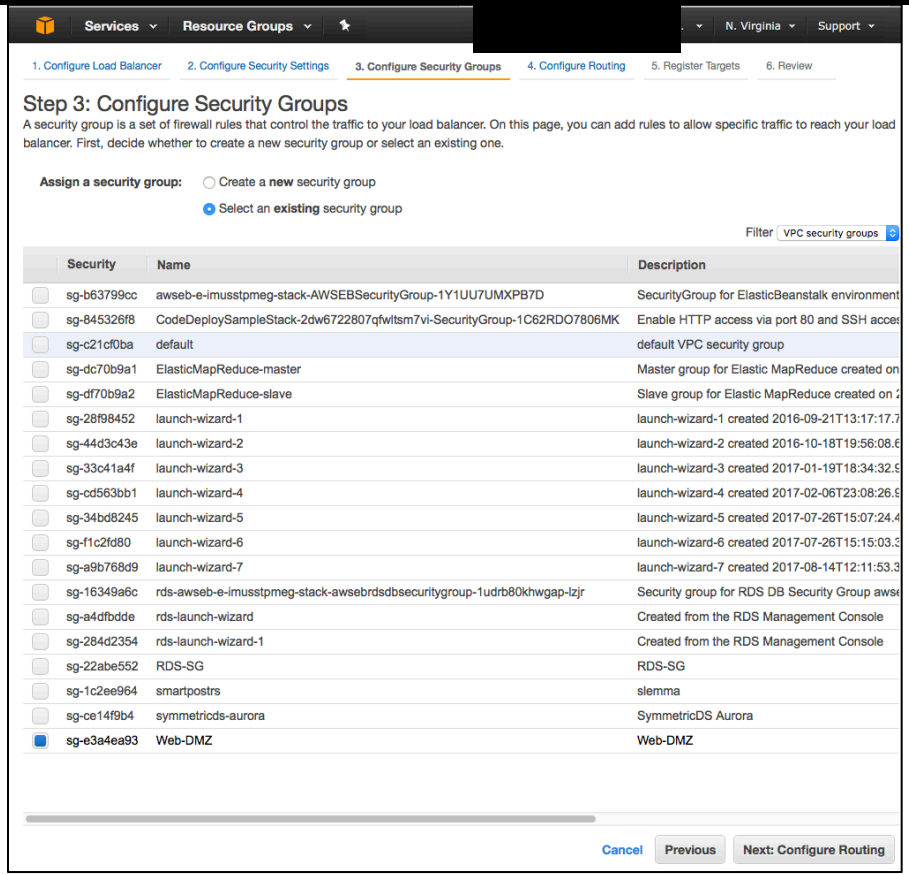


Re-Create ALB (ELB)

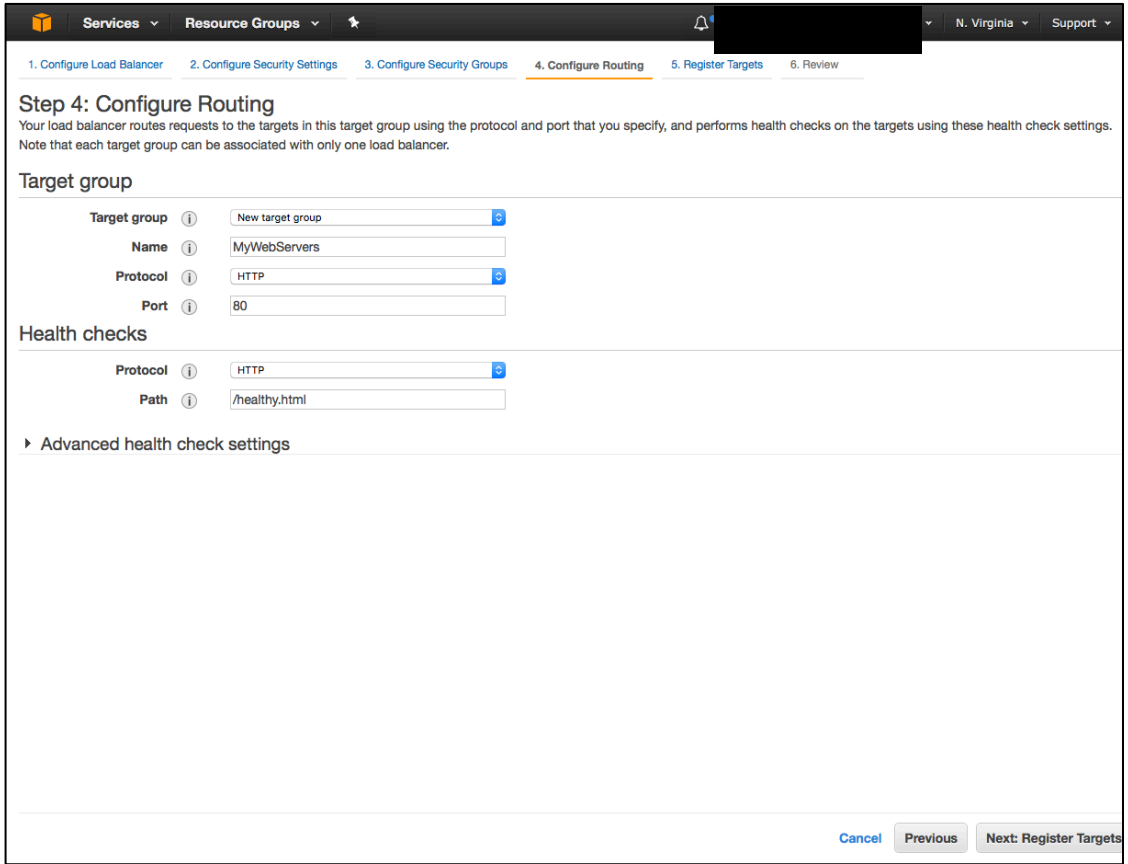


Re-Create ALB (ELB)

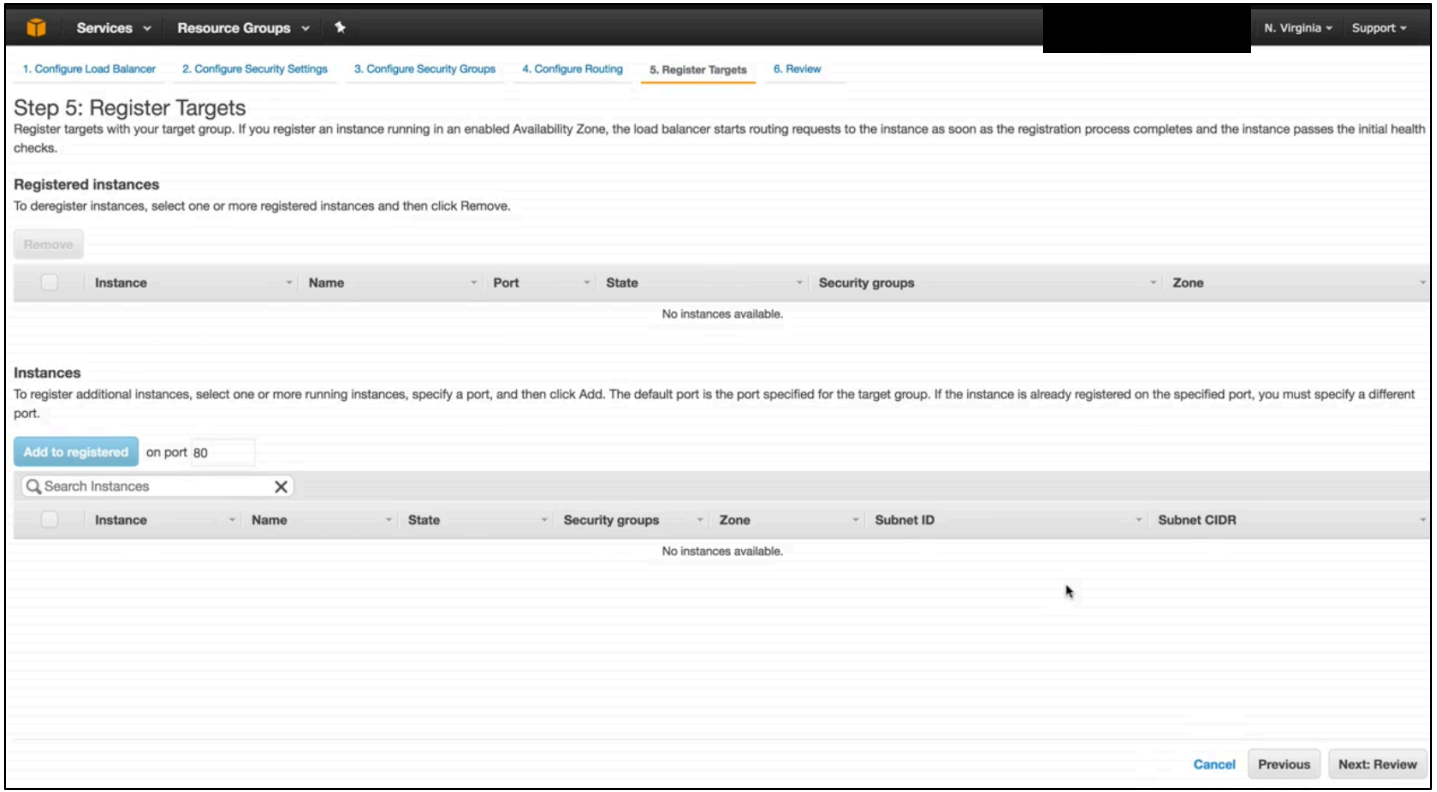
aws okc



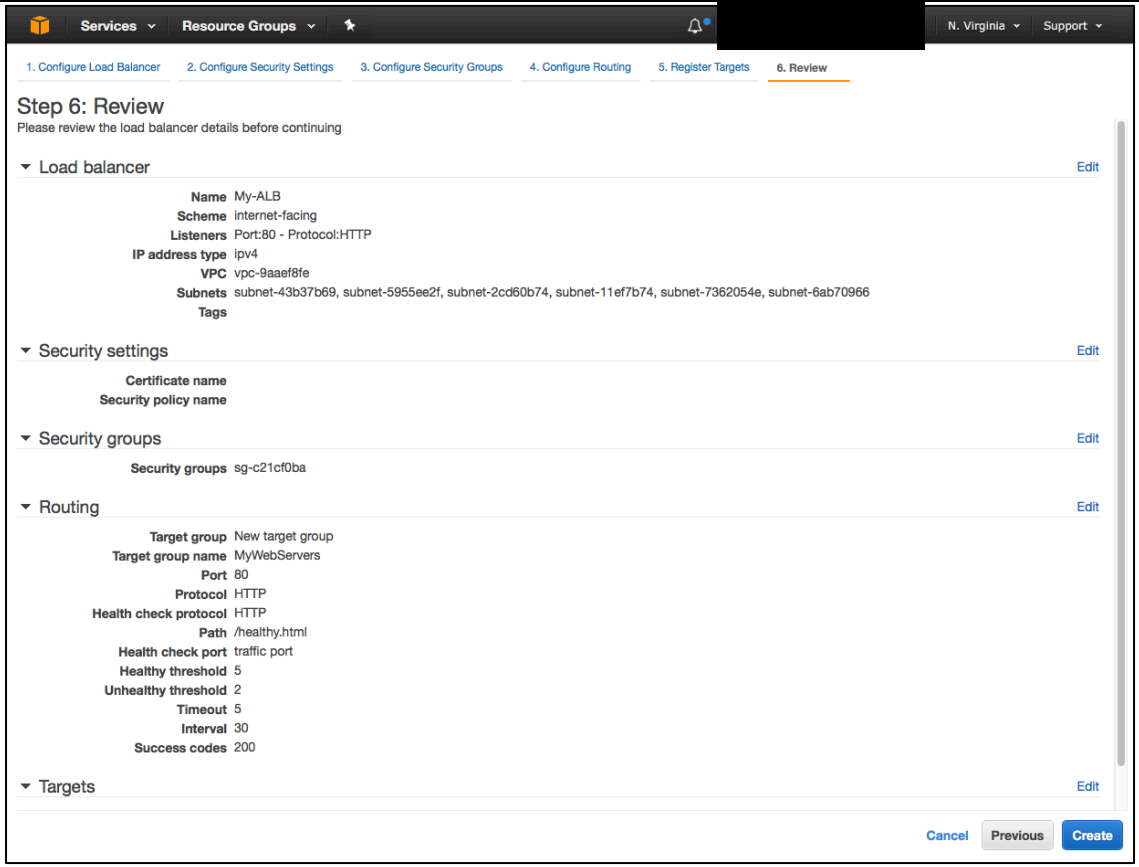
Re-Create ALB (ELB)



Re-Create ALB (ELB)

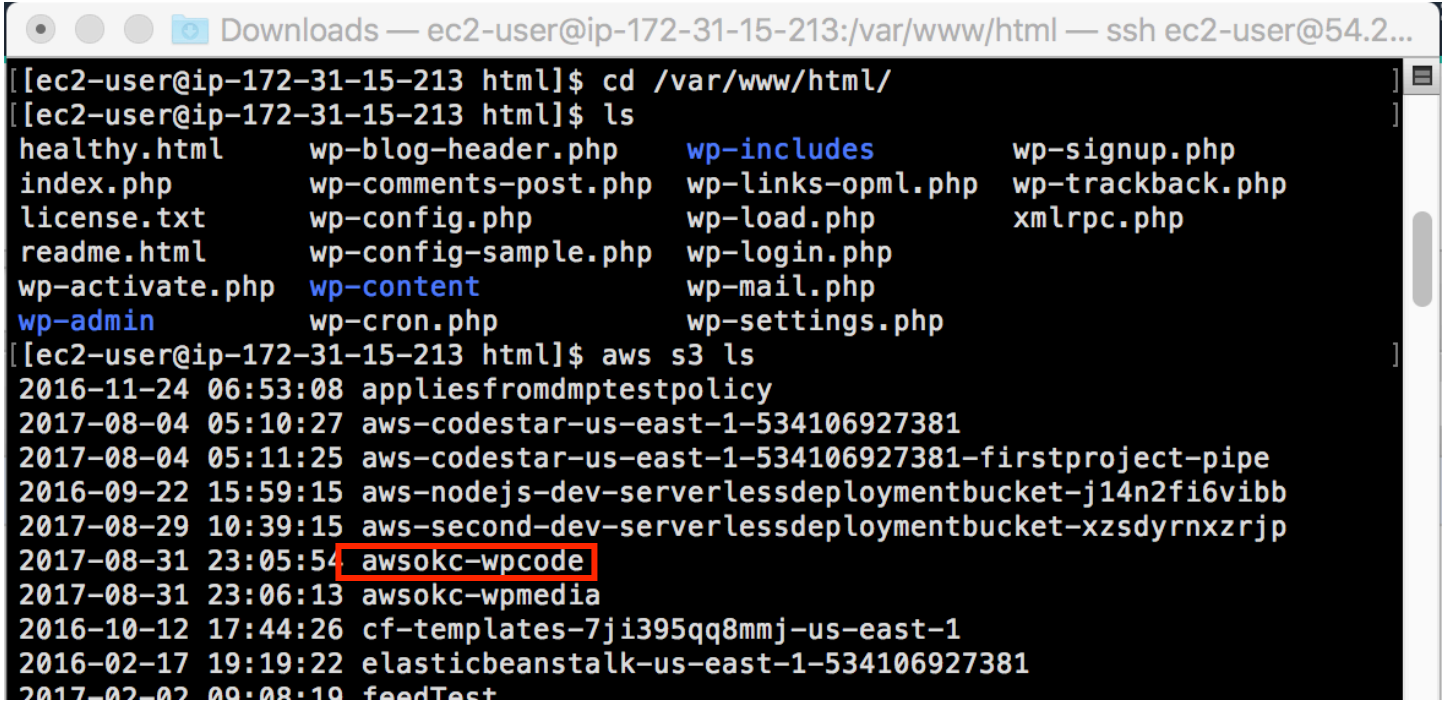


Re-Create ALB (ELB)



Step 1 – Backup WP Files

List S3 bucket names with $ aws s3 ls

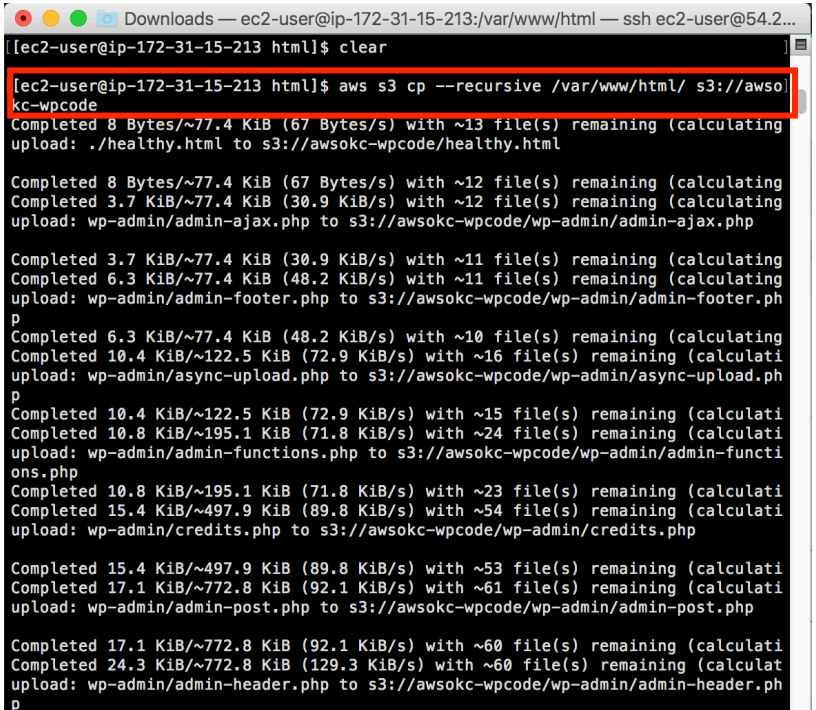


Step 1 – Backup WP Files

aws okc

Copy WP code to S3 bucket with:

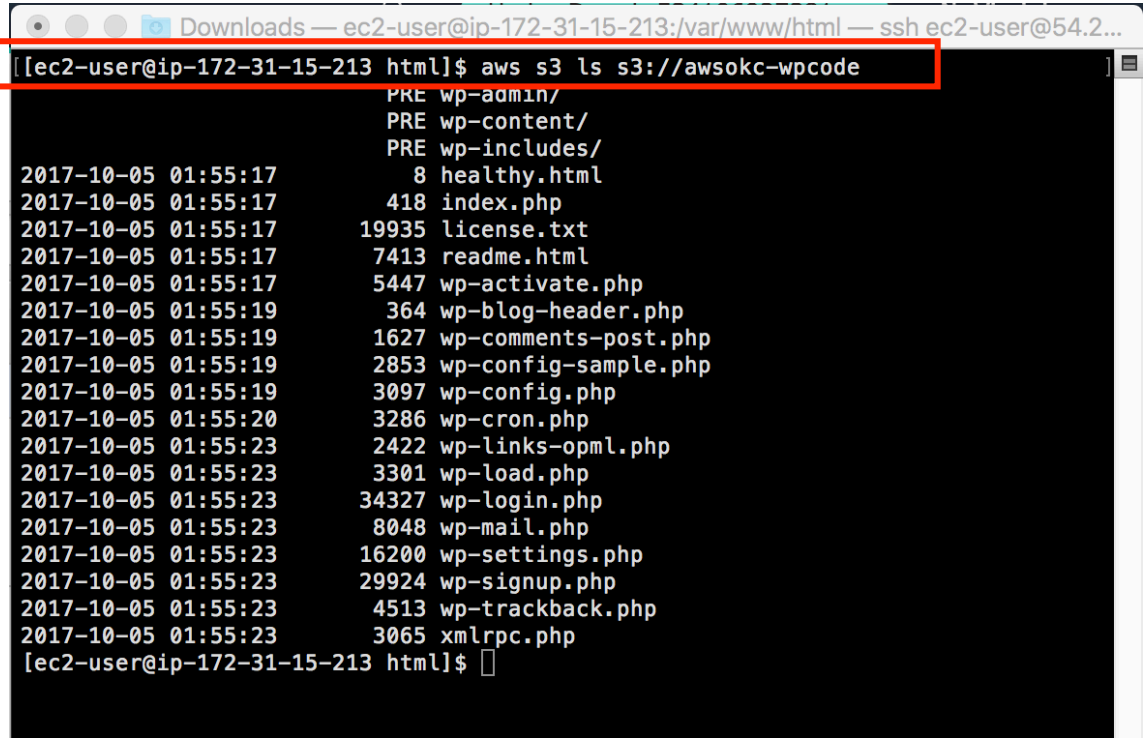
$ aws s3 cp –recursive /var/www/html/ s3://awsokc-wpcode



Step 1 – Backup WP Files

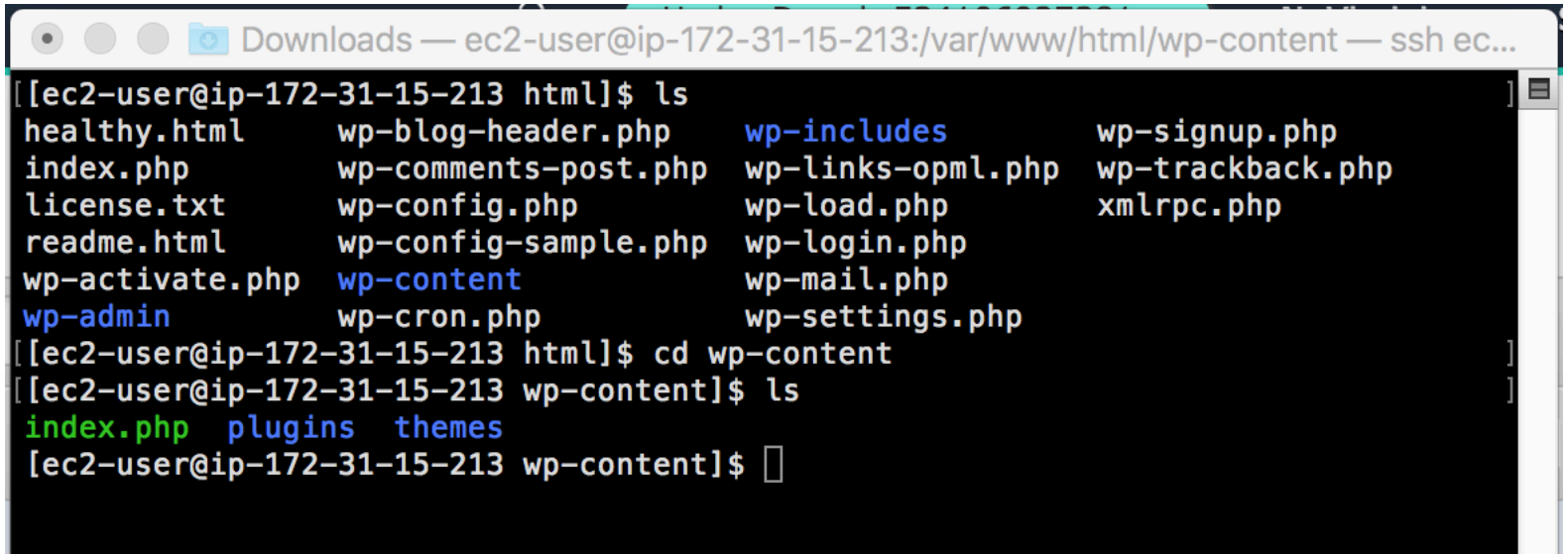
Confirm successful copy with:

$ aws s3 ls s3://yourbucket



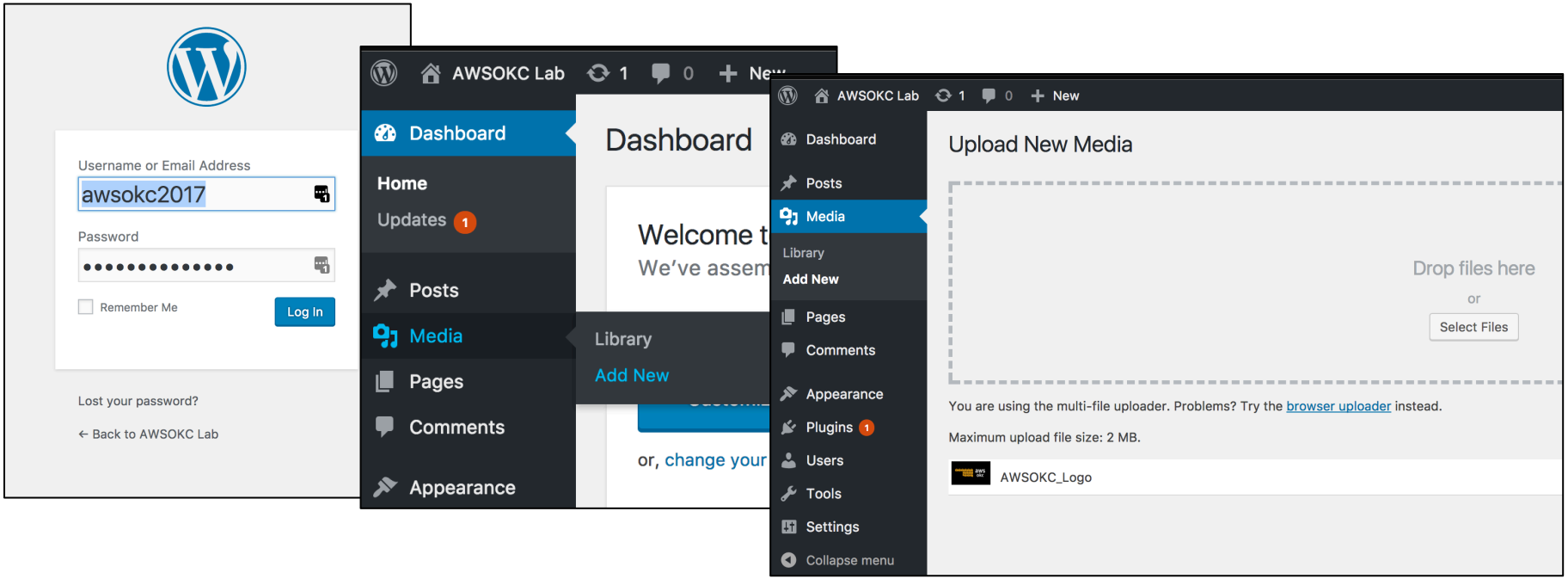
Step 2 – Upload an Image File

First, confirm that there are no images in wp-content directory:



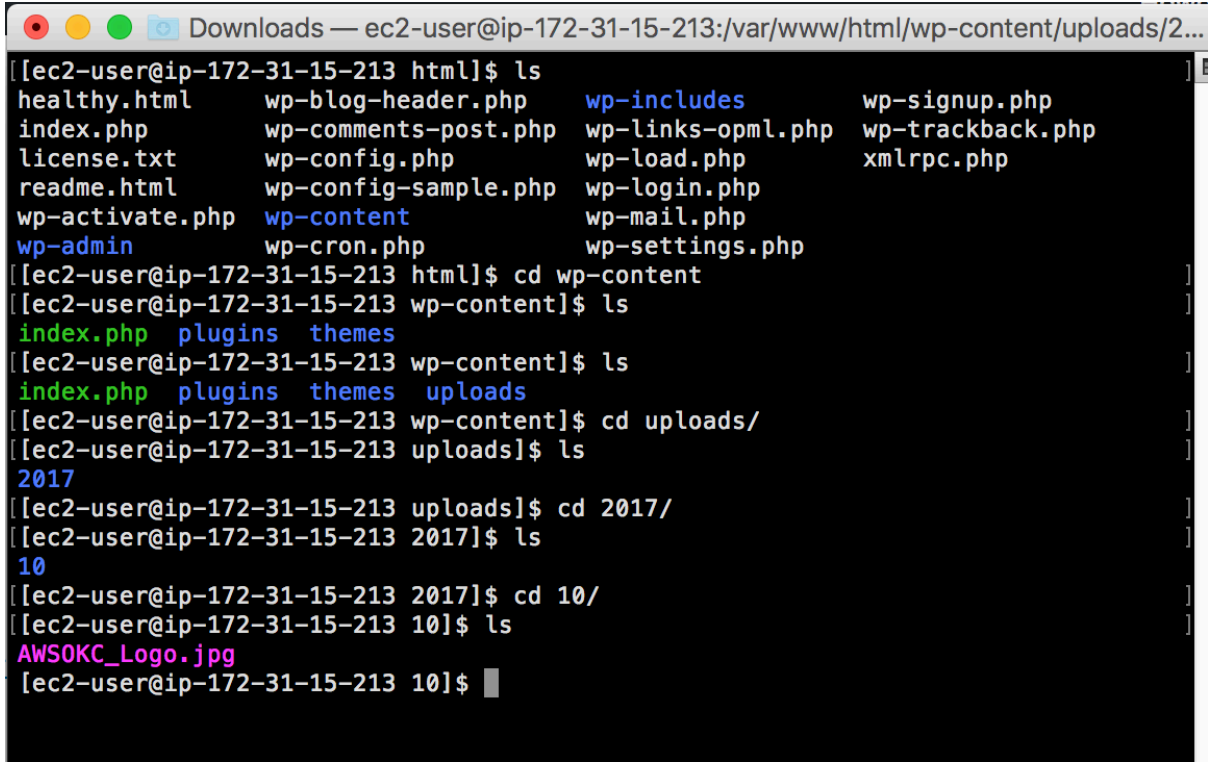
Step 2 – Upload an Image File

Upload an image:



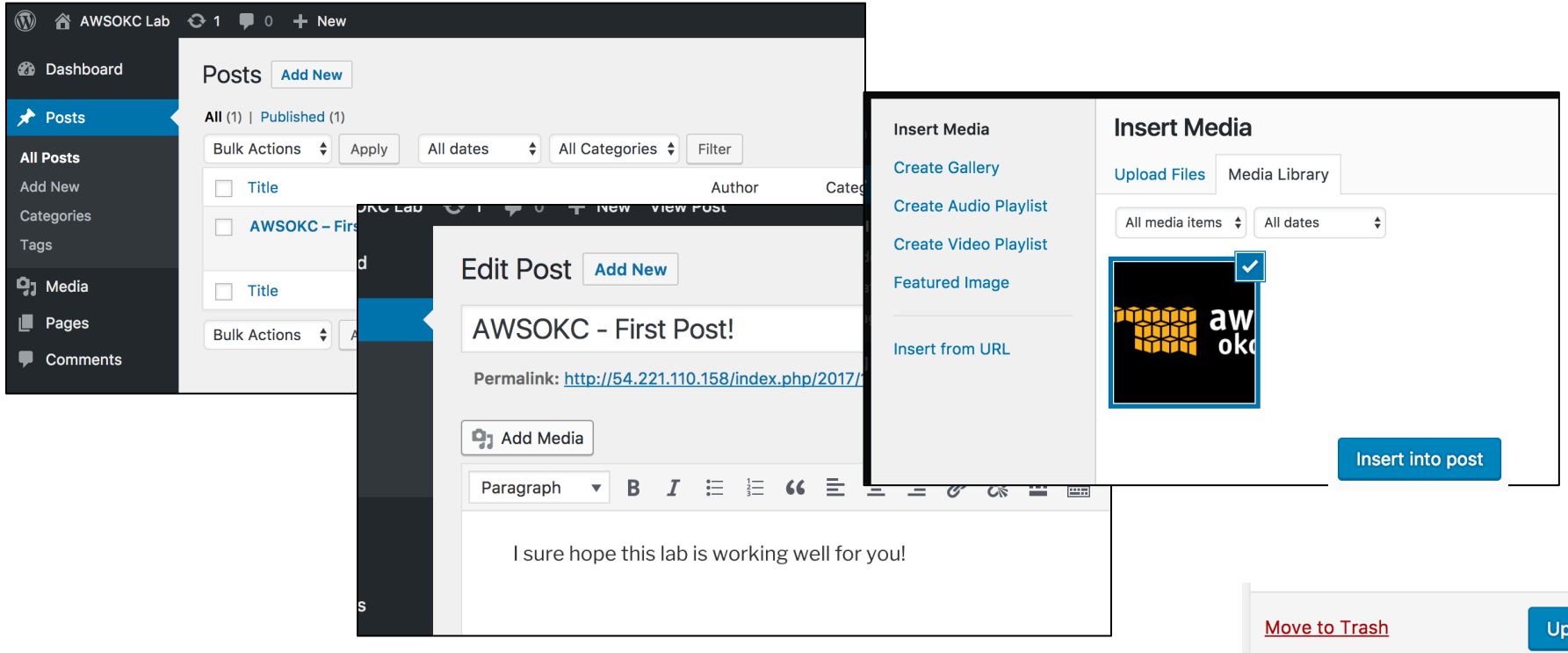
Step 2 – Upload an Image File

Confirm new uploads directory



Step 2 – Upload an Image File

Insert Pic into Post



Step 2 – Upload an Image File

Confirm pic in post

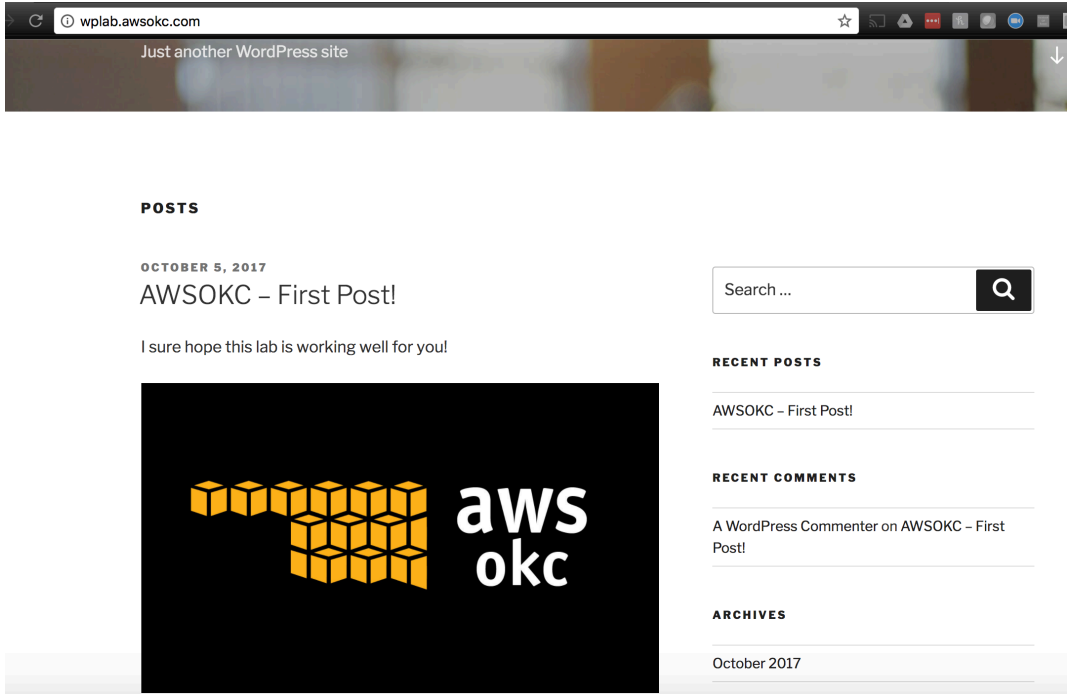


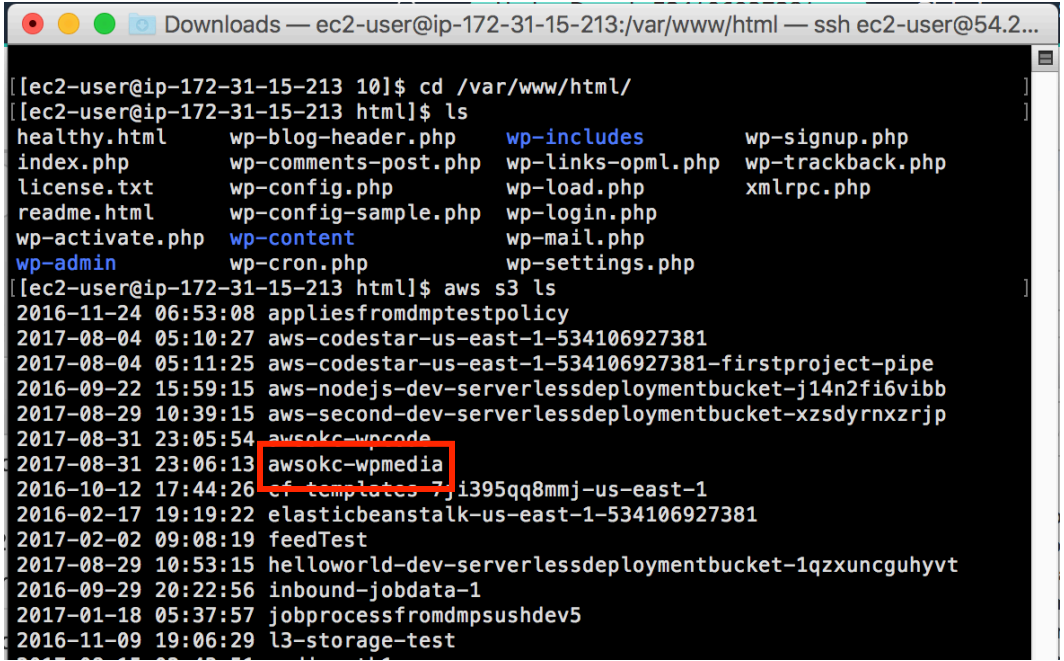
Image Address: http://wplab.awsokc.com/wp-content/uploads/2017/10/AWSOKC_BlackBackground.jpg

Step 3 – Rewrite Images to S3 / CF

Part 1 – update WP to push images to S3 (served through CloudFront)

Part 2 – Update WP to reference Cloudfront.

First confirm S3 bucket - $ aws s3 ls

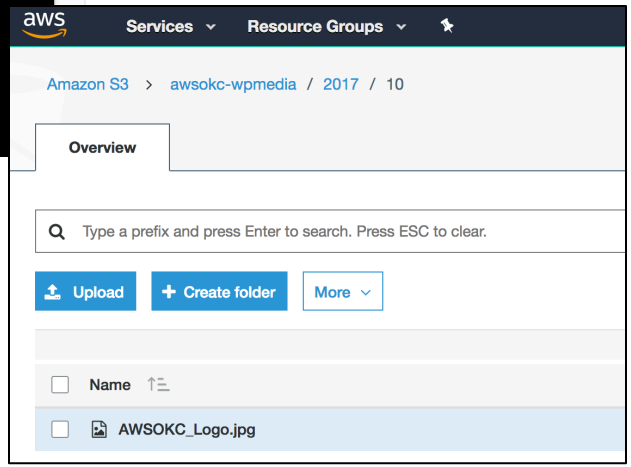


Step 3 – Rewrite Images to S3 / CF

S3 Sync Command:

$ aws s3 sync --delete /var/www/html/wp-content/uploads/ s3://awsokc-wpmedia

```
Downloads — ec2-user@ip-172-31-15-213:/var/www/html — ssh ec2-user@54.2...
[ec2-user@ip-172-31-15-213 html]$ aws s3 sync --delete /var/www/html/wp-content/
uploads/ s3://awsokc-wpmedia
upload: wp-content/uploads/2017/10/AWSOKC_Logo.jpg to s3://awsokc-wpmedia/2017/1
0/AWSOKC_Logo.jpg
[ec2-user@ip-172-31-15-213 html]$ aws s3 ls s3://awsokc-wpmedia
                           PRE 2017/
[ec2-user@ip-172-31-15-213 html]$ cd
```



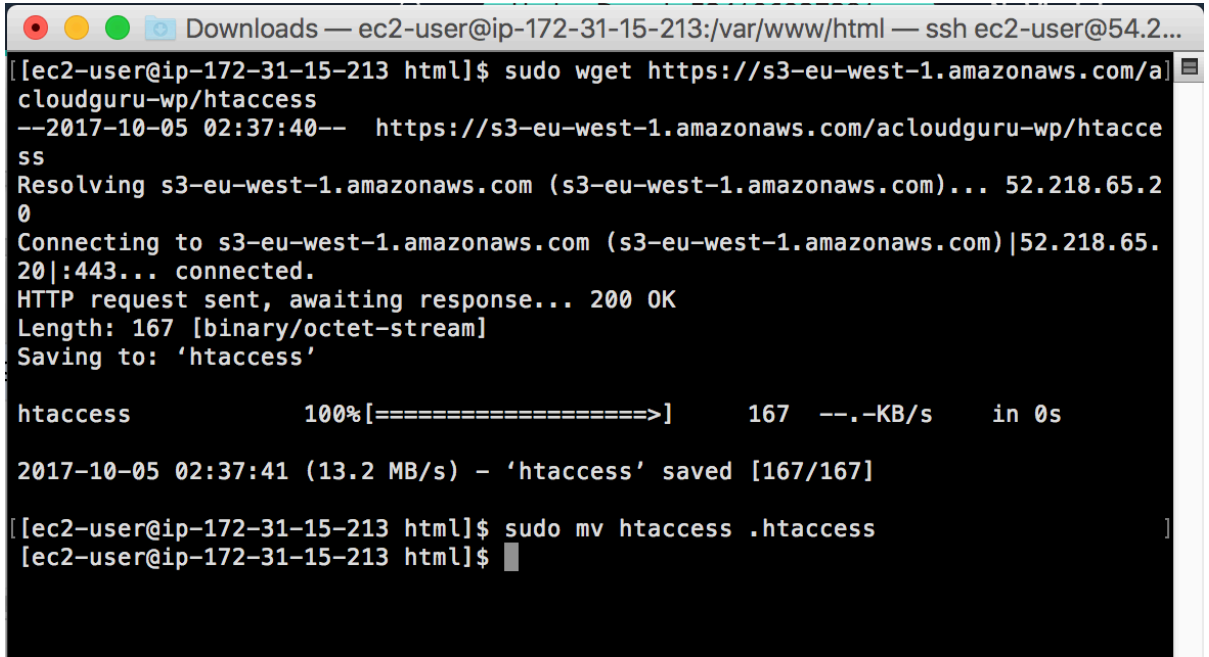
Step 3 – Rewrite Images to S3 / CF

aws okc

Add URL rewrite rule (while in html directory):

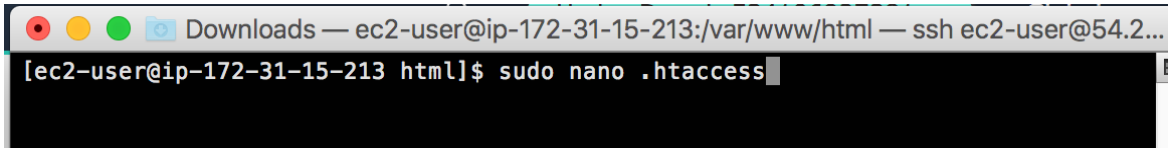
$ sudo wget https://s3-eu-west-1.amazonaws.com/acloudguru-wp/htaccess

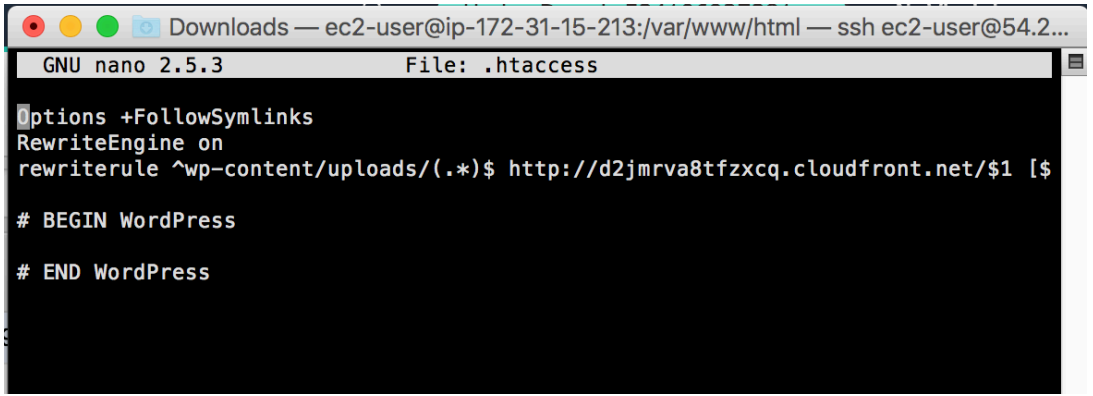
$ sudo mv htaccess .htaccess



Step 3 – Rewrite Images to S3 / CF

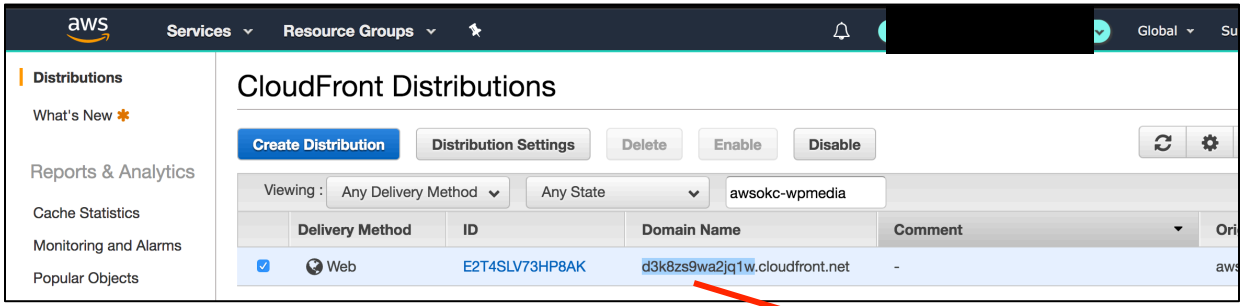
View rewrite rule in .htaccess file:
$ sudo nano .htaccess

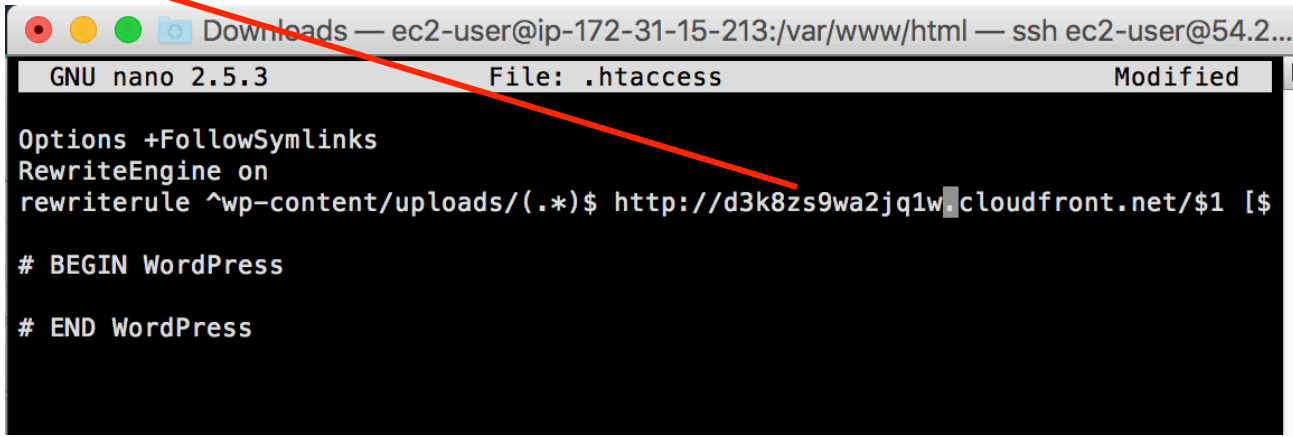




Step 3 – Rewrite Images to S3 / CF

Update with your Cloudfront distribution ID:





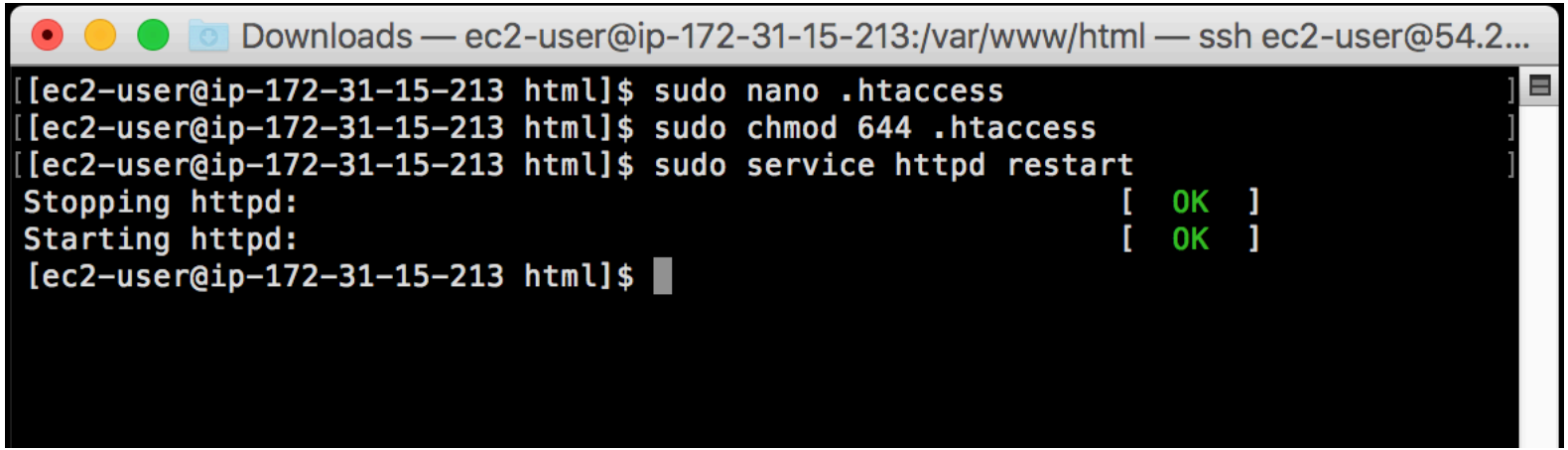
CTRL + X
Y
Enter

Step 3 – Rewrite Images to S3 / CF

Change Permissions, then Restart Apache Webserver

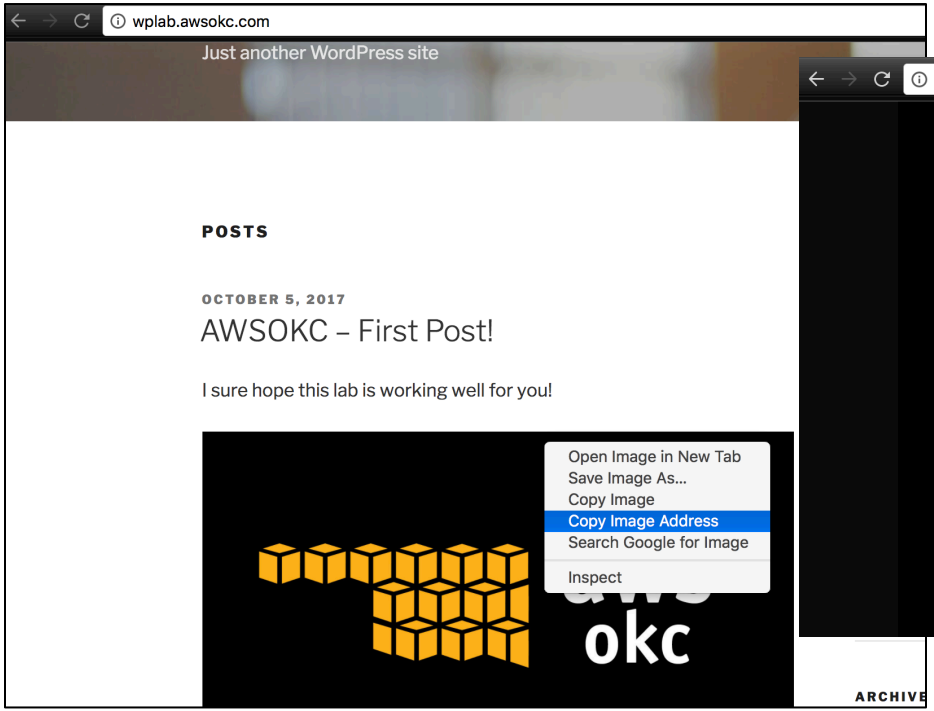
$ sudo chmod 644 .htaccess

$ sudo service httpd restart



Step 3 – Rewrite Images to S3 / CF

Check re-writing:

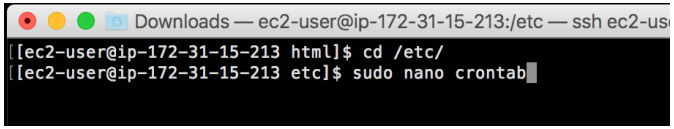


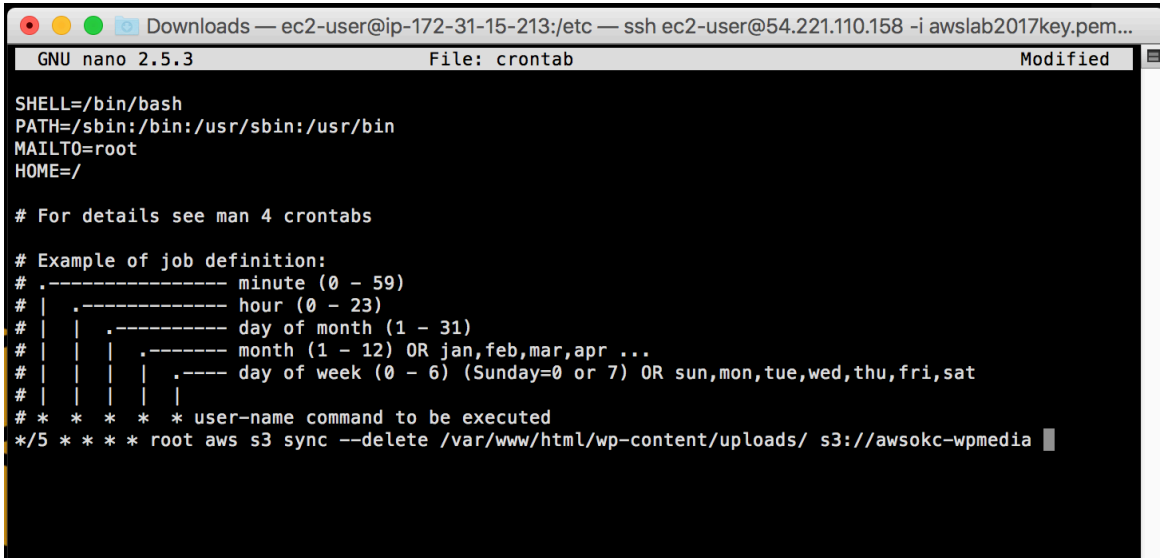
Step 3 – Rewrite Images to S3 / CF

Now automate the Sync command:

$ cd /etc/

$ sudo nano crontab



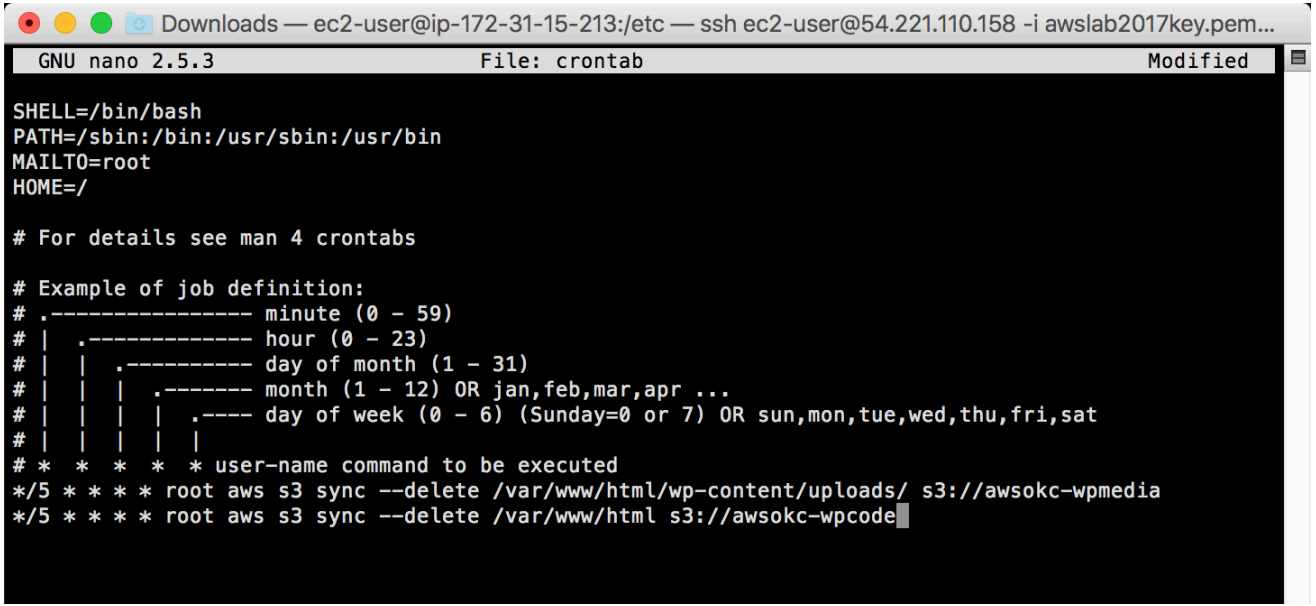


Step 3 – Rewrite Images to S3 / CF

Add Cron Jobs:

*/5 * * * * root aws s3 sync --delete /var/www/html/wp-content/uploads/ s3://awsokc-wpmedia

*/5 * * * * root aws s3 sync --delete /var/www/html s3://awsokc-wpcode



CTRL + X

Y

Enter

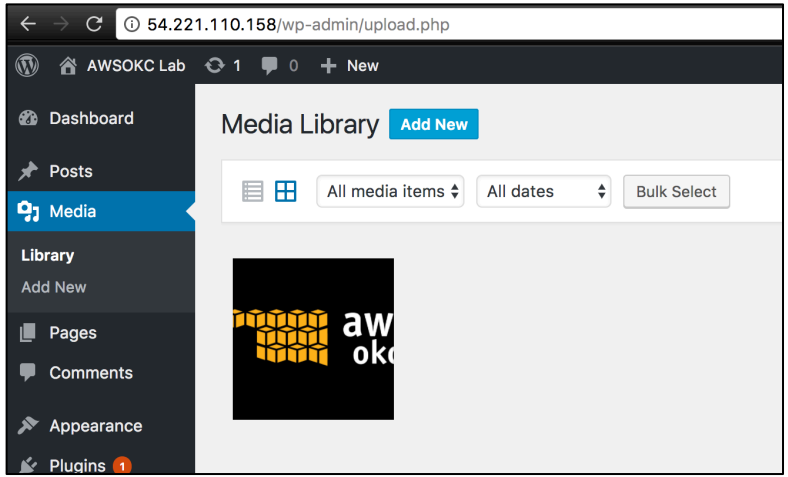
Step 3 – Rewrite Images to S3 / CF

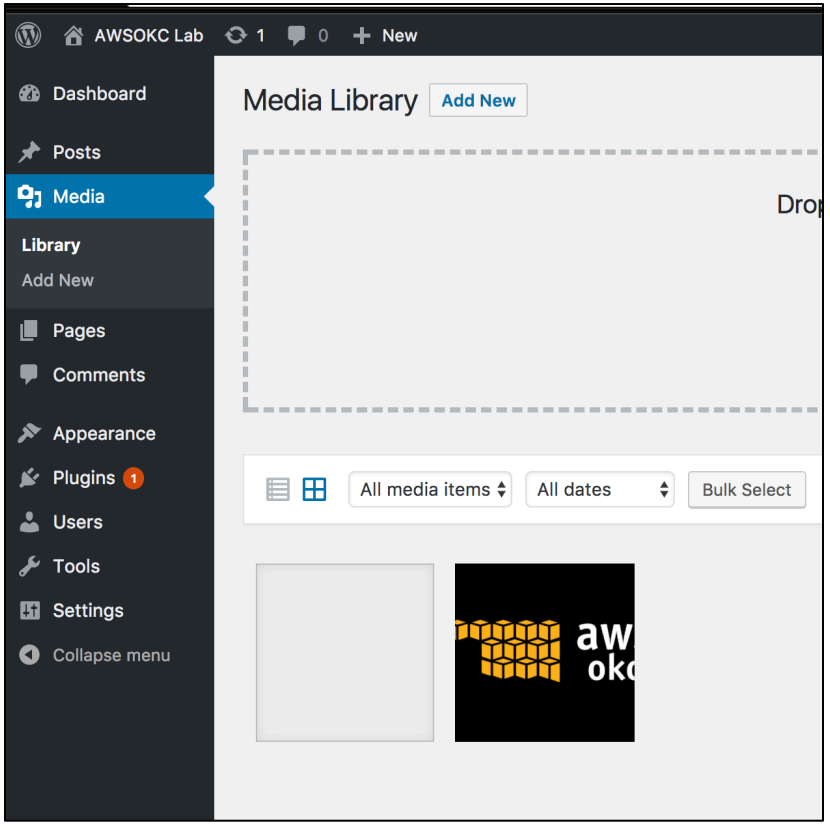
Restart Cron to force the jobs to rerun:

$ sudo service crond restart

Step 3 – Rewrite Images to S3 / CF

Create file and test sync:

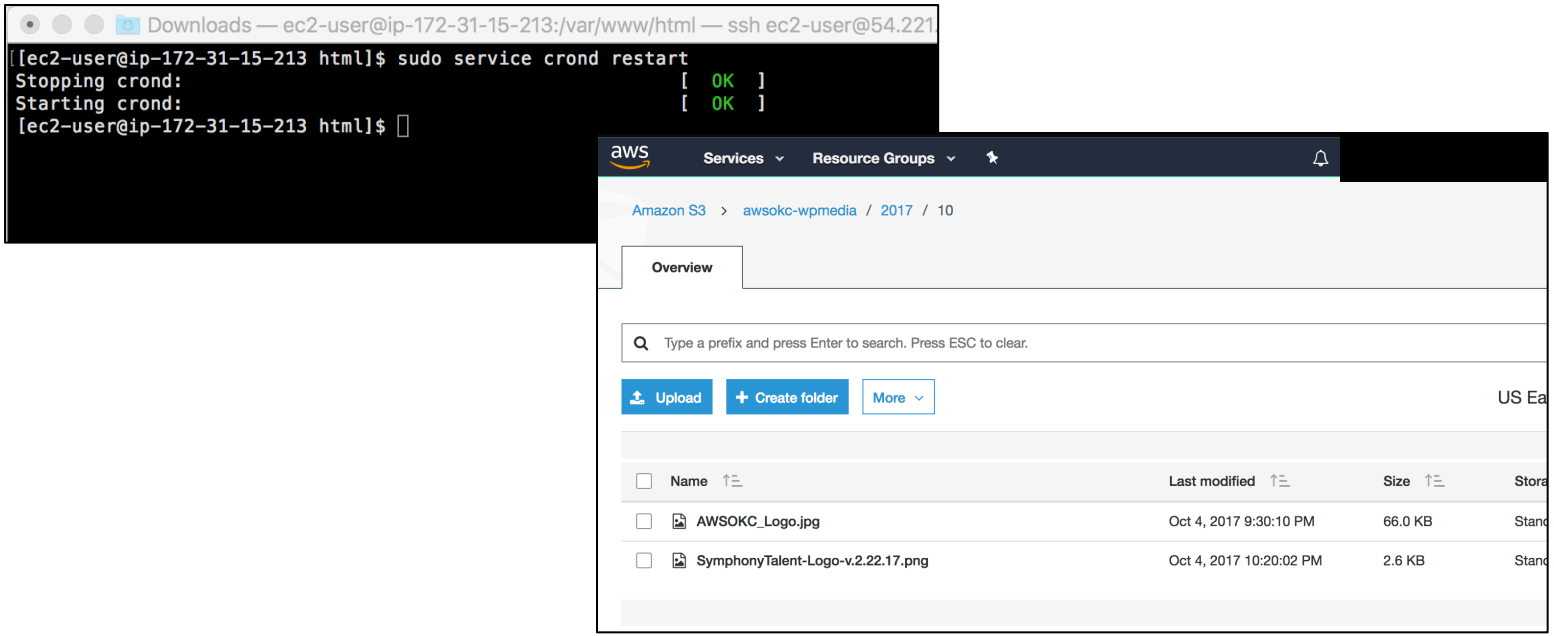




Step 3 – Rewrite Images to S3 / CF

Restart Cron:

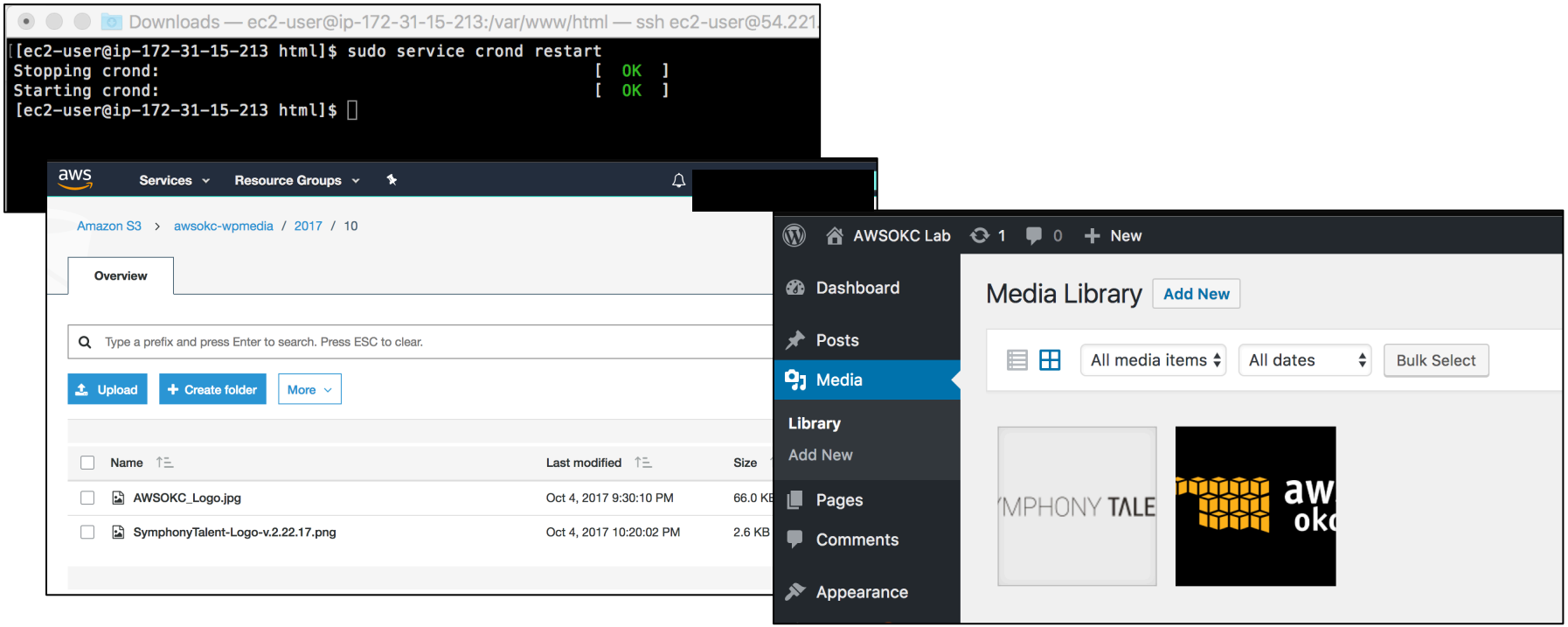
$ sudo service crond restart



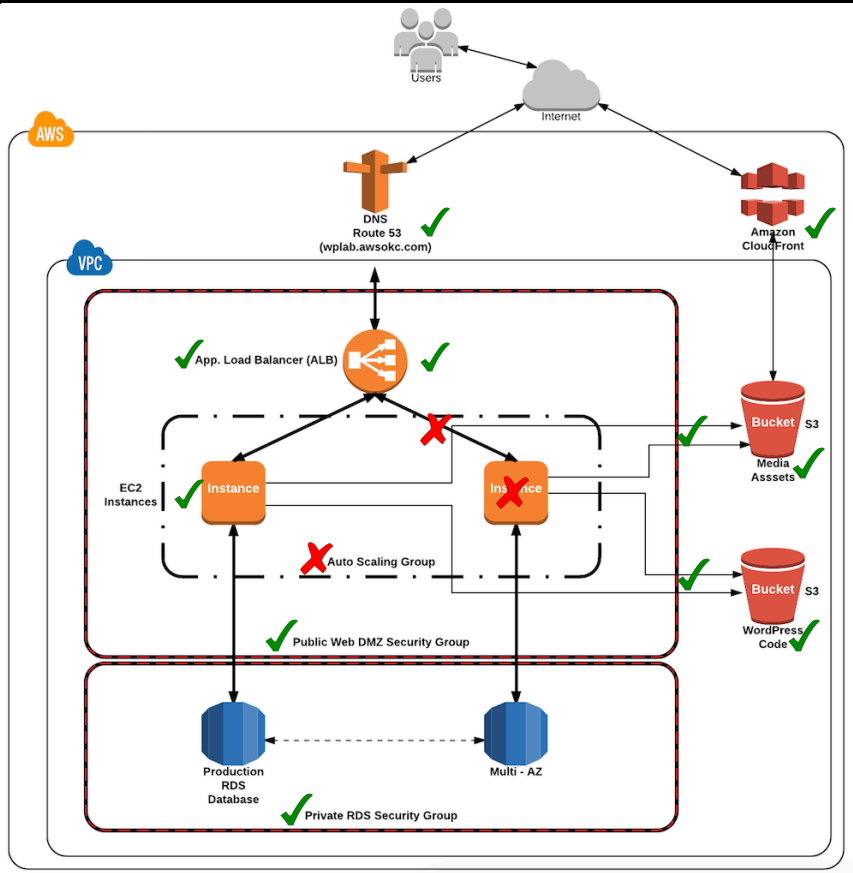
Step 3 – Rewrite Images to S3 / CF

Restart Cron:

$ sudo service crond restart

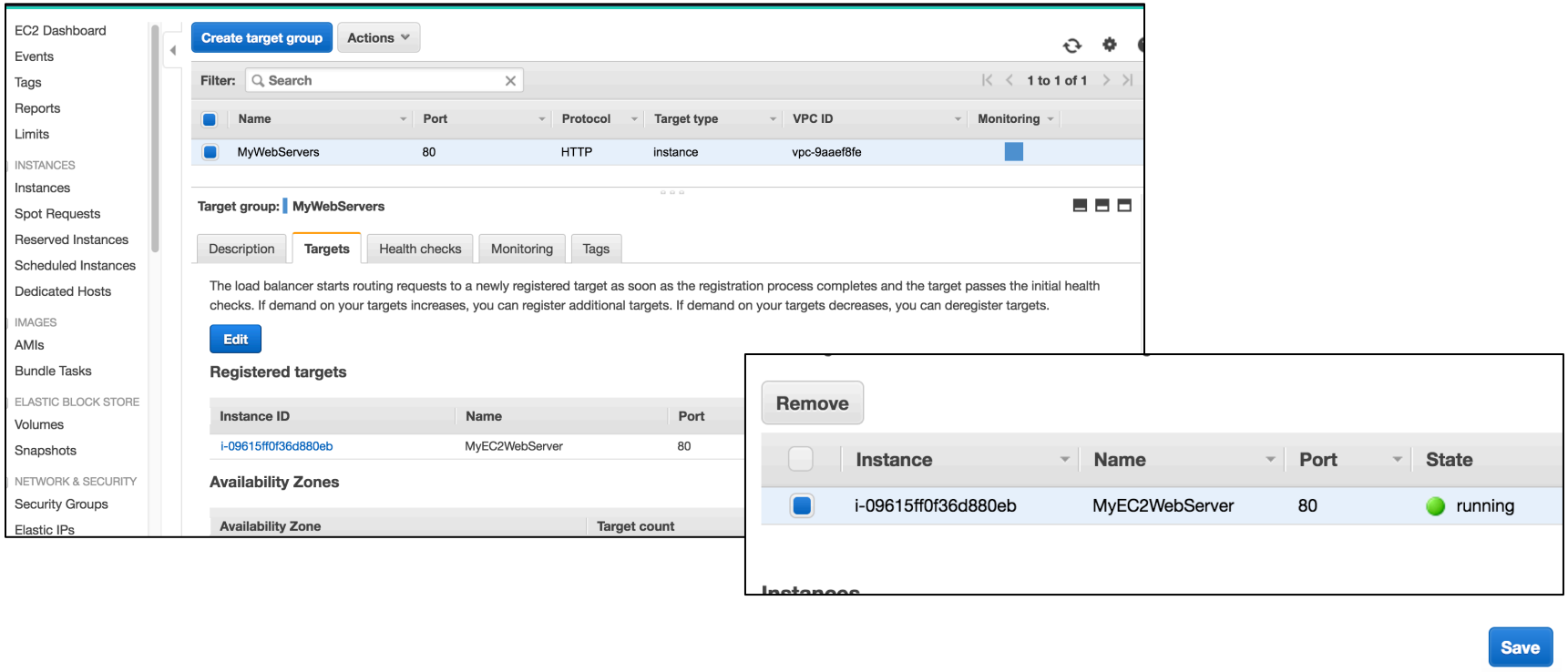


Status Check



Step 4 – Creating our AMI

First remove our EC2 instance from the ELB target group:



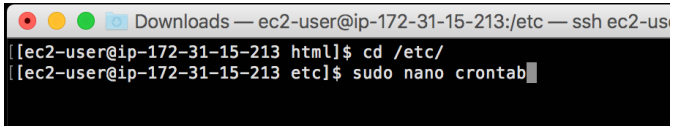
Step 4 – Creating our AMI

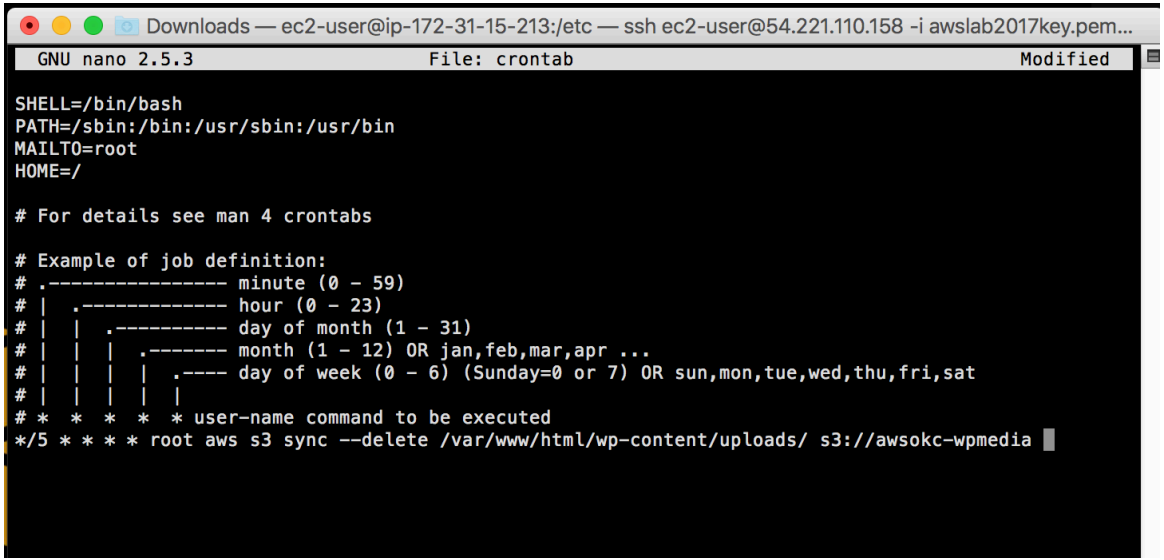
Discuss need for a Write node in the Architecture in addition to the Web Nodes / ASG

Save

Step 4 – Creating our AMI

Reconfigure the Cron jobs to Sync down to the EC2 instance instead of up.
(Make the Web Nodes read nodes – not write nodes)
$ cd /etc/
$ sudo nano crontab



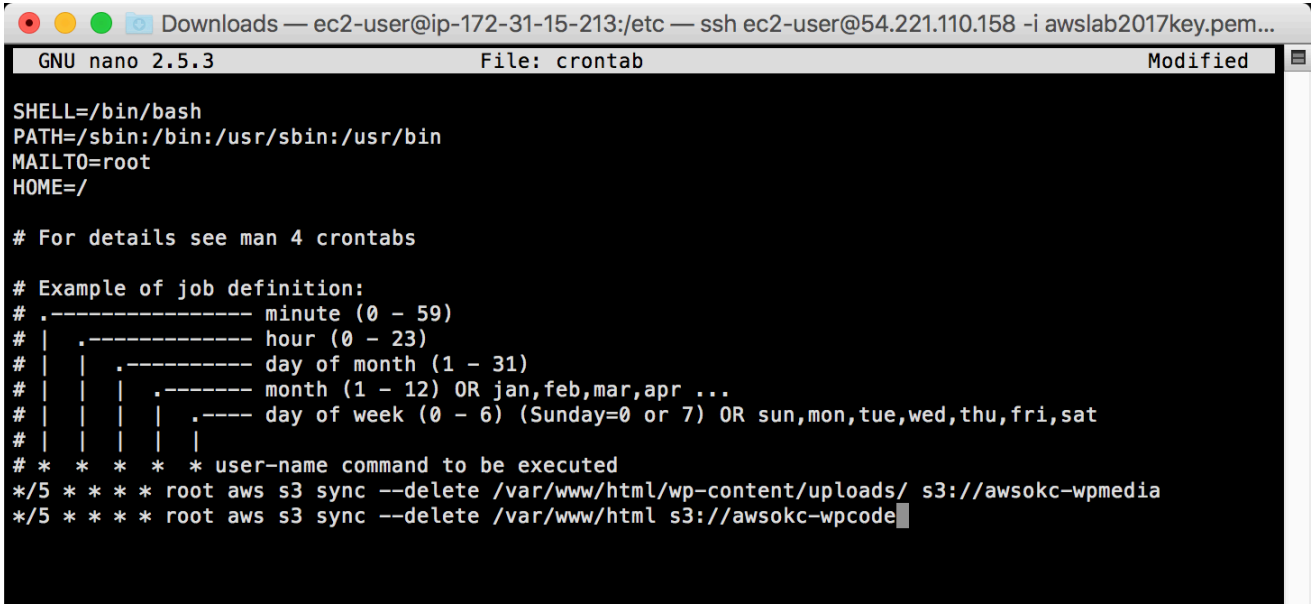


Step 4 – Creating our AMI

Add Cron Jobs:

*/5 * * * * root aws s3 sync --delete s3://awsokc-wpmedia /var/www/html/wp-content/uploads/

*/5 * * * * root aws s3 sync –delete s3://awsokc-wpcode /var/www/html



CTRL + X

Y

Enter

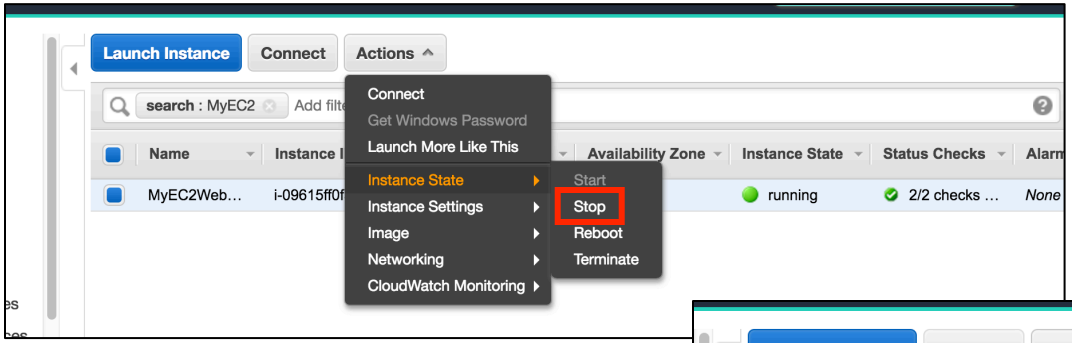
Step 4 – Creating our AMI

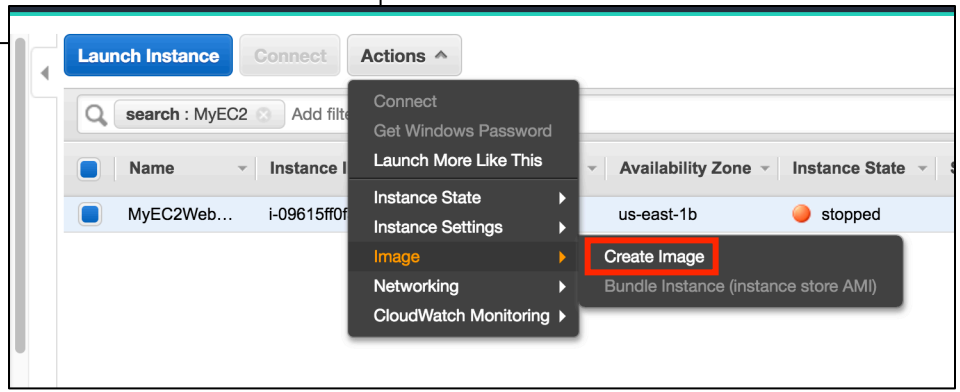
Restart Cron to force the jobs to rerun:

$ sudo service crond restart

Step 4 – Creating our AMI

Now stop the EC2 instance and create the AMI:

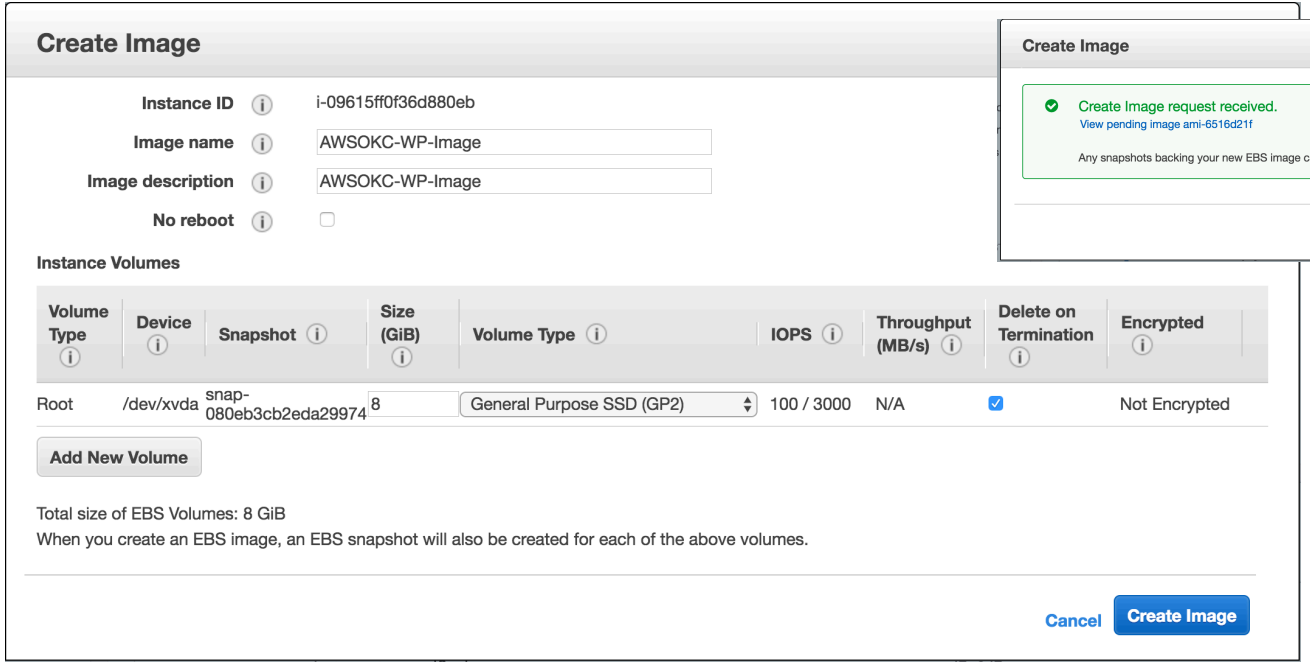


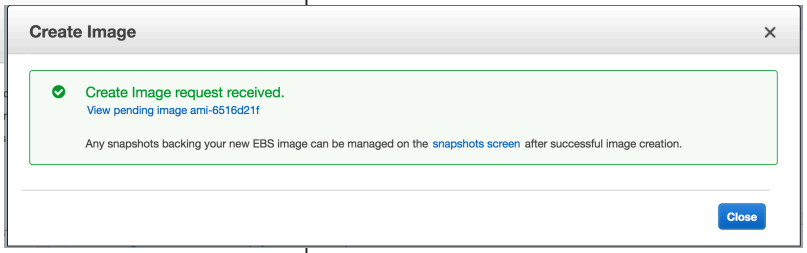


Step 4 – Creating our AMI

aws okc

Give Image Name

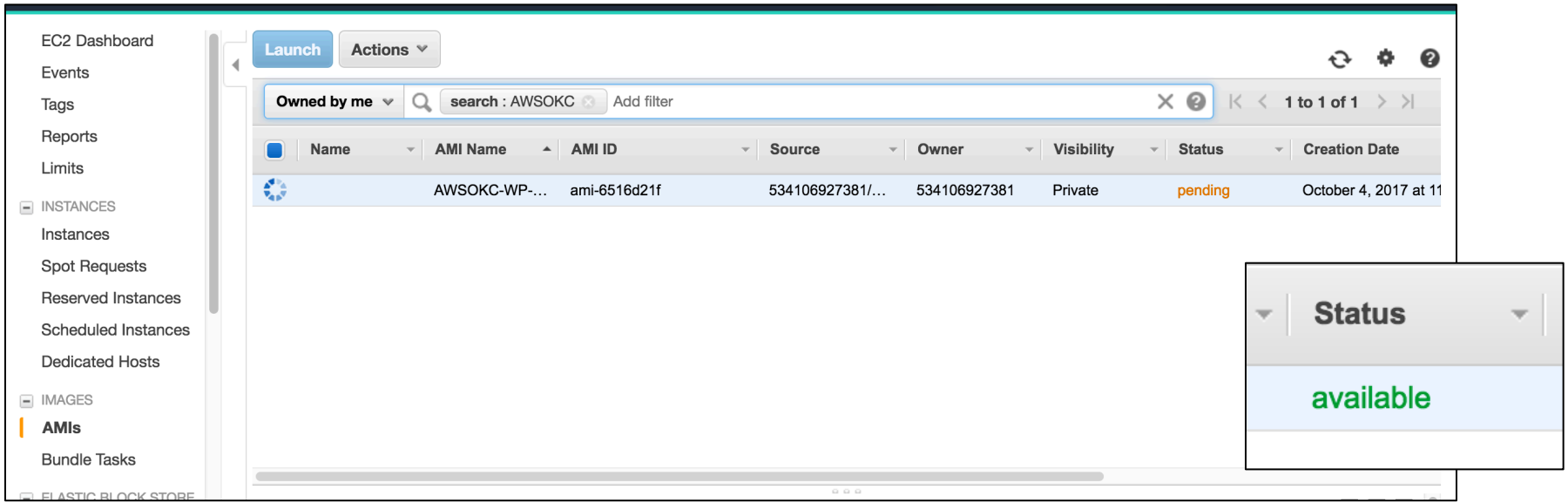




Step 4 – Creating our AMI

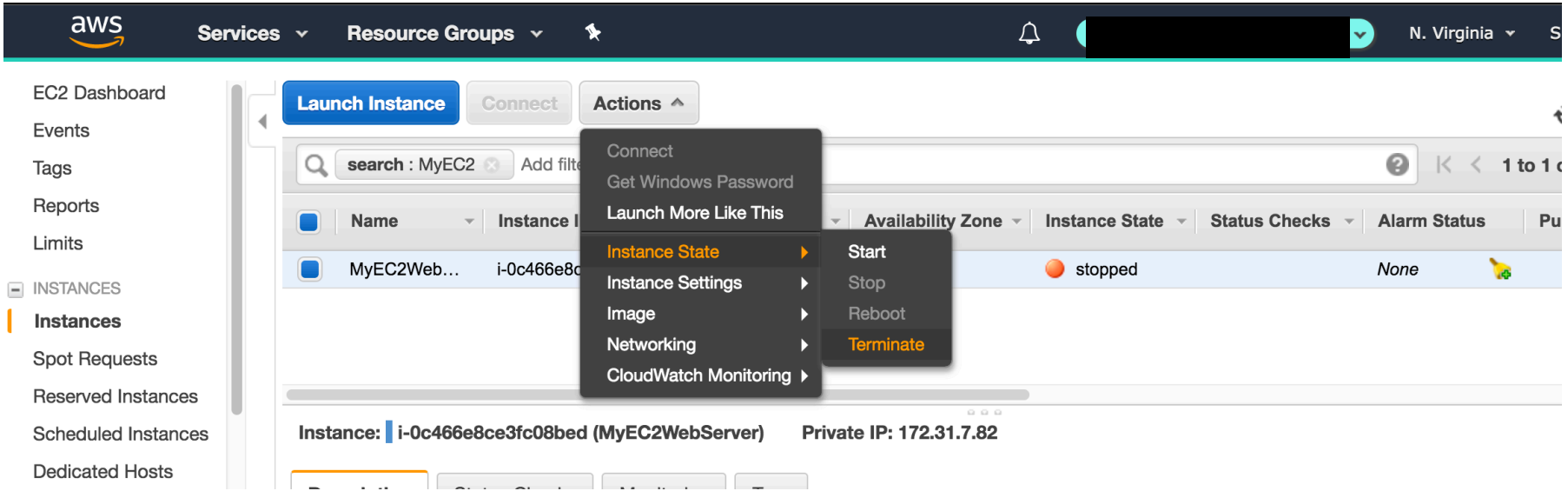
aws okc

AMI Creation:



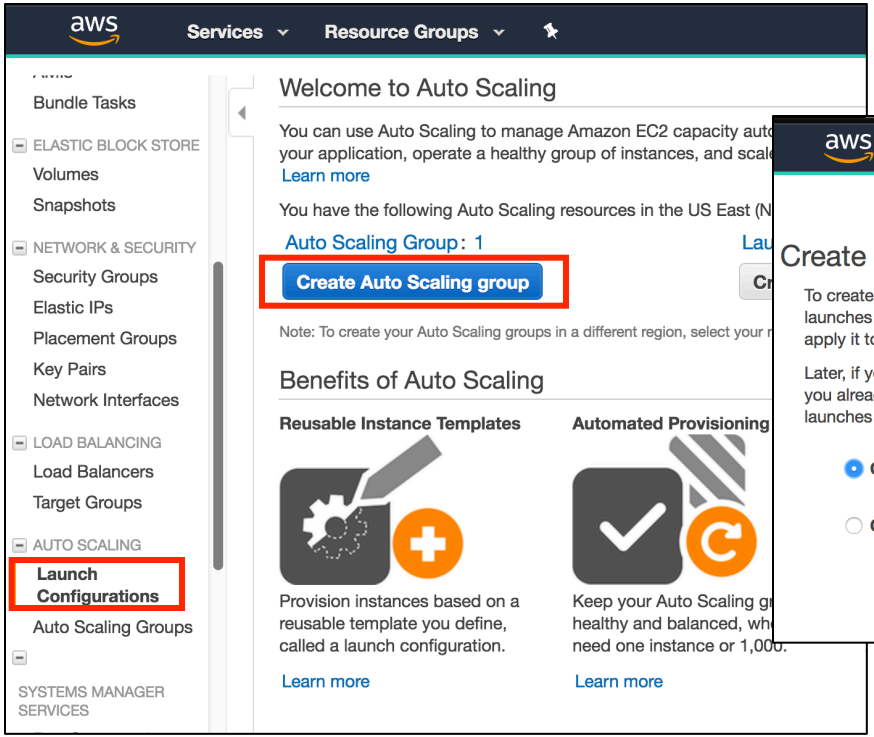
Step 4 – Creating our AMI

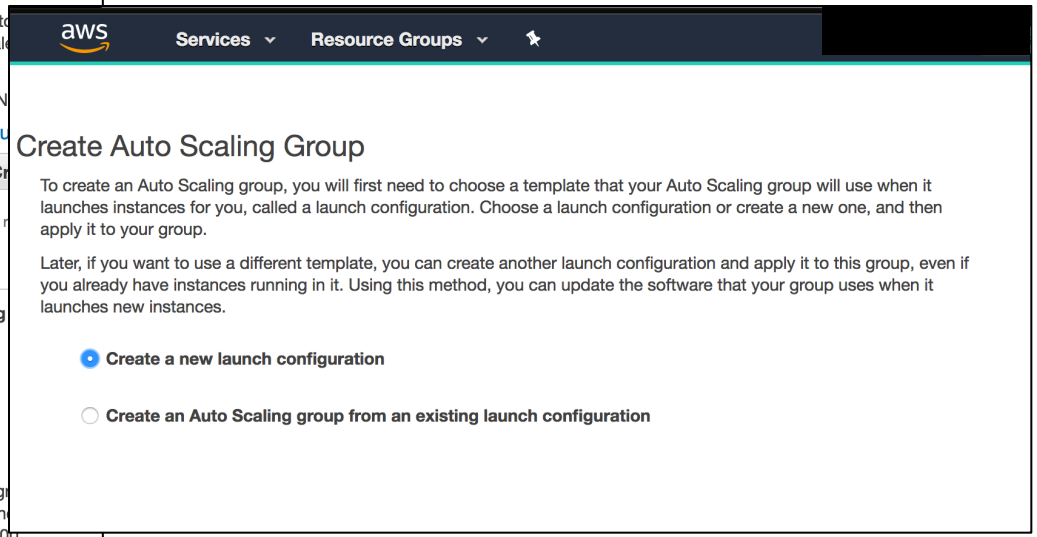
Now that we have our AMI – you can terminate the stopped instance.



Step 5 – Creating our ASG / LC

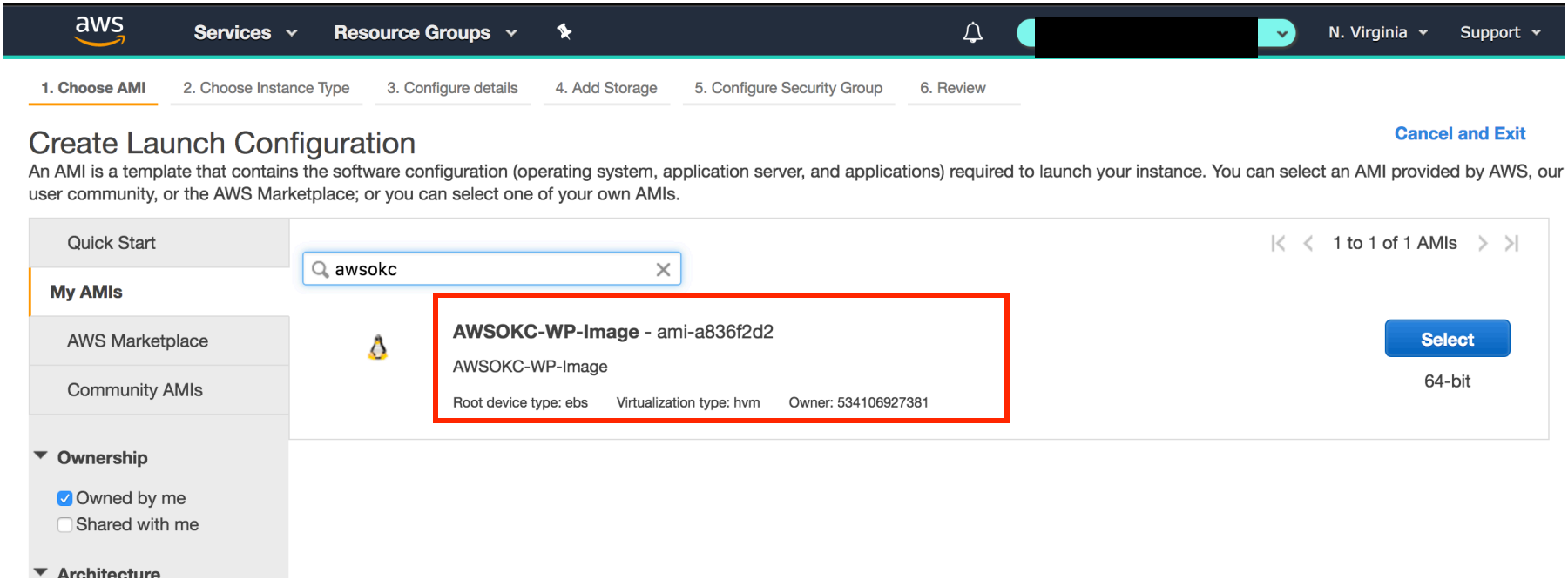
Create Auto-Scaling Group / Launch Config:





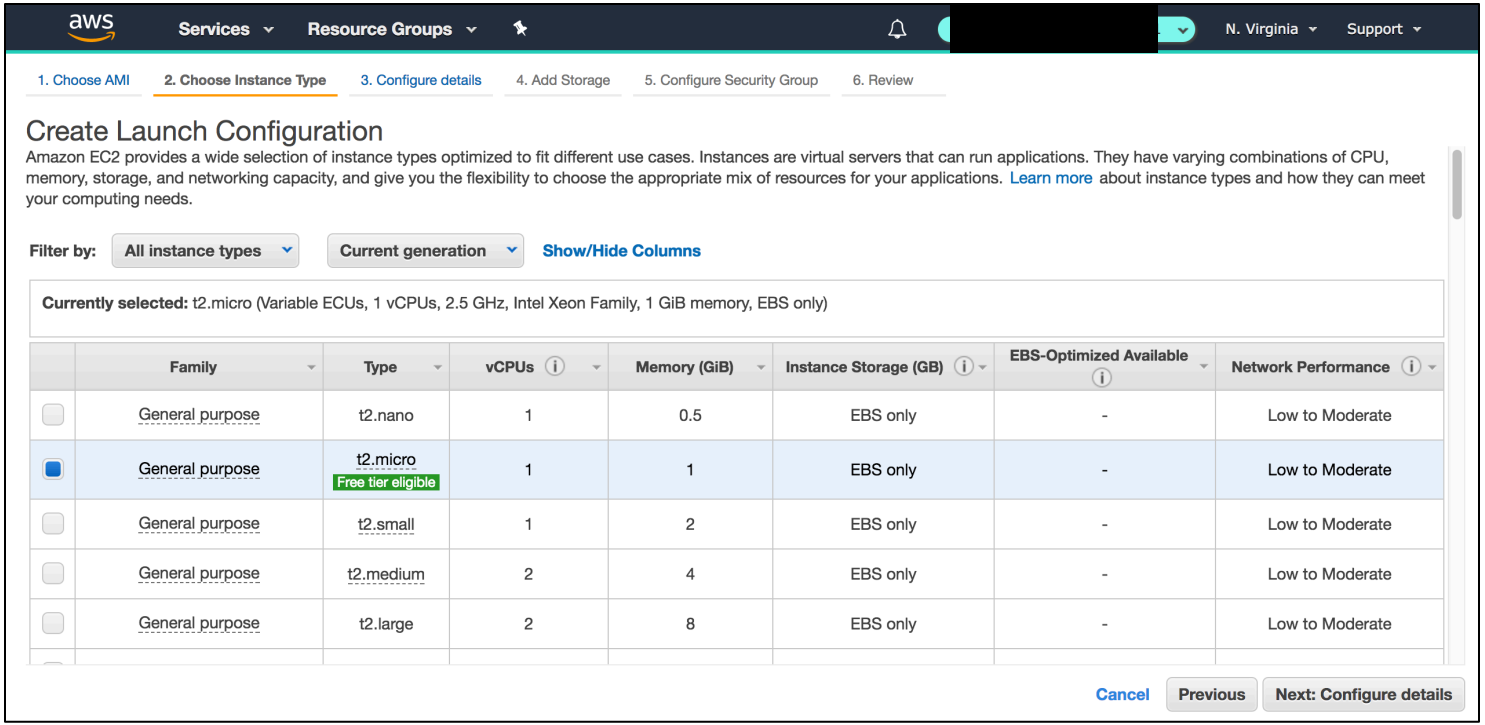
Step 5 – Creating our ASG / LC

Select your new AMI:



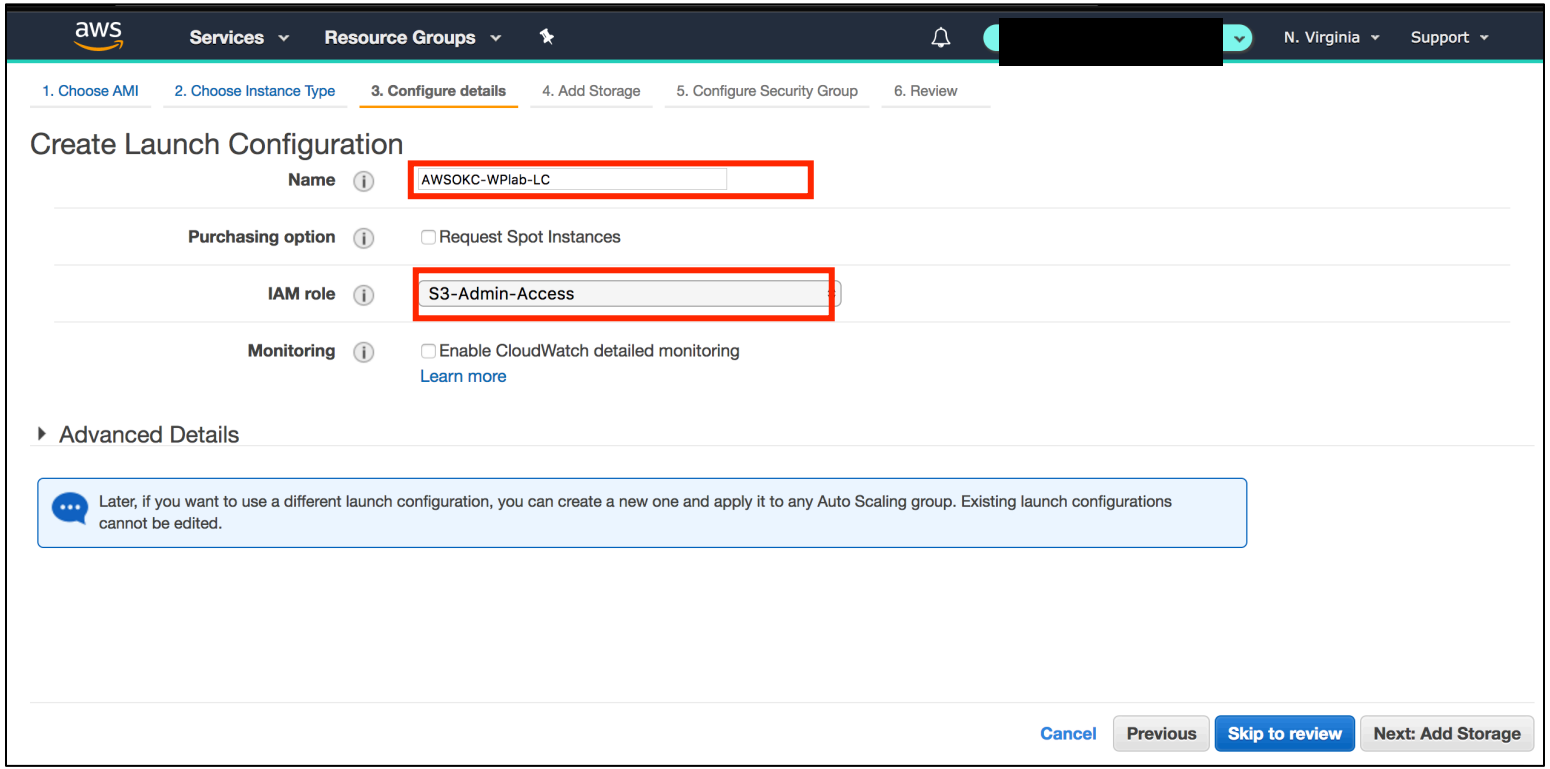
Step 5 – Creating our ASG / LC

Keep size as t2.micro:



Step 5 – Creating our ASG / LC

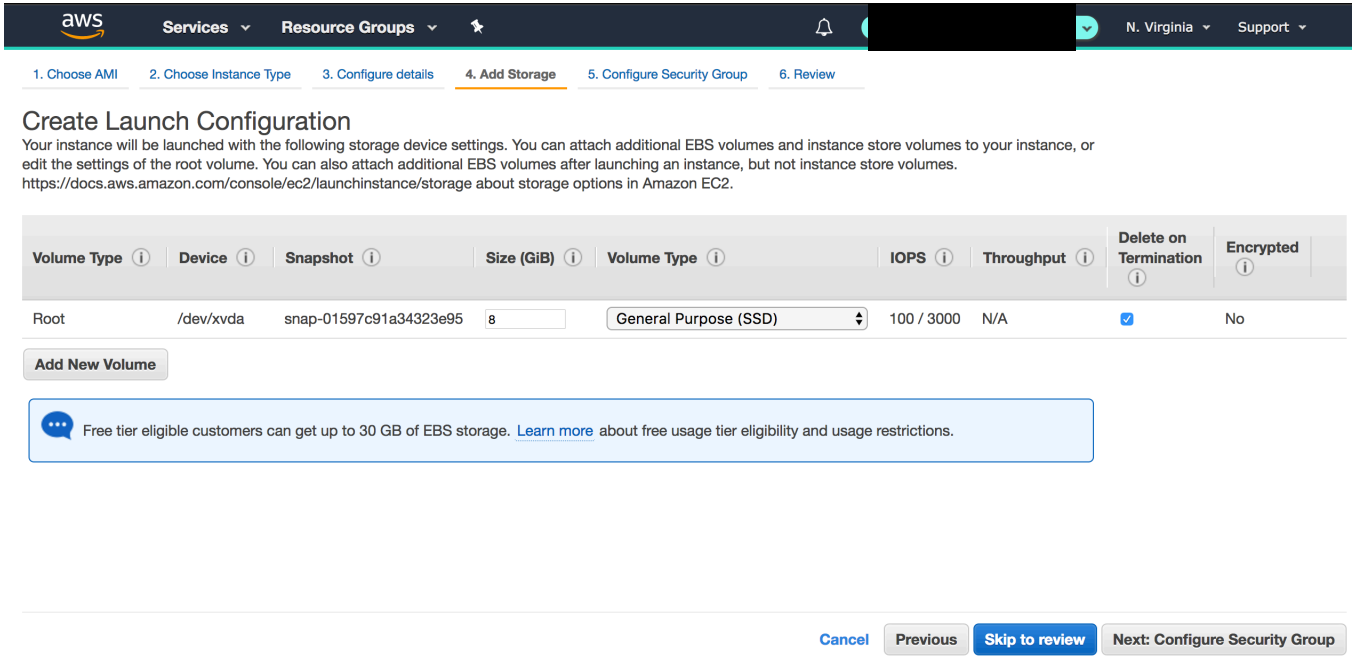
Give it a name and IAM role:



Step 5 – Creating our ASG / LC

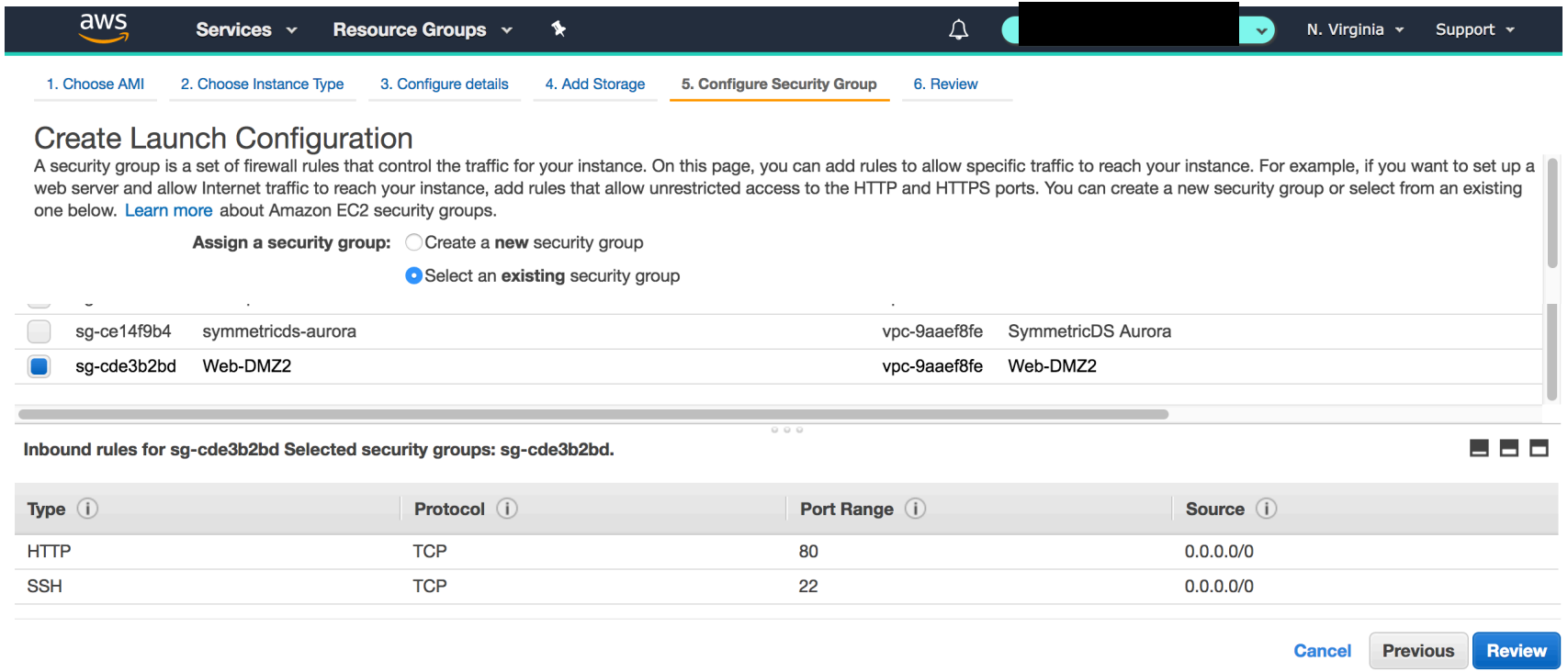
aws okc

Use default storage size:



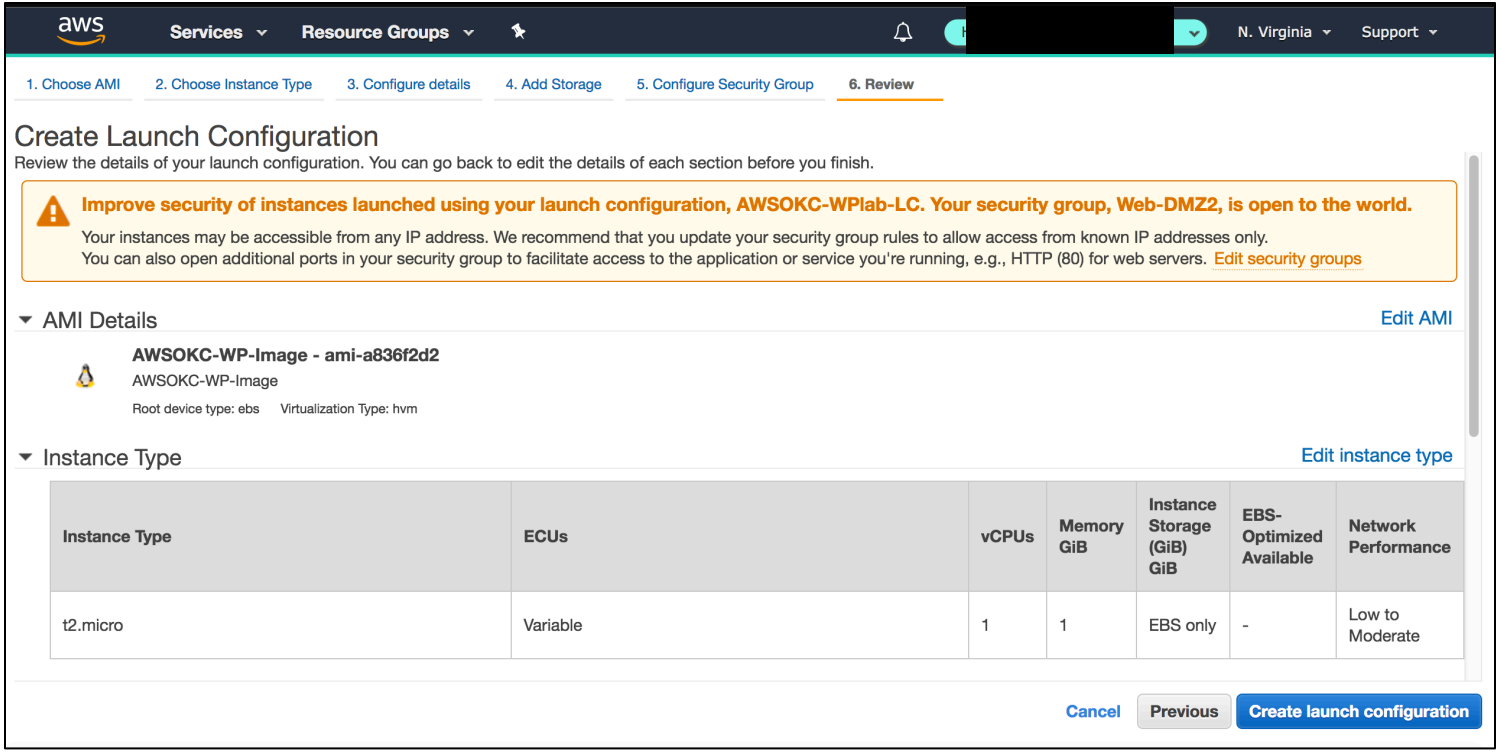
Step 5 – Creating our ASG / LC

Use the Web-DMZ security group:



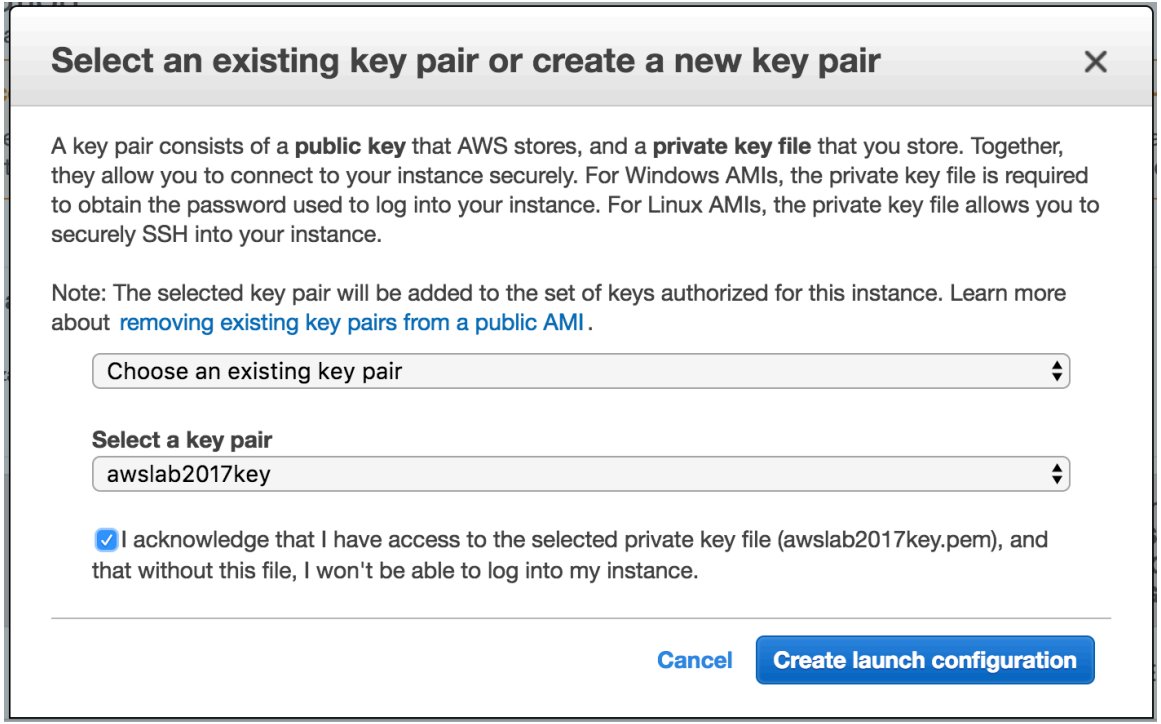
Step 5 – Creating our ASG / LC

Create:



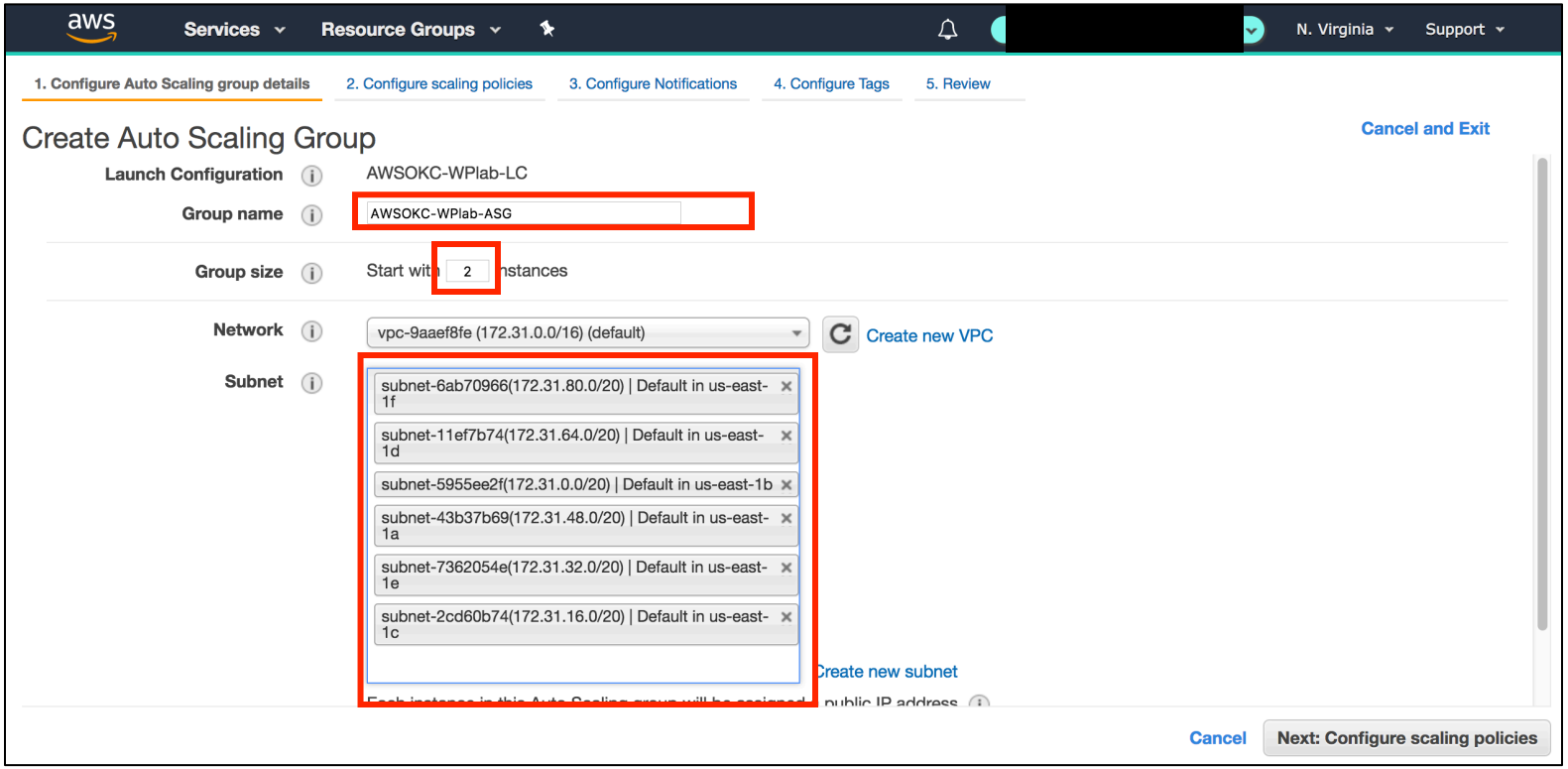
Step 5 – Creating our ASG / LC

Select your key pair:



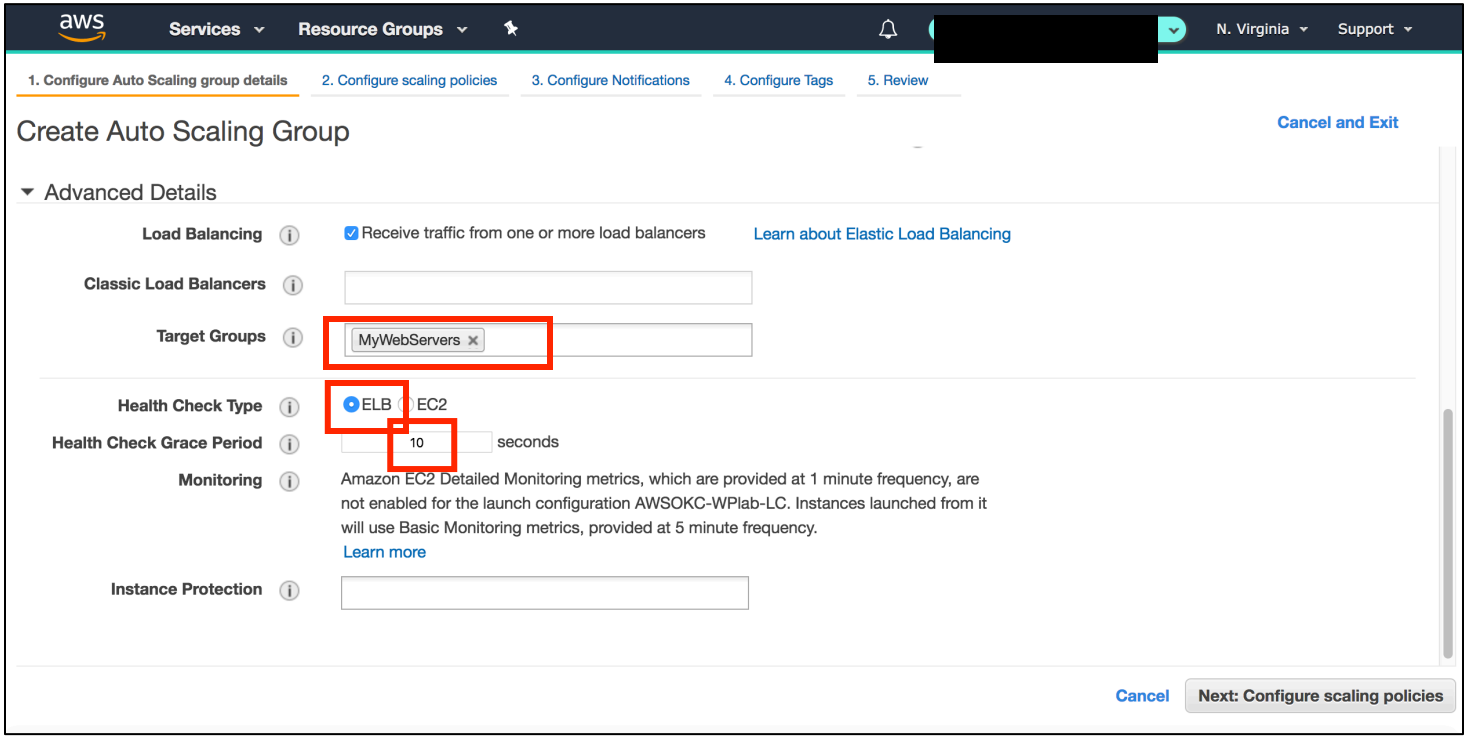
Step 5 – Creating our ASG / LC

Now we'll create the Auto Scaling Group:



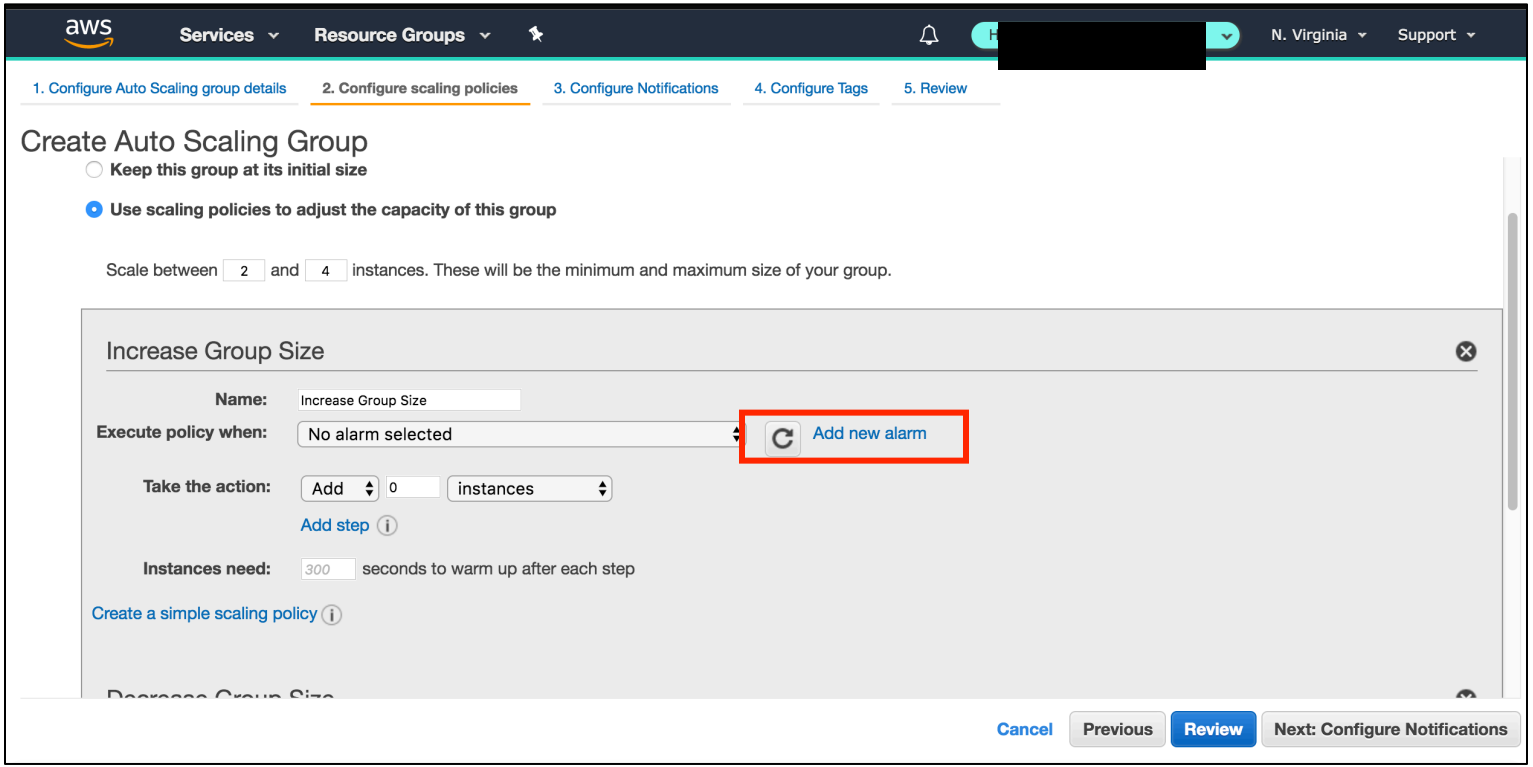
Step 5 – Creating our ASG / LC

Configure advanced options:



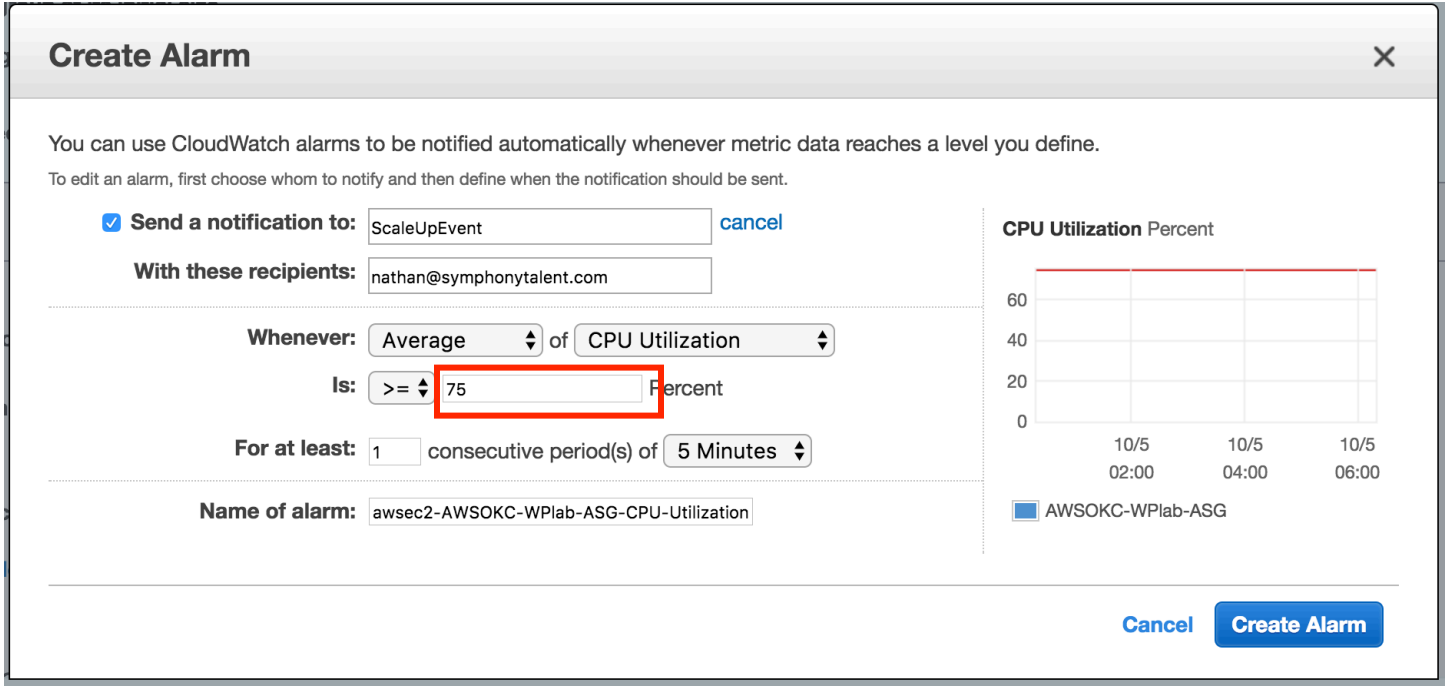
Step 5 – Creating our ASG / LC

Set scaling policies:



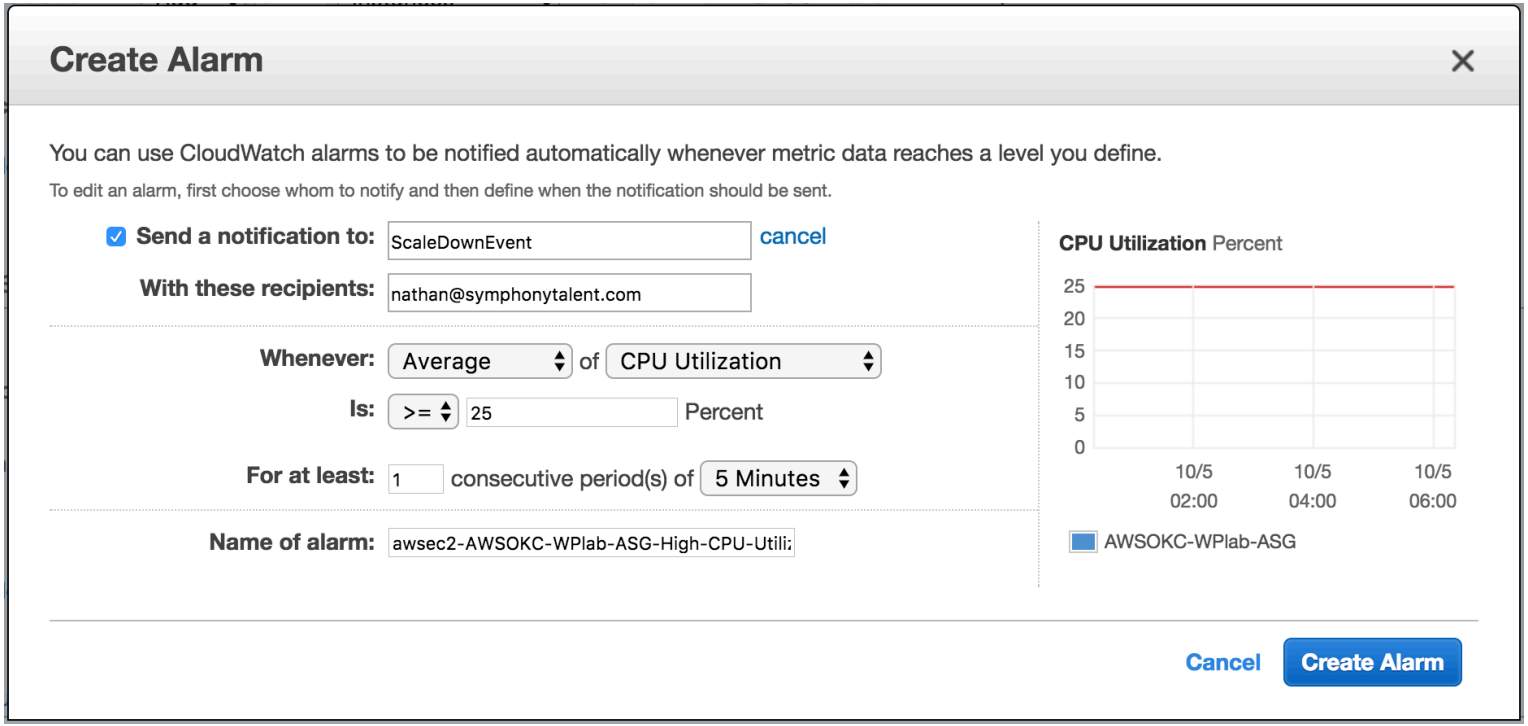
Step 5 – Creating our ASG / LC

Set scaling policies:



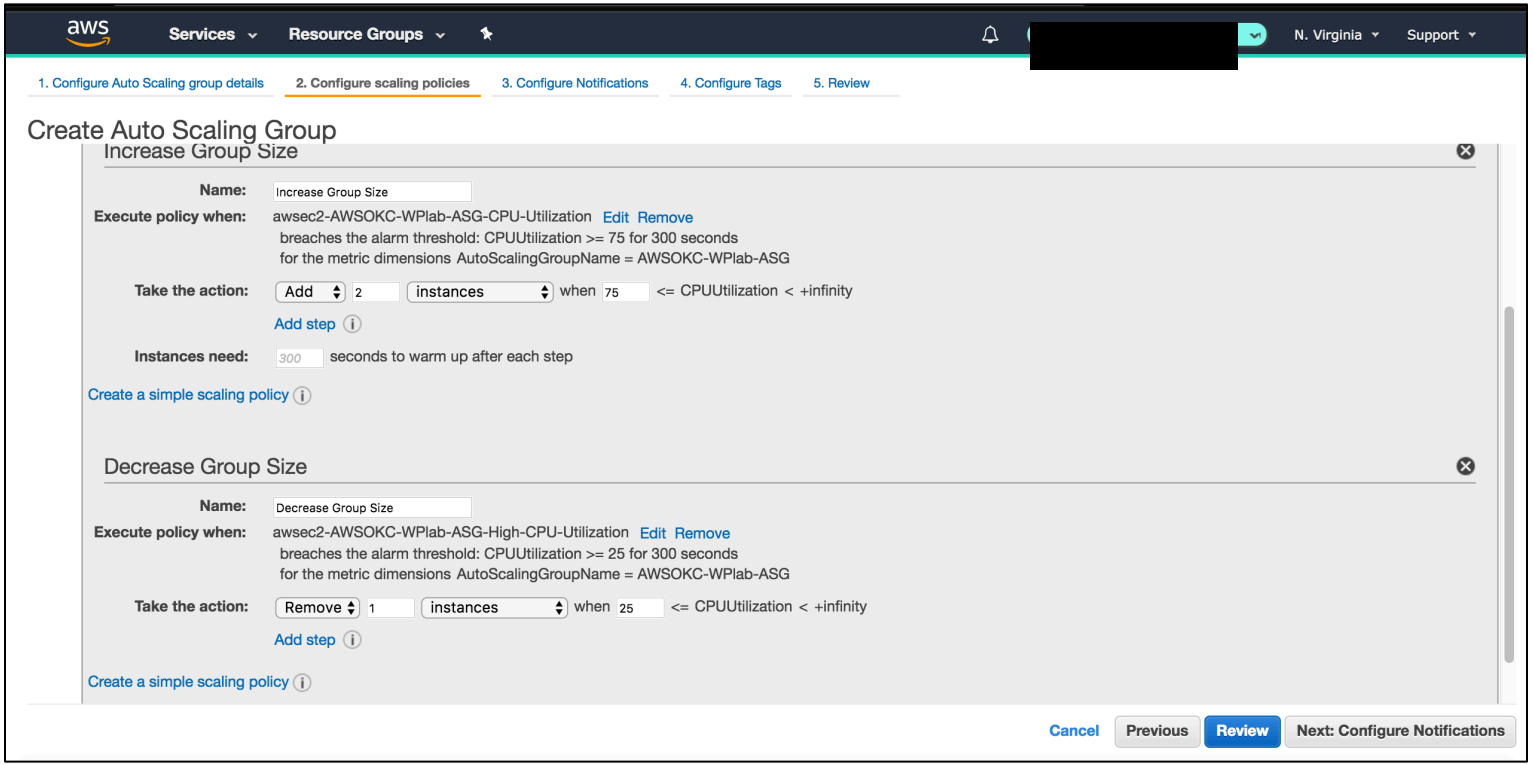
Step 5 – Creating our ASG / LC

Set scaling policies:



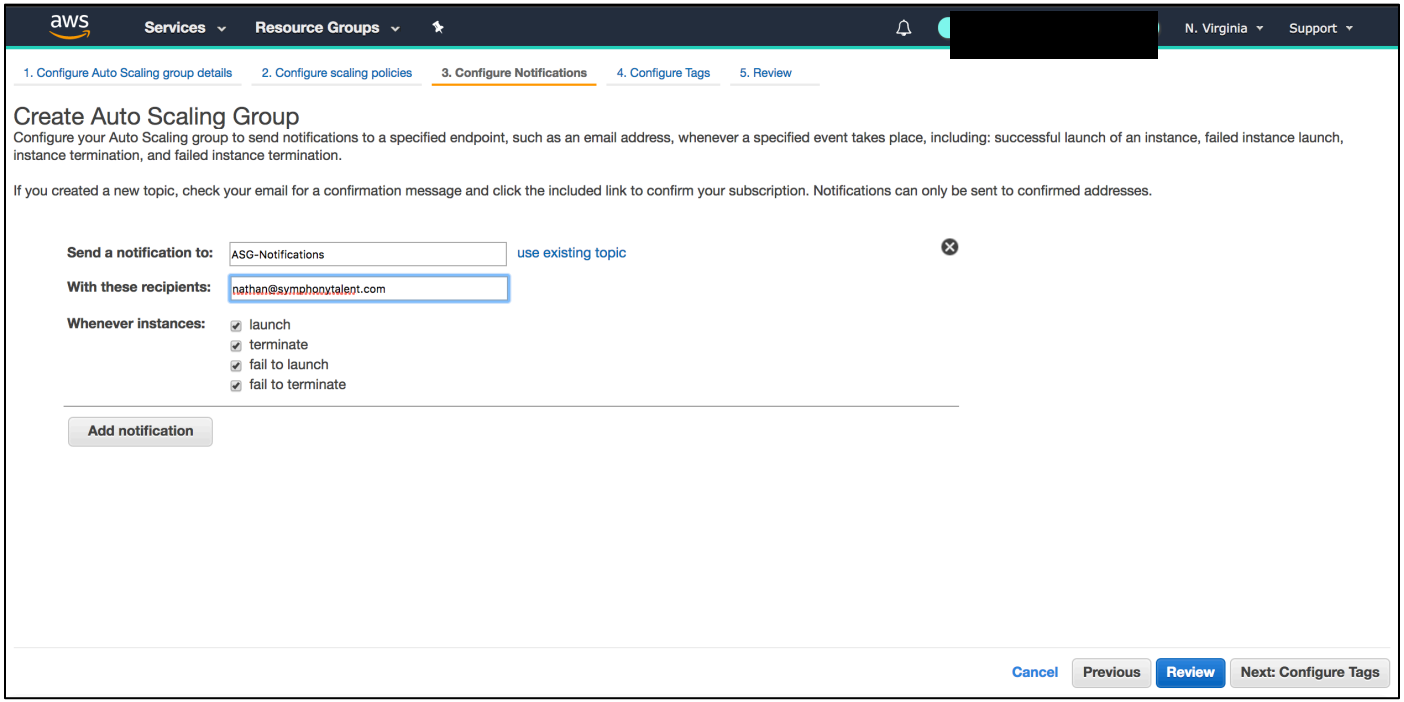
Step 5 – Creating our ASG / LC

Set scaling policies:



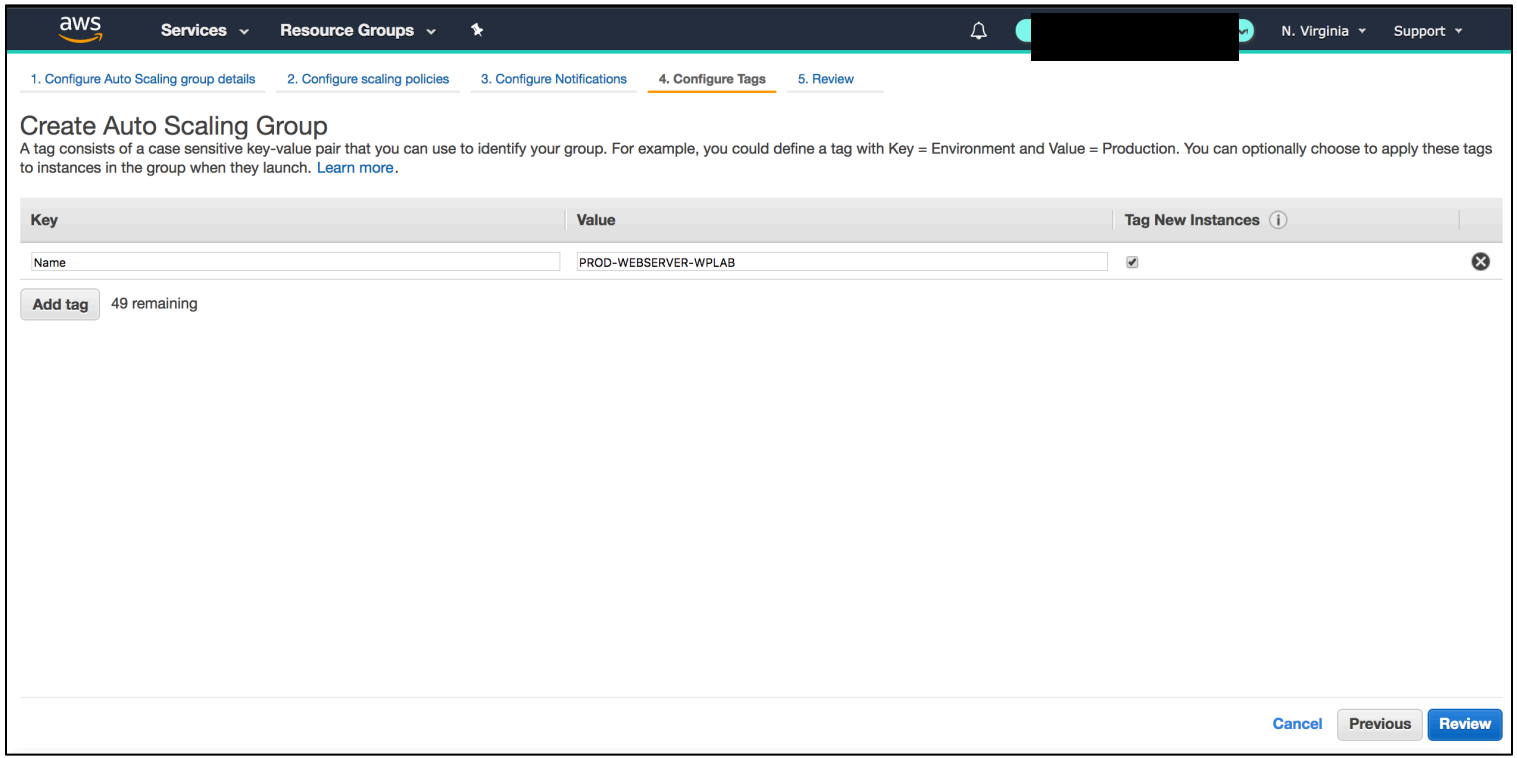
Step 5 – Creating our ASG / LC

Set scaling policies:



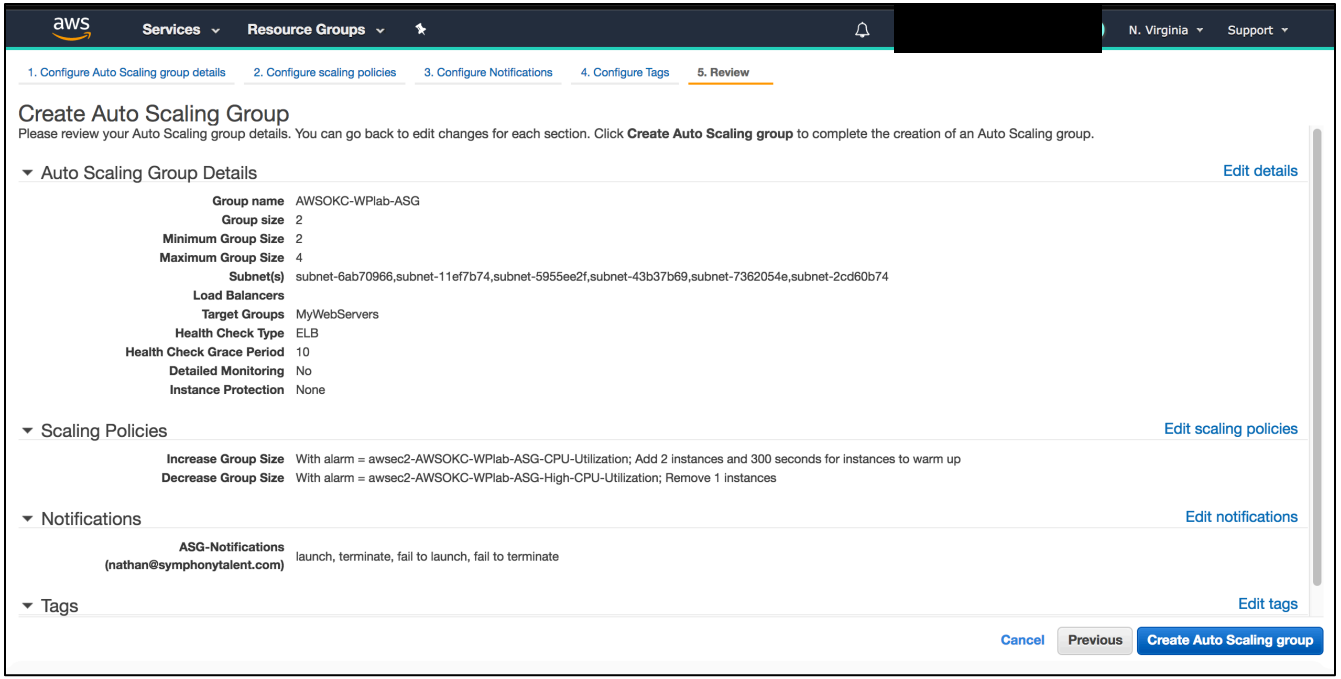
Step 5 – Creating our ASG / LC

Set scaling policies:



Step 5 – Creating our ASG / LC

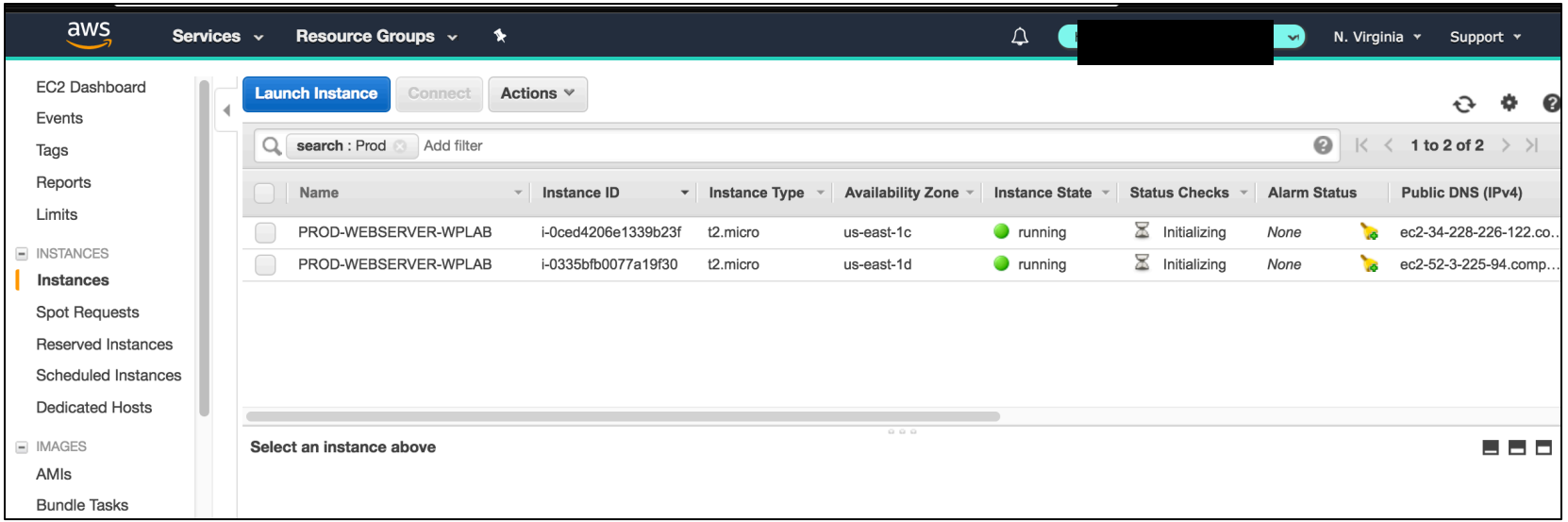
Set scaling policies:



Step 5 – Creating our ASG / LC

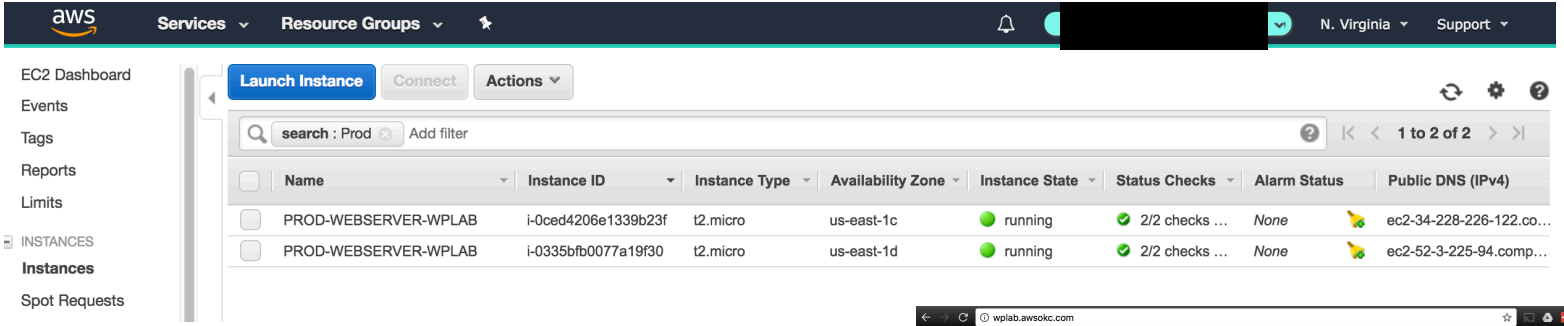
aws okc

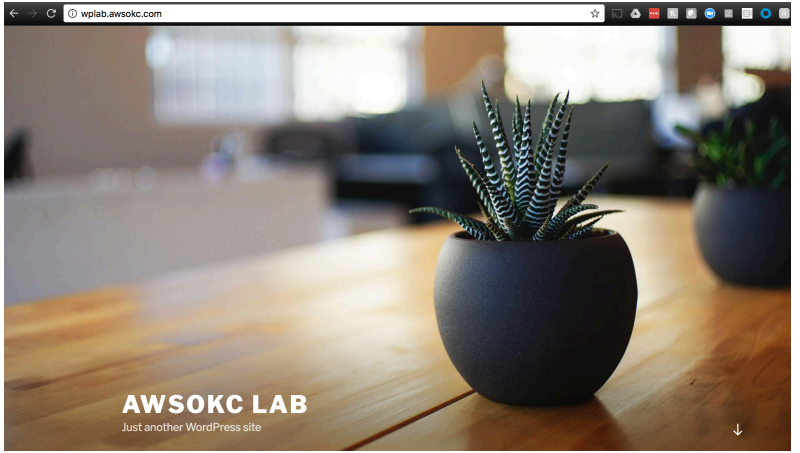
See new instances coming online:



Step 5 – Creating our ASG / LC

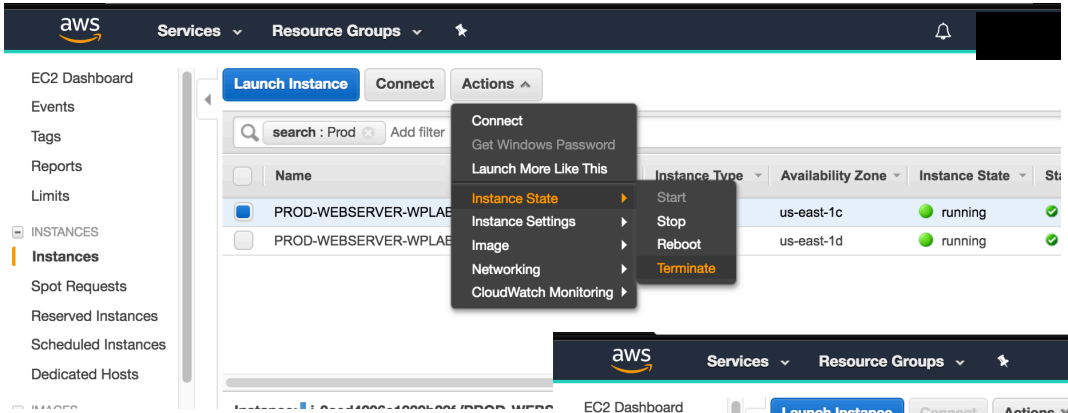
Once instances are healthy – verify site availability:

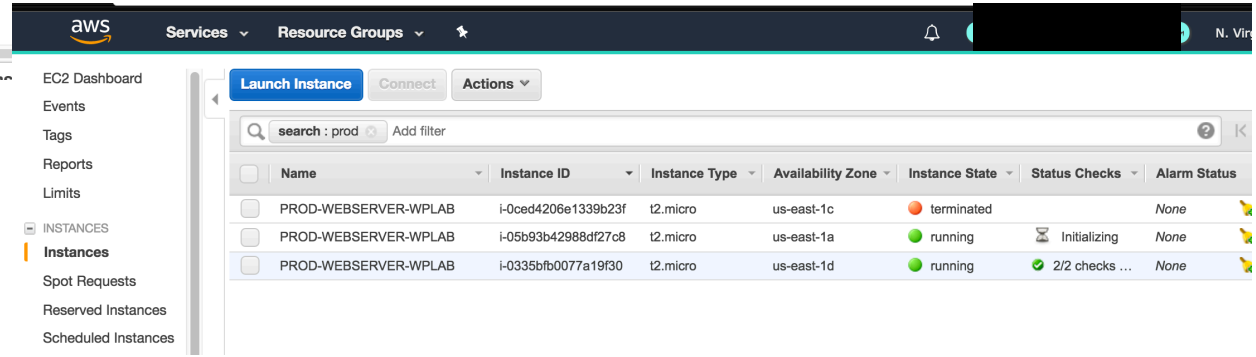




Step 6 – Test Availability / Load

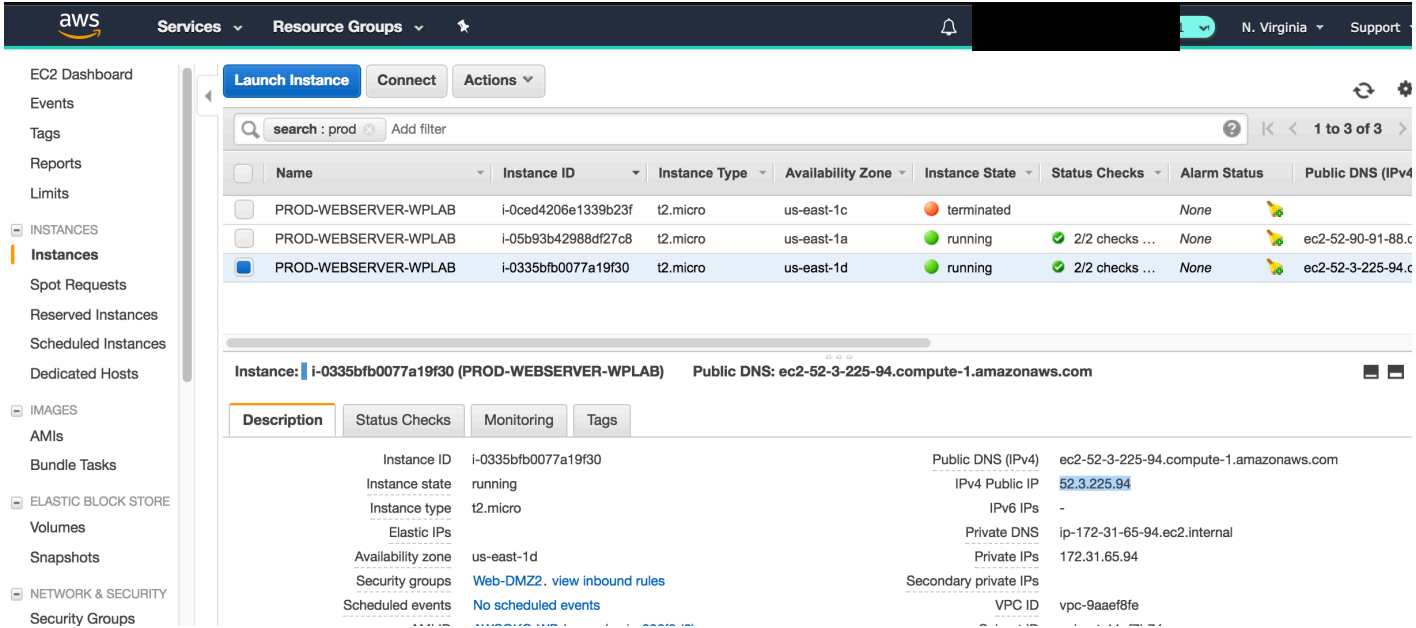
Test 1 – Terminate Instances:





Step 6 – Test Availability / Load

Test 2 – Testing High Load:

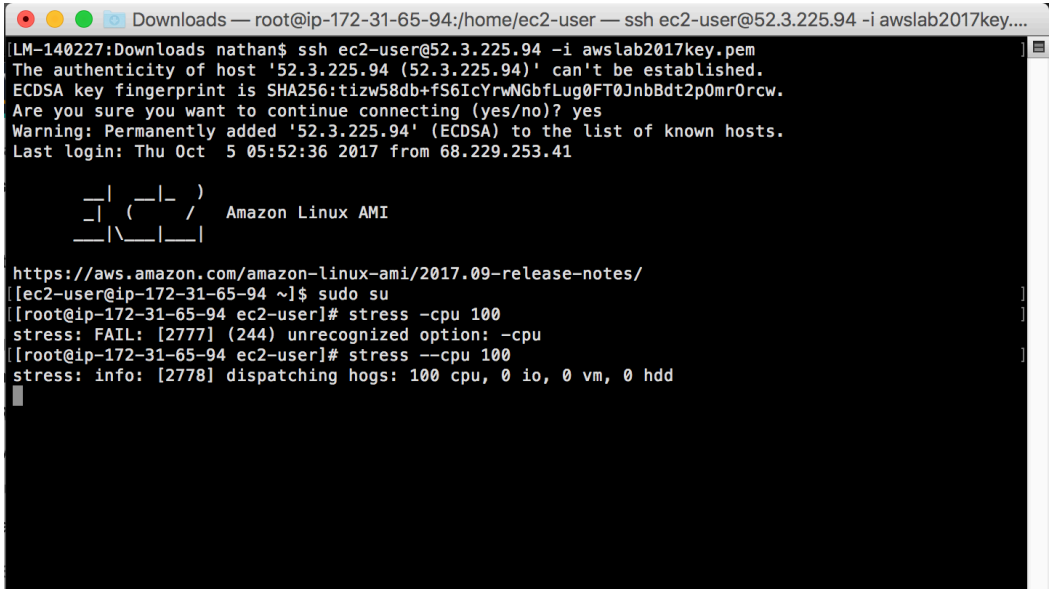


Step 6 – Test Availability / Load

SSH and run Stress Testing Tool:

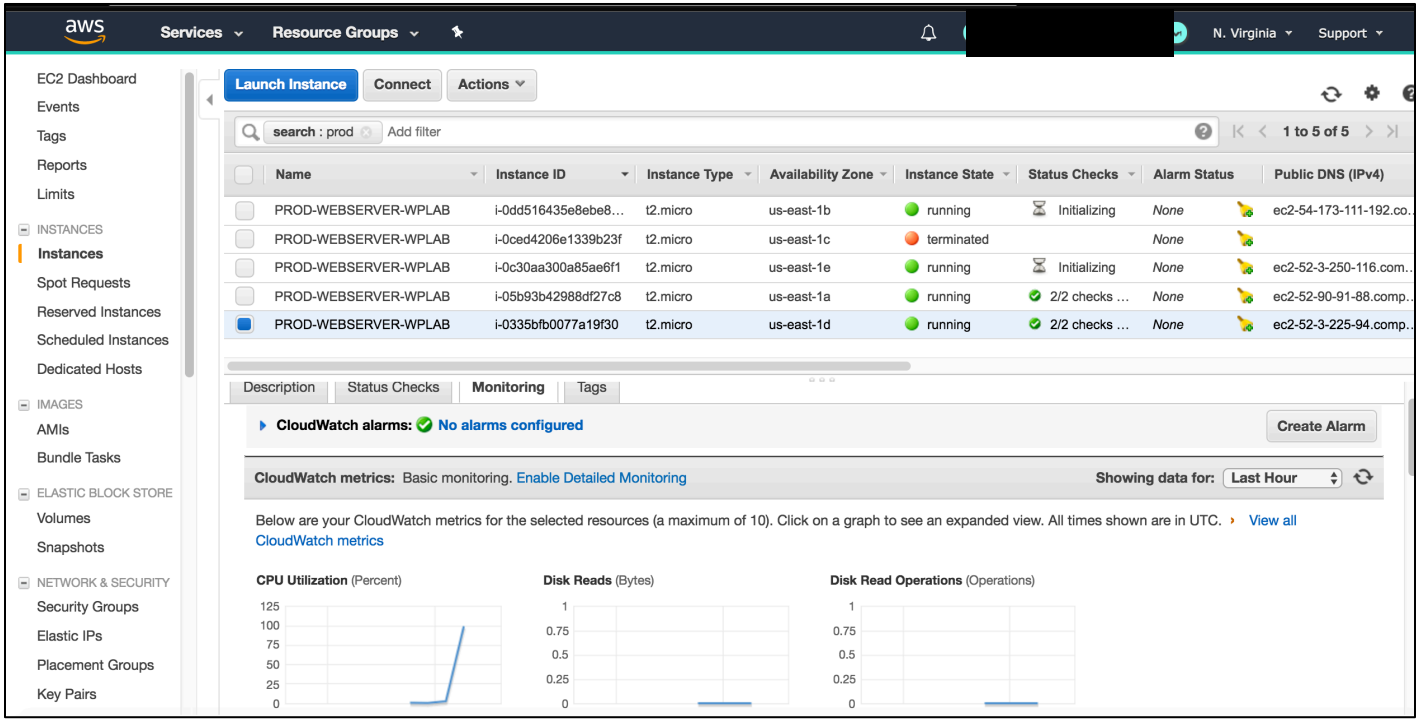
$ sudo su

$ stress --cpu 100

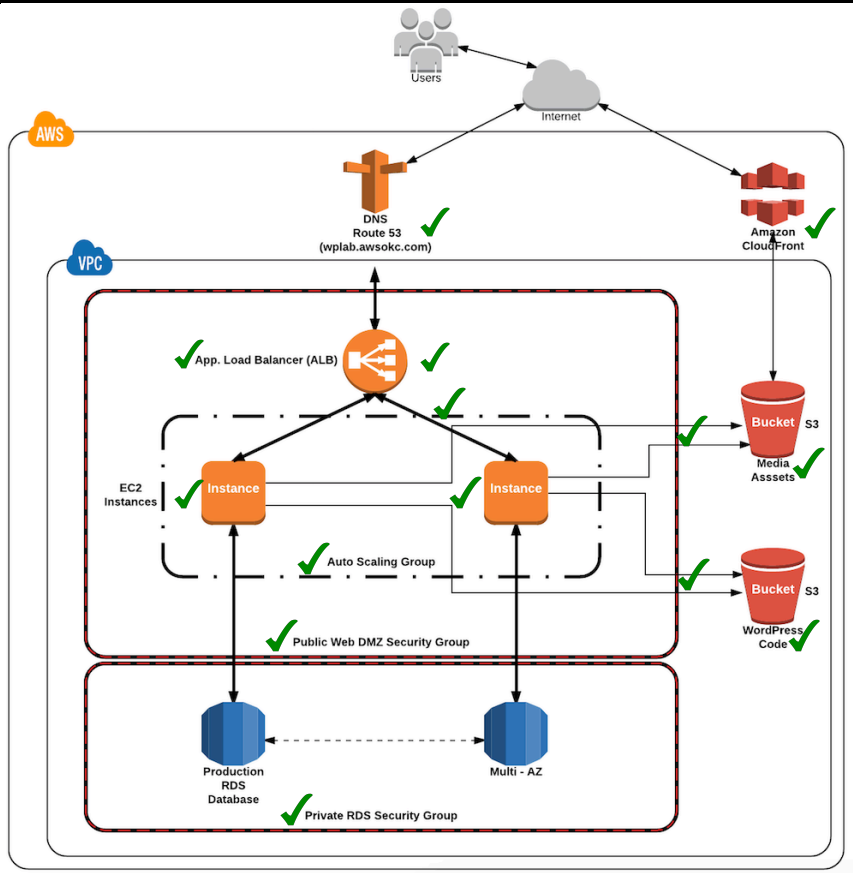


Step 6 – Test Availability / Load

Test 2 – Testing High Load:



Status Check



Power Down Recommendations:

Don’t forget to terminate your resources:

- Auto-Scaling Group and Launch Config
- AMI
- Any Running EC2 Instances
- ALB (This costs you money!)
- RDS Instance (This could cost you money depending on config)
- CloudFront Distribution
- S3 Buckets
- Security Groups
- IAM Role (S3-Admin-Access)
- Route53 DNS Entries (if you used them)