

Demo 1: Nginx

Introduction

Following are the learning objectives of this demonstration:

- Learning about Nginx installation
- Taking a look at the Nginx folder structure
- Understanding server blocks and custom conf files
- Hosting two websites: Hello upGradlearners.com and DevOpslearner.com

Nginx Installation

1. Run the following commands:

```
sudo apt-get update
```

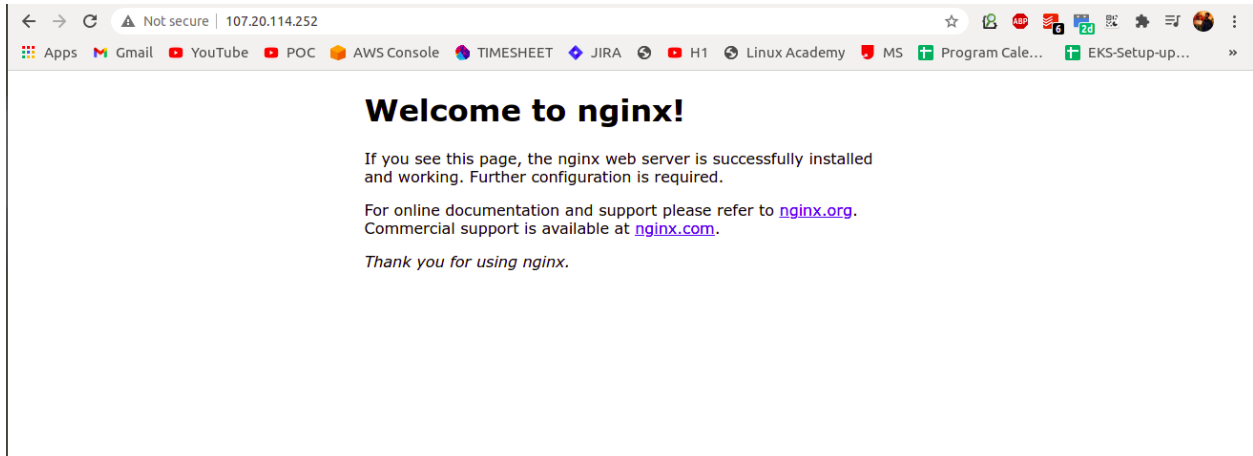
```
sudo apt-get install nginx -y
```

```
sudo service nginx start
```

```
ubuntu@ip-172-31-5-46:~$ sudo service nginx start
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$ sudo service nginx status
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-03-06 09:17:12 UTC; 4s ago
     Docs: man:nginx(8)
  Process: 16812 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 16801 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 16815 (nginx)
   Tasks: 2 (limit: 2348)
  CGroup: /system.slice/nginx.service
          └─16815 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─16819 nginx: worker process

Mar 06 09:17:12 ip-172-31-5-46 systemd[1]: Starting A high performance web server and a reverse proxy server...
Mar 06 09:17:12 ip-172-31-5-46 systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Mar 06 09:17:12 ip-172-31-5-46 systemd[1]: Started A high performance web server and a reverse proxy server.
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$
```

2. Now, hit the instance IP from the web browser to check whether the Nginx installation is done properly.



3. Run `ps -ef --forest | grep nginx` to check the processes running.

```
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$ ps -ef --forest | grep nginx
ubuntu  18534 18304  0 10:32 pts/1    00:00:00      \_ grep --color=auto nginx
root    18483      1  0 10:30 ?        00:00:00  nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
www-data 18484 18483  0 10:30 ?        00:00:00      \_ nginx: worker process
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$
```

Nginx Folder Structure

1. When you run `service nginx start`, Nginx reads its configuration file `nginx.conf` present in the `/etc/nginx` folder.

```
ubuntu@ip-172-31-5-46:~$
ubuntu@ip-172-31-5-46:~$ cd /etc/nginx/
ubuntu@ip-172-31-5-46:/etc/nginx$ ls
conf.d      fastcgi_params  koi-win        modules-available  nginx.conf      scgi_params     sites-enabled  uwsgi_params
fastcgi.conf  koi-utf        mime.types     modules-enabled    proxy_params    sites-available  snippets       win-utf
ubuntu@ip-172-31-5-46:/etc/nginx$
ubuntu@ip-172-31-5-46:/etc/nginx$
```

2. `nginx.conf` looks like the file given below.

```
user www-data; # user by which worker process will be launched
worker_processes auto; # how many worker process will be launched
pid /run/nginx.pid; # pid information
include /etc/nginx/modules-enabled/*.conf; # okay, its saying along with me read these files too
```

```

events {
    worker_connections 768; # how much each worker can handle connections
                           # so if total worker are 4, total connections nginx will
accept will be 4 * 768
}

http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/access.log; # where nginx will store its access logs
    error_log /var/log/nginx/error.log; # where nginx will store error logs
    gzip on;
    include /etc/nginx/conf.d/*.conf; # along with me read these files too
    include /etc/nginx/sites-enabled/*; # along with me read these files too
}

```

3. Now, the question arises: From where is the “welcome to nginx” page is getting served? Let's take a look at the folders present in the nginx.conf file one by one, starting with the 'default' file present at path **/etc/nginx/sites-enabled/**.

```

ubuntu@ip-172-31-5-46:~$ ls /etc/nginx/sites-enabled/
default
ubuntu@ip-172-31-5-46:~$ cat /etc/nginx/sites-enabled/default
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other

```

- 4.

```
root /var/www/html;
```

```
# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
}
```

Hosting Multiple Websites on Nginx

Hosting two websites: upGradlearner.com and DevOpslearner.com using custom conf files

1. Create two files in the `/etc/nginx/sites-enabled` directory of your instance.

```
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$ ls
default  devops  upgrad
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$
ubuntu@ip-172-31-5-46:/etc/nginx/sites-enabled$
```

2. The contents of the two files should look like the following.

devops

```
server {
    listen 80;
    listen [::]:80;

    root /var/www/html/devops;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name devopslearner.com;

    location / {
```

```

        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
}

```

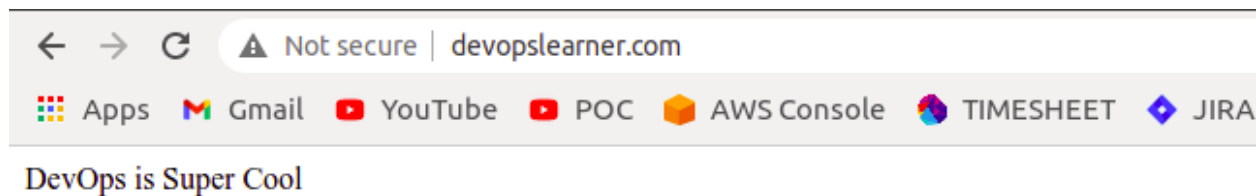
upgrad
<pre> server { listen 80; listen [::]:80; root /var/www/html/upgrad; # Add index.php to the list if you are using PHP index index.html index.htm index.nginx-debian.html; server_name upgradlearner.com; location / { # First attempt to serve request as file, then # as directory, then fall back to displaying a 404. try_files \$uri \$uri/ =404; } } </pre>

3. Now, put the 'index.html' at the respective roots of the websites mentioned above.

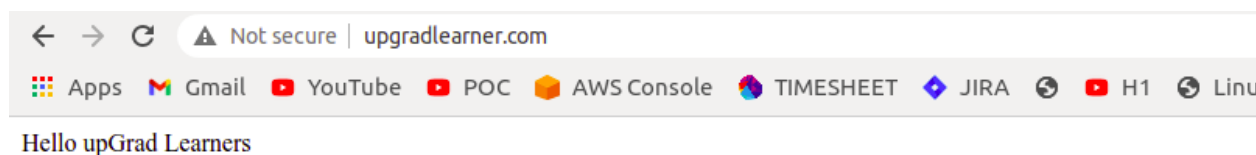
/var/www/html/upgrad/index.html	/var/www/html/devops/index.html
<pre> <html> <head> <title> upGrad Website </title> </head> <body> Hello upGrad Learners </pre>	<pre> <html> <head> <title> DevOps Website </title> </head> <body> DevOps is Super Cool </pre>

<code></body></code> <code></html></code>	<code></body></code> <code></html></code>
--	--

4. Hit the two websites mentioned above.



- 5.



6. Remember that as you have not registered your domain anywhere, it will not resolve to the instance's IP address automatically. So, you need to make changes in your local machine's **/etc/host** file to map both websites to the instance's IP address.

```
$ vi /etc/hosts
$ cat /etc/hosts
127.0.0.1      localhost
54.236.221.246 upgradlearner.com
54.236.221.246 devopslearner.com
$
```

Demo 2: Scaling

Introduction

Let's see how we can scale an application in an AWS environment.

- Vertical scaling in EC2 and RDS
- Horizontal scaling using ASG (auto scaling groups)

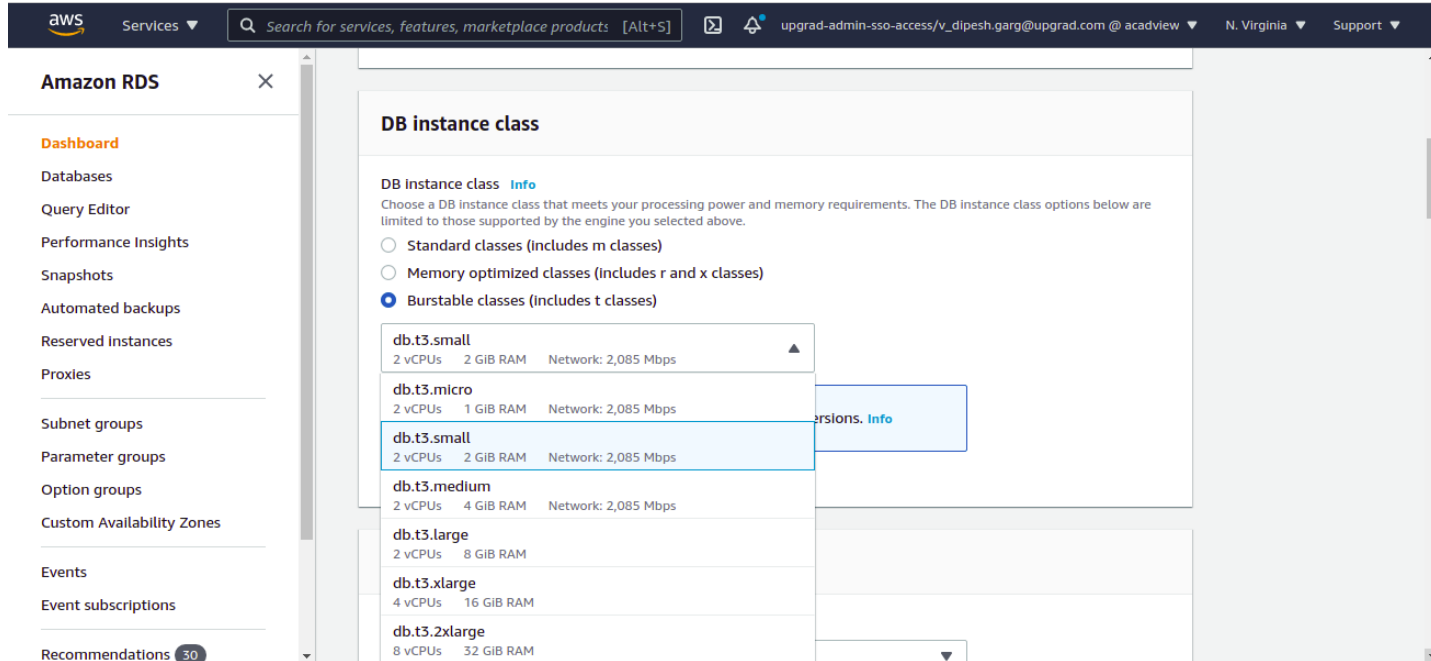
Vertical Scaling in EC2 and RDS

1. Show a running EC2 and RDS, including size, core, etc. and the current utilisation.
2. Vertically scale up and down.

Show the downtime involved in both cases.

You need to stop the EC2 before changing the instance type.

The screenshot displays the AWS Management Console interface for the EC2 Instances page. The top navigation bar includes the AWS logo, a search bar, and user information. The left sidebar shows the navigation menu with options like 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances', 'Instance Types', 'Launch Templates', and 'Spot Requests'. The main content area shows a list of instances with columns for Name, Instance type, and Instance state. The 'Instance state' filter is set to 'running'. The 'TBD-Dev' instance is selected, and the 'Actions' menu is open, showing options like 'Attach to Auto Scaling Group', 'Change instance type', 'Change Nitro Enclaves', 'Change termination protection', 'Change shutdown behavior', 'Change credit specification', and 'Modify instance placement'. The 'Instance details' panel on the right shows various configuration options and their status, including 'Status check' (2/2 checks pass), 'Networking' (2/2 checks pass), 'Security' (2/2 checks pass), 'Image and templates' (2/2 checks pass), 'Monitor and troubleshoot' (2/2 checks pass), 't3a.xlarge' (2/2 checks pass), and 't3.medium' (2/2 checks pass).



Amazon RDS

DB instance class

DB Instance class [Info](#)

Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.

☐ Standard classes (includes m classes)

☐ Memory optimized classes (Includes r and x classes)

☒ Burstable classes (includes t classes)

db.t3.small	2 vCPUs	2 GiB RAM	Network: 2,085 Mbps
db.t3.micro	2 vCPUs	1 GiB RAM	Network: 2,085 Mbps
db.t3.small	2 vCPUs	2 GiB RAM	Network: 2,085 Mbps
db.t3.medium	2 vCPUs	4 GiB RAM	Network: 2,085 Mbps
db.t3.large	2 vCPUs	8 GiB RAM	
db.t3.xlarge	4 vCPUs	16 GiB RAM	
db.t3.2xlarge	8 vCPUs	32 GiB RAM	

Horizontal Scaling

1. In the ASG console, create an ASG of two desired instances.

2. Set the auto scaling policy of Target 60% CPU.

aws Services upgrad-admin-ss0-access/v_dipesh.garg@upgrad.com @ acadview N. Virginia Support

EC2 > Auto Scaling groups > upgrad-staging-apps-on-demand-critical20210120131758550800000005

Create scaling policy

Policy type
Target tracking scaling ▼

Scaling policy name
Target Tracking Policy

Metric type
Average CPU utilization ▼

Target value
60

Instances need
300 seconds warm up before including in metric

Demo 3: Load Balancing

- Implementing different types of load balancing using Nginx
- Load balancing in an AWS environment

Let's create a load balancer between the two websites **facebook.com** and **upgrad.com** with the help of **nginx**. When you hit the nginx web server on your browser, it will serve from any of the two websites based on the load balancing technique.

Create a file with the name **upgrad** in the path **/etc/nginx/sites-enabled/** with the content given below and mention upstream backend as mentioned below with proper parameters to configure different types of load balancers using Nginx.

```
upstream backend {  
    server upgrad.com;  
    server facebook.com;  
}  
  
server {  
    listen 80;  
    listen [::]:80;  
  
    server_name upgradloadbalancing.com;  
  
    location / {  
        proxy_pass http://backend;  
    }  
}
```

```
upstream backend {  
    least_conn; #try changing the type here
```

```
upstream backend {  
    server facebook.com weight=5;
```

```
upstream backend {  
    server upgrad.com down;
```

upGrad

upGrad learning week

What's In Store For You?

Explore Data-Driven Management

- 30+ hours of content
- 8 Offline sessions
- 5 Online webinars
- Resume building interview preparation

Not secure | upgradloadbalancing.com

Gmail YouTube POC AWS Console TIMESHEET JIRA H1 Linux Academy MS



Sorry, something went wrong.

We're working on getting this fixed as soon as we can.

[Go Back](#)

Facebook © 2021 · [Help](#)




Load Balancing in an AWS Environment

In the module on 'AWS Services', you learnt the steps in creating load balancers in AWS. This [document](#) contains a summary of the steps needed to set up a load balancer in AWS.

Select load balancer type

Elastic Load Balancing supports four types of load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. Choose the load balancer type that meets your needs.

[Learn more about which load balancer is right for you](#)

Application Load Balancer	Network Load Balancer	Gateway Load Balancer
		
Create	Create	Create
<p>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.</p> <p>Learn more ></p>	<p>Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.</p>	<p>Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.</p> <p>Learn more ></p>

[Cancel](#)