System Administration & Security

COMP 175 | Fall 2021 | University of the Pacific | Jeff Shafer

Lab 5 Discussion
Server Processes, Load Balancing, and SSH pivoting

Lab 5 – Web Server (Part 2)

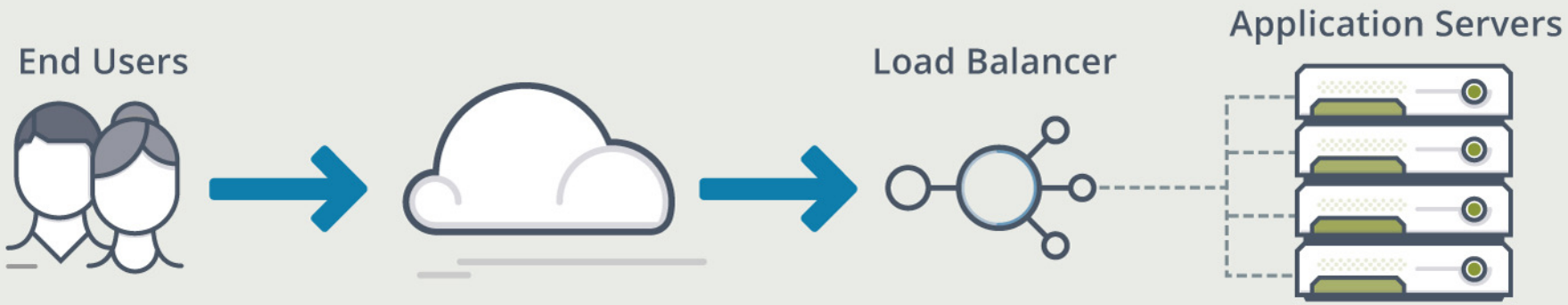
Objectives

- Install NGINX load balancer in front of web server
- SSH Pivoting
- Assign hostnames to each machine

Discussion

- Load balancing
  - Elements & Design
- Security Groups
- Server Processes
  - Run as root or non-root user?
- SSH Pivoting

Load Balancing


End Users
Load Balancer
Application Servers

Load Balancing

- Distribute workload over a larger collection of servers
- Benefits
  - Optimize resource utilization across servers
  - Maximize application throughput
  - Reduce latency
  - Fault-tolerance

Load Balancing with AWS

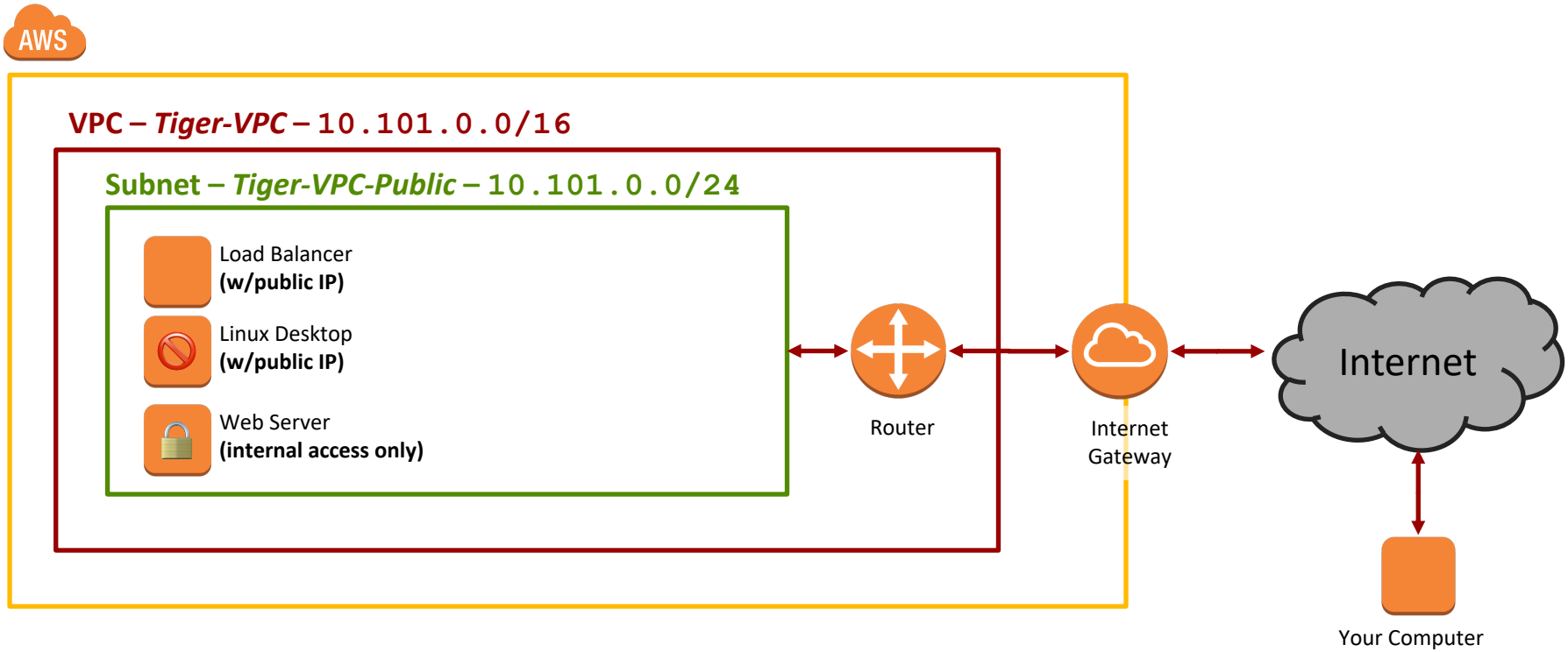
- AWS Elastic Load Balancing
  - Application Load Balancer
    - HTTP or HTTPS only (on VPC)
  - Network Load Balancer
    - TCP or UDP traffic (on VPC)
  - “Classic” Load Balancer
- Pricing
  - Charged by the hour that the load balancer is running, plus usage (connections per second, GB per hour, rule evaluations per second)

Load Balancing with NGINX

- NGINX is a load balancer (and reverse proxy, and HTTP cache) in addition to a web server
- Balancing methods supported
  - Round-robin — Requests to the application servers are distributed in a round-robin fashion
  - Least-connected — Next request is assigned to the server with the least number of active connections
    - Avoids overloading a busy server
  - IP-Hash — a hash-function is used to determine what server should be selected for the next request *(based on the client's IP address)*
    - Session persistence – the same client will always communicate with the same server

Load Balancing

- Why would I pick one over the other?
  - Capabilities?
  - Cost?
    - In direct payments to vendor vs indirect payments to employees
  - Portability / vendor independence?


AWS
VPC – Tiger-VPC – 10.101.0.0/16
Subnet – Tiger-VPC-Public – 10.101.0.0/24
Load Balancer
(w/public IP)
Linux Desktop
(w/public IP)
Web Server
(internal access only)
Router
Internet
Gateway
Internet
Your Computer

Security Groups

- The private web servers should only be accessible:
  - SSH (for administration)
  - HTTP (web server)
- **Where should this traffic be allowed from?**
  - The world?
  - Does the world have any business accessing these web servers?

Security Groups

- Security groups allow you to specify **from where** certain traffic is allowed
  - Individual IP address
    - 203.0.113.1
  - A range of IP addresses in CIDR block
    - 203.0.113.0/24
  - **Another security group**
    - i.e. only instances that are members of a specified security group are allowed access via this rule
    - Set this to the Load Balancer security group
    - Flexible!

Server Processes

Server Processes

- Security vulnerability: Code injection / Remote code execution
  - Commonly due to buffer overflows
- Code will run with the same privileges as the software it is injected into
  - Server process running as *root*? Malicious code will have *root* privileges

Server Processes

- In Linux systems, ports < 1024 are considered system ports and can only be opened by the *root* user
- How to handle this and maintain security?
  - `CAP_NET_BIND_SERVICE` – Grants low-numbered port access to a specific *binary* (if you replace the binary, this needs to be repeated)
  - `authbind` – Grants low-numbered port access to a specific *user / group*
- Or: **Privilege Separation** in program design

Privilege Separation

- **Process 1** – Runs as superuser and `bind()` / `listen()` to the network socket (e.g. port 80 for web)
  - Small, simple, easily auditable
- **Program 2** - Spawned by first program and *drops privileges*
  - Small, simple, easily auditable
- **Program 3** – Spawned by the second program and runs under a under a non-superuser account
  - Runs all the *complex code* handling *untrusted* user data
  - Passed an open file descriptor for the network socket

https://superuser.com/questions/710253/allow-non-root-process-to-bind-to-port-80-and-443/710404#710404

NGINX

- NGINX listens on port 80 as root, but spawns non-root processes (under the www-data username) to do the real work

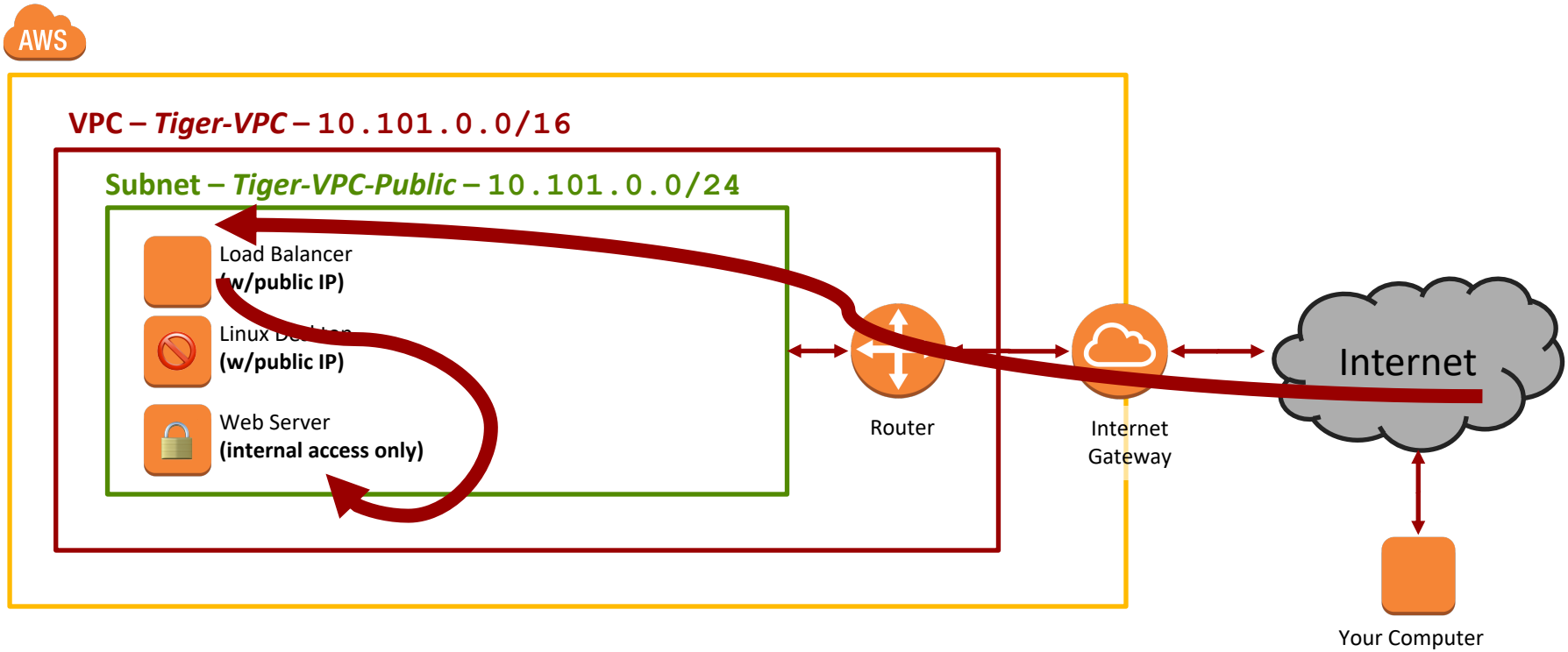
```
ubuntu@cyberlab:~$ sudo netstat -lnptu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      971/sshd
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      999/nginx: master
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      999/nginx: master

ubuntu@cyberlab:~$ sudo ps aux | grep 999
root       999  0.0  1.3 224452  6424 ?        Ss   Jul22   0:00 nginx: master process
/usr/sbin/nginx -g daemon on; master_process on;

ubuntu@cyberlab:~$ sudo pstree -a -s 999
systemd --system --deserialize 38
  └─nginx
      ├─nginx
      └─nginx

ubuntu@cyberlab:~$ ps aux | grep nginx
root       999  0.0  1.3 224452  6424 ?        Ss   Jul22   0:00 nginx: master process
/usr/sbin/nginx -g daemon on; master_process on;
www-data 10049  0.0  2.3 227372 11256 ?        S    Aug23   1:23 nginx: worker process
www-data 10050  0.0  1.5 227100  7200 ?        S    Aug23   0:05 nginx: worker process
```

SSH Pivoting


AWS
VPC – Tiger-VPC – 10.101.0.0/16
Subnet – Tiger-VPC-Public – 10.101.0.0/24
Load Balancer
(w/public IP)
Linux
(w/public IP)
Web Server
(internal access only)
Router
Internet
Gateway
Internet
Your Computer

Wrap-Up

- Questions?
- Concerns?

- **Today**
  - **Lab 4** – Web Server (Part 1)
  - **Lab 5** – Web Server (Part 2)