Docker Interview Questions

edureka!

1. What is the advantage of Docker over hypervisors?

Answer: Docker is light weight and more efficient in terms of resource uses because it uses the host underlying kernel rather than creating its own hypervisor.

2. What is a Docker Image?

Answer: Docker Image is the source of the Docker container. In other words, Docker images are used to create containers. It is possible create multiple isolated containers from a single image.

TIP: Be aware of DockerHub to be able to answer questions on pre-available images.

3. What is a Docker container?

Answer: A Docker Container is an instantiation of Docker image. It is the run time instance of images. Images are set of files whereas containers are the one which runs the image.

4. What is the difference between a Docker Image and a container?

Answer: Docker container is the runtime instance of Docker image. A Docker Image does not have a state and its state never changes as it is just set of files whereas Docker container has its execution
state.

5. How do you create Docker images?

Answer: Docker image are created using Docker file. Docker build is the command to create Docker image out of Docker file. Once an image has been created, as many containers are required can be spawned.

6. How is a Docker container created?

Answer: We can create Docker container by running this command
docker run -t -i <image name> <command name>

7. Can you name some commonly used Docker commands?

Command	Description
docker run -t -i <image name> <command name>	Start and run a container
Docker ps -a	List all running containers
Docker stop <container id>	Stop a container
Docker start <container id>	Start a container
Docker restart <container id>	Restart a container

8. Can you name some other container technologies?

Answer: It has been around in *NIX for quite some time. Some examples include:

- Solaris container (aka Solaris Zones)
- FreeBSD Jails
- AIX Workload Partitions (aka WPARs)
- Linux OpenVZ

9. How to check/configure IP in docker?

Answer: Use docker inspect <container ID>

e.g. docker inspect `docker run -d -p 4321 base nc -lk 4321`|grep IPAddress| cut -d ':' -f 4

(or)

docker inspect --format '{{ .NetworkSettings.IPAddress }}' ${CID}

10. What is DockerHub?

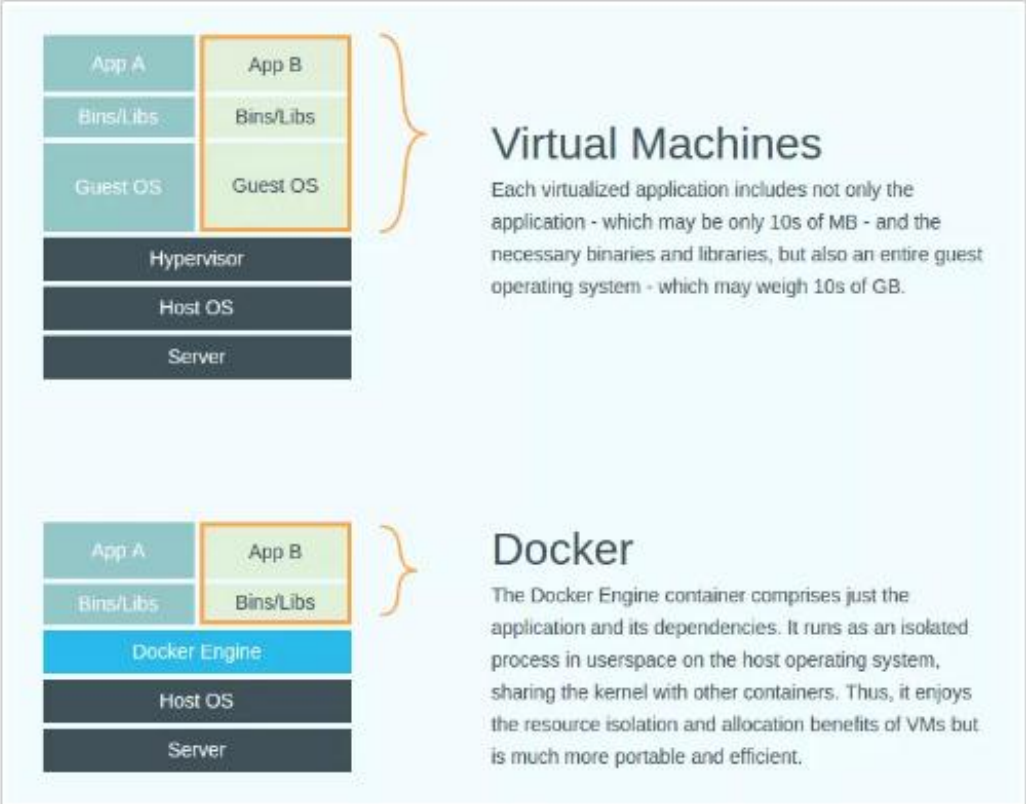
Answer: It is essentially a repository from where you can download containers and use the same. You can also share containers with other applications.

11. How exactly does containerization differ from Virtualization?

Answer: To run an application in virtualized environment (e.g. vSphere), we first need to create a VM, install an OS inside and only then deploy the application.

To run same application in Docker all you need is to deploy that application in Docker. There is no need of additional OS layer. You just deploy the application with its dependent libraries, the rest (kernel, etc.) is provided by Docker engine.

This table from a Docker official website shows it in a quite clear way.



12. Why do my services take a long time to recreate or stop?

Answer: Compose stop attempts to stop a container by sending a SIGTERM. It then waits for a default timeout of 10 seconds. After the timeout, a SIGKILL is sent to the container to forcefully kill it. If you are waiting for this timeout, it means that your containers aren't shutting down when they receive the SIGTERM signal.

There has already been a lot written about this problem of processes handling signals in containers.

To fix this problem, try the following:

- Make sure you're using the JSON form of CMD and ENTRYPOINT in your Dockerfile.
  For example, use ["program", "arg1", "arg2"] not "program arg1 arg2". Using the string form causes Docker to run your process using bash which doesn't handle signals properly. Compose always uses the JSON form, so don't worry if you override the command or entrypoint in your Compose file.
- -If you are able, modify the application that you're running to add an explicit signal handler for SIGTERM.
- -Set the stop_signal to a signal which the application knows how to handle:
  -web: build: . stop_signal: SIGINT
- -If you can't modify the application, wrap the application in a lightweight init system (like s6) or a signal proxy (like dumb-init or tini). Either of these wrappers take care of handling SIGTERM properly.

13. What's the difference between up, run, and start?

Answer: Typically, you want docker-compose up. Use up to start or restart all the services defined in a docker-compose.yml. In the default "attached" mode, you'll see all the logs from all the containers. In "detached" mode (-d), Compose exits after starting the containers, but the containers continue to run in the background.

The docker-compose run command is for running "one-off" or "adhoc" tasks. It requires the service name you want to run and only starts containers for services that the running service depends on. Use run to run tests or perform an administrative task such as removing or adding data to a data volume container. The run command acts like docker run -ti in that it opens an interactive terminal to the container and returns an exit status matching the exit status of the process in the container.

The docker-compose start command is useful only to restart containers that were previously created, but were stopped. It never creates new containers.

14. How does your organization handle multiple languages, libraries and repositories?

Answer: TIP TO ANSWER: Docker works best when there are uniform tools and processes. Implementing Docker where there are multiple tools can be a nightmare. If you have answered yes to having multiple toolsets, including logging, be prepared to answer deeper questions on how these problems were handled at technical, process and personnel levels.

The golden thumb rule is:

> If it's not broken, don't fix it. If you've already invested the engineering time required to build a continuous integration/continuous delivery (CI/CD) pipeline, containerizing legacy apps may not be worth the time investment.

15. What is Docker Swarm?

Answer: Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Because Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts. Supported tools include, but are not limited to, the following:

- Dokku
- Docker Compose
- Docker Machine
- Jenkins

16. Can you name some supported Swarm Discovery backends?

Answer:

- Static file or list of nodes (Not supported with replicating Swarm Managers)
- Hosted Discovery Key store
    - Zookeper

- Etcd
- Consul

17. What are the steps for making Swarm work with TLS?

Answer: These are some of the steps:

a) Ensure all servers can be accessed via SSH and have properly qualified DNS Names
b) Open ports to communicate between Manager and Nodes
c) Open port to communicate between client and Manager
d) Create a Certificate Authority (CA) Server
e) Create and sign Keys
f) Install Keys
g) On Nodes, edit /etc/default/docker and edit DOCKER_OPTS line to use the keys
h) Start Cluster and Swarm Manager
i) Verify same

18. What is Kubernetes?

Answer: Kubernetes (commonly referred to as "k8s") is an open source container cluster manager originally designed by Google and donated to the Cloud Native Computing Foundation. It aims to provide a "platform for automating deployment, scaling, and operations of application containers across clusters of hosts".

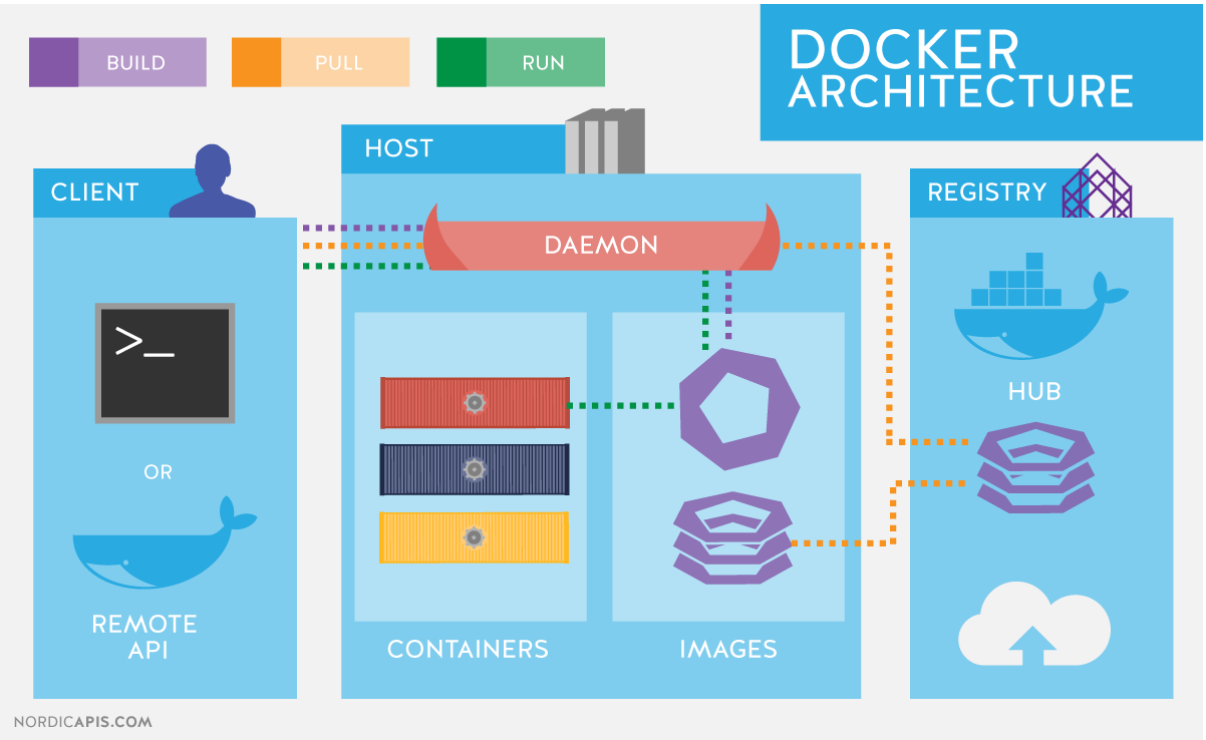
19. You are asked to choose between Kubernetes and Docker. What will influence your decision?

Answer:

(A) IF you have sufficient Docker experience and transitioning to a new platform is cost-prohibitive, choose Swarm.
(B) Before Docker 1.9, we did not have software networking, persistent volumes, and unless-stopped policy. So, if you are using older versions of Docker, you might want to choose Kubernetes (or) more simply plan a migration to Docker 1.9+.

TIP: This question is geared to ask If you are aware of technological alternatives. There are other tools in this space and awareness, if not necessarily practical experience, is a bonus.

20. Explain the Docker Architecture



Tip to Answer: The below is the full answer. Instead, just mention the components, images/containers and then lead with a question of your own on which component to elaborate on.

Each Docker setup includes a Docker client (typically a command line interface, but Docker also features a Remote API and a daemon, the persistent process that runs on each host and listens to API calls. Both the client and the daemon can share a single host, or the daemon can run in a remote host.

Docker images are read-only templates from which containers are generated. An image consists of a snapshot of a Linux distribution like Ubuntu or Fedora — and maybe a set of applications or runtime environments, like Apache, Java, or ElasticSearch. Users can create their own Docker images, or reuse one of the many images created by other users and available on the Docker Hub.

Docker registries are repositories from which one can download or upload Docker images. The Docker Hub is a large public registry, and can be used to pull images within a Docker workflow, but more often teams prefer their own registry containing the relevant subset of public Docker images that it requires along with its own private images.

Docker containers are directories containing everything needed for the application to run, including an operating system and a file system, leveraging the underlying system's kernel but without relying on anything environment-specific. This enables containers to be created once and moved from host to host without risk of configuration errors. In other words, the

exact same container will work just as well on a developer's workstation as it will on a remote server.
A Docker workflow is a sequence of actions on registries, images and containers. It allows a team of developers to create containers based on a customized image pulled from a registry, and deploy and run them on a host server. Every team has its own workflow — potentially integrating with a continuous integration server like Jenkins, configuration management tools like Chef or Puppet, and maybe deploying to cloud servers like Amazon Web Services. The daemon on each Docker host enables further actions on the containers — they can be stopped, deleted or moved. The result of these actions is called lifecycle events.

21. How do I run multiple copies of a Compose file on the same host?

Answer: Compose uses the project name to create unique identifiers for all a project's containers and other resources. To run multiple copies of a project, set a custom project name using the -p command line option or the COMPOSE_PROJECT_NAME environment variable.

22. Can I use JSON instead of YAML for my Compose file?

Answer: Yes. YAML is a superset of JSON so any JSON file should be valid YAML. To use a JSON file with Compose, specify the filename to use

23. Should I include my code with COPY/ADD or a volume?

Answer: You can add your code to the image using COPY or ADD directive in a Dockerfile. This is useful if you need to relocate your code along with the Docker image, for example when you're sending code to another environment (production, CI, etc.). Prefer COPY over ADD as image size is reduced with COPY over ADD.

You should use a volume if you want to make changes to your code and see them reflected immediately, for example when you're developing code and your server supports hot code reloading or live-reload.

There may be cases where you'll want to use both. You can have the image include the code using a COPY, and use a volume in your Compose file to include the code from the host during development. The volume overrides the directory contents of the image.

24. What is Immutable infrastructure?

Answer: The common use case is to have a container run a web server and mount the source code from the host. This is referred to as mutable *images* because if contents of the mounted volume are changed, container behaviour changes.

An *immutable image* is an image that contains everything it needs to run the application, so obviously including your source code. The "only" difference is that your Dockerfile will copy your application code and eventually run any build process or dependencies installation. 2 main advantages:

(A) Self-contained images are more portable, scalable and easily change run time dependencies.
(B) Given tag of that image will always have the same behaviour making it easier to
    a. Test
    b. Roll back
    c. Perform A/B testing

25. Can you explain about Docker networking?

Answer: When you install Docker, it creates three networks automatically. You can list these networks using the Docker network ls command. These 3 networks are: Bridge, None, Host. When you run a container, you can specify the network on which the container should run using –network.

The bridge network is docker0. This is the default network. The none network adds a container to a container-specific network stack. That container lacks a network interface. The host network adds a container on the hosts network stack. You'll find the network configuration inside the container is identical to the host.

TIP: For additional details, please understand subnetting, gateways and the subnet 172.17.0.1/16. Related questions on same: Docker attach/detach Docker network inspect In fact, most question on Dockers seem to focus on the syntax/arguments for various Docker commands. Please be very thorough on same

26. What is Docker Compose?

Answer: Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application's services. Then, using a single command, you create and start all the services from your configuration.

27. What is your typical use flow for Compose?

Answer:

- I would define my application environment with a Dockerfile so it can be reproduced anywhere.

- Define the services that make up my application in docker-compose.yml so they can be run together in an isolated environment.
- Lastly, run docker-compose up and Compose will start and run your entire app.

28. What is a Dockerfile used for?

Answer: Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using Docker build users can create an automated build that executes several command-line instructions in succession.

29. Can you talk about some of the best practices you have used in using Docker Files?

TIP to answer: Answer honestly on this one. Some of the points below are taken from Docker' own guidance, answer them in the context of your experience.

Answer:

1. Keep dockerfiles short and quick to build
2. Using of .dockerignore files and/or keeping DockerFiles in empty directories
3. Avoiding using unnecessary packages
4. Running only one service per container
5. Avoid ADD/COPY as much as possible since this means computing checksums for each file.
6. Prefer to use Copy over ADD.
7. As far as possible, use current Official repositories
8. Avoiding use of apt-get upgrade over apt-get install -y
9. Always combine apt-get update and apt-get install -y
10. Keep absolute paths for WORKDIR
11. ……

30. Tell us how you have used Docker in your past position.

Answer: Explain how you have used Docker to help rapid deployment. Explain how you have scripted Docker and used Docker with other tools like Puppet, Chef or Jenkins. If you have no past practical experience in Docker and have experience with other tools in similar space, be honest and explain the same. In this case, it makes sense if you can compare other tools to Docker in terms of functionality.

Networking Concepts

1. What is DHCP?

Answer: Dynamic Host Configuration Protocol (DHCP) is a network protocol that enables a server to automatically assign an IP address to a computer from a defined range of numbers (i.e., a scope) configured for a given network.

2. How does it work?

Answer: TIP TO ANSWER: Explain the various messages Types including DHCPDiscover. Then explain how this is done as below:

When a client needs to start up TCP/IP operations, it broadcasts a request for address information. The DHCP server receives the request, assigns a new address for a specific time (called a lease period) and sends it to the client together with the other required configuration information. This information is acknowledged by the client, and used to set up its configuration. The DHCP server will not reallocate the address during the lease period and will attempt to return the same address every time the client requests an address. The client may extend its lease with subsequent requests, and may send a message to the server before the lease expires telling it that it no longer needs the address so it can be released and assigned to another client on the network.

3. What are DHCP Messages?

Answer:

1. DHCPDISCOVER
2. DHCPOFFER
3. DHCPREQUEST
4. DHCPACK
5. DHCPNAK
6. DHCPDECLINE
7. DHCPRELEASE
8. DHCPINFORM

Expect to get individual questions on what each of these messages represent and how they are used.

4. Can you Explain the flow of messages in assigning a new address?

Answer:

- A Client sends a DHCPDISCOVER Message on Ethernet Broadcast Address.
- Servers respond with DHCPOFFER message.
- Client then sends DHCPREQUEST Message on Ethernet Broadcast Address accepting one server and rejecting other servers.
- Server sends a DHCPACK Message to client and client completes Initialization
- On Shutdown. Client sends DHCPRELEASE Message to server.

5. What is meant by a lease?

Answer: With all the necessary information on how DHCP works, one should also know that the IP address assigned by DHCP server to DHCP client is on a lease. After the lease expires the DHCP server is free to assign the same IP address to any other host or device requesting for the same. For example, keeping lease time 8-10 hours is helpful in case of PC's that are shut down at the end of the day. So, lease must be renewed from time to time. The DHCP client tries to renew the lease after half of the lease time has expired. This is done by the

exchange of DHCPREQUEST and DHCPACK messages. While doing all this, the client enters the renewing stage.

6. What is meant by DHCP Scope?

Answer: DHCP scopes are used to define ranges of addresses from which a DHCP server can assign IP addresses to clients.

7. What are the types of Scope?

Answer:

- Normal Scope - Allows A, B and C Class IP address ranges to be specified including subnet masks, exclusions and reservations. Each normal scope defined must exist within its own subnet.
- Multicast Scope - Used to assign IP address ranges for Class D networks. Multicast scopes do not have subnet masks, reservation or other TCP/IP options.
  Multicast scope address ranges require that a Time To Live (TTL) value be specified (essentially the number of routers a packet can pass through on the way to its destination).
- Superscope - Essentially a collection of scopes grouped together such that they can be enabled and disabled as a single entity.

8. Explain why you need integration between DHCP and DNS.

Answer: Traditionally, DNS and DHCP servers have been configured and managed one at a time. Similarly, changing authorization rights for a particular user on a group of devices has meant visiting each one and making configuration changes.

DHCP integration with DNS allows the aggregation of these tasks across devices, enabling a company's network services to scale in step with the growth of network users, devices, and policies, while reducing administrative operations and costs. This integration provides practical operational efficiencies that lower total cost of ownership.

Creating a DHCP network automatically creates an associated DNS zone, for example, reducing the number of tasks required of network administrators. And integration of DNS and DHCP in the same database instance provides unmatched consistency between service and management views of IP address-centric network services data.

Refer to class instruction guides on how to set this integration in Ubuntu 14. Refer to other guides on setting up DDNS (Dynamic DNS)

9. Can a BOOTP Client use DHCP

Answer: Only if the DHCP server is specifically written to also handle BOOTP queries.
TIP: Learn about BOOTP and its integration with DHCP and PXE.

10. What's a PTR in DNS?

Answer: Pointer records are used to map a network interface (IP) to a host name. These are primarily used for reverse DNS. Reverse DNS is setup very like how normal (forward) DNS is setup. When you delegate the DNS forward, the owner of the domain tells the registrar to let your domain use specific name servers.

11. What is an MX record in DNS?

Answer: MX records are mail exchange records used for determining the priority of email servers for a domain. The lowest priority email server is the first destination for email. If the lowest priority email server is unavailable, mail will be sent to the higher priority email servers.

12. How does HTTP work?

Answer: The HTTP protocol works in a client and server model like most other protocols. A web browser using which a request is initiated is called as a client and a web server software which responds to that request is called a server. World Wide Web Consortium and the Internet Engineering Task Force are two important spokes in the standardization of the HTTP protocol. HTTP allows improvement of its request and response with the help of intermediates, for example a gateway, a proxy, or a tunnel. The resources that can be requested using the HTTP protocol, are made available using a certain type of URI (Uniform Resource Identifier) called a URL (Uniform Resource Locator). TCP (Transmission Control Protocol) is used to establish a connection to the application layer port 80 used by HTTP.

13. Give an Example of how you have optimized Apache (or) NGINX

Answer: TIP to answer: Demonstrate understanding of Application Performance Management (APM) tools and frameworks, benchmarking, using tools for apache like ab, load testing tools (JMETER). Explain key parameters like thread count, threading models (pre-fork/worker), worker processes and how you have a measurement strategy and SLA Tracking. Do not just answer in terms of technical answers. Especially if you can demonstrate how you use tool sets like ELK to graphically visualize data to quickly identify problem areas, that will be great.

14. If you had to choose a load balancer, what determines the choice of algorithms?

Answer:

- Understand the application stack.
- Understand the nature of the application (Static/Dynamic/Requires Session proximity/Stickiness)
- Network layers – Layer 4 vs Layer 7 vs Connection Oriented vs Message Oriented
- Choice of Protocols

In general, one of the most misunderstood aspects of load balancing is that a load balancing algorithm is designed to choose from a pool of resources and that an application is or can be made up of multiple pools of resources. These pools can be distributed (cloud balancing) or localized, and they may all be active or some may be designated as solely existing for failover

purposes. Ultimately this means that the algorithm does not actually choose the pool from which a resource will be chosen – it only chooses a specific resource. The relationship between the choice of a pool and the choice of a single resource in a pool is subtle but important when making architectural decisions – especially those that impact scalability. A pool of resources is (or should be) a set of servers serving similar resources. For example, the separation of image servers from application logic servers will better enable scalability domains in which each resource can be scaled individually, without negatively impacting the entire application. You will be asked on this if you are going to be serious about scalability.

15. You are asked to implement HA Proxy to load balance 2 Apache Servers. What challenges do you foresee?

Answer: Centralized logging is one of the expected answers. In addition, ensuring SPOF for HAProxy itself is another challenge. If the servers are dynamically spun off, including script to update proxy configurations dynamically is another.

16. Have you integrated DHCP and DNS together? Give an example.

Answer: Answer honestly on this one. If you just have theoretical knowledge, say so. (HINT: This is presented together as an installation guide for this module on Ubuntu 12+).

17. Given you are free to select a web server, which of these would you choose? Apache or NGINX? Why?

Answer: Tip to answer: The temptation is say NGINX automatically. However, the nature of the application, the programming language behind the application, expected volumes of visitors and SLA are all important factors. NGINX is better off the bat for static applications, but the same can easily be achieved with multiple servers If an existing load balancer is already in place and with judicious tuning of parameters. PHP Servers require separate cache and backend servers and are easier to configure/setup in Apache.