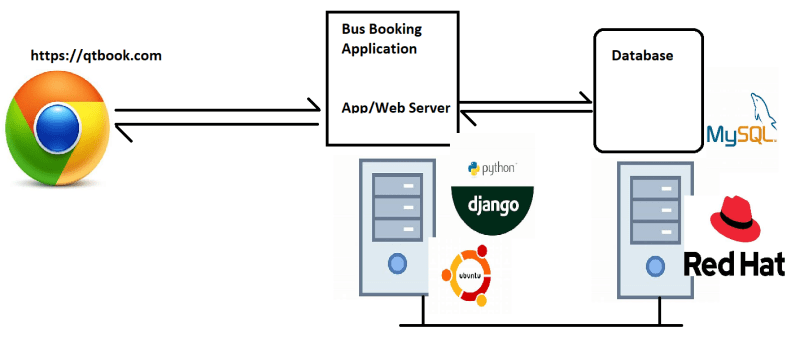
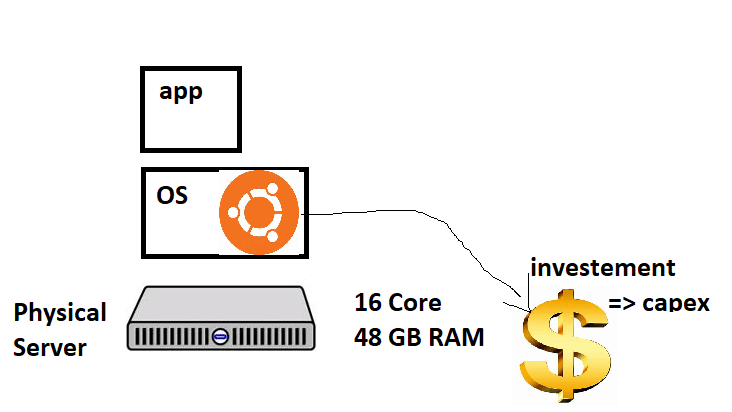
**Generations of Running Applications**

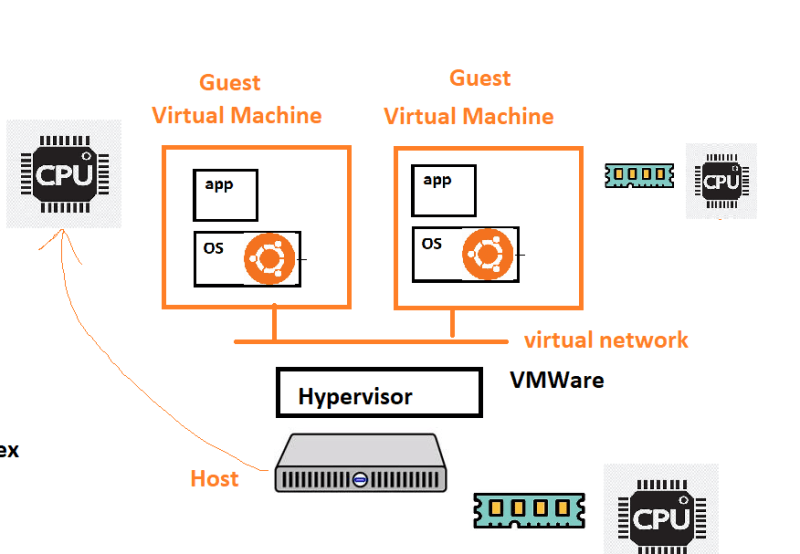
* Applications bring revenue, to make applications run we need servers.
* There are 3 major generations/approaches of running applications on servers
* Sample Application Architecture  
  

**Generation 1: Run directly on physical server**

* Steps involved
  + Procure Physical server (costs involved => capex)
  + Install Os (Licensing costs might be involved)
  + Deploy/install application & necessary software’s.
* Problems:
  + In the cases of under usage, hardware resources are wasted and Return on Investment will be under question.

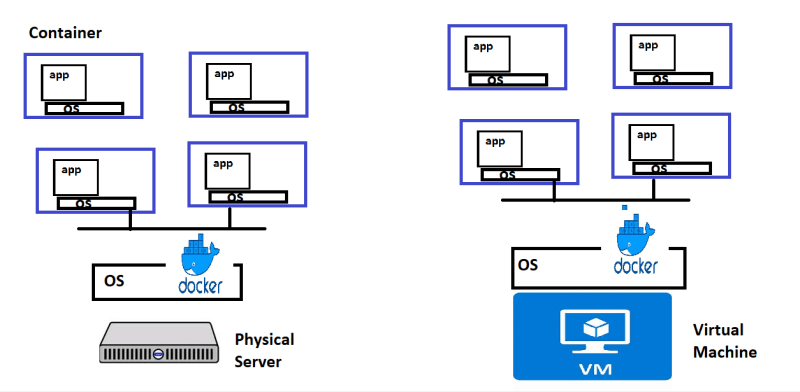


**Generation 2: Hypervisors**

* Install Hypervisor on Physical Server which will help in creating multiple virtual machines  
  
* Hypervisor licensing costs are involved, in addition to os costs

**Generation 3: Containers**

* Container is an isolated area created by container engine (docker) which has a lean os and application runs inside container.
* Each container gets
  + an ip address
  + cpu
  + RAM



**Advantage of running application in dockized way:**

**1. Return on investment & cost savings**

The first advantage of using docker is the ROI. The biggest driver of most management decisions when selecting a new product is the return on investment. The more a solution can drive down costs while raising profits, the better the solution is, especially for large, established companies, that need to generate steady revenue on the long term.  
In this sense, Docker can help facilitate this type of savings by dramatically reducing infrastructure resources. The nature of Docker is that fewer resources are necessary to run the same application. Because of the reduced infrastructure requirements that Docker has, organizations are able to save on everything from server costs to the employees needed to maintain them. Docker allows engineering teams to be smaller and more effective.

**2. Standardization & productivity**

Docker containers ensure consistency across multiple development, release cycles and standardising your environment. One of the biggest advantages to a Docker-based architecture is actually standardization. Docker provides repeatable development, build, test, and production environments. Standardizing service infrastructure across the entire pipeline allows every team member to work on a production parity environment. By doing this, engineers are more equipped to efficiently analyze and fix bugs within the application. This reduces the amount of time wasted on defects and increases the amount of time available for feature development.

As we mentioned, Docker containers allow you to commit changes to your Docker images and version control them. For example, if you perform a component upgrade that breaks your whole environment, it is very easy to rollback to a previous version of your Docker image. This whole process can be tested in a few minutes. Docker is fast, allowing you to quickly make replications and achieve redundancy. Also, launching Docker images is as fast as running a machine process.

**3. CI efficiency**

Docker enables you to build a container image and use that same image across every step of the deployment process. A huge benefit of this is the ability to separate non-dependent steps and run them in parallel. The length of time it takes from build to production can be sped up notably.

**4. Compatibility & maintainability**

Eliminate the “it works on my machine” problem once and for all. One of the benefits that the entire team will appreciate is parity. Parity, in terms of Docker, means that your images run the same no matter which server or whose laptop they are running on. For your developers, this means less time spent setting up environments, debugging environment-specific issues, and a more portable and easy-to-set-up codebase. Parity also means your production infrastructure will be more reliable and easier to maintain.

**5. Simplicity & faster configurations**

One of the key benefits of Docker is the way it simplifies matters. Users can take their own configuration, put it into code and deploy it without any problems. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

**6. Rapid Deployment**

Docker manages to reduce deployment to seconds. This is due to the fact that it creates a container for every process and does not boot an OS. Data can be created and destroyed without worry that the cost to bring it up again would be higher than affordable.

**7. Continuous Deployment & Testing**

Docker ensures consistent environments from development to production. Docker containers are configured to maintain all configurations and dependencies internally. So, you can use the same container from development to production making sure there are no discrepancies or manual intervention.  
If you need to perform an upgrade during a product’s release cycle, you can easily make the necessary changes to Docker containers, test them, and implement the same changes to your existing containers. This sort of flexibility is another key advantage of using Docker. Docker really allows you to build, test and release images that can be deployed across multiple servers. Even if a new security patch is available, the process remains the same. You can apply the patch, test it and release it to production.

**8. Multi-Cloud Platforms**

This is possibly one of Docker’s greatest benefits. Over the last few years, all major cloud computing providers, including Amazon Web Services (AWS) and Google Compute Platform (GCP), have embraced Docker’s availability and added individual support. Docker containers can be run inside an Amazon EC2 instance, Google Compute Engine instance, Rackspace server or VirtualBox, provided that the host OS supports Docker. If this is the case, a container running on an Amazon EC2 instance can easily be ported between environments, for example to VirtualBox, achieving similar consistency and functionality. Also, Docker works very well with other providers like Microsoft Azure, and OpenStack, and can be used with various configuration managers like Chef, Puppet, and Ansible,etc.

**9. Isolation**

Docker ensures your applications and resources are isolated and segregated. Docker makes sure each container has its own resources that are isolated from other containers. You can have various containers for separate applications running completely different stacks. Docker helps you ensure clean app removal since each application runs on its own container. If you no longer need an application, you can simply delete its container. It won’t leave any temporary or configuration files on your host OS.  
On top of these benefits, Docker also ensures that each application only uses resources that have been assigned to them. A particular application won’t use all of your available resources, which would normally lead to performance degradation or complete downtime for other applications.

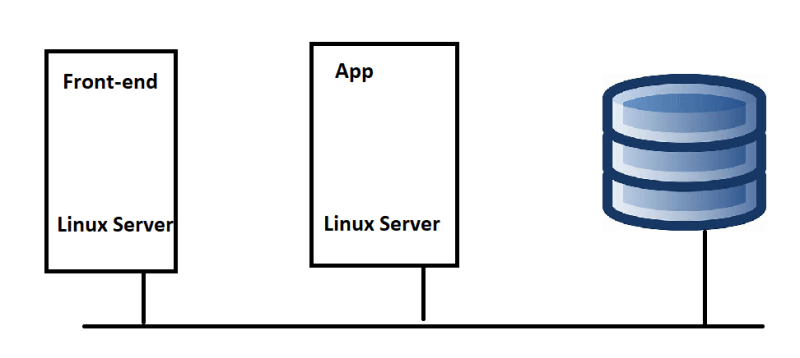
**10. Security**

And the last benefit of using docker is – security. From a security point of view, Docker ensures that applications that are running on containers are completely segregated and isolated from each other, granting you complete control over traffic flow and management. No Docker container can look into processes running inside another container. From an architectural point of view, each container gets its own set of resources ranging from processing to network stacks.

**Containers contd…**

* Let’s try to run some applications in container.
* Note: In this demonstration, I have created a ubuntu Linux and installed docker and executed some commands (you can ignore all of this)
* Docker has made it extremely simple to create containers (which was built in Linux concept almost from day1)

**The idea of microservices**

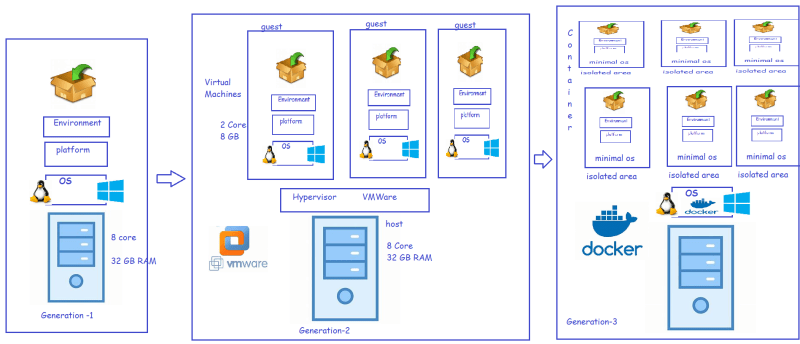
* Let’s look at a typical ecommerce application
* Layered architecture of e-commerce application  
  
* Let’s focus on app tier/layer. This does the following functions
  + Administration
  + User Management
  + Inventory
  + Catalog
  + Cart
  + Payments
  + Notifications
* Micro Services:
  + This is all about breaking a monolith to multiple smaller services
  + Each micro service generally represents a module/functionality which can run on its own
  + Smaller changes are possible and replacing services is easier (new versions)

Note: Primary role of docker is:

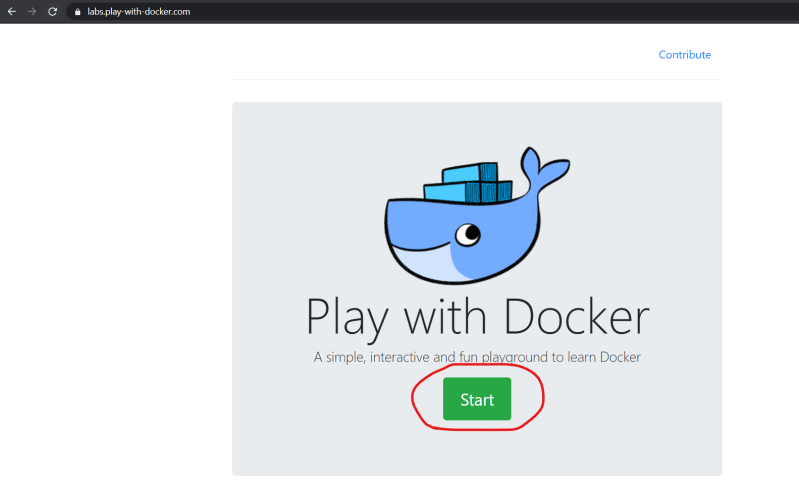
1. Converting monolithic application to microservice application
2. Packing application with dependencies.

Advantage of this approach: No dependencies in environment.

**Containers: Big picture**

* Applications are heart of business.
* Most of the Applications run on Servers
* Earlier we used to run one application on each server
* Then we had hypervisors which helped in creating virtual machines
* VM’s are great but they are far from perfect
* Every vm requires its own dedicated OS is a major flaw, Every OS consumes CPU, RAM and other resources that could otherwise be used to power more applications
* For a long time, big players like Google have been using container technologies to address the shortcoming of vm model
* But an organization called as Docker Inc has made it very simple to create container which was not the case earlier
* Companies like Microsoft joined hands with Docker and made changes in their OS to accommodate docker container 

**Lab Environment**

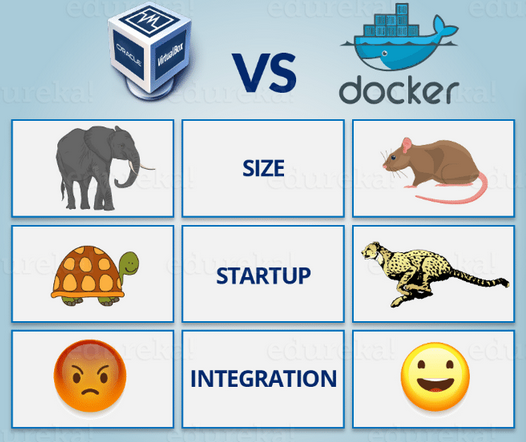
* Create a Docker hub account (https://hub.docker.com/)
* Once your account creation is completed login into docker playground(https://labs.play-with-docker.com/) using docker hub account 



To toggle full screen use ALT+Enter

Note: Key Selection point is what is important, application or infrastructure. If application is important then go for docker, if infrastructure is important then go for VM.

Docker VS VM:



Drawback of container: Security, container doesn’t provide security. But you can achieve security though host OS.