

HOW TO INTRODUCE DOCKER CONTAINERS IN THE ENTERPRISE



Docker [container technology](#) has seen a rapid rise in early adoption and broad market acceptance. It is a technology that is seen to be a strategic enabler of business value because of the benefits it can provide in terms of:

- Reduced cost
- Reduced risk
- Increased speed

For enterprises that haven't worked with Docker, introducing it can seem daunting. How do you achieve business value, run Docker in development, test, and production, or effectively use [automation](#) with Docker?

As experienced users of this transformative tool, we have had success with a three-step yellow brick road approach. This process will enable your enterprise to embark on the Docker journey too.

(This is part of our [Docker Guide](#). Use the right-hand menu to navigate.)

Getting started with Docker containers

Step 1: Evaluation

In the early phases, engineers play and evaluate Docker technology by dockerizing a small set of applications.

1. First, you'll need a Docker host. Ubuntu or Redhat machines can be used to setup Docker in a

few minutes by following instructions at the [Docker website](#).

2. Once the Docker host is set, at least initial development can be done in an insecure mode (no need for certificates in this phase). You can login to the Docker host and use Docker pull and run commands to run a few containers from the public Docker hub.
3. Finally, selecting the right applications to dockerize is extremely important. Stateless internal or non-production apps would be a good way to start converting them to containers. Conversion requires the developer to [write Docker files](#) and become familiar with [Docker build commands](#) as well. The output of the build is a Docker image. Usually, an internal private Docker registry can be installed or the public Docker hub can be used with a private account so your images do not become public.

Step 2: Pilot

In the pilot phase, the primary goals are to start bringing in [IT and DevOps teams](#) to go through infrastructure and operations to setup Docker applications. An important part of this phase is to "IT-ize" the Docker containers to run a pilot in the IT production so that IT operations team can start managing them. This phase requires that IT operations manage dual stacks:

- [Virtualization platforms](#) like VMware vCenter and vSphere infrastructure for [virtual machines \(VMs\)](#)
- New infrastructure for running Docker application containers

Management systems and software tools will be needed in four primary areas:

1. **Build Docker infrastructure.** Carve out a new Docker infrastructure consisting of a farm of Docker hosts to run containers alongside with traditional virtualization platforms and [hybrid clouds](#).
2. **Define & deploy your app as a collection of containers.** Management system software can provide blueprints to define application topology consisting of Docker containers. Spin them up and then provide "Day 2" management of the containers for end users, such as start/stop and monitoring of Docker applications. They can also integrate with Docker Hubs or Docker Trusted Registry for sourcing images.
3. **Build your delivery pipeline.** [DevOps products](#) can offer [CI/CD](#) workflows for continuous integration and continuous deployment of Docker images.
4. **Vulnerability testing of containers.** [Server automation tools](#) can be used to do SCAP vulnerability testing of Docker images.

Step 3: Production

Now, you can deploy Docker containers to your production infrastructure. This will require not just [DevOps](#) and deployment of containers to a set of Docker hosts, but also security, compliance, and [monitoring](#).

Supporting complex application topologies is a degree of sophistication many enterprises will, in fact, desire in order to:

- Allow gradual introduction to the benefits of containers
- Keeping the data in the traditional virtual or physical machines

Another degree of sophistication is the introduction of more complex distributed orchestration to

improve data center utilization and reduce operational placement costs.

While in the previous phase we had used static partitioning of infrastructure resources into clusters, this phase will use more state of the art cluster schedulers such as [Kubernetes](#) or Fleet.

Governance, [change control](#), CMDB integration, and quota management are some of the ways enterprise can start governing the usage of Docker as it grows in the enterprise. Container sprawl reduction through reclamation are additional processes that need to be automated at this level.

Final thoughts

Evaluate the business benefits at the end of each of these steps to determine if you've achieved ROI and accomplished your goals.

We believe that using this three-step phased approach to introducing Docker, with increasing sophisticated usage and automation, will make it easy to test drive and productize Docker inside enterprises.

Related reading

- [BMC DevOps Blog](#)
- [Docker Commands: A Cheat Sheet](#)
- [Docker vs Kubernetes: What's The Difference?](#)
- [How To Run MongoDB as a Docker Container](#)
- [The State of Containers Today: A Report Roundup](#)
- [Managing Containers & Code for DevOps](#)