Chapter 1.

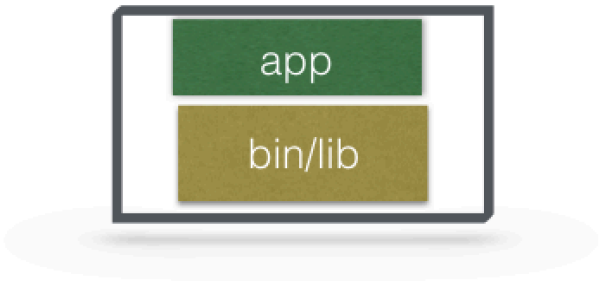
Container Orchestration Introduction

Learning Objectives

- By the end of this chapter, you should be able to:
    - Define the concept of container orchestration.
    - Explain the reasons for doing container orchestration.
    - Discuss different container orchestration options.
    - Discuss different container orchestration deployment options.

What Are Containers?

Containers are an application-centric way to deliver high-performing, scalable applications on the infrastructure of your choice.

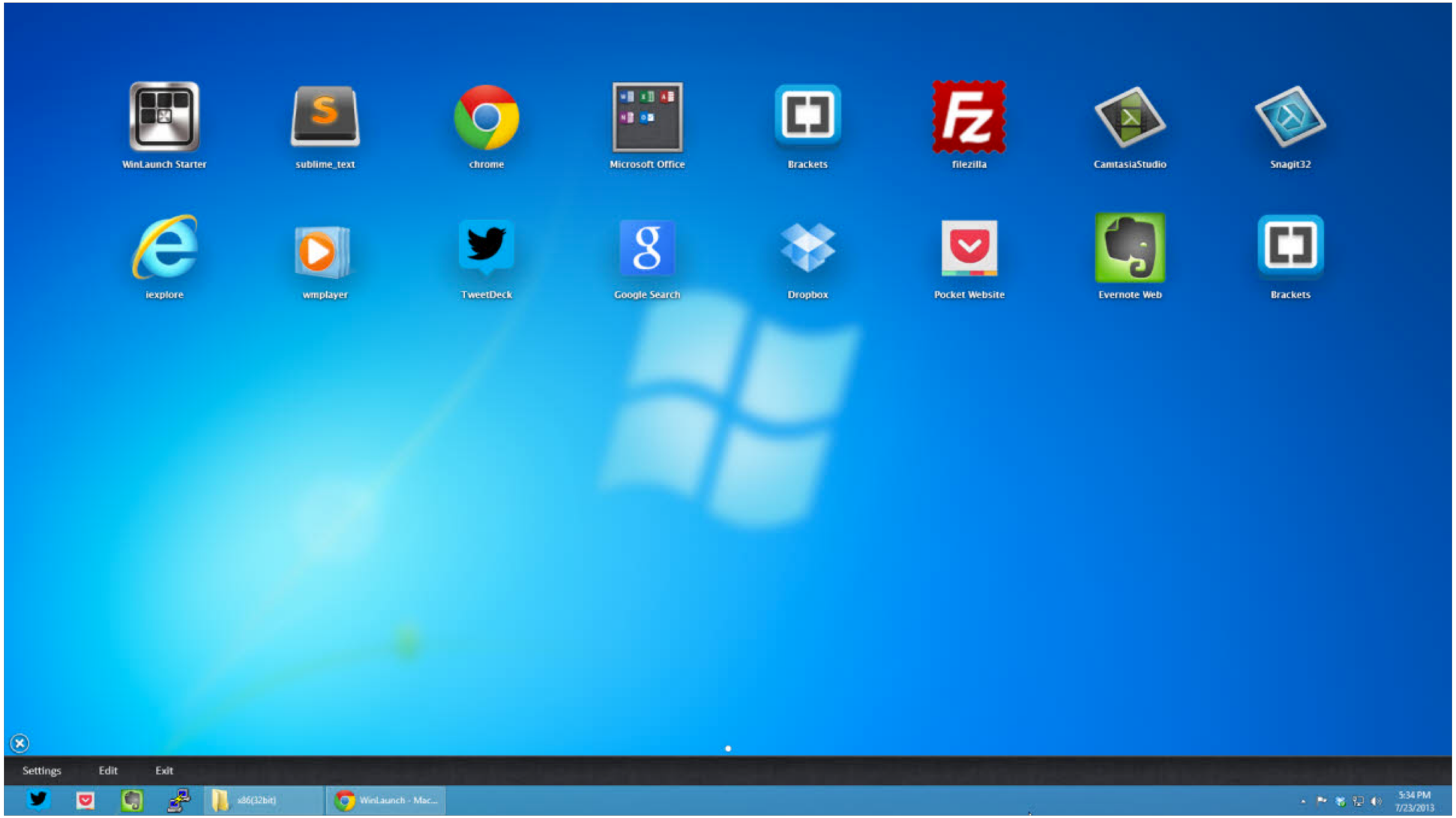


Desktop | Dev VM | QA Env. | Public Cloud | Private Cloud | Customer Site

With a **container image**, we bundle the application along with its runtime and dependencies. We use that image to create an isolated executable environment, also known as container. We can deploy containers from a given image on the platform of our choice, such as desktops, VMs, cloud, etc.

```
front:
  build: .
  ports:
    - "80:80"
    - "443:443"
    - "9000:9000"
  links:
    - mysql:mysql
    - mongo:mongo
    - memcached:memcached
    - redis:redis
    - elasticsearch:elasticsearch
  volumes:
    - ./www:/var/www
    - ./sites:/etc/nginx/conf.d
    - ./logs:/var/log/supervisor

mysql:
  image: mysql
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: password

mongo:
  image: mongo
  ports:
    - "27017:27017"

memcached:
  image: memcached
  ports:
    - "11211:11211"

redis:
  image: redis
  ports:
    - "6379:6379"

elasticsearch:
  image: elasticsearch
  ports:
    - "9200:9200"
    - "9300:9300"
```

Docker application vs Windows application



What Is Container Orchestration?

In the **quality assurance (QA) environments**, we can get away with running containers on a single host to develop and test applications. However, when we go to production, we do not have the same liberty, as we need to ensure that our applications:

- Are fault-tolerant
- Can scale, and do this on-demand
- Use resources optimally
- Can discover other applications automatically, and communicate with each other
- Are accessible from the external world
- Can update/rollback without any downtime.

Container orchestrators are the **tools which group hosts together to form a cluster, and help us fulfill the requirements mentioned above.**

Container Orchestrators

Nowadays, there are many container orchestrators available, such as:

- **Docker Swarm**
  Docker Swarm is a container orchestrator provided by Docker, Inc. It is part of Docker Engine.
- **Kubernetes**
  Kubernetes was started by Google, but now, it is a part of the Cloud Native Computing Foundation project.
- **Mesos Marathon**
  Marathon is one of the frameworks to run containers at scale on Apache Mesos.
- **Amazon ECS**
  Amazon EC2 Container Service (ECS) is a hosted service provided by AWS to run Docker containers at scale on its infrastructrue.
- **Hashicorp Nomad**
  Nomad is the container orchestrator provided by HashiCorp.

Why Use Container Orchestrators?

Though we can argue that containers at scale can be maintained manually, or with the help of some scripts, container orchestrators can make things easy for operators.

Container orchestrators can:

- **Bring multiple hosts together and make them part of a cluster**
- **Schedule containers to run on different hosts**
- **Help containers running on one host reach out to containers running on other hosts in the cluster**
- **Bind containers and storage**
- Bind containers of similar type to a higher-level construct, like services, so we don't have to deal with individual containers
- Keep resource usage in-check, and optimize it when necessary
- Allow secure access to applications running inside containers.

With all these **built-in benefits**, it makes sense to use container orchestrators to manage containers. In this course, we will explore **Kubernetes**.

Where to Deploy Container Orchestrators?

Most container orchestrators can be deployed on the infrastructure of our choice. We can deploy them on bare metal, VMs, on-premise, or on a cloud of our choice. For example, Kubernetes can be deployed on our laptop/workstation, inside a company's datacenter, on AWS, on OpenStack, etc. There are even one-click installers available to set up Kubernetes on the cloud, like Google Kubernetes Engine on Google Cloud, or Azure Container Service on Microsoft Azure. Similar solutions are available for other container orchestrators, as well.

There are companies that offer managed Container Orchestration as a Service. We will explore them for Kubernetes in one of the later chapters.

Learning Objectives (Review)

You should now be able to:

- Define the concept of container orchestration.
- Explain the reasons for doing container orchestration.
- Discuss different container orchestration options.
- Discuss different container orchestration deployment options.