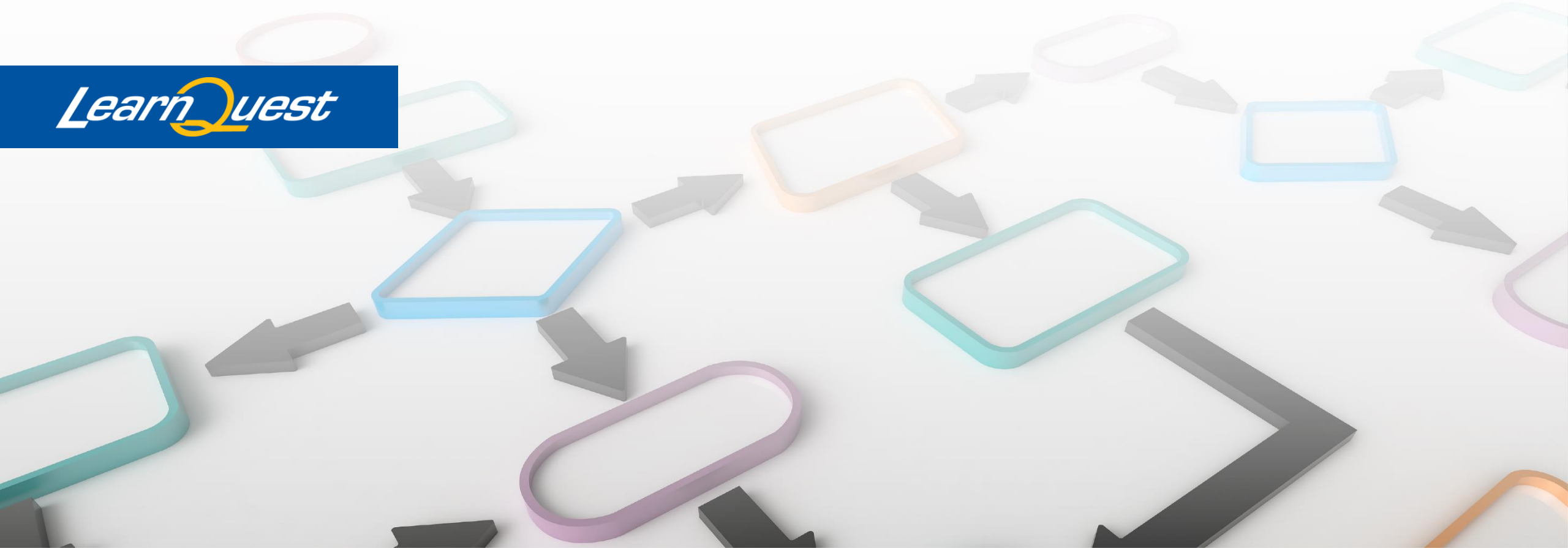
Linux Cloud and DevOps

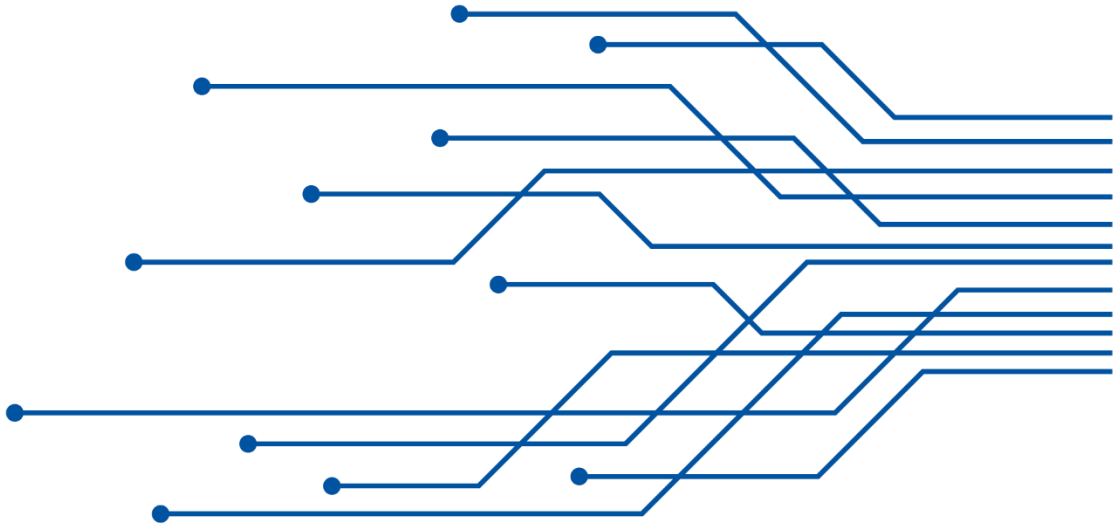
4th Course in Linux Foundations Specialization



DevOps Basics

In this module, we look at DevOps and the use of Linux and the cloud.

4



Learning Objectives

DevOps Basics

Upon completion of this module, learners will be able to:

- Describe DevOps
- Deploy Docker Containers
- Describe Orchestration
- List Container Orchestration Engines

Lesson 1

DevOps

In this lesson we look at the need and goals for DevOps

DevOps

- DevOps is the combination of cultural philosophies, practices, and tools that increase an organization's ability to deliver applications and services at higher speeds than traditional software development methodologies.

DevOps Goals

Speed - Move at a high rate so you can innovate faster, adapt to changing markets quicker, and grow more efficiently.

Rapid Delivery - Increase the frequency and pace of releases to innovate and improve the product faster.

Reliability - Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for the end-users.

Scale - Operate and manage infrastructure and development processes at scale.

Improved Collaboration - Build more effective teams under a DevOps cultural model, emphasizing ownership and accountability values.

Security - Retaining control and preserving compliance while the team moves faster.

Lesson 1 Review

A goal of DevOps is to innovate faster

Another goal of DevOps is improved collaboration

A third goal of DevOps is improved reliability

Lesson 2

Containers

In this lesson we look at the need for containers

Virtual Machines

- A virtual machine (VM) is the virtualization or emulation of a computer system
- VMs run a complete operating system–including its own kernel

Containers

- A container is an isolated, lightweight silo for running an application on the host operating system
- Containers build on top of the host operating system's kernel
- The host and container OSs must be the same

Common VM Providers

VMware
vSphere

VirtualBox

Xen

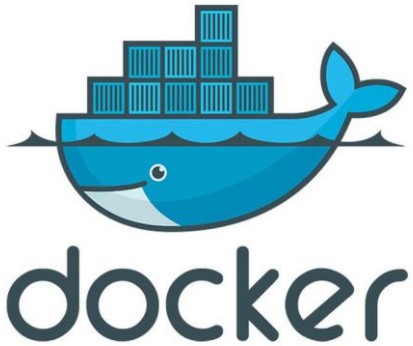
Hyper-V

KVM

Common Container Providers

Linux Containers

- LXC
- LXD
- CGManager



Docker

Windows Server Containers

Benefits of Containers for DevOps

Less Overhead
Containers are typically faster

Reproducibility
Containers are much easier to reproduce

Immutability
By default Containers are immutable which is essential for software testing

Lesson 2 Review

Each VM guest can have its own OS

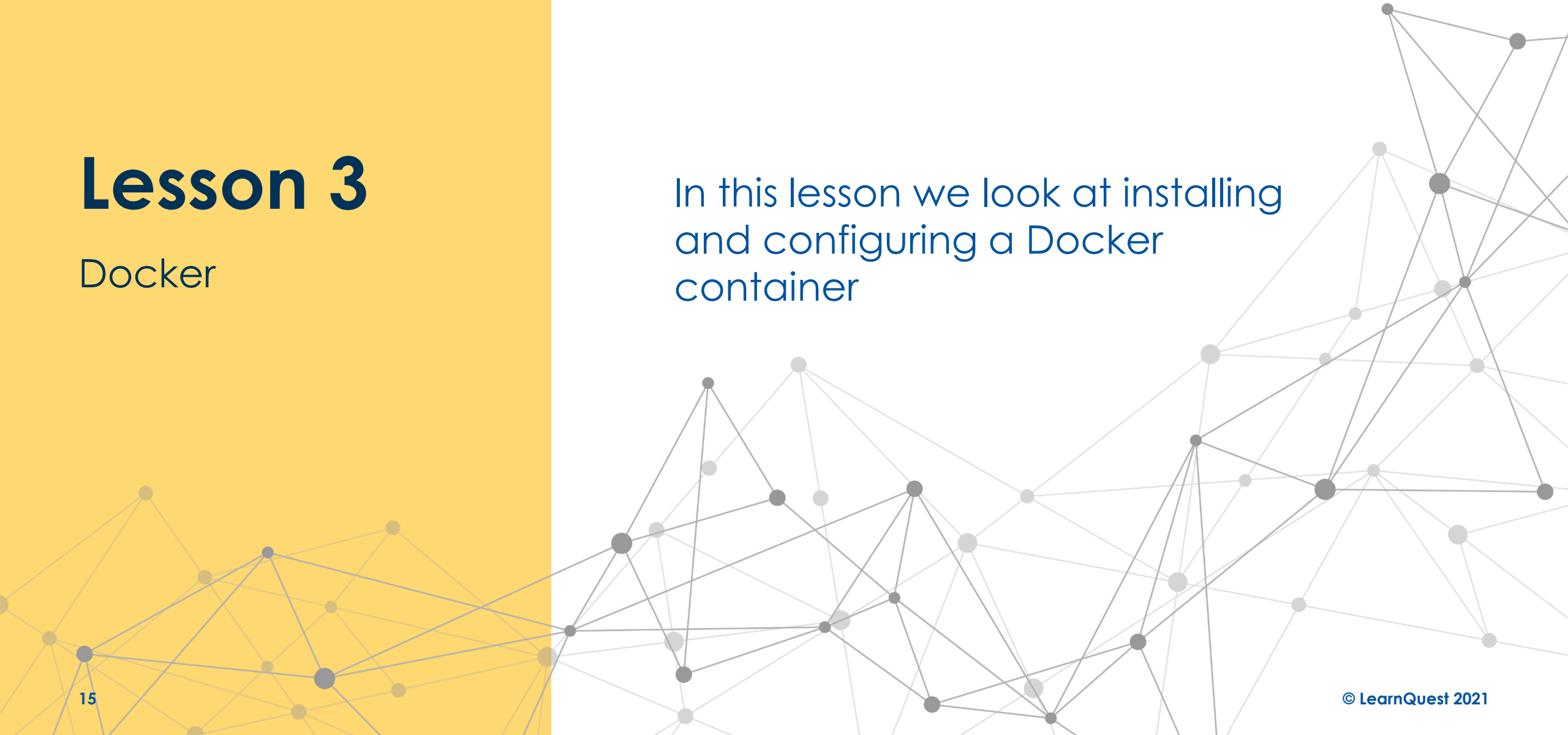
Containers share an OS

Containers are immutable by default

Lesson 3

Docker

In this lesson we look at installing and configuring a Docker container

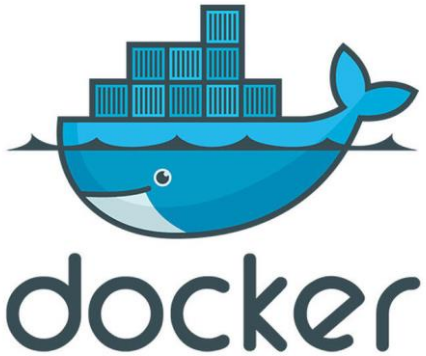


Installing Docker

- Download Docker Desktop for Mac: https://desktop.docker.com/mac/stable/Docker.dmg
- Download Docker Desktop for Windows: https://desktop.docker.com/win/stable/Docker%20Desktop%20Installer.exe
- Install Docker Engine on Linux: https://docs.docker.com/engine/install/

Dockerfile

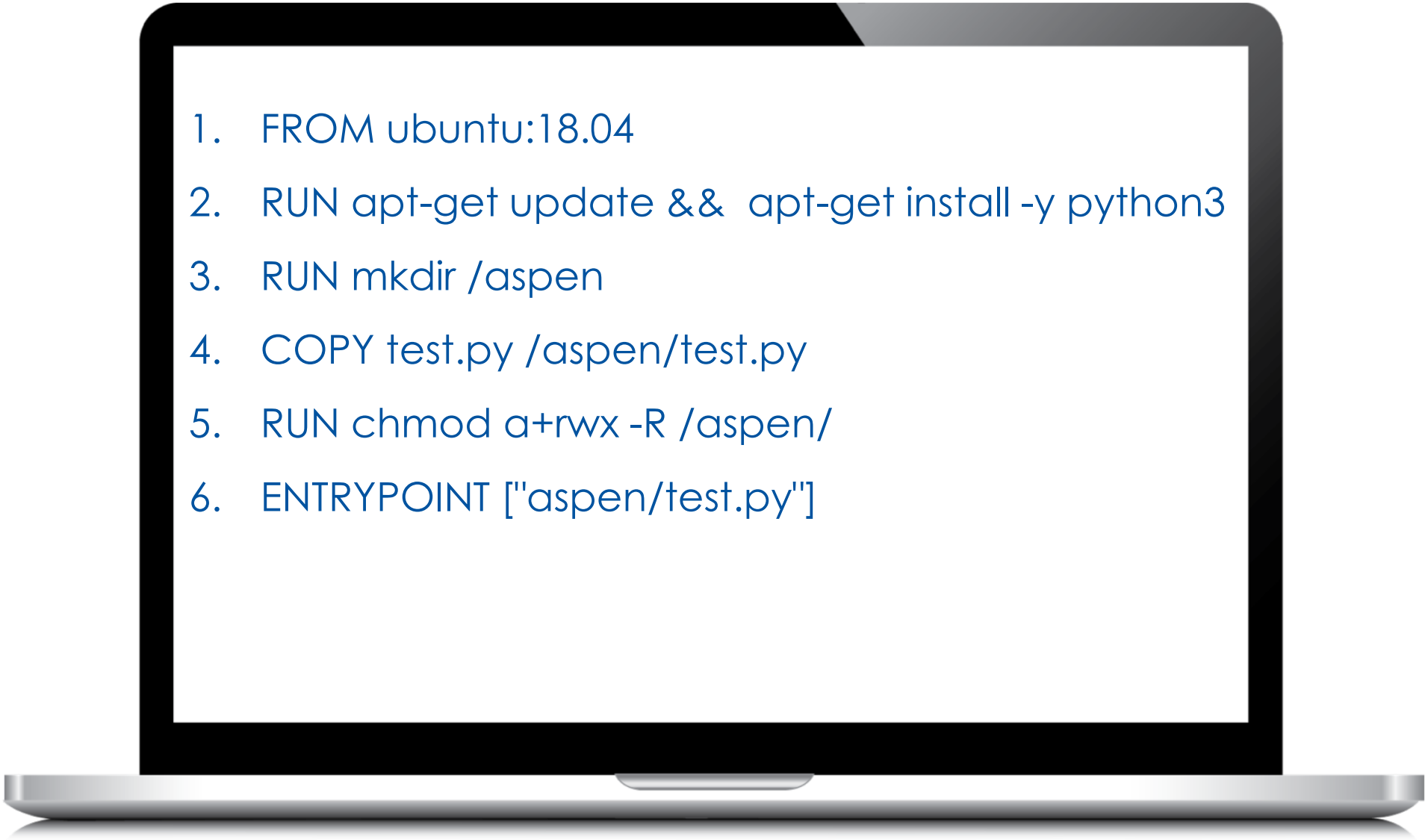
- A Dockerfile is simply a text-based script of instructions that is used to create a container image
- The file is named DockerFile
- A Dockerfile is a step by step set of instructions
- Docker provides a set of standard instructions to be used in the Dockerfile, like FROM, COPY, RUN, ENV, EXPOSE, CMD just to name a few basic ones
- Docker will build a Docker image automatically by reading these instructions from the Dockerfile



Docker Hub

- Docker Hub is a service provided by Docker for finding and sharing container images with your team.
- It is the world's largest repository of container images with an array of content sources including container community developers, open-source projects and independent software vendors (ISV) building and distributing their code in containers.
- Users get access to free public repositories for storing and sharing images or can choose a subscription plan for private repositories.

Example Dockerfile



Docker Command Line

- To build the image: docker build –t [image name]
- To show images: docker Images
- To run an image: docker run [image name]

Lesson 3 Review

A Dockerfile is simply a text-based script of instructions that is used to create a container image

The file is named DockerFile

You build a DockerFile into an Image

Lesson 4

Orchestration

In this lesson we look at Orchestration

Continual Software Revision Control

Continuous integration = Quickly integrating app changes into main software

Continuous testing = App modifications undergo automated testing to avoid breaking app

Continuous delivery = Software is delivered to customer on a continual basis

Fixing State Across Environments

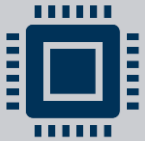
Production environment must equal development environment

Environment modifications must be controlled in similar manner to software modifications

Tested new environments added to a journal and old environments maintained for rollbacks

Standardize Environments

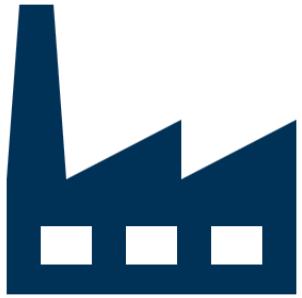
Configuration management = Nonhardware specs implemented into environment via automated code



Policy as code = Security measures (e.g., firewall ACLs) and authentication policies implemented into environment via automated code

Infrastructure as code = Umbrella term that includes configuration management and policy as code

Application Deployment



Application and development environment are moved into production in a continual manner

This process is automated via infrastructure automation

Monitoring Environment

App production environment needs monitoring and logging of various items, such as:

- Software metrics
- Infrastructure resource usage
- Performance statistics

Monitoring also provides alerts. For example:

- Failures
- Resource depletion events

Containers Provide

- Static Environment
- Version Control
- Replace Not Update
- High Availability

Coding the Infrastructure

Determine the required infrastructure

Document the required infrastructure

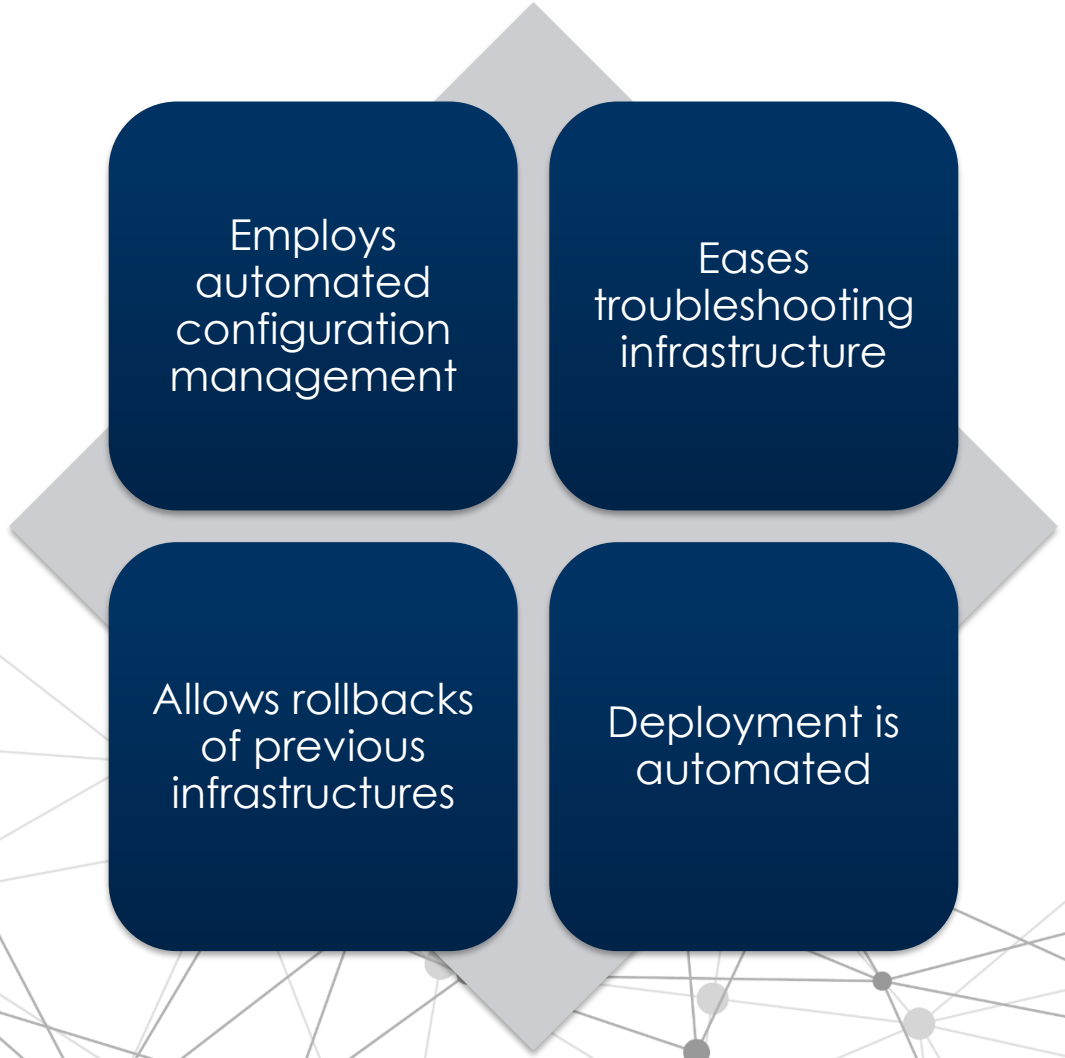
Automated configuration management = Process of recording infrastructure as code settings

Build automation = Using the infrastructure as code data to automate the replicating and deployment of app containers

Provide revision control for infrastructure as code data

Troubleshoot the infrastructure

Automating Infrastructure



Agent vs Agentless Monitoring

Agent monitoring tools - Orchestration utilities that require software to be installed in the app container being monitored.

Agentless monitoring tools - Orchestration utilities that do not require software to be installed in the app container being monitored. Preexisting tools within the container are utilized instead.

Orchestration Systems

Kubernetes

- Designed by Google
- Open-source orchestration system
- Platform for automating deployment, scaling, and operations of application containers across clusters of hosts
- Supported Docker containers in first version, and later added the rkt container engine

Docker Swarm

- Docker container engine's orchestration tool
- Creates a groups of Docker containers (cluster)
- Monitor cluster's health
- Return cluster to a desired state automatically

Lesson 4 Review

Containers provide many of the components needed in DevOps

Agents run a piece of software to monitor the containers

Kubernetes is a popular orchestration system