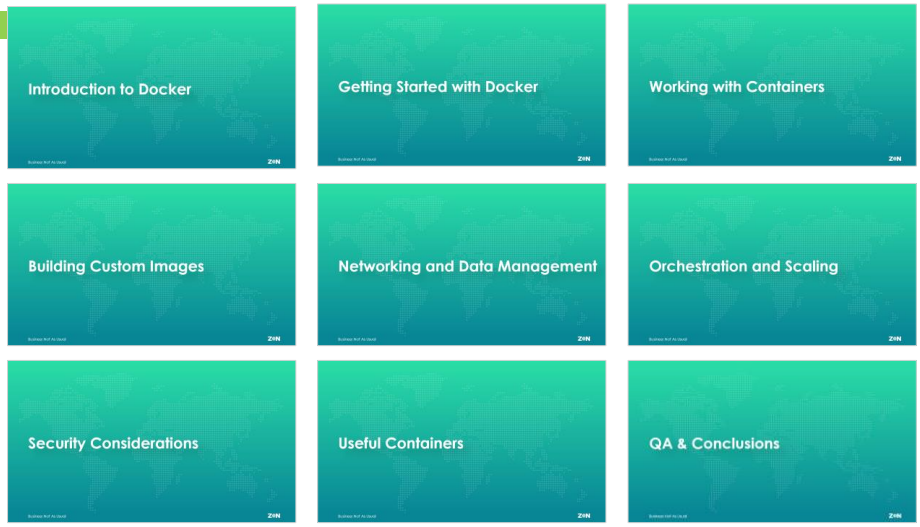
Latihan Teknikal Docker Sistem Pinjaman Penuntut (ePPN)

Training Outline

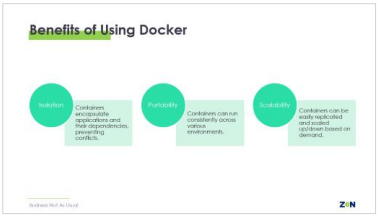


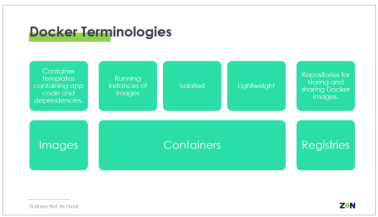
Business Not As Usual

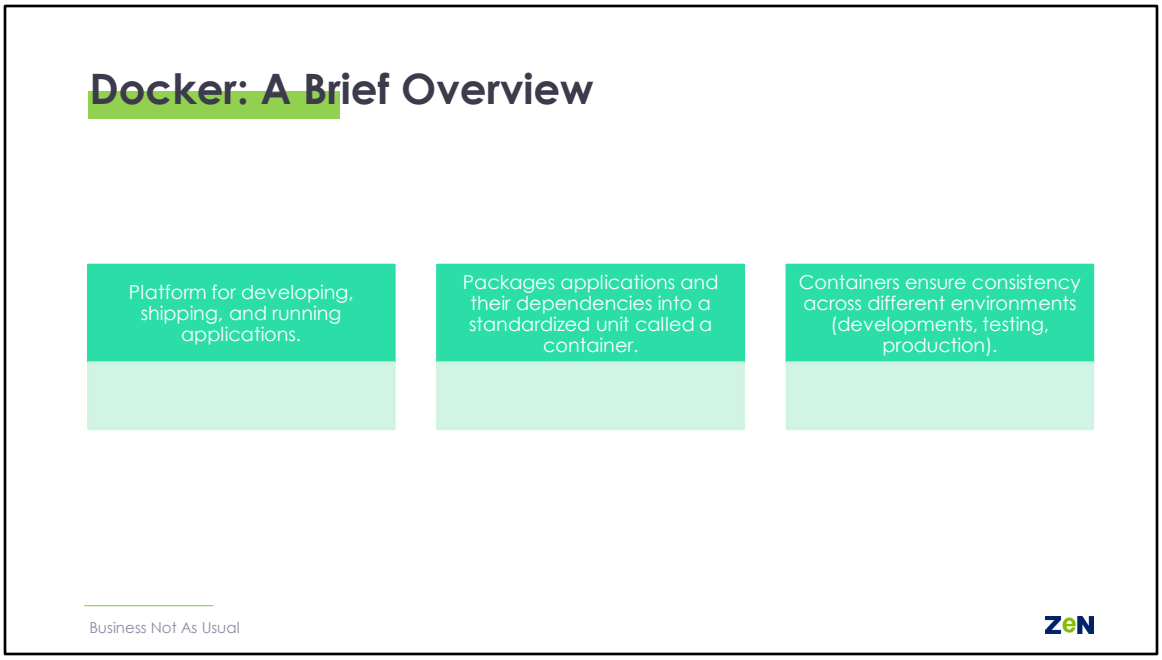
ZeN


Introduction to Docker
Business Not As Usual
ZeN

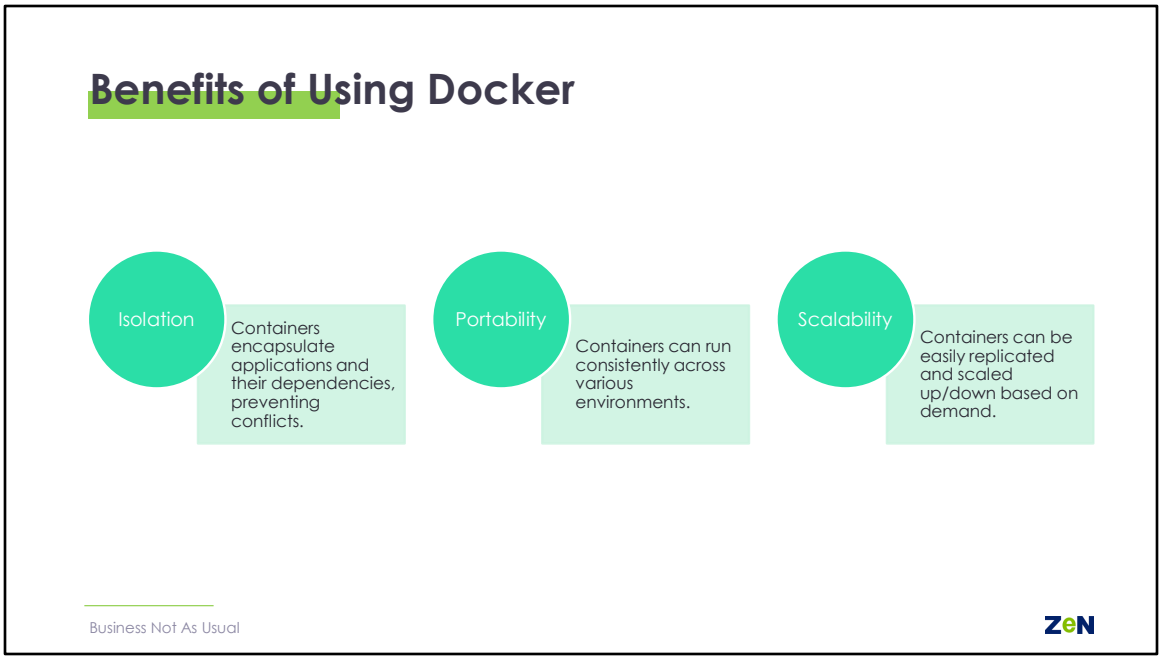
Introduction to Docker



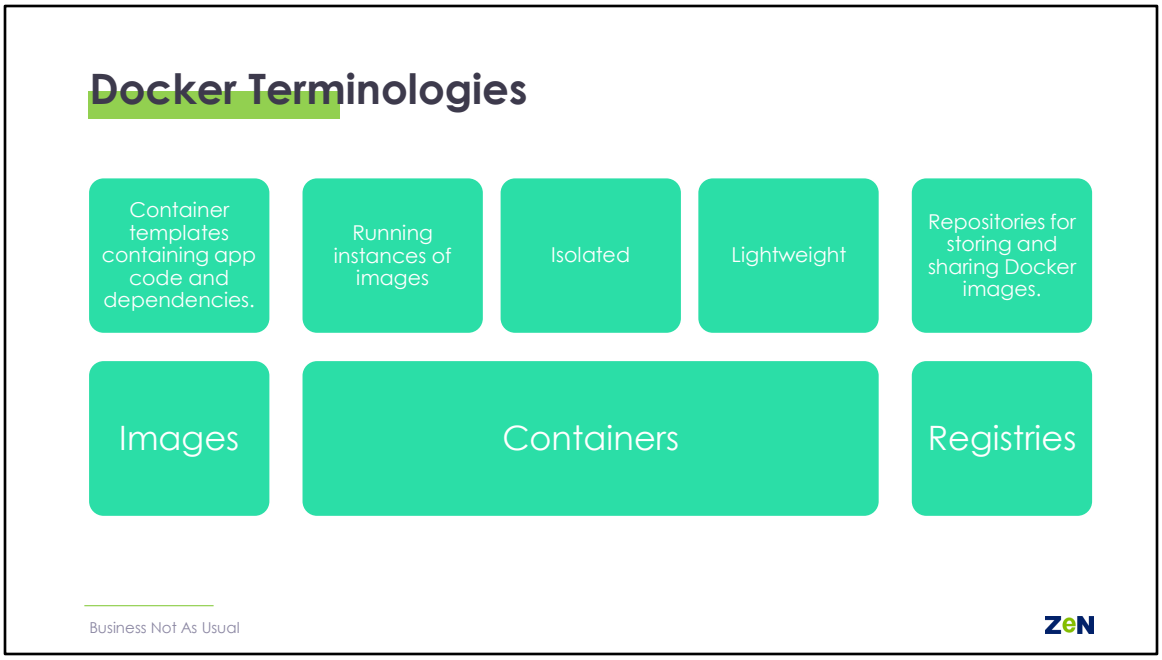




- Docker has revolutionized the software development process by enabling developers to bundle applications and their dependencies into isolated containers. These containers can then be easily moved between different environments without causing compatibility issues.



- Docker provides several benefits that simplify the application lifecycle. Containers offer isolation, ensuring that each application runs independently without affecting others. Portability ensures that the same containerized application runs consistently across different environments. Scalability is simplified with Docker, as you can quickly scale containers up or down to handle changing workloads.



- Docker introduces some important terminology to understand:
  - **Images:** These are like blueprints or templates for containers. An image includes the application code, runtime, libraries, and other dependencies.
  - **Containers:** When you run an image, it becomes a container. Containers are isolated instances that share the host OS's kernel but have their own filesystem, processes, and network.
  - **Registries:** These are repositories for Docker images. You can think of them as libraries where you store, share, and retrieve images. Docker Hub is a popular example.


Getting Started with Docker
Business Not As Usual
ZeN

Getting Started with Docker

Installing Docker

Docker is available for various OS

Where to get Docker?

Which version is suitable for you?

Windows

macOS

Linux

Basic Docker Commands

`docker run`	• Launches a container from an image
`docker ps`	• Lists running containers
`docker images`	• List available images

Installing Docker

Docker is available for various OS

Where to get Docker?

Which version is suitable for you?

Windows

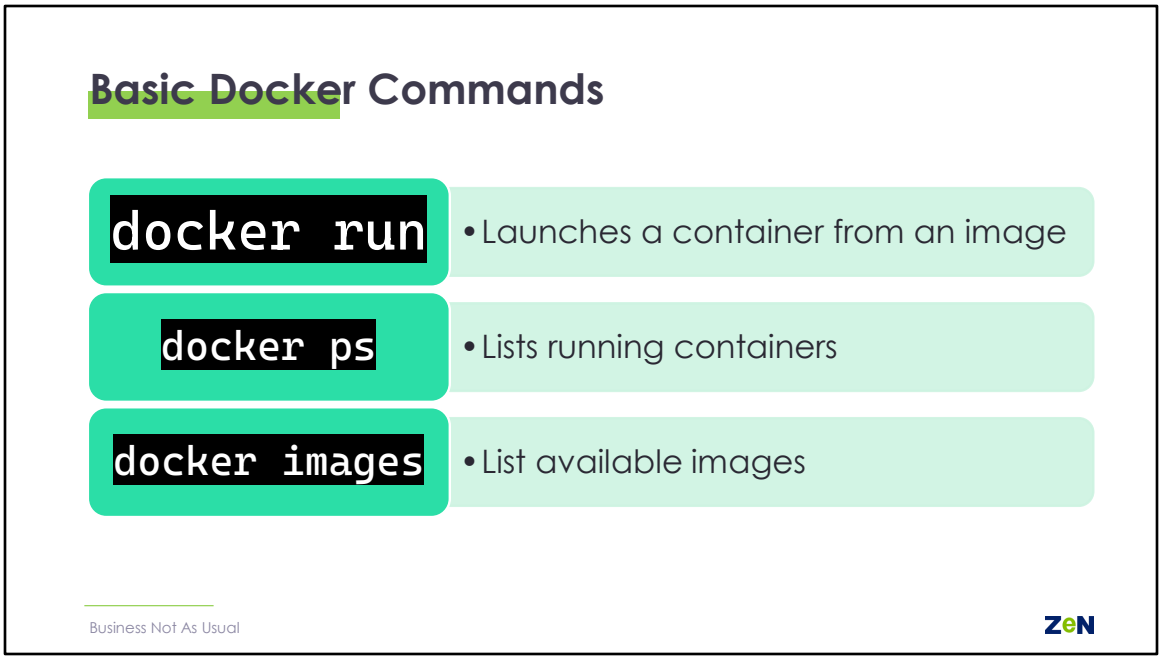
macOS

Linux

Business Not As Usual

ZeN

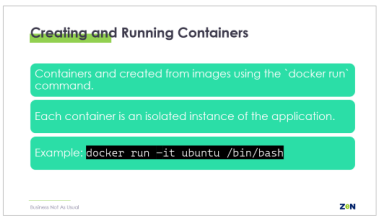
- Installing Docker is the first step to start using containers.
- Docker offers versions for Windows, macOS, and Linux.
- Let's take a moment to guide you through the installation process based on your operating system.
- Please refer to the provided download links and follow the step-by-step instructions.

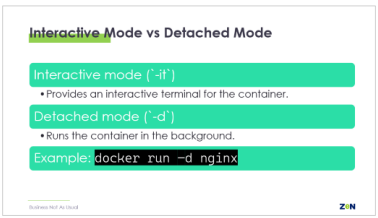


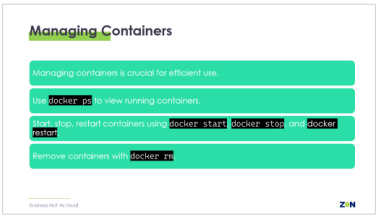
- Now that you have Docker installed, let's look at some fundamental commands.
- The **`docker run`** command is used to start a container from an image. This is where the magic happens.
- **`docker ps`** helps you see which containers are currently running.
- Use **`docker images`** to see the images available on your system.


Working with Containers
Business Not As Usual
ZeN

Working with Containers







Creating and Running Containers

Containers and created from images using the `docker run` command.

Each container is an isolated instance of the application.

Example: `docker run -it ubuntu /bin/bash`

Business Not As Usual

ZeN

- Once you have an image, you can create and run a container using the **`docker run`** command.
- Containers are isolated environments that run your application.
- Think of them as virtual machines, but much lighter.
- For example, you can use **`docker run -it ubuntu /bin/bash`** to start an interactive Ubuntu container.

Interactive Mode vs Detached Mode

Interactive mode (`-it`)

- Provides an interactive terminal for the container.

Detached mode (`-d`)

- Runs the container in the background.

Example: `docker run -d nginx`

Business Not As Usual

ZeN

- You can run containers interactively, which is useful for debugging or executing commands within the container.
- This is achieved using the **`-it`** flag with the **`docker run`** command.
- Conversely, you can run containers in detached mode using the **`-d`** flag.
- This is ideal for running applications that don't require constant interaction.
- For instance, **`docker run -d nginx`** starts an Nginx web server in the background.

Managing Containers

Managing containers is crucial for efficient use.

Use `docker ps` to view running containers.

Start, stop, restart containers using `docker start`, `docker stop`, and `docker restart`.

Remove containers with `docker rm`.

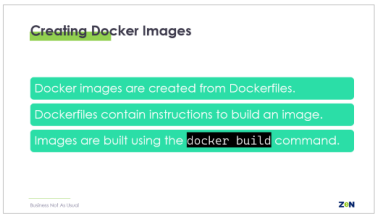
Business Not As Usual

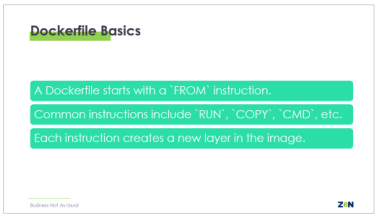
ZeN

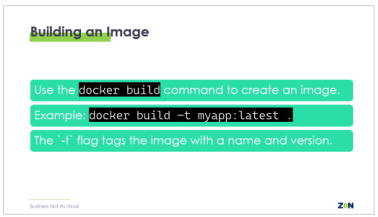
- Beyond just creating and running containers, managing them is equally important.
- You can use various commands to control containers:
  - **`docker ps`**: Lists running containers and provides basic information.
  - **`docker start <container_id>`**: Starts a stopped container.
  - **`docker stop <container_id>`**: Stops a running container.
  - **`docker restart <container_id>`**: Restarts a running container.
  - **`docker rm <container_id>`**: Removes a stopped container.


Building Custom Images
Business Not As Usual
ZeN

Building Custom Images







Creating Docker Images

Docker images are created from Dockerfiles.

Dockerfiles contain instructions to build an image.

Images are built using the `docker build` command.

Business Not As Usual

ZeN

- Building custom Docker images is a powerful feature.
- Images are defined by Dockerfiles, which are text files containing instructions to assemble the image.
- Once the Dockerfile is ready, you use the **`docker build`** command to create the image.

Dockerfile Basics

A Dockerfile starts with a `FROM` instruction.

Common instructions include `RUN`, `COPY`, `CMD`, etc.

Each instruction creates a new layer in the image.

Business Not As Usual

ZeN

- Dockerfiles begin with a **`FROM`** instruction that specifies the base image.
- From there, you can add other instructions like **`RUN`** for executing commands, **`COPY`** for copying files, and **`CMD`** for defining the default command when the container starts.
- Each instruction creates a new layer in the image, allowing for efficient caching and reusability.

Building an Image

Use the `docker build` command to create an image.

Example: `docker build -t myapp:latest .`

The `-t` flag tags the image with a name and version.

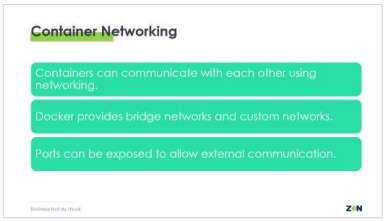
Business Not As Usual

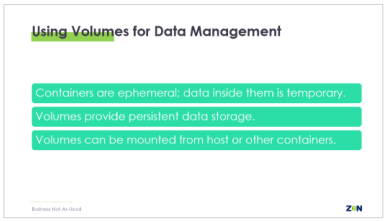
ZeN

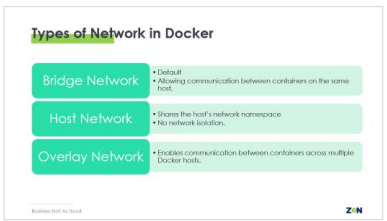
- Now, let's see how to actually build a custom image.
- The d**ocker build** command is used, and you can provide a tag using the *-t* flag.
- For example, `***docker build -t myapp:latest .***` builds an image named *myapp* with the tag *latest* using the current directory (.) as the build context.

Networking and Data Management

Networking and Data Management







Container Networking

Containers can communicate with each other using networking.

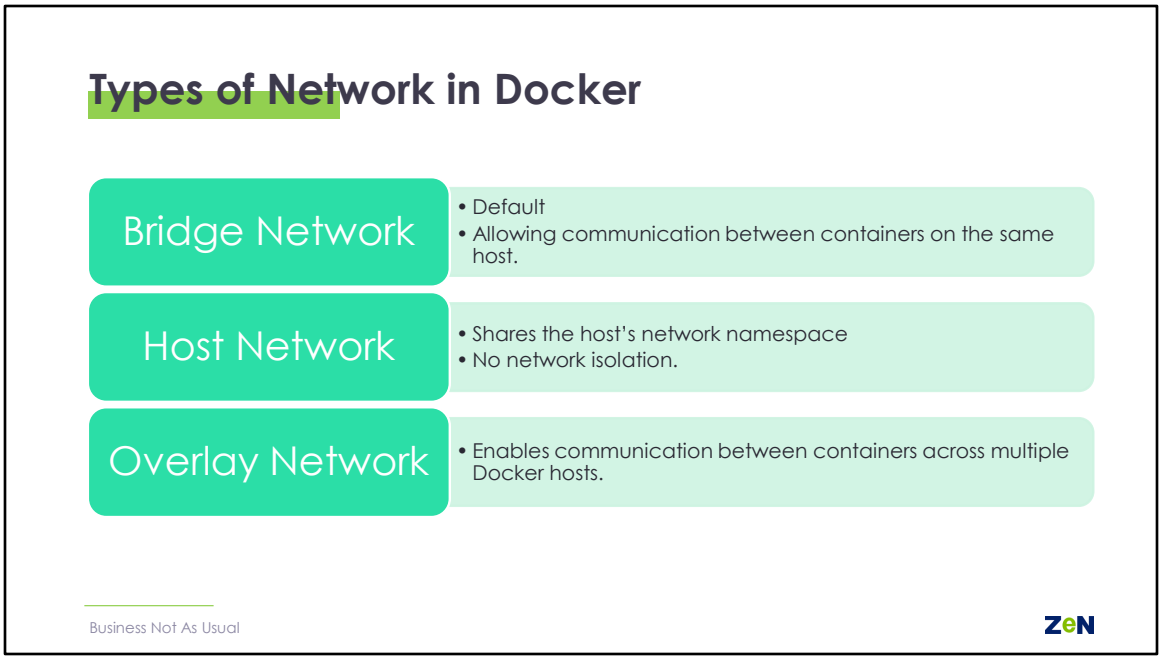
Docker provides bridge networks and custom networks.

Ports can be exposed to allow external communication.

Business Not As Usual

ZeN

- Containers can communicate with each other within a Docker network.
- Docker offers default bridge networks and allows you to create custom networks.
- Exposing ports enables external access to containers.
- This networking flexibility allows for creating complex application architectures.



•In Docker, you can choose from different network modes to facilitate communication between containers. Here are three main types:

- **•Bridge Network:** This is the default network mode. Containers within the same bridge network can communicate, but they're isolated from containers in other bridge networks.
- **•Host Network:** Containers share the host's network namespace, making them visible on the host's network.
- **•Overlay Network:** Useful for multi-host setups, it enables containers from different hosts to communicate as if they're on the same network.

Using Volumes for Data Management

Containers are ephemeral; data inside them is temporary.

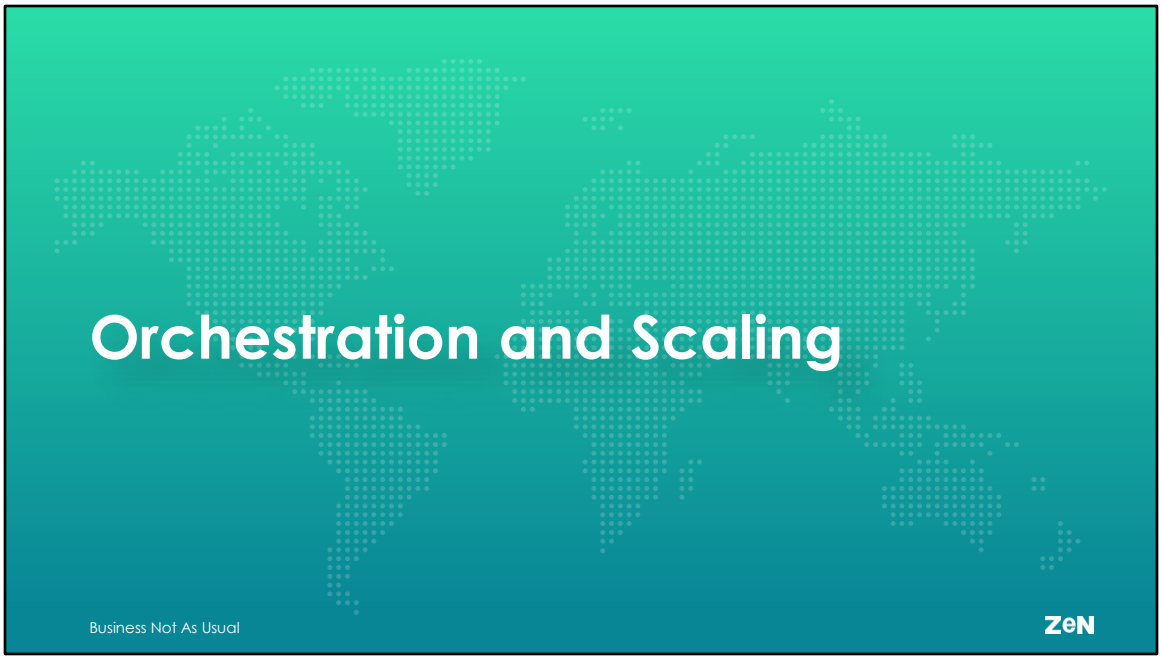
Volumes provide persistent data storage.

Volumes can be mounted from host or other containers.

Business Not As Usual

ZeN

- Containers are meant to be stateless and disposable.
- Any data stored inside a container is temporary.
- Volumes, on the other hand, offer a way to store data persistently even if containers are removed.
- Volumes can be mounted from the host system or even shared between containers.


Orchestration and Scaling
Business Not As Usual
ZeN

Orchestration and Scaling

Docker Compose: Simplifying Multi-Container Applications

- Docker Compose is a tool for defining and running multi-container applications.
- It uses a YAML file to define services, networks, and volumes.
- `docker compose up` starts the defined services.

Running Multiple Services Using Docker Compose

- Use `docker compose up` to start services defined in the Compose file.
- Services are launched in the order they are listed.
- Use `docker compose down` to stop and remove the services.

Brief overview of Docker Swarm and Kubernetes

- Docker Swarm and Kubernetes are orchestration platforms.
- They help manage and scale containerized applications.
- Docker Swarm is simpler, while Kubernetes is more feature-rich and complex.

Docker Compose: Simplifying Multi-Container Applications

Docker Compose is a tool for defining and running multi-container applications.

It uses a YAML file to define services, networks, and volumes.

`docker compose up` starts the defined services.

Business Not As Usual

ZeN

- Managing multi-container applications manually can be complex.
- Docker Compose simplifies this by allowing you to define services, networks, and volumes in a single YAML file.
- Using **`docker-compose up`**, you can start all the defined services together, making development and testing easier.

Running Multiple Services Using Docker Compose

Use `docker compose up` to start services defined in the Compose file.

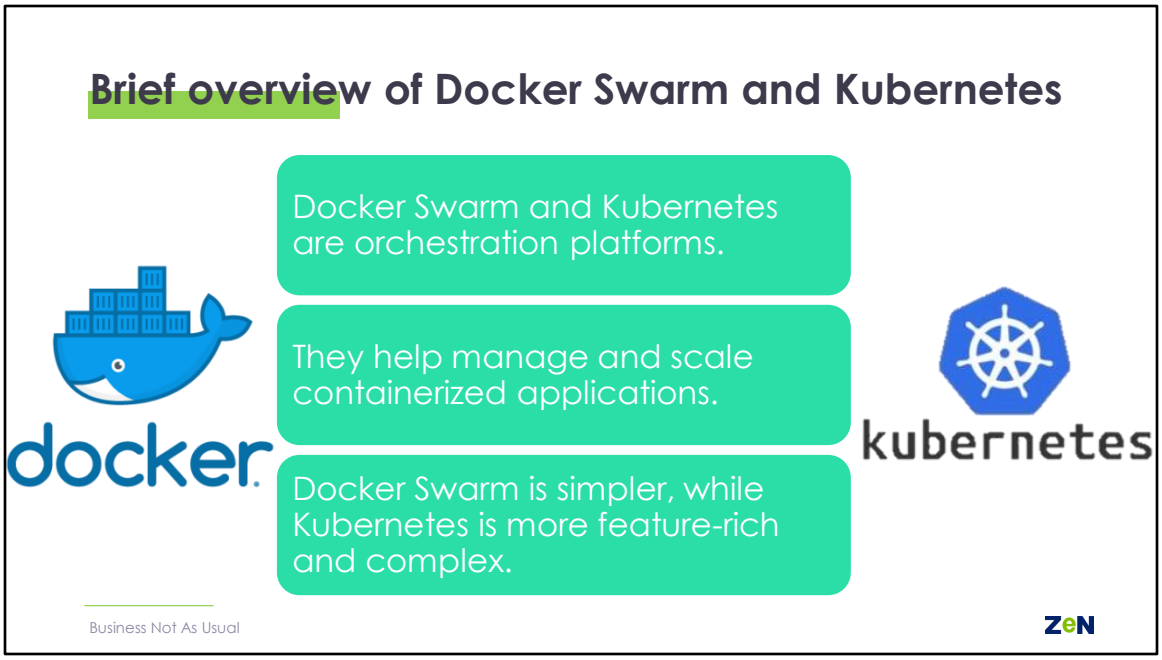
Services are launched in the order they are listed.

Use `docker compose down` to stop and remove the services.

Business Not As Usual

ZeN

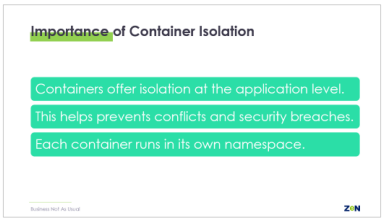
- After defining services in the Compose file, you can use **`docker-compose up`** to start all the services together.
- Compose ensures that dependencies between services are maintained.
- You can use **`docker-compose down`** to stop and remove the services when you're done.

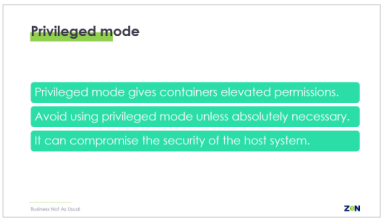


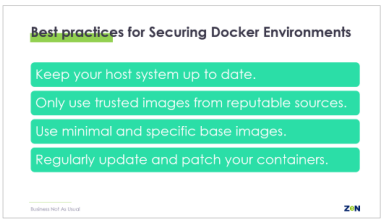
- As your application grows, you might need more advanced orchestration tools.
- Docker Swarm and Kubernetes are popular options.
- Docker Swarm is built into Docker and provides simpler orchestration.
- Kubernetes is more complex but offers powerful features for managing large, distributed applications.


Security Considerations
Business Not As Usual
ZeN

Security Considerations







Importance of Container Isolation

Containers offer isolation at the application level.

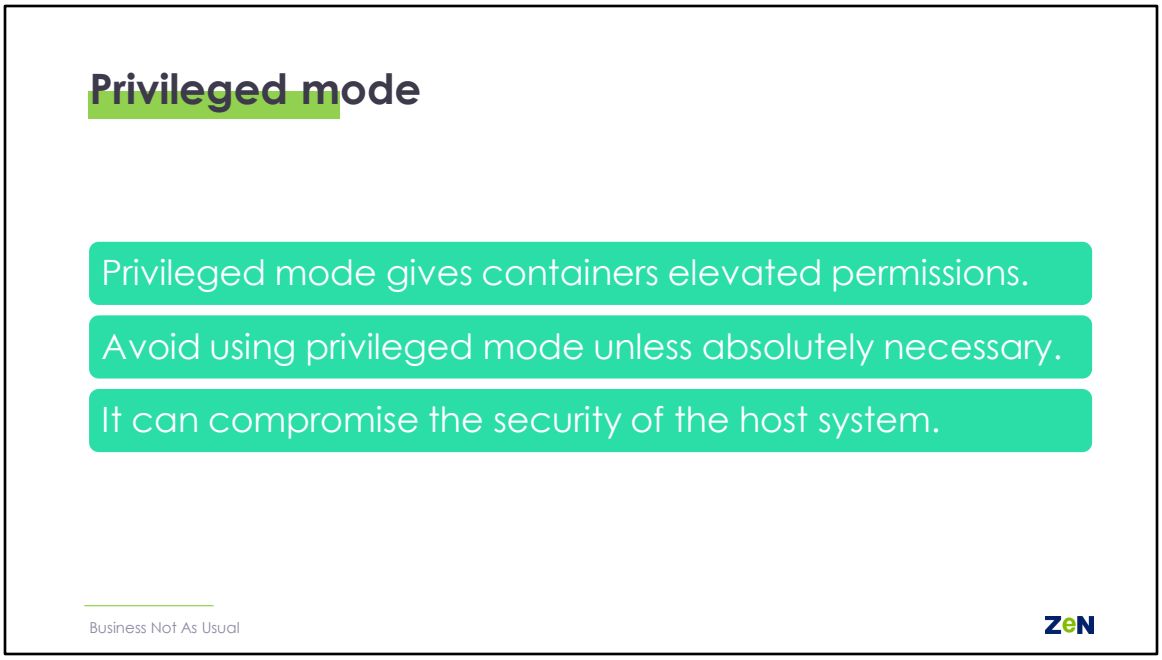
This helps prevents conflicts and security breaches.

Each container runs in its own namespace.

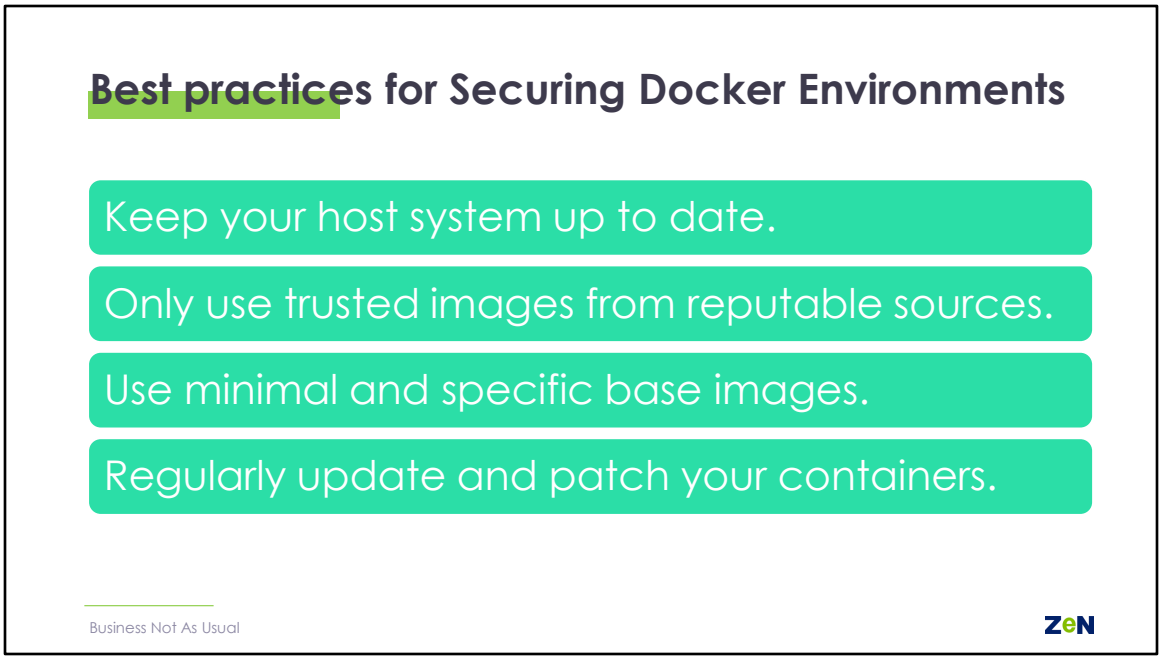
Business Not As Usual

ZeN

- One of the key advantages of using containers is the isolation they provide.
- Each container operates in its own environment, isolated from others.
- This isolation helps prevent conflicts between applications and limits the impact of potential security vulnerabilities.



- Docker allows you to run containers in privileged mode, which provides containers with elevated permissions and access to the host system's devices.
- However, using privileged mode should be avoided unless it's absolutely necessary, as it can undermine the security boundaries between containers and the host system.



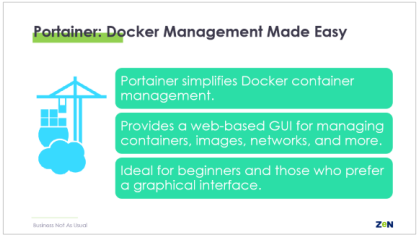
- Securing Docker environments requires a combination of practices.
- Regularly updating your host system and using trusted images are fundamental.
- Using minimal base images reduces the attack surface.
- Additionally, keeping containers updated with patches is crucial to address security vulnerabilities.


Useful Containers
Business Not As Usual
ZeN

Useful Containers

Enhancing Docker experience with additional containers

Introduction to Portainer and Nginx Proxy Manager



Portainer: Docker Management Made Easy

Portainer simplifies Docker container management.

Provides a web-based GUI for managing containers, images, networks, and more.

Ideal for beginners and those who prefer a graphical interface.

- While the command-line interface is powerful, some users prefer a graphical interface.
- Portainer is a tool that provides a user-friendly web-based GUI for managing Docker containers, images, networks, and more.
- It's particularly helpful for newcomers to Docker and those who prefer a visual approach.

Nginx Proxy Manager: Simplifying Reverse Proxy Configuration

Nginx Proxy Manager makes setting up reverse proxies easy.

Simplifies the process of routing traffic to different services based on domain names.

Useful for hosting multiple services on a single server.

Business Not As Usual

ZeN

- Hosting multiple services on a single server can be complex, especially when it comes to managing reverse proxies.
- Nginx Proxy Manager simplifies this by providing a user-friendly interface for configuring reverse proxies.
- This enables you to route traffic to different services based on domain names, making it easier to manage and access multiple services on one server.

QA & Conclusions

Recommended Resources

- Docker Documentation: https://docs.docker.com
- Docker Compose Documentation: https://docs.docker.com/compose
- Docker Swarm Documentation: https://docs.docker.com/engine/swarm
- Kubernetes Documentation: https://kubernetes.io/docs

- To recap, we've covered the fundamentals of Docker, from introduction and basic commands to creating images, networking, security, and even scaling with Docker Compose.
- Remember, these are just the building blocks.
- Docker offers extensive documentation that provides in-depth information about each topic.
- For those interested in orchestration, Kubernetes is a powerful tool with its own comprehensive documentation.

Reach out to me


khairulnizam@zen.com.my


(+60)16-4461846