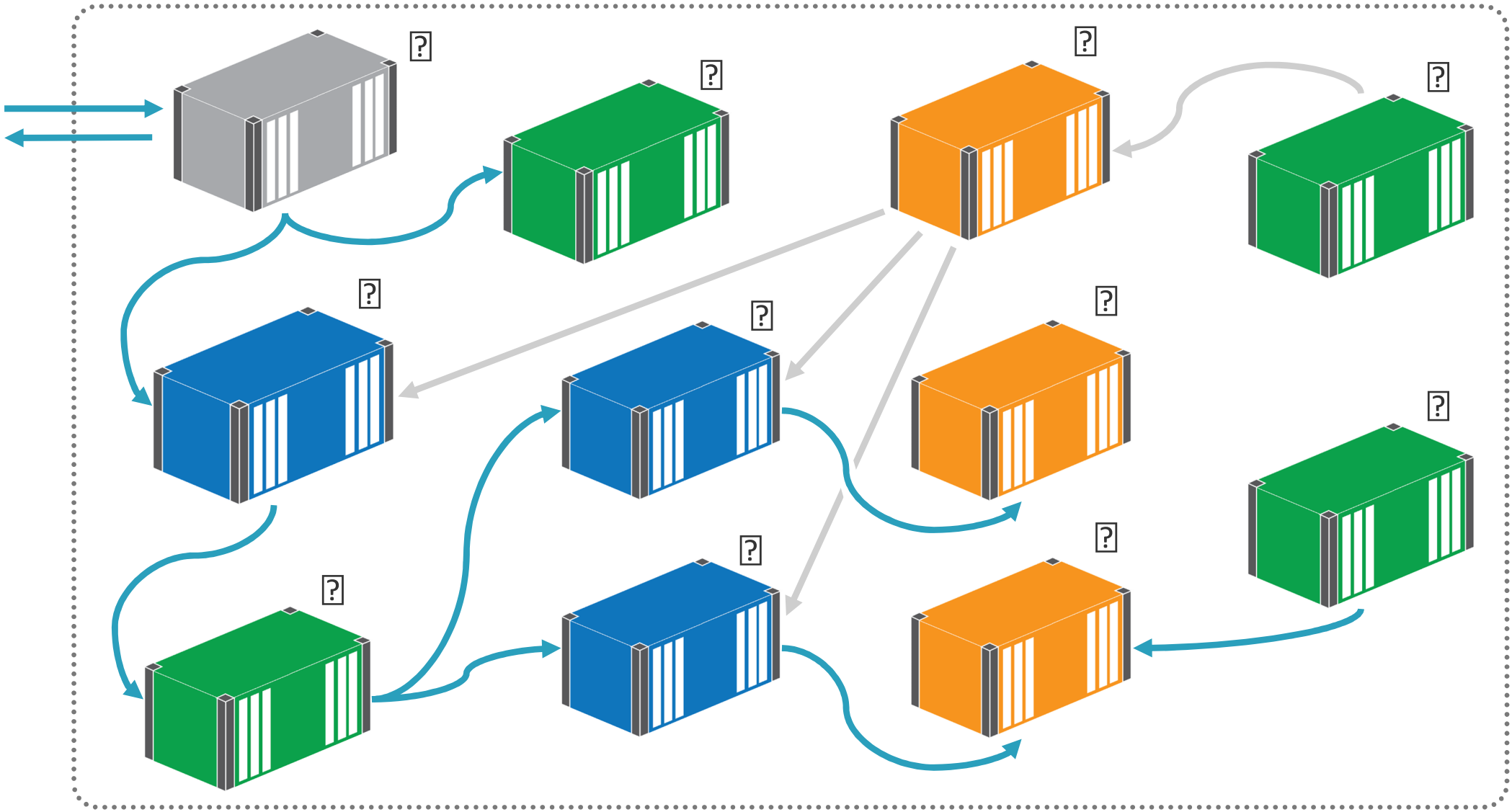
Understanding the Path to Production

Elton Stoneman
DEVELOPER ADVOCATE

@EltonStoneman https://blog.sixeyed.com



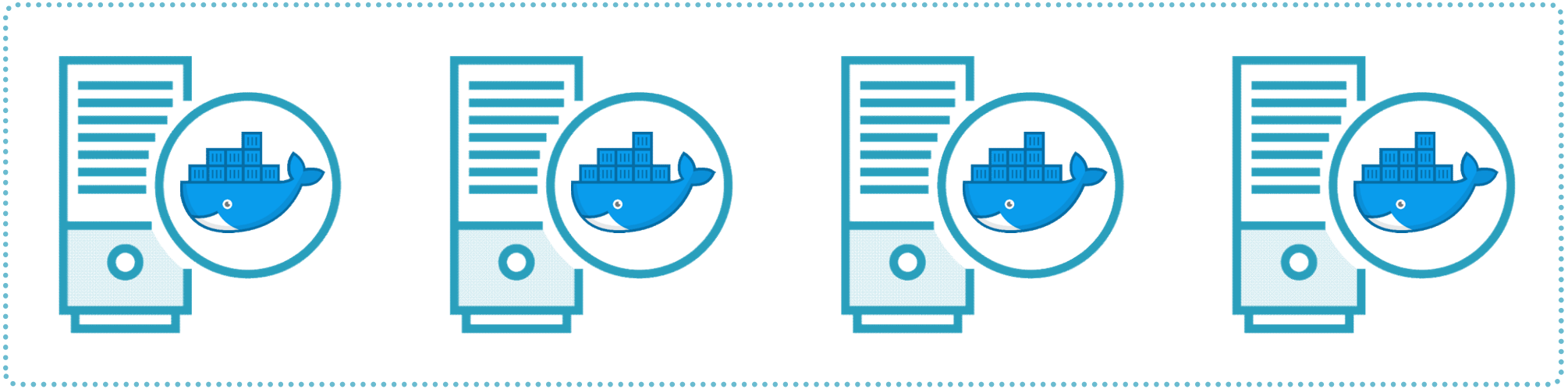
Before Production

- High availability
- Security
- The support cycle

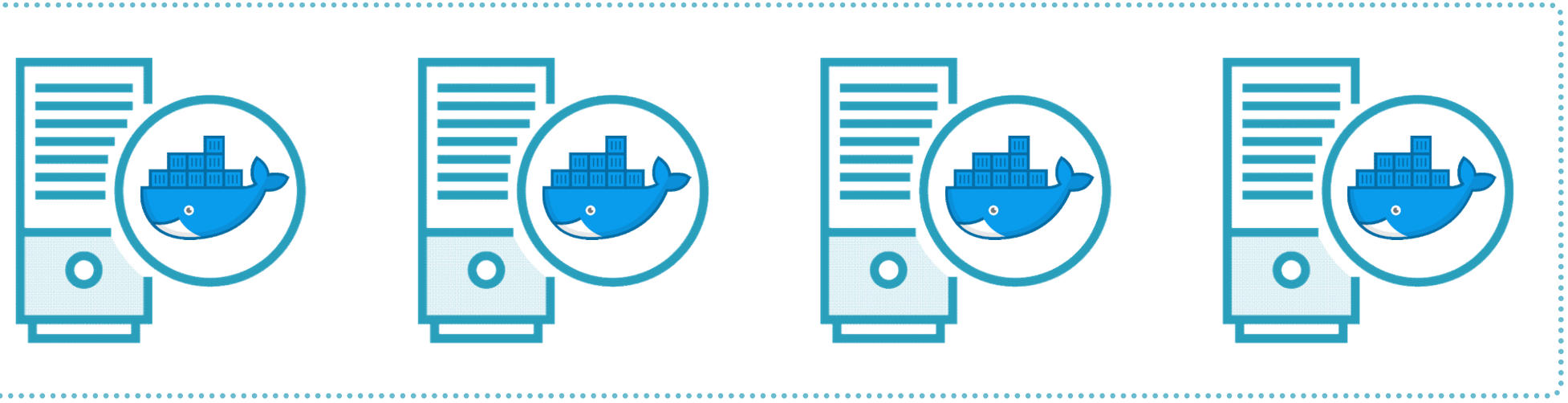
High Availability
Self-healing apps

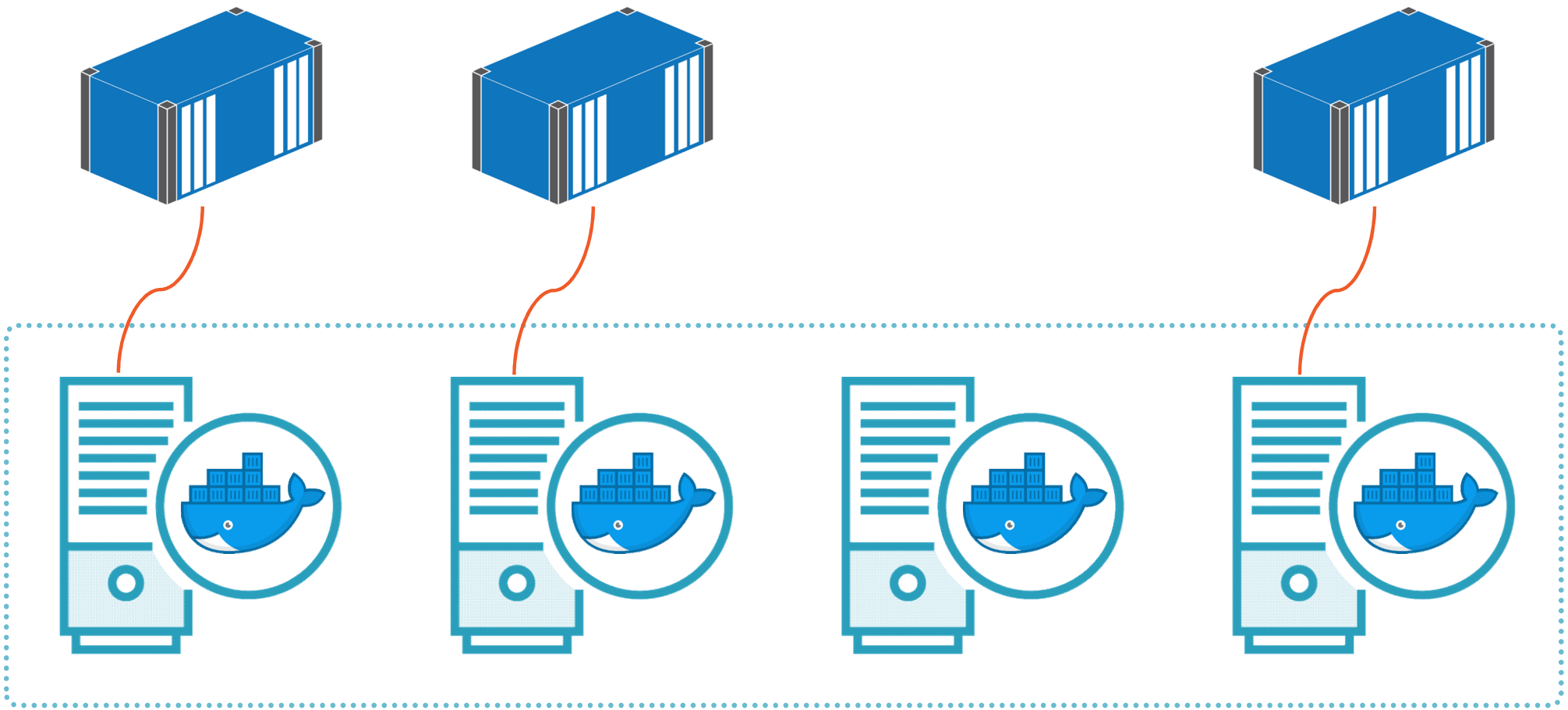
Security
Secure secret management

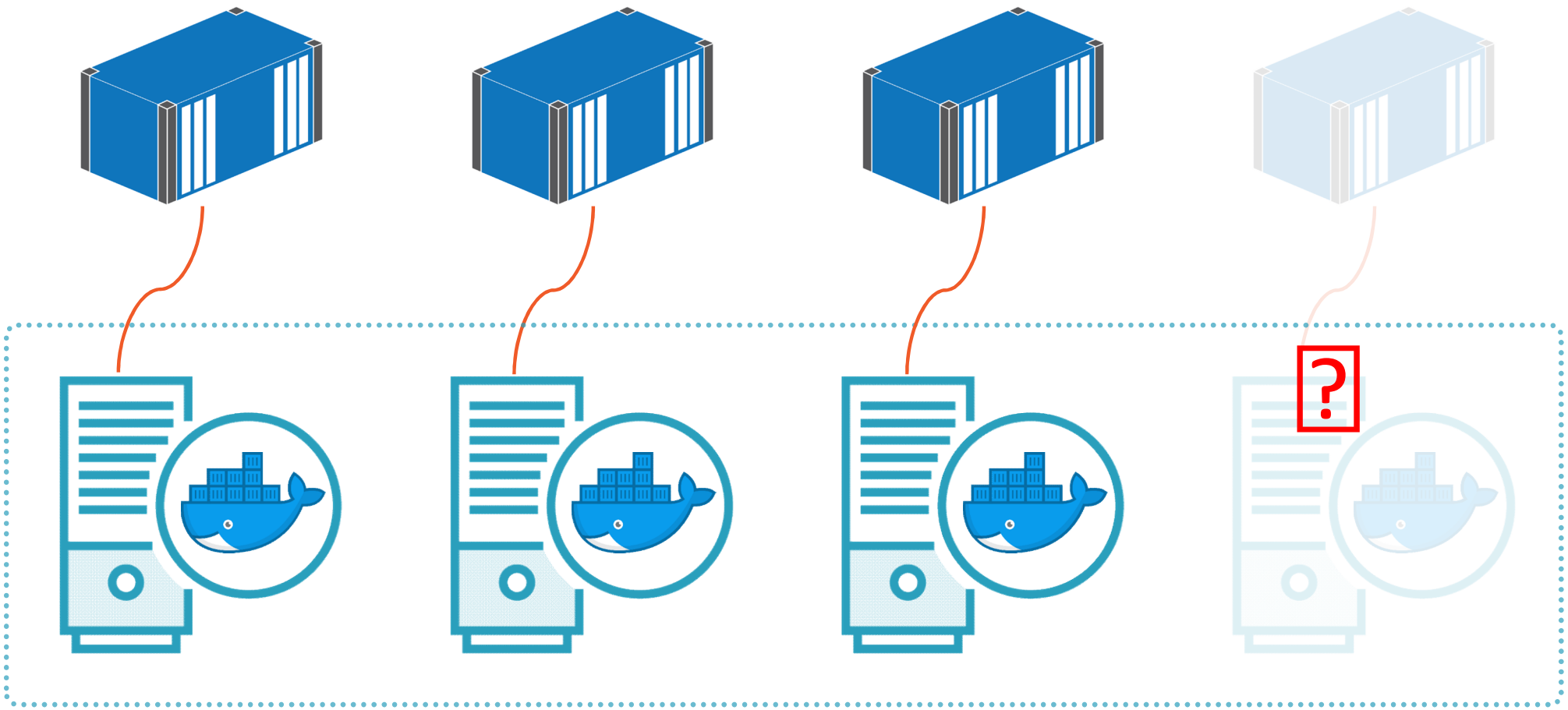
Automated Updates
And rollback

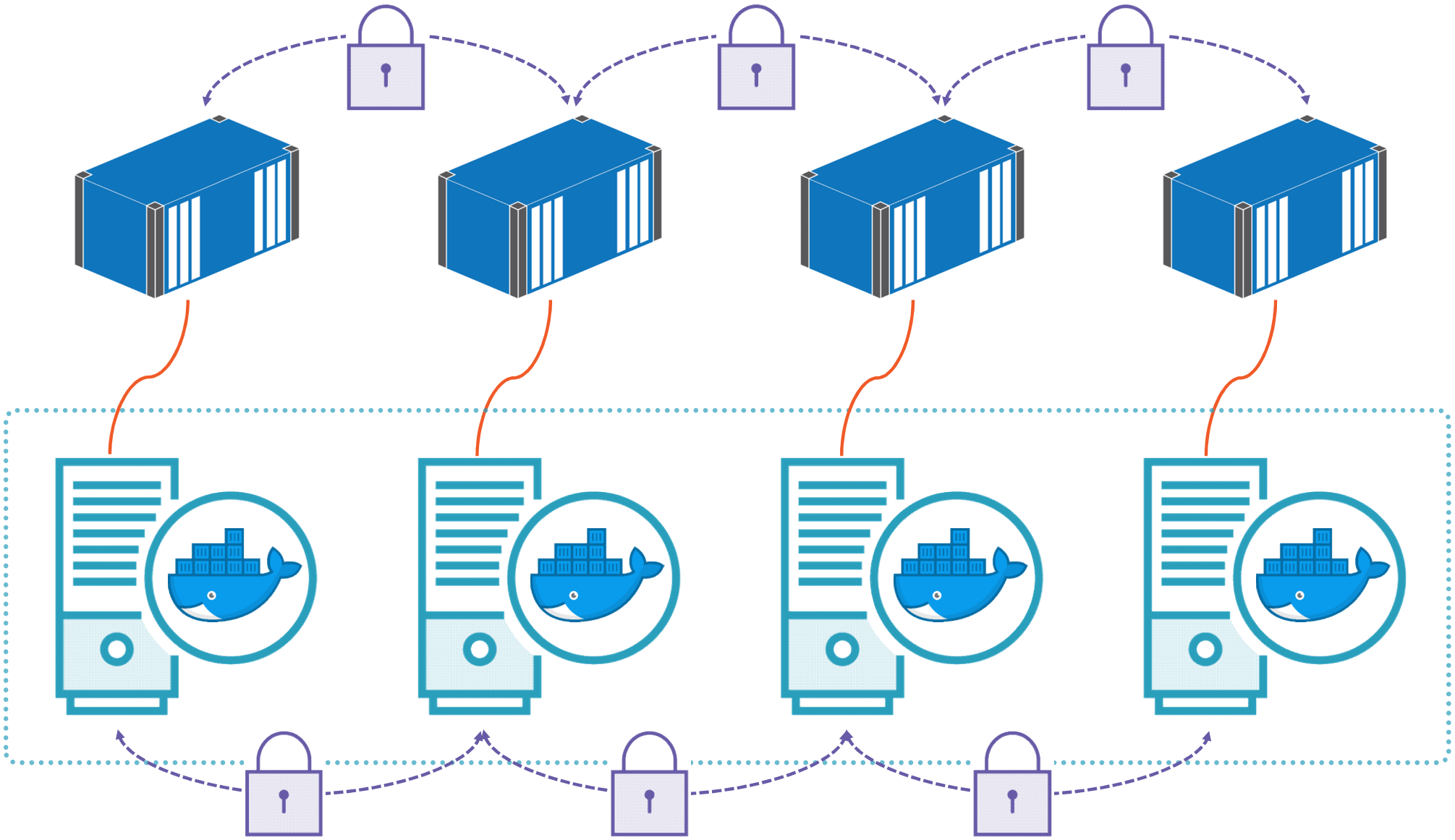


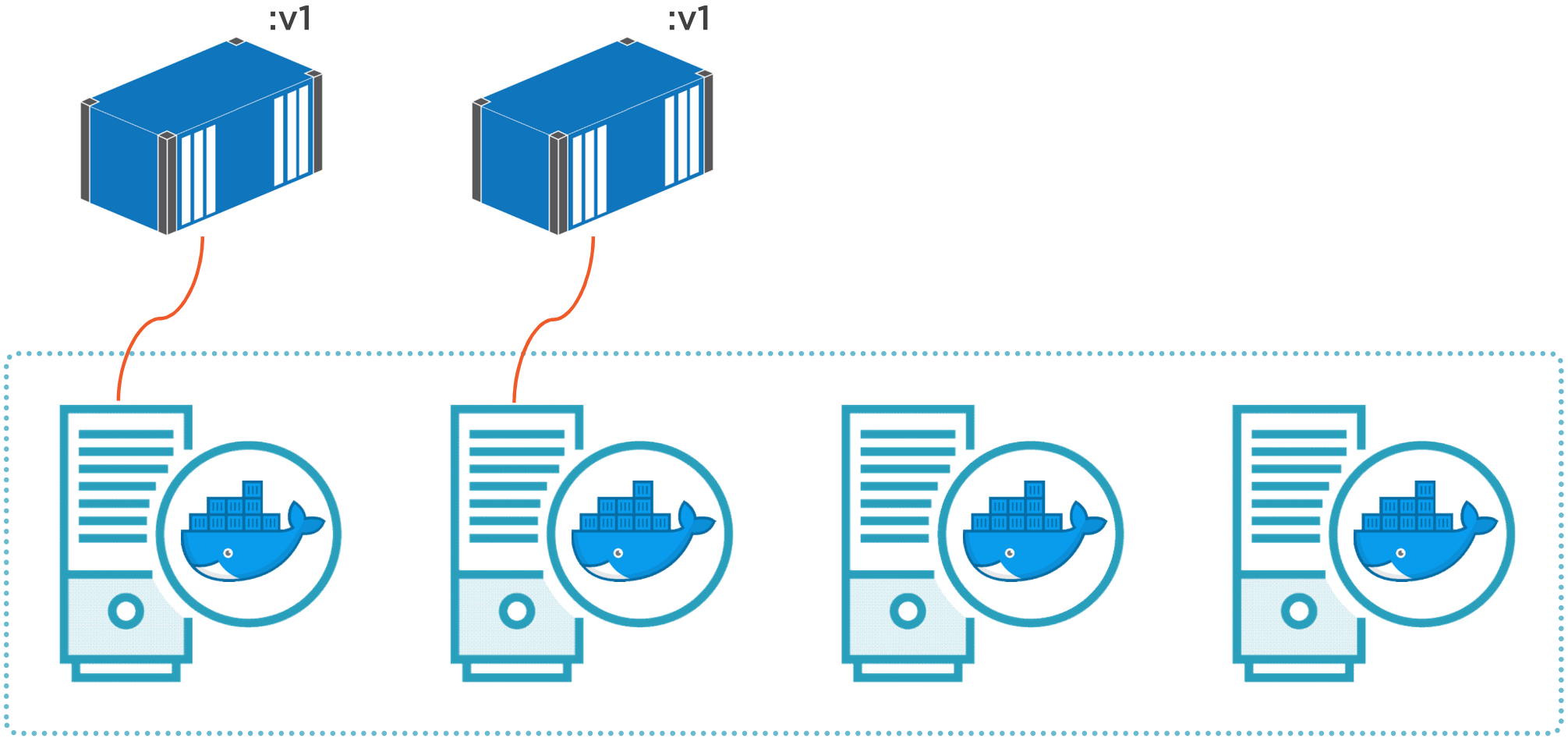

docker
SWARM

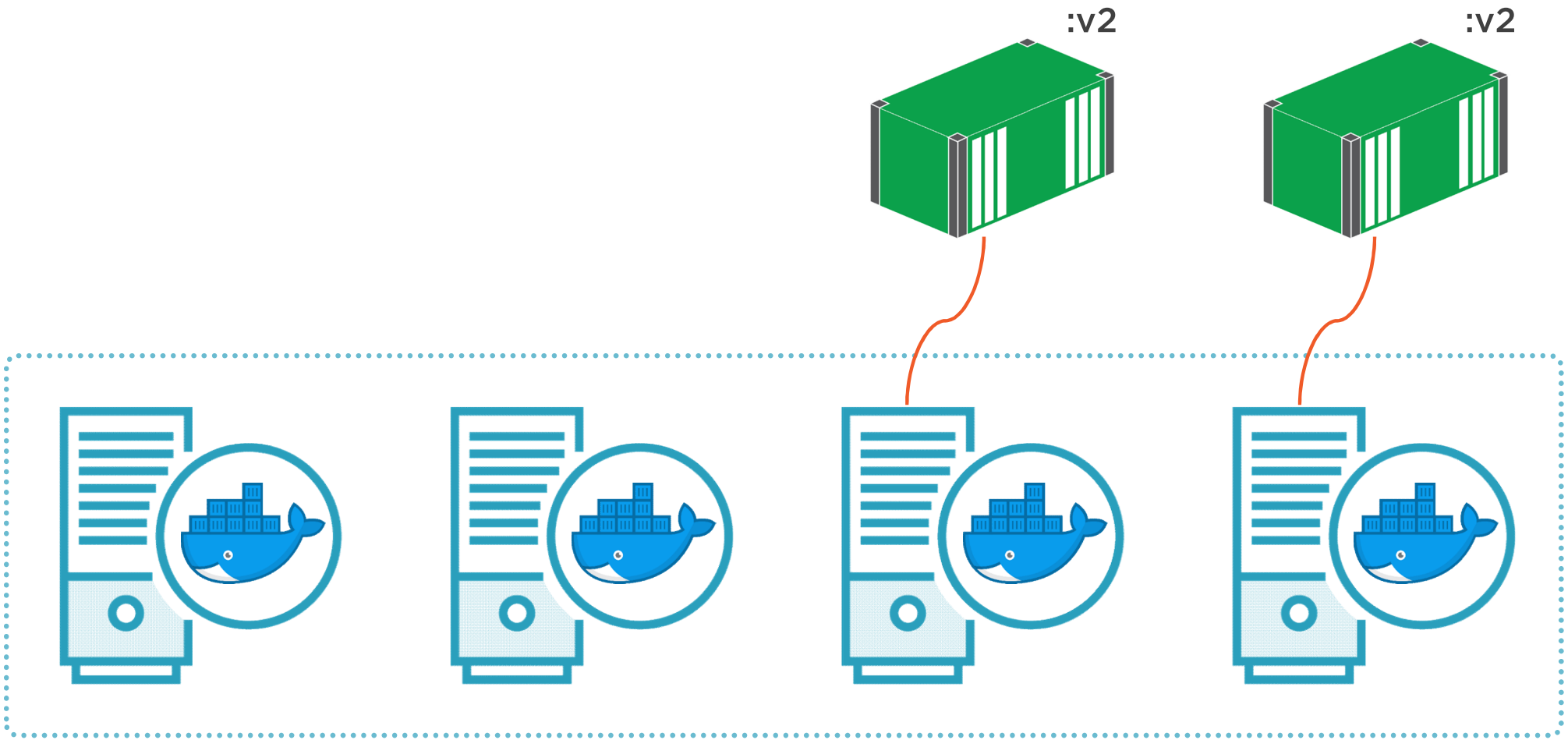




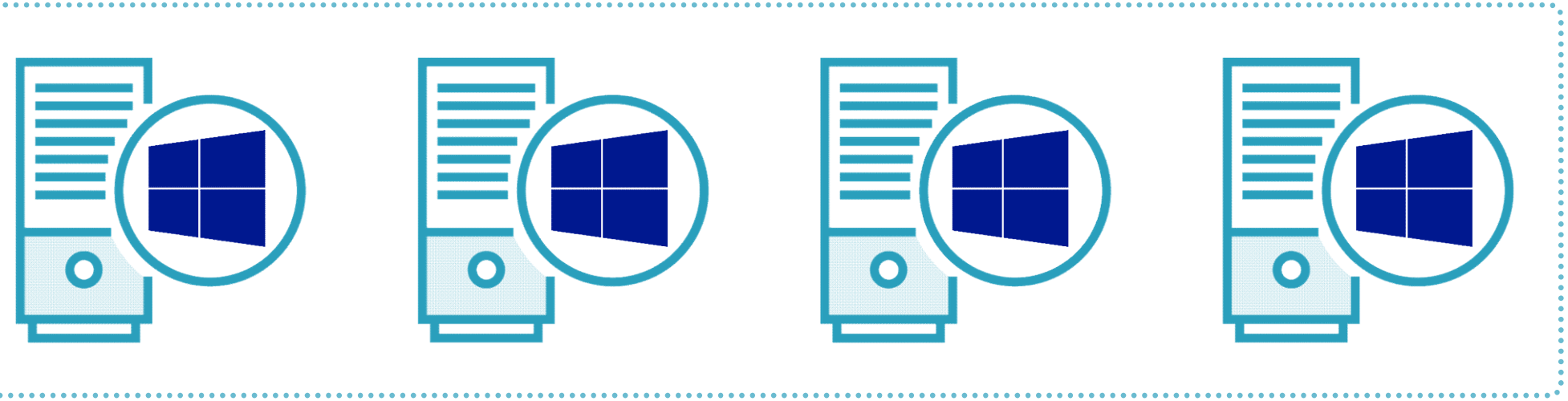





:v1
:v1

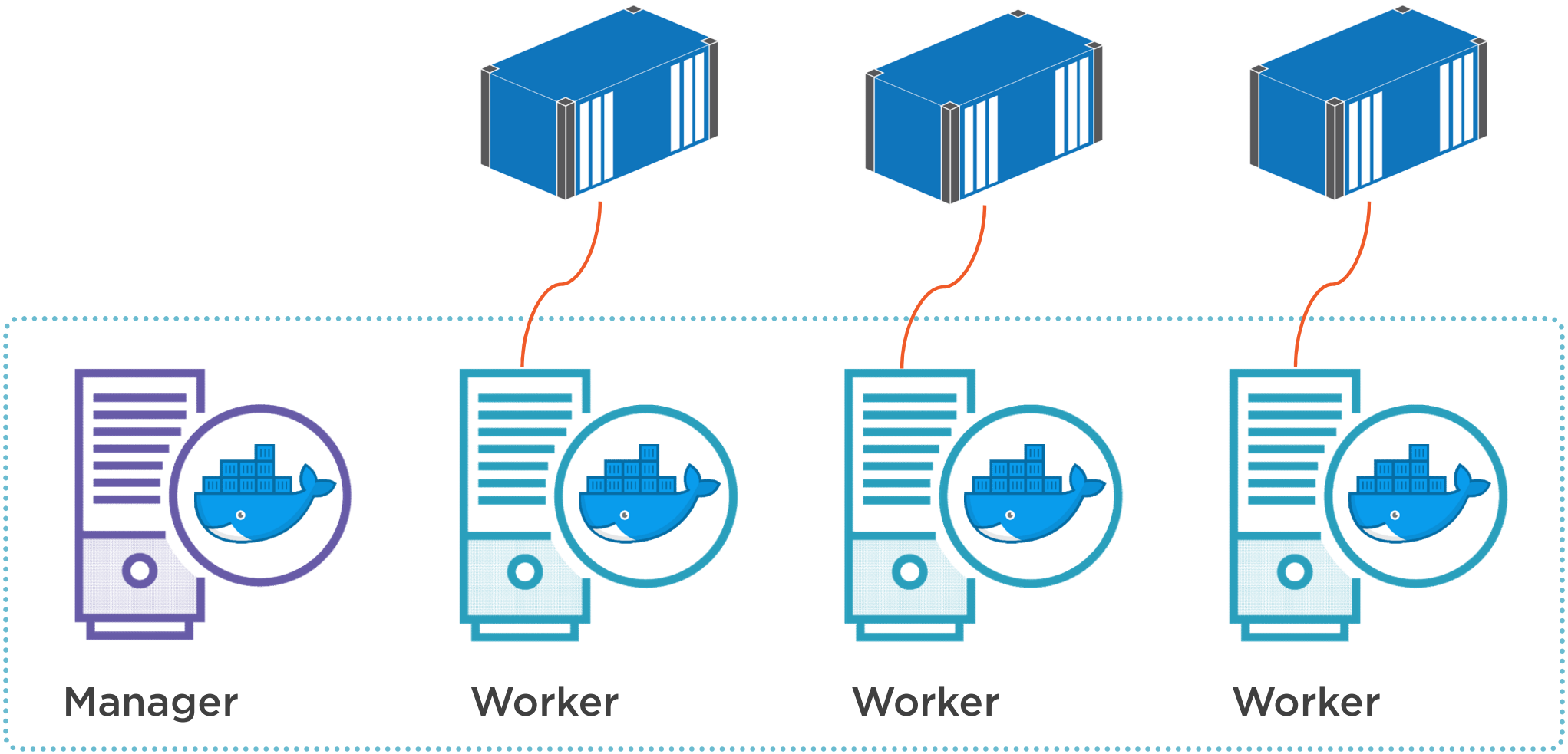

:v2
:v2

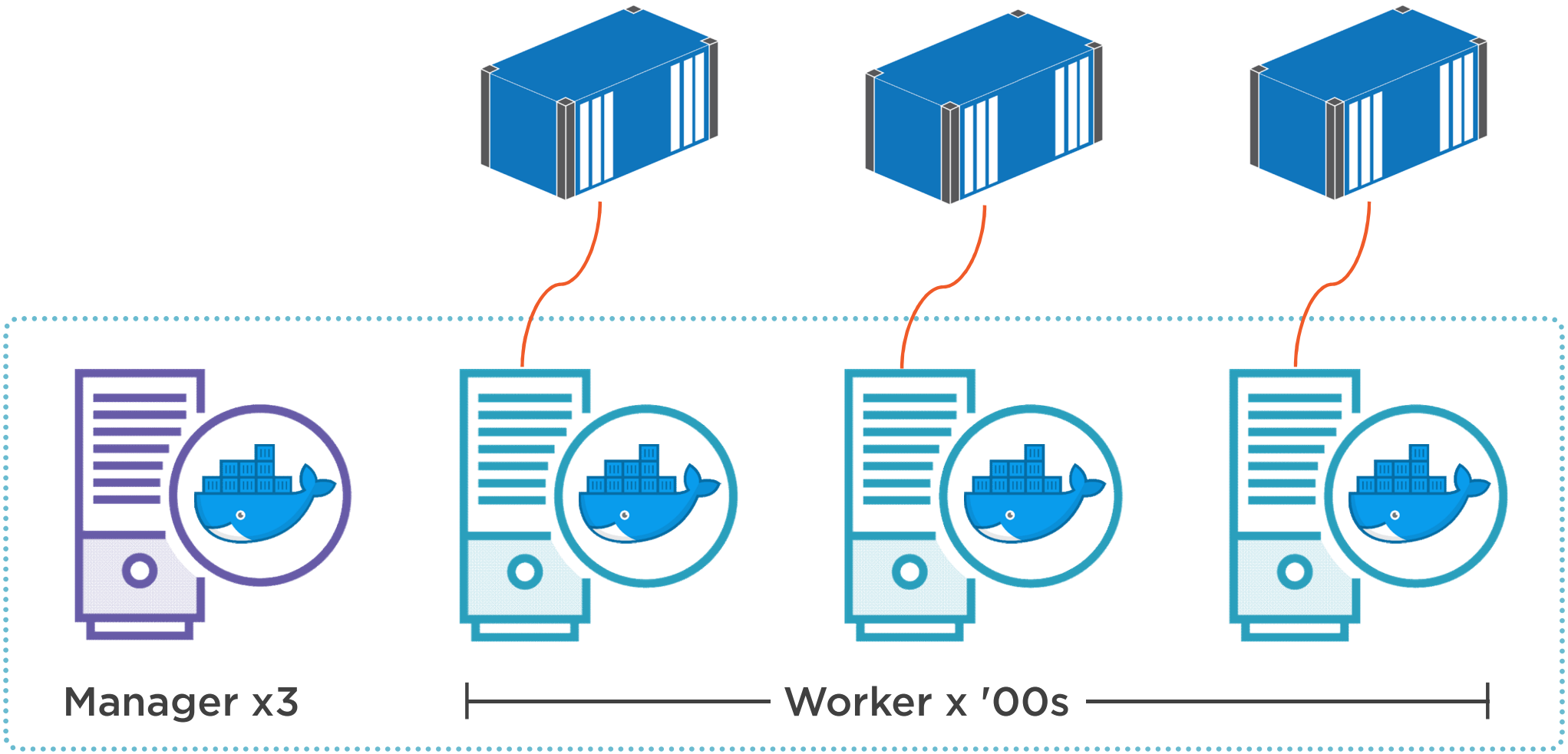

docker
SWARM

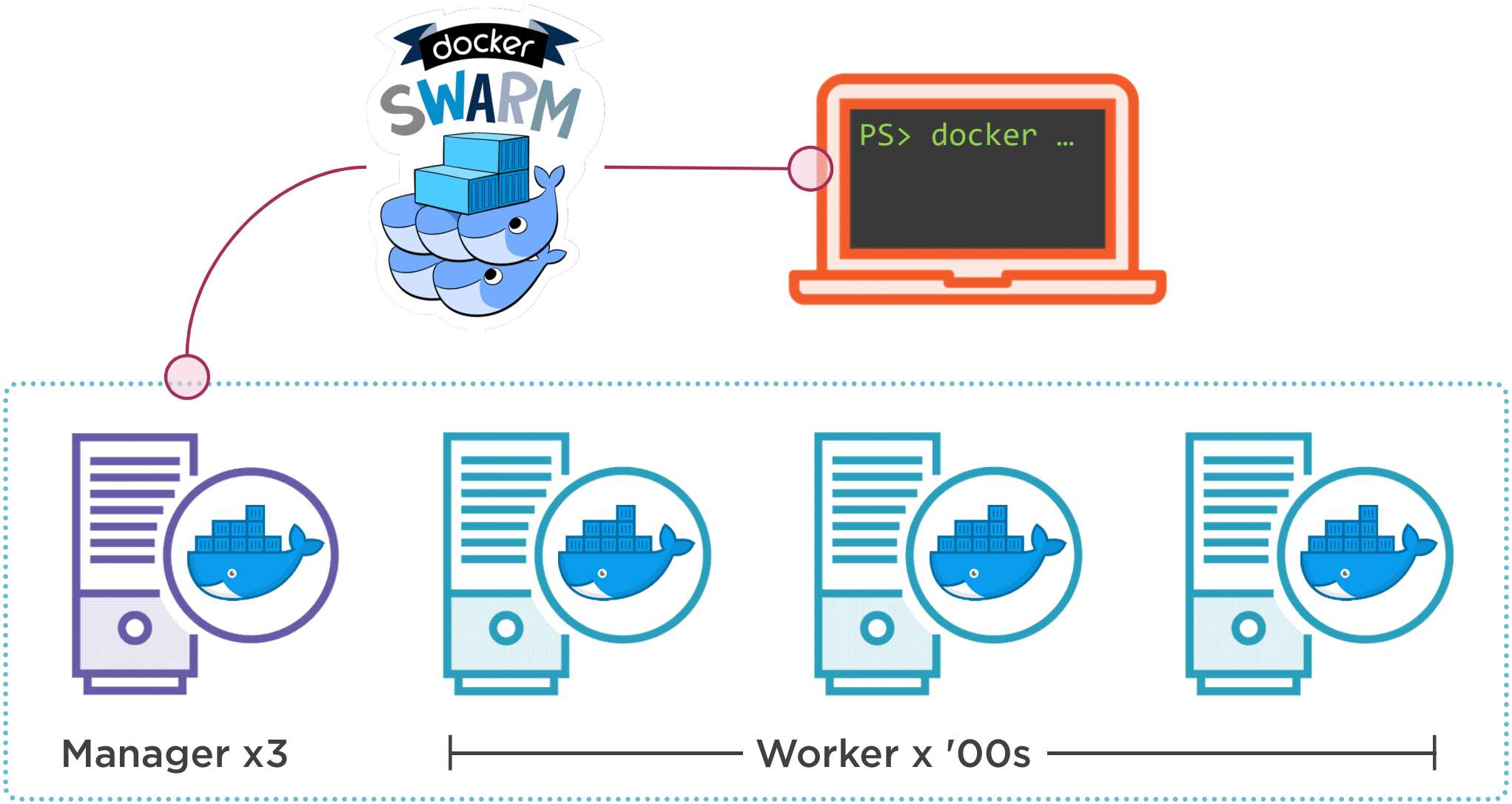


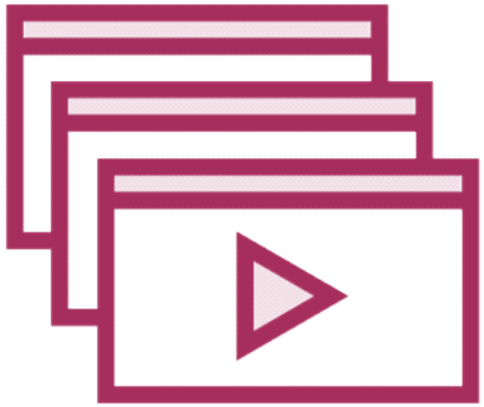
Production with Docker Swarm

- Creating a swarm in Azure
- Deploying to the swarm
- Using production features


Manager
Worker
Worker
Worker

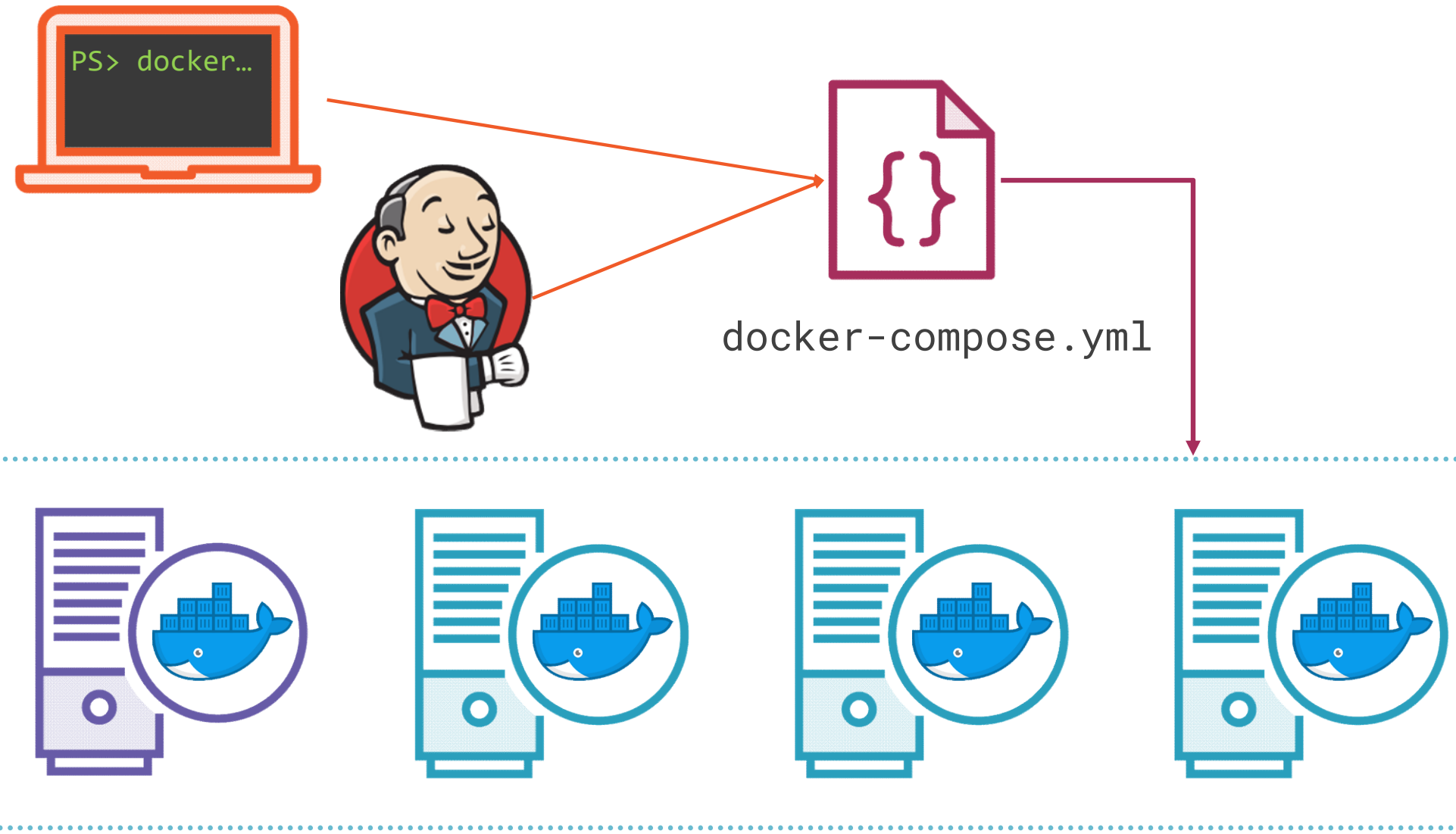

Manager x3
Worker x '00s

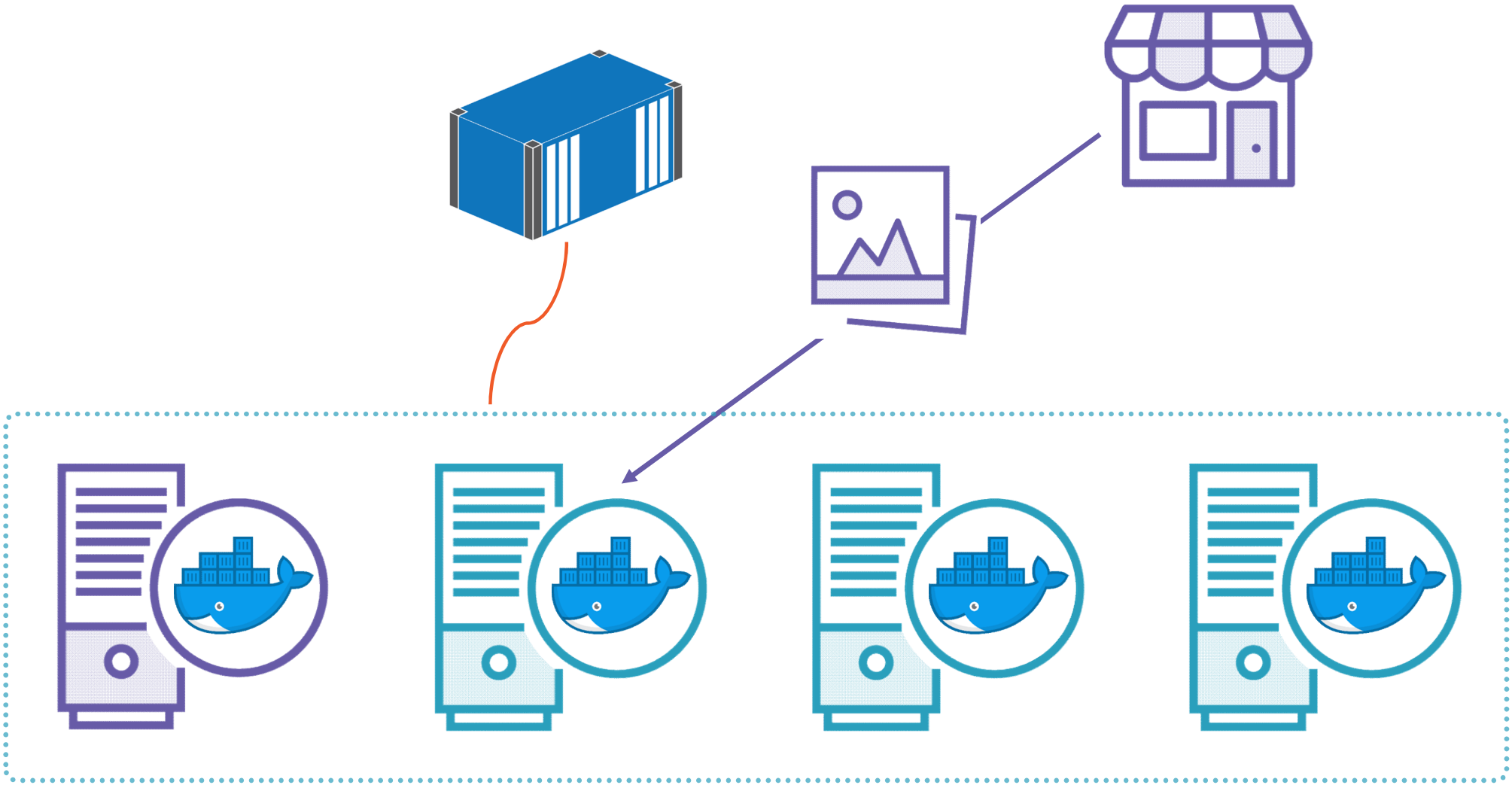

docker
SWARM
PS> docker ...
Manager x3
Worker x '00s

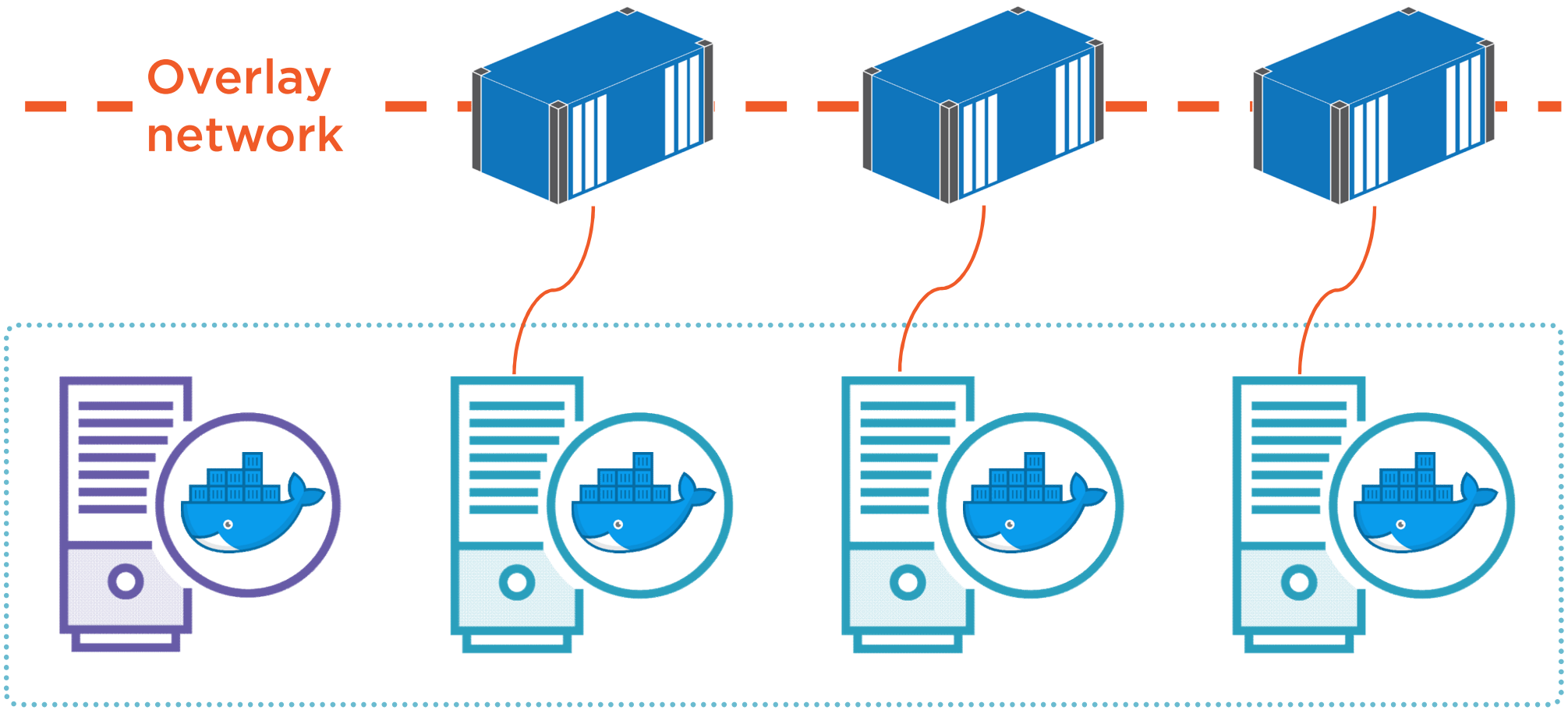


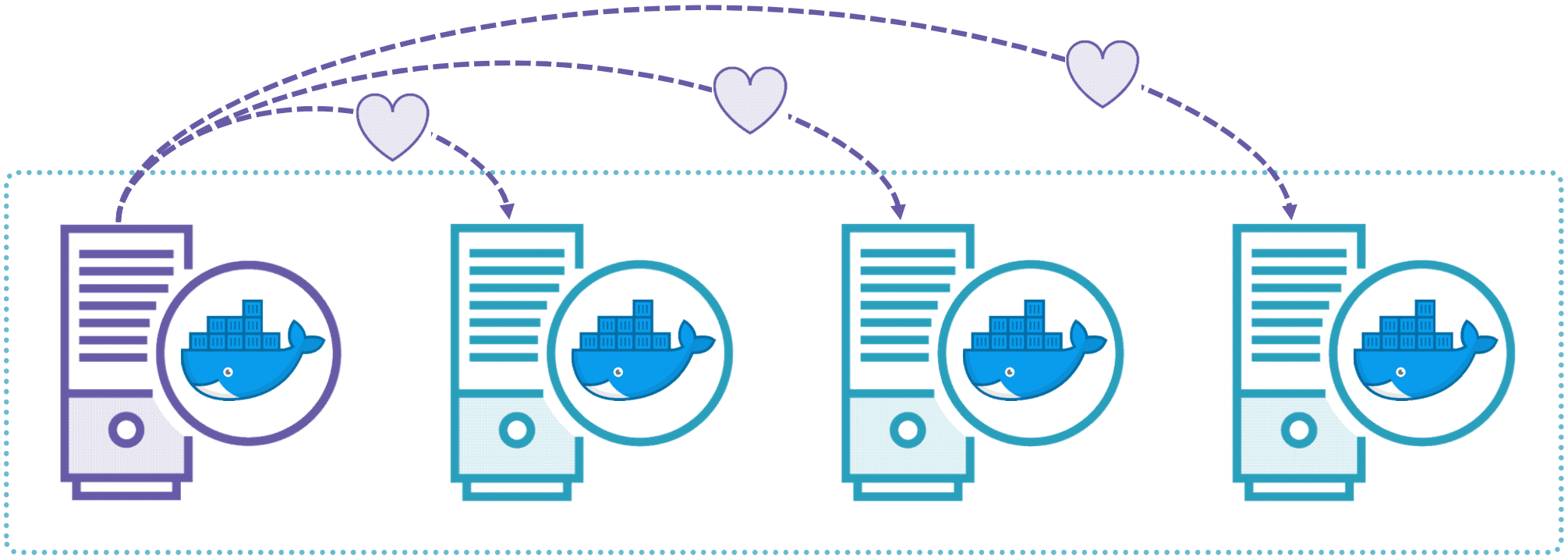
Docker Swarm: Native Docker Clustering

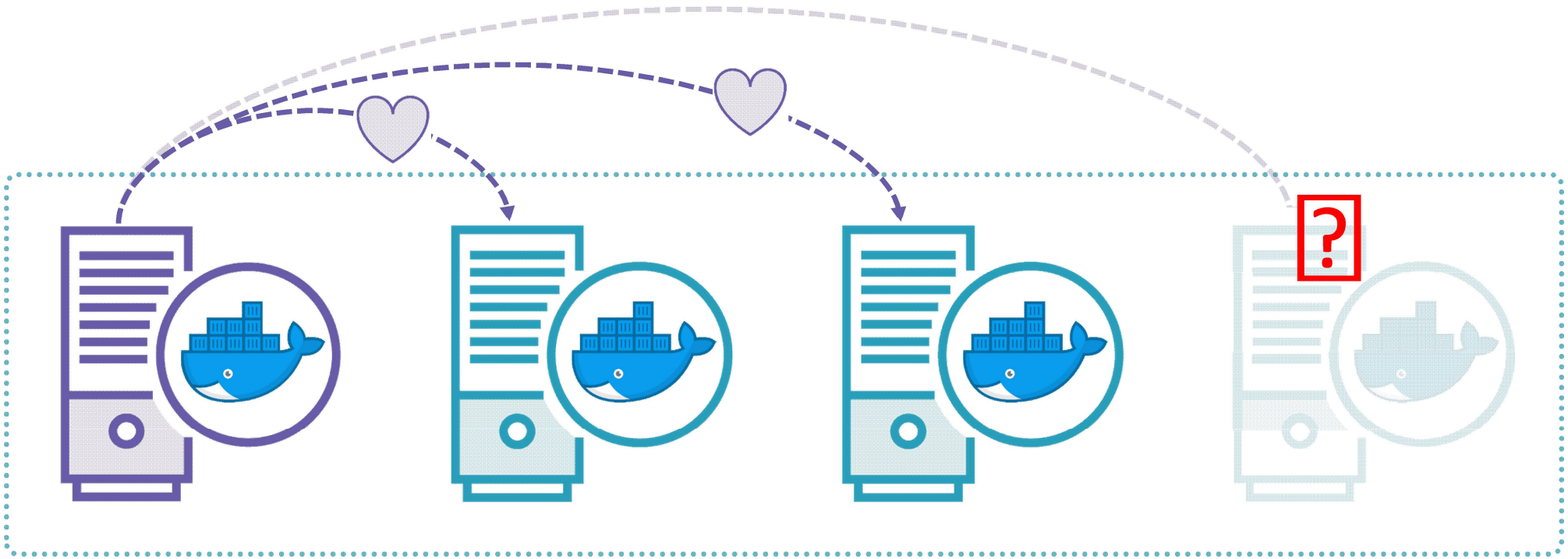
- Nigel Poulton

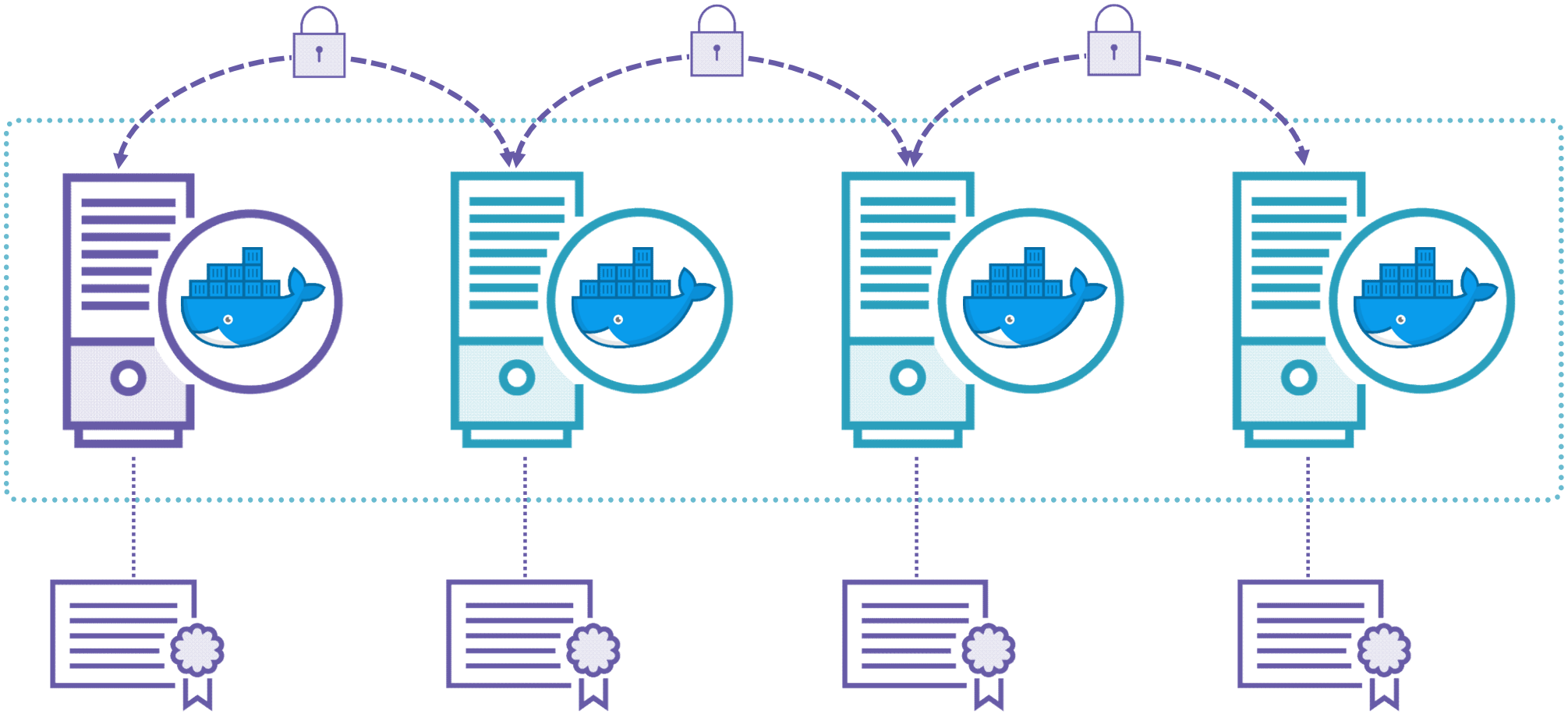

PS> docker…
{ }
docker-compose.yml

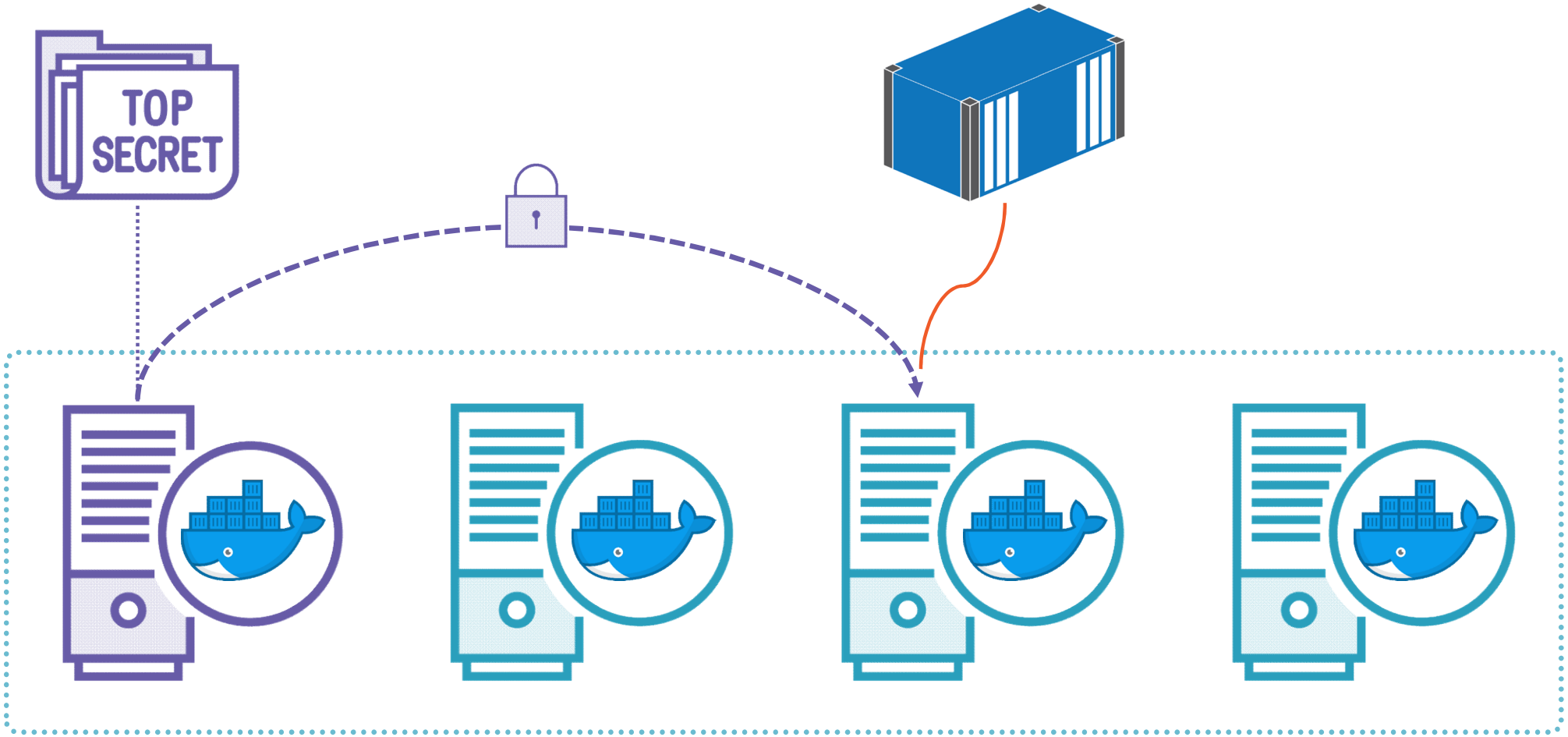


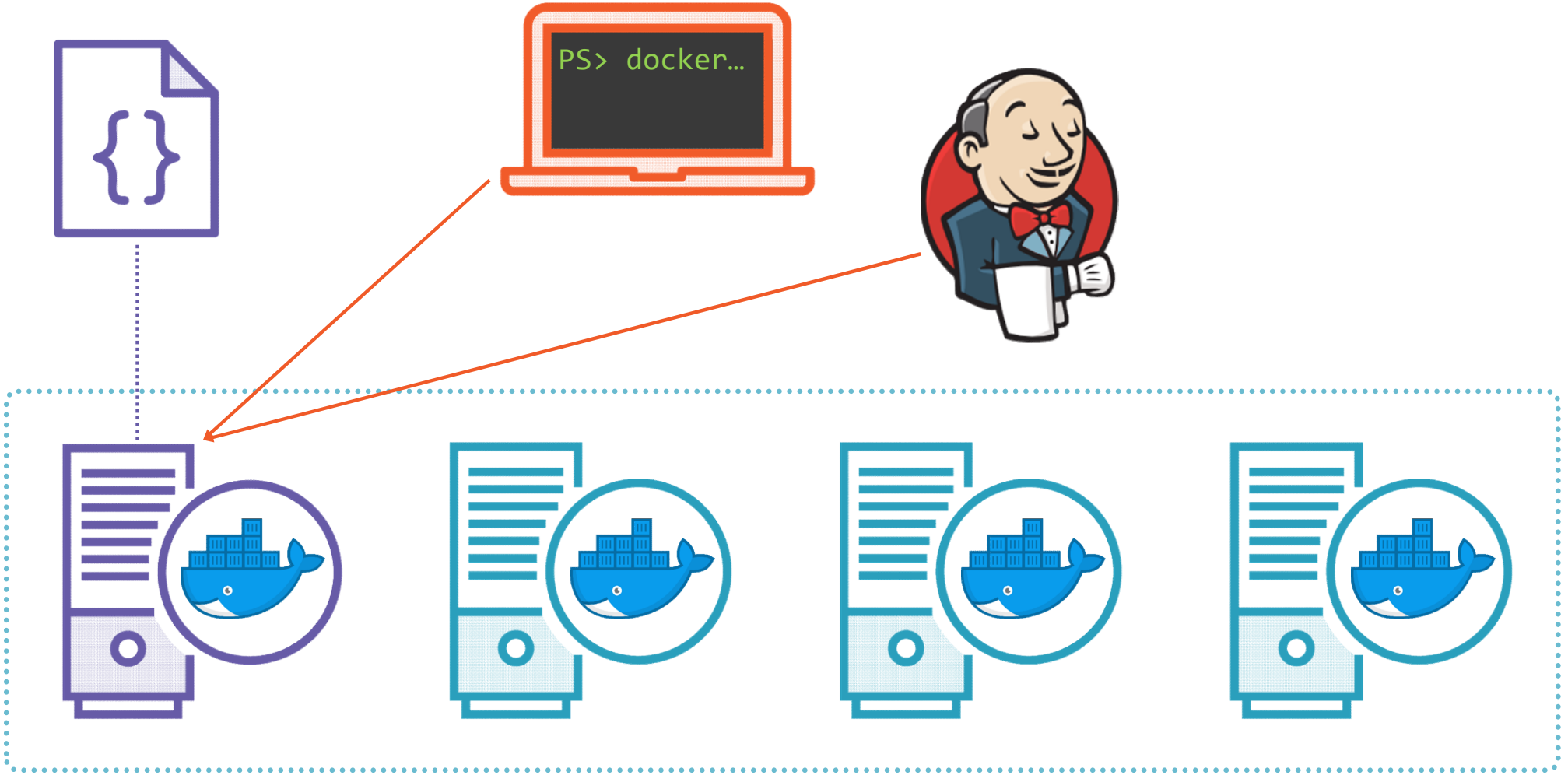

Overlay
network



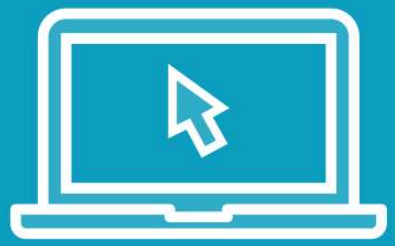





TOP
SECRET


PS> docker…

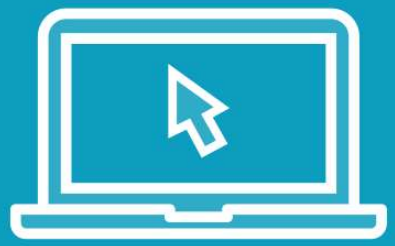
Demo



Creating a Production Swarm

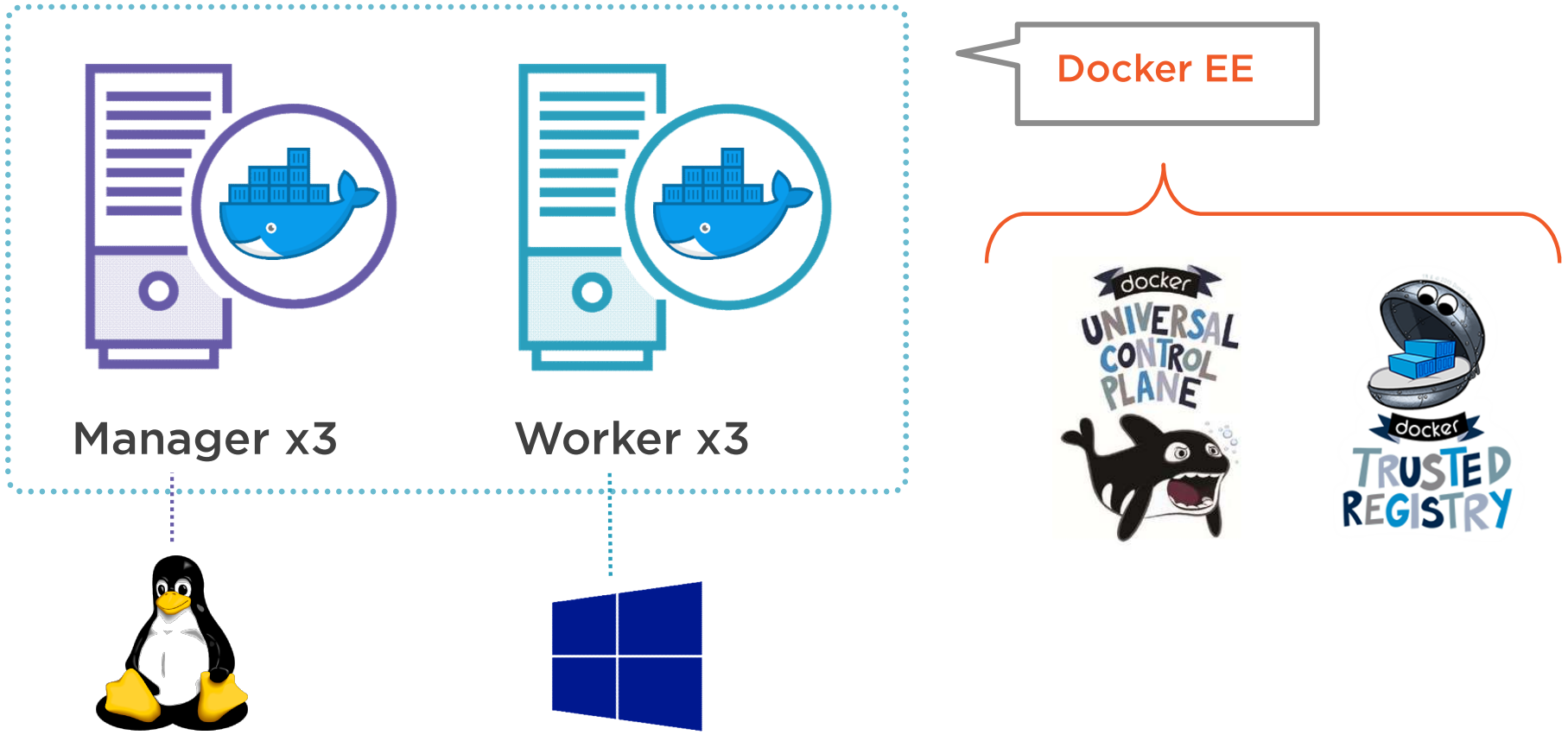
- Docker on Azure Marketplace
- Deploying Docker EE
- Hybrid Linux and Windows swarm

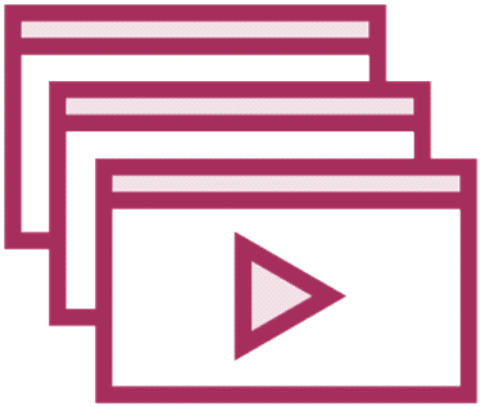
Demo



Deploying to the Swarm

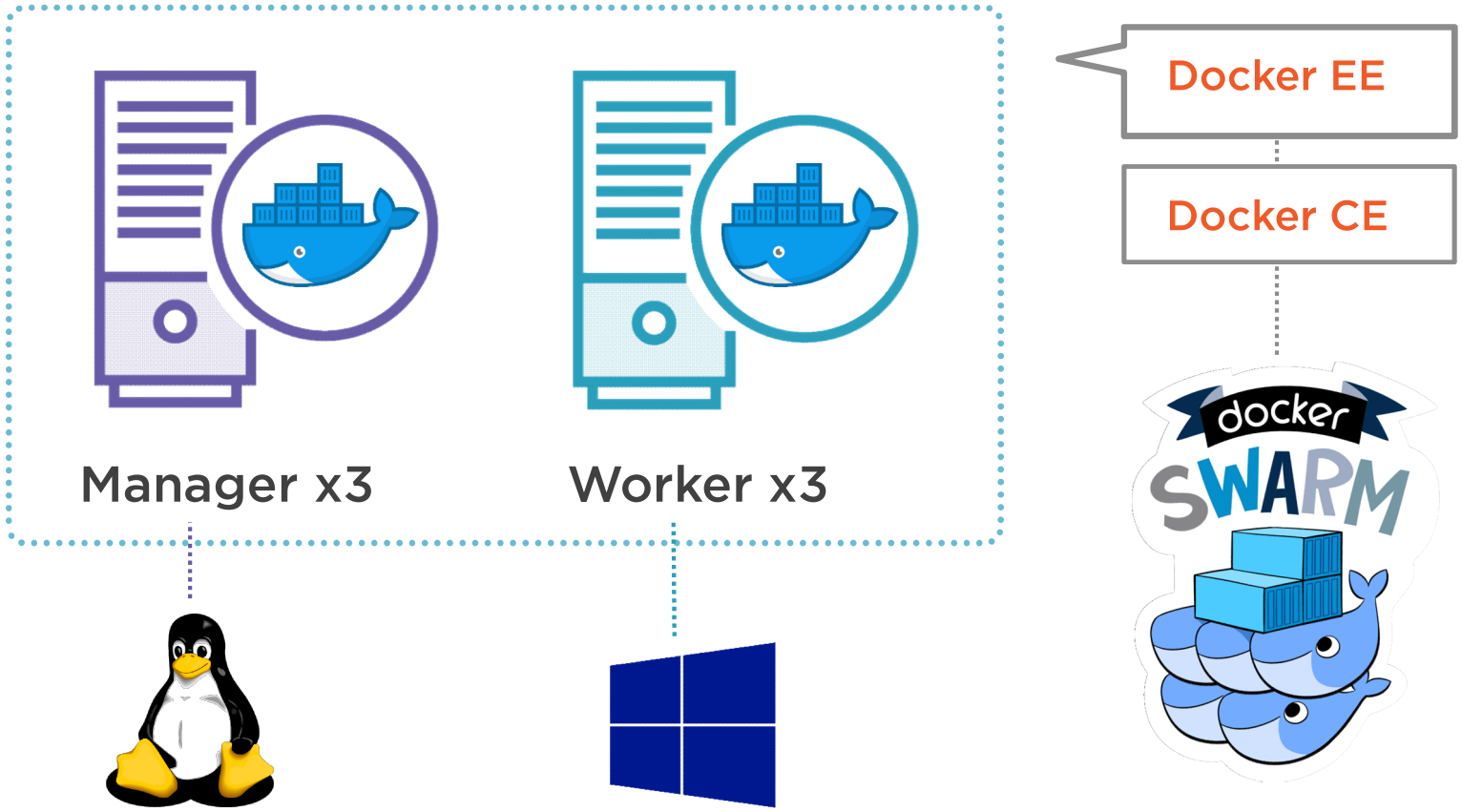
- Extended Docker Compose definition
- Private registry for image storage
- Creating a Docker stack


Manager x3
Worker x3
Docker EE
docker
UNIVERSAL CONTROL PLANE
docker
TRUSTED REGISTRY



Getting Started with Docker Datacenter

- Elton Stoneman


Manager x3
Worker x3
Docker EE
Docker CE
docker
SWARM

```
# app image

FROM microsoft/windowsservercore:10.0.14393.1884
```

Pinned Dockerfiles

Using explicit FROM image versions

```
# app image
FROM microsoft/aspnet
```

Avoiding Latest

Implicit `:latest` tag changes over time

```
# app image
FROM microsoft/aspnet:windowsservercore-10.0.14393.1884
```

Avoiding Latest

Specific tag doesn’t change

```
message-queue:
  image: dtrlb-...azure.com/webinar/nats:nanoserver
  networks:
    - app-net
```

Core Docker Compose File

Basic service definitions – images and networks

```
kibana:
  ports:
    - "5601:5601"
  depends_on:
    - elasticsearch
```

Test Docker Compose File

Adds non-production configuration

```
message-queue:
  deploy:
    placement:
      constraints:
        - node.platform.os == windows
```

Production Docker Compose File

Adds configuration for running in swarm mode

```
message-queue:
  deploy:
    endpoint_mode: dnsrr
```

Production Docker Compose File

Adds configuration for running in swarm mode

```
docker-compose `
 -f docker-compose.yml `
 -f docker-compose-prod.yml `
 config > docker-stack.yml
```

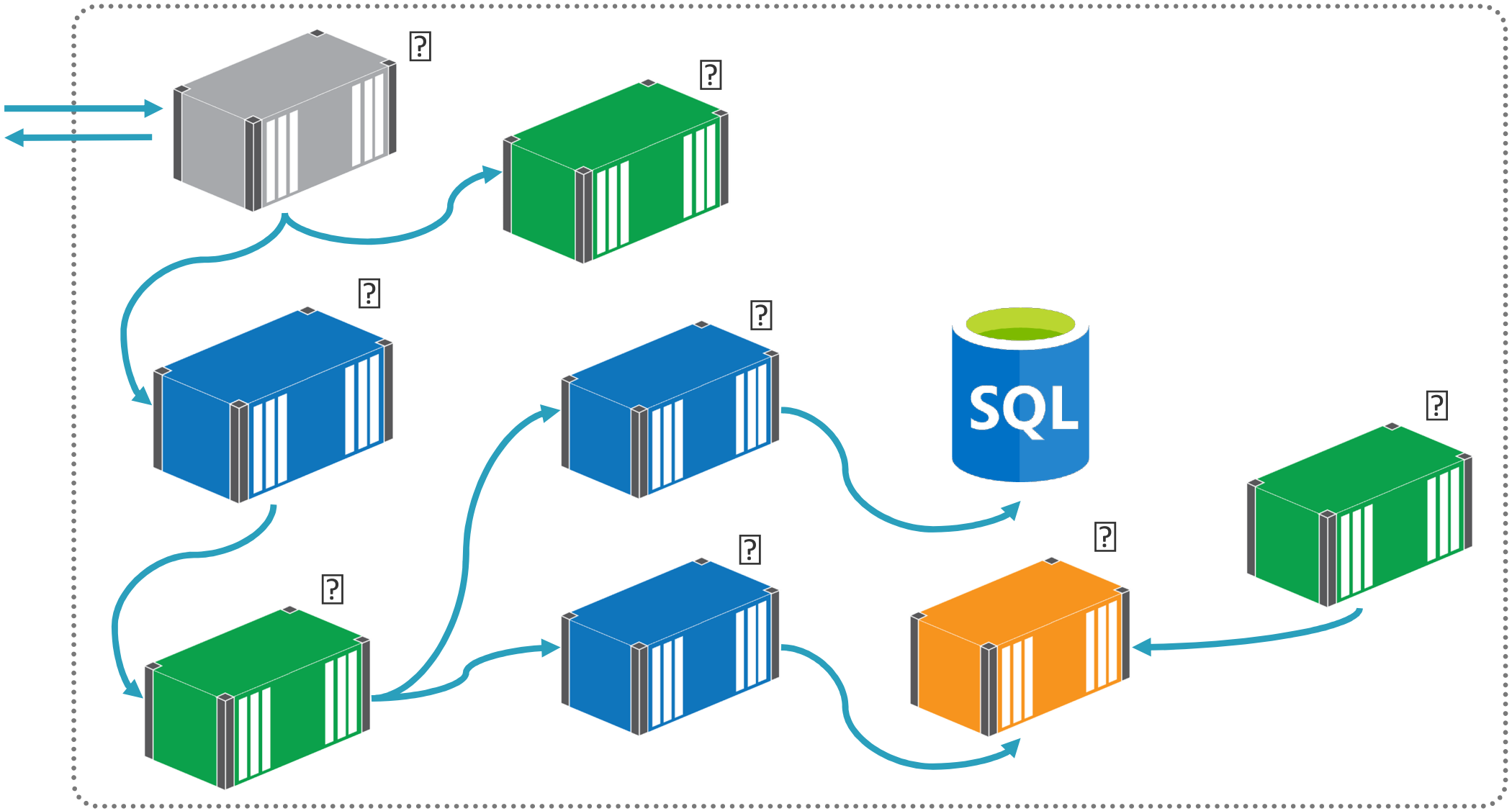
Docker Compose Config

Validate and merge input files

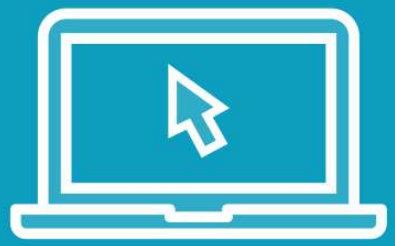
```
docker stack deploy `
 -c docker-stack.yml `
 webinar
```

Deploying to Docker Swarm

Creating a stack from the compose file


?
?
?
?
?
?
?
?
SQL

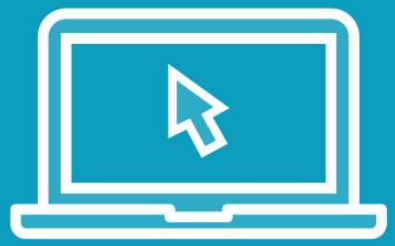
Demo



Integrating with Docker Secrets

- Extracting connection string config
- Injecting the secret into the app
- Updating the stack

Demo



Integrating with SQL Azure

- Creating swarm secrets
- Defining secrets in Docker Compose
- Deploying the updated application

```
<connectionStrings
  configSource="connection-strings.config"
/>
```

Separating Config Sections

Loading connection strings outside of `app.config`

```
ENV CONNECTION_STRINGS_PATH= `
    "C:\ProgramData\Docker\secrets\connection-strings"
```

Injecting Docker Secrets

Specifying the location of the secret file

```
if (Test-Path $env:CONNECTION_STRINGS_PATH) {
  New-Item -Path C:\...\connection-strings.config `
           -ItemType SymbolicLink `
           -Value $env:CONNECTION_STRINGS_PATH
```

Injecting Docker Secrets

Creating a symbolic link to the target path

```
docker secret create `
 webinar-connection-strings `
 .\connection-strings-prod.config
```

Creating the Docker Secret

Stored encrypted in the swarm

```
save-handler:
  secrets:
    - connection-strings
```

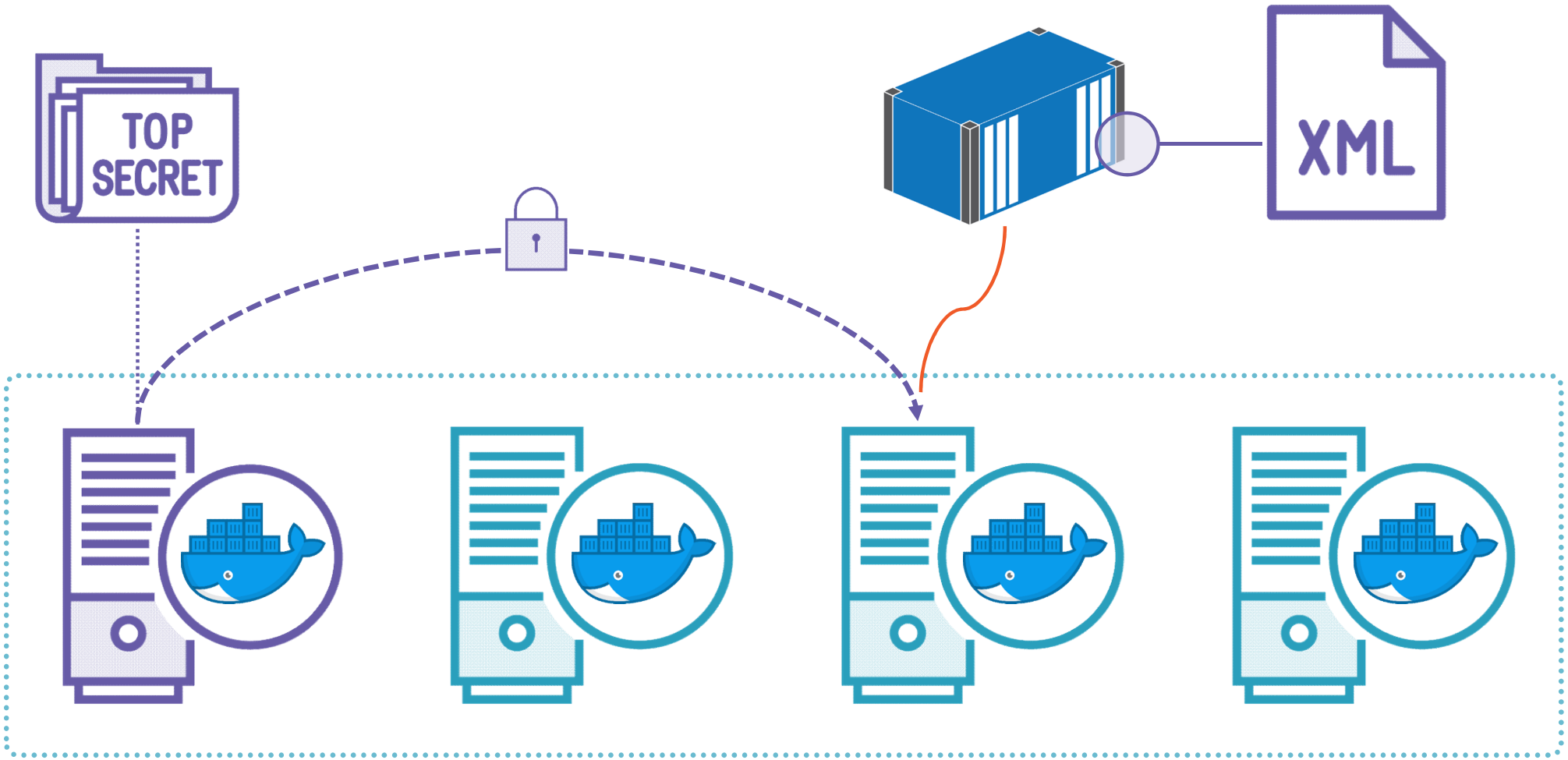
Specifying Secrets for Services

In production override Docker Compose file

```
secrets:
  connection-strings:
    external:
      name: webinar-connection-strings
```

Specifying Secrets

Linking service secrets to swarm secrets

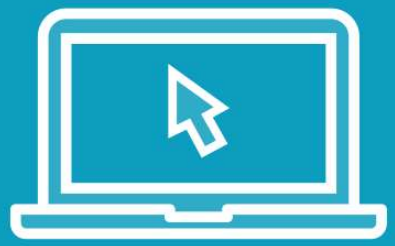

TOP
SECRET
XML

High Availability
Self-healing apps

Security
Secure secret management

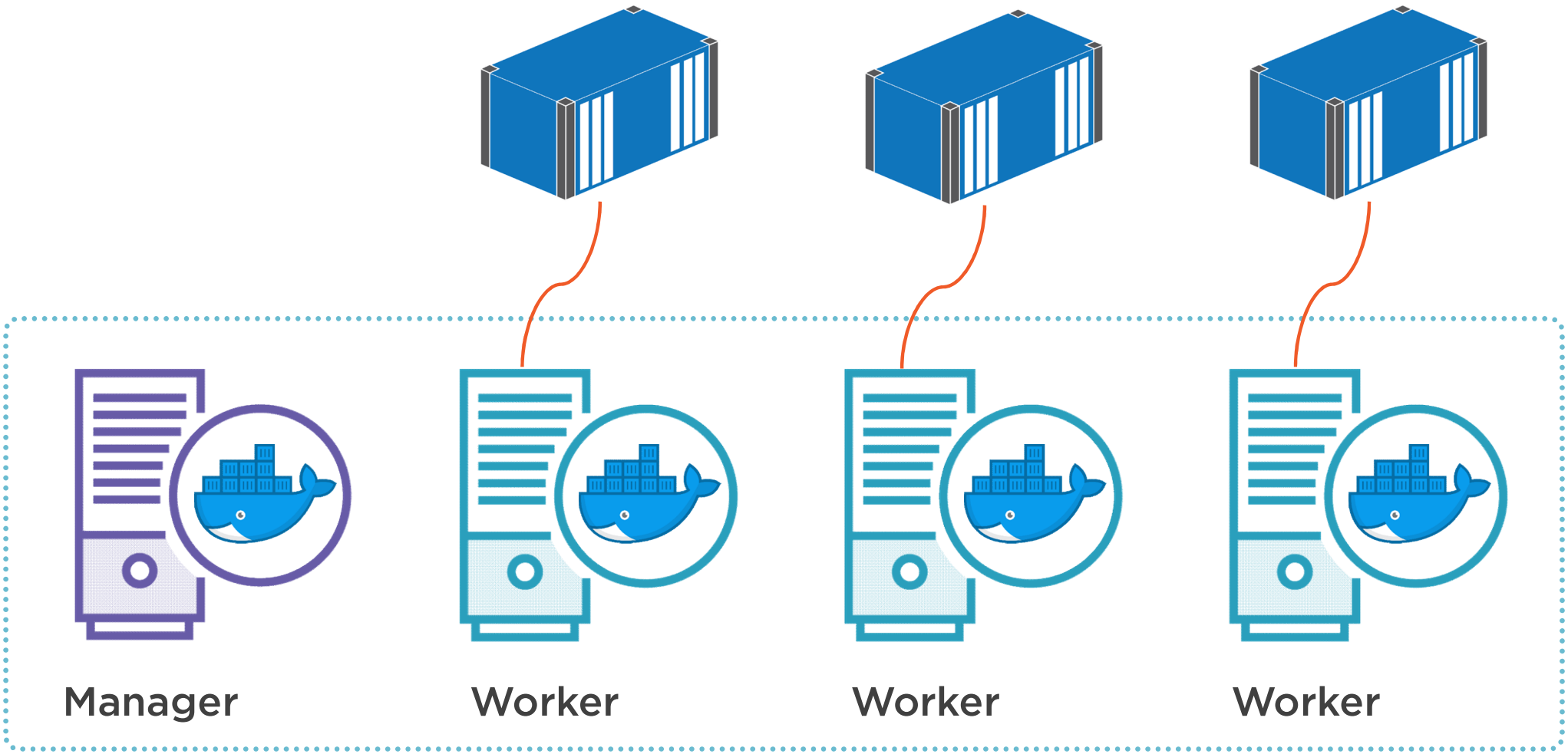
Automated Updates
And rollback

Demo

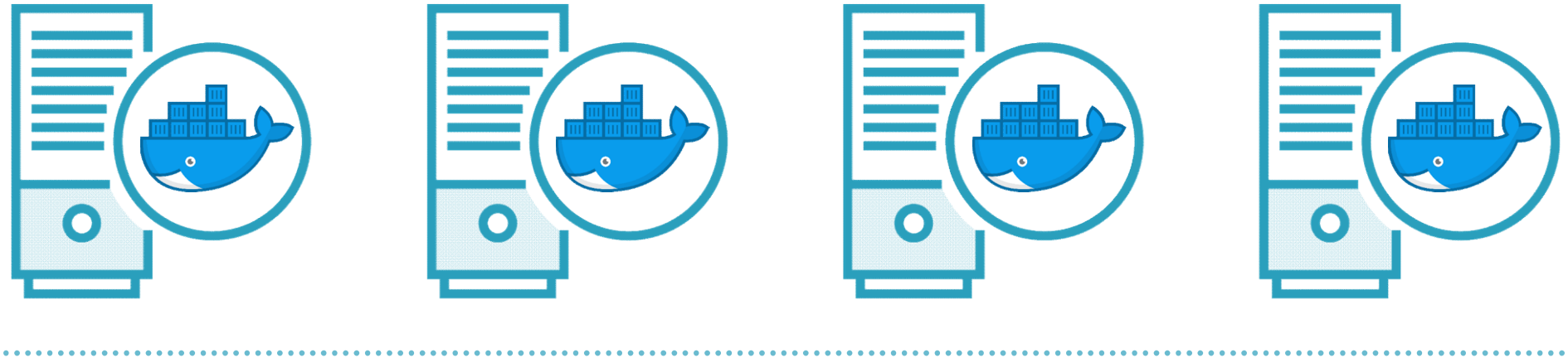


Managing Production Docker Apps

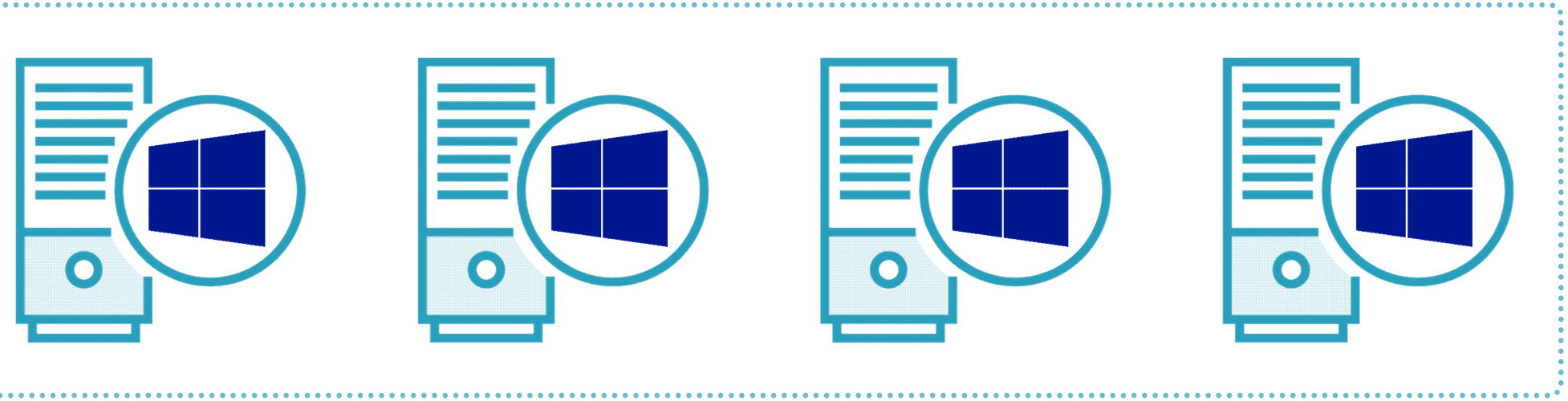
- Scaling up services and nodes
- Managing server failure
- Visibility with application dashboards


Manager
Worker
Worker
Worker


docker
SWARM




docker
SWARM



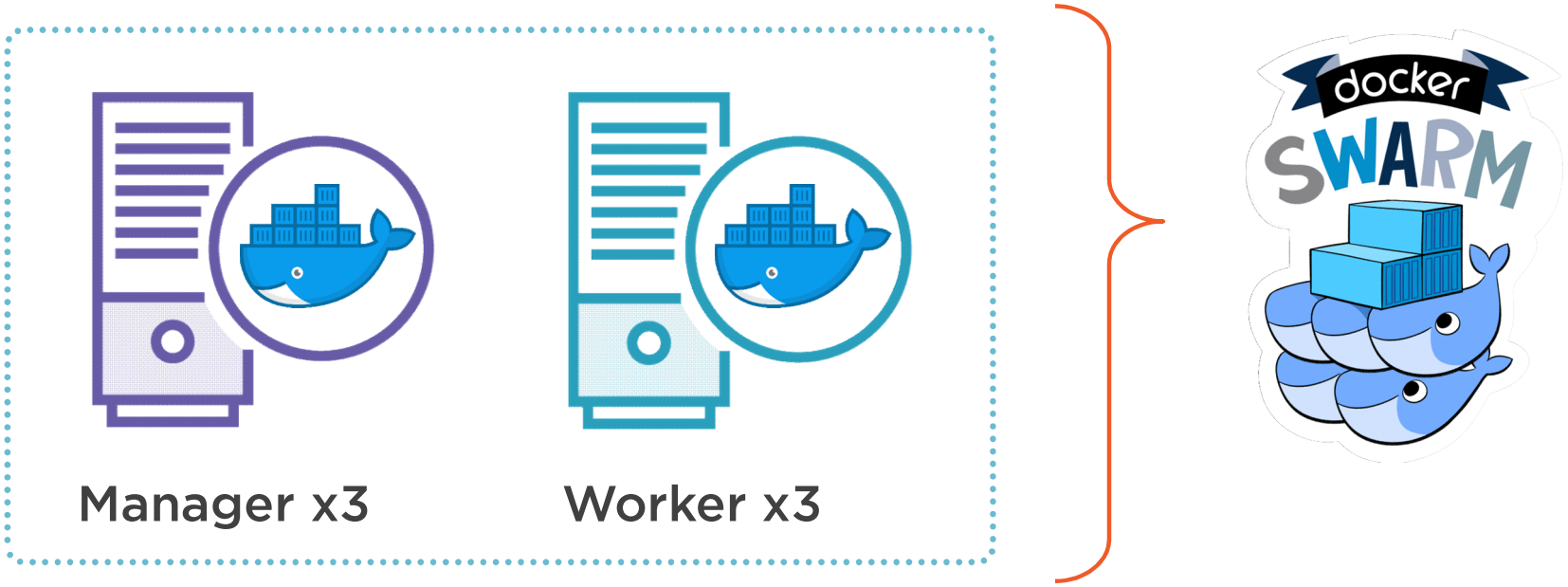
Manager x3

Worker x3

Docker Enterprise Edition

Production support for Windows containers

Enterprise management tooling



Docker Community Edition

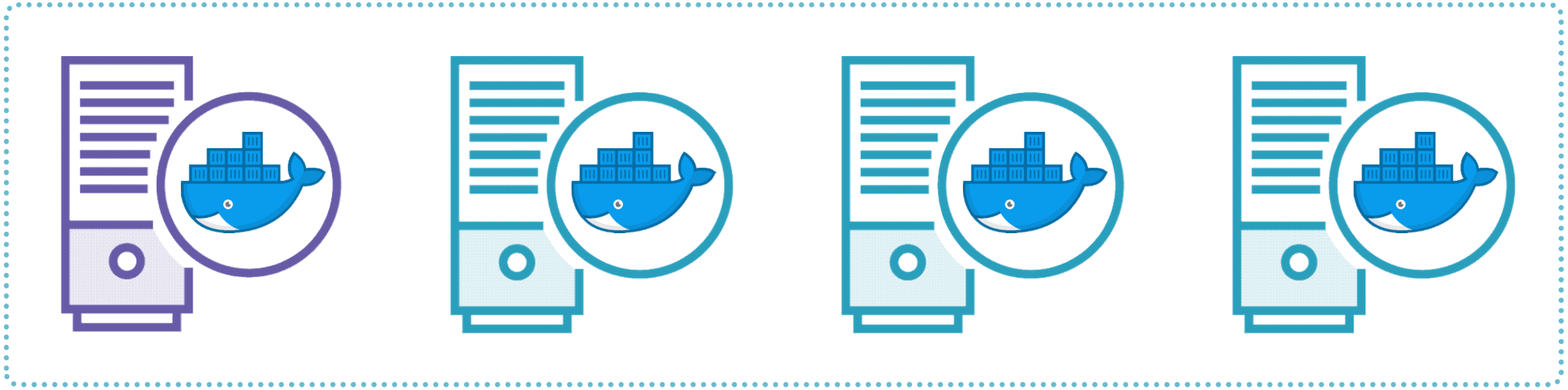
Same core platform as Docker EE

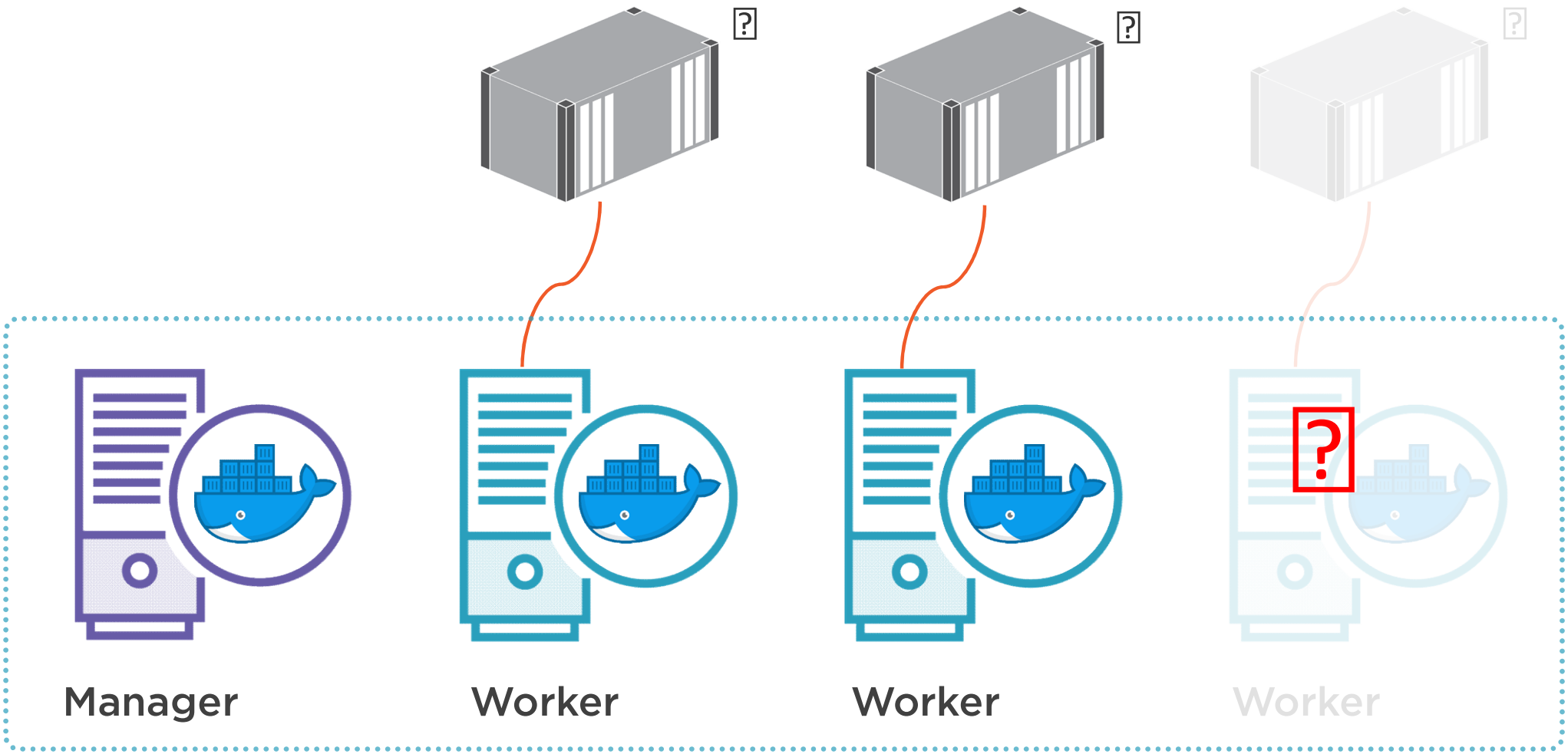
Full swarm functionality

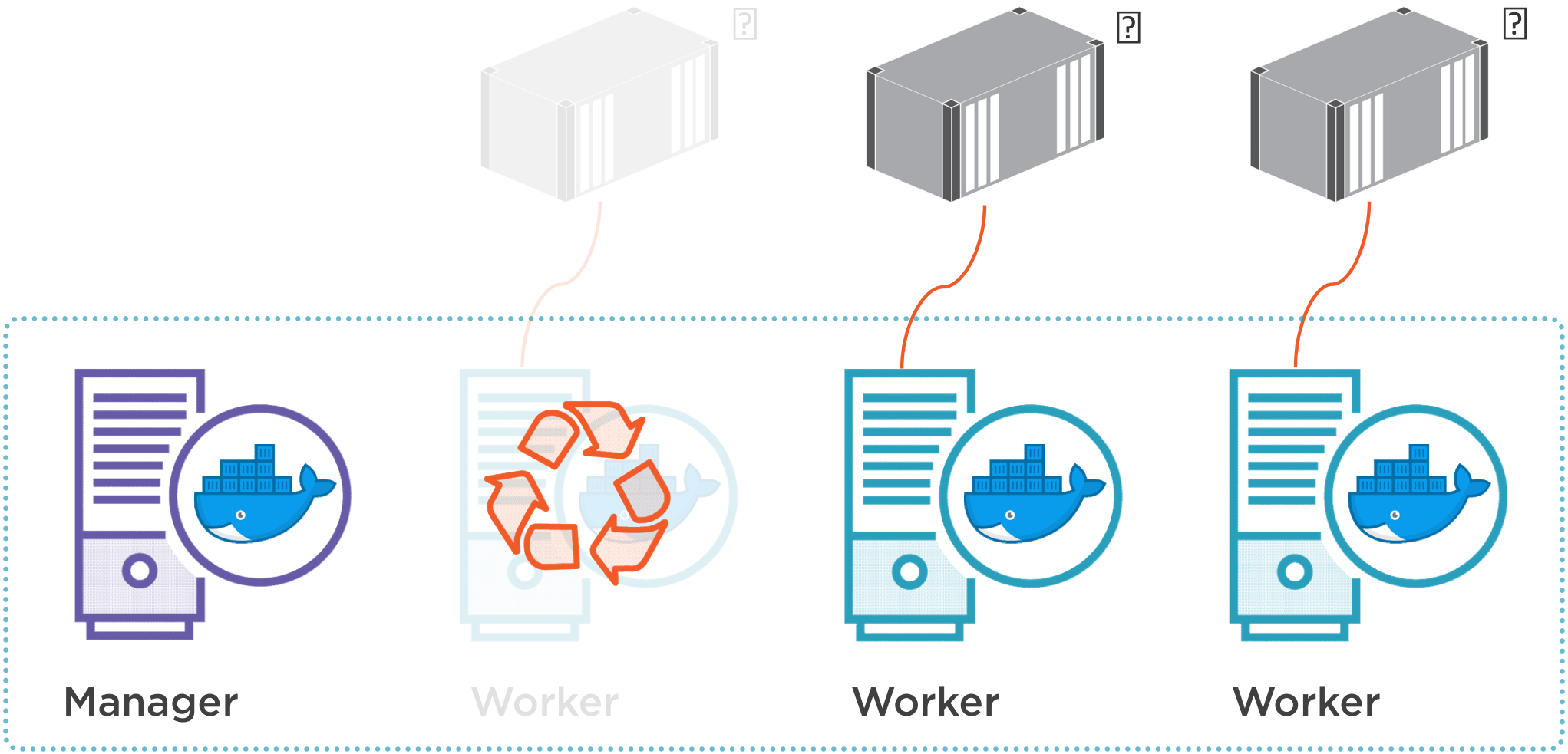
Application Stacks
Docker Compose definition

Docker Secrets
Secure storage for config

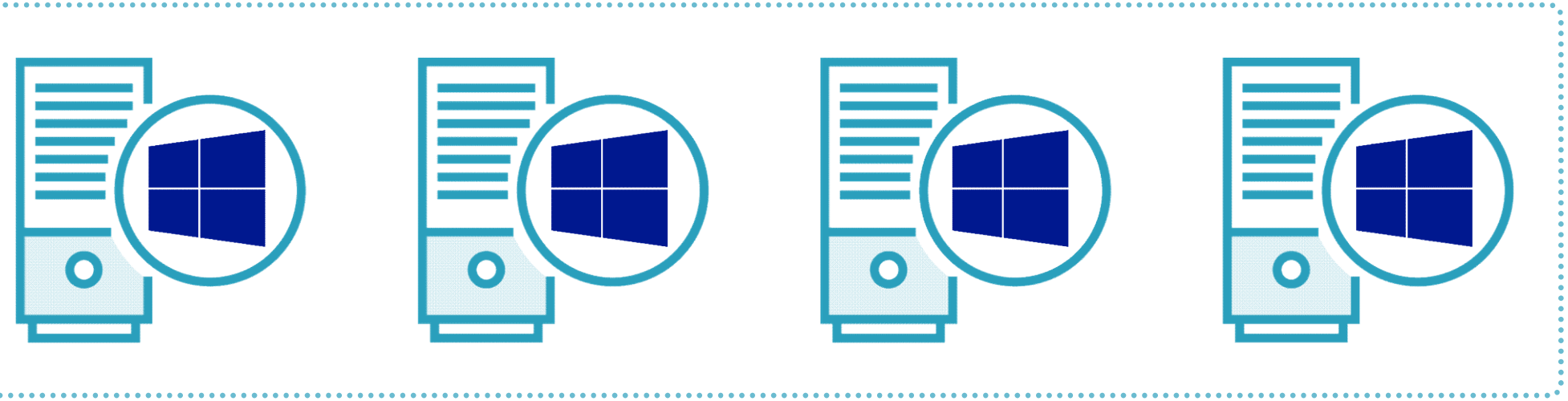
Support Cycle
Maintenance & updates

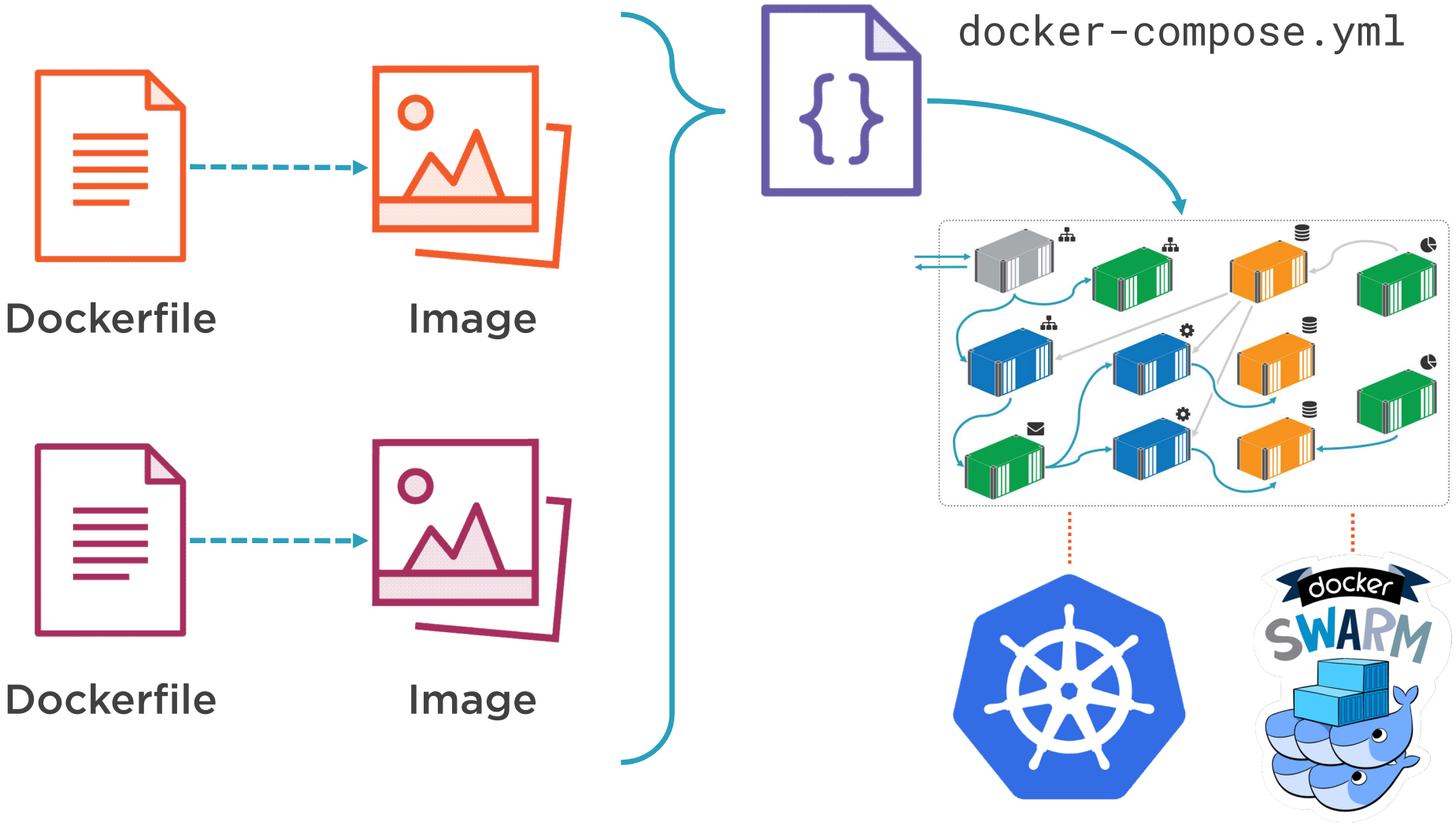


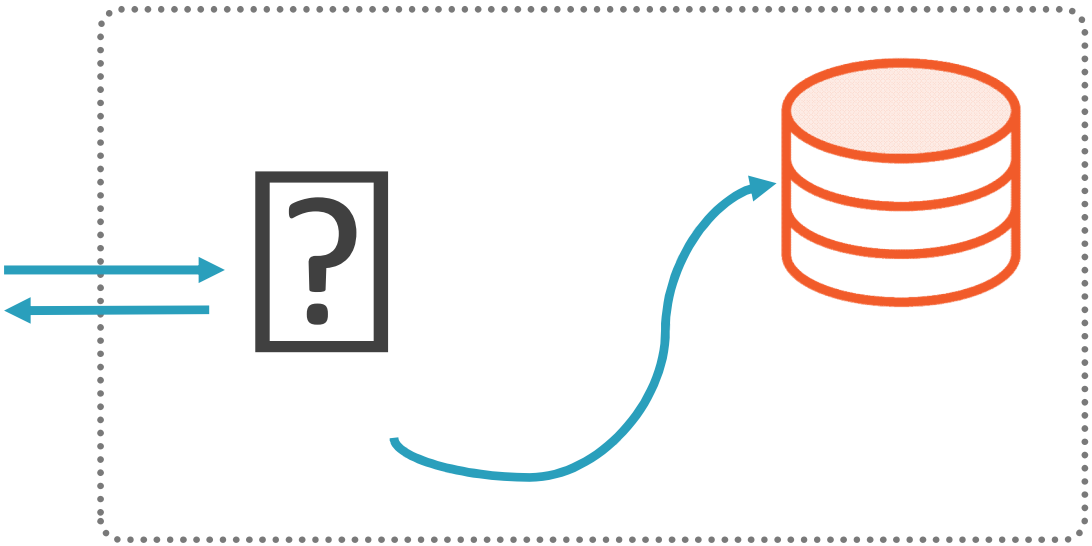

?
?
?
?
Manager
Worker
Worker
Worker

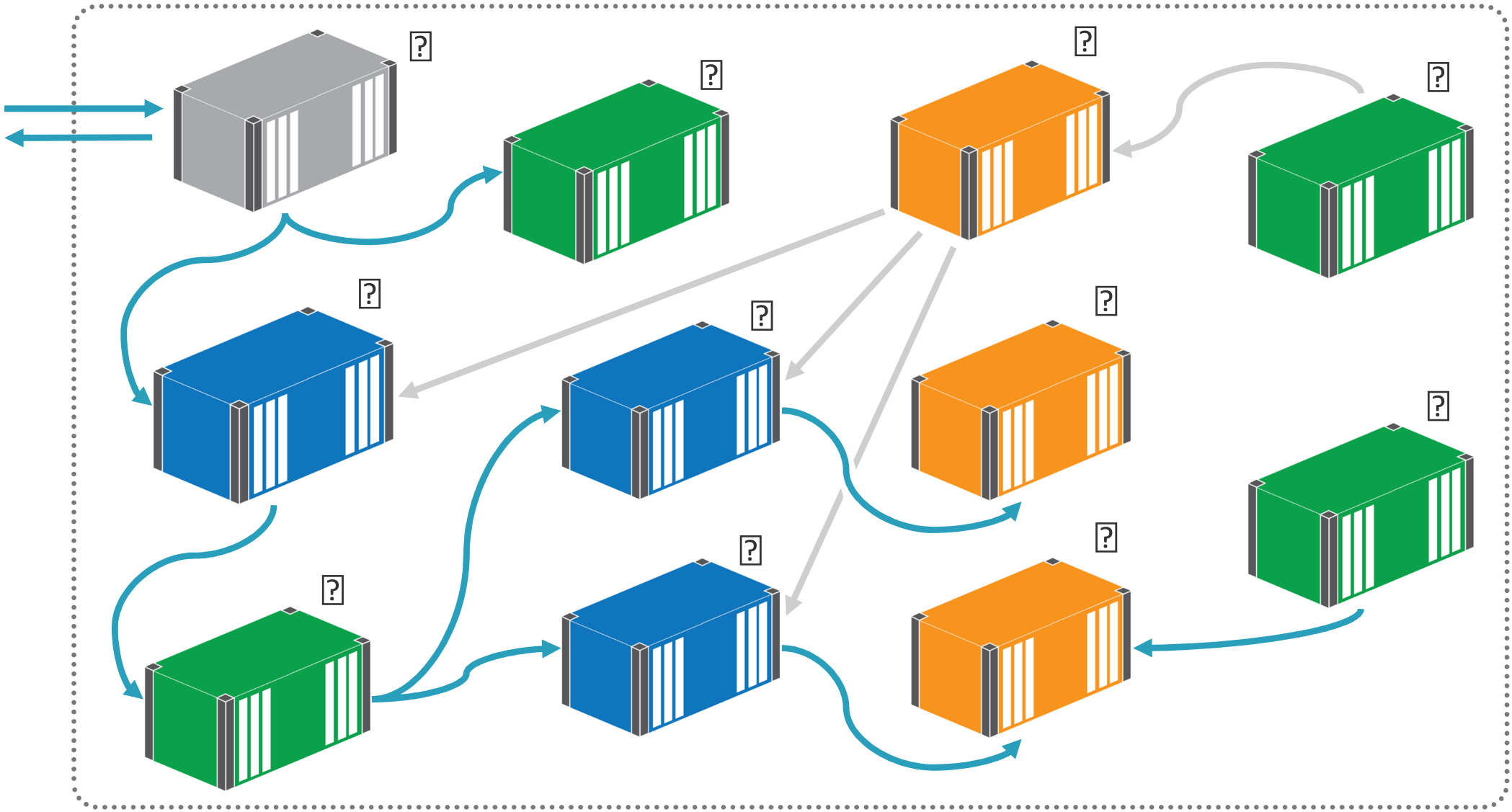

Manager
Worker
Worker
Worker

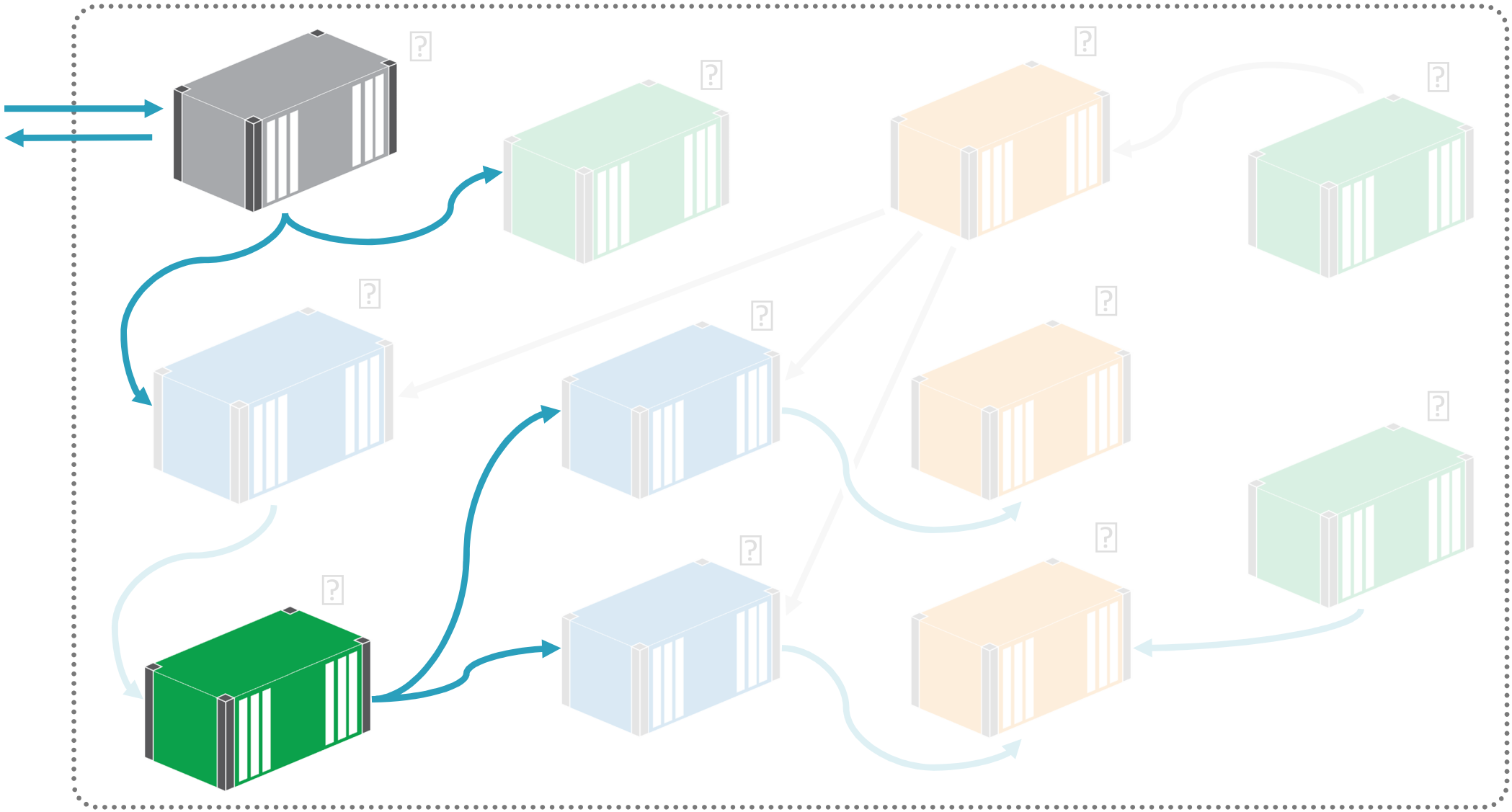

docker
SWARM

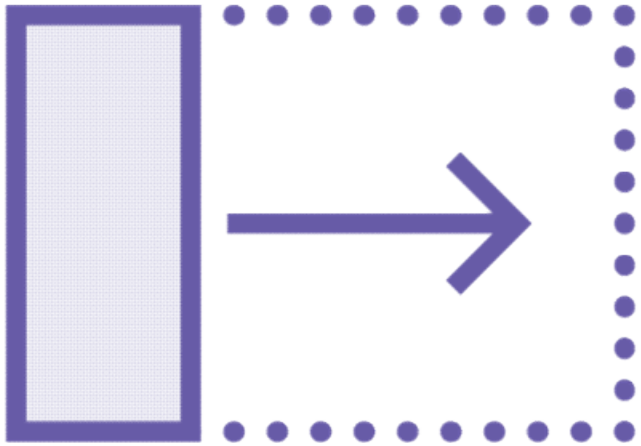



Dockerfile
Image
Dockerfile
Image
docker-compose.yml
docker
SWARM









Modernization is a Spectrum

- From migrating monoliths
- To full re-architecture

Docker is Part of Your Toolkit

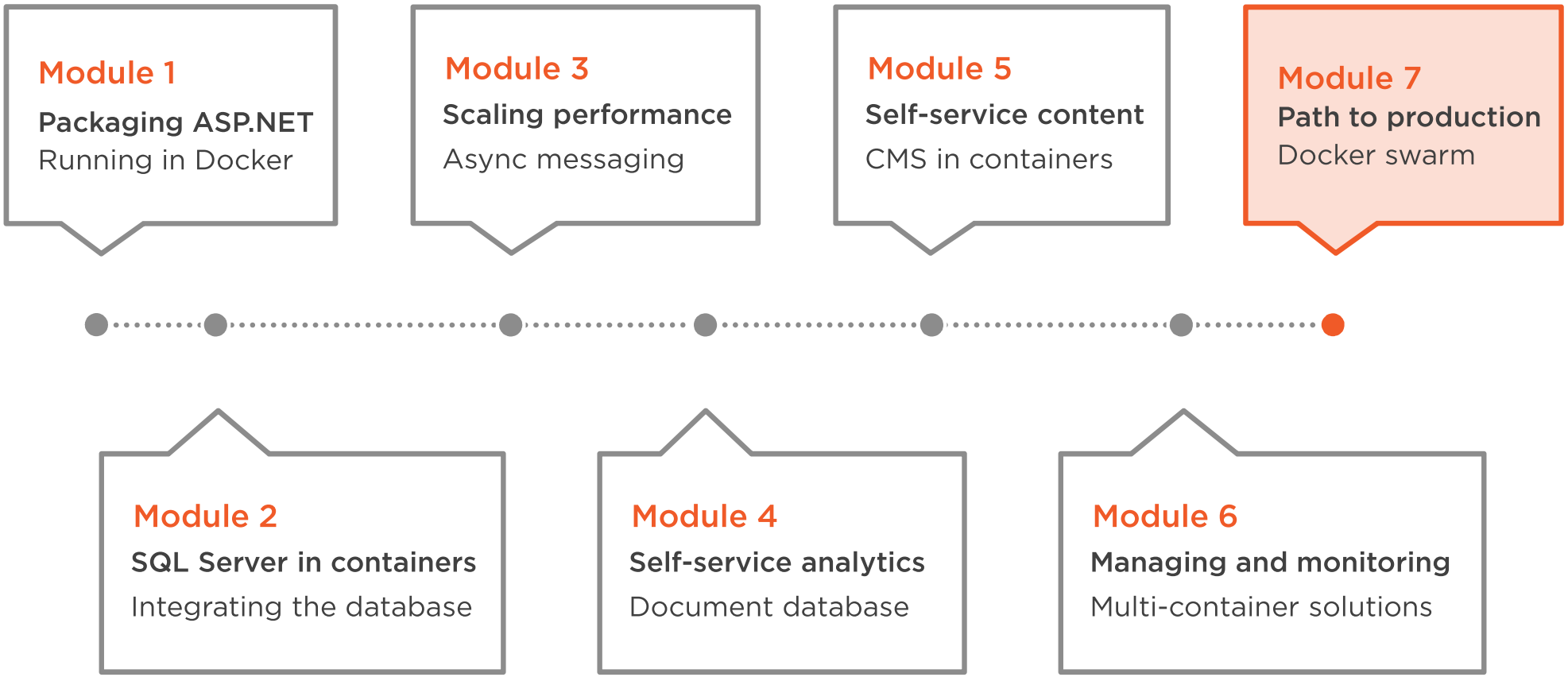
- Easy to learn and use
- Integrates with your landscape

Container-first Design

- Run third-party apps in Docker
- Manage them in the same way

Environmental consistency

- Consistent from dev to prod
- Tools, processes and artifacts


Module 1
Packaging ASP.NET
Running in Docker
Module 3
Scaling performance
Async messaging
Module 5
Self-service content
CMS in containers
Module 7
Path to production
Docker swarm
Module 2
SQL Server in containers
Integrating the database
Module 4
Self-service analytics
Document database
Module 6
Managing and monitoring
Multi-container solutions