Managing Containers with Docker Compose

Rajesh Kumar

DevOps Architect

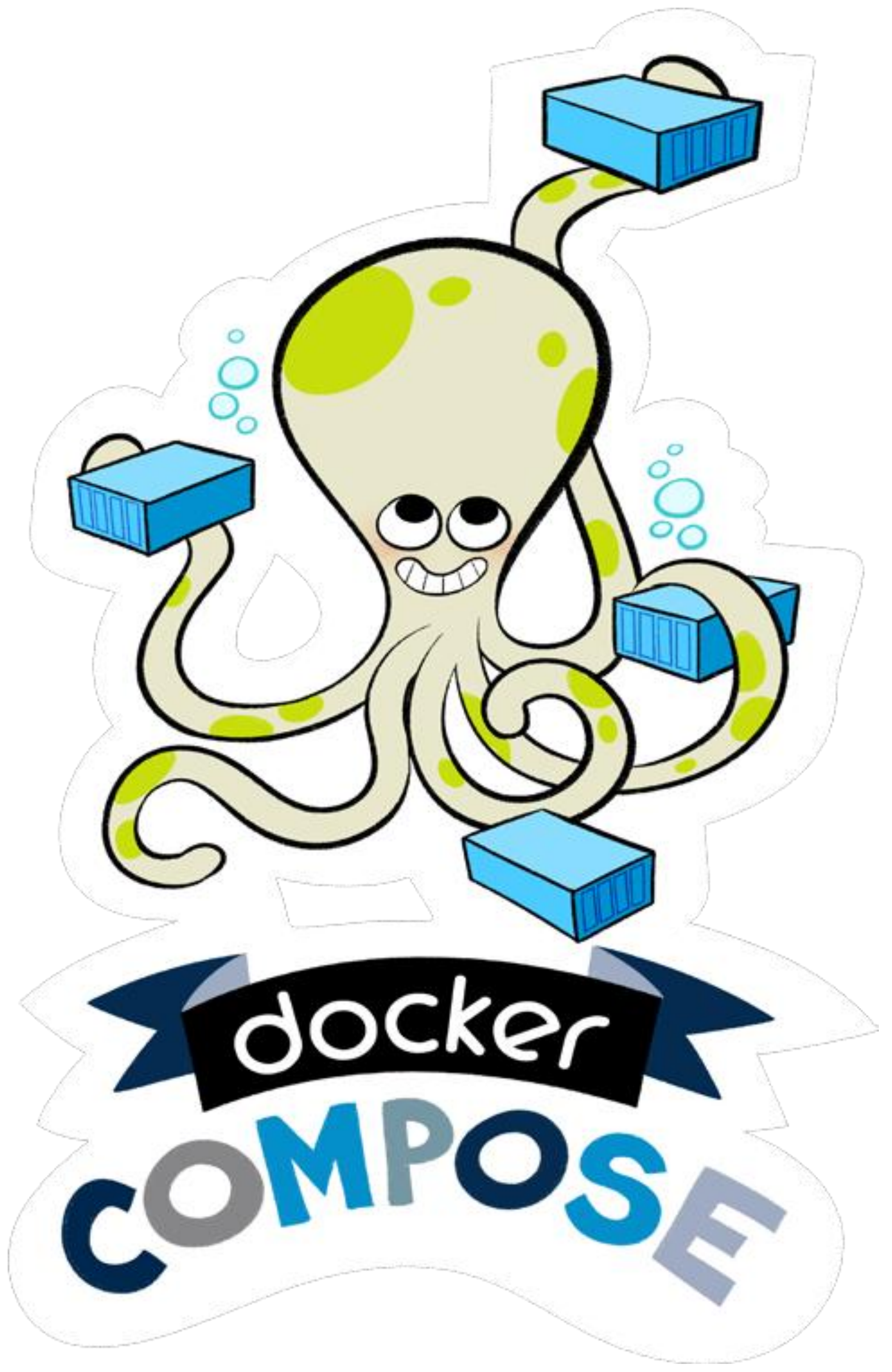
@RajeshKumarIN | www.RajeshKumar.xyz

Module Agenda

- Getting Started with Docker Compose
- The docker-compose.yml File
- Docker Compose Commands
- Docker Compose in Action
- Setting Up Development Environment Services
- Creating a Custom docker-compose.yml File
- Managing Development Environment Services

Getting Started with Docker Compose

Docker Compose Manages Your Application Lifecycle



Docker Compose Features

Manages the whole application lifecycle:

- Start, stop and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

Docker Compose Features

Manages the whole application lifecycle:

- Multiple isolated environments on a single host
- Preserve volume data when containers are created
- Only recreate containers that have changed
- Variables and moving a composition between environments

Multiple isolated environments on a single host

- Compose uses a project name to isolate environments from each other. You can make use of this project name in several different contexts:

Preserve volume data when containers are created

- Compose preserves all volumes used by your services. When docker-compose up runs, if it finds any containers from previous runs, it copies the volumes from the old container to the new container. This process ensures that any data you’ve created in volumes isn’t lost.

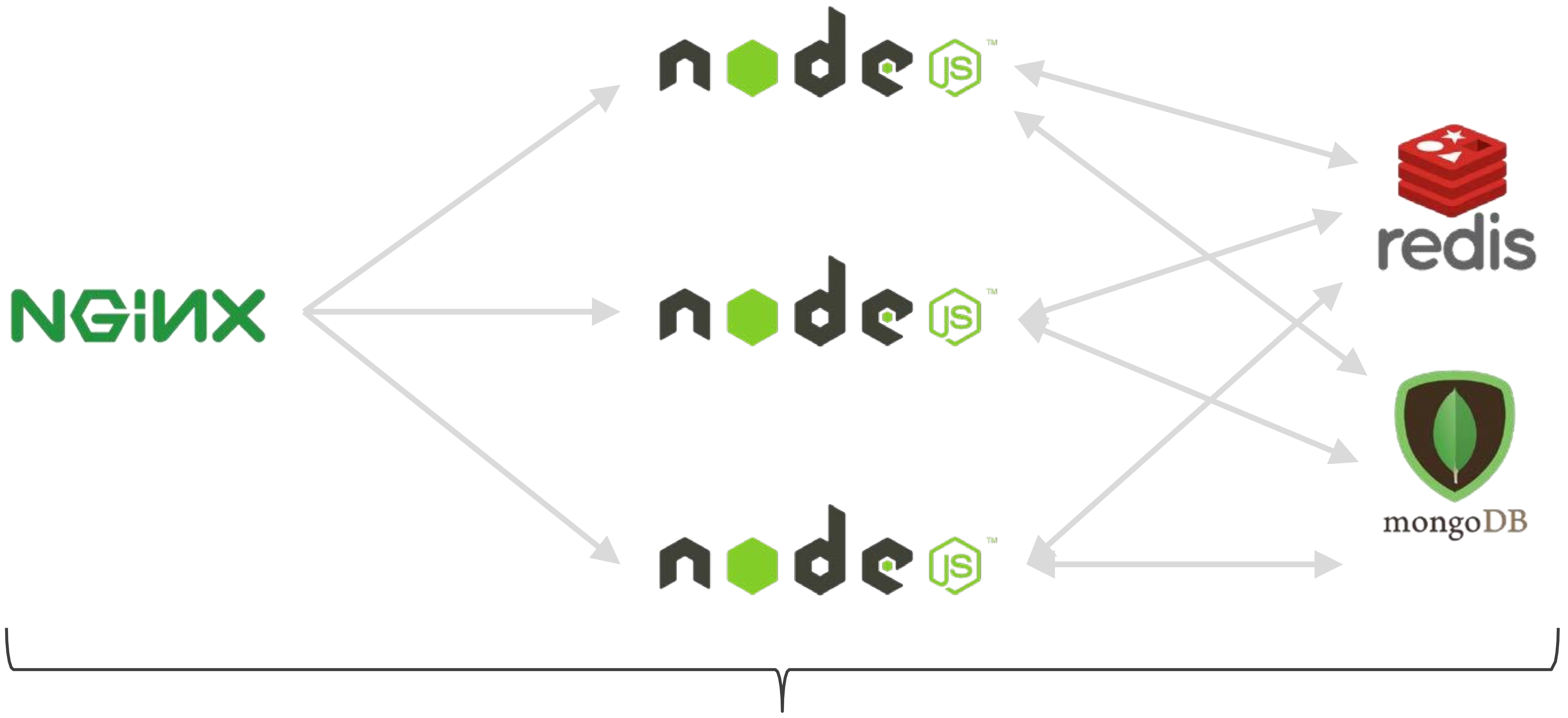
Only recreate containers that have changed

Compose caches the configuration used to create a container. When you restart a service that has not changed, Compose re-uses the existing containers. Re-using containers means that you can make changes to your environment very quickly.

Variables and moving a composition between environments

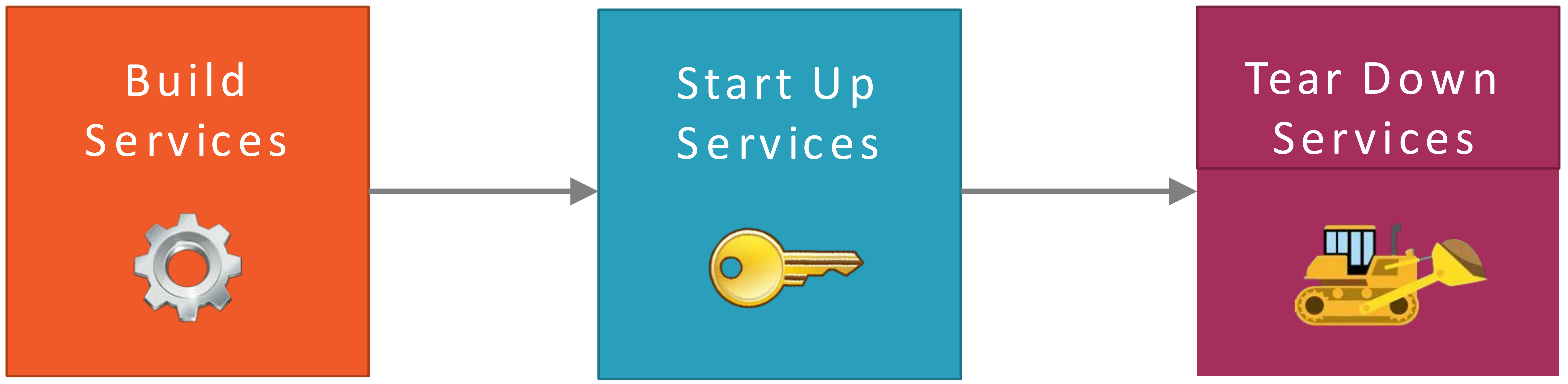
- Compose supports variables in the Compose file. You can use these variables to customize your composition for different environments, or different users.

The Need for Docker Compose



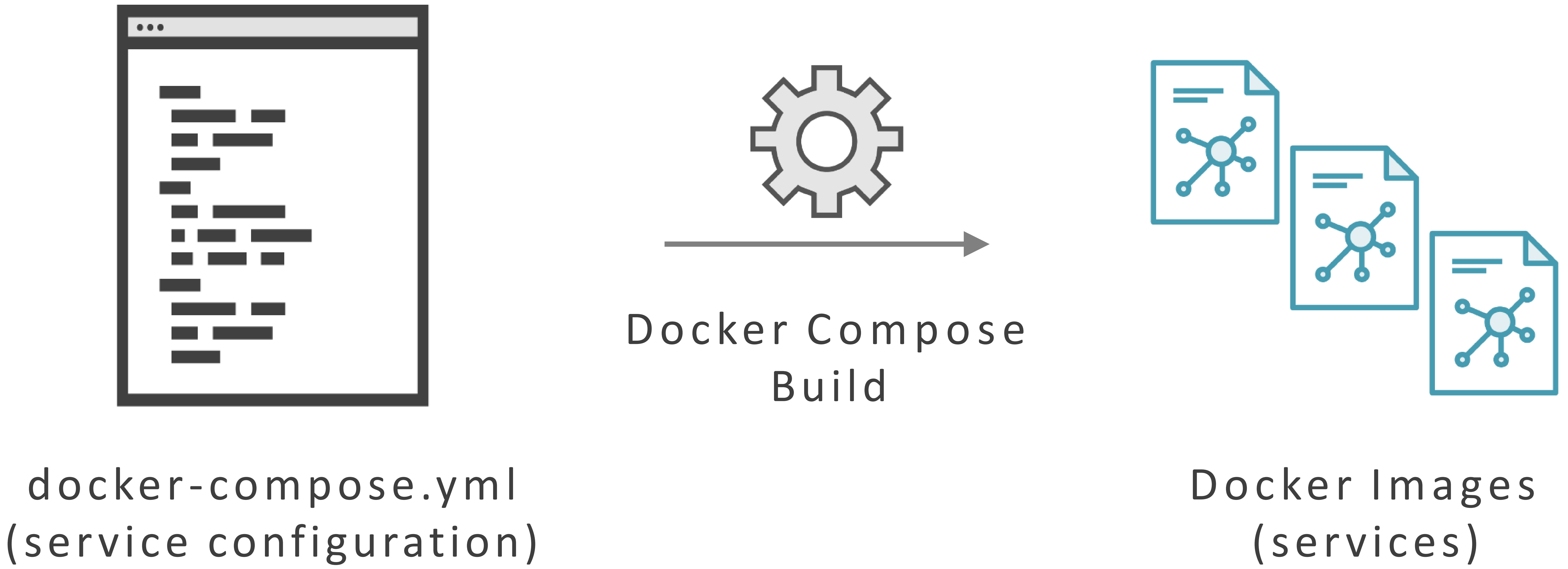
Docker Compose
(docker-compose.yml)

Docker Compose Workflow

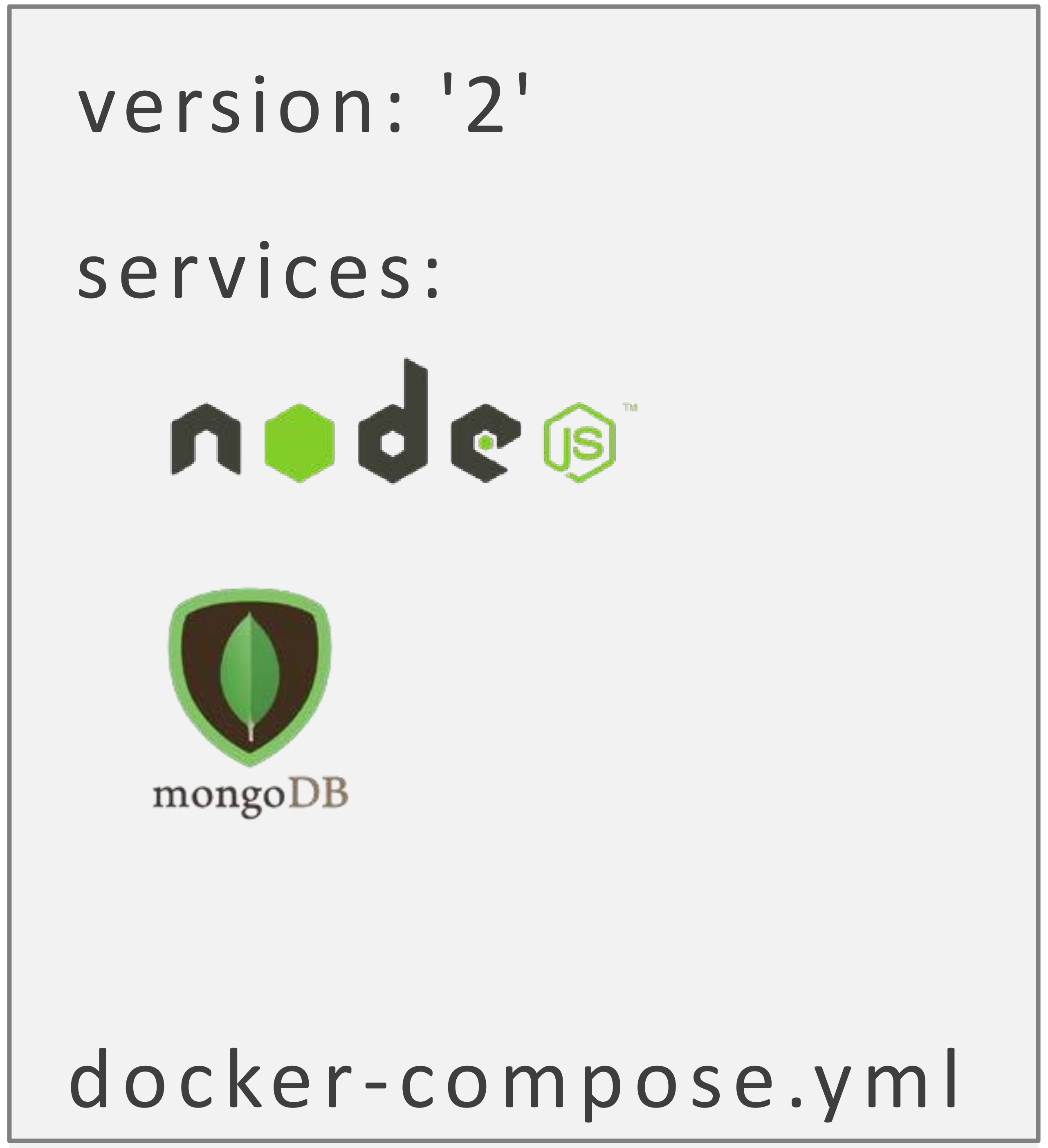


The docker-compose.yml File

The Role of the Docker Compose File



Docker Compose and Services



```
version: '2'

services:
```

docker-compose.yml

Key Service Configuration Options

build

environment

image

networks

ports

volumes

docker-compose.yml Example

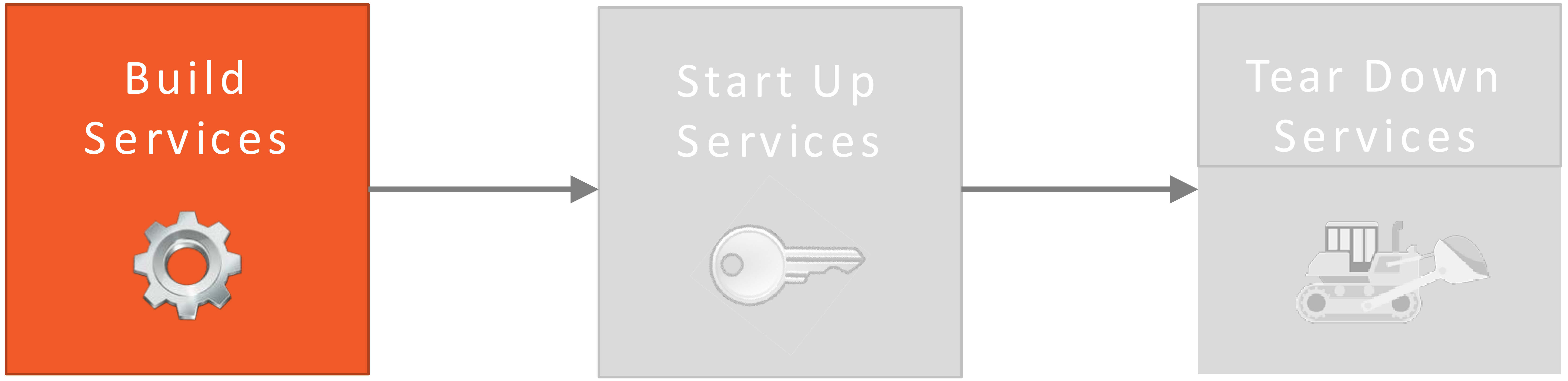
```
version: '2'
services:

  node: build:
    context: .
      dockerfile: node.dockerfile
    networks:
      -nodeapp-network

  mongodb:
     image: mongo
     networks:
       - nodeapp-network


networks:
  nodeapp-network
  driver: bridge
```

Docker Compose Commands

Key Docker Compose Commands

```
docker-compose build

docker-compose up

docker-compose down

docker-compose logs

docker-compose ps

docker-compose stop

docker-compose start

docker-compose rm
```

Building Services



Building Services

```
docker-compose build
```

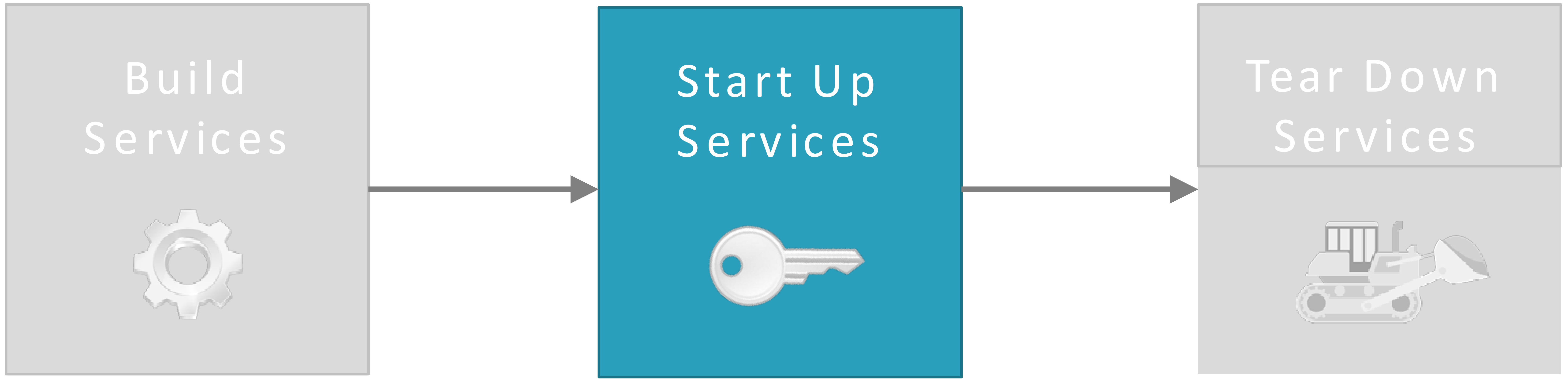
Build or rebuild services defined in docker-compose.yml

Building Specific Services

```
docker-compose build mongo
```

Only build/rebuild mongo service

Starting Services Up



Creating and Starting Containers

```
docker-compose up
```

Create and start the containers

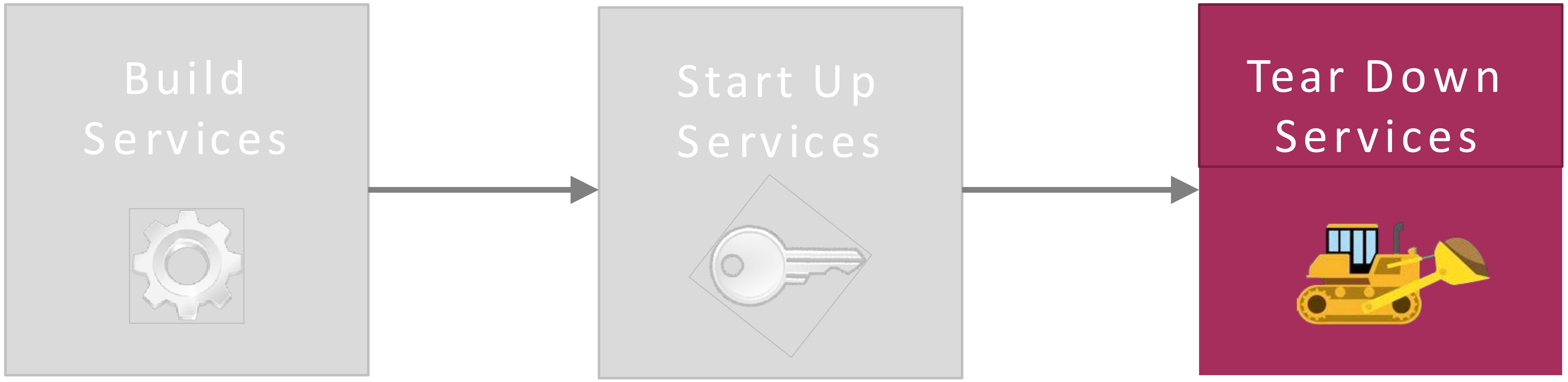
Creating and Starting Containers

Rebuild node image and stop, destroy and recreate *only* node

```
docker-compose up --no-deps node
```

Do not recreate services that node depends on

Tearing Down Services



Stop and Remove Containers

```
docker-compose down
```

Take all of the containers down (stop and remove)

Stop and Remove Containers, Images, Volumes

Remove all volumes

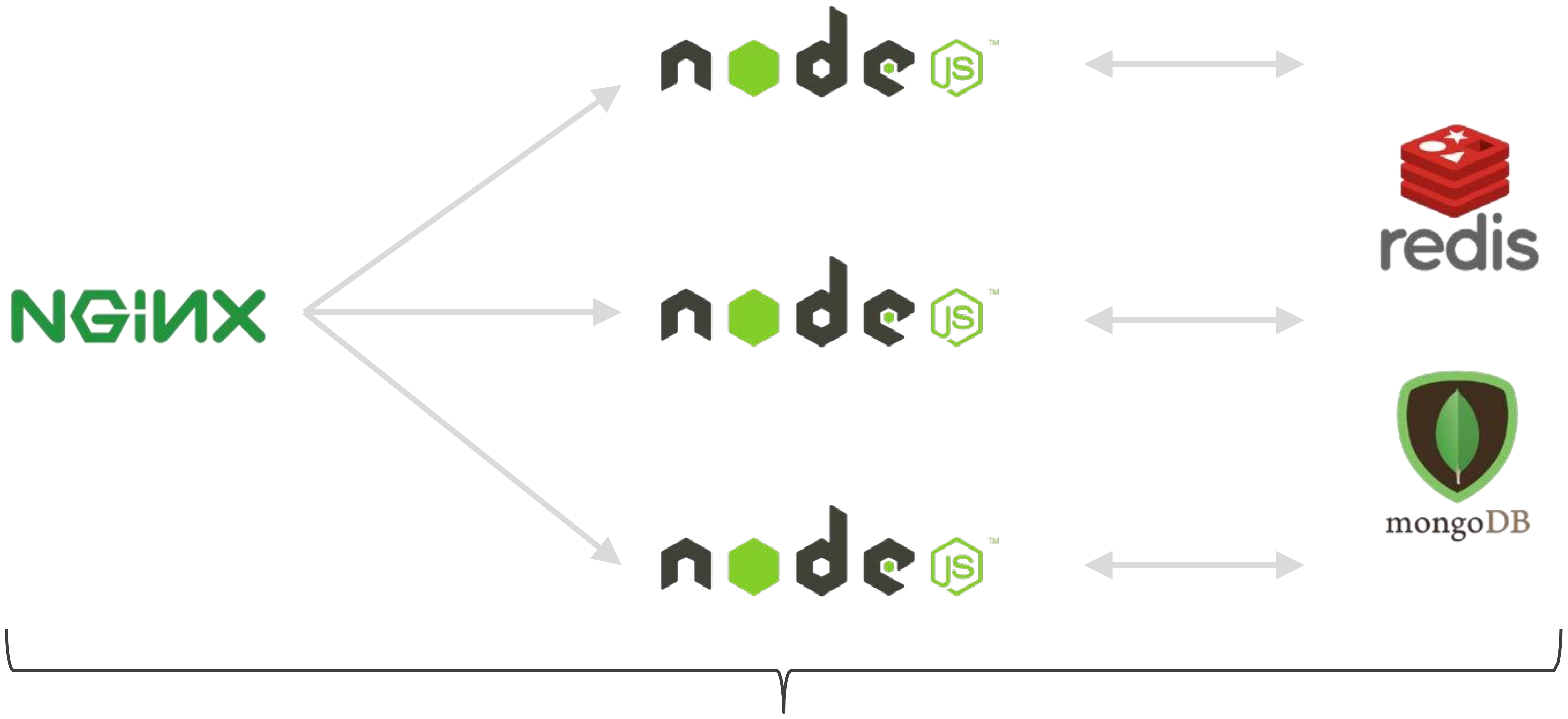
```
docker-compose down --rmi all --volumes
```

Remove all images

Docker Compose in Action

Setting up Development Environment Services

Development Environment Services



Docker Compose
(docker-compose.yml)

Creating a Custom docker-compose.yml File

Managing Development Environment Services

LAB

Thank You.