Angular Deployment to Production

Dev vs Production

Dev

Development and build speed over runtime efficiency and performance

Second Change Detection Round

Performs Deep Object Creation

Readable bundle files

Source map to view original code

Servers optimized for development and debugging: Ng Dev Server

Production

Runtime efficiency and performance

No Repeated Change Detection Round

No Deep Object Creation for Comparison

Lighter bundle files

No source map

Hashing for increased code reverse engineering security

Production servers: Express server, Apache Tomcat, Nginx

Enabling Production

```
import {enableProdMode} from '@angular/core';

if (environment.production) {
    enableProdMode();
}
```

Compilation Types

- Just-in-Time (JIT), which compiles your app in the browser at runtime
- Ahead-of-Time (AOT), which compiles your app at build time.

Why AOT Compilation?

- Faster rendering
- Fewer requests
- Smaller download size
- Catch template errors early
- Secure

AOT Compiler Options

tsconfig.json

```
{
  "compilerOptions": {
    "experimentalDecorators": true,
    ...
  },
  "angularCompilerOptions": {
    "fullTemplateTypeCheck": true,
    "preserveWhiteSpaces": false,
    ...
  }
}
```

```
ng build --target production --aot
```

The **`--target production`** Flag (build/serve)

```
ng build --target production
```

=

```
ng build --aot --output-hashing all --extract-css true --sourcemaps false
    --named-chunks false
```

--output-hashing all : Hash contents of the generated files and append hash to the file name to facilitate browser cache busting (any change to file content will result in different hash and hence browser is forced to load a new version of the file)
--extract-css true: Extracts all the css into separate style-sheet file
--sourcemaps false: Disables generation of source maps
--named-chunks false: Disables using human readable names for chunk and use numbers instead

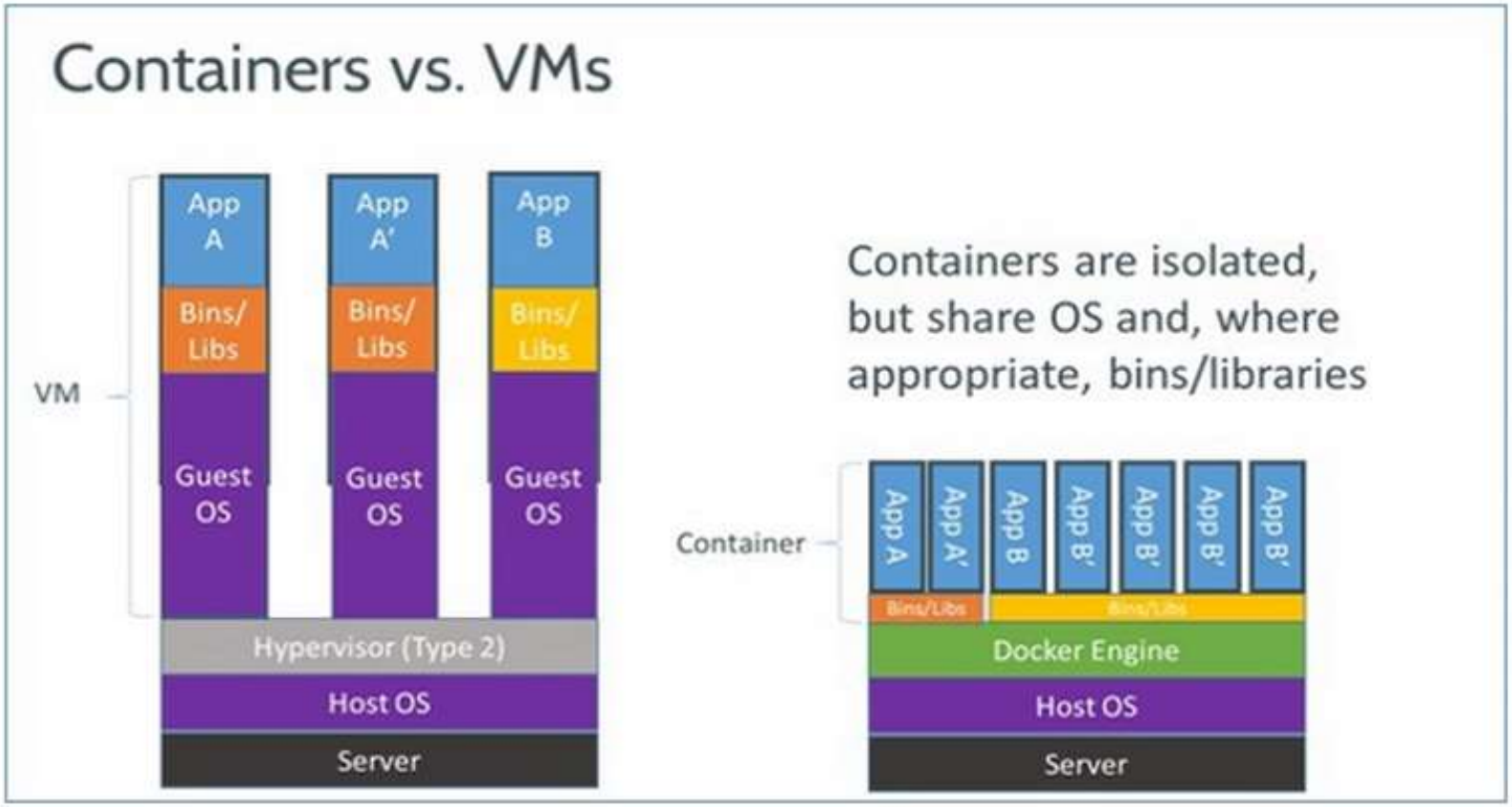
--build-optimizer: New feature which results in smaller bundles but much longer build times so use with caution! (also should be enabled by default in the future)
--vendor-chunk: Extracts all vendor (library) code into separate chunk

Why Virtual environment?

1. To easily package and ship software
2. To work in an isolated environment
3. To ensure exact versioning between dev, test, and production beds

What is Docker? Container Technology to Build, Ship, and Run Any App, Anywhere

What is Docker? Container Technology to Build, Ship, and Run Any App, Anywhere



Docker - Components

Container image: A package with all the dependencies and information needed to create a container. An image includes all the dependencies (such as frameworks) plus deployment and execution configuration to be used by a container runtime. Usually, an image derives from multiple base images that are layers stacked on top of each other to form the container's filesystem. An image is immutable once it has been created.

Container: An instance of a Docker image. A container represents the execution of a single application, process, or service. It consists of the contents of a Docker image, an execution environment, and a standard set of instructions.

The Dockerfile

Dockerfile: A text file that contains instructions for how to build a Docker image.

```
FROM node:6.9.1

RUN mkdir -p /usr/src/sportsstore

COPY dist /usr/src/sportsstore/app

COPY authMiddleware.js /usr/src/sportsstore/
COPY data.js /usr/src/sportsstore/
COPY deploy-server.js /usr/src/sportsstore/server.js
COPY deploy-package.json /usr/src/sportsstore/package.json

WORKDIR /usr/src/sportsstore

RUN npm install

EXPOSE 3000
EXPOSE 3500

CMD ["npm", "start"]
```

Build the Container Image

Build: The action of building a container image based on the Dockerfile, plus additional files in the folder where the image is built.

You build images with the **docker build** command.

```
docker build . -t sportsstore -f Dockerfile
```

Start the Contatiner

```
docker run -p 3000:3000 -p 3500:3500 sportsstore
```

Access the Application

Access the Application:

`http://192.168.99.100:3000/store`

Docker Cheat Sheet

Pull Image `docker pull mvertes/alpine-mongo`	View Images `docker images`	Run Container `docker run <image name>` *To run in background use -d: `docker run -d <image name>`*
List Running Containers `docker ps`	Stop container `docker stop <container_id>/<container_name>`	View Container Logs `docker log <container_id/name>` *Use -f to view log real time*
Remove all stopped containers `docker ps -aq --no-trunc | xargs docker rm`	Remove container by name `docker rm $(docker ps -aq --filter name=myContainerName)`	
List all Exited Containers `docker ps -aq -f status=exited`		

Port Forwarding
`docker run -d -p <host_port>:<guest_port> <image name>`
`docker run -d -p 2700:2700 <image name>`		

Contact Me

Simanta Sarma

Senior Mentor StackRoute

simanta.sarma@stackroute.in
ximanta.sarma@gmail.com

(M) 9972623673

https://simantas.wordpress.com/
https://github.com/ximanta