Notes

AWS Project Documentation

Containerized Web Application Deployment on AWS Elastic Beanstalk

@sAnandha

Notes

DEPLOY AN APP WITH DOCKER

What is Docker?

Docker is a platform that enables developers to build, package, and distribute applications as containers. A container is a lightweight, standalone, executable package that includes everything needed to run an application: code, runtime, system tools, libraries, and settings.

Why Do We Need Docker?

1. **Consistency**: Eliminates "it works on my machine" problems by packaging dependencies with the application
2. **Isolation**: Applications run in isolated environments, preventing conflicts
3. **Efficiency**: Containers share the host OS kernel, making them more lightweight than virtual machines
4. **Portability**: Containers run the same way regardless of the infrastructure
5. **Scalability**: Easy to scale applications horizontally by deploying multiple container instances.

Notes

Common Uses of Docker

- Application Deployment: Consistent deployment across development, testing, and production.
- Microservices Architecture: Deploying individual services as separate containers.
- Development Environments: Creating standardized development environments for teams.

What is AWS Elastic Beanstalk?

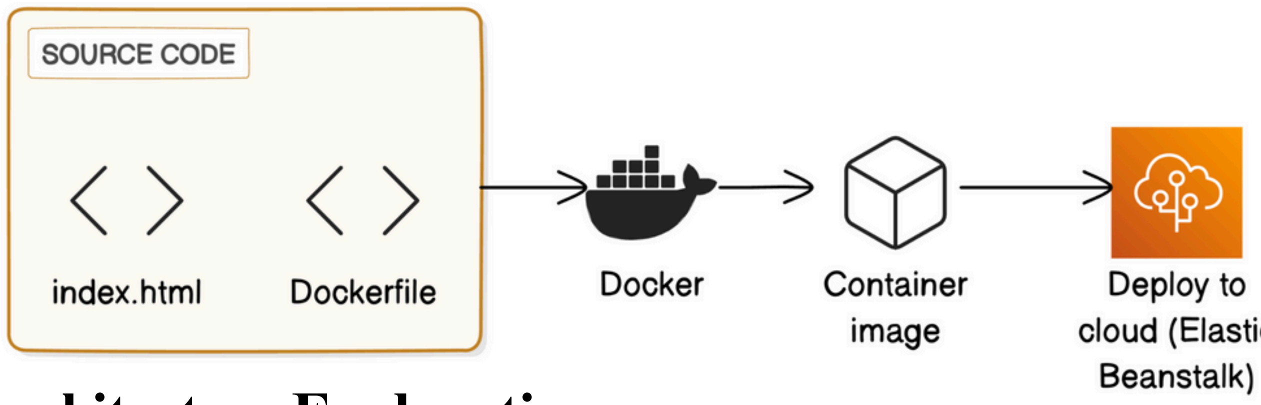
AWS Elastic Beanstalk is a Platform as a Service (PaaS) offering that simplifies deploying and managing applications. It automatically handles infrastructure provisioning,scaling and monitoring while allowing developers to retain full control over AWS resources powering their application.

Uses of AWS Elastic Beanstalk

- Rapid Application Deployment: Deploy web applications without managing infrastructure.
- Multiple_Language_Support:Supports applications in Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker.
- Auto-scaling: Automatically adjusts capacity based on application needs.

Notes

ARCHITECTURE DIAGRAM :



Architecture Explanation

1. **Source Code** (represented by index.html and Dockerfile):
   - The starting point containing your web application files (like index.html)
   - A Dockerfile that defines how to package your application
2. **Docker** (represented by the Docker whale logo):
   a. The containerization platform that processes your Dockerfile
   - Converts your application and its dependencies into a standardized package
3. **Container Image** (represented by the 3D cube):
   - A portable, executable package containing your application, runtime, libraries, and settings
4. **Deploy to Cloud** :
   - The final destination where your containerized application runs
   - AWS Elastic Beanstalk handles the infrastructure, scaling, and management

@sAnandha

Notes

Steps to Build the Solution

1. **Prepare the Source Code**
    - Organize your web application files (HTML, CSS, JavaScript, backend code, etc.)
    - Ensure your application works locally
    - Create project dependencies files (package.json, requirements.txt, etc. depending on your technology stack)

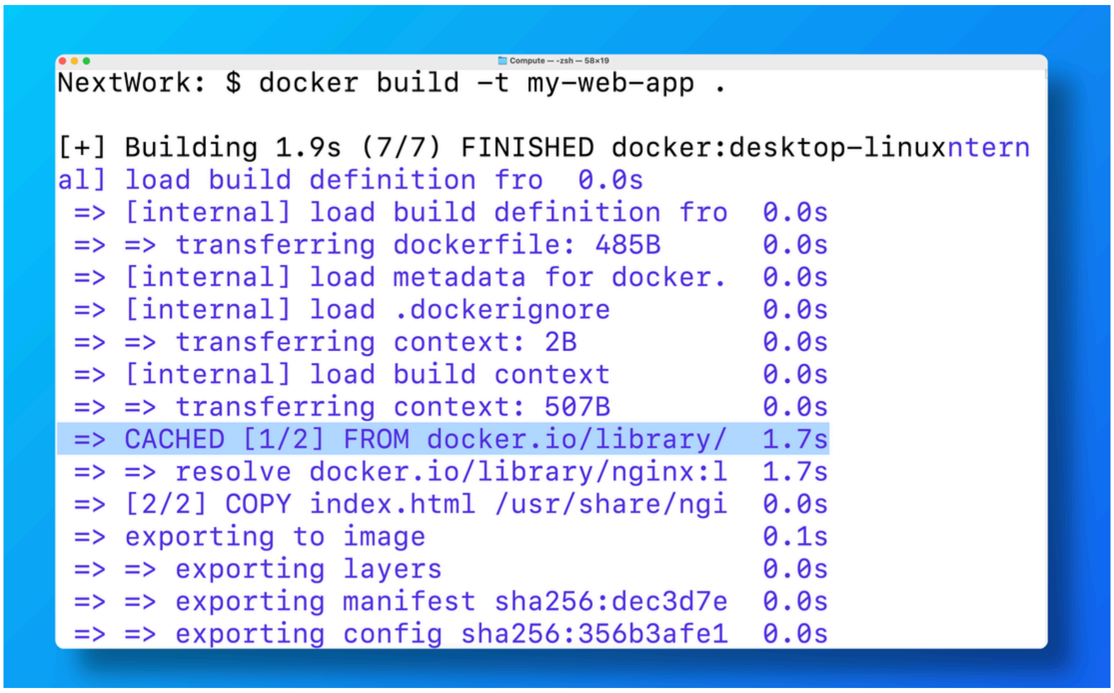
2. **Create a Dockerfile**
    - Write a Dockerfile in your project root that defines:
        - Base image (e.g., Node.js, Python, etc.)
        - Working directory
        - Dependencies installation
        - Application file copying
        - Ports to expose
        - Startup command
3. **Build the Docker Image**
    - Install Docker on your development machine
    - Open a terminal in your project directory
    - Build the image using Docker CLI
    - Test the container locally to ensure it works as expected

Notes

4. **Set Up AWS Requirements**
    - Create an AWS account if you don't have one
    - Install and configure AWS CLI
    - Install the EB CLI (Elastic Beanstalk Command Line Interface)
    - Create necessary IAM roles and permissions
5. **Configure Elastic Beanstalk**
    - Create a new Elastic Beanstalk application
    - Choose the Docker platform
    - Create a Dockerrun.aws.json file (if using a pre-built image from a registry)
    - Configure environment settings:
        - Instance type
        - Environment type (single instance or load balanced)
        - Security groups
        - Environment variables
6. **Deploy Your Application**
    - Upload your container image to a registry (Amazon ECR, Docker Hub, etc.) if you're not building on Elastic Beanstalk
    - Deploy your application using:
        - AWS Management Console
        - AWS CLI

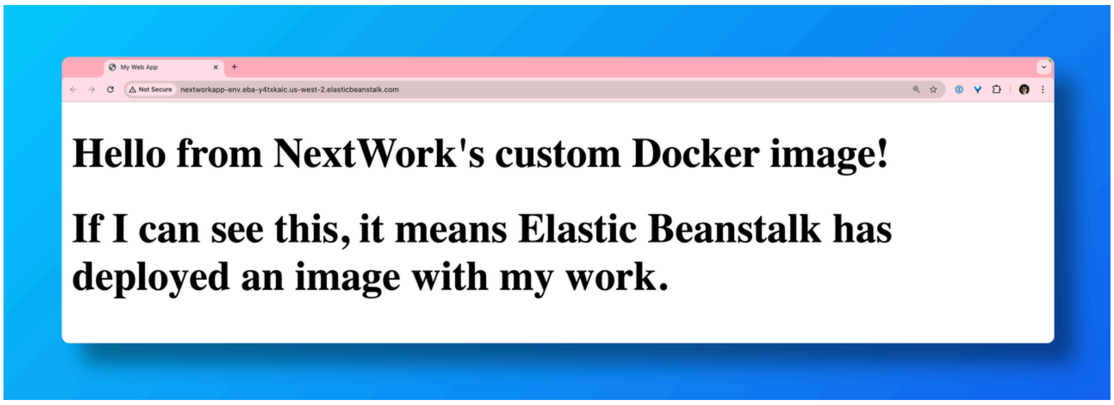
Notes

SAMPLE OUTPUT :



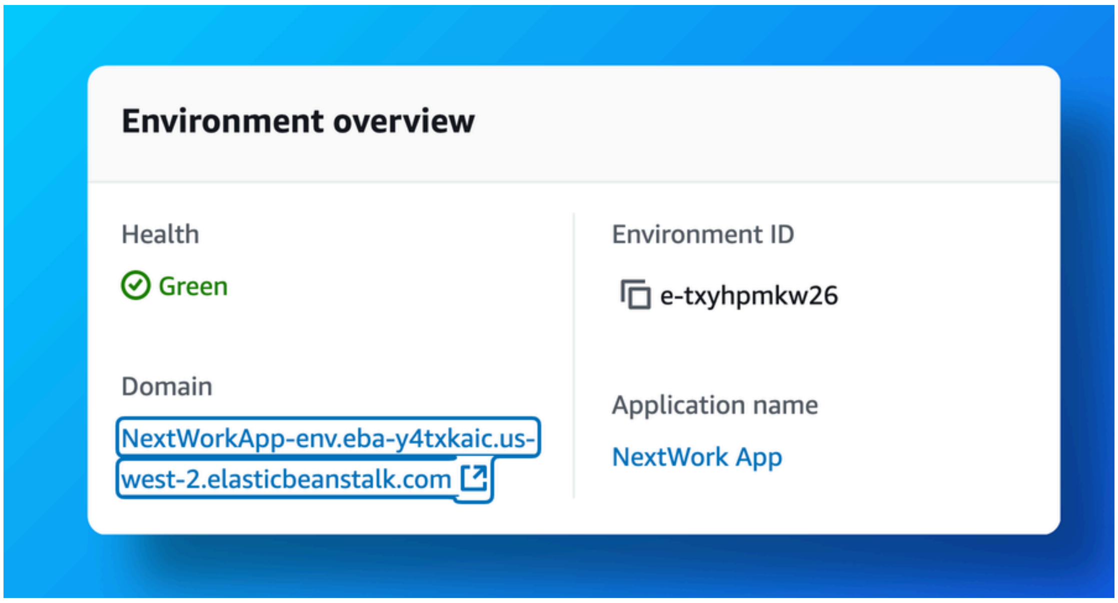
```
NextWork: $ docker build -t my-web-app .

[+] Building 1.9s (7/7) FINISHED docker:desktop-linuxntern
al] load build definition fro  0.0s
 => [internal] load build definition fro  0.0s
 => => transferring dockerfile: 485B      0.0s
 => [internal] load metadata for docker.  0.0s
 => [internal] load .dockerignore         0.0s
 => => transferring context: 2B           0.0s
 => [internal] load build context         0.0s
 => => transferring context: 507B         0.0s
 => CACHED [1/2] FROM docker.io/library/  1.7s
 => => resolve docker.io/library/nginx:l  1.7s
 => [2/2] COPY index.html /usr/share/ngi  0.0s
 => exporting to image                    0.1s
 => => exporting layers                   0.0s
 => => exporting manifest sha256:dec3d7e  0.0s
 => => exporting config sha256:356b3afe1  0.0s
```



Hello from NextWork's custom Docker image!

If I can see this, it means Elastic Beanstalk has deployed an image with my work.



Environment overview

Health
✓ Green

Environment ID
e-txyhpmkw26

Domain
NextWorkApp-env.eba-y4txkaic.us-west-2.elasticbeanstalk.com

Application name
NextWork App

For References :
https://learn.nextwork.org/projects/aws-compute-eb?track=high

@sAnandha