AWS Microservices and Containers

Course Link - [TOC - Developing on AWS using Microservices & Containers | Lex (infosysapps.com)](https://lex.infosysapps.com/web/en/app/toc/lex_auth_012702124759810048546/overview)

Objective- I:

* Cloud Native Build
* Containerization using Docker

**Cloud Native Build**

* Updates Fast
* Adopt changes
* Deliver Fast
* Go Global
* Want to get maximum benefits from cloud
* App is going to run on Cloud Server
* You will have less control

**4 – Key Tenet of Cloud Native Application**

1. Service Based Architecture
2. API driven communication
3. Container based infrastructure
4. DevOps Process
5. **Service Based Architecture**

* Use available services
* Microservices Architecture – loosely coupled
* How microservices will communicate??? – API
* Service is easier to deploy and maintain
* Scaled independently / flexible scaling
* Remove hard dependency

1. **API driven communication**

* All services should communicate through the API
* Services are exposed through via APIs
* Avoid the risk of direct linking
* Extend reach of services to different devices

1. **Container Based Architecture**

* Bundle all dependencies packaged together
* Scale apps horizontally

1. **DevOps Process**

* Dev and Ops team need to come together
* Build and deliver apps fast
* Agile Methodology

**12-Factor App Methodology** [The Twelve-Factor App (12factor.net)](https://12factor.net/)

* 12 Best Practices suggested by Developer
* Cloud-Native Approach

Why???

* + Be automated
  + Having maximum portability
  + Deployed on cloud
  + Continuous Deployment
  + Scale Up

1. **CodeBase**

* There is only one codebase per app, but there will be many deploys of the app.
* Codebase is same across all deploys / one codebase per app

1. **Dependencies**

* We need to remove system dependencies
* Package it into - Containers

1. **Config**

* Store config in the environment
* strict separation of config from code/app

1. **Backing Service**

* Likes App Database, Queuing System & Caches
* Treat backing services as attached resources

1. **Build, Release & Run**

* Strictly separate
* One stage failure should handle independently

1. **Process**

* Executes the app as stateless process

1. **Port Binding**

* Export services through / via ports only / binding

1. **Concurrency**

* Scale out / process model

1. **Disposability**

* Maximize robustness with fast startup and graceful shutdown

1. **Dev/Prod Parity**

* Minimize the gap (time gap /personal gap / tool gap)
* Keep equal

1. **Logs**

* Treat logs as event streams
* Logs provide visibility in the behavior of the running app

1. **Admin Processes**

* Run admin/management tasks as one-off processes

**Containerization using Docker**

Scenario

Test Env.

(python 2.7)

Debian 7

Production Env.

(python 3.7)

RHEL 7.4

Staging Env.

(python 3.1)

Debian 9

Dev Environment

(Python 2.7)

Win 8 /10

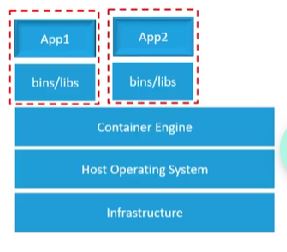
Think?

* Unexpected Outcome
* Something weird will happen

**Container** comes with resolution

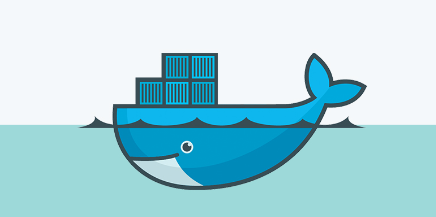
* Container is a standardized unit of software, that packages your application and associated dependencies, which allows the app run seamlessly
* Container ensures quickly, reliably, and consistent deployment,

regardless of environment

****

**Docker**

* It is a containerization engine



* Designed to benefit both Developers and System Admins
* Provide agility and security
* Faster time to market
* Remove Dependencies

**Demo-1: Installation and Configuration of Docker** [Labguide on Installation and Configuration of Docker - Viewer Page | Lex (infosysapps.com)](https://lex.infosysapps.com/web/en/viewer/pdf/lex_auth_0132375211398266887684?collectionId=lex_auth_012702124759810048546&collectionType=Course&pathId=lex_auth_0132332961466368001928)

**Build an Application Image**

**Docker File**

* + Text File
  + Set of instructions to build docker image

**Docker Image**

* + Read Only File
  + Can be used to build one or more containers

**Docker Repository**

* + It is registry
  + Help to store, retrieving and managing docker images

**Demo-2:** [Labguide to create docker file - Viewer Page | Lex (infosysapps.com)](https://lex.infosysapps.com/web/en/viewer/pdf/lex_auth_0132375289261588487685?collectionId=lex_auth_012702124759810048546&collectionType=Course&pathId=lex_auth_0132332961466368001928)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Life Cycle of Application Containerization**

* Pull image
* Create Container
* Connect to container
* Customize Container
* Commit the changes
* Tag Image
* Push to Docker Repository (docker hub)

**Demo-3:** [Labguide on Life cycle of Application Containerization - Viewer Page | Lex (infosysapps.com)](https://lex.infosysapps.com/web/en/viewer/pdf/lex_auth_0132375275679662087687?collectionId=lex_auth_012702124759810048546&collectionType=Course&pathId=lex_auth_0132332961466368001928)