Cloud Native Applications

Cloud Computing

Amr Eid | Cloud Architect Lead, Cloud Platform, MEA | amreid@eg.ibm.com
By 2021, 94 percent of workloads and compute instances will be processed by cloud data centers.¹

¹ Cisco Global Cloud Index 2018
By 2021, 73 percent of the cloud workloads and compute instances will be in public cloud data centers, up from 58 percent in 2016.²

² Cisco Global Cloud Index 2018
Cloud, defined...

“Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

NIST
Cloud service models

Today's apps must keep up with the speed of the app revolution
Cloud growth has reasons...

- Applications with a **short lead time to delivery**.
- **Predefined services** for developers.
- Scaling Apps up and down dynamically.
- **Flexible billing** (pay-as-you-go).
Deploy **the right type of cloud** to accelerate innovation

**Private**
- When to use: Organization has requirement to utilize on-premises infrastructure (security policy, investment equity, limited WAN bandwidth, etc).

**Dedicated**
- When to use: Organization has isolation requirement for security and/or performance reasons.

**Public**
- When to use: Organization seeks the security and compliance benefits of public cloud.
Compute Options

- **Bare Metal**: Maximum performance and control
- **Virtual Server**: Leverage existing languages and tools
- **Containers**: Maximum portability
- **Platform as a Service**: Extensive runtime options
- **Serverless**: Maximum speed with serverless apps
Cloud Native Applications
Cloud Native

An **application architecture** designed to leverage the strengths of a **standardized** cloud environment.

Continuous Integration, Container Engines and Orchestrators.
# The Twelve-Factor App

<table>
<thead>
<tr>
<th>I. Codebase</th>
<th>VII. Port binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>One codebase tracked in revision control, many deploys</td>
<td>Export services via port binding</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>II. Dependencies</th>
<th>VIII. Concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicitly declare and isolate dependencies</td>
<td>Scale out via the process model</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>III. Config</th>
<th>IX. Disposability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store config in the environment</td>
<td>Maximize robustness with fast startup and graceful shutdown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IV. Backing Services</th>
<th>X. Dev/prod parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treat backing services as attached resources</td>
<td>Keep development, staging, and production as similar as possible</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V. Build, release, run</th>
<th>XI. Logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strictly separate build and run stages</td>
<td>Treat logs as event streams</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VI. Processes</th>
<th>XII. Admin processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute the app as one or more stateless processes</td>
<td>Run admin/management tasks as one-off processes</td>
</tr>
</tbody>
</table>
Microservices

Engineering Approach

focused on decomposing an application into single-function modules with well defined interfaces which are independently deployed.

Operated by small teams who own the entire lifecycle of the service.
Microservices architecture

Simplistically, microservices architecture is about breaking down large silo applications into more manageable, fully decoupled pieces.

A microservice is a granular, decoupled component within a broader application.
Monolithic vs microservices architecture

**Monolithic application**
- Services
- Repositories
- Relational database

**Server side**
- Create, retrieve, update, and delete
- Code is run in the same process

**Client side**
- JSON
- Notifications
- HTTP, WebSocket

**Web client**
- IoT
- Mobile client

**API gateway**
- Service
- Service
- NoSQL

**Client side**
- JSON
- Notifications
- HTTP, WebSocket

**Server side**
- Publish
- Message queue
- Subscribe

“An Overview of Microservices Architecture” by Khoa Dinh
http://khoadinh.github.io/2015/05/01/microservices-architecture-overview.html
Advantages of microservices

In a word: **Independent**

- Single small team to develop and support.
- timeline/ language and technology is totally *independent*.
- Data Management is *independent*.
- Scales and fails faster and *independent*.
- New versions delivered *independently* of other services.
Containers

A **standard** way to package an application and all its dependencies so that it can be moved between environments and run without changes.

Containers work by **isolating** the differences between applications inside the container so that everything outside the container can be standardized.
Docker Containers
Vs Traditional Virtualization
is an **open platform** for building distributed applications for **developers** and **system administrators**

![Diagram showing the processes of Build, Ship, and Run with various logos of different platforms, including ubuntu, MySQL, MongoDB, Node, CentOS, redis, NGINX, vmware, openstack, and IBM Bluemix.]

Any App

Anywhere
Container Orchestration

Which container orchestration tools does your organization use?

ANSWERED: 214

Source: devops.com
What is Kubernetes?

- Container orchestrator
- Runs and manages containers
- Inspired and informed by Google's experiences and internal systems
- 100% Open source, written in Go
- Manage applications, not machines
- Rich ecosystem of plug-ins for scheduling, storage, networking
Thank you!