Serverless and APIs: Rethinking Curriculum in Higher Education

Munir Mandviwalla and Jeremy Shafer
Temple University
Serverless

Serverless computing refers to the concept of building and running applications that do not require server management. It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

Source: CNCF Serverless Whitepaper v1.0
An API is a software component that — in terms of its operations, inputs, outputs, and underlying data types — implements a function that is independent of its "host" app.

Source: Malinverno, P. Gartner Report, 2017
The end of the AA’s?

Serverless

API
Challenges

- Secure configuration requires sophisticated understanding
- Connectivity may be unreliable
- Function “cold starts” may cause hiccup on start
- Debugging and error recovery is harder
- Identity – multiple actors involved in an API call
- Triggering lots of functions may lead to inefficiencies
But

Cost
• Bring in talent embedded in the API
• Outsource hardware, data center, database management

Opportunity
• Use APIs to open/test new markets
• Scale quickly
• Loosely coupled integration enables exploration
  • Business rules and processes are embedded in the API
  • Enable business units – no servers, databases, security – new era of “PC”? 
The (near) future

- **Identity management**: API gateway does the job or call a verify API
- **Increase security**: API gateway does the work and the serverless vendor cares more
- **Achieve scalability**: Function and user level with serverless
- **Expand scope**: Experience APIs for mobile vs. desktop
- **Data is simpler**: Gone: database server, even normalization?
- **Bring in new capabilities**: Microservices: Internal APIs and External APIs can play well together
- **Bugs are outsourced***: Middleware layer is somebody else’s problem
- **Built in Project management**: Functions are automatically versioned
- **Processes are key***: Scope and interdependency of functions
- **Code is more efficient**: Charge by use exposes bad code quickly
Current state of the curriculum

Introductory Course
- Business processes
- Enterprise systems
- Platforms and cloud computing

Enterprise architecture
- Network architecture
- Server technologies
- Computing platforms

Data analytics
- Transactional databases and SQL
- Enterprise data architectures
- Drawing insight from data using analytics

Application development
- Cloud-based architectures
- Data-driven applications
- Web interfaces

Digital design and innovation
- Requirements gathering
- Process mapping
- Use cases and scenarios

Change leadership
- Project management
- Change management

IT value and service delivery
- Management of the IT function
- Disruptive innovation
- Systems thinking
### Implications for entire curriculum

<table>
<thead>
<tr>
<th>Introductory Course</th>
<th>Enterprise architecture</th>
<th>Data analytics</th>
<th>Application development</th>
<th>Digital design and innovation</th>
<th>Change leadership</th>
<th>IT value and service delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process thinking</td>
<td>Http and TCP</td>
<td>Data as a service</td>
<td>Cloud</td>
<td>Designing modular processes</td>
<td>Manage use of APIs</td>
<td>Strategic partnerships: Vendors and APIs</td>
</tr>
<tr>
<td>Evolving role of IT</td>
<td>Virtual networks</td>
<td>Data lakes</td>
<td>API-based</td>
<td>Build/buy at service level</td>
<td>DevOps</td>
<td>Systems thinking</td>
</tr>
<tr>
<td>Distributed technologies (i.e., Blockchain)</td>
<td>_aaS</td>
<td>Analytics as a Service</td>
<td>Microservices</td>
<td>Integrating external and internal APIs</td>
<td>Managing infrastructure shifts</td>
<td>Evaluation and costing of services</td>
</tr>
<tr>
<td></td>
<td>Serverless</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Revised skill set

- Http protocol
- HTML, CSS, JavaScript
- JSON for data exchange
- How to use and manage APIs
- How to manipulate data objects
- Security settings: Port and IP restrictions, CORS, XSS and injection
New School

• Server: Amazon S3 Bucket
  • Configured to serve HTML
  • http://www.vote4movies.shafer.com.s3-website.us-east-2.amazonaws.com/

• Amazon API Gateway
  • Establish the API calls to the DB
  • Point and define the mapping to the DB

• Bring in abstraction library
  • https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js

• Web API endpoint to POST (INSERT) Data
  • https://wab2aq6r67.execute-api.us-east-2.amazonaws.com/shafer_dynamodb_test/movieid

• Web API endpoint to GET (SELECT) one record
  • https://wab2aq6r67.execute-api.us-east-2.amazonaws.com/shafer_dynamodb_test/movieid/9d43100e-6053-11e8-a4b6-a1fc9129a52b

• Web API endpoint to GET (SELECT) all records
  • https://wab2aq6r67.execute-api.us-east-2.amazonaws.com/shafer_dynamodb_test/movieid
Old School

- VMware Virtual Machine running Linux, Apache, and MySQL which serves: https://misdemo.temple.edu
- PHP script on the above
- SQL statements:
  - Insert into movies (moviename) values ('star wars');
  - Select * from movies where movieid = 1
  - Select * from movies