Documenting Serverless Architectures

Asher Sterkin, VP Engineering, BST
London, October 2018
Who I am

VP Engineering at Black Swan Technologies

Software Technologist/Architect

- (Strategic) Domain-Driven Design
- Serverless Architecture
- Cynefin
- Wardley Maps
- Promise Theory
Who we are

BlackSwan Technologies
Big Data, Cognitive Computing & Artificial Intelligence Software Company

- London, United Kingdom
- Tel-Aviv, Israel
- Delaware, United States

- 100% current client engagements
- 65 employees
Strategic Partnerships

4 Major Partnerships

Major brands that have committed resources and are now building their engineering teams around BlackSwan solution to later release it to the market.
- Nano probability event
- Huge, long lasting impact
- “Big surprise” for everybody
You cannot predict it, but could be better prepared.

- detect weak signals
- assess risk exposure
- discover “connected dots” in data

● Nano probability
● Huge, long lasting impact
● “Big surprise” for everybody
What is Serverless Architecture?

- Pure highly available on-demand computing
- Two flavors:
  - AWS Lambda - no pay for idle
  - AWS Fargate - log running containers
- Large and constantly growing number of fully managed services
- Tectonic shift in the software industry - has to be taken seriously
  - S. Wardley “Why the fuss about serverless?”
What's the Problem?

Without clear guidelines the choice will most likely be arbitral, informal and inconsistent.
What Does this Mean?

AWS Service?
SAM Template?
AWS Lambda Deployment?
Running Instance?

SDK Function Call?
Data Flow?
Address Reference?
Access Rights?

AWS Service?
SAM Template?
DynamoDB Schema?
DynamoDB Table Deployment?
DynamoDB Table Instance?
Why Bother?

Yan Cui, “Yubl’s Road to Serverless Architecture”

- Is it correct?
- Is it efficient?
- Is it reliable?
- Is it secure?
- What it does?

All what I could atest, is that its author is extremely smart.
Contrarily to the popular belief, the main responsibility of software architect is not to show the world how smart (s)he is, but to make the overall system structure abundantly clear to the whole organization. Of course, there are other responsibilities, but this one is utterly important and widely ignored.
System Modeling Basics
(always good to repeat)

- “Reality”, its model, and the model’s representation are not the same
- Each model reflects only some limited aspect of “reality”
- Semantic consistency is hard to achieve and maintain
- The more coherent the model, the narrower its context
- For any non-trivial system, more than one model is needed
- To get the whole picture, multiple models are mapped onto each other
4+1 View of Software Architecture

- Logical View
- Implementation View
- Use-Case View
- Process View
- Deployment View

Conceptual
- Analysts/Designers
  - Structure
- End-user
  - Functionality

Physical
- Programmers
  - Software management
- System integrators
  - Performance
    - Scalability
    - Throughput
- System engineering
  - System topology
  - Delivery, installation
    - Communication

P. Krutchten, “Architectural Blueprints - The "4+1" View Model of Software Architecture”
<table>
<thead>
<tr>
<th>View</th>
<th>Reflects</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case</td>
<td>HOW external actors interact with the system</td>
<td>User’s Manual</td>
</tr>
<tr>
<td>Logical</td>
<td>WHAT the system does</td>
<td>Business Rules</td>
</tr>
<tr>
<td>Process</td>
<td>HOW the system runs internally</td>
<td>Operations Manual</td>
</tr>
<tr>
<td>Implementation</td>
<td>HOW the system is built</td>
<td>Developer’s Guide</td>
</tr>
<tr>
<td>Deployment</td>
<td>HOW the system is deployed</td>
<td>Operations Manual</td>
</tr>
</tbody>
</table>
4+1 View of Serverless Architecture

Use-Case View

Conceptual

End-user Functionality

Process View

System integrators
- Performance
- Scalability
- Throughput

Analysts/Designers
- Structure

Logical View

Analytical View

End-user Functionality

Implementation View

Programmers
- Software management

Deployed View

System topology
- Delivery, installation communication

System engineering

Physical

Ideally, there should be a different icon set for each view, but it’s probably too much to ask for now
Views Interaction

Use Case View

Logical View (use case realizations)

Process View

Implementation View

Deployment View

Functionality
Business Agility

Performance
Scalability
Throughput
Resilience
Security

Development Agility
Quality
Security

System Topology
Cost
Security

feedback
Use Case View

- Here, we focus on architecturally essential use cases:
  - What justifies developing the system in eyes of external actors
  - Most important external integration points
- Normally traditional UML notation is enough
- All further analysis is conducted within the scope of particular use case(s)
Use Case View Examples

User
- Sign Up
- Upload Pictures
- Manage Albums

scaffolding

User
- Manage Watchlists
- Manage Scoring
- Perform Screening

main service
Logical View

- How the core business logic is realized
- Interaction with managed services where appropriate
- Custom automation also fits there
  - Need to be kept separate, unless this is your core domain
- UML Sequence diagrams are usually enough
- Sometimes UML Collaboration diagrams might be more convenient
- For modelling complex workflows UML Activity Diagrams are suitable option
Example: Classify New Image (sequence)

Good for analyzing multiple interactions between limited number of elements, reflecting operation sequencing, and reasoning about latency
Example: Classify New Image (collaboration)

Good for analyzing major interactions between large number number of elements, reflecting whole interaction graph, and reasoning about dependencies
Process View

- Suitable for reflecting:
  - What deployment and concurrency units (Lambda Functions, Fargate Services) exist
  - How much RAM/CPU is allocated to each computation service
  - How many instances will run in parallel
  - What system resources, e.g. DynamoDB tables, are involved
  - Cost estimations
  - Access permissions
  - How internal concurrency is arranged (advanced topic)

- Could be treated as a basic skeleton of SAM template without extra details
- By structure is similar to Collaboration Diagram, but semantics is different
Good for getting a bird view of the whole picture, accurate yet comprehensible
Within this context, an arrow means visibility and access rights and NOT data or command flow.

Good for getting a bird view of the whole picture, accurate yet comprehensible
Good for getting a bird view of the whole picture, accurate yet comprehensible
Good for getting a bird view of the whole picture, accurate yet comprehensible
In Process View, we deal with concurrency, visibility and access rights since all three aspects are closely related.
Scaling Process View Diagrams

Microservice wrapped in a Cloud Formation Stack
Implementation View

● Here, we reflect
  ○ How deployable units are built
  ○ Source code sharing, if any
  ○ Run-time environments and versions
  ○ 3rd party packages
  ○ CI/CD pipeline organization

● Could be considered as further clarifications on SAM template plus
  ○ AWS Build script
  ○ AWS Pipeline script
  ○ AWS CodeStar template
Example: Image Transformation Stack

Avoid putting excessive details where it's not necessary, e.g. do not depict embedded roles and policies as in this case.
Here, we reflect:

- How the system is deployed across multiple accounts/regions
- Cross-region replication
- CDN
- Network routing
- Network security
Example: Cross-Region Replication Deployment
Concluding Remarks

- All views in full detail would seldom be required
- Process View is often the most actively used
- Custom resources help with improving clarity
- Concentrate on the most important architectural decisions
  - Pickup relevant views to reflect these decision in high fidelity
- Might constitute a form of Lightweight Architecture Decision Records
What Could We Do About It?

- Developers and solution architects do not like to be told how to work
- Greedy platform and tool vendors could ruin any good initiative
- Too often great achievements of industry leaders are forgotten or ignored
- Grassroot community project with true organic growth might work better:
  - Open source catalog of modeling elements collected so far
  - Case studies of real-life projects
  - Collaborative research of implications on important operational aspects such as security, monitoring and disaster recovery
  - Meetup presentations and blog posts
  - Consulting and mentoring services where appropriate
More Information

- P. Krutchen, “Architectural Blueprints - The “4+1” View Model of Software Architecture”
- S. Wardley “Why the fuss about serverless?”
- E. Evans, “Domain-Driven Design Reference”
- A. Sterkin, “Serverless Architecture Language”
- A. Sterkin, “Consistent Modeling of Serverless Long-Running Background Tasks”
- A. Sterkin, “Serverless Steps”
Contact/Follow

https://medium.com/@asher.sterkin  https://www.slideshare.net/AsherSterkin  asher@blackswan-technologies.com
asher.sterkin@gmail.com