W18
Cloud Testing
Wednesday, May 2nd, 2018
3:00 PM

How to Test Serverless Cloud Applications

Presented by:

Glenn Buckholz

Coveros, Inc.

Brought to you by:

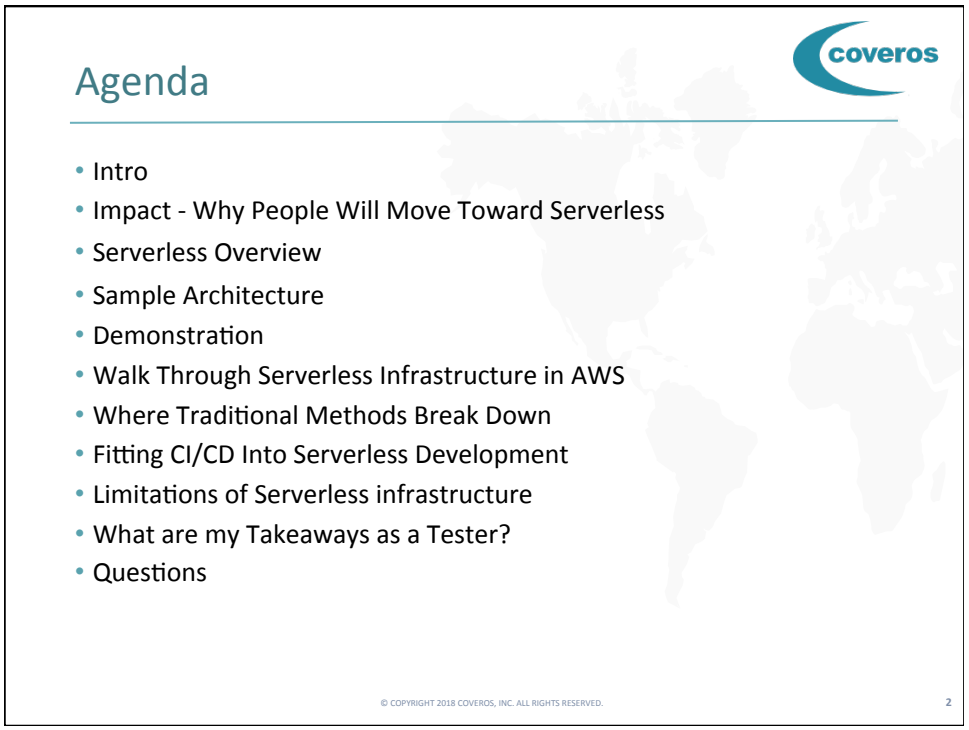
350 Corporate Way, Suite 400, Orange Park, FL 32073
888-268-8770 ∙ 904-278-0524 - info@techwell.com - http://www.stareast.techwell.com/

Glenn Buckholz

Coveros, Inc.

Glenn Buckholz leads CI and deployment automation efforts at Coveros. Using his more than fifteen years of industry experience, Glenn brings success to his customers. Beginning his career as a consultant implementing automated test frameworks, he introduced the concept of change management to many, many projects. After moving on from consulting, Glenn settled down at the Public Company Accounting Oversight Board as their full-time enterprise change manager. He eventually ventured back into the real world at Coveros, where he specializes in implementing agile practices, implementing CI, and engineering configuration management instead of simply documenting it.


coveros
Agility. Security. Delivered.
coveros
How to Test Serverless Cloud Applications
Glenn Buckholz
E-mail: glenn.buckholz@coveros.com
Coveros, Inc.
© COPYRIGHT 2016 COVEROS, INC. ALL RIGHTS RESERVED.
1


coveros
Agenda
• Intro
• Impact - Why People Will Move Toward Serverless
• Serverless Overview
• Sample Architecture
• Demonstration
• Walk Through Serverless Infrastructure in AWS
• Where Traditional Methods Break Down
• Fitting CI/CD Into Serverless Development
• Limitations of Serverless infrastructure
• What are my Takeaways as a Tester?
• Questions
© COPYRIGHT 2018 COVEROS, INC. ALL RIGHTS RESERVED.
2

Introduction

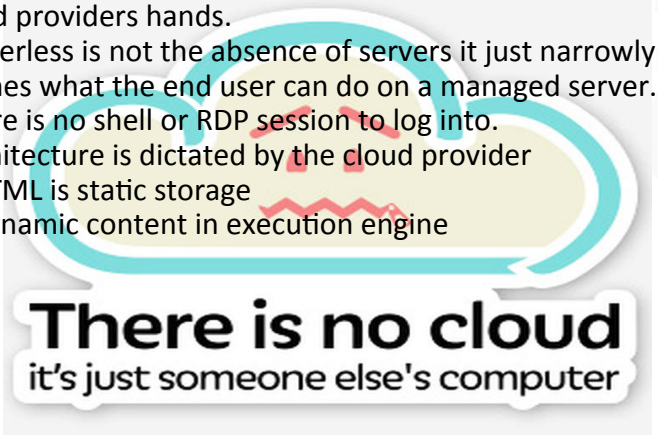
- IaaS puts hardware management into the cloud providers hands. Serverless puts Platform management into the cloud providers hands.
- Serverless is not the absence of servers it just narrowly defines what the end user can do on a managed server.
- With the change in architecture certain testing activities must change to accommodate.
- Other testing practices do not need to change significantly, but may need slight modification.
- CI/CD change significantly under the covers, but on the surface looks the same and follows the same pattern.

Impact - Why People will Move to Serverless (IMHO)

- Cost $$$$$$$$$$$$$$$$
  - You pay a very minimal fee per function invocation.
- More customizable than most SaaS offerings
- Business can implement IT with less staff.
  - Developers, Cloud specialists, DevOps Engineers
- Potential scalability limited only by provider and budget
  - Space/Bandwidth/Compute
- Security
  - Can focus on code and architecture and leave the rest to the cloud provider.
- Log aggregation and analytics.
  - Cloud Watch etc..

Serverless Overview

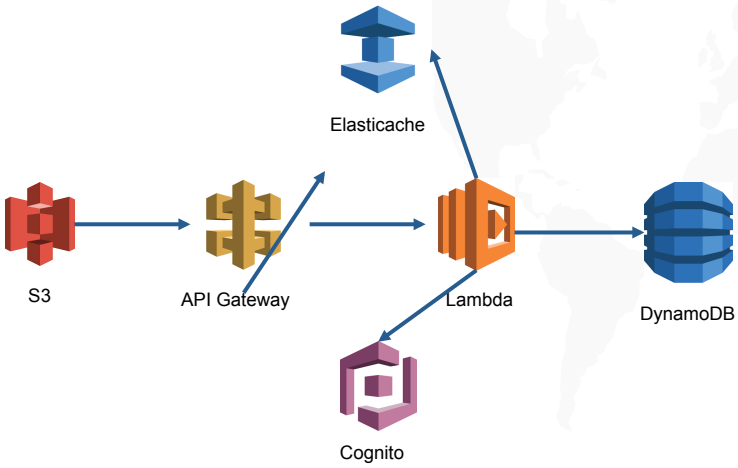
- IaaS puts hardware management into the cloud providers hands. Serverless puts Platform management into the cloud providers hands.
- Serverless is not the absence of servers it just narrowly defines what the end user can do on a managed server.
- There is no shell or RDP session to log into.
- Architecture is dictated by the cloud provider
  - HTML is static storage
  - Dynamic content in execution engine



Serverless Overview

- Static Content
  - S3
- Web Connector
  - API Gateway
- Execution Engine (Dynamic Content)
  - Lambda
- Identity Management
  - Cognito
- Database Backend (Persistent Content)
  - DynamoDB
- Session Management
  - ElastiCache (Redis, memcached)

Sample Architecture



Demonstration

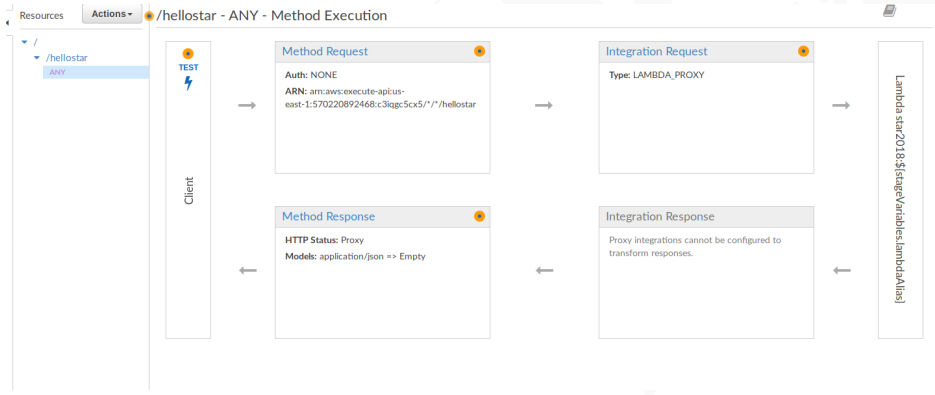
AWS Serverless Walkthrough S3

- Object storage built to store and retrieve any amount of data from anywhere.
  - FINRA stores Peta/Exo bytes or more
- 99.999999999% durability.
- ACL's
- Allows URL to access objects stored there.
- Encryption at rest in the datacenter.
  - IF YOU ARE ACCESSING THE DATA IT IS NOT ENCRYPTED IT JUST CAN'T BE PHYSICALLY STOLEN.
- SDK for client side encryption.
  - The app decrypts the data.
- Used for and by CDNs.

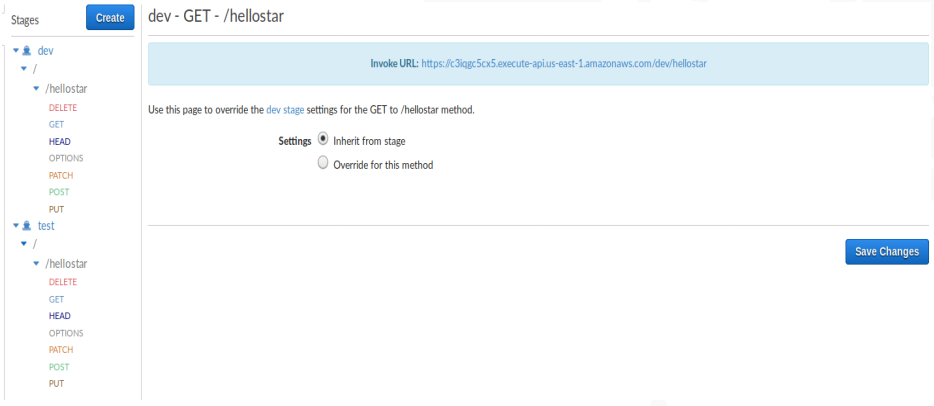
AWS Serverless Walkthrough API Gateway

- A web method with out all the trouble of NGINX or Apache.
- Will service http calls and scale up to what AWS can handle.
  - It is wise to set throttling limits or you can be DDoSed and have to pay for it.
  - Throttling controls cost as well.
- Front it with a friendly DNS name
- Facility for custom response to valid http conditions
  - 400s, 500s, etc.
- Ability to transform requests on the fly.
- Multiple versions and environments of the same interface can run simultaneously.
- $3.50 per 1 million API calls.

AWS Serverless Walkthrough API Gateway



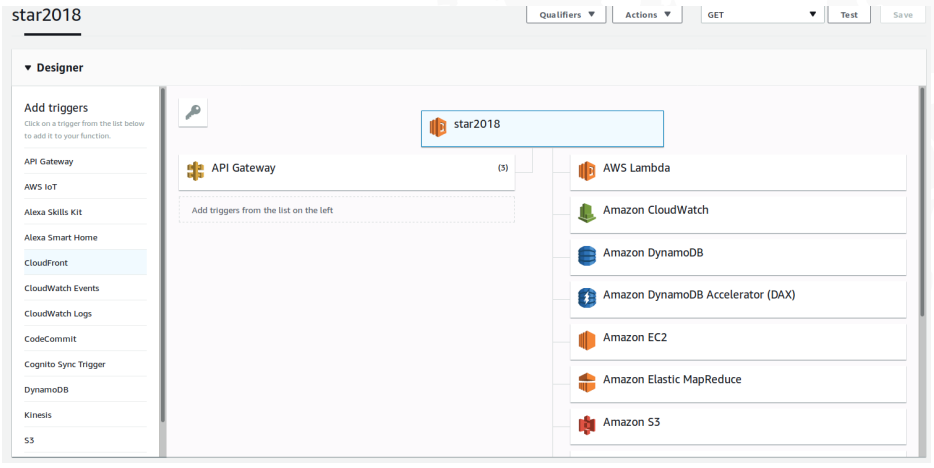
AWS Serverless Walkthrough API Gateway



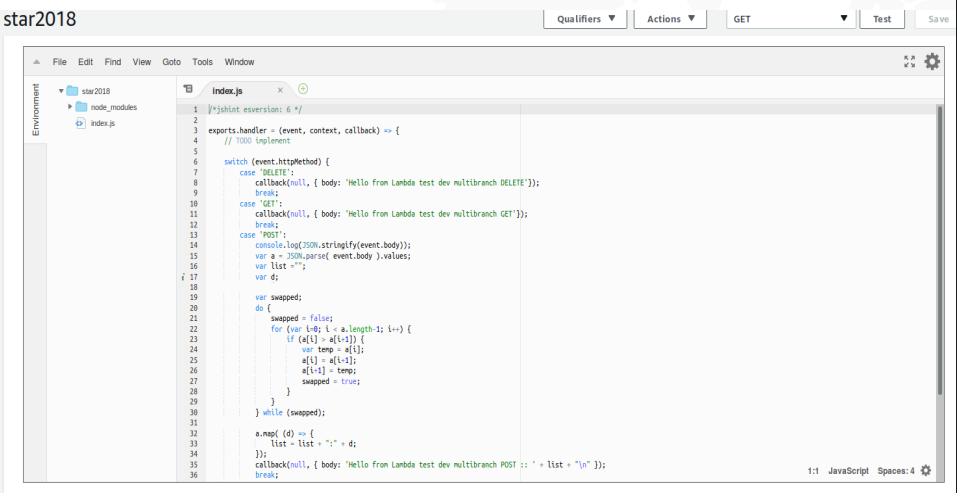
AWS Serverless Walkthrough Lambda

- Code that AWS just executes.
- Almost useless on its own, must be triggered or called by something.
  - API call, Cloud Trails Event, Timed Event, S3 change, code commit.
- You only pay for the time the code is executing.
  - No server spin up/tear down.
- Versions can be tagged or aliased so dependant services can be logically grouped.
- CPU, Memory limits, calls can be throttled.
- AWS IAM controls what resources function has access to.
- UI to link services together in AWS console.
- Integrated Test Harness

AWS Serverless Walkthrough Lambda



AWS Serverless Walkthrough Lambda



Where Traditional Testing Methods Break Down

- Log Retrieval and Log Monitoring
  - You cannot just ssh in somewhere and tail the log.
  - Cloud admin must use the AWS console or CLI to retrieve events.
  - Coordinating test times for log retrieval is imperative.
- Having a private testing environment
  - All URLs are exposed, you must use some sort of authentication to keep the public out of the test sites. (cognito)
- RBAC testing requires a test case for accessing all of amazon.

Where Traditional Testing Methods Break Down

- Recognizing where there is a real functionality error and when you have just hit a memory or other resource cap.
- Collecting coverage results from integration testing.
  - Hard to interface with things inside AWS.
- Existing bug reporting methods need to change.
  - Code version + API version + AWS service configurations + other AWS components and configs + regions
    - Must find a way to gather this information automatically.
- Most environments must be shared.
- Security testing must be coordinated with AWS and some methods will just not be allowed.

Fitting CI/CD into Serverless Development

- The cloud provider must have an API to access and create parts of the application.
- CI/CD engine must have compatible way to interface with the Cloud API.
- There must be a reasonable way to simulate running on the cloud infrastructure locally.
  - Deployments take time you want to fail early and fast.
- There must be a way to package cloud configuration AND code.

Limitations of the Serverless Infrastructure

- DDOS can get expensive
  - Your costs will scale with the size of the attack
  - Increased Monitoring
- Only a few languages and versions of that language
  - Java, NodeJS, Go, Python, C#
- Vendor lock in.
  - There is no moving away from the cloud provider without significant rework.
- Everything is public, even your test environment.
- SDK's for local testing are still immature.

What are My Takeaways as a Tester?

- Functional Testers
  - How will we manage test data may be different.
    - Auto load of seed data
    - Automated pull back and sanitation from production. The process becomes physically easier.
    - Credentials management. Every role must have a real login. There are no "test accounts".
- Automated Testers
  - More of a focus on API testing
  - UI testing stays relatively the same.
  - Static analysis should stay the same.

What are My Takeaways as a Tester?

- Testing in General
  - AWS configuration is as important as code.
  - Find a way to test the configuration and make test cases that explicitly look for AWS configuration issues.
  - Ensure code artifacts have a way to clearly map to AWS configurations.
  - Ensure AWS configurations can be managed in an auditable way.

What are My Takeaways as a Tester?

- Caused the event class to be null in my function but not in unit or integration tests using SAM. The error was in the API gateway config.

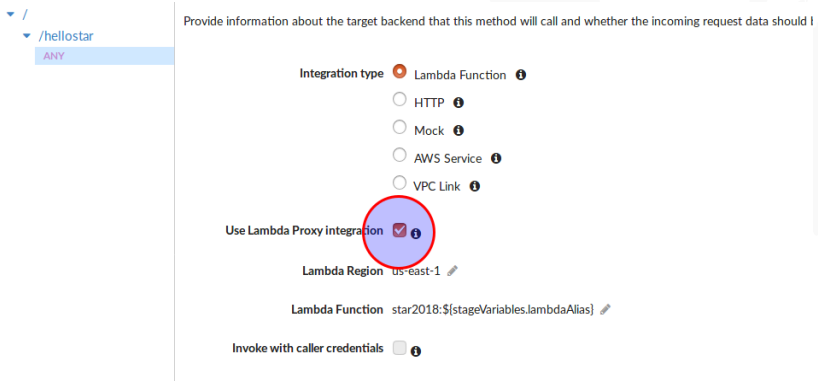
```
exports.handler = (event, context, callback) => {
    // TODO implement
    switch (event.httpMethod) {
        case 'DELETE':
            callback(null, { body: 'Hello from Lambda test dev multibranch DELETE'});
            break;
        case 'GET':
            callback(null, { body: 'Hello from Lambda test dev multibranch GET'});
            break;
        case 'POST':
            console.log(JSON.stringify(event.body));
            var a = JSON.parse( event.body ).values;
            var list ="";
            var d;

            var swapped;
            do {
                swapped = false;
                for (var i=0; i < a.length-1; i++) {
                    if (a[i] > a[i+1]) {
                        var temp = a[i];
                        a[i] = a[i+1];
                        a[i+1] = temp;
                        swapped = true;
                    }
                }
            } while (swapped);

            a.map( (d) => {
                list = list + ":" + d;
            });
            callback(null, { body: 'Hello from Lambda test dev multibranch POST :: ' + list + "\n" });
            break;
        case 'PUT':
            callback(null, { body: 'Hello from Lambda test dev multibranch PUT'});
            break;
        default:
            var dto = { eventClass: event };
            callback(null, { body: "Unsupported Method: " + event.httpMethod + "::" + JSON.stringify(dto)});
    }
};
```

Switch to decide action was null

Falling through to default

What are My Takeaways as a Tester?

- Differences between SDK and actual implementation.
  - SDK may have limited configurability
- While creating the Demo the following checkbox



Conclusion

- A good DevOps team can easily embrace the serverless world.
- Testing from a 10,000 foot level looks mostly the same.
- The AWS configuration is code that should be tested as part of application testing.
- Identity management is very important there are no more innocuous test users since everything is public. Also, your app can now touch your cloud infrastructure. RBAC testing positive and negative is essential.
- Security testing changes significantly since the cloud infrastructure is now apart of you application.
- Test locally everywhere you can even of the SDKs are immature.
- Understand what information needs to be in a defect report and how to automate as much of that collection as possible.

Questions?