Best Practices for Building Alexa Smart Home Camera Skills

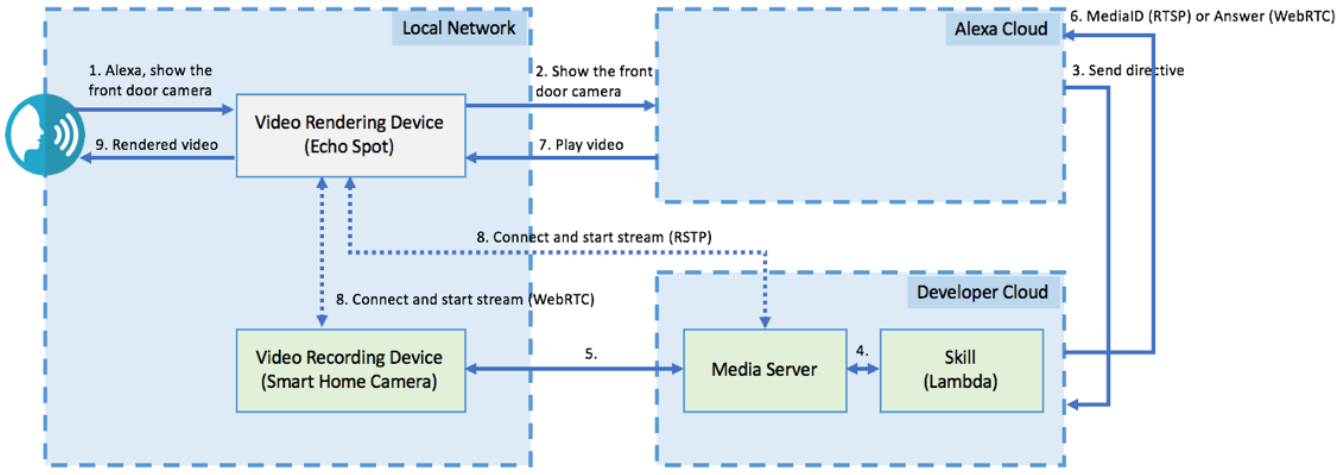
Anna Khabibullina May 07, 2019 Share:

Cameras Tips & Tools Smart Home

Low latency is a key contributor to customer satisfaction with smart home camera skills. Customers who connect their smart home cameras and doorbells to Alexa love streaming live camera feeds to Echo Show and Echo Spot, but can be frustrated if the stream takes a long time to appear. If it takes too long for the stream to appear, customers may stop using the feature, provide a negative review or low star ratings on the skill, or both. This blog describes 10 tips to help you optimize the AWS Lambda function and related services for your smart home camera skill.

Background

The diagram below demonstrates the current interaction model between Alexa and developer services.



Optimization Best Practices

To improve performance of the Lambda function for your Alexa smart home skill, you can take the

more CPU power to your function and will execute it faster. This can result in positive customer experience with faster response times and lower running cost by reducing the execution time for Lambda function.

2. Adjust Skill Lambda timeout to 6 seconds: increase the Lambda timeout from the default of 3 to 6 seconds. You must send a response to Alexa within 6 sec but to ensure better CX, we recommend to optimize the time to get it out as soon as you can:
    - For RTSP path, all responses must occur 6 seconds or less after request received.
    - For WebRTC path, for any offer sent to your skill, you must respond with the answer within 6 seconds.
3. Decouple init, config and core logic: any externalized configuration or dependencies that your code retrieves should be stored and referenced locally after initial execution. This will limit re-initialization of variables/objects on every invocation.
4. Cold start / runtime optimizations: if your Lambda function is not invoked for some time, it is removed from the underlying container. When you try to access it for the first time after this idle period, it loads it again and that takes some extra time in loading all the required code and initializing it. One of the benefits with using Node.js and Python over Java to write Lambda functions is that they produce much smaller packages—which we know translates to lower code start time.
5. Minimize the complexity of your dependencies: prefer simpler frameworks that load quickly on Execution Context startup. For example, if you use Java, prefer simple dependency injection (IoC) frameworks like Dagger or Guice, over more complex ones like Spring Framework and consider removing unused/dead code. By deleting Lambda functions you no longer need, the unused functions won't needlessly count against deployment package size limit. This will reduce the amount of time that it takes for your deployment package to be downloaded and unpacked ahead of invocation.
6. Core logic optimizations: once the directive is received, parallelize as many event-processing tasks because many of them are i/o bound and by parallelizing them you can get performance better. Check if any logic (other than the critical path) is executing in the Lambda. For RTSP path, we recommend to return the URI (RTSP) or the answer (WebRTC) at the earliest possible opportunity

before continuing with any further processing. Specifically, operations like returning of the URI and waking a camera to begin streaming should be done asynchronously as background. This technique will improve a user's perceived performance. You can send out interstitial video frames until the camera stream is actually available such as LoopUntilLive feature from Wowza.

7. Regionalization: do not do any cross-region calls. We recommend placement of Lambda functions and media services in additional regions based on geographic distances contributing to latency and regions supported by Alexa Service. At the very minimum, if you serve in NA & EU, your AWS should support 3 regions: N. Virginia (us-east-1), Dublin (eu-west-1), Oregon (us-west-2). The list of all supported regions for Lambda can be found on AWS website.

Lambda Latency Optimization Recommendations

Additionally, having the following metrics, dashboards and alarms will help you proactively make sure customers have a great experience, with the added benefit of meeting one of the eligibility standards of the Works with Alexa (WWA) program:

1. Setup CloudWatch monitoring and alerts for your Lambda & media server (if non-existent today) to automatically notify your dev team about anomalous activity when critical incidents happened (e.g. customer-impacting issue such as outage or severe regression which may require interaction with Alexa team and/or fixes on developer side). At the very minimum, we suggest to setup alarms for the following rules / thresholds: min & max thresholds:
    - Lambda Volume and TPS (P50, P90) per region and camera device model
    - Lambda response latency (P50, P90) per region and camera device model
    - Lambda success rate (P50, P90) i.e. successful response rate, per region and camera device model
    - Lambda failure rate (P50, P90) i.e. invocation error, by error type, per region and camera device model
    - Stream responsiveness duration (P50, P90) per region and camera device model.
        - For RTSP path, this can be measured as URI responsiveness time is the duration between the time when the skill Lambda sent media URI to Alexa Service and the first time rendering device attempts to play it.
        - For WebRTC path, this can be measured as URI responsiveness time is the duration between the time when the partner Lambda sent AnswerGeneratedForSessionEvent to Alexa Service and the first time the camera/doorbell device started the stream.
2. Prepare the quality / operational reports on regular cadence (recommended: weekly or bi-weekly) you will review internally as part of your weekly meetings. You can use AWS X-Ray to show for debugging / troubleshooting and generating reports with latency improvements for your AWS services. You can also enable X-Ray in Prod system and sample X% of traffic to get visibility into common issues. During each skill launch, check your operational dashboards to verify no unusual activity (such as spikes or drops for the traffic volume or success / failure rates) occurred. If you found the missing metrics / dashboards / alarms, account time to add it. Avoid deploying changes to Lambda before a weekend or other periods of time where engineers will not be around to address issues.
3. Run load / performance tests for your Lambda functions to determine the optimum memory size configuration. We also recommend auditing your hardware and bandwidth connectivity for reliability

Related Links

1. AWS Lambda Developer Guide
2. AWS Lambda Scaling Guidelines
3. Steps to Build a Smart Home Skill
4. Amazon Lex and AWS Lambda Blueprints
5. Best Practices for Scaling Your Alexa Skill Using Amazon Web Services

Back to Top

Launch

Launch

Resources

Getting Started

Tutorials

Documentation

Developer Forum

Agencies and Tools

AVS Resources

Getting Started

AVS Device SDK

AVS API

Dev Kits for AVS

Agreements

Agreements and Terms

Program Materials License Agreement

Amazon Developers Services Portal Terms of Use

Support

Amazon Developer Support

Contact Us

Forums

Manage Email Preferences

Follow Us: