AWS Well-Architected Tool

AWS Well-Architected Tool nlaws - Serverless Lens Report

AWS Account ID: 785169158894

AWS Well-Architected Tool Report

Table of contents

Workload properties 4

Lens overview 6

Improvement plan 7

- High risk 7
- Medium risk 8

Lens details 10

- Operational Excellence 10
- Security 13
- Reliability 17
- Performance Efficiency 20
- Cost Optimization 22

Workload properties

Workload name
nlaws

ARN
arn:aws:wellarchitected:us-east-1:785169158894:workload/9a9dc4d25657166bb43cc25680fab32d

Description
project

Review owner
Nathanael Laws

Industry type
-

Industry
-

Environment
Production

AWS Regions
US East (N. Virginia)

Non-AWS regions
-

Account IDs
-

Architectural design
-

Application

-

Lens overview

Questions answered

9/9

Version

Serverless Lens, 4th Feb 2020

Pillar	Questions answered
Operational Excellence	2/2
Security	3/3
Reliability	2/2
Performance Efficiency	1/1
Cost Optimization	1/1

Lens notes

-

Improvement plan

Improvement item summary

High risk: 5

Medium risk: 3

Pillar	High risk	Medium risk
Operational Excellence	0	2
Security	2	1
Reliability	2	0
Performance Efficiency	0	0
Cost Optimization	1	0

High risk

Operational Excellence
No improvements identified

Security
• SEC 2.How do you manage your Serverless application's security boundaries? • SEC 3.How do you implement application Security in your workload?

Reliability
• REL 1.How do you regulate inbound request rates? • REL 2.How do you build resiliency into your Serverless application?

Performance Efficiency
No improvements identified

Cost Optimization
• COST 1.How do you optimize your Serverless application's costs?

Medium risk

Operational Excellence
• OPS 1.How do you evaluate your Serverless application's health? • OPS 2.How do you approach application lifecycle management?

Security
• SEC 1.How do you control access to your Serverless API?

Reliability
No improvements identified

Performance Efficiency
No improvements identified

Cost Optimization
No improvements identified

Lens details

Operational Excellence

Questions answered

2/2

Question status

- High risk: 0
- Medium risk: 2
- No improvements identified: 0
- Not Applicable: 0
- Unanswered: 0

Pillar notes

-

1. How do you evaluate your Serverless application's health?

⚠ Medium risk

Selected choice(s)

- Understand, analyze, and alert on metrics provided out of the box

Not selected choice(s)

- Use application, business, and operations metrics
- Use distributed tracing and code is instrumented with additional context
- Use structured and centralized logging
- None of these

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Use application, business, and operations metrics
- Use distributed tracing and code is instrumented with additional context
- Use structured and centralized logging

Ask an expert

2. How do you approach application lifecycle management?

⚠ Medium risk

Selected choice(s)

- Use infrastructure as code and stages isolated in separate environments
- Review the function runtime deprecation policy

Not selected choice(s)

- Use CI/CD including automated testing across separate accounts
- Prototype new features using temporary environments
- Use a rollout deployment mechanism
- Use configuration management
- None of these

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Use CI/CD including automated testing across separate accounts
- Prototype new features using temporary environments
- Use a rollout deployment mechanism
- Use configuration management

Ask an expert

Security

Questions answered

3/3

Question status

- High risk: 2
- Medium risk: 1
- No improvements identified: 0
- Not Applicable: 0
- Unanswered: 0

Pillar notes

-

1. How do you control access to your Serverless API?

⚠ Medium risk

Selected choice(s)

- Use appropriate endpoint type and mechanisms to secure access to your API

Not selected choice(s)

- Scope access based on identity's metadata
- Use authentication and authorization mechanisms
- None of these

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Scope access based on identity's metadata
- Use authentication and authorization mechanisms

Ask an expert

2. How do you manage your Serverless application's security boundaries?

High risk

Selected choice(s)

- None of these

Not selected choice(s)

- Evaluate and define resource policies
- Use temporary credentials between resources and components
- Design smaller, single purpose functions
- Control network traffic at all layers

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Evaluate and define resource policies
- Use temporary credentials between resources and components
- Design smaller, single purpose functions
- Control network traffic at all layers

Ask an expert

3. How do you implement application Security in your workload?

High risk

Selected choice(s)

- None of these

Not selected choice(s)

- Review security awareness documents frequently
- Store secrets that are used in your code securely
- Automatically review workload's code dependencies/libraries
- Validate inbound events
- Implement runtime protection to help prevent against malicious code execution

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Review security awareness documents frequently
- Store secrets that are used in your code securely
- Automatically review workload's code dependencies/libraries
- Validate inbound events
- Implement runtime protection to help prevent against malicious code execution

Ask an expert

Reliability

Questions answered

2/2

Question status

- High risk: 2
- Medium risk: 0
- No improvements identified: 0
- Not Applicable: 0
- Unanswered: 0

Pillar notes

-

1. How do you regulate inbound request rates?

High risk

Selected choice(s)

- None of these

Not selected choice(s)

- Use throttling to control inbound request rates
- Use mechanisms to protect non-scalable resources
- Use, analyze, and enforce API quotas

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Use throttling to control inbound request rates
- Use mechanisms to protect non-scalable resources
- Use, analyze, and enforce API quotas

Ask an expert

2. How do you build resiliency into your Serverless application?

High risk

Selected choice(s)

- None of these

Not selected choice(s)

- Manage transaction, partial, and intermittent failures
- Manage duplicate and unwanted events
- Consider scaling patterns at burst rates
- Orchestrate long-running transactions

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Manage transaction, partial, and intermittent failures
- Manage duplicate and unwanted events
- Consider scaling patterns at burst rates
- Orchestrate long-running transactions

Ask an expert

Performance Efficiency

Questions answered

1/1

Question status

- High risk: 0
- Medium risk: 0
- No improvements identified: 1
- Not Applicable: 0
- Unanswered: 0

Pillar notes

-

1. How do you optimize your Serverless application's performance?

No improvements identified

Selected choice(s)

- Measure, evaluate, and select optimum capacity units
- Integrate with managed services directly over functions when possible
- Measure and optimize function startup time

Not selected choice(s)

- Take advantage of concurrency via async and stream-based function invocations
- Optimize access patterns and apply caching where applicable
- None of these

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

No risk detected for this question. No action needed.

Cost Optimization

Questions answered

1/1

Question status

- High risk: 1
- Medium risk: 0
- No improvements identified: 0
- Not Applicable: 0
- Unanswered: 0

Pillar notes

-

1. How do you optimize your Serverless application's costs?

High risk

Selected choice(s)

- Optimize function configuration to reduce cost

Not selected choice(s)

- Minimize external calls and function code initialization
- Optimize logging output and its retention
- Use cost-aware usage patterns in code
- None of these

Best Practices marked as Not Applicable

-

Notes

-

Improvement plan

- Minimize external calls and function code initialization
- Optimize logging output and its retention
- Use cost-aware usage patterns in code

Ask an expert