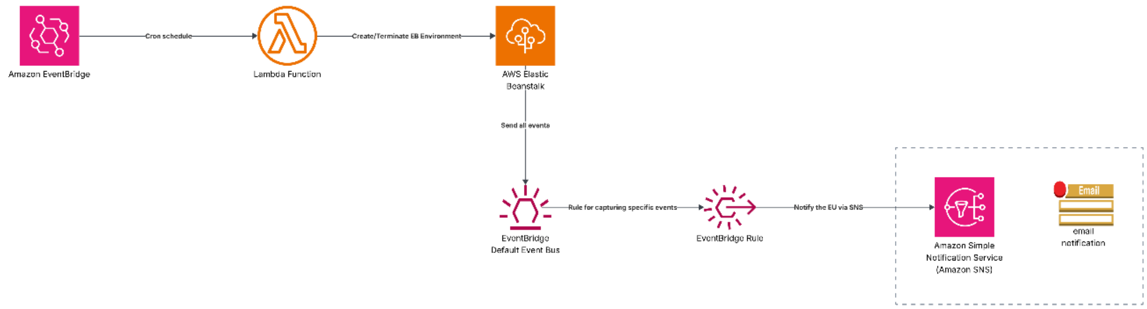
Elastic Beanstalk Automation & Notification Architecture



This document describes the architecture for automating AWS Elastic Beanstalk environment creation and termination using Amazon EventBridge, AWS Lambda, and Amazon SNS for email notifications. The main objective behind this architecture and not using Step Functions is that AWS Step Functions lets you coordinate multiple AWS services into serverless workflows so you can build and update apps quickly. Using Step Functions, you can design and run workflows that stitch together services such as AWS Lambda into feature-rich applications.

This may involve cost components that are not needed. There is no need to keep the Step Function running in a "wait state" for the most part of the day just to control the provisioning and termination of Elastic Beanstalk environment. This is better handled via a "serverless cron" type of solution that can be invoked twice a day for provisioning and termination of Elastic Beanstalk environment.

1. Architecture Overview

The architecture consists of the following flow:

- Amazon EventBridge Scheduler (cron) triggers a Lambda function.
- Lambda creates or terminates an Elastic Beanstalk (EB) environment.
- Elastic Beanstalk emits lifecycle events to the Default Event Bus.
- An EventBridge Rule filters specific EB events.
- Matching events are sent to Amazon SNS.
- SNS sends email notifications to subscribed users.

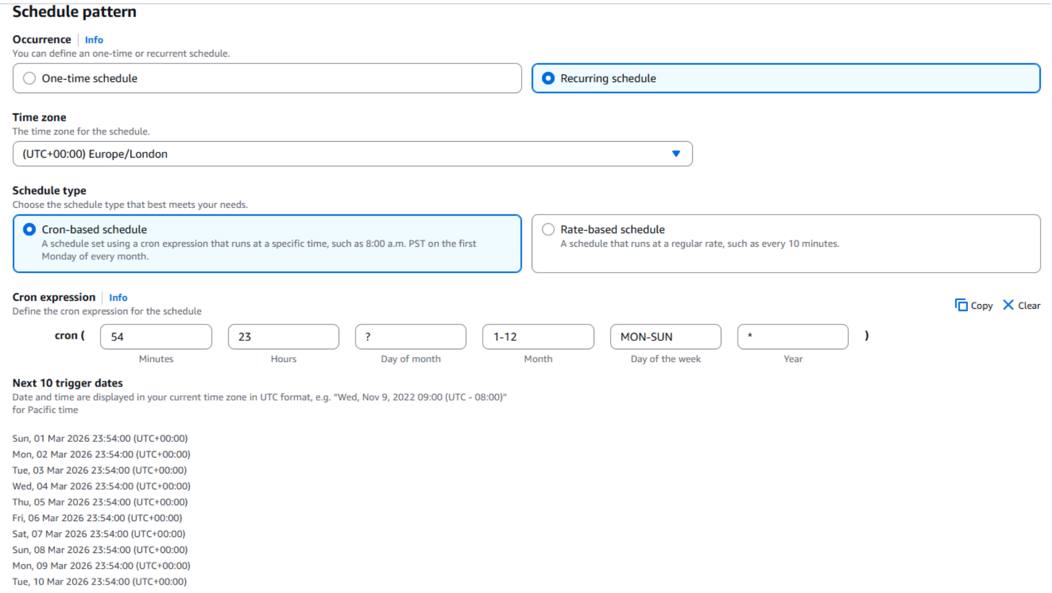
2. Components

2.1 Amazon EventBridge (Scheduler)

Amazon EventBridge Scheduler is a serverless scheduler that allows you to create, run, and manage tasks from one central, managed service. Highly scalable, EventBridge Scheduler allows you to schedule millions of tasks that can invoke any AWS service as a target.

EventBridge Scheduler runs a cron expression that triggers the Lambda function at predefined intervals. Example cron expressions:

- Create environment daily at 8 AM UTC
- Terminate environment daily at 8 PM UTC



Schedule pattern

Occurrence | Info
You can define an one-time or recurrent schedule.

○ One-time schedule ● Recurring schedule

Time zone
The time zone for the schedule.

(UTC+00:00) Europe/London

Schedule type
Choose the schedule type that best meets your needs.

● Cron-based schedule
A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

○ Rate-based schedule
A schedule that runs at a regular rate, such as every 10 minutes.

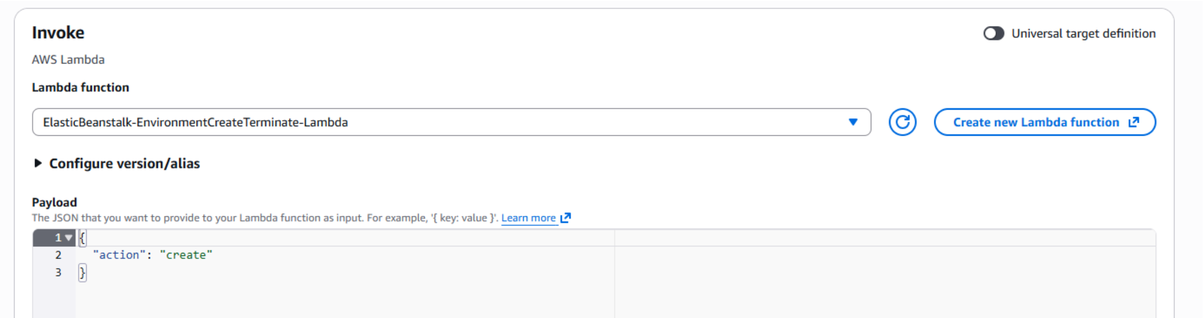
Cron expression | Info
Define the cron expression for the schedule

	Minutes	Hours	Day of month	Month	Day of the week	Year	
cron (	54	23	?	1-12	MON-SUN	*	)

Next 10 trigger dates
Date and time are displayed in your current time zone in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC - 08:00)" for Pacific time

Sun, 01 Mar 2026 23:54:00 (UTC+00:00)
Mon, 02 Mar 2026 23:54:00 (UTC+00:00)
Tue, 03 Mar 2026 23:54:00 (UTC+00:00)
Wed, 04 Mar 2026 23:54:00 (UTC+00:00)
Thu, 05 Mar 2026 23:54:00 (UTC+00:00)
Fri, 06 Mar 2026 23:54:00 (UTC+00:00)
Sat, 07 Mar 2026 23:54:00 (UTC+00:00)
Sun, 08 Mar 2026 23:54:00 (UTC+00:00)
Mon, 09 Mar 2026 23:54:00 (UTC+00:00)
Tue, 10 Mar 2026 23:54:00 (UTC+00:00)

Please note, when you select lambda as a target to create/terminate your Elastic Beanstalk environment it is important to pass a payload with the cron schedule.



Invoke

AWS Lambda

Lambda function

ElasticBeanstalk-EnvironmentCreateTerminate-Lambda

▸ **Configure version/alias**

Payload
The JSON that you want to provide to your Lambda function as input. For example, '{ key: value }'. Learn more

```
{
  "action": "create"
}
```

You can set further settings like Retry policy which allows EventBridge Scheduler to re-run a schedule if it fails to invoke its target. You can specify the maximum age of the event and the maximum number of times to retry. You can set SQS as in your account or a separate account.

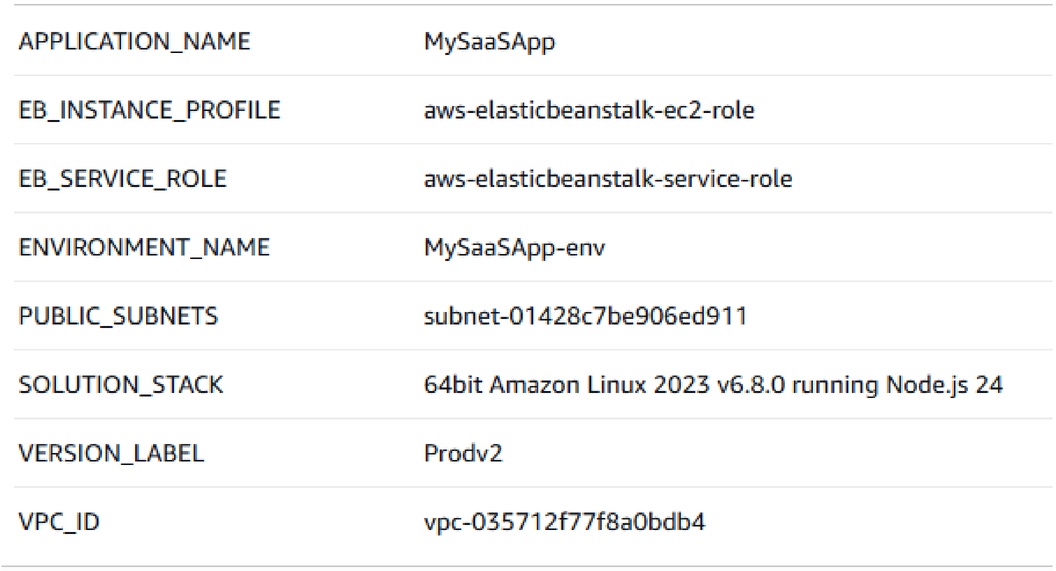
Please make sure that proper IAM permissions are configured as EventBridge Scheduler requires permission to send events to the target.

You can find a detailed documentation on cron jobs here https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-create-rule-schedule.html

2.2 AWS Lambda Function

The Lambda function performs the following actions:

- Based on the invocation payload from EventBridge Scheduler, lambda will create/terminate the Beanstalk environment.
- Make sure you pass proper Env variables to Beanstalk, please find an example screenshot below:

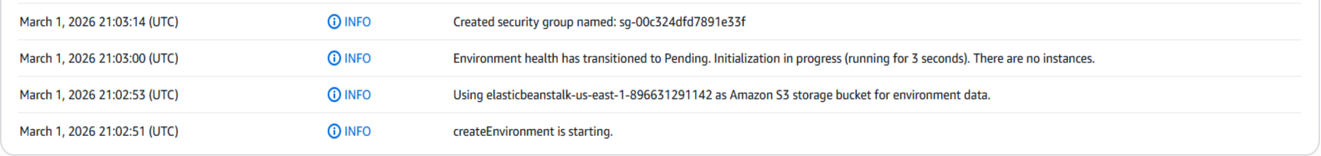


APPLICATION_NAME	MySaaSApp
EB_INSTANCE_PROFILE	aws-elasticbeanstalk-ec2-role
EB_SERVICE_ROLE	aws-elasticbeanstalk-service-role
ENVIRONMENT_NAME	MySaaSApp-env
PUBLIC_SUBNETS	subnet-01428c7be906ed911
SOLUTION_STACK	64bit Amazon Linux 2023 v6.8.0 running Node.js 24
VERSION_LABEL	Prodv2
VPC_ID	vpc-035712f77f8a0bdb4

- It's important to pass EB_SERVICE_ROLE, EB_INSTANCE_PROFILE, PUBLIC_SUBNETS, SOLUTION_STACK, VERSION_LABEL, VPC_ID. Otherwise you will keep getting launch issues in Beanstalk. Please note, this is just a recommendation as it also depends on how you build your own lambda function but make sure lambda execution role has proper permission to pass role to Beanstalk.
- Building your lambda function completely depends on you how you want to configure it depending on the requirements.

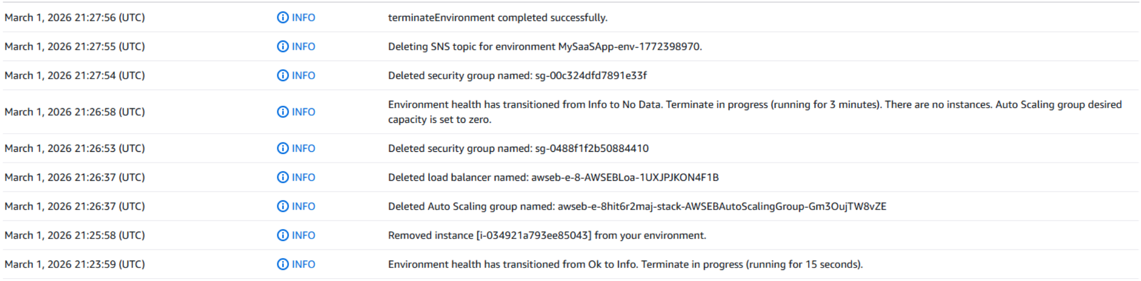
2.3 AWS Elastic Beanstalk

Elastic Beanstalk provisions application environments.



March 1, 2026 21:03:14 (UTC)	ⓘ INFO	Created security group named: sg-00c324dfd7891e33f
March 1, 2026 21:03:00 (UTC)	ⓘ INFO	Environment health has transitioned to Pending. Initialization in progress (running for 3 seconds). There are no instances.
March 1, 2026 21:02:53 (UTC)	ⓘ INFO	Using elasticbeanstalk-us-east-1-896631291142 as Amazon S3 storage bucket for environment data.
March 1, 2026 21:02:51 (UTC)	ⓘ INFO	createEnvironment is starting.

Elastic Beanstalk terminating the application environments.



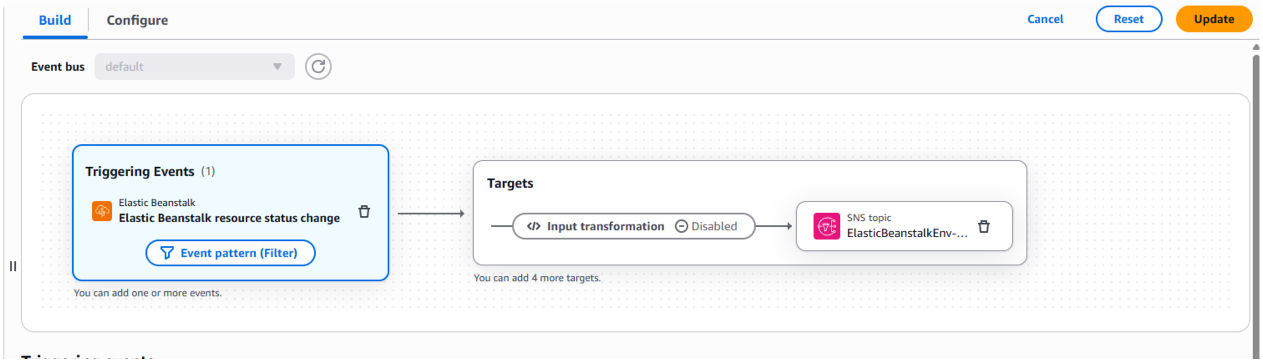
March 1, 2026 21:27:56 (UTC)	ⓘ INFO	terminateEnvironment completed successfully.
March 1, 2026 21:27:55 (UTC)	ⓘ INFO	Deleting SNS topic for environment MySaaSApp-env-1772398970.
March 1, 2026 21:27:54 (UTC)	ⓘ INFO	Deleted security group named: sg-00c324dfd7891e33f
March 1, 2026 21:26:58 (UTC)	ⓘ INFO	Environment health has transitioned from Info to No Data. Terminate in progress (running for 3 minutes). There are no instances. Auto Scaling group desired capacity is set to zero.
March 1, 2026 21:26:53 (UTC)	ⓘ INFO	Deleted security group named: sg-0488f1f2b50884410
March 1, 2026 21:26:37 (UTC)	ⓘ INFO	Deleted load balancer named: awseb-e-8-AWSEBLoa-1UXJPJKON4F1B
March 1, 2026 21:26:37 (UTC)	ⓘ INFO	Deleted Auto Scaling group named: awseb-e-8hit6r2maj-stack-AWSEBAutoScalingGroup-Gm3OujTW8vZE
March 1, 2026 21:25:58 (UTC)	ⓘ INFO	Removed instance [i-034921a793ee85043] from your environment.
March 1, 2026 21:23:59 (UTC)	ⓘ INFO	Environment health has transitioned from Ok to Info. Terminate in progress (running for 15 seconds).

2.4 EventBridge Rule (Default Event Bus)

When environment status changes occur, EB automatically emits events to the Default Event Bus with:

- source: aws.elasticbeanstalk
- detail-type: Elastic Beanstalk resource status change

An EventBridge rule captures specific EB events.



Event bus: default

Triggering Events (1)

Elastic Beanstalk
Elastic Beanstalk resource status change

You can add one or more events.

Targets

Input transformation: Disabled

SNS topic
ElasticBeanstalkEnv-...

You can add 4 more targets.

Filter Pattern:

{

"source": ["aws.elasticbeanstalk"],

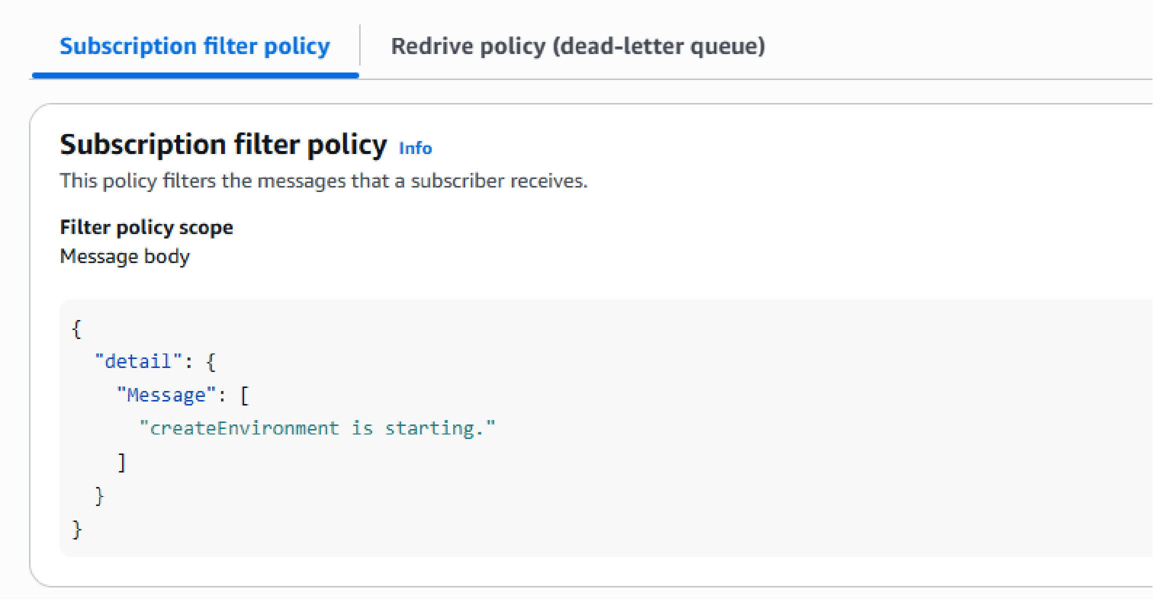
```
  "detail-type": ["Elastic Beanstalk resource status change"],
  "detail": {
    "Message": ["createEnvironment is starting.", "terminateEnvironment completed successfully."]
  }
}
```

This rule ensures only relevant lifecycle events trigger notifications (SNS).

2.5 Amazon SNS (Notification Layer)

SNS receives matched events from EventBridge and distributes them to subscribers. You can use Subscription filter policy specifying attribute names and adding a list of one or more values to each name. A subscription filter will accept the message only if the message contains attributes that match those specified in the subscription's filter policy.

Filter policy for **createEnvironment**


Subscription filter policy
Redrive policy (dead-letter queue)
Info

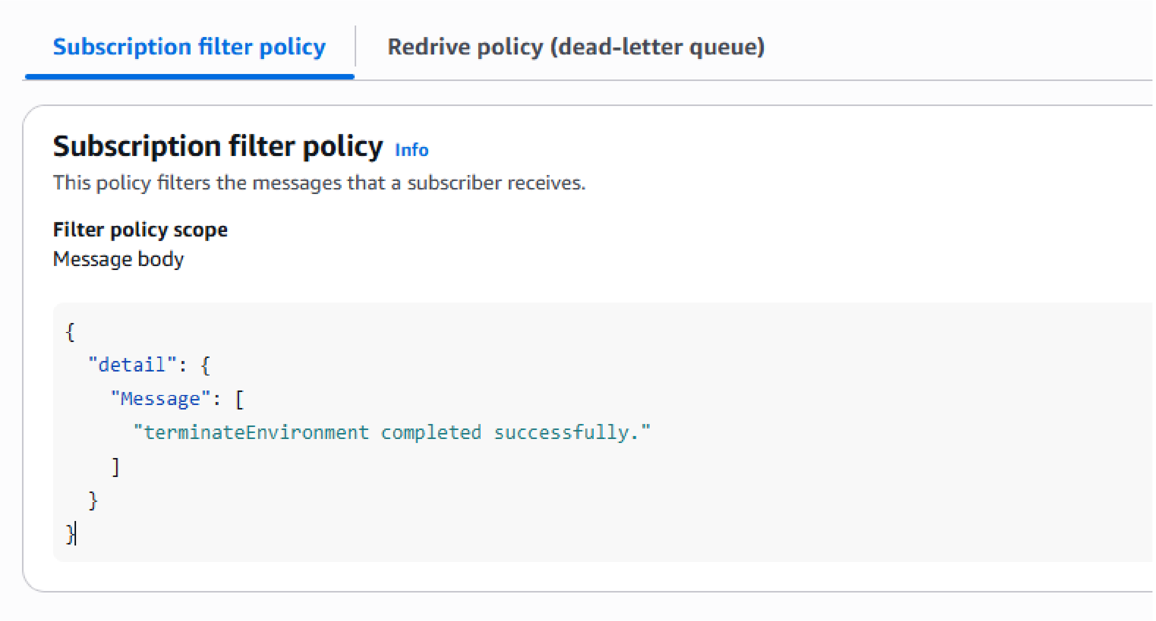
Subscription filter policy

This policy filters the messages that a subscriber receives.

Filter policy scope
Message body

```
{
  "detail": {
    "Message": [
      "createEnvironment is starting."
    ]
  }
}
```

Filter policy for **terminateEnvironment**


Subscription filter policy
Redrive policy (dead-letter queue)

Subscription filter policy Info

This policy filters the messages that a subscriber receives.

Filter policy scope

Message body

```
{
  "detail": {
    "Message": [
      "terminateEnvironment completed successfully."
    ]
  }
}
```

Configuration:

- Topic with email/SMS subscription
- Subscription must be confirmed
- Filter policy scope for message body

3. Event Flow in Detail

1. Scheduler triggers Lambda via cron.
2. Lambda calls Elastic Beanstalk API.
3. EB provisions or terminates resources.
4. EB emits status events to Default Event Bus.
5. EventBridge rule matches event pattern.
6. SNS publishes notification.
7. Email delivered to stakeholders.

4. IAM & Security Considerations

- Lambda execution role must allow EB API calls.
- EventBridge must have permission to publish to SNS.
- SNS topic policy should allow events.amazonaws.com to publish.
- Principle of least privilege should be applied.

5. Best Practices

- Keep rules on Default Bus for AWS service events.

- Use custom bus only when forwarding is required.
- Use Input Transformers for structured SNS messages.
- Log Lambda responses clearly.
- Implement retry and idempotency in Lambda.

6. Summary

This architecture delivers fully automated Elastic Beanstalk environment lifecycle management with real-time operational notifications, eliminating the need for manual infrastructure provisioning or decommissioning. By leveraging EventBridge scheduling and Lambda automation, environments are dynamically created and terminated based on defined business requirements such as operating only during business hours ensuring compute resources are consumed strictly when needed. The solution optimizes cost efficiency by preventing unnecessary runtime outside operational windows, supports governance through event-driven monitoring, and provides immediate visibility via SNS notifications. With no manual intervention required for provisioning, scaling, or teardown, the system enables zero-touch infrastructure management while maintaining operational transparency and control.