Contents

Technical Specification: EC2 Auto-Recovery System with Predictive Failure Detection .. 3

1. Solution Overview 3

1.1 Purpose 3

1.2 Scope 3

1.3 Target Audience 4

2. Technical Architecture 4

2.1 Architecture Pattern 4

2.2 Component Architecture 5

3. AWS Services and Integration 7

3.1 Primary AWS Services 7

3.2 External Integrations 8

4. Deployment Requirements 9

4.1 Prerequisites 9

4.2 Deployment Steps 10

4.3 Configuration Parameters 11

5. Performance Characteristics 12

5.1 Latency 12

5.2 Scalability 13

5.3 Resource Utilization 13

6. Security 13

6.1 IAM Permissions 13

6.2 Data Security 14

6.3 Compliance 14

7. Monitoring & Observability 15

7.1 CloudWatch Logs 15

7.2 Custom CloudWatch Metrics 15

8. Error Handling & Resilience 15

9. Cost Optimization 16

10. Limitations & Best Practices 16

11. Support & Maintenance 17

12. Version History 17

13. References 17

Technical Specification: EC2 Auto-Recovery System with Predictive Failure Detection

Document Information

- **Solution Name:** EC2 Auto-Recovery System with Predictive Failure Detection
- **Version:** 1.0
- **Date:** November 2024
- **Document Type:** Technical Specification
- **Classification:** Public

---

1. Solution Overview

1.1 Purpose

This document defines the technical specifications for the EC2 Auto-Recovery System — a fully serverless, event-driven automation platform built to monitor EC2 instances, predict failures using CloudWatch metric analysis, and automatically perform recovery actions with no manual intervention.

1.2 Scope

This specification covers:

- System architecture and component breakdown
- AWS service integrations
- Deployment prerequisites & steps

- Configuration variables & environment parameters
- Performance metrics and reliability characteristics
- Security, IAM, encryption, and compliance controls
- Operational considerations and limitations

1.3 Target Audience

- DevOps & Cloud Engineers
- AWS Solutions Architects
- SRE & Operations Teams
- Platform Engineering Teams

---

2. Technical Architecture

2.1 Architecture Pattern

Architecture Pattern: Event-Driven, Fully Serverless

Key Characteristics:

- No EC2-based system components
- EventBridge-driven orchestration
- Stateless Lambda processing
- DynamoDB for state/config/history
- CloudWatch for observability and metric ingestion

- SSM for automated on-instance recovery

2.2 Component Architecture

2.2.1 Lambda Functions

Function	Runtime	Memory	Timeout	Purpose
Predictive Monitor	Python 3.12	512 MB	300s	Analyzes 7-day CloudWatch metrics and generates predictions
Health Monitor	Python 3.12	512 MB	300s	Responds to EC2 status checks and triggers reactive recovery
Auto-Recovery Engine	Python 3.12	512 MB	900s	Performs restart, failover, EBS repair, SSM, quarantine
Notification Handler	Python 3.12	256 MB	60s	Sends alerts to Slack, Teams, SNS

2.2.2 EventBridge Rules

Name	Type	Schedule/Pattern	Target
Predictive Schedule	Scheduled Rule	rate(1 hour)	Predictive Monitor

Name	Type	Schedule/Pattern	Target
EC2 Status Monitor	Event Pattern	EC2 state/status change events	Health Monitor
Recovery Trigger	Event Pattern	High-confidence predictions	Auto-Recovery Engine
Notification Trigger	Event Pattern	Recovery complete event	Notification Handler

---

2.2.3 DynamoDB Tables

Table	Partition Key	Sort Key	Purpose	TTL
recovery-events	instance_id	timestamp	History of recovery actions	No
prediction-events	instance_id	timestamp	Prediction results	30 days
instance-config	instance_id	—	Per-instance policies & settings	No

Table Config:

- Billing: On-Demand
- Encryption: Enabled
- PITR: Enabled
- Streams: Optional

2.2.4 SSM Documents

Document	Type	Purpose
restart-services	Command	Restart application/system services
verify-health	Command	Post-recovery verification
diagnostics	Command	Collect diagnostics before/after action

---

3. AWS Services and Integration

3.1 Primary AWS Services

AWS Lambda

- Python 3.12
- Docker-based packaging
- Environment variables for config
- JSON structured logging

Amazon EventBridge

- Orchestrates predictive workflow
- Executes reactive triggers
- Routes recovery-complete notification events

Amazon CloudWatch

- Metrics: CPU, Memory, Disk, Network
- Logs for all Lambdas
- Optional custom alarms

Amazon DynamoDB

- Fully-managed NoSQL store
- Used for history, predictions, configuration

AWS Systems Manager (SSM)

- Executes automated commands on EC2
- Requires SSM agent installed on target instances

Amazon SNS (Optional)

- Notification fallback/alternative

3.2 External Integrations

Slack Webhook

- JSON over HTTPS
- Channel, username configurable
- Uses environment variable for secure injection

Microsoft Teams Webhook

- JSON adaptive card format
- Optional

---

4. Deployment Requirements

4.1 Prerequisites

AWS Requirements

- Enabled services: Lambda, EventBridge, DynamoDB, SSM
- Permissions for Terraform deployment
- Region: Any (default: us-east-1)

Local Requirements

- Terraform ≥ 1.0
- Docker
- Python 3.12
- AWS CLI configured

EC2 Requirements

- CloudWatch Agent installed
- SSM Agent installed
- IAM instance profile with SSM permissions

- Tagged with:

AutoRecovery=enabled

4.2 Deployment Steps

1. Backend Setup (Optional)

cd infra/terraform/backend

terraform init

terraform apply

2. Deploy Main Infrastructure

cd infra/terraform

terraform init -backend-config="backend/backend-config.tfvars"

terraform plan

terraform apply

3. Package Lambda

./scripts/package-lambda.ps1 -OutputFile "infra/terraform/lambda_package.zip"

4. Configure EC2 Instances

- Add AutoRecovery=enabled tag
- Update instance-config table

4.3 Configuration Parameters

Terraform Variables

Variable	Type	Default	Description
aws_region	string	us-east-1	Deployment region
environment	string	prod	Deployment environment
project_name	string	ec2-auto-recovery	Prefix for resources
slack_webhook_url	string	—	Slack integration
slack_channel	string	#general	Slack channel
slack_username	string	EC2 Auto-Recovery	Username
teams_webhook_url	string	—	Optional Teams webhook
enable_sns	bool	false	Enable SNS

Lambda Environment Variables

Variable	Used By	Purpose
RECOVERY_EVENTS_TABLE	All	Table for recovery logs
PREDICTION_EVENTS_TABLE	Predictive Monitor	Table for predictions

Variable	Used By	Purpose
INSTANCE_CONFIG_TABLE	All	Per-instance config
SLACK_WEBHOOK_URL	Notifier	Slack integration
SLACK_CHANNEL	Notifier	Slack channel
SLACK_USERNAME	Notifier	Display name
TEAMS_WEBHOOK_URL	Notifier	Teams integration
SNS_TOPIC_ARN	Notifier	Optional SNS
NOTIFICATION_ENABLED	All	Global flag
HIGH_CONFIDENCE_THRESHOLD	Predictive Monitor	Default: 0.8
PREDICTION_LOOKBACK_HOURS	Predictive Monitor	Default: 168

---

5. Performance Characteristics

5.1 Latency

Operation	Typical	Notes
Predictive Analysis	30–60s	Metric query dependent
Health Check	5–10s	EC2 API latency
Recovery Actions	60–600s	Restart/failover dependent

Operation	Typical	Notes
Notifications	1–3s	Slack/Teams response

5.2 Scalability

- Supports **unlimited EC2 instances**
- No bottlenecks due to serverless design
- EventBridge + Lambda scale automatically

5.3 Resource Utilization

Service	Typical	Peak
Lambda Invocations	100–1000/day	10,000/day
DynamoDB Reads	500–5000/day	50,000/day
DynamoDB Writes	100–1000/day	10,000/day

---

6. Security

6.1 IAM Permissions

Lambda Execution Role

- EC2: describe/start/stop/reboot/snapshot

- CloudWatch: get metric data
- DynamoDB: read/write tables
- SSM: send command
- SNS: publish
- Logs: create/put log streams

Principle: Least privilege only.

6.2 Data Security

- DynamoDB encrypted at rest
- All network communications via TLS
- Encrypted Lambda environment variables
- S3 Terraform state encrypted

6.3 Compliance

- Audit logs stored in DynamoDB
- CloudWatch logging for all actions
- Supports SOC2 / ISO 27001 audit trails
- PITR enabled

---

7. Monitoring & Observability

7.1 CloudWatch Logs

Log Group	Retention	Purpose
Predictive Monitor	14 days	Metric analysis logs
Health Monitor	14 days	Status check logs
Auto-Recovery	30 days	Recovery workflow logs
Notification Handler	14 days	Notification logs

7.2 Custom CloudWatch Metrics

Namespace: EC2/AutoRecovery

Metrics:

- RecoveryActionsExecuted
- RecoverySuccessRate
- PredictionsGenerated
- HighConfidencePredictions

---

8. Error Handling & Resilience

- Automatic retries with backoff

- Idempotent recovery routines
- Graceful fallback when SSM fails
- Notification failures do not block recovery

---

9. Cost Optimization

Estimated Monthly Cost (100 EC2 Instances)

Service	Cost
Lambda	$5–20
EventBridge	$1–5
DynamoDB	$10–50
CloudWatch	$5–15
SNS	$0–5

Total: ~$21–95/month

---

10. Limitations & Best Practices

Limitations

- Requires SSM + CloudWatch Agent
- Single-region unless deployed again

- Lambda cold starts (1–3s)

Best Practices

- Monitor logs regularly
- Tune prediction thresholds
- Test recovery flows in non-prod

---

11. Support & Maintenance

- No maintenance downtime required
- Updates via Terraform
- Debugging via CloudWatch Logs
- DynamoDB tables provide incident history

---

12. Version History

Version	Date	Notes
1.0	November 2024	Initial Release

---

13. References

AWS Docs

- Lambda — https://docs.aws.amazon.com/lambda/
- EventBridge — https://docs.aws.amazon.com/eventbridge/
- DynamoDB — https://docs.aws.amazon.com/dynamodb/
- SSM — https://docs.aws.amazon.com/systems-manager/

Project Resources

- Repository: https://github.com/Sumanth12-afk/ec2-auto-recovery-system
- README: https://github.com/Sumanth12-afk/ec2-auto-recovery-system/blob/main/README.md
- Commands: https://github.com/Sumanth12-afk/ec2-auto-recovery-system/blob/main/commands.md
- Challenges: https://github.com/Sumanth12-afk/ec2-auto-recovery-system/blob/main/challenges.md
- Architecture Diagram: https://github.com/Sumanth12-afk/ec2-auto-recovery-system/blob/main/docs/architecture-diagram.png

---

Document Status: Published

Last Updated: November 2024

Next Review: As Required