# **AWS Secret Manager Parameter Store & AWS Web Application Firewall (WAF)**

**Overall Task Title:**

**Implementation of AWS Secrets Manager, Systems Manager Parameter Store, and Web Application Firewall (WAF) for Secure Application Management**

**Overall Objective:**

To implement a secure and centralized approach for managing application secrets and configuration data using AWS Secrets Manager and AWS Systems Manager Parameter Store, while protecting the application from unauthorized access and malicious traffic using AWS Web Application Firewall (WAF). This task demonstrates secure credential storage, controlled access to sensitive parameters, and web-layer security enforcement for applications deployed on AWS.

**Overall Prerequisites:**

* Active AWS account with permissions for Secrets Manager, Systems Manager (Parameter Store), WAF & Shield, EC2, and Elastic Load Balancing.
* An existing application deployed on EC2 or behind an Application Load Balancer.
* IAM roles and policies configured to allow secure access to secrets and parameters.
* Sample secrets (database credentials, API keys) available for storage and testing.
* Application Load Balancer configured in the same AWS region as WAF resources.
* Public client IP address available for validating WAF rules.

## AWS Secret Manager Parameter

**Task Title**

AWS Systems Manager Parameter Store – Creating and Retrieving Parameters Using Console and CLI

**Objective**

To securely store application configuration values and sensitive information using AWS Systems Manager Parameter Store, and to retrieve them using AWS CLI with and without decryption, including hierarchical parameter access.

**Prerequisites**

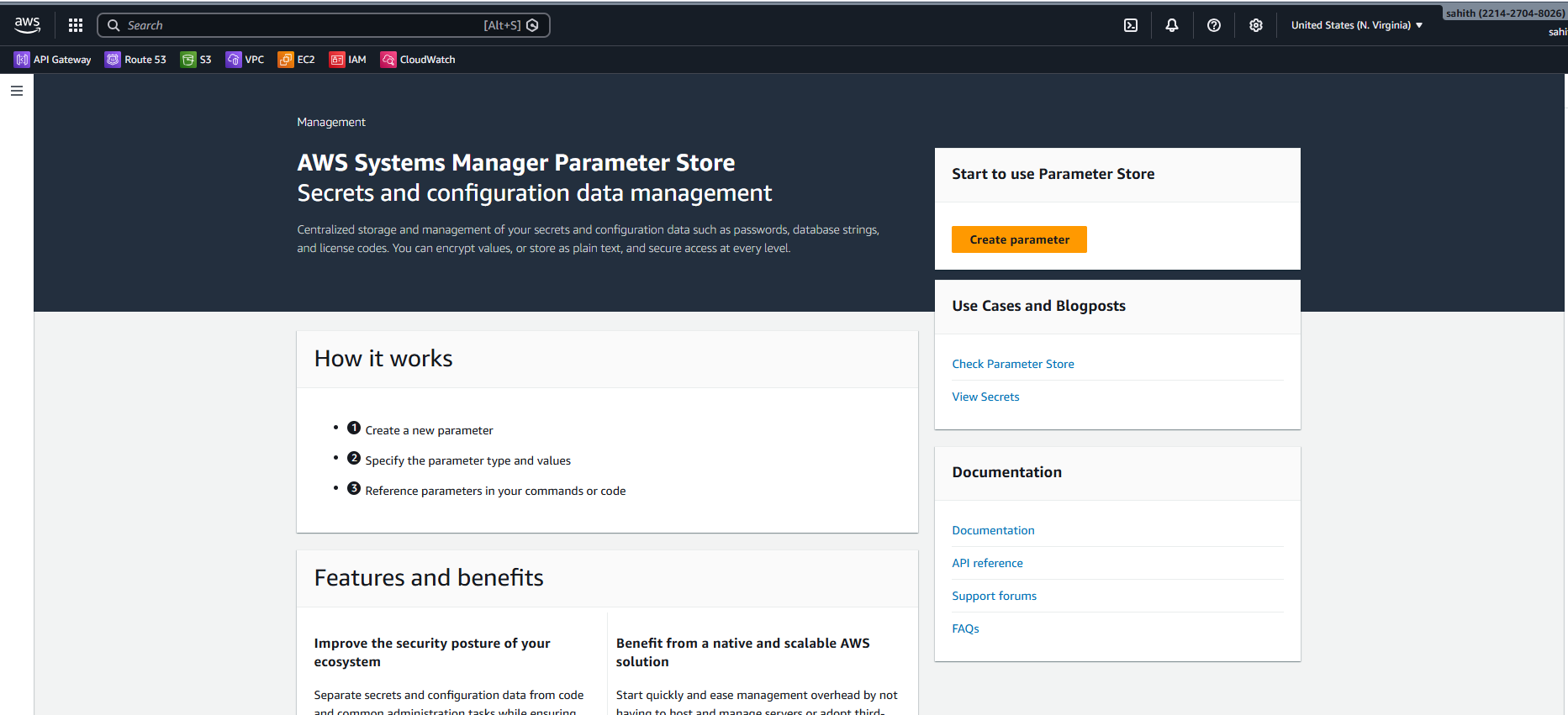
* Active AWS account
* IAM user or role with ssm:GetParameter, ssm:GetParameters, and ssm:GetParametersByPath permissions
* AWS CLI installed and configured
* AWS region set to **us-east-1**
* Access to AWS Systems Manager in the same region

**Steps Performed Using AWS Systems Manager Parameter Store**

* Opened **AWS Management Console** and navigated to **AWS Systems Manager**.
* Selected **Parameter Store** from the left navigation panel.
* Clicked on **Create parameter** to add a new parameter.

**Creating a Standard String Parameter**

* Entered the **parameter name** as task.
* Added an optional **description** for the parameter.
* Selected **Tier** as **Standard**.
* Chose **Type** as **String**.
* Kept **Data type** as text.
* Entered the parameter **value** as task.parameterstore.
* Left **Tags** empty.
* Clicked **Create parameter**.
* Verified that the parameter task appeared successfully in **My parameters** list.



A screenshot of a computer

AI-generated content may be incorrect.

A white rectangular object with a black border

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Creating a SecureString Parameter (Password)**

* Clicked **Create parameter** again.
* Entered the **parameter name** as task-password.
* Added description as taskparameter password.
* Selected **Tier** as **Standard**.
* Chose **Type** as **SecureString**.
* Selected **KMS key source** as **My current account**.
* Used the default KMS key **alias/aws/ssm**.
* Entered the sensitive **password value** in the value field.
* Clicked **Create parameter**.
* Confirmed successful creation from the green success message.
* Verified task-password is listed with **Type = SecureString**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Viewing and Verifying Secure Parameter**

* Clicked on the parameter name **task-password**.
* Opened the **Overview** tab.
* Verified details such as:
  + Name
  + ARN
  + Tier (Standard)
  + Type (SecureString)
* Enabled **Show decrypted value** to confirm the stored password.

A screenshot of a computer

AI-generated content may be incorrect.

**Creating a Hierarchical Parameter (Database URL)**

* Clicked **Create parameter** again.
* Entered the **parameter name** using hierarchy format:  
  /my/prod/db/url
* Added description as my product database url.
* Selected **Tier** as **Standard**.
* Chose **Type** as **String**.
* Set **Data type** to text.
* Entered the value prod.app.task.com:3306.
* Left **Tags** empty.
* Clicked **Create parameter**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Viewing Parameters in Parameter Store**

* Opened **AWS Systems Manager → Parameter Store**.
* Viewed **My parameters** list.
* Confirmed the following parameters exist:
  + /myapp/prod/url-db (String)
  + task (String)
  + task-password (SecureString)
* Verified parameters were created successfully from the green success banner.

A screenshot of a computer

AI-generated content may be incorrect.

**Fetching Parameters Using AWS CLI (Without Decryption)**

* Opened **PowerShell**.
* Executed the aws ssm get-parameters command with:
  + Parameter names: task, task-password
  + Region: us-east-1
* Observed output:
  + task value returned in plain text.
  + task-password returned as an **encrypted value**.
* Confirmed no invalid parameters in the response.

A black screen with white text

AI-generated content may be incorrect.

**Fetching Parameters Using AWS CLI (With Decryption)**

* Executed the same aws ssm get-parameters command.
* Added the --with-decryption flag.
* Observed output:
  + task value returned normally.
  + task-password value returned in **decrypted plain text**.
* Verified SecureString decryption works correctly.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Fetching Parameters by Path (Non-Recursive)**

* Executed aws ssm get-parameters-by-path.
* Provided path as /myapp/prod.
* Set region to us-east-1.
* Observed output:
  + Retrieved /myapp/prod/url-db parameter.
  + Verified database URL value and metadata.

A computer screen shot of a computer code

AI-generated content may be incorrect.

**Fetching Parameters by Path (Recursive)**

* Executed aws ssm get-parameters-by-path.
* Provided path as /myapp/prod.
* Enabled --recursive option.
* Set region to us-east-1.
* Observed output:
  + Retrieved all parameters under the specified path.
  + Verified parameter name, value, version, ARN, and data type.

A screenshot of a computer

AI-generated content may be incorrect.

**Validation:**

* Verified parameters are visible under **AWS Systems Manager → Parameter Store → My parameters**
* Confirmed creation of:
  + Standard String parameter (task)
  + SecureString parameter (task-password)
  + Hierarchical parameter (/myapp/prod/url-db)
* Successfully retrieved parameters using:
  + aws ssm get-parameters without decryption (SecureString returned encrypted)
  + aws ssm get-parameters --with-decryption (SecureString returned decrypted)
  + aws ssm get-parameters-by-path for hierarchical access
* No invalid parameters returned in CLI output

**Conclusion:**

AWS Systems Manager Parameter Store was successfully used to manage both plain text and sensitive configuration values. Parameters were securely stored, retrieved, and decrypted using AWS CLI. Hierarchical parameters enabled organized access using path-based queries, demonstrating a secure and scalable approach for application configuration management in AWS.

## AWS Web Application Firewall (WAF)

**Task Title**

Configure AWS WAF Protection Pack with IP-Based Rules for an Application Load Balancer

**Objective**

To secure an Application Load Balancer using AWS WAF by creating a protection pack (web ACL), configuring IP sets, and implementing IP-based rules to control traffic access. The task also demonstrates validating WAF behaviour by blocking and allowing requests from a specific client IP address.

**Prerequisites**

* Active AWS account with required permissions for **WAF & Shield**, **EC2**, and **Load Balancer**.
* An existing **Application Load Balancer** deployed and running.
* Backend EC2 instances registered in the target group.
* Knowledge of the client public IP address to be added to the WAF IP set.
* All resources created and configured in the **same AWS region** (us-east-1).

**Steps Performed – VPC Creation**

* Opened **AWS Management Console** and navigated to **VPC → Your VPCs**.
* Clicked **Create VPC**.
* Selected **VPC only** under Resources to create.
* Entered **Name tag** as web01.
* Chose **IPv4 CIDR manual input**.
* Entered IPv4 CIDR block as 178.168.0.0/16.
* Selected **No IPv6 CIDR block**.
* Kept **Tenancy** as Default.
* Selected **None** under VPC encryption control.
* Added tag with:
  + Key: Name
  + Value: web01
* Clicked **Create VPC**.
* Verified VPC creation from the success message and VPC dashboard.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Internet Gateway Creation and Attachment**

* Navigated to **VPC → Internet Gateways**.
* Clicked **Create internet gateway**.
* Entered **Name tag** as igwweb.
* Added tag with:
  + Key: Name
  + Value: igwweb
* Clicked **Create internet gateway**.
* Selected the newly created internet gateway.
* Attached the internet gateway to the VPC web01.
* Verified successful attachment from the green success banner.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Subnet Creation (Public Subnets)**

* Navigated to **VPC → Subnets**.
* Clicked **Create subnet**.
* Selected VPC web01.
* For Subnet 1:
  + Entered subnet name as public-1a.
  + Selected Availability Zone us-east-1a.
  + Chose IPv4 VPC CIDR block 178.168.0.0/16.
* Added second subnet configuration.
* For Subnet 2:
  + Entered subnet name as public-1b.
  + Selected Availability Zone us-east-1b.
  + Entered IPv4 subnet CIDR block 178.168.2.0/24.
  + Added tag with:
    - Key: Name
    - Value: public-1b
* Clicked **Create subnet**.
* Verified successful creation of two subnets from the success message.
* Confirmed both subnets are in **Available** state under the Subnets list.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Two subnets has been created sucessfully

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Route Table Creation**

* Navigated to **VPC → Route tables**.
* Clicked **Create route table**.
* Entered route table name as webpublic-rt.
* Selected VPC web01.
* Added tag with:
  + Key: Name
  + Value: webpublic-rt
* Clicked **Create route table**.

**Steps Performed – Subnet Association with Route Table**

* Opened the newly created route table.
* Selected **Edit subnet associations**.
* Selected subnets:
  + public-1a
  + public-1b
* Saved subnet associations successfully.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Adding Internet Route**

* **Opened Edit routes for the route table.**
* **Added a new route with:**
  + **Destination: 0.0.0.0/0**
  + **Target: Internet Gateway igwweb**
* **Saved route changes.**
* **Verified route status as Active.**

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – EC2 Instance Launch**

* Navigated to **EC2 → Instances**.
* Clicked **Launch instance**.
* Entered instance name as WAF-Public.
* Selected **Amazon Linux 2023 AMI**.
* Selected instance type t3.micro.
* Selected existing key pair NV.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Network Configuration**

* Selected VPC web01.
* Selected subnet public-1a.
* Enabled **Auto-assign public IP**.
* Selected existing security group (default).

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – User Data Configuration**

* Added user data script to:
  + Update packages
  + Install Apache HTTP server
  + Start and enable httpd
  + Create a custom index.html showing hostname and IP address

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Instance Launch Verification**

* Launched the EC2 instance.
* Verified instance state as **Running**.
* Confirmed **2/2 status checks passed**.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Security Group Inbound Rules**

* Opened the instance’s security group.
* Edited inbound rules.
* Allowed:
  + SSH (port 22) from 0.0.0.0/0
  + HTTP (port 80) from 0.0.0.0/0
* Saved inbound rules.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Apache Service Verification**

* Connected to the EC2 instance.
* Checked Apache service status.
* Verified httpd service is:
  + Loaded
  + Enabled
  + Actively running
  + Listening on port 80

A computer screen with text and numbers

AI-generated content may be incorrect.

**Steps Performed – Application Access Verification**

* Opened a web browser.
* Accessed the EC2 instance using its **public IP address**.
* Verified the web page displays:
  + Server hostname
  + Private IP address

A white and blue screen

AI-generated content may be incorrect.

**Steps Performed – EC2 Instances Verification**

* Opened **EC2 → Instances**.
* Verified multiple EC2 instances are present:
  + waf-01 in running state.
  + waf-02 in running/initializing state.
  + WAF-Public instance showing terminated state.
* Confirmed instances are in different Availability Zones (us-east-1a, us-east-1b).

A screenshot of a computer

AI-generated content may be incorrect.

**Target Group Creation**

* Navigated to **EC2 → Target Groups**.
* Clicked **Create target group**.
* Selected **Target type** as **Instances**.
* Entered target group name as waf.
* Selected protocol **HTTP**.
* Entered port **80**.
* Proceeded to **Register targets**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Registering Targets**

* Selected available EC2 instances:
  + waf-01
  + waf-02
* Confirmed both instances are in **Running** state.
* Set port for selected instances as **80**.
* Registered the selected instances.
* Completed target group creation.
* Verified success message indicating target group creation.

**Steps Performed – Target Group Verification**

* Opened the newly created target group waf.
* Verified:
  + Target type: Instance
  + Protocol: HTTP
  + Port: 80
  + Total targets: 2
* Checked target distribution across Availability Zones.
* Confirmed target health status section is visible

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Load Balancer Type Selection**

* Navigated to **EC2 → Load Balancers**.
* Clicked **Create load balancer**.
* Selected **Application Load Balancer (ALB)**.

**Steps Performed – Application Load Balancer Configuration**

* Entered load balancer name as Appload-waf.
* Selected **Internet-facing** scheme.
* Selected **IPv4** IP address type.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Network Mapping for ALB**

* Selected VPC web01.
* Selected Availability Zones:
  + us-east-1a with subnet public-1a
  + us-east-1b with subnet public-1b

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Security Group Configuration for ALB**

* Selected existing **default security group**.
* Associated the security group with the load balancer.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Listener and Routing Configuration**

* Configured listener with:
  + Protocol: **HTTP**
  + Port: **80**
* Selected **Forward to target groups** as default action.
* Selected target group **waf**.
* Kept weight as **100%**.
* Did not enable target group stickiness.

A screenshot of a computer

AI-generated content may be incorrect.

**Steps Performed – Load Balancer Creation Verification**

* Created the Application Load Balancer successfully.
* Verified load balancer Appload-waf is in **Active** state.
* Confirmed load balancer type as **Application**.
* Verified scheme as **Internet-facing**.
* Confirmed it is deployed across **two Availability Zones**.
* Copied the **DNS name** of the load balancer for access.

A screenshot of a computer

AI-generated content may be incorrect.

* Verified in browser using dns URL from load balancer

A computer screen shot of a computer

AI-generated content may be incorrect.

**Steps Performed**

* Opened **AWS Management Console** and navigated to **WAF & Shield**.
* Selected **Protection packs (web ACLs)** from the left menu.
* Clicked on **Create protection pack (web ACL)**.
* Selected **App category** as **Other**.
* Chose **App focus** as **Both API and web**.
* Selected the resource **Appload-waf** of type **Application Load Balancer** to protect.
* Reviewed the **estimated cost** for the protection pack.
* Entered the protection pack name as **waf-task**.
* Proceeded to create the protection pack.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Navigated to **IP sets** under **AWS WAF**.
* Created an IP set named **waf-ipset**.
* Added a specific IPv4 address (/32) to the IP set to represent a single client IP.
* Saved the IP set successfully.

A screenshot of a computer

AI-generated content may be incorrect.

* Opened the **waf-task** protection pack.
* Chose to **Add new rule**.
* Selected **Custom rule**.
* Chose **IP-based rule** as the rule type.
* Set the rule action to **Block**.
* Provided the rule name as **blockmylaptop**.
* Selected **Use existing IP set** and chose **waf-ipset**.
* Kept **Source IP address** as the request origin.
* Added and saved the rule.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Navigated to **EC2 → Load Balancers**.
* Selected the Application Load Balancer **Appload-waf**.
* Copied the **DNS name** of the load balancer.

A screenshot of a computer

AI-generated content may be incorrect.

* Copied the **DNS name** of the load balancer.
* Accessed the ALB DNS URL from a browser.
* Observed **403 Forbidden** response, confirming that the WAF rule is blocking traffic from the specified IP

A screenshot of a computer

AI-generated content may be incorrect.

**Validation Steps Performed :**

* Opened **WAF & Shield → Protection packs (web ACLs)**.
* Selected the existing protection pack **waf-task**.
* Opened the rule **blockmylaptop** for editing.
* Changed the **Action** from **Block** to **Allow**.
* Kept the rule name as **blockmylaptop**.
* Confirmed the condition **If a request matches the statement**.
* Verified the inspection condition as **Originates from an IP address in**.
* Selected the existing IP set **waf-ipset**.
* Saved the rule changes.

A screenshot of a computer

AI-generated content may be incorrect.

* Copied the **Application Load Balancer DNS name**.
* Accessed the ALB URL in the browser.
* Successfully received the application response instead of a 403 error.
* Verified server details displayed:
* Hostname of the EC2 instance
* Private IP address of the backend server

A screen shot of a computer

AI-generated content may be incorrect.

**Conclusion:**

AWS WAF protection was successfully applied to the Application Load Balancer using a protection pack. An IP set was created and associated with an IP-based rule to control access. Blocking the specified IP resulted in a 403 Forbidden response, and changing the rule action to Allow restored access to the application. This confirms that AWS WAF rules are effectively filtering traffic based on defined security policies.

**Overall Task Conclusion:**

The assignment successfully integrates AWS Secrets Manager and Parameter Store to securely store and manage sensitive application data, reducing the risk of hard-coded credentials. AWS WAF was effectively configured to protect the application by enforcing IP-based access controls and validating traffic behaviour. Together, these services provide a robust security foundation by combining secure configuration management with proactive web application protection, enhancing the overall security posture of the AWS environment.