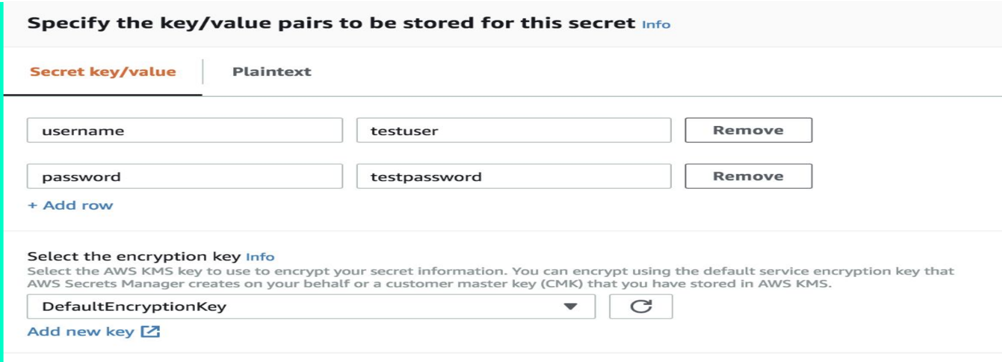
Terraform Recommendations

Use a Secrets Management Solution: Utilize a dedicated secrets management solution like HashiCorp Vault, Azure Key Vault, AWS Secrets Manager, or Google Cloud Secret Manager. These tools provide secure storage, encryption, access control, and auditability for secrets.

Separate Secrets from Code: Avoid hardcoding secrets directly into your pipeline scripts or code repositories. Instead, store them separately in a secrets management solution and retrieve them dynamically during pipeline execution.

Create an IAM Role: Create an IAM role in the AWS Management Console with the necessary permissions to access the AWS resources your CI/CD pipeline requires. Ensure that the role has the appropriate policies attached for the desired operations.



Specify the key/value pairs to be stored for this secret Info

Secret key/value | Plaintext

username	testuser	Remove
password	testpassword	Remove

+ Add row

Select the encryption key Info

Select the AWS KMS key to use to encrypt your secret information. You can encrypt using the default service encryption key that AWS Secrets Manager creates on your behalf or a customer master key (CMK) that you have stored in AWS KMS.

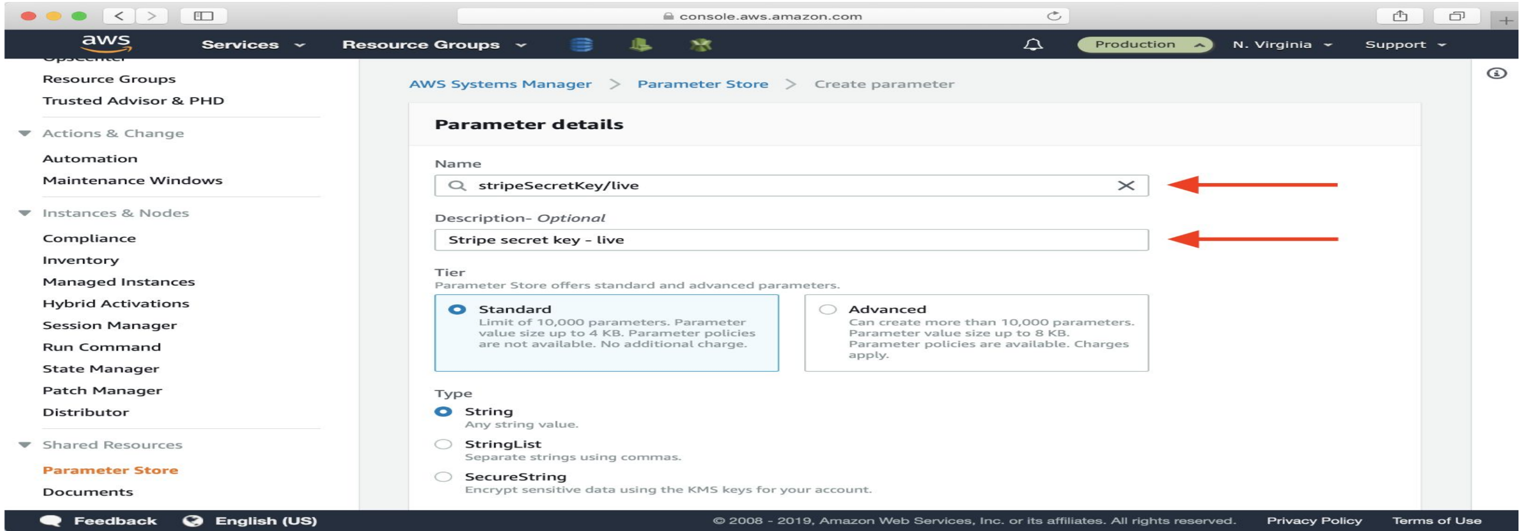
DefaultEncryptionKey

Add new key

Terraform Recommendations

Use Variables and Parameterization: Leverage Terraform variables to parameterize your infrastructure code. This allows you to customize configurations for different environments or deployments without modifying the code itself.

State Management: Consider using remote state storage, such as Amazon S3 or HashiCorp Consul, to store and share your Terraform state. Remote state storage enhances collaboration, enables team coordination, and provides a consistent view of the infrastructure.



AWS Systems Manager > Parameter Store > Create parameter

Parameter details

Name: stripeSecretKey/live

Description- *Optional*: Stripe secret key - live

Tier
Parameter Store offers standard and advanced parameters.

[✓ Standard] Limit of 10,000 parameters. Parameter value size up to 4 KB. Parameter policies are not available. No additional charge.

Advanced: Can create more than 10,000 parameters. Parameter value size up to 8 KB. Parameter policies are available. Charges apply.

Type

[✓ String] Any string value.

StringList: Separate strings using commas.

SecureString: Encrypt sensitive data using the KMS keys for your account.

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Terraform Recommendations

Monitoring and Logging: Incorporate monitoring and logging into your Terraform workflows. Leverage tools like AWS CloudWatch, Azure Monitor, or Google Cloud Logging to capture infrastructure events and metrics, enabling proactive monitoring and troubleshooting.

Regular Updates and Upgrades: Stay up to date with the latest Terraform version and provider releases to benefit from new features, bug fixes, and security patches. Regularly review and update your Terraform configurations to incorporate improvements and optimize your infrastructure.

Pinning the provider version in your Terraform configuration is an important best practice .

Consistency: Pinning the provider version ensures that your infrastructure remains consistent and reproducible across deployments.

Avoid Breaking Changes: Provider updates can introduce breaking changes or modifications to resource behavior, configuration options, or syntax.

Testing Terraform: Static analysis of Terraform code involves examining the code without actually executing it, aiming to identify potential issues, errors, or best practice violations

terraform validate

TFLint : is a popular open-source static analysis tool specifically designed for Terraform. It analyzes your Terraform code against a set of predefined rules to catch potential errors, security issues, and best practice violations.

Checkov : is an open-source static analysis tool that focuses on security scanning for infrastructure-as-code files, including Terraform.

Terrascan : is an open-source static analysis tool specifically designed for scanning Terraform code for security vulnerabilities and best practice violations. It helps identify potential misconfigurations, insecure settings, and adherence to industry standards and compliance requirements.