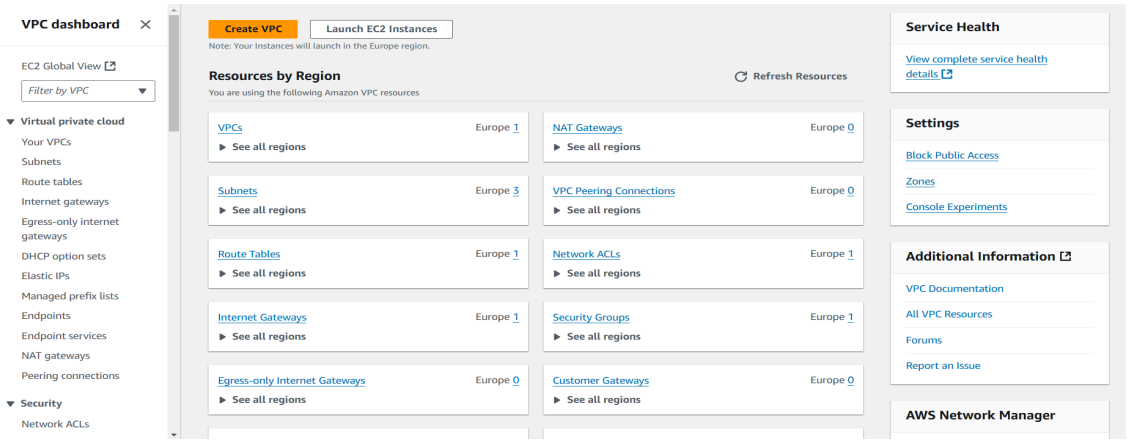
Defining a Parameter

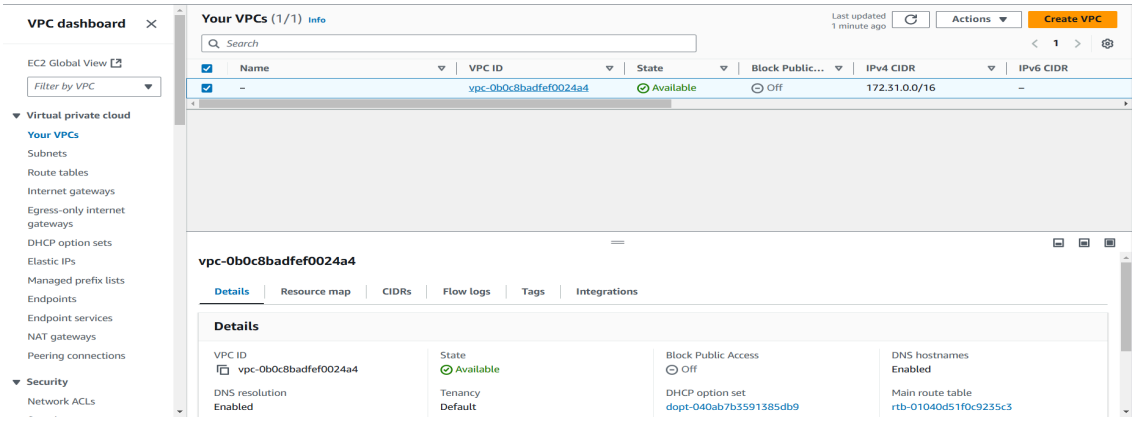
The process involves configuring and managing AWS CloudFormation stacks with advanced parameter handling. Start by locating the template file in the GitHub repository, edit it locally in VS Code, and update placeholders with actual values for VPC and subnet IDs. Define parameters like MasterUsername, MasterUserPassword, and DbClass with constraints, allowed values, and patterns. Upload the template to CloudFormation to create a stack, ensuring all parameter values meet defined rules. Test stack functionality by validating resource creation and updating parameters as needed. Use features like masking sensitive data (NoEcho). The goal is to securely automate infrastructure creation and management while ensuring parameter integrity.

Activity

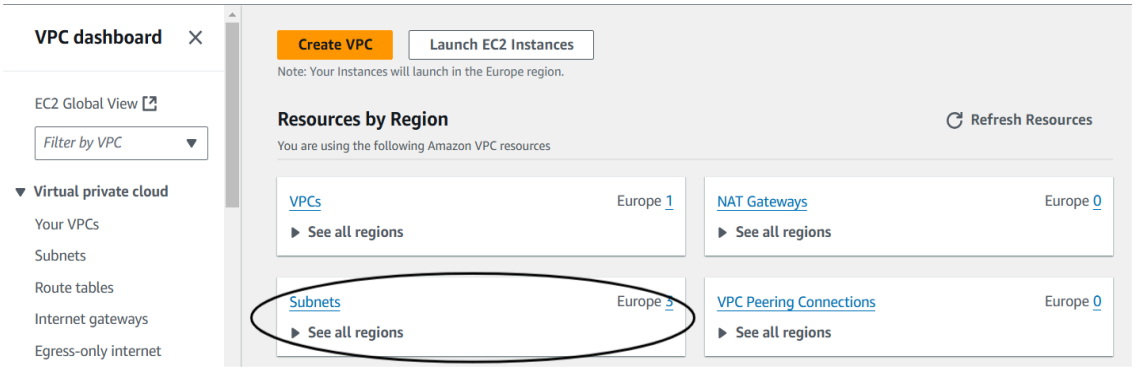
1. Find the template files in our GitHub repository under the same name as the heading for easy access and edits. Find and Save the attached template locally, open it in VS Code for edits.
2. Go to the AWS VPC Console to locate the default VPC ID.



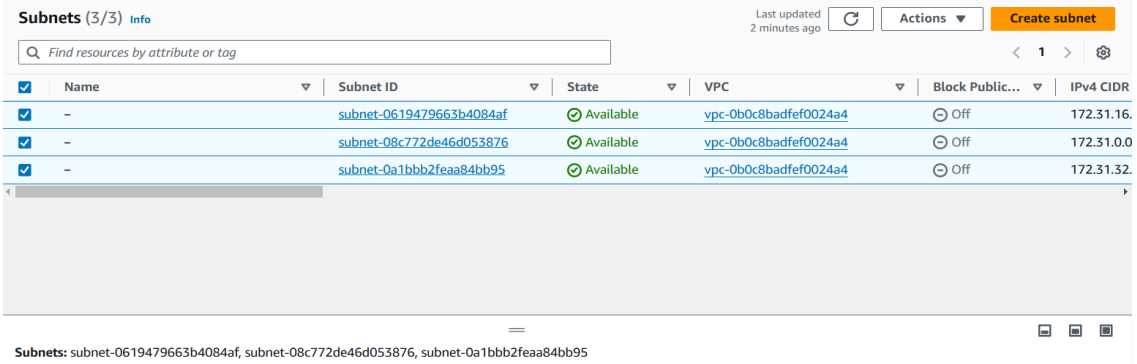
3. Select the default VPC and copy its VPC ID from the details tab. Replace the VPC ID placeholder in the template with the copied VPC ID.



4. Navigate to the VPC Console and locate the default subnet IDs.

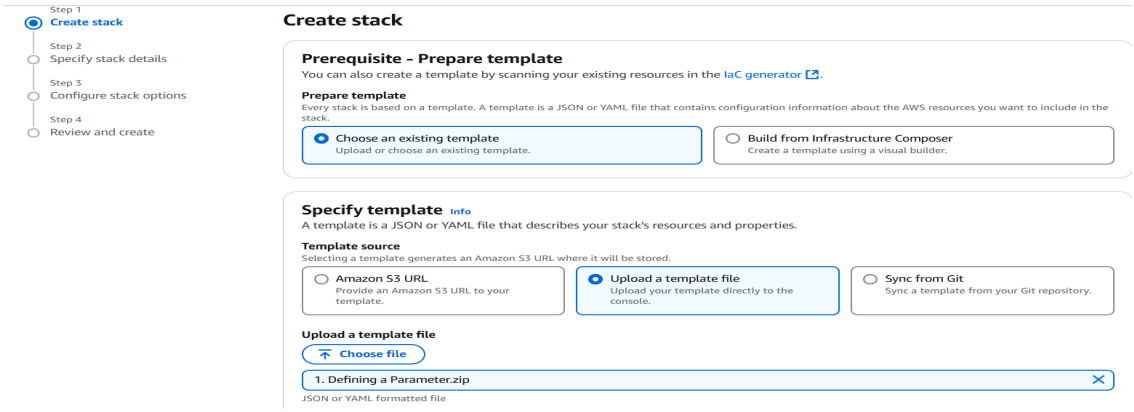


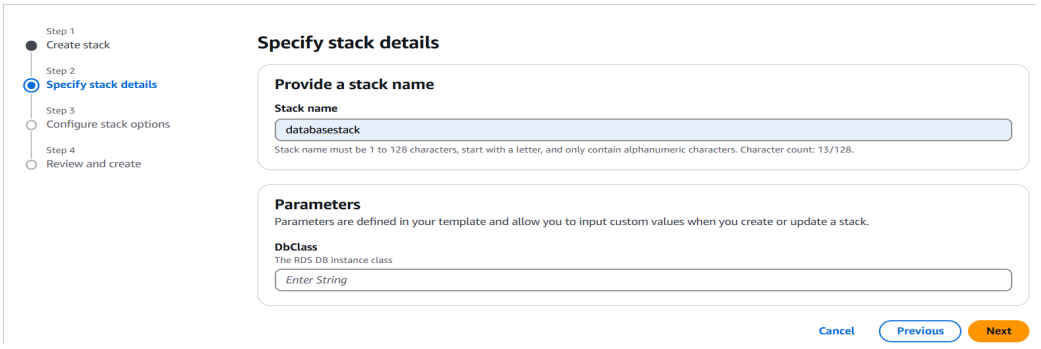
5. Copy the subnet IDs as a comma-separated array and update the template's placeholder.



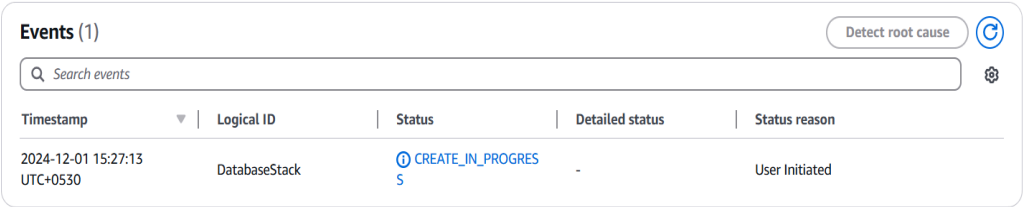
6. Define a parameter in the template by adding a 'Parameters' section. Configure the parameter name, type ('String'), and optional description. Reference the parameter in the 'DBInstanceClass' property using the intrinsic 'Ref' function.
7. Save the template and upload it to the CloudFormation Console to create a new stack.

8. Upload the template file and click 'Next'. Name the stack (e.g., 'DatabaseStack') and proceed.Review the new 'DbClass' parameter field in the parameters section. Provide a valid value for the 'DbClass' parameter (e.g., 'db.t4g.micro'). Click 'Next' to continue.

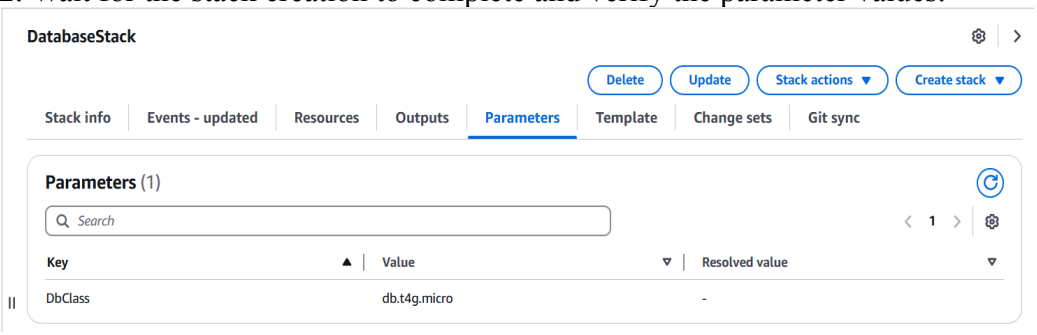




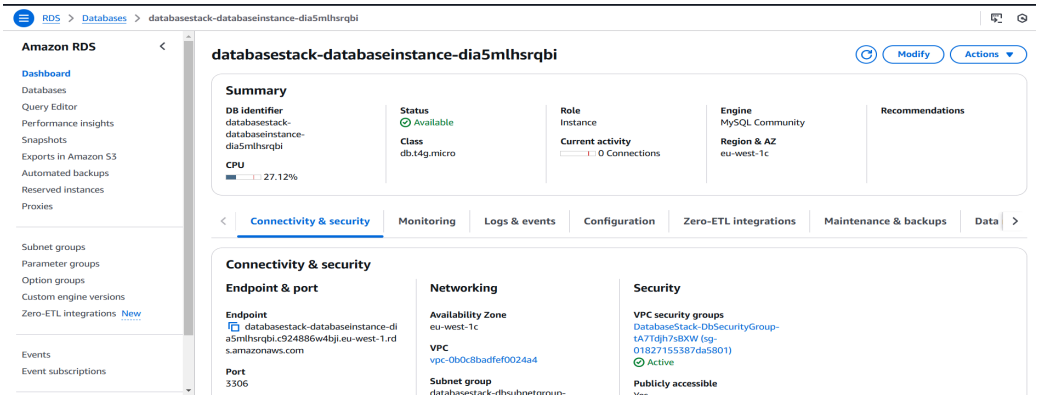
9. Optionally, assign a CloudFormation service role in the '**Permissions**' section to separate permissions, though this is advanced and can be skipped. Skip other advanced options not covered in this lecture. Click '**Next**' to proceed. Now, We are on the review page, reviewing your stack configuration and going back if needed. Scroll down to the bottom and click the 'Submit' button to initiate stack creation.
10. CloudFormation starts creating the stack and redirects you to the stack details.



11. Refresh the event list to monitor the stack creation progress.
12. Wait for the stack creation to complete and verify the parameter values.



13. Check the physical RDS DB instance details in the RDS Console.

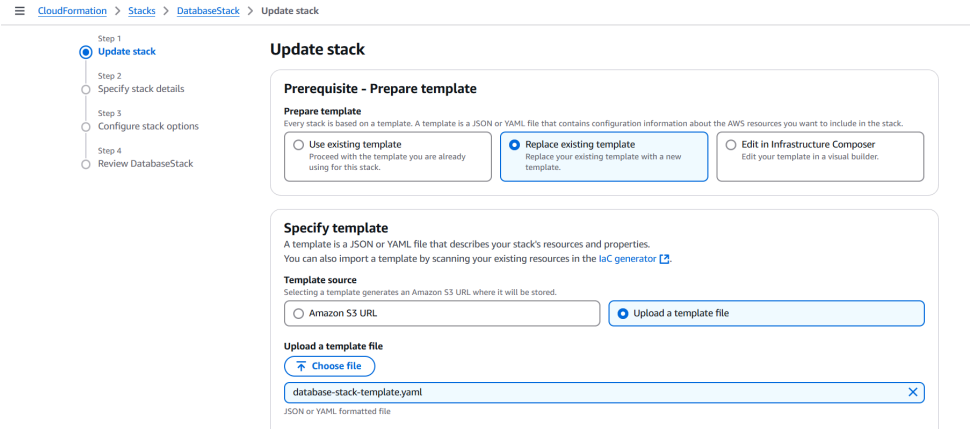


Updating your Stack with More Parameters

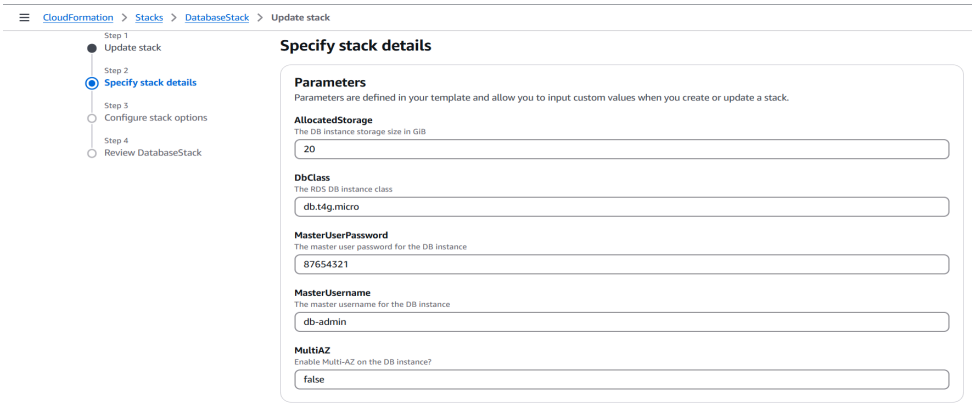
1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Add a parameter for the MasterUsername property with a type of String and a description. Create a parameter for the MasterUserPassword, also of type String, by copying the previous parameter and modifying the name and description. Define a parameter for the MultiAZ property, using type String for Boolean-like values (true or false) and include a question-style description.

```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: A sample database stack for the AWS CloudFormation Step by Step course series
Parameters:
  DbClass:
    Type: String
    Description: The RDS DB instance class
  MasterUsername:
    Type: String
    Description: The master username for the DB instance
  MasterUserPassword:
    Type: String
    Description: The master user password for the DB instance
  MultiAZ:
    Type: String
    Description: Enable Multi-AZ on the DB instance?
  AllocatedStorage:
    Type: Number
    Description: The DB instance storage size in GiB
```

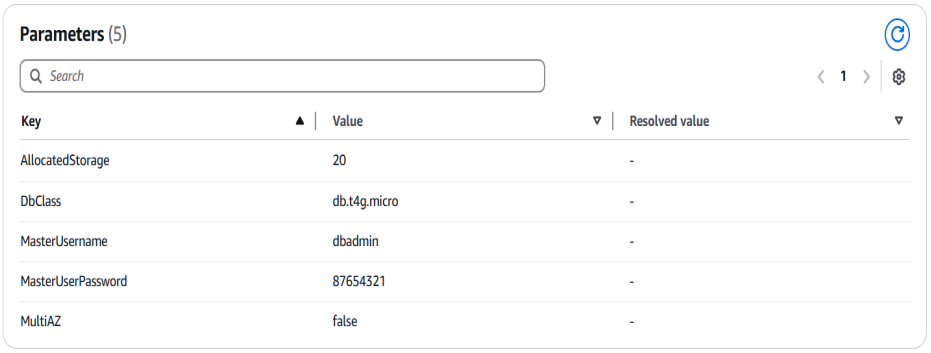
2. Add a Number-type parameter for the AllocatedStorage property, specifying it as the storage size in GiB and change the Image Id, Subnet id and Vpc id according to your region.
3. Save and Reference these new parameters in the corresponding resource properties within the CloudFormation template to update.



4. Click ‘Next’ and continue on the Parameter page and enter the following details.



5. Save the updated template and upload it to the CloudFormation console to update the existing stack.
6. Provide runtime values for the new parameters during the stack update, ensuring no unnecessary replacements occur.

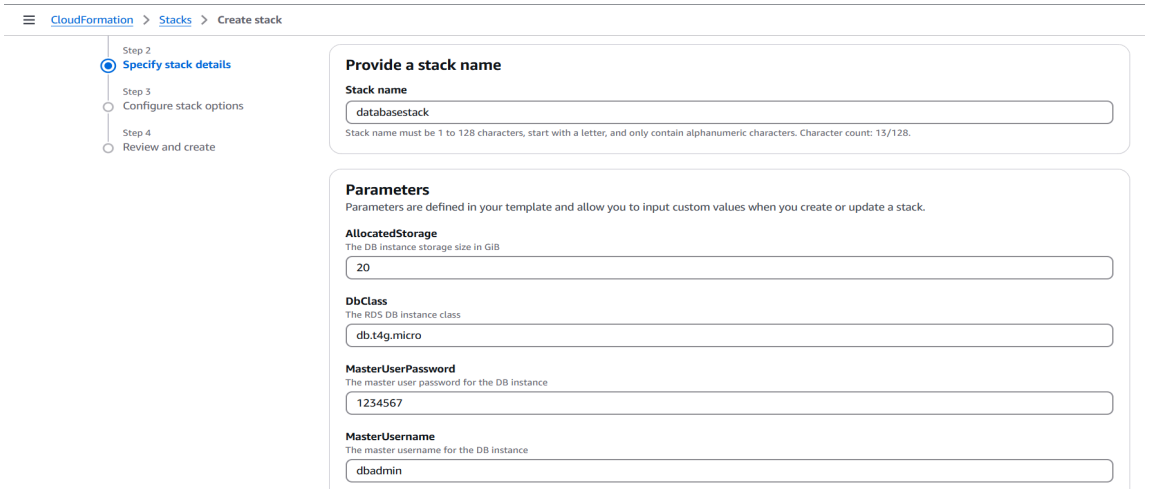


7. Test the changes by connecting to the updated RDS instance using the new MasterUserPassword.
8. Clean up by deleting the stack if needed and prepare for the next steps, such as adding constraints.

Setting Parameter Constraints

1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Start by creating a new stack in CloudFormation. Upload the database stack template file.

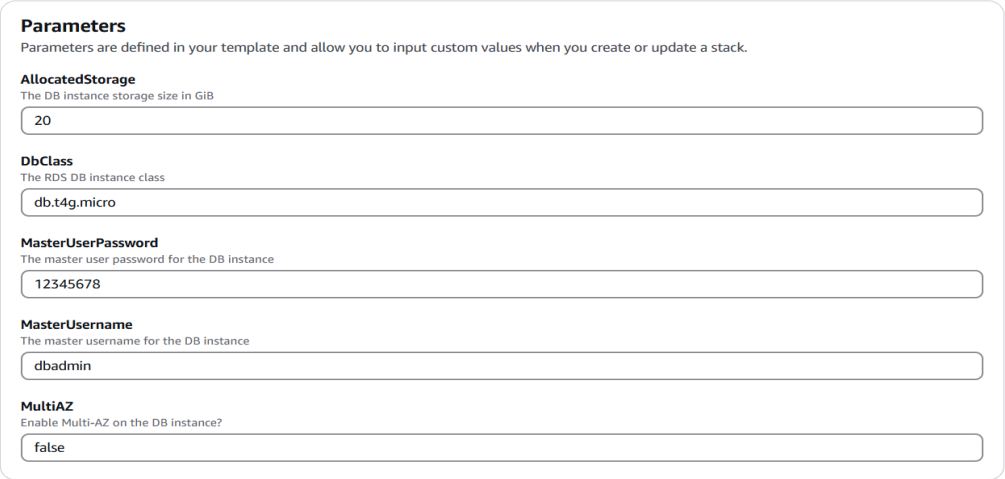
2. Name the stack (e.g., "DatabaseStack") and configure parameters. Provide an invalid MasterUserPassword (8 characters).



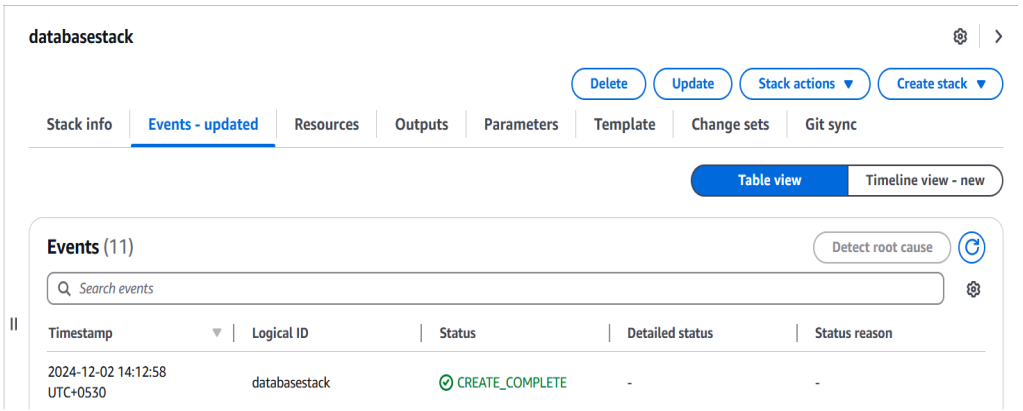
3. Proceed to stack creation, and observe the failure due to the invalid password. CloudFormation rolls back the stack creation. Delete the failed stack before creating a new one.
4. Modify the template to add a minimum length constraint (8 characters) for MasterUserPassword. Add a maximum length constraint (10 characters) for MasterUsername. Upload the updated template and create a new stack.

```
  Type: String
  Description: The master username for the DB instance
  MaxLength: 10
MasterUserPassword:
  Type: String
  Description: The master user password for the DB instance
  MinLength: 8
```

5. Provide invalid values for both MasterUserPassword and MasterUsername. Correct the MasterUserPassword value and proceed to the review page again.



6. Fix the MasterUsername value to comply with the maximum length constraint. Submit the stack creation after all parameters are valid.
7. Wait for successful stack creation.



8. Delete the stack to clean up the resources.

Timestamp	Logical ID	Status	Detailed status
2024-12-02 14:28:47 UTC+0530	databasestackk	DELETE_COMPLETE	-

Setting Value Constraints for Number Parameters

1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Define the minimum value constraint for the AllocatedStorage parameter as 20 GiB and Define the maximum value constraint for the AllocatedStorage parameter as 30 GiB. Add a custom error message for the AllocatedStorage constraint. Save the template and upload it to the CloudFormation console.

```
Description: The DB instance storage size in GiB
MinValue: 20
MaxValue: 30
ConstraintDescription: AllocatedStorage cannot be less than 20 or greater than 30.
```

2. Create a new stack and provide parameters, including an invalid AllocatedStorage value of 10 GiB.

Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

AllocatedStorage
The DB instance storage size in GiB
10

DbClass
The RDS DB instance class
db.t4g.micro

MasterUserPassword
The master user password for the DB instance
12345678

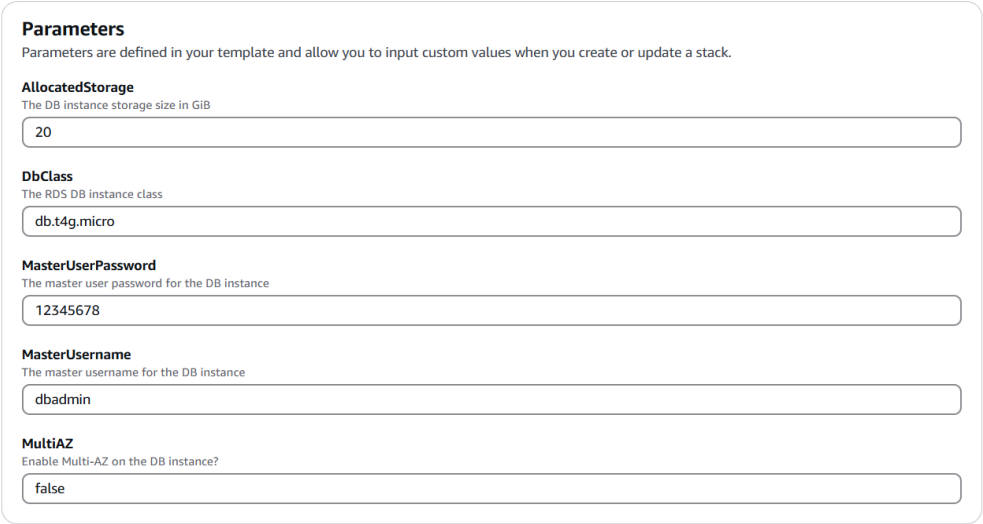
MasterUsername
The master username for the DB instance
dbadmin

MultiAZ
Enable Multi-AZ on the DB instance?
false

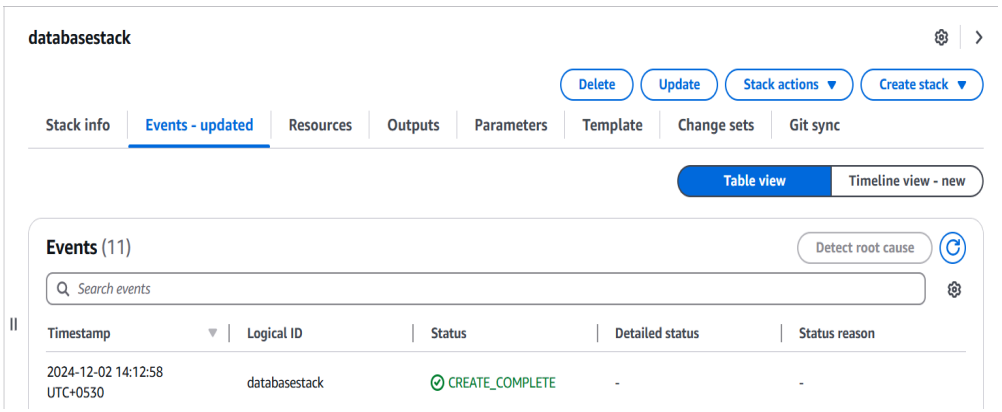
3. Review the error message for the AllocatedStorage constraint violation. Go back to the parameters and provide a value of 40 GiB for AllocatedStorage. Check the error message for the maximum value constraint violation.

Parameter AllocatedStorage failed to satisfy constraint: AllocatedStorage cannot be less than 20 or greater than 30.

4. Correct the AllocatedStorage value to 20 GiB, matching the minimum limit.



5. Successfully create the stack with valid parameter values. Wait for stack creation to complete, then delete the stack to clean up resources.



Setting Allowed Values for Your Parameters

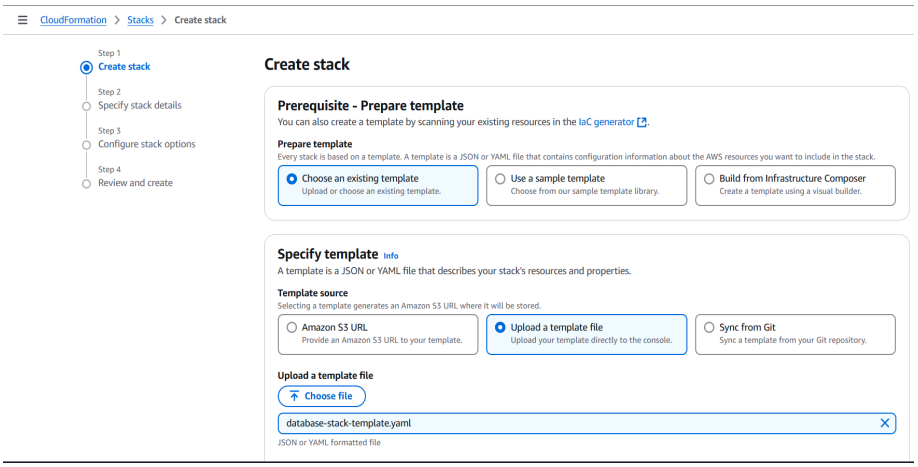
1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Define allowed values for the DbClass parameter using the 'AllowedValues' attribute.

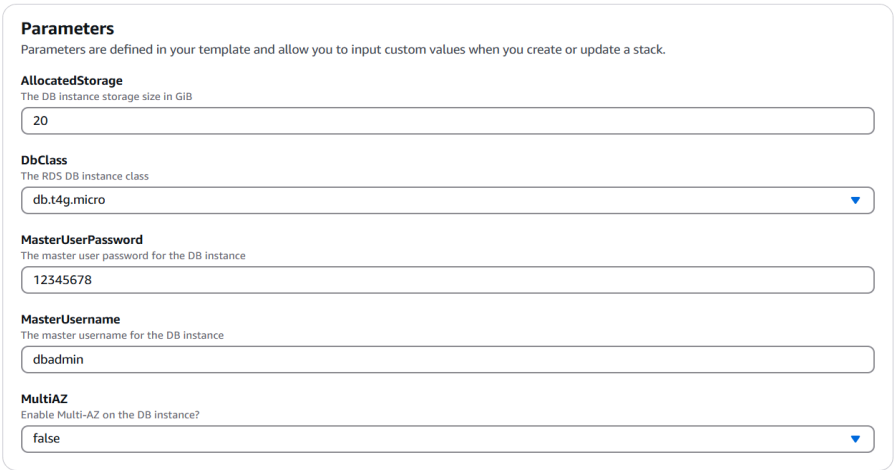
```
6     Description: The RDS DB instance class
7     AllowedValues:
8       - db.t4g.micro
9       -db.t3.micro
```

2. Define allowed Boolean values for the MultiAZ parameter.

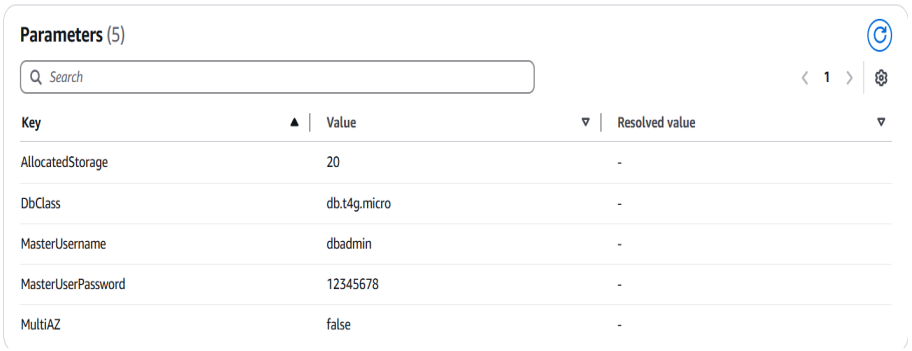
```
18  MultiAZ:
19    Type: String
20    Description: Enable Multi-AZ on the DB instance?
21    AllowedValues: [ true, false ]
22  AllocatedStorage:
```

3. Save the template and upload it to the CloudFormation Console.
4. Create a new stack and review the parameters, noting the dropdown for DbClass(choose ‘db.t4g.micro’) and MultiAZ(choose ‘false’).





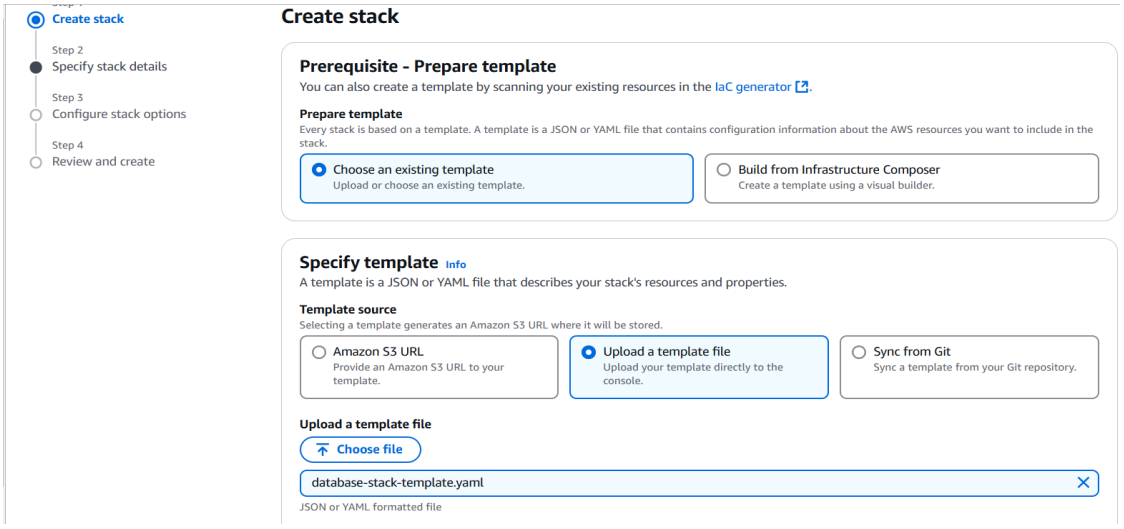
5. Submit the stack creation and ensure it proceeds successfully.
6. Verify the created DB instance details in the RDS console.



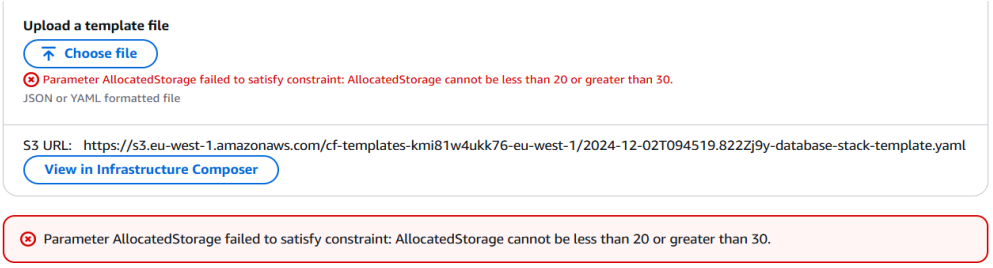
7. Delete the stack to clean up resources.

Setting Pattern Constraints for Your Parameters

1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Define a pattern constraint for the MasterUsername parameter using a regular expression. Write a regular expression to ensure the MasterUsername starts with a letter and can include numbers afterward. Save the template and upload it to CloudFormation.
2. Create a new stack, providing the master password and an invalid master username to test the pattern constraint.



3. Attempt to submit the stack, expecting a failure due to the AllocatedStorage.



4. Correct the AllocatedStorage and try with the suitable storage.

```
26      MaxValue: 30
27      ConstraintDescription: AllocatedStorage cannot be less than 20 or greater than 30.
28      Default: 10
29
```

5. Attempt to submit the stack again, expecting a failure due to the master username. Correct the master username, and set default values as shown and then, confirm the pattern and try submitting the stack again.

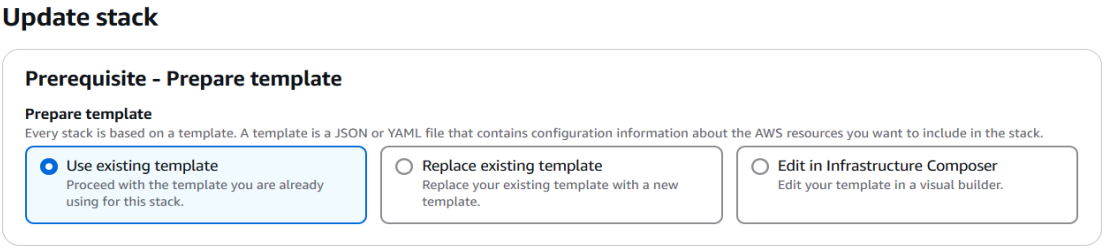
```
7        AllowedValues:
8          - db.t4g.micro
9          - db.t3.micro
10         Default: db.t4g.micro
11     MasterUsername:
12       Type: String
13       Description: The master username for the DB instance
14       MaxLength: 10
15       Default: dbadmin
```

```
21         AllowedValues: [ true, false ]
22         Default: false
23       AllocatedStorage:
```

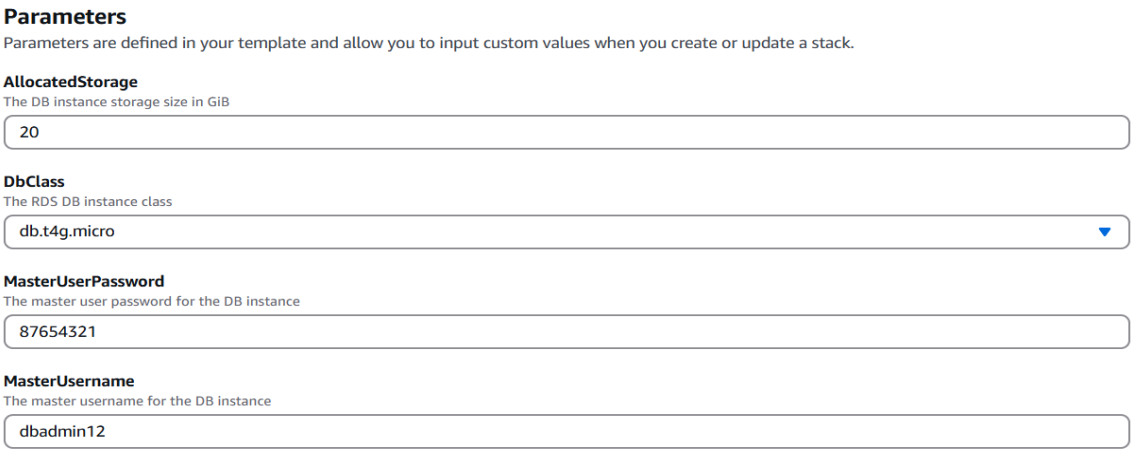
6. Verify that the stack creation proceeds successfully with a valid master username and click on ‘Next’ to submit and create a stack successfully.

Updating only Parameters of Stacks

1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Recreate the stack, if needed. Click the 'Update' button on your stack in the AWS CloudFormation Console. Select the 'Use existing template' option, and click 'Next'.



2. Modify the MasterUserPassword parameter in the parameters section. Ensure the new password meets the minimum length constraint (at least 8 characters). Click 'Next' twice to skip stack options and proceed to the review page.



3. Confirm the changes in the review page and click 'Submit'. Wait for the stack update process to complete.
4. Verify the new parameter value by connecting to the RDS instance using the updated credentials and Clean up by deleting the stack if no longer needed.

Parameters (5)

Search

Key	Value
AllocatedStorage	20
DbClass	db.t4g.micro
MasterUsername	dbadmin12
MasterUserPassword	87654321
MultiAZ	false

Hiding Parameter Values

1. Save the attached template locally, open it in VS Code for edits, and upload it during stack creation.Add the NoEcho attribute with the value true to the MasterUserPassword parameter in the template to enable masking. Repeat the same for the MasterUsername parameter in the template. Save the updated template file.

```
18      NoEcho: true
19    MasterUserPassword:
20      Type: String
21      Description: Master user password for the db instance
22      MinLength: 8
23      NoEcho: true
```

2. Create a new stack in the AWS Management Console and upload the updated template.

Create stack

Prerequisite - Prepare template

You can also create a template by scanning your existing resources in the IaC generator.

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

- (●) Choose an existing template — Upload or choose an existing template.
- ( ) Use a sample template — Choose from our sample template library.
- ( ) Build from Infrastructure Composer — Create a template using a visual builder.

Specify template Info

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

- ( ) Amazon S3 URL — Provide an Amazon S3 URL to your template.
- (●) Upload a template file — Upload your template directly to the console.
- ( ) Sync from Git — Sync a template from your Git repository.

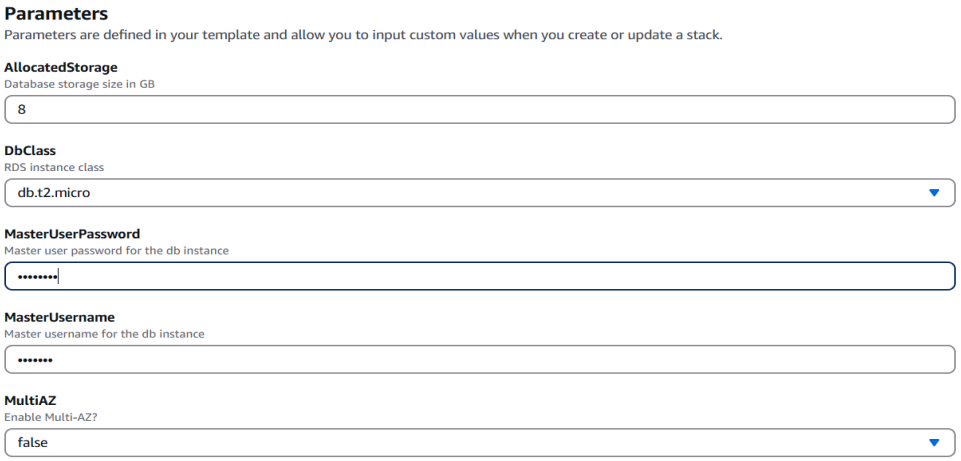
Upload a template file

Choose file

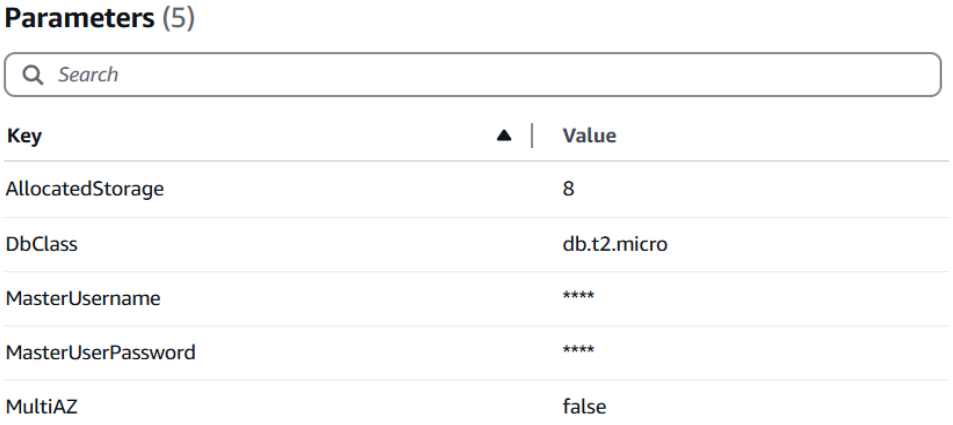
database-stack-template.yaml

JSON or YAML formatted file

3. Provide values for the masked parameters (MasterUsername and MasterUserPassword) during stack creation; their input will display as dots.



4. Review the parameters on the Review page, noting that masked values appear as asterisks. Complete the stack creation and verify the parameters under the Parameters tab, where masked values are hidden.



5. To update the stack, click Update stack and select the current template. Use the "use existing value" checkbox to retain masked parameter values or uncheck it to input new values. Review changes and update the stack, noting that changes to masked parameters (like MasterUsername) result in resource replacement. Verify the update stack's Parameters tab to ensure masked values remain hidden.
6. Delete the stack to clean up resources.