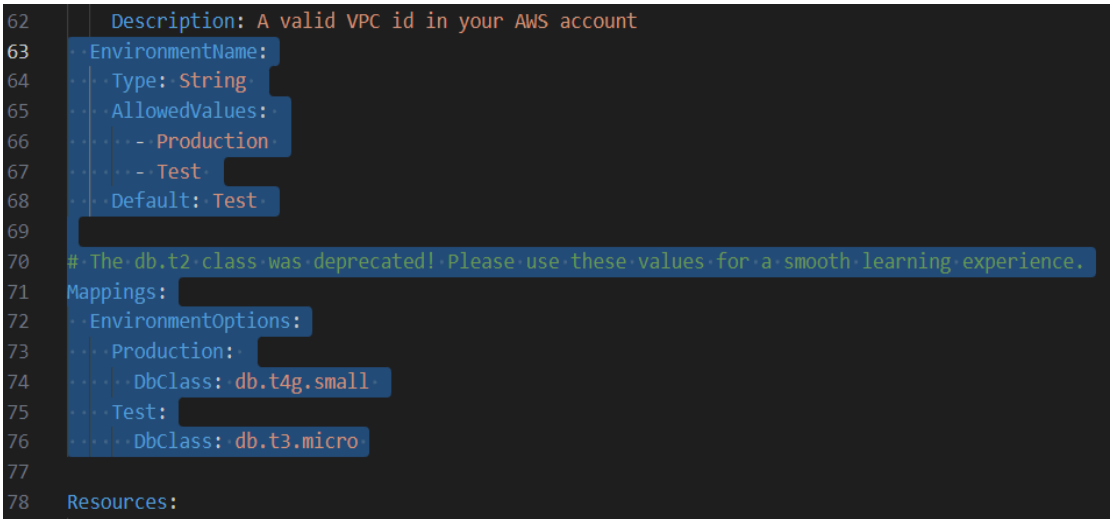
Using Mappings with FindMap Function

The process involves integrating a Mappings section in the CloudFormation template for environment-specific configurations. Add a mapping called EnvironmentOptions with top-level keys (e.g., Production and Test) and sub-keys (e.g., DbClass with respective values). Introduce an EnvironmentName parameter of type String with valid values (Production, Test) and a default value (Test). Replace the DbClass parameter with a dynamic mapping using Fn::FindInMap. Save and upload the updated template to the AWS CloudFormation Console. Create stacks for Test and Production environments, verify the corresponding instance classes (db.t3.micro and db.t2.small), and delete stacks to clean up resources.

Activity

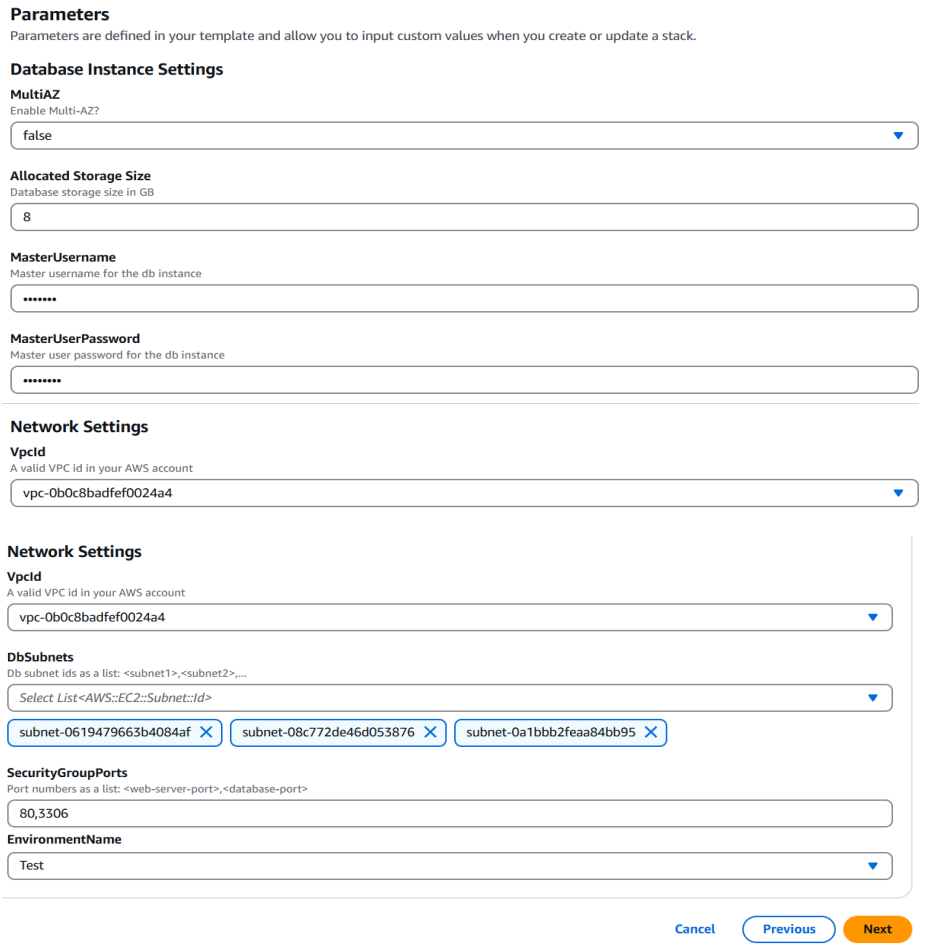
1. Find the template files in our GitHub repository under the same name as the heading for easy access and edits. Find and Save the attached template locally, open it in VS Code for edits.
2. Defined a Mappings section in the template between Parameters and Resources sections as a best practice. Created a mapping named EnvironmentOptions for environment-specific configurations. Added top-level keys for environments (Production and Test) under the mapping. Defined a second-level key (DbClass) with values (db.t2.small for Production and db.t2.micro for Test). Created a new parameter EnvironmentName of type String to capture the environment name during stack creation.



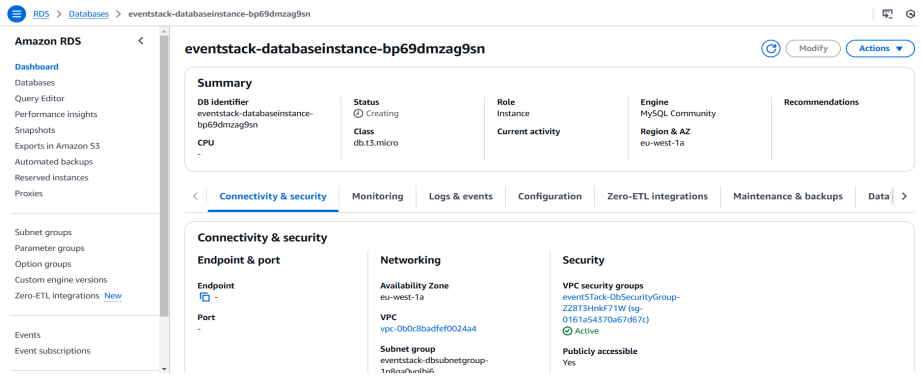
3. Limited the valid values for EnvironmentName to Production and Test and set a default value (Test). Removed the DbClass parameter from the template and metadata since it is now mapped dynamically.
4. Updated the DBInstanceClass property of the DatabaseInstance resource to use the Fn::FindInMap function with inputs EnvironmentOptions, EnvironmentName, and DbClass. Demonstrated both the long and short formats of Fn::FindInMap in the template.

```
125   DatabaseInstance:
126     Type: AWS::RDS::DBInstance
127     DeletionPolicy: Delete
128     Properties:
129       DBInstanceClass: !FindInMap [ EnvironmentOptions, !Ref EnvironmentName, DbClass ]
130       Engine: mysql
131       MultiAZ: !Ref MultiAZ
132       PubliclyAccessible: true
133       AllocatedStorage: !Ref AllocatedStorage
```

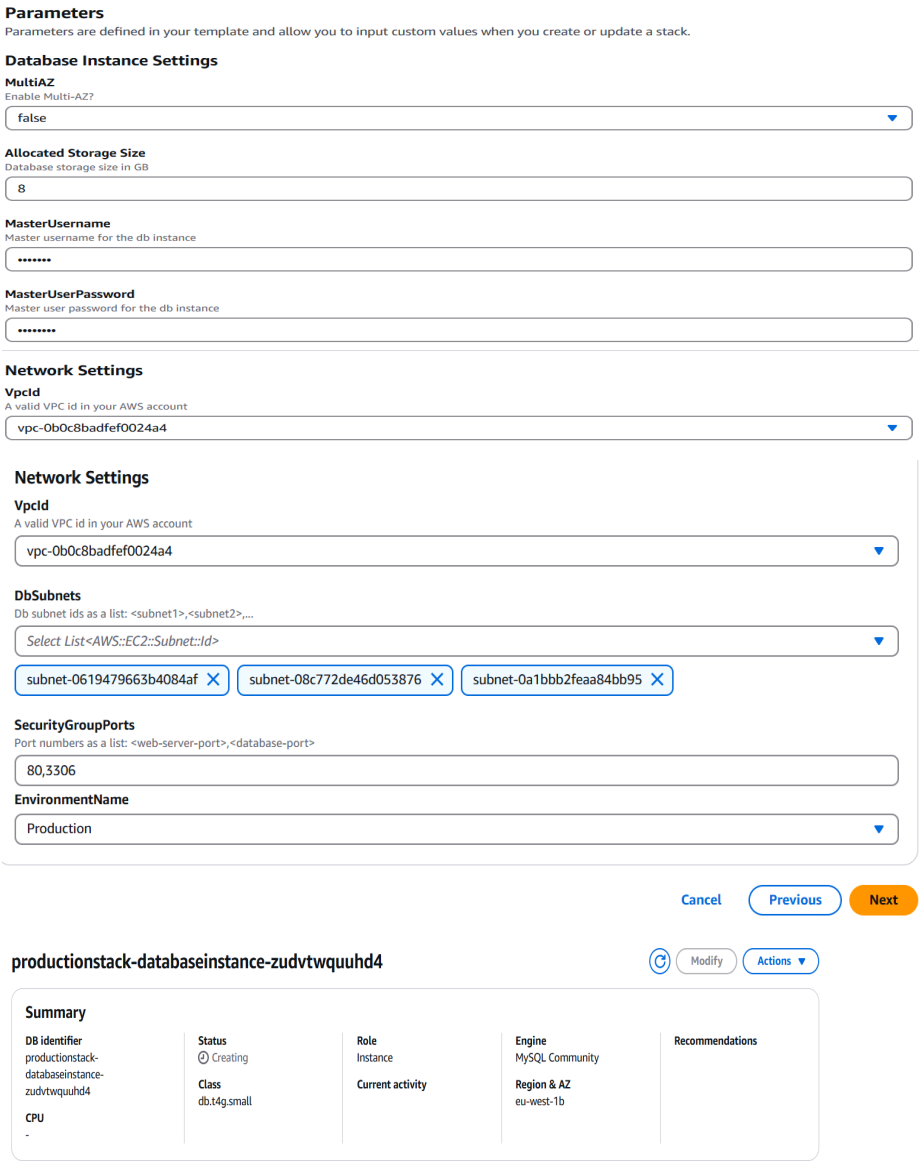
5. Change the Image Id, Subnet id and Vpc id according to your region.Saved and uploaded the updated template to AWS CloudFormation Console.
6. Created a stack for the test environment, verified the parameter grouping.



7. Observe the instance class from the mapping (db.t3.micro).



8. Created another stack for the production environment, selected Production for EnvironmentName, and verified the instance class from resource pane of stack(DataBaseInstance) and checked the class(db.t2.small).



9. Deleted both test and production stacks to clean up resources.