|  |
| --- |
|  |
| Cloud Infrastructure as Code |
|  |

|  |
| --- |
| Thong Thao  Student ID: 474587265 |

**Contents**

[PART 1 - Prepare to deploy and configure. 2](#_Toc161664928)

[PART 2 - Cloud infrastructure as code (IaC) pre-defined template 6](#_Toc161664929)

[PART 3 - Develop and update infrastructure as code (IaC) templates - Create own template 18](#_Toc161664930)

[PART 4 - Contingency task and cloud-related knowledge concepts 36](#_Toc161664931)

[Reference 42](#_Toc161664932)

# PART 1 - Prepare to deploy and configure.

* 1. Review the scenario presented. Research, evaluate, and select an appropriate cloud platform for the portfolio project. Provide a detailed justification for your selection.

**AWS (Amazon Web Services):**

AWS offers a wide range of cloud services, including computing, storage, databases, machine learning, and more. It has a vast global infrastructure with multiple availability zones and regions, providing high availability and scalability.

AWS has a rich ecosystem of tools, services, and third-party integrations, including AWS CloudFormation for infrastructure as code, AWS Lambda for serverless computing, and AWS Elastic Beanstalk for easy application deployment.

**Microsoft Azure:**

Azure seamlessly integrates with Microsoft's enterprise products, such as Windows Server, Active Directory, and Office 365, making it a preferred choice for organizations already invested in Microsoft technologies.

Azure provides a robust set of developer tools and services, including Azure DevOps for CI/CD pipelines, Visual Studio integration, and support for a wide range of programming languages.

**Google Cloud Platform (GCP):**

GCP excels in data analytics and machine learning capabilities with services like Big Query for data warehousing, TensorFlow for machine learning, and Dataflow for stream and batch processing.

GCP provides a global network infrastructure with low-latency, and high-performance networking, making it suitable for latency-sensitive applications and global deployments.

I suggest using Amazon Web Services (AWS) as the cloud platform since it provides a wide range of services including computing, storage, databases, networking, AI, and more, that are suitable for hosting a microservices-based application. Additionally, AWS offers a free tier that includes various services relevant to this project, enabling you to explore and experiment without worrying about immediate cost concerns.

**Compare and contrast three (3) infrastructure as code (IaC) technologies:**

1. Ansible

Ansible, developed by Red Hat, is an open-source tool that provides software provisioning, configuration management, and application deployment. Ansible uses human-readable YAML code for its definitions, making it easier for developers and system admins to understand.

1. Terraform

Terraform, developed by HashiCorp, is an open-source tool that allows you to define and provide data centre infrastructure using a declarative configuration language. Terraform is a platform-agnostic tool, meaning it works with multiple cloud service providers, including AWS. Terraform uses HashiCorp Configuration Language (HCL).

1. AWS CloudFormation

AWS CloudFormation is an AWS-native IaC tool, tightly integrated with AWS services, supports change sets for reviewing and applying changes and uses the standards notation language YAML or JSON, making it easier to learn and manage for beginners compared to Terraform and Ansible.

According to the compatibility of AWS, for scenarios that are not larger and more complex, the infrastructure as code (IaC) technology that can be used is AWS CloudFormation. However, if the project is larger or more complex, Terraform might be a better option due to its modularity and flexibility.

* 1. Research and review at least three (3) benefits of using infrastructure as code (IaC) for the scenario presented. Explain each benefit in detail and provide examples to illustrate your answer.

**Three benefits of IaC for the Portfolio Project**

Using Infrastructure as Code (IaC) offers several advantages for our portfolio projects, particularly in transitioning the LAMP stack application to a staging platform and then production.

* **Improved automation**: The simplicity of the template allows you to declare what you want your resources to look like. This eliminates the need to rely on other scripting tools to create the resources.
* **Quick infrastructure replication**: You can quickly replicate your infrastructure without affecting other resources that your template previously created. The template can be used to create as many stacks as needed.
* **Infrastructure consistency**: The declarative way of defining templates allows for consistency—you can be assured that stacks created with the template will be identical.

1.3 Research and determine at least three (3) ways or techniques in which automation can leverage cloud platforms. For each automation technique, provide a detailed explanation and example (s).

1. **Infrastructure Provisioning and Scaling.**

Cloud platforms offer a convenient solution to the time-consuming task of manually configuring servers, networks, and storage. With Infrastructure as Code (IaC) tools, you can use scripting templates to automatically create, configure, and manage cloud resources such as virtual machines, databases, and storage. For example, in our LAMP stack application, we initially deployed it on a single EC2 instance with RDS databases and load balancers. By defining a template that outlines the desired infrastructure, AWS CloudFormation can automatically provision additional resources based on the template.

1. **Continuous Integration and Deployment (CI/CD).**

CI/CD is an automated process that takes care of the entire software development lifecycle, from code changes to deployment. It guarantees that the code is tested, built, and deployed consistently and quickly. For instance, in our project, every update made to a microservice triggers a CI/CD pipeline. The pipeline automatically builds the code, executes unit and integration tests, and deploys the updated service to the staging environment.

1. **Backup and Disaster Recovery.**

Backup and disaster recovery (DR) strategies are essential to protect your data and applications from any loss due to accidental deletion, corruption, or catastrophic events. Cloud platforms provide powerful solutions for data protection and swift recovery. For instance, if we want to back up the databases for our microservices, we can utilize AWS RDS automated backups.

1.4 Identify three (3) types of problems or errors that can occur when infrastructure is implemented as code as opposed to manual processes. In your response consider testing and debugging techniques.

* Template Syntax Errors.

Infrastructure as Code (IaC) involves using programming languages or scripts like YAML or JSON to automate the deployment and configuration of infrastructure. However, sometimes deployment failures or misconfiguration can occur due to errors in the IaC code. To prevent these issues, it is important to test and debug the IaC templates before deployment. This can be done by using syntax validation tools or linters to check the syntax of the code for errors. If errors are found, debugging can be done by examining the error logs generated by the IaC tools. These logs often indicate the specific lines in the code that contain syntax errors, making it easy to pinpoint and fix them.

* Configuration Drift.

It is important to avoid making changes manually to resources outside of the Infrastructure as Code (IaC) management process. Such changes can create inconsistencies and deviations from the defined configuration.

To ensure proper testing and debugging, it is recommended to implement continuous monitoring and automated configuration validation. This can help to detect and prevent configuration drift. Additionally, debugging can be performed by comparing the current state of a resource with the IaC templates. This can reveal any discrepancies caused by drift.

Another helpful tool is AWS Config, which can be used to identify discrepancies between the actual and desired states of resources. By using these best practices, it is possible to maintain a consistent and reliable infrastructure.

* Role-Based Access Control

Managing Role-Based Access Control permissions within Infrastructure-as-Code (IaC) can be a challenging task. Misconfigured permissions may result in potential security vulnerabilities or operational disruptions. To avoid such issues, we can use the least privilege principle. Review and audit IAM policies regularly. Additionally, we can simulate permissions using tools such as aws-vault or assume-role to test and debug the setup.

1.5 For each problem or error identified, assess the potential consequences. Provide examples to illustrate your answer in reference to the scenario presented.

Based on the three identified problems in IaC implementation the potential consequences of each type of problem in the infrastructure as code (IaC):

|  |  |  |
| --- | --- | --- |
| **Problem/Error** | **Potential Consequences** | **Examples in Scenario** |
| Template Syntax Errors | Deployment failures, misconfigured resources | Missing commas or spelling, incorrect indentation in CloudFormation templates |
| Configuration Drift | Inconsistent environments, security vulnerabilities | Manual changes to deployed resources not reflected in IaC templates |
| Role-Based Access Control | Unauthorised access, security breaches | Misconfigured IAM roles granting excessive permissions to resources |

1.6 Review the functionality of the scenario presented and investigate any required industry or organisational standards, and legislative requirements necessary for the project. Document your findings.

1. Organisational Standards.

ISO/IEC 2700: It is an internationally recognized standard that provides a framework for establishing, implementing, maintaining, and continually improving an information security management system (ISMS). It ensures that organisations have robust processes in place to manage information security risks and protect sensitive data. Compliance with this standard demonstrates a commitment to maintaining the confidentiality, integrity, and availability of information assets.

PCI DSS: The Payment Card Industry Data Security Standard (PCI DSS) is a set of security standards designed to ensure the secure handling of credit card information during transactions. Compliance with PCI DSS is mandatory for organisations that process, store, or transmit payment card data. It includes requirements for securing networks, implementing strong access control measures, regularly monitoring, and testing systems, and maintaining information security policies. Failure to comply with PCI DSS can result in fines, penalties, and reputational damage.

2. Legislation

Privacy Act 1988 (Cth): The Privacy Act 1988 (Cth) is Australian legislation that regulates the handling of personal information by organisations. It sets out the Australian Privacy Principles (APPs), which govern the collection, use, storage, and disclosure of personal information. Compliance with the Privacy Act is mandatory for Australian government agencies and businesses that have an annual turnover above a certain threshold. Failure to comply with the Privacy Act can result in significant fines, penalties, and reputational damage.

Australian Cyber Security Centre (ACSC) Essential Eight.: The Australian Cyber Security Centre (ACSC) Essential Eight is a set of baseline security controls developed by the Australian government to help organisations mitigate cyber security incidents. It includes recommendations for implementing essential security measures, such as application whitelisting, patching applications, and restricting administrative privileges. Compliance with the ACSC Essential Eight helps organisations strengthen their cyber security defences and reduce the risk of cyber-attacks and data breaches.

# PART 2 - Cloud infrastructure as code (IaC) pre-defined template

2.1 Review the pre-defined template provided and determine and explain the resources and dependencies used. Provide detailed explanations for each resource.

* CFormationTemp-0-Network

|  |  |  |
| --- | --- | --- |
| **Resource** | **Description** | **Dependencies** |
| AWSTemplateFormatVersion | This is the current version template of AWS CloudFormation | None |
| Parameters | These are input parameters that allow users to customize the deployment of the CloudFormation stack. There are two parameters: *“checkVPC”* for defining the VPC CIDR block and *“checkVPCPUBSUB”* for defining the CIDR block of the public subnet. | None |
| cfVPC | This resource defines a Virtual Private Cloud (VPC) with the specified CIDR block. It enables DNS support and hostnames and specifies the default instance tenancy. | None |
| cfIGW | This resource creates an Internet Gateway (IGW) and Tags for the Internet Gateway. | None |
| cfIGWA | This resource attaches the Internet Gateway (cfIGW) to the VPC (cfVPC). This attachment enables traffic from the VPC to the internet and vice versa. | cfVPC, cfIGW |
| cfVPCPUB | Defines a public subnet within the VPC (cfVPC) with the specified CIDR block. It also specifies an availability zone and enables auto-assignment of public IP addresses to instances launched within the subnet. | cfVPC, checkVPCPUBSUB |
| cfVPCPUBROUTES | Creates a route table for the public subnet (cfVPCPUB). A route table contains a set of rules, called routes, that are used to determine where network traffic from instances within the subnet is directed. | cfVPCPUB |
| cfVPCPUBPUBLICRT | Adds a default route to the Internet Gateway (cfIGW) in the public route table (cfVPCPUBROUTES). This route enables outbound internet access for instances within the public subnet. | cfIGW, cfVPCPUBROUTES |
| cfVPCPUBROUTEA | Associates the public subnet (cfVPCPUB) with the public route table (cfVPCPUBROUTES). This association ensures that traffic from instances within the public subnet follows the routes defined in the route table. | cfVPCPUB, cfVPCPUBROUTES |
| Outputs: | These are output values of the CloudFormation stack that can be used by other stacks or resources. In this script, two outputs are defined: cfVPC which provides the ID of the VPC, and cfVPCPUB which provides the ID of the public subnet. | cfVPC, cfVPCPUB |

* CFormationTemp-1-Security

|  |  |  |
| --- | --- | --- |
| **Resource** | **Description** | **Dependencies** |
| AWSTemplateFormatVersion | This is the current version template of AWS CloudFormation | None |
| Parameters | Input parameters allow customization of the stack. Here, SSHLOC specifies the IP address range permitted for SSH access. | None |
| sgPING | This resource defines a Security Group used to allow ping traffic to EC2 instances within the specific VPC. | stackNetwork-cfVPC, cfVPC |
| sgWEB | Defines another security group (sgWEB) to allow SSH (port 22) and HTTP (port 80) traffic. It specifies the group name, and description, and allows inbound SSH and HTTP traffic from any source (0.0.0.0/0 for HTTP, and the value of SSHLOC for SSH). | stackNetwork-cfVPC |
| Outputs: | The output values of the stack, which can be used by other stacks or resources. Two outputs are defined: sgPING providing the security group ID for ICMP traffic, and sgWEB provides the security group ID for SSH and HTTP traffic. | sgPING, sgWEB |

* CFormationTemp-2-EC2

|  |  |  |
| --- | --- | --- |
| **Resource** | **Description** | **Dependencies** |
| AWSTemplateFormatVersion | This is the current version template of AWS CloudFormation | None |
| Parameters | Input parameters that customize the EC2 instance rollout. Here, instType allows the selection of instance type, and kpNAME specifies an existing EC2 Keypair for SSH access. | None |
| Mappings | Mapping of instance types to architecture and AMI IDs based on the AWS region. This allows the selection of the appropriate AMI for the specified instance type and region. | None |
| Ec2WEBSERVER | Defines an EC2 instance (ec2WEBSERVER) with specified properties including instance type, network configuration, security groups, key pair for SSH access, selected AMI based on region and instance type, tags, and user data script. The user data script installs and configures the Apache web server, starts the service, and creates a basic webpage. | stacSecurity-sgWEB, stacSecurity-sgPING, stacNetwork-cfVPCPUB, kpNAME |
| Outputs: | Output values of the stack, providing information about the created EC2 instances such as InstanceId, Availability Zone, Public DNSName, and Public IP address. | Ec2WEBSERVER |

2.2 Demonstrate knowledge of the selected IaC syntax by using the CLI (Command-line interface) or console to configure resources and confirm/orchestrate the deployment. Perform the following tasks:

1. Start service
2. Access to AWS Console and search AWS CloudFormation

A screenshot of a computer

Description automatically generated

Figure 1 Search CloudFormation Service

1. Configure template
2. Open Notepad++
3. Create a Security Group script using YAML and save as SecurityGroup.yml.

A screenshot of a computer

Description automatically generated

Figure 2 Security Group Script

1. Deploy resources
2. Go to the CloudFormation console: <https://console.aws.amazon.com/cloudformation/>
3. Click on "Create Stack" -> Choose "Upload a template file" and select the template file.

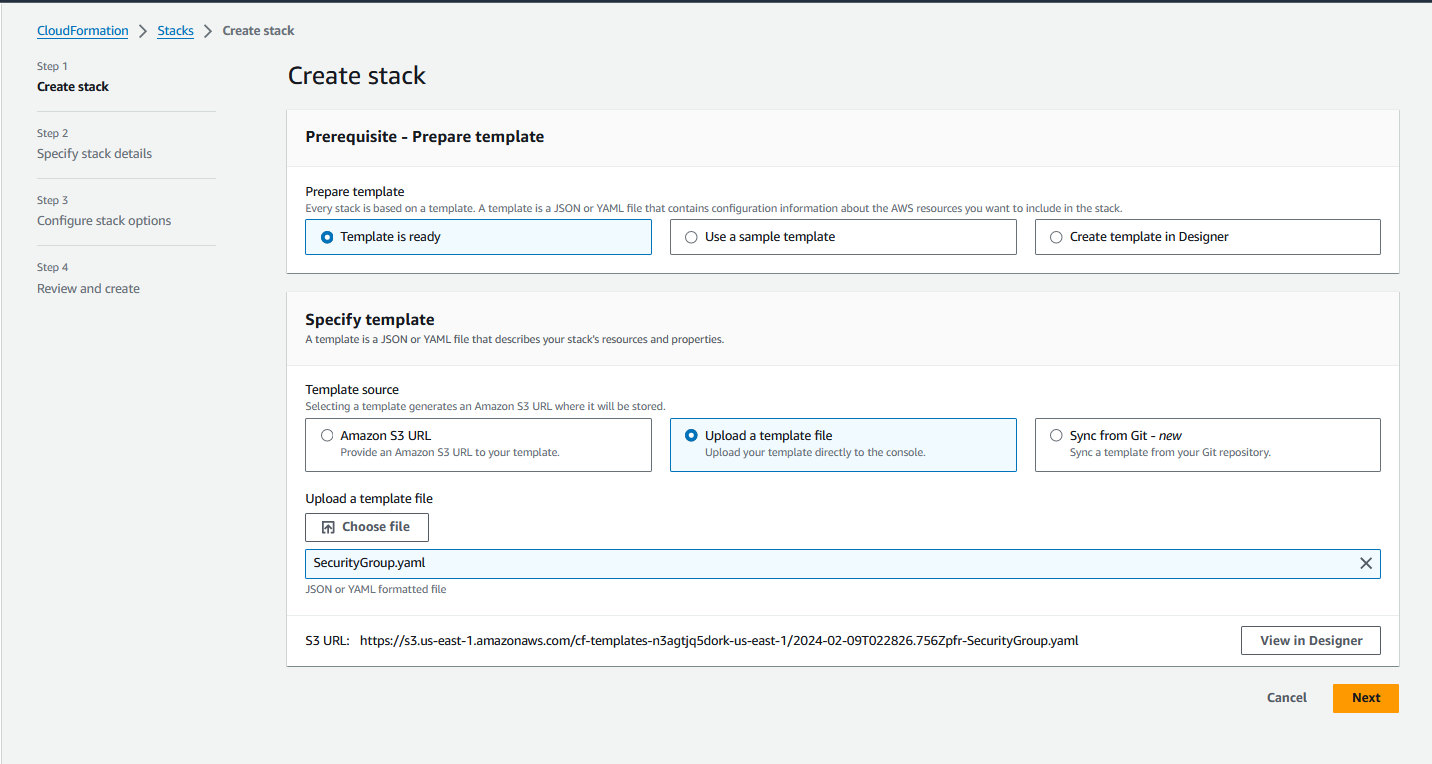


Figure 3 Create CloudFormation Stack and Upload the template file

1. Specify your stack name and parameters.

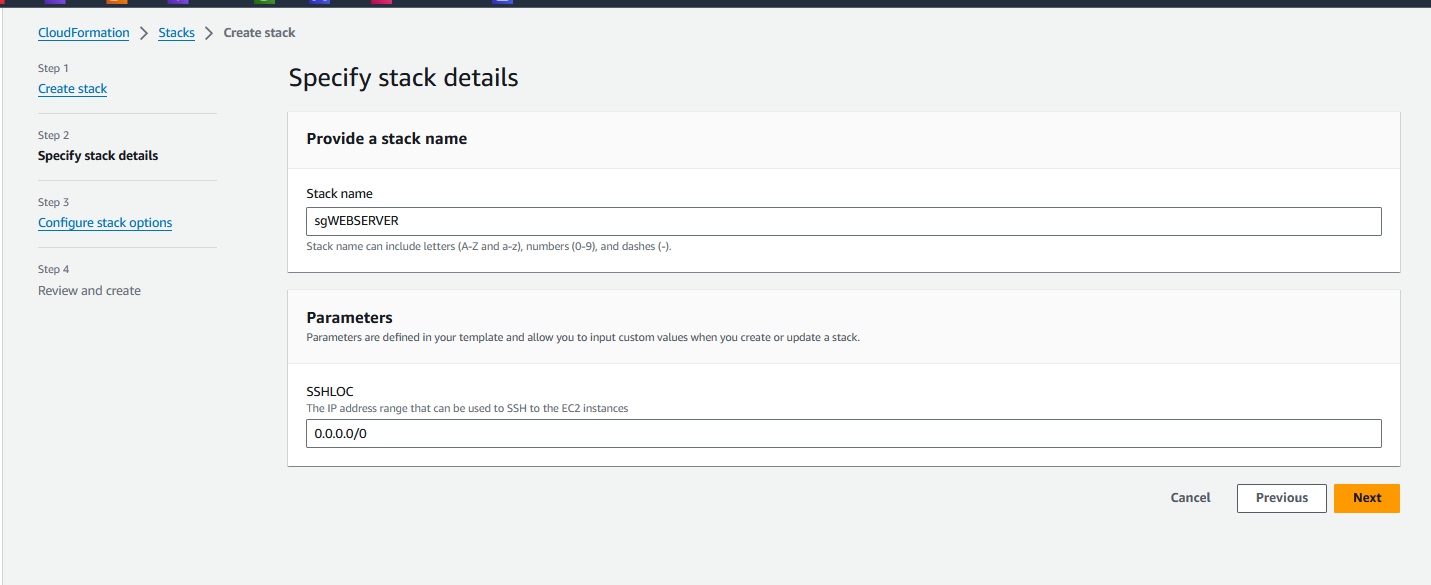


Figure 4 Name the stack

1. Configure stack options tags, permissions stack failure options

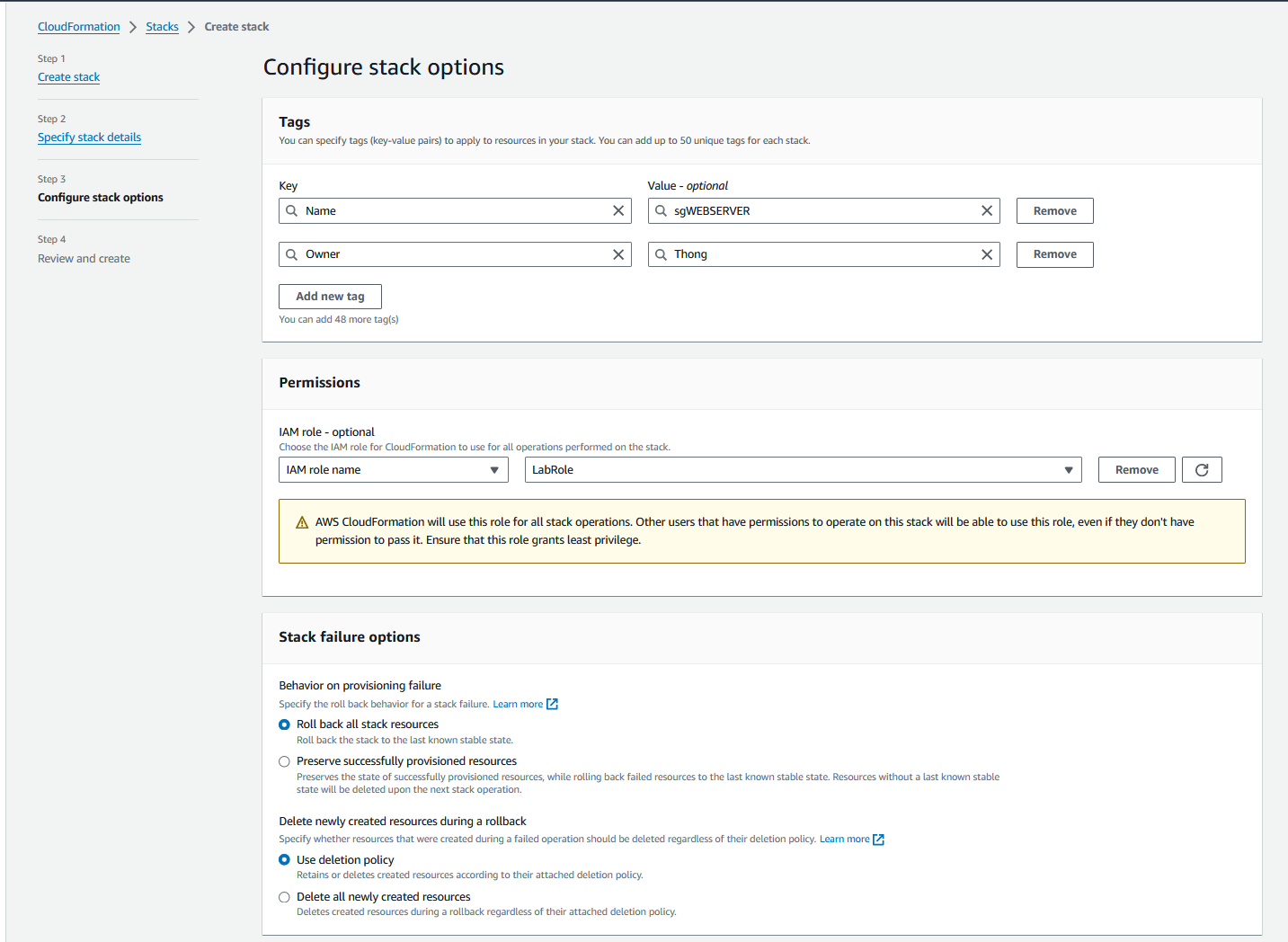


Figure 5 Tags and Permission for the stack

1. Review and Create -> Click Submit
2. Update a resource
3. Add HTTPS port for Security Group sgWEB and save it.

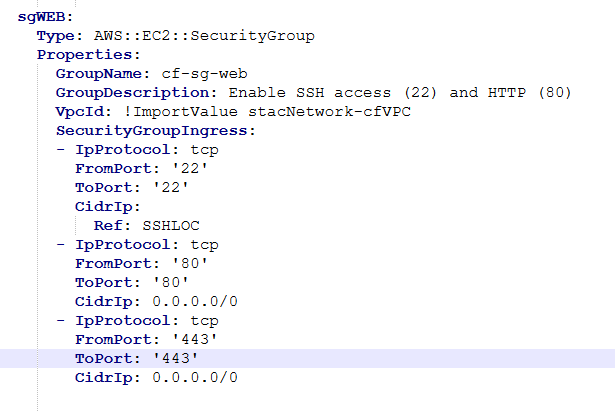


Figure 6 Add additional rule (HTTPS) for Security Group

1. Go to AWS CloudFormation console -> Click the stack want to update -> Click Update -> Replace current template -> Upload a template.

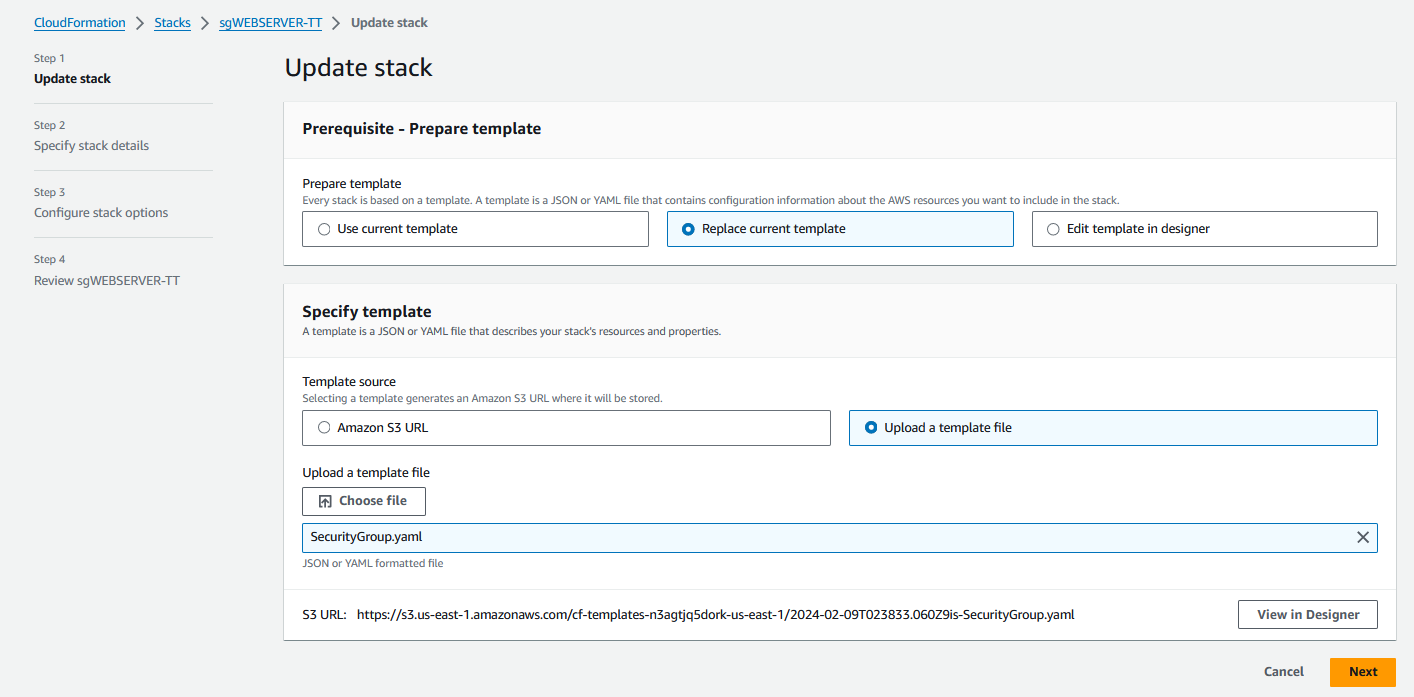


Figure 7 Update Security Group Stack

1. Next until submit and see the result.

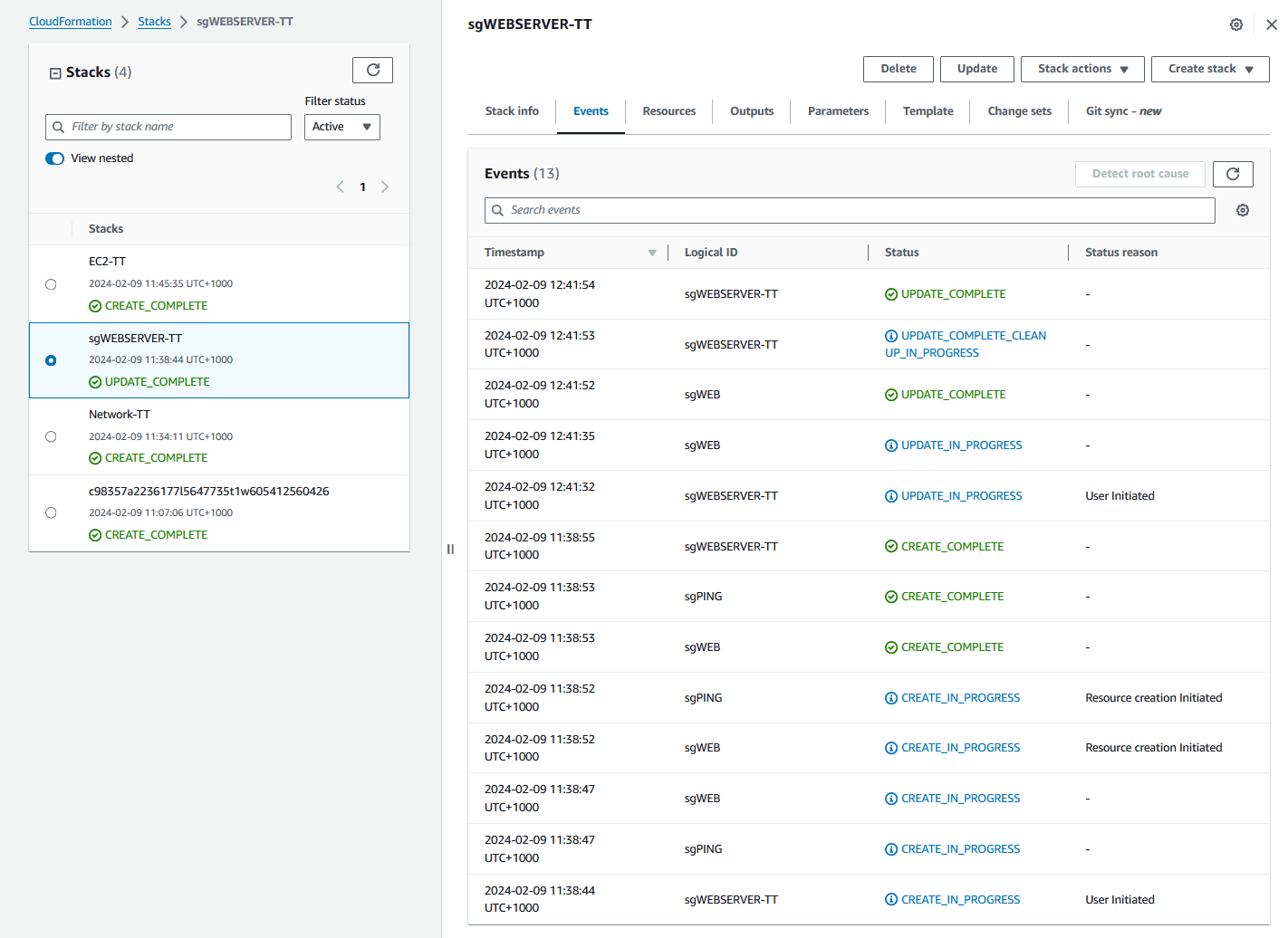


Figure 8 The Security Group Update Successful

A screenshot of a computer

Description automatically generated

Figure 9 The inbound rule HTTPS is apply

1. Delete a resource.
2. Go to the CloudFormation console and select the stack. Click on "Delete" button.

A screenshot of a computer

Description automatically generated

Figure 10 Delete Security Group (sgWEBSERVER-TT)

1. Confirm the deletion.

A screenshot of a computer

Description automatically generated

Figure 11 Confirm Delete the Stack

1. Stop service.

We can use AWS CLI for stop service. For example, stop service of EC2, we can use this CLI:

aws ec2 stop-instances \

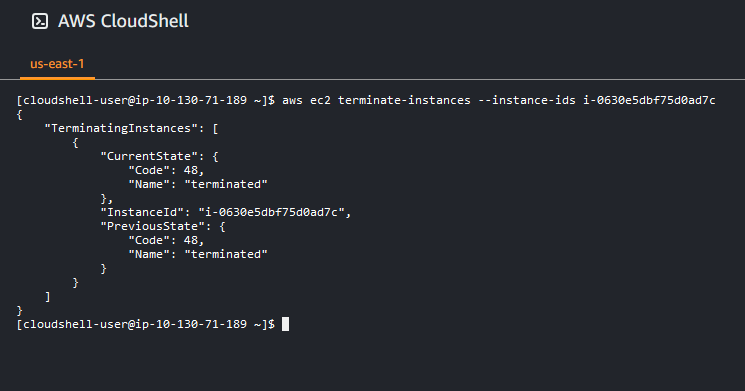
--instance-ids *i-093a9882473f4c19b*

*A screenshot of a computer program

Description automatically generated*

*Or we can use this CLI for terminate EC2 instance.*

aws ec2 terminate-instances --instance-ids *i-0630e5dbf75d0ad7c*



2.3 Test and troubleshot template errors. Document each error and the solution applied. Provide screenshots of the process.

* 1. Unrecognised resource types: [AWS::EC2::SecurtiyGroups]

A screenshot of a computer

Description automatically generated

Figure 12 Template format error: [AWS::EC2::SecurtiyGroups]

This error is unrecognised resource types which means the security group type is not valid let check the script of the type in resources specific in the error notify.

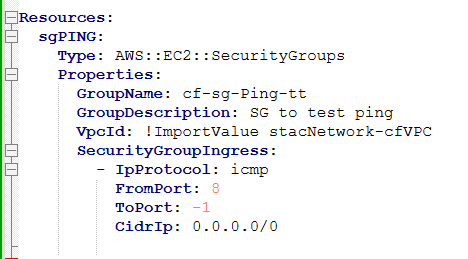


Figure 13 Check the error type of EC2 mentioned

I found the error, the Type: ***AWS::EC2::SecurityGroups*** it is incorrect, the correct is ***AWS::EC2::SecurityGroup***. Then fix and save the script.

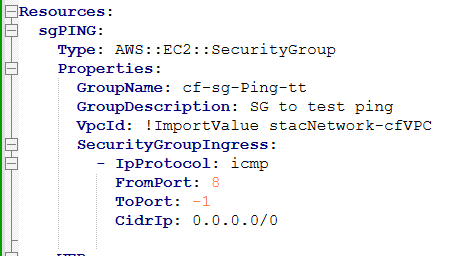


Figure 14 Correct the EC2 instance type.

* 1. Template format error: YAML not well-formed.

A screenshot of a computer

Description automatically generated

Figure 15 Template format error: YAML not well-formed.

This error is template format error. Let check with the script and the format error mentioned line 29, column 17.

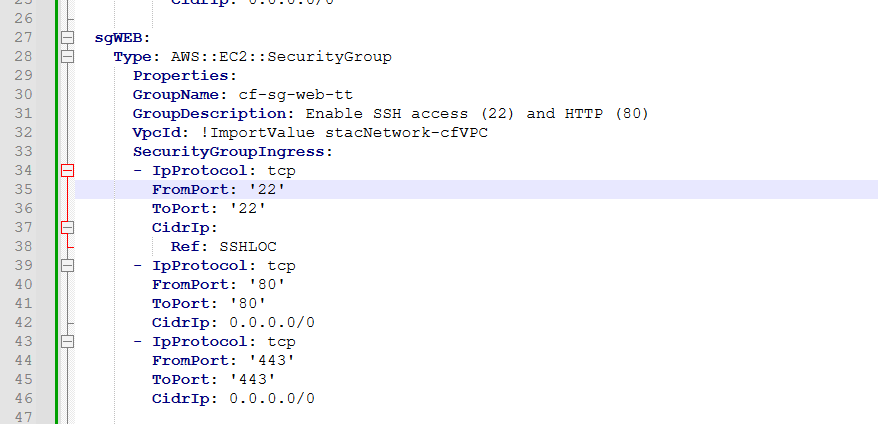


Figure 16 Checking the error in the line 29.

I found the error; the format is incorrect because the **Properties** must be column 5 same as **Type** line, just fix by make it same column as **Type**.

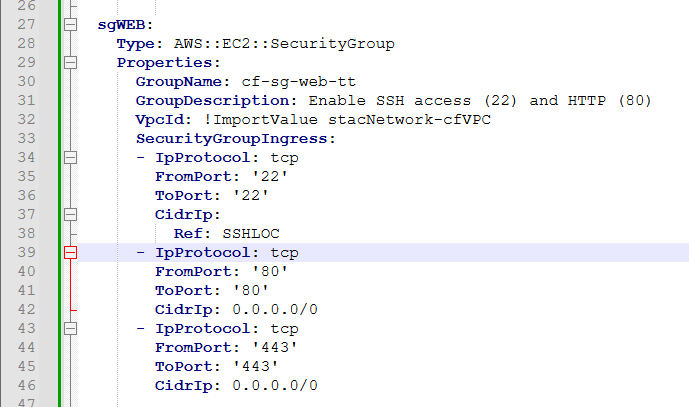


Figure 17 Correct the Properties format.

2.4 Create user documentation in the form of a README file that includes what the package is and how to run the containerised application.

***Attached (README\_Part2.md)***

2.5 Organise a meeting and present the completed infrastructure as code (IaC) solution to your manager or relevant person in the organisation for approval and signoff.

|  |  |
| --- | --- |
| **Cloud Infrastructure as code (IaC) – PRE-DEFINED Template SIGNOFF**  Signing off on this document signifies that the cloud infrastructure as code presented complies with the Client’s Business requirements. | |
| Project Manager or relevant stakeholder  Signature: Frans de Oude  Date: 15/3/2024 | Web/Cloud Developer  Signature: Thong Thao  Date: 18/3/2024 |
| **Documentation APPROVED**  Please provide feedback on the changes needed. | |
| **APPROVAL**  Granted  Not Granted | |

# PART 3 - Develop and update infrastructure as code (IaC) templates - Create own template

3.1 Demonstrate knowledge of the selected IaC syntax by creating your own template to deploy a multi-tier application to a cloud orchestration environment for the scenario presented. Ensure that the template pod can manage and run multiple containers and provisions the resources required by the scenario.

I am targeting AWS as my cloud provider; I will focus on using CloudFormation as my Infrastructure as Code (IaC) solution. I will aim to define the templates that provision VPC, Security Group, Database RDS and EC2 instance named below using YAML language programming:

* Deploy the template cf\_01\_VPC\_TT

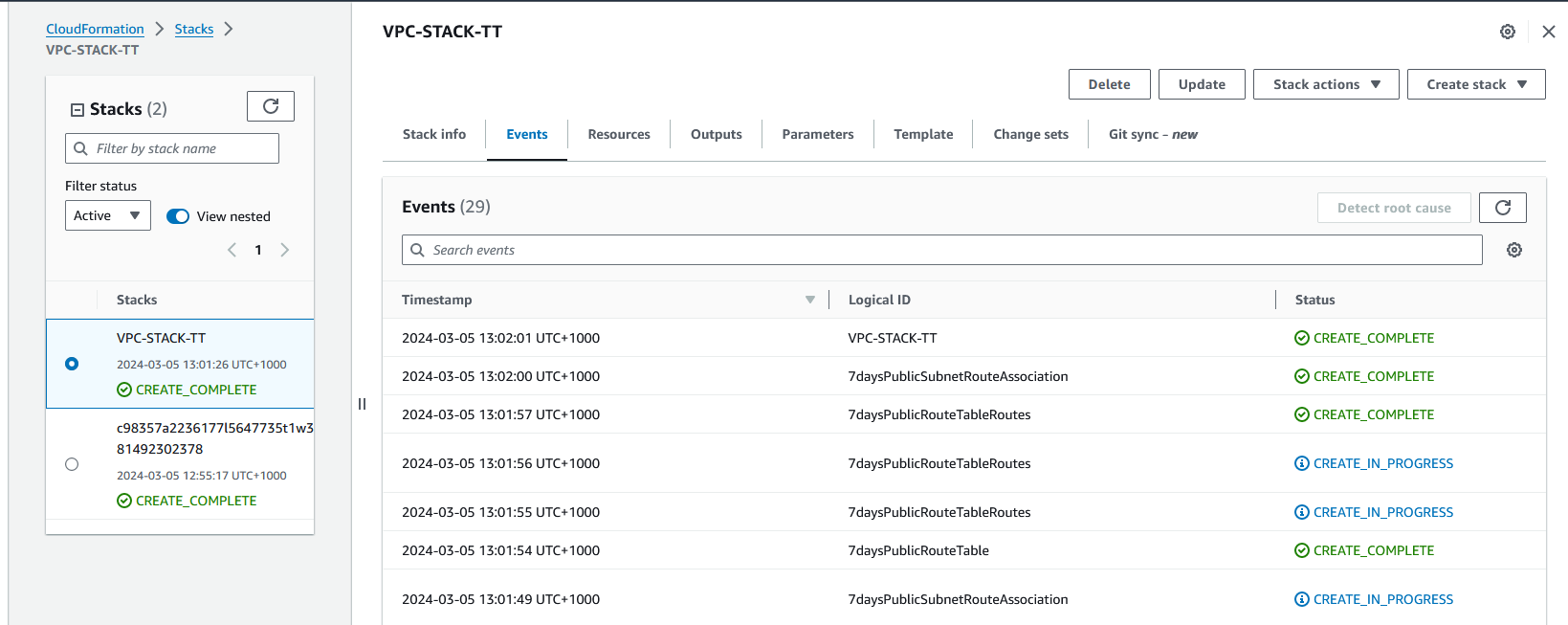


Figure 18 Deploy VPC Template

A screenshot of a computer

Description automatically generated

Figure 19 Completed VPC Creation

* Deploy the template cf\_02\_SG\_TT

A screenshot of a computer

Description automatically generated

Figure 20 Deploy Security Group Template

A screenshot of a computer

Description automatically generated

Figure 21 Completed the Security Group Creation

* Deploy the template cf\_03\_RDS\_TT

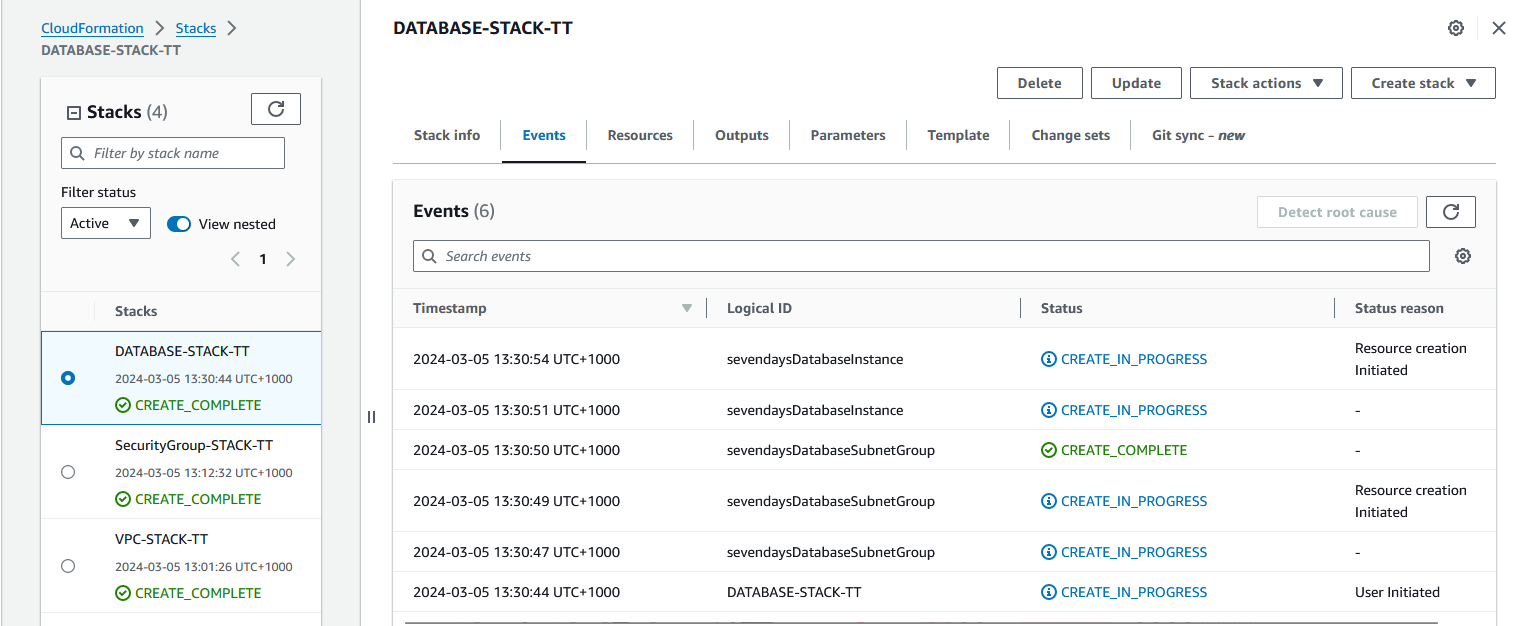


Figure 22 Deploy Database Template

A screenshot of a computer

Description automatically generated

Figure 23 Completed Database Creation

* Deploy the template cf\_04\_EC2\_TT

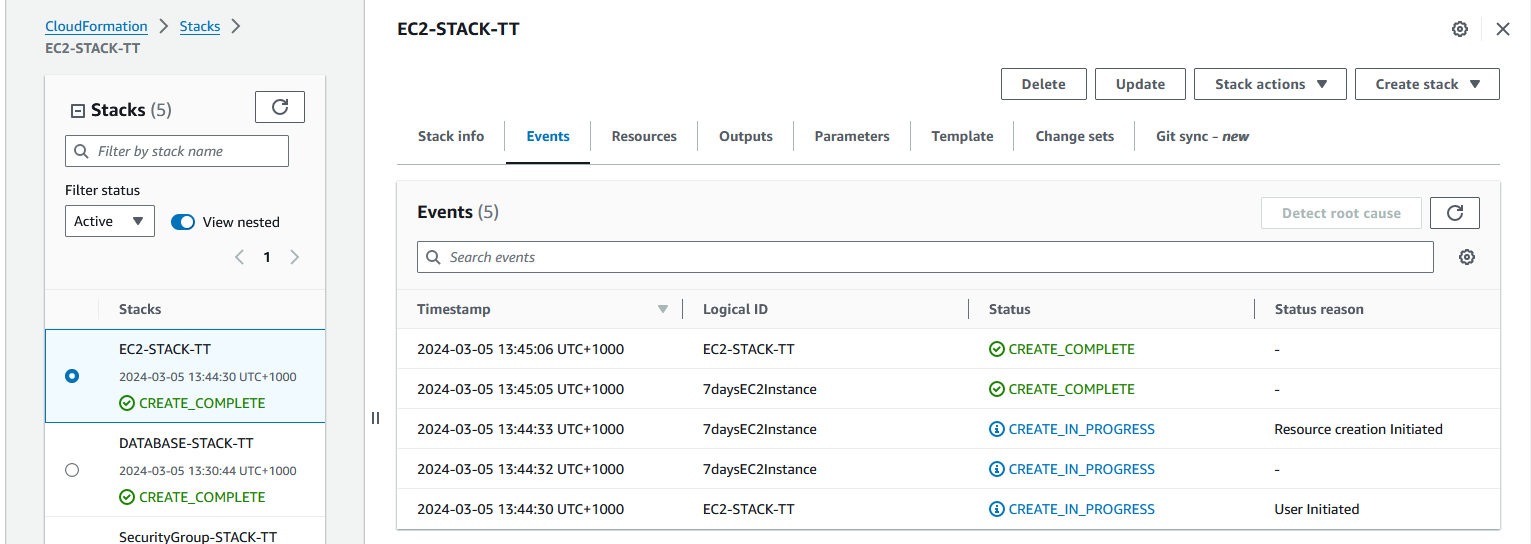


Figure 24 Deploy EC2 Template

A screenshot of a computer

Description automatically generated

Figure 25 Completed EC2 Creation

3.2 Configure the template to accept parameters at runtime. Deploy the template. Provide screenshots.

* The Parameters of cf\_01\_VPC\_TTA screenshot of a computer program

  Description automatically generated

Figure 26 The Parameters of VPC Script

A screenshot of a computer

Description automatically generated

Figure 27 Display Parameters of VPC

* The Parameters of cf\_01\_RD\_TT

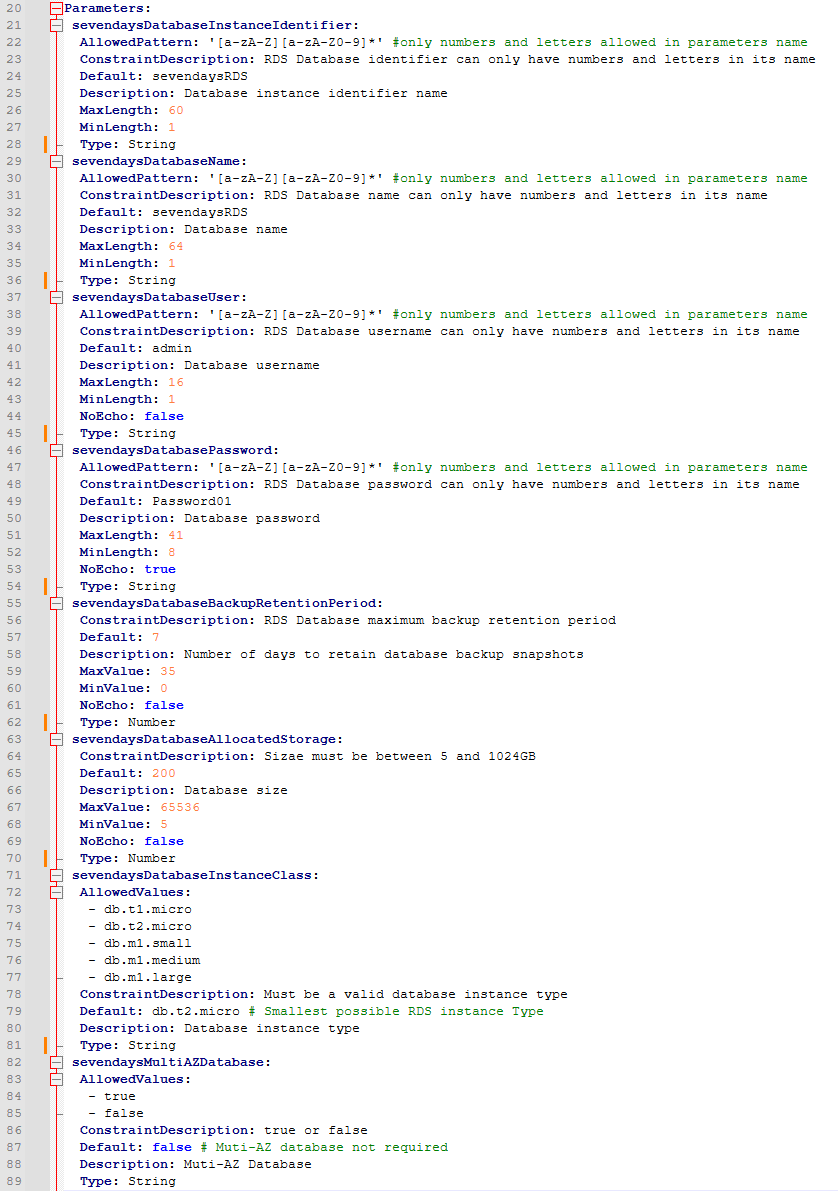


Figure 28 The Parameters of RDS Script

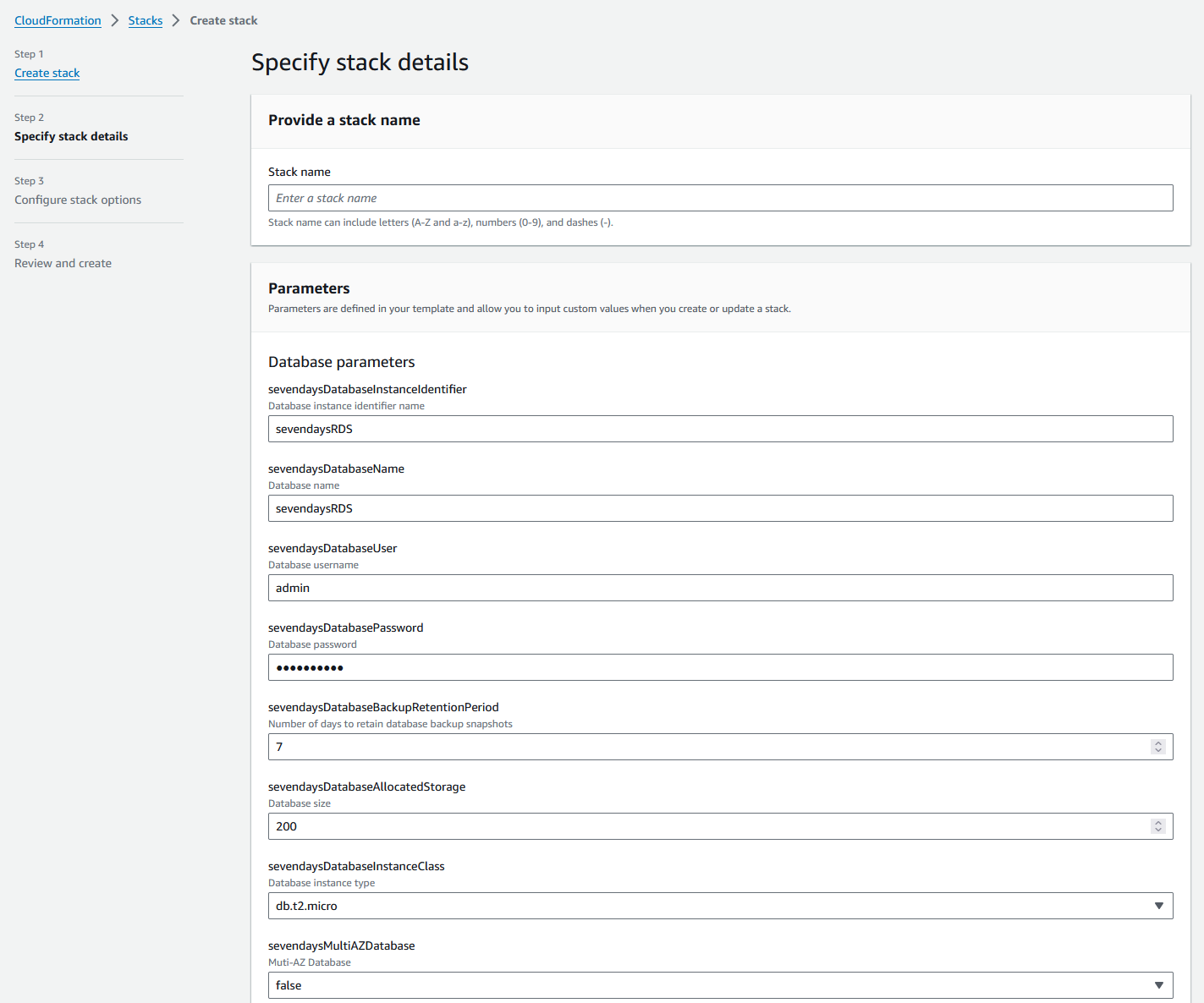


Figure 29 Display Parameters of RDS Stack

3.3 Update the template to:

a) Modify a previously deployed resource

I want to modify the resource VPC Name: 7daysVPC to sevendaysVPC.

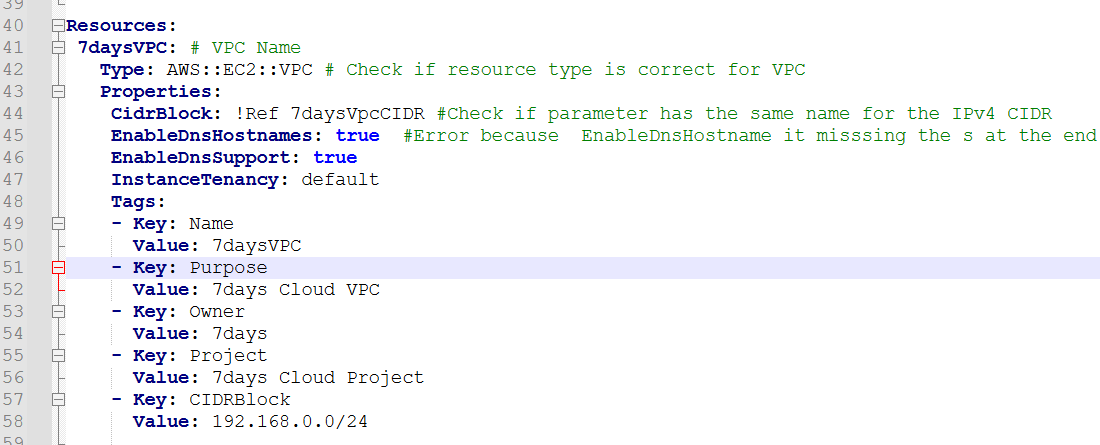


Figure 30 VPC name 7daysVPC.

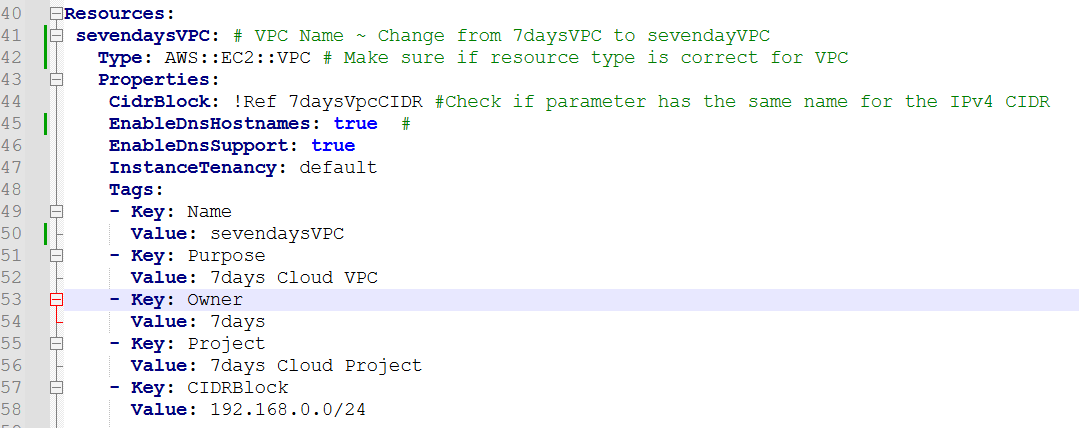


Figure 31 Modify the VPC name from 7daysVPC to sevendaysVPC

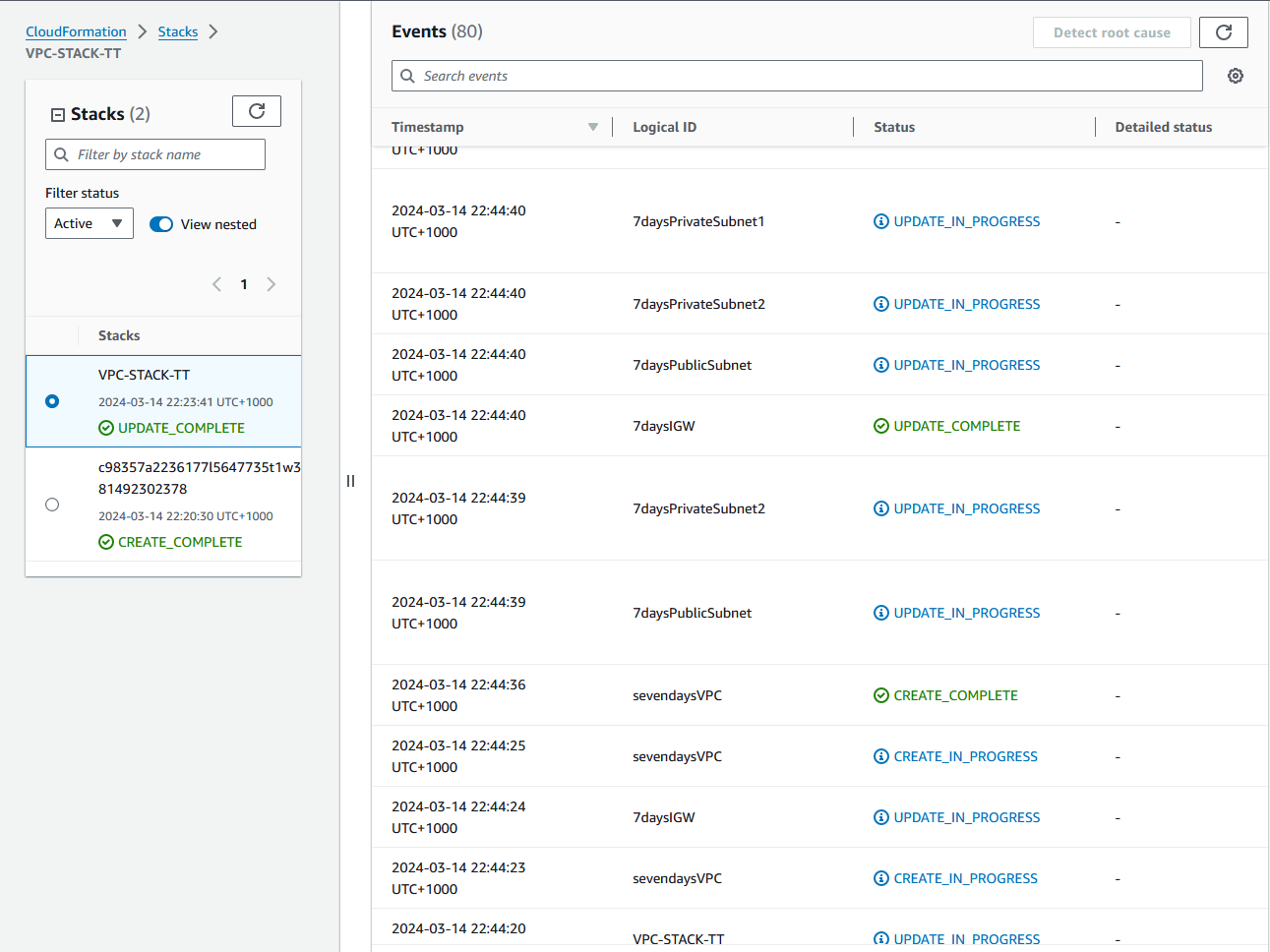


Figure 32 Update VPC Template Completed

A screenshot of a computer

Description automatically generated

Figure 33 After Modify and Update the Stack, VPC name changed to sevendaysVPC.

b) Add a new resource

I want to add a new resource for creating ***7daysPrivateSubnet3*** before adding to resource add new parameter for Private Subnet3 first.

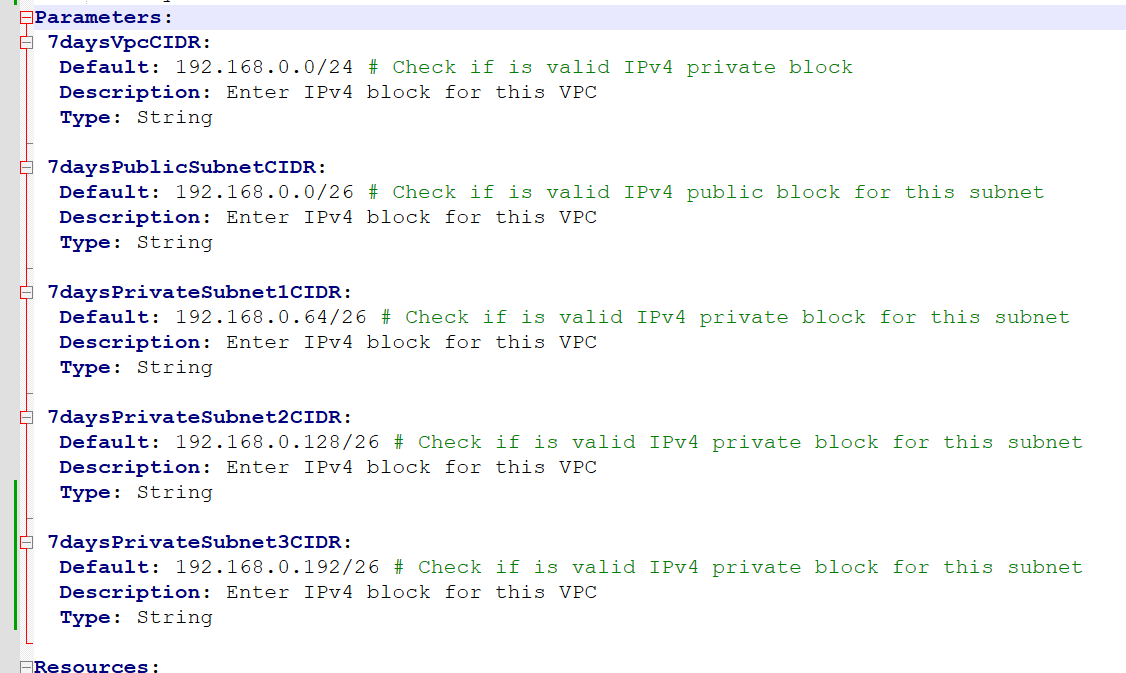


Figure 34 Add Parameter for Private Subnet 3 for VPC

A screenshot of a computer

Description automatically generated

Figure 35 Add New Resource PrivateSubnet3 for VPC under Resource

A screenshot of a computer program

Description automatically generated

Figure 36 In Outputs add the new resource for update.

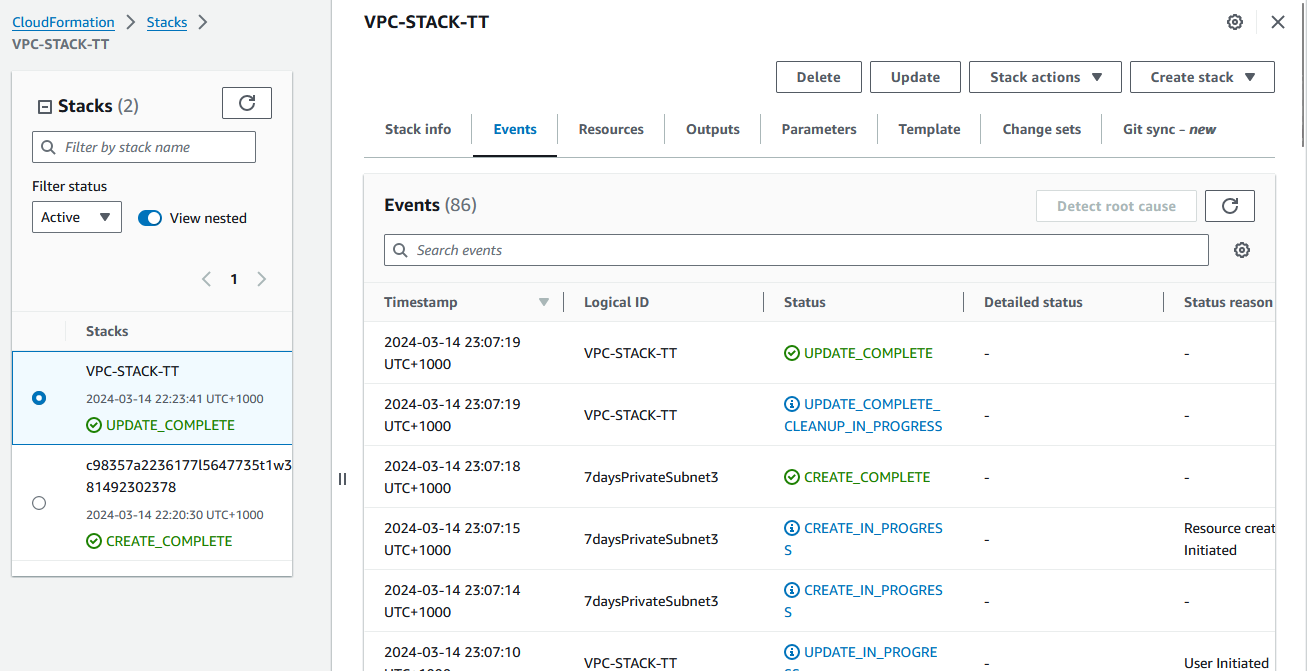


Figure 37 The Private Subnet 3 Create Completed

A screenshot of a computer

Description automatically generated

Figure 38 Check Subnets After Update the Stack

c) Remove a deployed resource

I want to delete Private Subnet 3 in VPC script that was deployed previously.

* Remove the resource 7daysPrivateSubnet3 script.

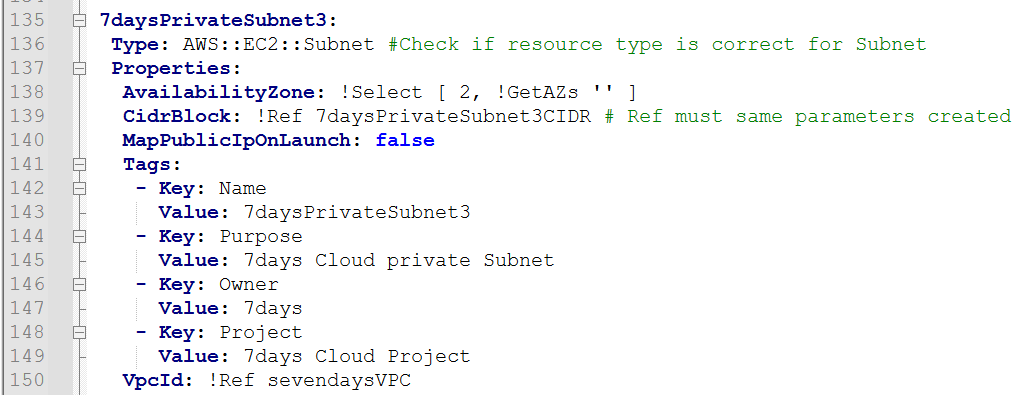


Figure 39 Delete 7daysPrivateSubnet3 in VPC Script.

After delete the resource and save the script, then update the script by CloudFormation

* Update the script, Go to CloudFormation Console -> Choose VPC-STACK-TT was deployed and click Update button.

A screenshot of a computer

Description automatically generated

Figure 40 Update the VPC-STACK -TT

* Choose Replace current template -> Upload a template file and upload the VPC script removed private subnet3.

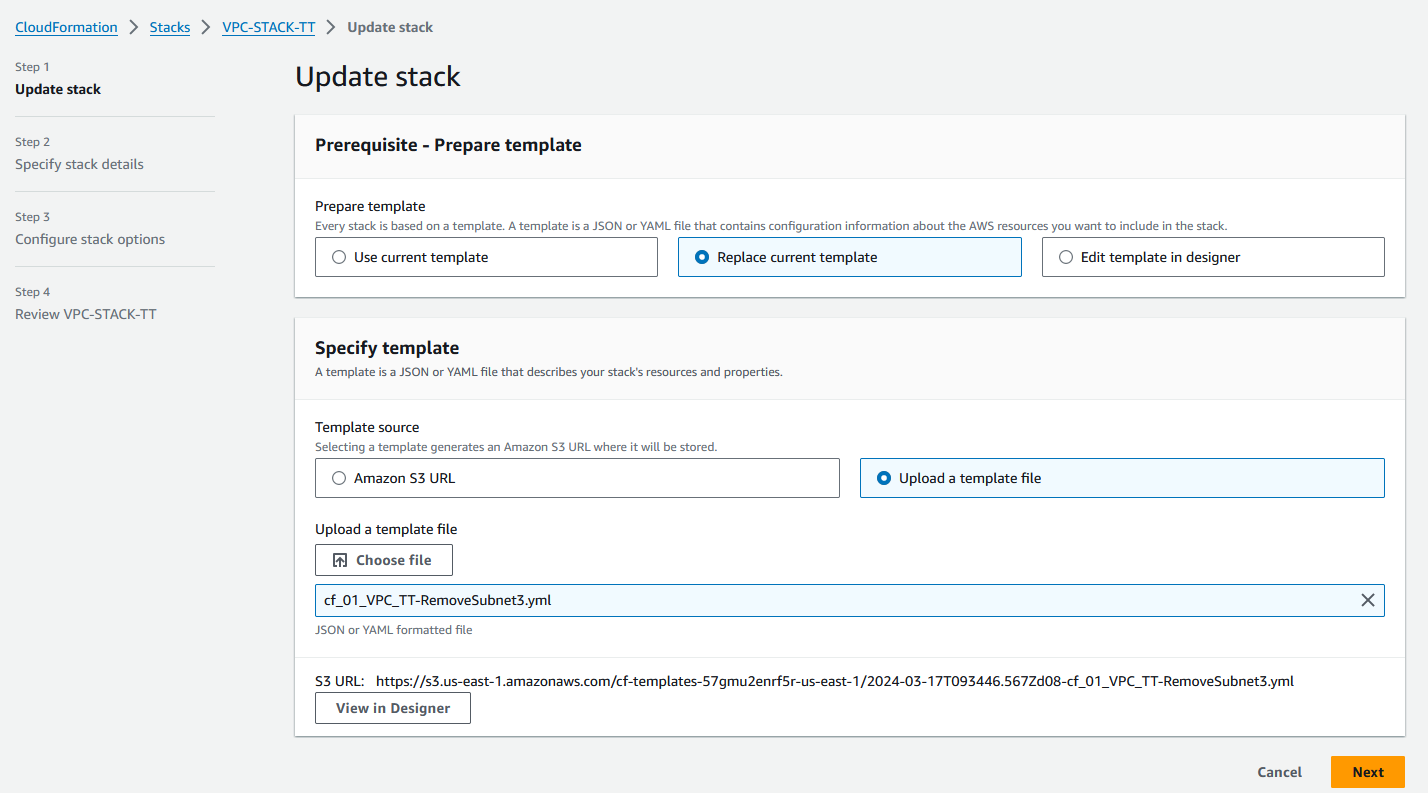


Figure 41 Upload the VPC script that removed 7daysPrivateSubnet3

A screenshot of a computer

Description automatically generated

Figure 42 Display VPC Parameters, 7daysPrivateSubnet3 is removed.

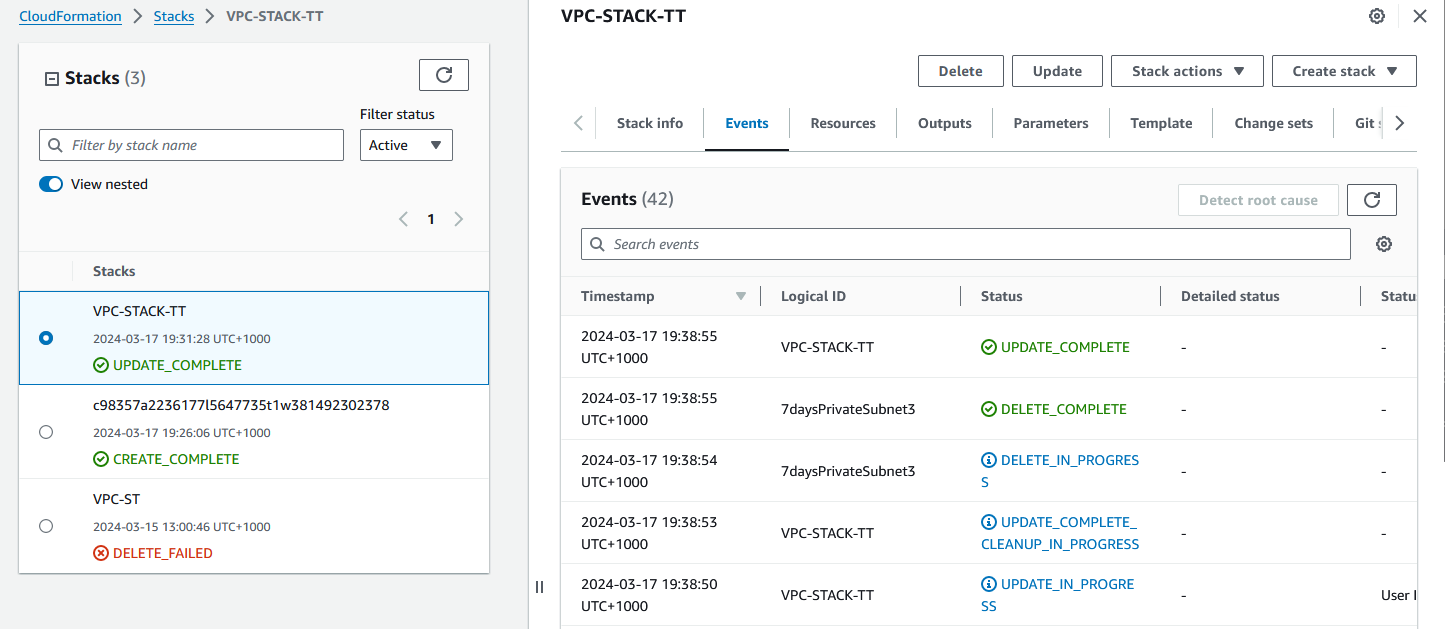


Figure 43 Update Completed and 7daysPrivateSubnet3 is deleted.

A screenshot of a computer

Description automatically generated

Figure 44 Checking the Subnets, 7daysPrivateSubnet3 is deleted.

3.4 Use CLI (command line interface) or console to confirm the deployment. Provide screenshots to evidence the actions performed.

A screenshot of a computer program

Description automatically generated

Figure 45 Checking the Security Group Template Deployment

3.5 Using the CLI or console, modify the template to add or change parameters (e.g., API version, spec).

I want to change *7daysPrivateSubnet2CIDR IPv4 block* to *192.168.0.192/26*

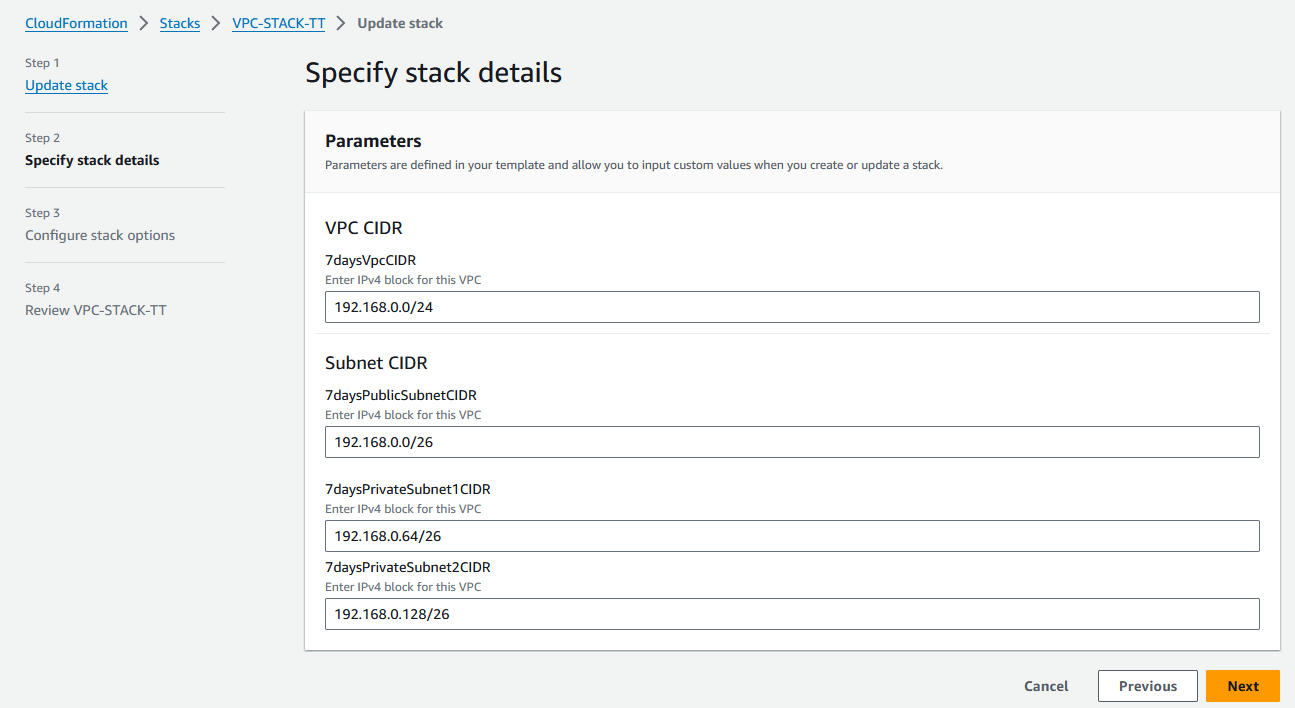


Figure 46 Update the VPC-STACK-TT Parameters of CIDR



Figure 47 Update the Parameter Successful

A screenshot of a computer

Description automatically generated

Figure 48 The Subnet has been changed.

3.6 Create scenario-appropriate secrets using the CLI or configuration file and ensure that it is accessible to the pod created in the template (3.1, 3.2). Provide screenshots of the process.

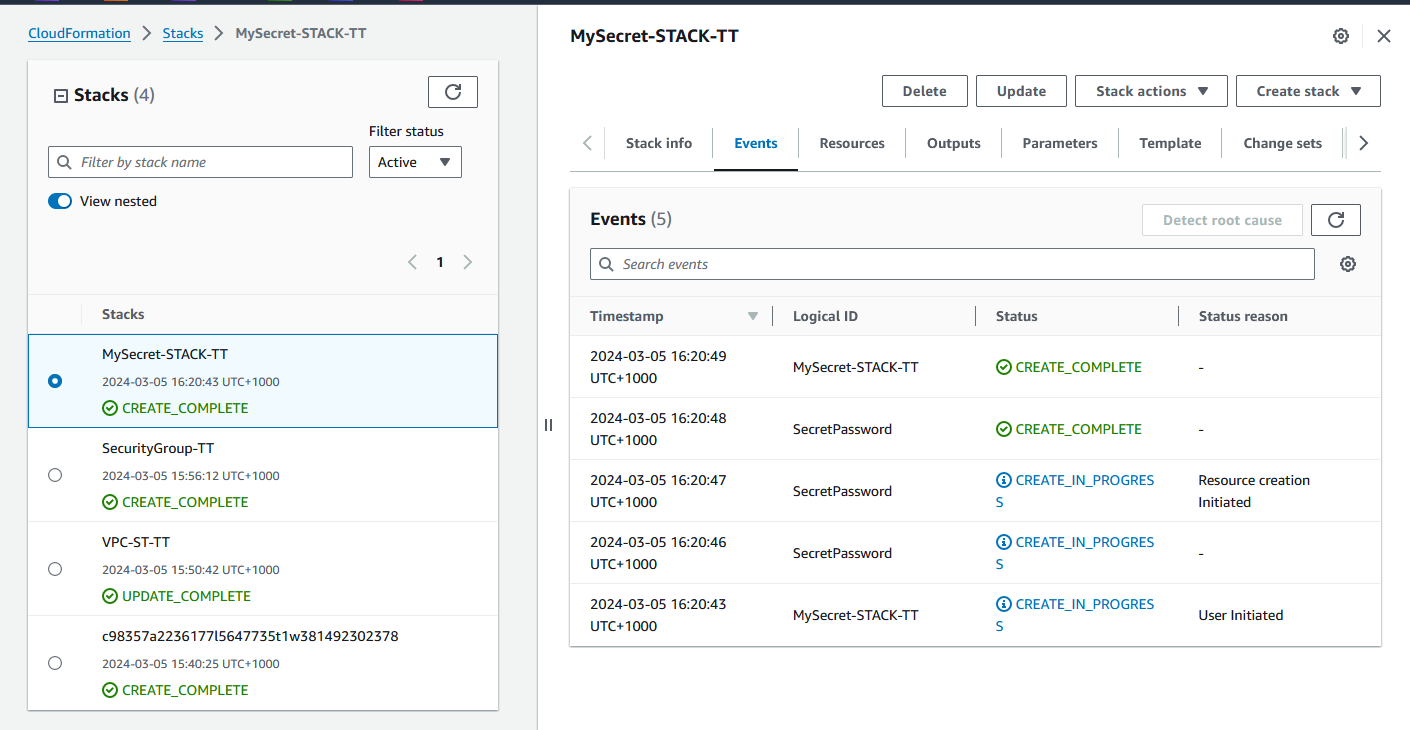


Figure 49 Deploy Secret Template

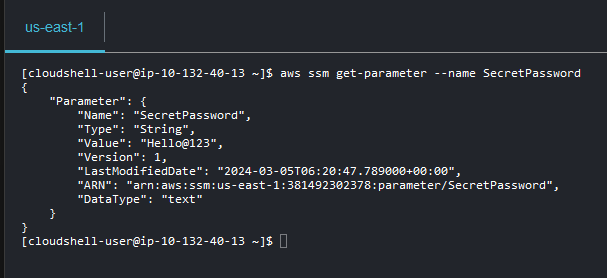


Figure 50 AWS CLI - Check Parameter of SecretPassword

3.7 Delete templates as required. Provide screenshot.

I want to delete SecurityGroup-TT template, we can access to CloudForamtion console for delete or using AWS CLI to delete, I also provide *CLI for deleting in Figure 30*.

A screen shot of a computer

Description automatically generated

Figure 51 Deleting VPC-ST-TT Stack



Figure 52 Completed Deletion

* 1. Test and troubleshot template errors. Document each error and the solution applied. Provide screenshots of the process.
* Deploy Security Group Template and error *VPC ID does not exist*.

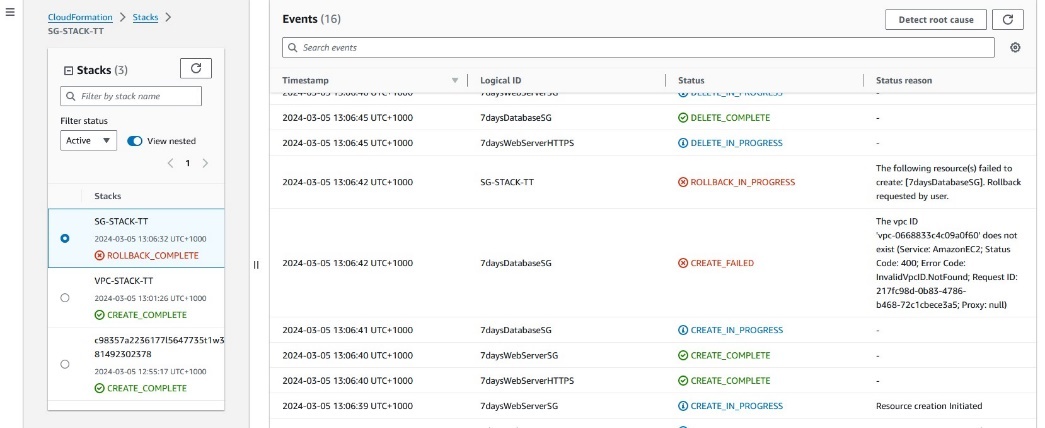
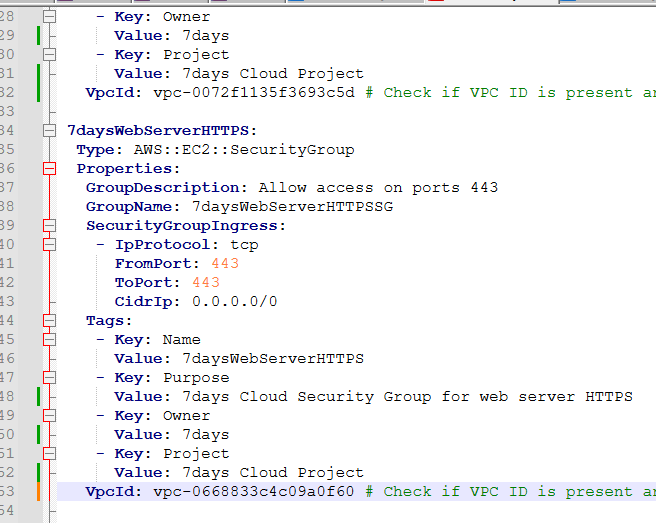


Figure 53 Create Failed the vpc ID does not exist.

* Checking the vpc ID

I found that the VPC ID does not exist because it is the wrong ID. Change to vpc-0072f1135f3693c5d same as current VPC ID.



After changing the VPC ID and deploying the Template, it is completed.

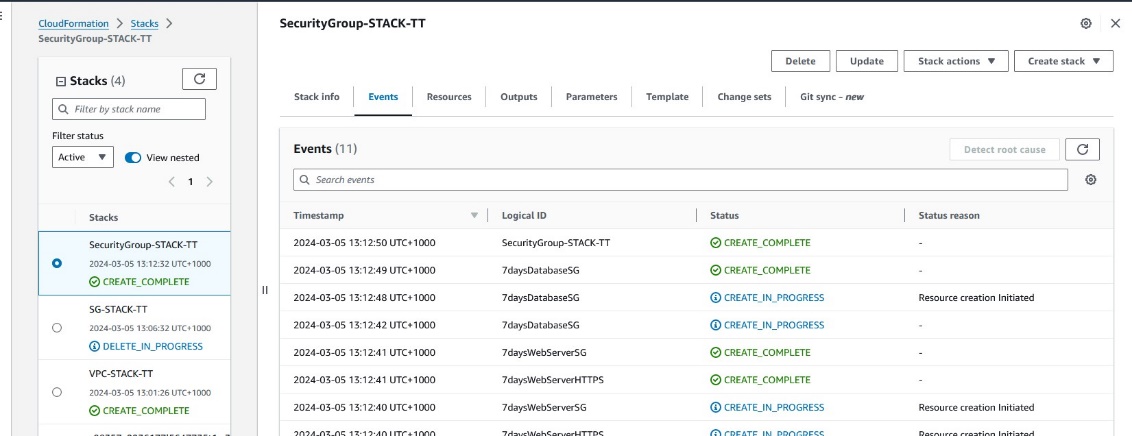


Figure 54 Completed Deployment the Security Group Stack

* Deploy RDS Template.

A screenshot of a computer

Description automatically generated

Figure 55 The Database Stack failed, and rollback failed.

I found that the LabRole user doesn’t have permission to create RDS. Therefore, go to IAM Console to add RDS permission to LabRole User.

A screenshot of a computer

Description automatically generated

Figure 56 Choose LabRole User

A screenshot of a computer

Description automatically generated

Figure 57 Attach policies to LabRole User

A screenshot of a computer

Description automatically generated

Figure 58 Add Policy AmazonRDSFullAccess to LabRole User

After Attach Policy to LabRole User the RDS Template created complete.

A screenshot of a computer

Description automatically generated

Figure 59 Database Stack Completed Deployment

3.9 Create user documentation in the form of a README.md file that includes what the package is and how to run the containerised application.

*Attached File README\_Part3.md*

3.10 Organise a meeting and present the completed infrastructure as code (IaC) solution to your manager or relevant person in the organisation for approval and signoff.

|  |  |
| --- | --- |
| **Cloud Infrastructure as code (IaC) - Template Development SIGNOFF**  Signing off on this document signifies that the cloud infrastructure as code presented complies with the Client’s Business requirements. | |
| Project Manager or relevant stakeholder  Signature: Frans de Oude  Date: 15/3/2024 | Web/Cloud Developer  Signature: Thong Thao  Date: 18/3/2024 |
| **Documentation APPROVED**  Please provide feedback on the changes needed. | |
| **APPROVAL**  Granted  Not Granted | |

# PART 4 - Contingency task and cloud-related knowledge concepts

4.1 **Contingency task:** What happens if the cloud service drops?

1. Failover to a Secondary Cloud Provider

* Automate failover processes: If possible, establish mechanisms to automatically route workloads to the secondary provider if the primary service fails.
* Data Synchronization: Maintain updated data replicas across providers to prevent loss during failures.

1. Disaster Recovery to an On-Premises Backup

* Critical data replication: Keep local copies of the most essential data for quick recovery, should the cloud service become completely unavailable.
* Offline functionality: Where feasible, design applications or key modules to have basic offline functionality allowing limited operations to continue even without cloud connectivity.
* Temporary on-premises infrastructure: For extended outages, identify if you can rapidly utilize internal infrastructure or obtain temporary setups to host scaled-down versions of critical systems.

1. Graceful Degradation of Functionality

* Clear communication: Designate communication with internal teams, clients, and stakeholders. Use templates for updates to convey status clearly and rapidly.
* Incident Response Plan: Align any contingency responses with your broader Incident Response Plan. Define the roles and responsibilities of the team members involved in handling cloud outages.
* Stakeholder Notification:

4.2 Select three (3) from the following list of infrastructure as code tools or technologies used in cloud platforms and compare them in terms of features, compatibility with cloud platforms and ease of learning and using the tool.

* AWS CloudFormation
* Azure Resource Manager
* Pulumi
* Terrafom
* Ansible
* Chef
* Puppet
* Docker
* Podman

|  |  |  |  |
| --- | --- | --- | --- |
| TECHNOLOGY/TOOL | FEATURES | COMPATIBILITY | EASE of LEARN/USE |
| AWS CloudFormation | * Native to AWS, deep integration * YAML/JSON format | * Support only AWS. * Limited use outside of the AWS ecosystem | * Easier for those familiar with AWS * Less flexible for muti-cloud scenarios |
| Terraform | * Declarative language * State management for tracking changes. * Plan and preview changes before executing | * Muti-cloud broad provider * Open source with a large community | * Medium requires HCL knowledge. * Can be complex for intricate scenarios |
| Pulumi | * Uses general-purpose programming languages like Python, JavaScript… * Imperative model * Strong type systems and testing frameworks | * Muti-cloud support * More options for configuration logic | * Highest flexibility * Steep learning curve for those unfamiliar with the chosen language |

4.3 Research international technology standards applied to cloud computing and select three (3) current ISO standards.

a) For each standard, identify its full name, and publication year and provide a summary of the purpose and scope of the standard.

|  |  |  |
| --- | --- | --- |
| STANDARD | YEAR | PURPOSE and SCOPE |
| ISO/IEC 27017 | 2015 | Provides security guidelines specifically for cloud service providers and customers. It builds upon ISO/IEC 27002 with cloud-focused controls for securing sensitive information in the cloud. |
| ISO/IEC 27018 | 2019 | Focuses on protecting personally identifiable information (PII) in public clouds. Offers guidelines for implementing controls based on privacy principles and risks concerning PII. |
| ISO/IEC 27001 | 2022 | A cornerstone standard that specifies requirements for establishing, implementing, maintaining, and improving an Information Security Management System (ISMS) within an organisation's context. |

b) Would compliance with these standards – or one of the three identified - be applied to the scenario completed in this portfolio? Justify your answer.

As moving towards real-world deployment, demonstrating adherence to the core concepts behind these standards can boost confidence for stakeholders and clients in the system's robustness and its protection of sensitive data. It establishes structured processes rather than ad-hoc security measures.

* ISO/IEC 27017: Relevant to services running in a cloud environment, considerations around the security of your application code, data storage, access controls, and communication between service layers (your microservices) would directly align with the guidelines proposed by ISO/IEC 27017.
* ISO/IEC 27018: Potentially highly relevant if your application in the "Gorgeous Cupcakes" scenario handles any customer data (names, addresses, payment information, etc.). This standard would establish best practices for safeguarding personally identifiable information within the cloud infrastructure.
* ISO/IEC 27001: While this standard provides a broader ISMS framework rather than a cloud-specific focus, applying its principles around secure development, system configuration, risk management, and incident response would significantly improve the overall security posture of your project.

4.4 Outline and explain four (4) industry standards or best practices used for creating, managing, and deploying infrastructure as code (IaC).

1. Version Control

Treat your IaC definitions with the same rigor as application code. Use a version control system like Git for collaboration, change tracking, and the ability to rollback.

Version control serves as a historical reference, facilitating troubleshooting while safeguarding against accidental changes, and promoting smoother collaboration among teams.

1. Modularity and Reusability

For better management and reusability, break down your IaC into modular templates or parameterized components.

Modularity enables efficient reuse of infrastructure patterns, improving maintainability and reducing code.

1. Testing and Validation

Before you put your changes into production, it's important to implement a thorough testing process for your IaC. This process can include the following steps:

* Linting: This helps you to identify syntax errors or style violations in your configuration files.
* Unit Testing: This allows you to verify that individual modules or components are working as intended.
* Integration Testing: This ensures that multiple IaC components, and the cloud resources they provision, are interacting correctly.

Thorough testing significantly reduces the risk of errors causing configuration drift or outages when IaC changes are deployed in live environments.

1. Security and Secrets Management

It is not recommended to include sensitive credentials such as API keys, passwords, or database connection strings directly in your IaC files. Instead, it is recommended to use secure secrets management systems like AWS Secrets Manager and HashiCorp Vault.

Proper secrets management is crucial to prevent exposing sensitive information in version control or other locations, as hardcoding secrets create massive security vulnerabilities.

4.5 Research and explain at least three (3) metrics currently used in the industry to measure the leverage of using templates to manage and deploy to cloud platforms. The metrics can include techniques, methods, or standards.

For each metric provide detailed information and examples.

1. Deployment Time Reduction

*What it Measures:* Creating a load balancer, VM with a web server, and basic networking can take up to 30 minutes with potential errors due to manual steps.

*Example:*

*Before Templates:* Manually creating a load balancer, VM with a web server, and basic networking could take 30 minutes with the potential for errors due to manual steps.

*With Templates:* A predefined template automates the provisioning process in under 5 minutes, ensuring consistency and eliminating manual errors.

*Benefit:* Faster deployments lead to time savings, quick delivery of new features, and faster scaling to meet user demand.

1. Configuration Consistency and Drift Reduction

*What it Measures:* Tracks how often infrastructure deviates from its desired state as defined in a template. This drift can arise from manual changes, accidental misconfigurations, or external forces.

*Example*: A template mandates that security group rules must remain strict to allow only required traffic. Using template-based checks prevents unintended modifications to these rules over time.

Benefits:

Reduced exposure to vulnerabilities from ad-hoc changes leads to greater security and fewer unexpected misconfigurations which contribute to system stability.

1. Cost Optimisation

What it Measures: Savings associated with template use. This could involve:

*Right-sizing:* Templates can accurately match instance size to load, avoiding over-provisioning.

*Scheduling Resources:* Automation within templates can be set up to turn resources on/off at specific times. For example, shutting down development environments outside work hours.

*Enforcement of Tagging*: Consistent tags in templates enable accurate cost analysis and prevent "orphaned" resources from running needlessly.

*Example*: A well-structured template can use auto-scaling groups to ensure sufficient resources during peak traffic and scale down instances during low-traffic periods to save on compute costs.

Benefit: Maximizes the efficiency of your cloud spending and enables predictable budgeting.

4.6 Research industry-standard hardware and software products utilised in cloud development and deployment. Select two (2) hardware and two (2) software products and summarise your research in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| PRODUCT | FEATURES/CAPABILITIES | APPLICATION  Suitable for | DATA STORAGE |
| Bare Metal Server (e.g., Dell PowerEdge) | * High-performance computing powerful CPUs (e.g., ADM EPHY, Intel Xeon) * Large amounts of RAM for caching and in-memory processing | * Hosting application * Processing, databases, and web hosting * HPC workloads. | Data storage typically involves utilising attached storage solutions like RAID arrays or Network Attached Storage (NAS) systems. |
| Network Switches | * High-performance switches 10Gbs/s or faster that form the backbone of cloud data centre network. * Support features like QoS, VLAN, and traffic aggregation | * Support heavy internal network traffic between cloud compute instances. * Prioritising or segmenting specific types of network traffic (database communication, frontend access…) | N/A |
| Terraform | * Infrastructure as Code Tool uses its domain-specific language HCL to define and deploy cloud resources. Cross-platform, supports change tracking and state management | * Replace manual clicking in cloud consoles. * Promotes audibility and change rollback | Can provision specific storage services on cloud platforms (object storage buckets, databases, etc.) |
| Kubernetes | * Container orchestration platform for managing containerized applications. * Automated deployments, scaling, and rollbacks * Portable and cloud-agnostic | Deploying microservices-based applications, managing large-scale containerized workloads, CI/CD pipelines | Container registries (e.g., Docker Hub), persistent volumes (e.g., cloud storage, local disks) |

# Reference

Contino | Global Transformation Consultancy. (n.d.). *What Is AWS CloudFormation?* [online] Available at: <https://www.contino.io/insights/aws-cloudformation>.

‌MetricFire (n.d.). *The Best Cloud Infrastructure Automation Tools*. [online] www.metricfire.com. Available at: <https://www.metricfire.com/blog/the-best-cloud-infrastructure-automation-tools/#strongAWS-CloudFormationstrong> [Accessed 20 Feb. 2024].

Codefresh. (n.d.). *Infrastructure as Code on AWS: Process, Tools & Best Practices*. [online] Available at: https://codefresh.io/learn/infrastructure-as-code/infrastructure-as-code-on-aws-process-tools-and-best-practices/ [Accessed 20 Feb. 2024].

‌ISO (2022). *ISO/IEC 27001 standard – information security management systems*. [online] ISO. Available at: <https://www.iso.org/standard/27001>.

‌Johann, Sven. (2017). Kief Morris on Infrastructure as Code. IEEE Software. 34. 117-120. 10.1109/MS.2017.13. Cloud specialist Kief Morris joins Software Engineering Radio host Sven Johann to discuss the benefits of infrastructure as code, including security, auditability, testing, documentation, and traceability.

14:00-17:00 (2015). ISO/IEC 27017:2015. [online] ISO. Available at: <https://www.iso.org/standard/43757.html>.

ISO. (2019). ISO/IEC 27018:2019. [online] Available at: <https://www.iso.org/standard/76559.html>.

ISO (2022). *ISO/IEC 27001 standard – information security management systems*. [online] ISO. Available at: <https://www.iso.org/standard/27001>.

‌SyntaxSteve (2023). Managing Costs with AWS Budgets and CloudFormation. [online] Syntax Sessions. Available at: <https://syntaxsessions.com/posts/aws-simple-budget-cloudformation/> [Accessed 20 Feb. 2024].

docs.aws.amazon.com. (n.d.). *stop-instances — AWS CLI 1.27.26 Command Reference*. [online] Available at: <https://docs.aws.amazon.com/cli/latest/reference/ec2/stop-instances.html>.

Wikipedia. (2023). *AWS CloudFormation*. [online] Available at: <https://en.wikipedia.org/wiki/AWS_CloudFormation>.

‌‌ www.youtube.com. (n.d.). *Best Practices of Infrastructure as Code with HashiCorp Terraform*. [online] Available at: <https://www.youtube.com/watch?reload=9&app=desktop&v=T56lZb7WNLc> [Accessed 05 Mar. 2024].

‌ Brikman, Y. (2020). *A comprehensive guide to managing secrets in your Terraform code*. [online] Medium. Available at: <https://blog.gruntwork.io/a-comprehensive-guide-to-managing-secrets-in-your-terraform-code-1d586955ace1> [Accessed 06 Mar. 2024].

GitHub. (n.d.). *Cloud Posse*. [online] Available at: <https://github.com/cloudposse> [Accessed 06 Mar. 2024].

‌