

# Amanox AWS Cloudformation Workshop

## Powered by Amanox

Lab Guide

Using

---



---

Authors:

David Horvath  
John Guentensperger  
Istvan Kappelmayer

## Contents

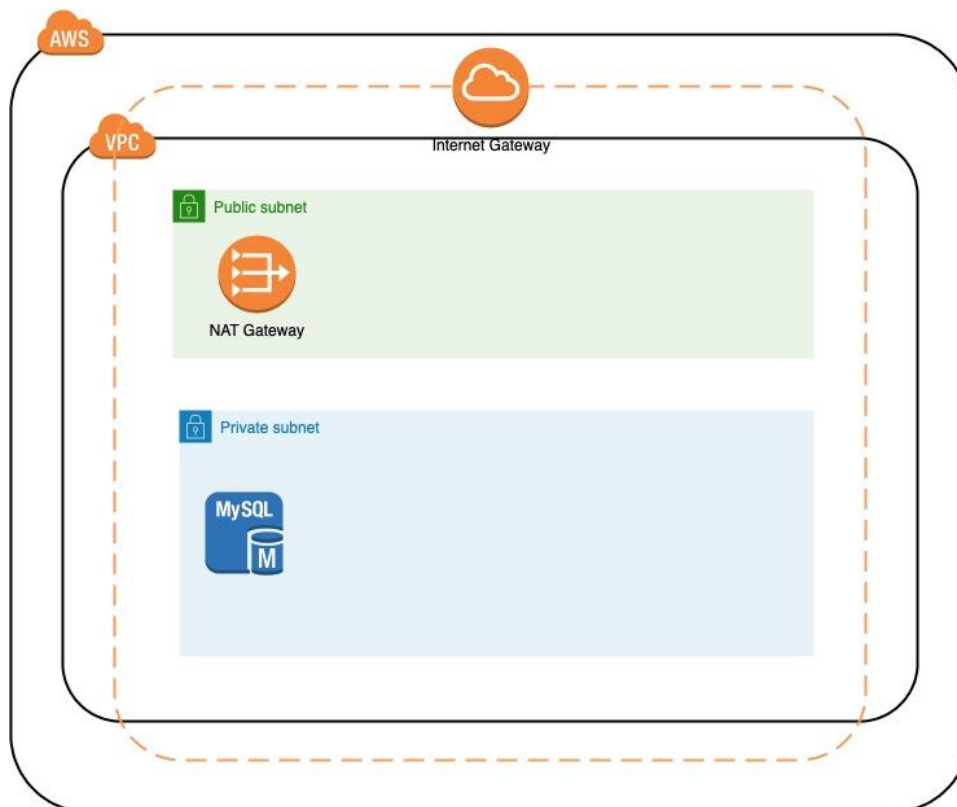
<b>1. Starting Small Lab .....</b>	<b>3</b>
1.1 Common Lab Info .....	3
1.2 AWS Login .....	3
<b>2. Part 1: Creating a Virtual Private Cloud Stack .....</b>	<b>4</b>
2.1 Introduction.....	4
2.2 Learning Goals .....	4
2.3 Duration .....	4
<b>3. Part 1 - Task 1: Upload a CloudFormation template .....</b>	<b>5</b>
3.1 Introduction.....	5
3.2 Learning Goals .....	5
3.3 Task Instructions .....	6
3.4 Summary.....	10
<b>4. Part 1 - Task 2: Create RDS .....</b>	<b>11</b>
4.1 Introduction.....	11
4.2 Learning Goals .....	11
4.3 Duration .....	11
4.4 Task Instructions .....	12
4.4.1 Store the VPC template in an S3 Bucket preparing for reuse.....	12
4.4.2 Create an RDS template .....	13
4.4.3 Create your stack based on your new template .....	17
4.5 Summary.....	17
<b>5. Part 2 – A stripped-down Drupal deployment .....</b>	<b>18</b>
5.1 Introduction.....	18
5.2 Learning Goals .....	19
5.3 Duration .....	19
5.4 Task Instructions .....	19
5.4.1 Store the VPC template structure in an S3 Bucket.....	19
5.4.2 Generate the stack from Amazon S3 URL .....	21
5.5 Summary.....	24
<b>6. Part 3: Add ElastiCache to the previous drupal stack .....</b>	<b>25</b>
6.1 Introduction.....	25
6.2 Learning Goals .....	25
6.3 Instructions.....	26
6.3.1 Look at the change.....	26
6.3.2 Update the S3 bucket .....	26
6.3.3 Update the stack.....	26
6.4 Summary.....	28
<b>7. Conclusion.....</b>	<b>29</b>

## 1. Starting Small Lab

This lab shows you how to start out with AWS CloudFormation.

In this lab you will:

- see CloudFormation in action making use of a ready-made open-source CloudFormation Stack File producing network components of a VPC.
- reuse this template and deploy an RDS database inside your VCP



### 1.1 Common Lab Info

You will find several red **X** markers in the lab guide. You will have to replace this **X** with a new value based on your name. Please follow this example: for **Mike Mosses**, **X = MiMo**.

All snippets and results are served using this git repo: <https://github.com/amanoxsolutions/cloudformation-lab>

### 1.2 AWS Login

Your e-mail address, unless specified otherwise by the Public Cloud Team

In the top-right corner of the AWS Management Console, choose the region you are going to work in. We are going to use **Europe (Frankfurt) eu-central-1**.

## 2. Part 1: Creating a Virtual Private Cloud Stack

### 2.1 Introduction

Services provided by AWS can be daunting at first. The learning curve is steep. Once you learned the tidbits of a service a new kind of problem arises: mundane repetitive tasks. This is where CloudFormation shines, taking away some burden making it easier to reuse commonly applicable building blocks. With the Infrastructure as Code approach people can share their take on configuration just like any other open-source developer would do.

You will deploy a CloudFormation Stack configuration and see:

- A CloudFormation template
- Configuration parameters GUI
- Usable outputs

### 2.2 Learning Goals

You are going to learn how to create a stack from an on-the-fly uploaded template file and how to compose complex structures nesting template inside template. You will learn how to use outputs from the generated stack as input for a consumer stack.

### 2.3 Duration

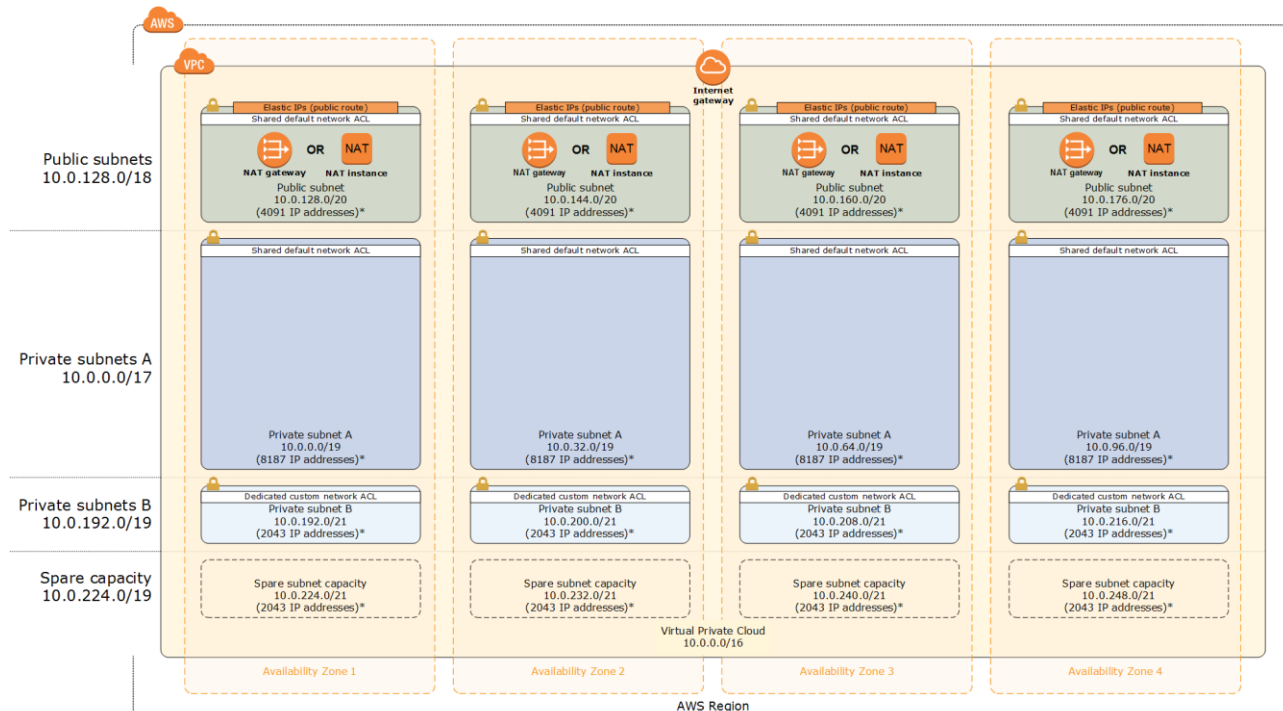
This task requires approximately **20 minutes** to complete.

### 3. Part 1 - Task 1: Upload a CloudFormation template

#### 3.1 Introduction

You will begin by downloading a template for an Amazon Virtual Private Cloud (VPC).

A template file provides full description for CloudFormation of the resources it will produce for you including input parameters, rules for them, interdependencies and outputs.



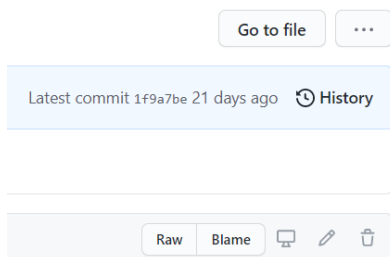
© 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

#### 3.2 Learning Goals

The goal is to familiarize yourself with the creation steps of a CloudFormation Stack. Then you will update it changing its parameters, finally clean up and dispose the resources. We will work with a ready-made big template, but no worries. In a real-world scenario it is also unlikely you can understand every internals of a third-party before it changes again. We will showcase how CloudFormation takes your parameters, produces resources and gives its outputs for further reuse.

### 3.3 Task Instructions

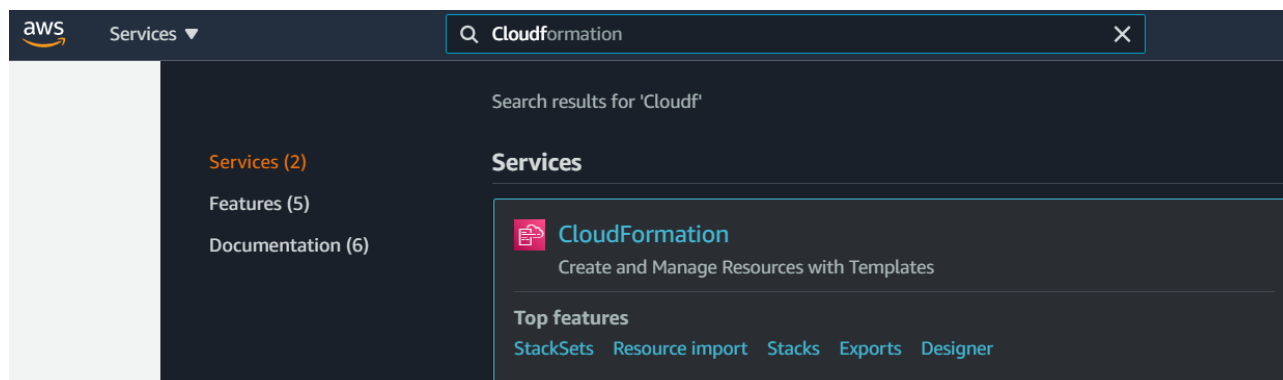
- Download this template which we took directly from the official quickstart-aws-vpc github repo: <https://github.com/amanoxsolutions/cloudformation-lab/blob/master/part1/aws-vpc.template.yaml>
  - a. You can either clone the whole repo as mentioned at the beginning
  - b. Or at the top right of the above url right-click “Raw” button and save the file as **aws-vpc.template.yaml**.



The template is huge. You can have a look into it, but do not feel intimidated by it. The goal is not to understand everything inside, rather to get acquainted how the CloudFormation UI works for you based on your templates.

In the top-right corner of the AWS Management Console, make sure you are using the right region intended for this workshop.

- In the AWS Management Console, on the **Services** menu, search for **CloudFormation**.



The CloudFormation console offers a wizard where you can create, update and destroy your infrastructure or application as stacks.

- In the right navigation pane, click **Create stack**.

A wizard is shown. On top leave **Template is ready** selected, and specify your template you just downloaded via the **Upload a template file** option it.

### Specify template

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

☒ Upload a template file

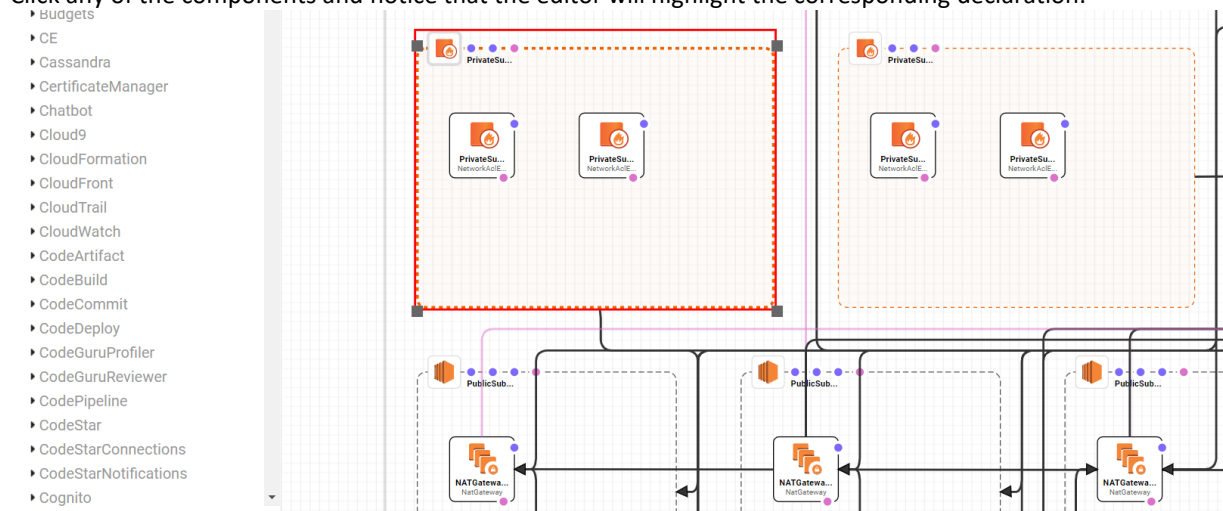
Upload a template file

aws-vpc.template.yaml

JSON or YAML formatted file

S3 URL: <https://s3.eu-central-1.amazonaws.com/cf-templates-xrsyuey2z8d-eu-central-1/2021138rjY-aws-vpc.template.yaml>

- Then click **View in Designer**.  
You will be presented with an auto-generated overview of the resources CloudFormation is about to generate. The Designer is very powerful for visualizing the connections between the components based on the template file.
- Click any of the components and notice that the editor will highlight the corresponding declaration.



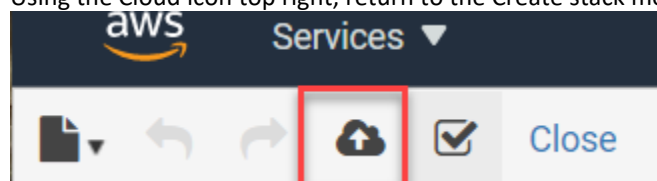
template1

```

2458 Type: 'AWS::EC2::SubnetRouteTableAssociation'
2459 Properties:
2460   SubnetId: !Ref PrivateSubnet2B
2461   RouteTableId: !Ref PrivateSubnet2BRouteTable
2462 Metadata:
2463   'AWS::CloudFormation::Designer':
2464     id: f146d07e-d6f4-49e7-9797-b22d66b10a25
2465 PrivateSubnet2BNetworkACL:
2466   Condition: AdditionalPrivateSubnetsCondition
2467   Type: 'AWS::EC2::NetworkACL'
2468 Properties:
2469   VpcId: !Ref VPC
2470 Tags:
2471   - Key: Name
2472     Value: NACL Protected subnet 2
2473   - Key: Network

```

- Using the Cloud icon top right, return to the Create stack menu:



- Click **Next** to specify stack details page

Name your stack VPC-Stack-**X**

Plenty of parameters are shown. Select exactly two **Availability Zones**.

You can configure any interesting other parameter if you'd like to.

On the bottom of the page click **next**.

- You land on the **Configure stack options** page.

There are two options to be highlighted here.

- **Permissions:** Here you can explicitly set what IAM role to be used during resource creation. For sake of simplicity we leave this as is, but in a more controlled environment you would have the possibility to prepare a template with your power-user rights and then test here how it would work for your restricted consumers.
- Stack creation options > **Rollback on failure:** more complex infrastructures fail frequently during experimentation. By default everything will be rolled back and the half-baked infra is destroyed. Instead of re-starting everything from scratch you can **Disable** the "Rollback on failure option", fix your error and update the stack to continue where you left off.

For now, we leave the defaults here as they are. At the bottom of the page click **next**.

- Review
  - Once you have reviewed your planned stack, scroll down, **acknowledge** the implication of potentially creating IAM resources
  - Click **Create stack**



- Watch as your resources get created.
- On the overview there are several tabs.

VPC-Stack-daho

Delete
Update
Stack actions ▼
Crea

Stack info
Events
Resources
Outputs
Parameters
Template
Change sets

Resources (24)

Q Search resources

Logical ID	Physical ID	Type	Status
DHCOPTIONS	dopt-0025c27cae548a737	AWS::EC2::DHCOPTIONS	CREATE_COMPLETE
InternetGateway	igw-0417742849d80fb71	AWS::EC2::InternetGateway	CREATE_COMPLETE
NAT1EIP	18.193.251.141	AWS::EC2::EIP	CREATE_COMPLETE
NAT2EIP	3.66.6.83	AWS::EC2::EIP	CREATE_COMPLETE
NATGateway1	nat-0705a8f39ae3273c9	AWS::EC2::NatGateway	CREATE_COMPLETE
NATGateway2	nat-	AWS::EC2::NatGateway	CREATE_COMPLETE

Examine **Events**, where you can see how subparts of the template get provisioned.

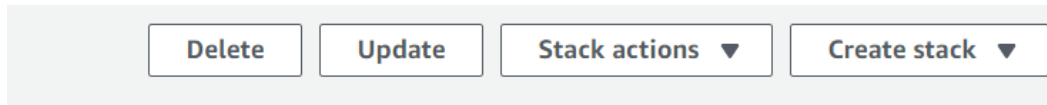
Everything on the **Resources** tab is from now on managed by this stack. Locate the VPC item and open its physical ID, you will land on the usual interface where you can inspect the details of the VPC.

- Wait for the stack to get ready in ca. 3 minutes.

On the **Outputs** tab you will find details about your resources and other produced values the template defined *once the stack creation has finished successfully*.

- Open again the Resources Tab.

Notice that there are as many Elastic Ips or Nat Gateways as many Availability Zones we specified (2). Let's **Update** the stack to include a third one. On top click **Update**.



Here you could choose to edit your template in the designer on the fly. **Use current template** for now and click **next**.

Add the third Availability Zone and set the Number of Availability Zones to 3.

#### Availability Zone Configuration

##### Availability Zones

List of Availability Zones to use for the subnets in the VPC. Note: The logical order is preserved.

eu-central-1a ✕ eu-central-1b ✕ eu-central-1c ✕

##### Number of Availability Zones

Number of Availability Zones to use in the VPC. This must match your selections in the list of Availability Zones parameter.

Note: Don't forget to also set the Number of Availability Zones to 3.

Click **Next** and on the Configure stack options again **Next**. Acknowledge and **Update stack**.

Notice on the **Events** and **Resources** tabs that CloudFormation creates the extra resources leaving the existing ones in place. The Stack update takes ca. 3 minutes.

- Once you see the **UPDATE\_COMPLETE** status of the Stack, on the top click **Delete** to remove the Stack and its associated resources.

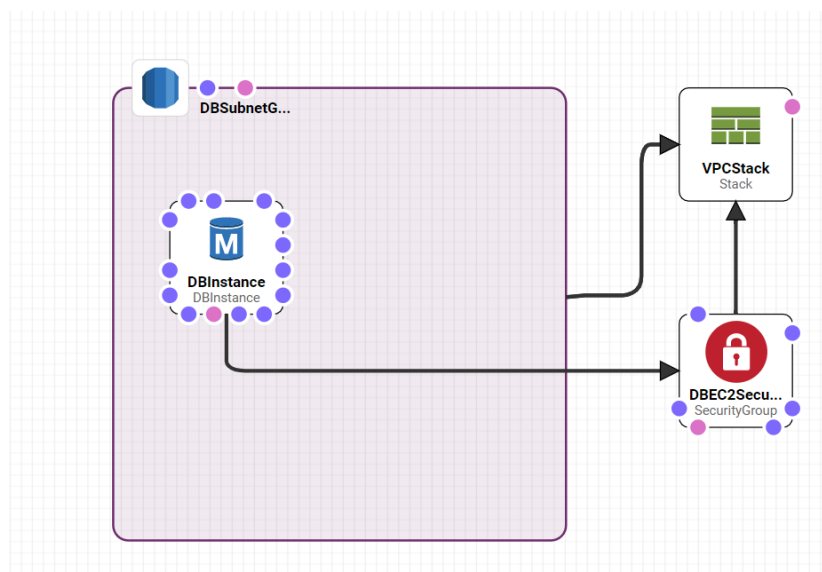
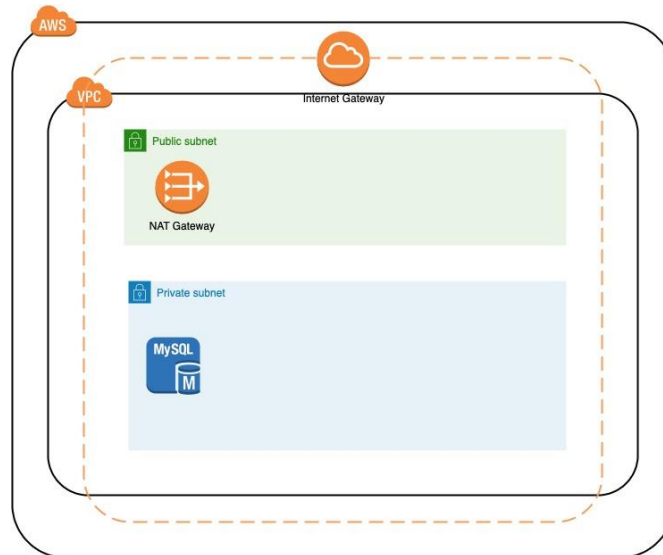
### 3.4 Summary

You now have played with a ready-made template, created a Stack based on it, updated it and finally destroyed it. The next step will be starting an RDS db reusing this VPC template but first let's jump back to the presentation.

## 4. Part 1 - Task 2: Create RDS

### 4.1 Introduction

RDS is the Relational Database Service offering from AWS where the mainstream database engines are covered. AWS manages the tiresome updates, patching and scaling of your database instances offering durability, high availability and security for you while you can focus on creating more value. RDS is at the hearth of many web applications, at the mercy of the agile nature of application changes so it is another great candidate to demonstrate the power of Infrastructure as Code.



### 4.2 Learning Goals

The goal is to learn and see the creation steps of an RDS instance started into a similar VPC reusing the template we have seen previously, creating a nested stack.

This task takes approximately 45 minutes. If you get stuck it is recommended to use the provided snippets, but please try to struggle through the task, try to also understand what you do, not just copy and paste.

### 4.3 Duration

The preparation takes ca. 25 minutes, then 20 minutes stack creation.

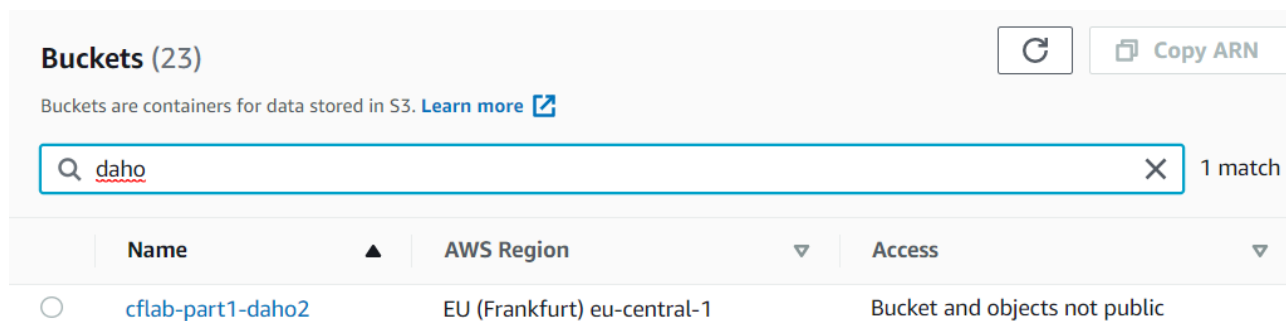
## 4.4 Task Instructions

### 4.4.1 Store the VPC template in an S3 Bucket preparing for reuse

- In the AWS Management Console, on the **Services** menu, search for **S3**.
- Click **Create bucket** and configure:
  - Bucket name: cflab-part1-**X**
  - **Make sure to NOT have any whitespace in the bucket name**
  - AWS Region: as specified at the beginning of this guide.
  - Leave the other set of options as is.

At the bottom, click create bucket.

You can efficiently search for your bucket entering your **X** part to the filter field:

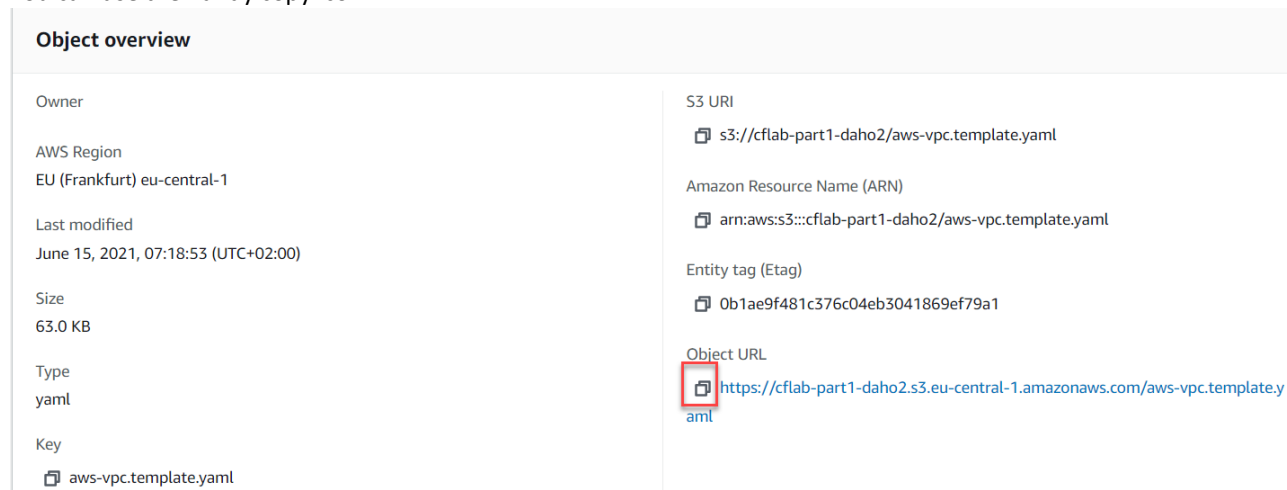


The screenshot shows the AWS S3 Buckets console. At the top, it says "Buckets (23)" with a refresh button and a "Copy ARN" button. Below this, a message states "Buckets are containers for data stored in S3. [Learn more](#)". A search bar contains the text "daho" and shows "1 match". Below the search bar, a table lists the bucket details:

Name	AWS Region	Access
cflab-part1-daho2	EU (Frankfurt) eu-central-1	Bucket and objects not public

- Click the name of your new bucket
- **Upload** your aws-vpc.template.yaml file.
- Click the created object in the Name Column to open its details. You will need the **Object URL** later.

You can use the handy copy icon:



The screenshot shows the "Object overview" page for the uploaded file "aws-vpc.template.yaml". The page is divided into two columns. The left column contains metadata: Owner, AWS Region (EU (Frankfurt) eu-central-1), Last modified (June 15, 2021, 07:18:53 (UTC+02:00)), Size (63.0 KB), Type (yaml), and Key (aws-vpc.template.yaml). The right column contains: S3 URI (s3://cflab-part1-daho2/aws-vpc.template.yaml), Amazon Resource Name (ARN) (arn:aws:s3:::cflab-part1-daho2/aws-vpc.template.yaml), Entity tag (Etag) (0b1ae9f481c376c04eb3041869ef79a1), and Object URL (https://cflab-part1-daho2.s3.eu-central-1.amazonaws.com/aws-vpc.template.yaml). The Object URL is highlighted with a red box and a copy icon.

#### 4.4.2 Create an RDS template

- Create a new textfile named e.g. myrds.template.yaml and start editing it with your IDE of choice.

We will use several snippets to assemble our working db.

You can find the template Skeleton here: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html> Copy the YAML format into your new template.

Set the **AWSTemplateFormatVersion** to '2010-09-09'

Write a **Description**, e.g. 'db template'

You can remove the **Metadata, Rules, Mappings, Conditions, Transform** sections

At the Resources section we add our DB. At the bottom of the **Amazon RDS template snippets** page we will make use of the section **Amazon RDS database instance in a VPC security group**'s YAML code: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/quickref-rds.html#w2ab1c27c21c76c15>

- Notice that (for YAML) you need to indent with 2 blank spaces the copied code to become a subsection of Resources.
- The **DBInstance**'s **VPCSecurityGroups** property refers to **!GetAtt DBEC2SecurityGroup.GroupId** which says basically "once **DBEC2SecurityGroup** exists, take it's **GroupId** attribute"

- There are many other properties in the form **Ref: SomeParameter**. We will need to add these at the **Parameters** section as string values. The format can be found here: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html> For our purpose now, these all will be strings.

Parameters:

```
DBAllocatedStorage:
  Type: String
  Default: '20'
DBName:
  Type: String
  Default: MyDB
DBUser:
  Type: String
  Default: admin
DBPassword:
  Type: String
  NoEcho: 'True'
  MaxLength: '41'
  MinLength: '8'
DBClass:
  Type: String
  AllowedValues:
    - db.t2.small
    - db.t2.medium
    - db.r3.large
    - db.r3.xlarge
    - db.r3.2xlarge
    - db.r3.4xlarge
    - db.r3.8xlarge
  Default: db.t2.small
MultiAZDatabase:
  Type: String
  AllowedValues:
    - 'true'
    - 'false'
  Default: 'true'
```

- Now let's have a look at **DBEC2SecurityGroup's SecurityGroupIngress** section, there is a reference pointing to **Web-ServerSecurityGroup**. This would be relevant if we configured some application working with the db, but for the purpose of this demo we can remove this property.

However, we want to launch our DB instance into a VPC, which will come from our existing VPC template, instantiated through a Stack resource: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-stack.html>

We recommend to name your stack resource VPCStack.

To create a stack resource, you need to specify the **TemplateURL** to be the same S3 url you noted at the beginning, e.g. <https://cflab-part1-x.s3.eu-central-1.amazonaws.com/aws-vpc.template.yaml>

As you can remember there are a lot of parameters for the VPC template available, however for the purpose of this lab we expose only few of them. We propagate these exposed parameters towards the vpc stack resource, omitting all the other vpc stack parameters, leaving them as default.

#### Parameters:

#...

##### VPCAvailabilityZones:

Description: 'List of Availability Zones to use for the subnets in the VPC'

Type: List<AWS::EC2::AvailabilityZone::Name>

##### VPCNumberOfAZs:

AllowedValues:

- '2'
- '3'
- '4'

Default: '2'

Description: Number of Availability Zones to use in the VPC

Type: String

#### Resources:

##### VPCStack:

Type: AWS::CloudFormation::Stack

Properties:

TemplateURL: "https://cflab-part1-x.s3.eu-central-1.amazonaws.com/aws-vpc.template.yaml"

# ^ TODO adjust your bucket url

Parameters:

AvailabilityZones: !Join

- ','
- !Ref 'VPCAvailabilityZones'

NumberOfAZs: !Ref 'VPCNumberOfAZs'

And that is it: we just "imported" our existing VPC template. We made some of it configurable for the consumers of our future template as well. We can now build onto the VPC.

- Now, according to [https://docs.amazonaws.cn/en\\_us/AmazonRDS/latest/UserGuide/USER\\_VPC.Working-WithRDSInstanceinaVPC.html#USER\\_VPC.InstanceInVPC](https://docs.amazonaws.cn/en_us/AmazonRDS/latest/UserGuide/USER_VPC.Working-WithRDSInstanceinaVPC.html#USER_VPC.InstanceInVPC) let's create a DB subnet group making use of our VPC-Stack's output: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbsubnet-group.html>

**DBSubnetGroup:**

Type: `AWS::RDS::DBSubnetGroup`

Properties:

`DBSubnetGroupDescription`: Subnets available for the DB Instance

`SubnetIds`:

- `!GetAtt VPCStack.Outputs.PrivateSubnet1AID`
- `!GetAtt VPCStack.Outputs.PrivateSubnet2AID`

- Then we need to adjust our Security Group for the DB. As mentioned, we don't care for the `SecurityGroupIngress` for now, but have to attach to the `VpcId`:

**DBEC2SecurityGroup:**

Type: `AWS::EC2::SecurityGroup`

Properties:

`GroupDescription`: Security group for the DB

`VpcId`: `!GetAtt VPCStack.Outputs.VPCID`

- Finally we have to specify the **DBSubnetGroup** for the **DBInstance** using the **DBSubnetGroupName** property

**DBInstance:**

Type: `AWS::RDS::DBInstance`

Properties:

`#...`

`VPCSecurityGroups`:

- `!GetAtt DBEC2SecurityGroup.GroupId`
- `DBSubnetGroupName`: `!Ref DBSubnetGroup`



#### 4.4.3 Create your stack based on your new template

In AWS Console go to the **CloudFormation** Service and chose **Create stack**.

Similarly to the earlier example, click **Upload a template file** and upload your newly created template yaml file. Click **Next**.

**Note: the VPC will be created based on the template you have uploaded to the S3 bucket. Make sure, the Bucket URL is correct in the yml file.**

Name your stack rds-**X** and take care of the following parameters:





- DBPassword
- Select exactly two VPCAvailabilityZones
- Leave VPCNumberOfAZs as the default 2

Click **Next**.

At the Stack creation options it is now strongly recommended to **Disable Rollback on failure** to spare some time in case of failures. Click **Next**.

On the next page **Acknowledge** everything and **Create Stack**

Notice how the process instantiates first the nested VPCStack and once it is up will be the dbinstance configured.

<p><b>NESTED</b> db8-VPCStack-1HUE7YQX4AZGZ 2021-05-18 10:46:29 UTC+0200  CREATE_IN_PROGRESS</p>	<p><b>NESTED</b> db8-VPCStack-1HUE7YQX4AZGZ 2021-05-18 10:46:29 UTC+0200  CREATE_COMPLETE</p>
<p>db8 2021-05-18 10:46:24 UTC+0200  CREATE_IN_PROGRESS</p>	<p>db8 2021-05-18 10:46:24 UTC+0200  CREATE_IN_PROGRESS</p>

Once it gets to the db creation, it takes relatively long. If you don't see any failure once the VPC Stack is ready, congratulation. The whole stack takes ca. 21 minutes.

If you get stuck with too many failures, feel free to take a look at the fully working example:

<https://github.com/amanoxsolutions/cloudformation-lab/blob/master/part1/myrds.template.yaml>

Once ready, delete all your stacks.

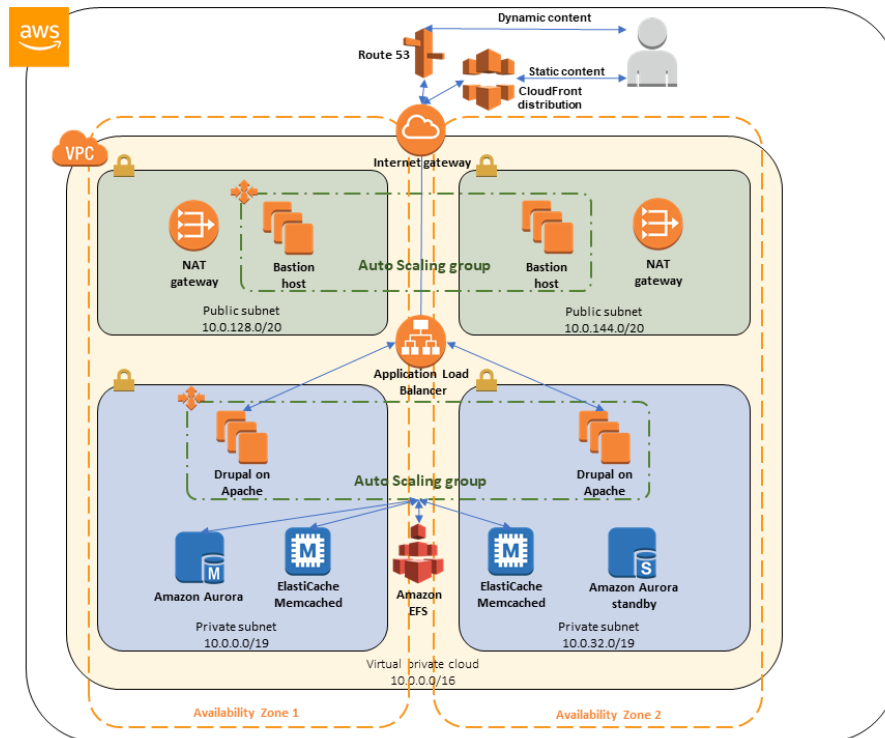
#### 4.5 Summary

Now you have seen a nested stack in action. Let's return to the presentation.

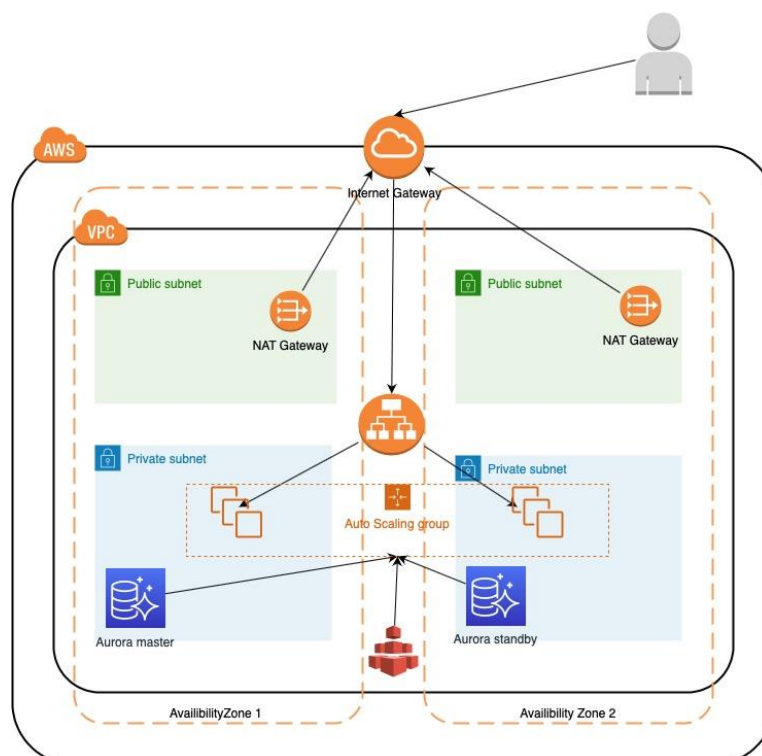
## 5. Part 2 – A stripped-down Drupal deployment

### 5.1 Introduction

There is a full-fledged template-set reference deployment on aws for **Drupal on AWS** here: <https://aws.amazon.com/quickstart/architecture/drupal/>



Drupal is a CMS which is strongly depending on a real domain name. To avoid having to work with own domains, we tweaked a little with the config, dropping the *Route 53* zone. For the purpose of this demo we also don't need *CloudFront* or any *bastion* host. We use the dns name implicitly provided by the Application Load Balancer:



## 5.2 Learning Goals

The goal is to see a bigger set of resources composed from well-rounded stack templates deployed using S3 bucket.

## 5.3 Duration

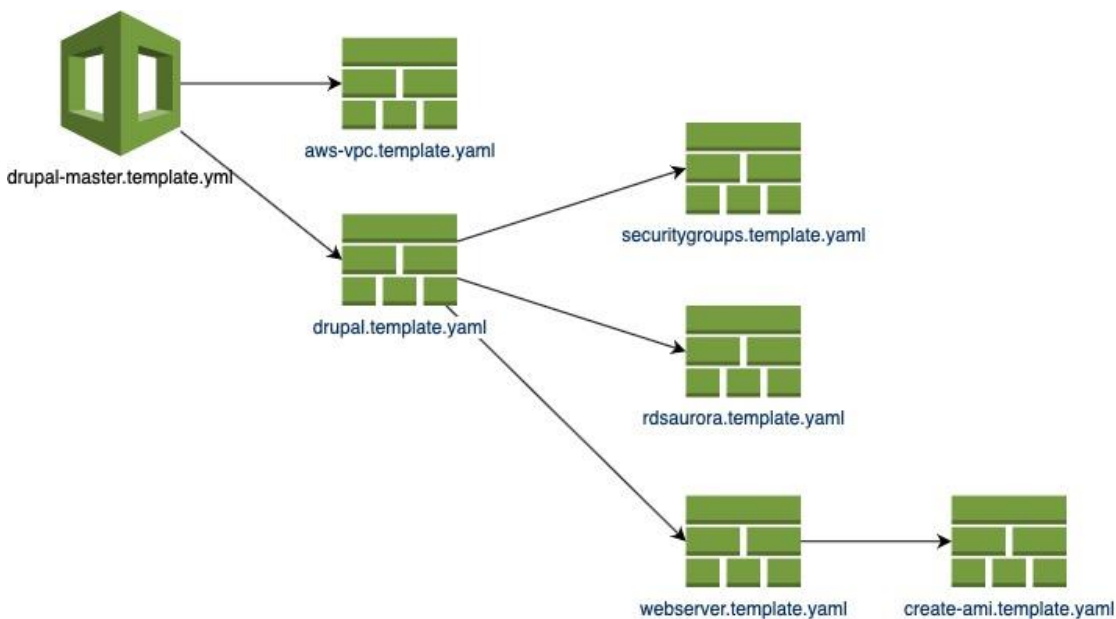
The preparation takes ca. 20 minutes. The whole stack creation takes ca. 30 minutes.

## 5.4 Task Instructions

We will work with the template structure from the git repo's following folder:

<https://github.com/amanoxsolutions/cloudformation-lab/tree/master/part2/quickstart-drupal>

There is a master-template where everything can be configured, and it nests other stacks to produce the drupal-stack:



### 5.4.1 Store the VPC template structure in an S3 Bucket

- In the AWS Management Console, on the **Services** menu, search for **S3**.
- Click **Create bucket** and configure:
  - Bucket name: cflab-part2-X
  - **Make sure to NOT have any whitespace in the bucket name**
  - AWS Region: as specified at the beginning of this guide.
  - Leave the other set of options as is.

At the bottom, click create bucket.

- Create the same folder structure in S3 as in the part2 git folder:  
<https://github.com/amanoxsolutions/cloudformation-lab/tree/master/part2>

It is important to have the same structure, meaning the top-most folder in the bucket should be named quickstart-drupal.

The desired structure:

```
quickstart-drupal
├── submodules
│   └── quickstart-aws-vpc
│       └── templates
│           └── aws-vpc.template.yaml
└── templates
    ├── create-ami.template.yaml
    ├── drupal-master.template.yaml
    ├── drupal.template.yaml
    ├── rdsaurora.template.yaml
    ├── securitygroups.template.yaml
    └── webserver.template.yaml
```

Open the whole part2 directory with your IDE. Locate some or all of the **AWS::CloudFormation::Stack** code fragments using your local IDE.

You will see a pattern like the following. Notice how the TemplateURL combines 4 parameters resulting in the TemplateURL property for a Stack. This is a good practice to make one's template really reusable and independent from regions.

#### SecurityGroupsStack:

Type: **AWS::CloudFormation::Stack**

Properties:

**TemplateURL:**

**!Sub**

```
- 'https://${S3Bucket}.s3.${S3Region}.${AWS::URLSuffix}/${QSS3KeyPrefix}templates/securitygroups.template.yaml'
- S3Region: !If [UsingDefaultBucket, !Ref 'AWS::Region', !Ref QSS3Bucket-Region]
```

```
    S3Bucket: !If [UsingDefaultBucket, !Sub '${QSS3BucketName}-${AWS::Region}', !Ref QSS3BucketName]
```

Parameters:

VPC: !Ref 'VPCID'

VPCCIDR: !Ref 'VPCCIDR'

#### 5.4.2 Generate the stack from Amazon S3 URL

Copy the master template url from S3 clicking onto the Object and locating the **Object URL**. It should look like the following:

<https://cflab-part2-x.s3.eu-central-1.amazonaws.com/quickstart-drupal/templates/drupal-master.template.yaml>

Go to the **CloudFormation** Service and create a stack:

### Specify template

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
 ☐ Upload a template file

Amazon S3 URL

Amazon S3 template URL

S3 URL: Will be generated when URL is provided

[View in Designer](#)

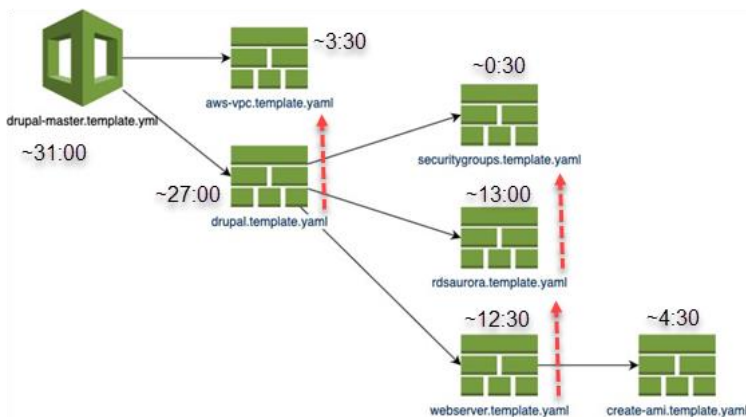
Cancel **Next**

- Click the created object in the Name Column to open its details. You will need the **Object URL** later.
- Name your stack drupal-~~X~~
- Fill in the following parameters carefully:

availability zones	eu-central-1a, eu-central-1b	
database admin password	dbadminpw	At your discretion, min 8 chars
drupal site admin email	Your mail	
drupal site admin password	siteadminpw	At your discretion, min 8 chars
drupal database password	databasepw	At your discretion, min 8 chars
autoscaling notification email	Your mail	
quick start s3 bucket name	cflab-part2-x	This one is really really important

- Click **Next**
- At **Stack creation options**, **Disable Rollback on failure** as this stack takes average 30 minutes to deploy.
- Click **Next**
- **Acknowledge** all, **Create stack**

The stack creation takes ca. 31 minutes. The building blocks in a column depend on the blocks above them:



During stack creation, you can inspect what is currently happening using the Events Tab if you click onto the nested stacks (which get created first due to dependencies):

**Stacks (2)**

Filter by stack name

Active View nested

drupal10-VPCTest-1LHHCFPD6Z6PW

2021-05-18 12:15:24 UTC+0200

CREATE\_IN\_PROGRESS

drupal10

2021-05-18 12:15:17 UTC+0200

CREATE\_IN\_PROGRESS

**drupal10-VPCTest-1LHHCFPD6Z6PW** NESTED

Delete Update Stack actions Create stack

Stack info Events Resources Outputs Parameters Template Change sets

**Events (65)**

Search events

Timestamp	Logical ID	Status	Status reason
2021-05-18 12:16:32 UTC+0200	NATGateway1	CREATE_IN_PROGRESS	Resource creation Initiated
2021-05-18 12:16:32 UTC+0200	NATGateway1	CREATE_IN_PROGRESS	-
2021-05-18 12:16:31 UTC+0200	NATGateway2	CREATE_IN_PROGRESS	Resource creation Initiated
2021-05-18 12:16:30 UTC+0200	NATGateway2	CREATE_IN_PROGRESS	-
2021-05-18 12:16:28 UTC+0200	PublicSubnet1RouteTableAssociation	CREATE_COMPLETE	-
2021-05-18 12:16:28 UTC+0200	NAT1EIP	CREATE_COMPLETE	-
2021-05-18 12:16:28 UTC+0200	PublicSubnet2RouteTableAssociation	CREATE_COMPLETE	-
2021-05-18 12:16:27 UTC+0200	NAT2EIP	CREATE_COMPLETE	-
2021-05-18 12:16:27 UTC+0200	PrivateSubnet2ARouteTableAssociation	CREATE_COMPLETE	-
2021-05-18 12:16:26 UTC+0200	PrivateSubnet1ARouteTableAssociation	CREATE_COMPLETE	-
2021-05-18 12:16:25 UTC+0200	PublicSubnetRoute	CREATE_COMPLETE	-

Examine the webserver instance code at: <https://github.com/amanoxsolutions/cloudformation-lab/blob/master/part2/quickstart-drupal/templates/webserver.template.yaml#L331>

This is where CloudFormation configures the ec2 instance upon startup. You can find more info about this here:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html>

While getting acquainted with the code, periodically look at the process of the stack creation. After it gets to the stage when the WebserverStack is being deployed, open the Resources Tab, locate the instance id and navigate to it.

drupal6-DrupalStack-1V9FQO2GQKZR-WebserverStack-12AHBWCDCL44X

Delete Update Stack actions Create stack

Stack info Events Resources Outputs Parameters Template Change sets

Resources (11)

Search resources

Logical ID	Physical ID	Type
ALBHTTPListener	arn:aws:elasticloadbalancing:eu-central-1:108695687567:listener/app/drupa-Appli-1RWCFTFLG1V8X/91aae00f0e366524/8f94f527e1ee5db8	AWS::ElasticLoadBalancingV2::Listener
ALBTargetGroup	arn:aws:elasticloadbalancing:eu-central-1:108695687567:targetgroup/drupa-ALBTa-1HINBNSZRE8JK/4dd672aea12d7055	AWS::ElasticLoadBalancingV2::TargetGroup
ApplicationLoadBalancer	arn:aws:elasticloadbalancing:eu-central-1:108695687567:loadbalancer/app/drupa-Appli-1RWCFTFLG1V8X/91aae00f0e366524	AWS::ElasticLoadBalancingV2::LoadBalancer
DescribeElastiCachePolicy	drupa-Desc-DX9R7KH0CY99	AWS::IAM::Policy
DrupalAMIEC2Instance	<a href="#">i-080785c37cab13f0b</a>	AWS::EC2::Instance

Select the sole instance, and open the **Actions | Monitor and troubleshoot | Get system log** menu.

Instances (1/1) Info

Filter instances

search: i-080785c37cab13f0b Clear filters

Name	Instance ID	Instance state	Instance type	Status check
drupal6-DrupalStack-1V9F...	i-080785c37cab13f0b	Terminated	t2.micro	-

Connect View details Manage instance state Instance settings Networking Security Image and templates Monitor and troubleshoot

Get system log Get instance screenshot Manage detailed monitoring Manage CloudWatch alarms EC2 Serial Console Replace root volume

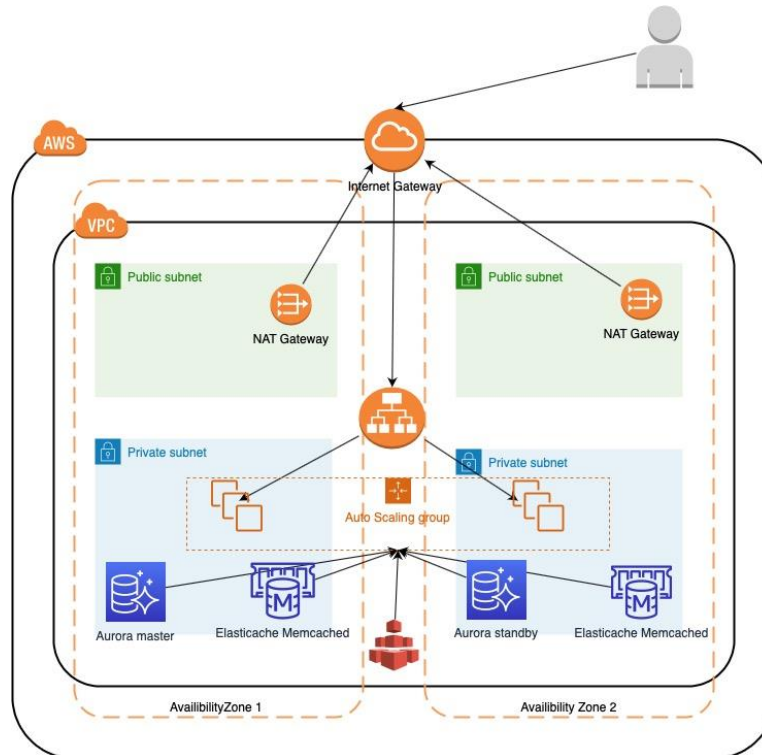




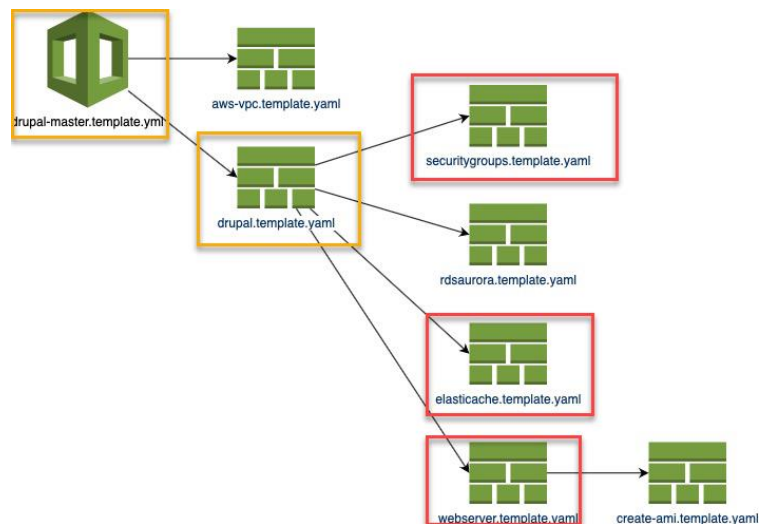
## 6. Part 3: Add ElastiCache to the previous drupal stack

### 6.1 Introduction

In this second part, you will enhance the drupal webserver stack with ElastiCache (using memcached).



The templates marked with red are involved in the structural extension, and the yellow boxes show the propagation of new parameters:



### 6.2 Learning Goals

The goal is to showcase updating the structure in a cross-cutting aspect.

## 6.3 Instructions

### 6.3.1 Look at the change

<https://github.com/amanoxsolutions/cloudformation-lab/commit/956719353fbcacb79e6d0144a49291f3b1816150>

There are 4 changes:

- The drupal-master.template.yml and drupal.template.yaml templates got new parameters for the elasticache configuration. As visible on the structure picture above these are merely propagating the configs to their nested sets.
- The drupal.template.yaml has an interesting construct:  
It creates a **Condition** to determine whether to produce the elasticache stack: <https://github.com/amanoxsolutions/cloudformation-lab/commit/956719353fbcacb79e6d0144a49291f3b1816150#diff-e6559fbbe40f78dae5503b3a30a20846e459eda32218afdd2631df477f5cd531R417>  
If this evaluates to true, <https://github.com/amanoxsolutions/cloudformation-lab/commit/956719353fbcacb79e6d0144a49291f3b1816150#diff-e6559fbbe40f78dae5503b3a30a20846e459eda32218afdd2631df477f5cd531R454> will go live
- elasticache.template.yaml is the stack where the required resources will be produced.
- The webserver.template.yaml is the most involved, installing Memcached, registering with the cluster, etc.

### 6.3.2 Update the S3 bucket

In S3 update the contents of your part2 bucket to represent <https://github.com/amanoxsolutions/cloudformation-lab/tree/master/part3>

### 6.3.3 Update the stack

In CloudFormation on the stack overview click Update and update the root stack.

## Update stack

### Prerequisite - Prepare template

#### 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.

☐ Use current template

☒ Replace current template

☐ Edit template in designer

### Specify template

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

☐ Upload a template file

#### Amazon S3 URL

<https://cflab-part2-x.s3.eu-central-1.amazonaws.com/quickstart-drupal/templates/drupal-master.template.yaml>

Amazon S3 template URL

S3 URL: <https://david-drupal-cflab-v2.s3.eu-central-1.amazonaws.com/quickstart-drupal/templates/drupal-master.template.yaml>

[View in Designer](#)

Cancel

Next

Leave every parameter as is and click **Next, Next**.

Notice in the preview that CloudFormation detects existing parts and will not replace but rather modify the stack:

Change set preview

Action	Logical ID	Physical ID	Resource type	Replacement	Module
<a href="#">Modify</a>	DrupalStack	arn:aws:cloudformation:eu-west-1:216751597651:stack/drupal-daho-DrupalStack-1FK4Z8S9XYXBI/4c5d4be0-cda6-11eb-9487-027d8cb6d92b <a href="#">🔗</a>	AWS::CloudFormation::Stack	False	-
<a href="#">Modify</a>	VPCStack	arn:aws:cloudformation:eu-west-1:216751597651:stack/drupal-daho-VPCStack-8FCDBEES9OLB/b94b9b90-cda5-11eb-9f7f-02f1248090e1 <a href="#">🔗</a>	AWS::CloudFormation::Stack	False	-

## Acknowledge all, Update stack

**Stacks (8)** [🔄](#)

Active

< 1 >

<b>NESTED</b> drupal10-DrupalStack-15ZV6QI3VWOGT-ElastiCacheStack-18YSOD017FGB2 2021-05-18 16:50:01 UTC+0200 <a href="#">🔍</a> <a href="#">CREATE_IN_PROGRESS</a>	<input type="radio"/>
<b>NESTED</b> drupal10-DrupalStack-15ZV6QI3VWOGT-WebserverStack-IMI8QCF8Zi9Z-CreateDrupalAMI-Q7PB10UK9IR7 2021-05-18 12:40:09 UTC+0200 <a href="#">🟢</a> <a href="#">CREATE_COMPLETE</a>	<input type="radio"/>
<b>NESTED</b> drupal10-DrupalStack-15ZV6QI3VWOGT-WebserverStack-IMI8QCF8Zi9Z 2021-05-18 12:33:32 UTC+0200 <a href="#">🟢</a> <a href="#">UPDATE_COMPLETE</a>	<input type="radio"/>
<b>NESTED</b> drupal10-DrupalStack-15ZV6QI3VWOGT-RDSAuroraStack-DB6B4K36LU7U 2021-05-18 12:20:16 UTC+0200 <a href="#">🟢</a> <a href="#">CREATE_COMPLETE</a>	<input type="radio"/>
<b>NESTED</b> drupal10-DrupalStack-15ZV6QI3VWOGT-SecurityGroupsStack-5D5P70HZVDRC 2021-05-18 12:19:38 UTC+0200 <a href="#">🔍</a> <a href="#">UPDATE_COMPLETE_CLEANUP_IN_PROGRESS</a>	<input type="radio"/>
<b>NESTED</b> drupal10-DrupalStack-15ZV6QI3VWOGT 2021-05-18 12:19:31 UTC+0200 <a href="#">🔍</a> <a href="#">UPDATE_IN_PROGRESS</a>	<input type="radio"/>
<b>NESTED</b> drupal10-VPCStack-1LHHCFPD6Z6PW 2021-05-18 12:15:24 UTC+0200 <a href="#">🟢</a> <a href="#">CREATE_COMPLETE</a>	<input type="radio"/>
<b>drupal10</b> 2021-05-18 12:15:17 UTC+0200 <a href="#">🔍</a> <a href="#">UPDATE_IN_PROGRESS</a>	<input checked="" type="radio"/>

The update takes about 7 minutes to complete.

#### **6.4 Summary**

To sum up, you now have seen a really complex system fully provisioned and configured via nested CloudFormation stacks.

## 7. Conclusion

Congratulations! You completed the Lab.

You now have learned how to:

- Understand, parts of CloudFormation template files, write and extend upon them
- Modularize cohesive subsets of your infrastructure with nested stacks and reuse parts in other scenarios.
- Organize your templates in an S3 bucket
- Update stacks