COMP5349– Cloud Computing Week 8: CloudFormation

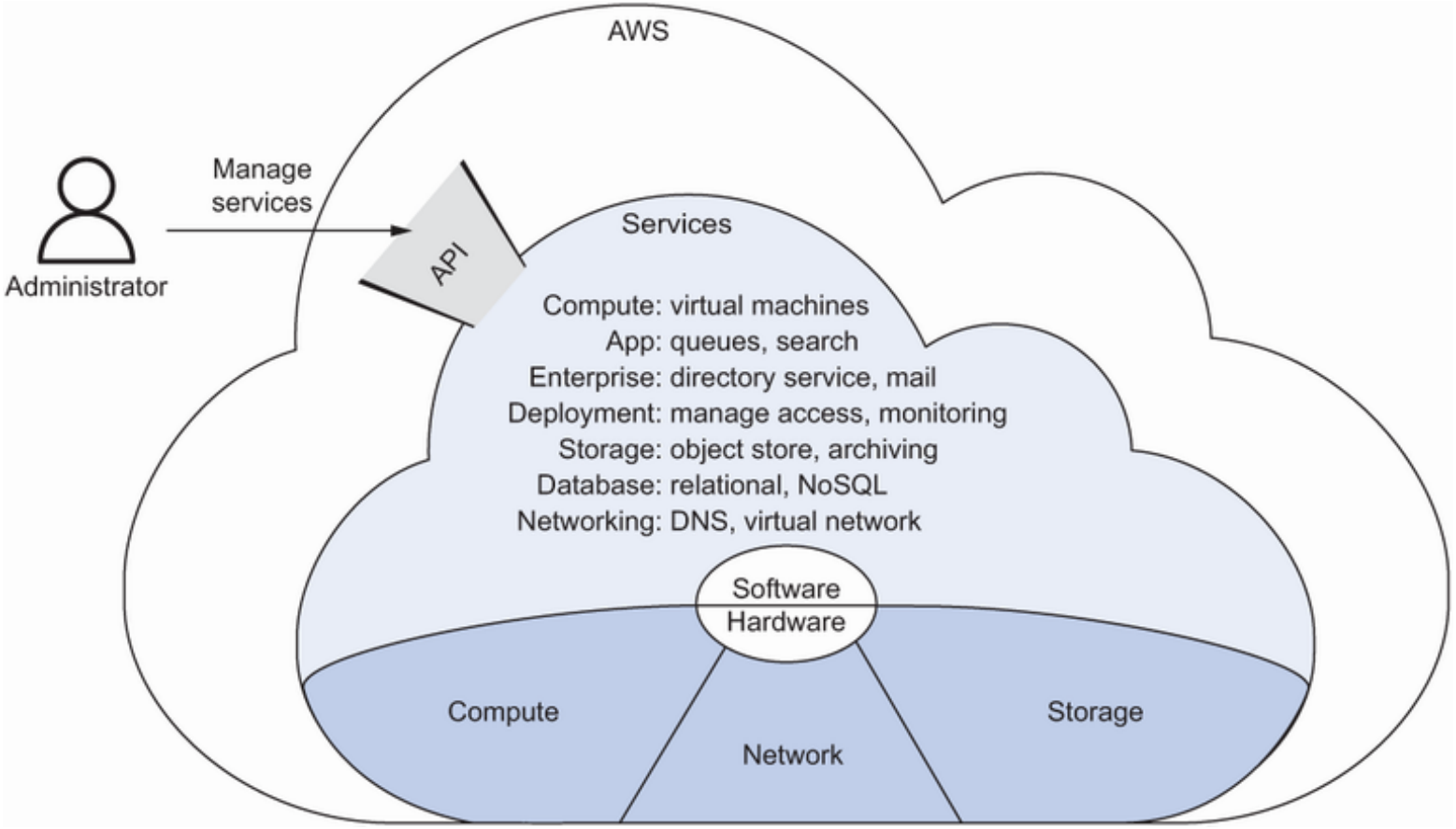
Dr. Ying Zhou
The University of Sydney

Table of Contents

AWS command line interface

AWS Services and API calls

- Services are created and managed by sending requests to the corresponding API
  - Through a web-based GUI like management console
  - Command line interface like AWS CLI
  - Programmatically via SDK
- Creating an EC2 instance
  - Perform an **ec2:RunInstances** call



Data center hardware

Figure 1.9 The AWS cloud is composed of hardware and software services accessible via an API.

Four options of interacting with AWS

- They represent different end user interfaces of the same API
  - **Management Console**
  - *Command line*
  - SDK
  - Blueprints

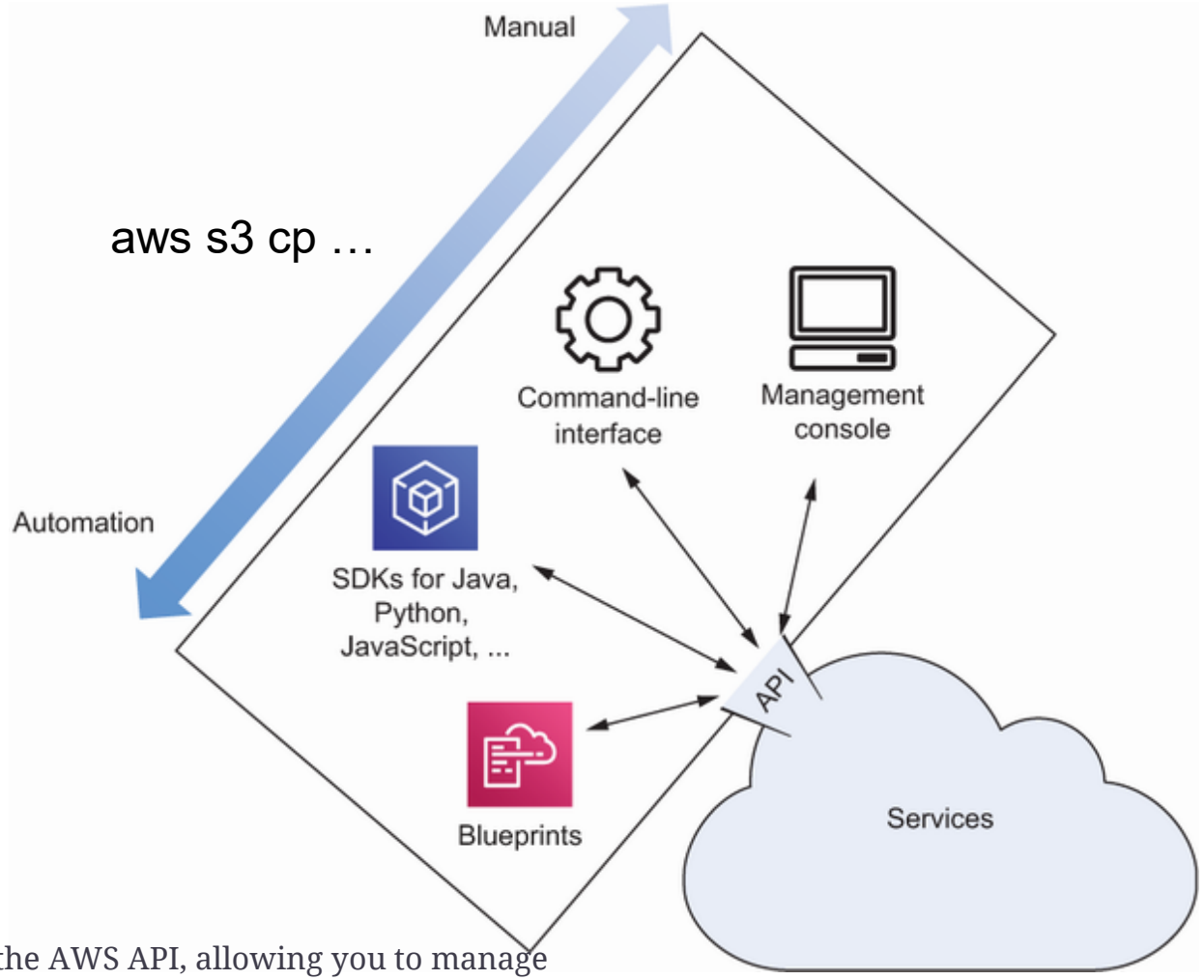
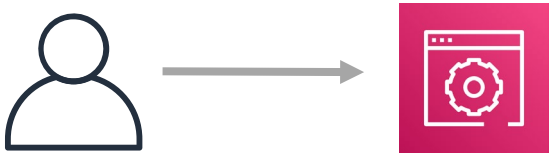


Figure 1.12 Different ways to access the AWS API, allowing you to manage and access AWS services

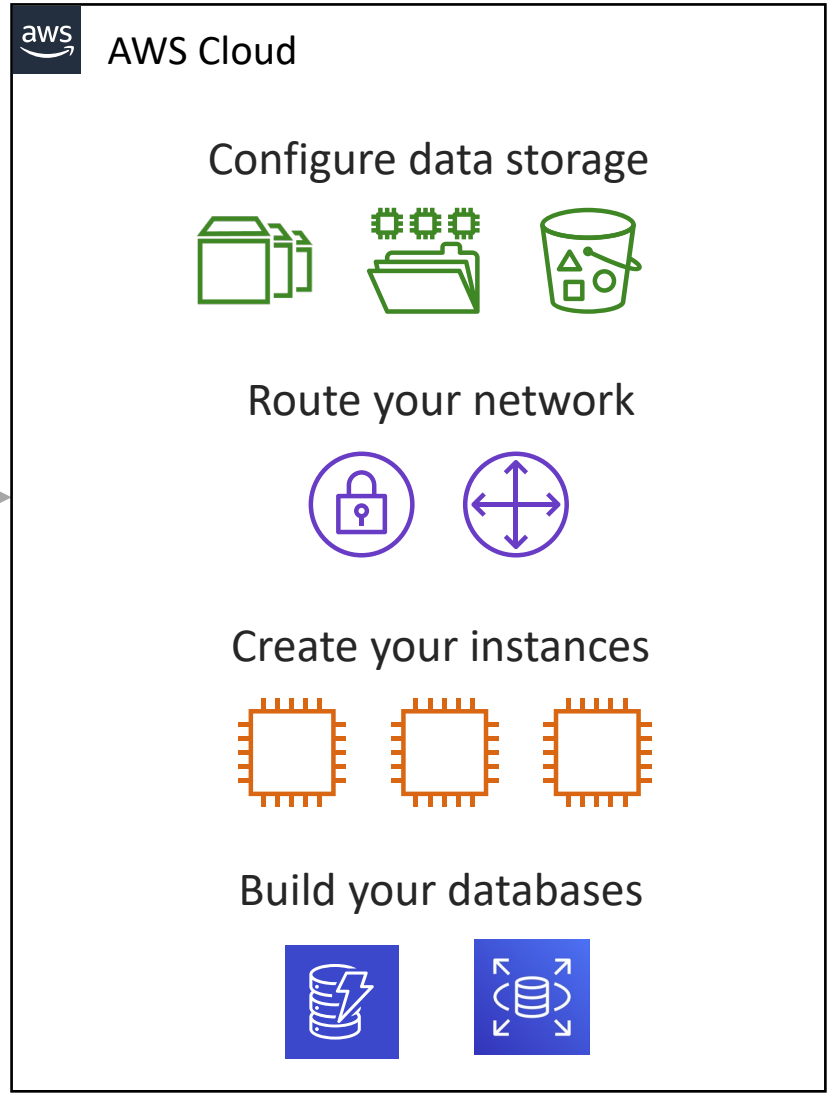
Management Console: manual process

Long *manual process* to build an architecture

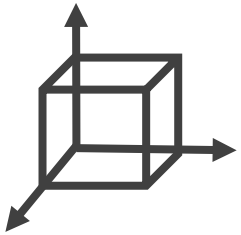


You

AWS Management Console



Risks from manual processes



Does not support repeatability at scale

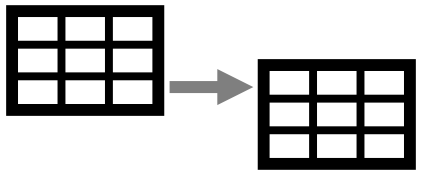
- How will you replicate deployments to multiple Regions?

No version control

- How will you roll back the production environment to a prior version?

Lack of audit trails

- How will you ensure compliance? How will you track changes to configuration details at the resource level?



Inconsistent data management

- For example, how will you ensure matching configurations across multiple Amazon Elastic Compute Cloud (Amazon EC2) instances?

AWS CLI

- A convenient way to interact with AWS from a terminal (Linux, macOS or Windows)
- Requires installation and configuration
  - CLI request authenticates the user by access key ID and secrete access key
  - A configuration process would store the information to be used by each request
- EC2 instance with Amazon AMI usually comes with the AWS CLI preinstalled and configured
- Example commands:
  - aws configure
  - aws s3 ls
  - aws ec2 describe-instances
  - aws cloudformation create-stack ..

CLI command examples

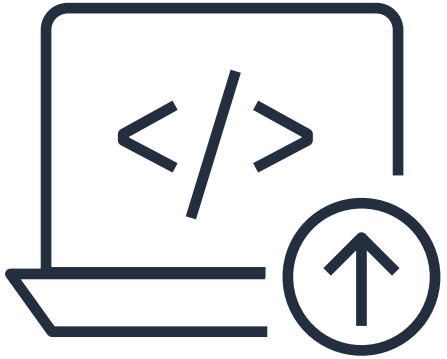
```
aws ec2 run-instances \
  --image-id ami-0ccedee93274bbb8d \
  --instance-type t2.micro \
  --count 1

aws s3api create-bucket --bucket abcd1234 --region us-east-1
```

Infrastructure as Code (IaC)

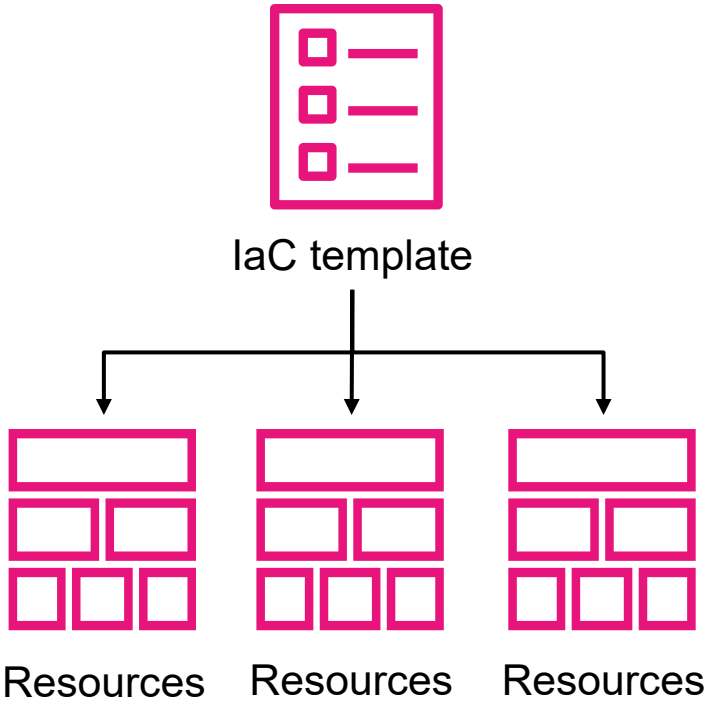
IaC overview

IaC is the process of writing a template that provisions and manages your cloud resources.

Human readable



Machine consumable



With IaC, you can replicate, redeploy, and repurpose infrastructure.

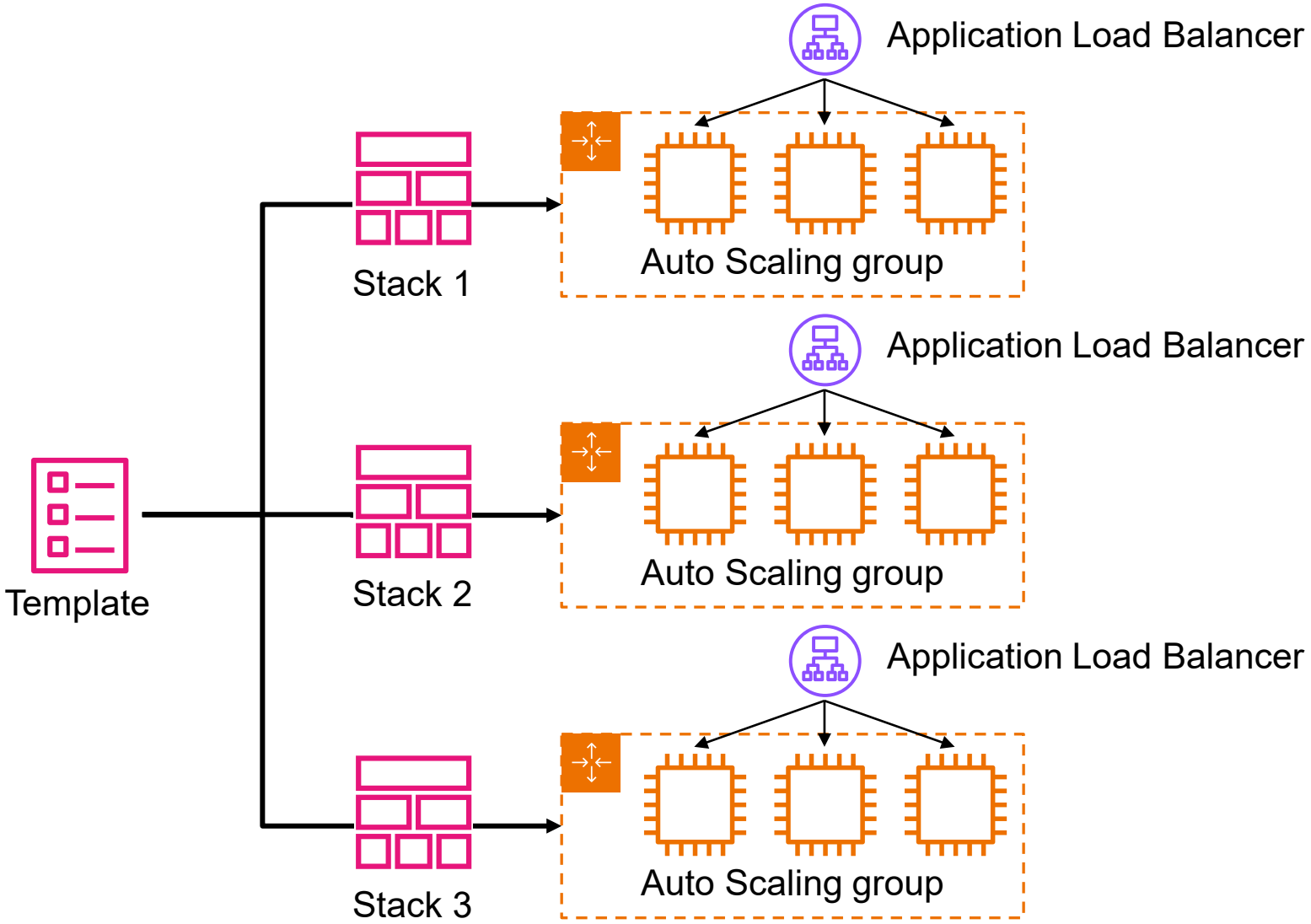
IaC benefits

Rapidly deploy complex environments with configuration consistency.

Propagate a change to all stacks by modifying the template.

Clean up by deleting the stack, which deletes the resources created.

The key benefits are reusability, repeatability, and maintainability.

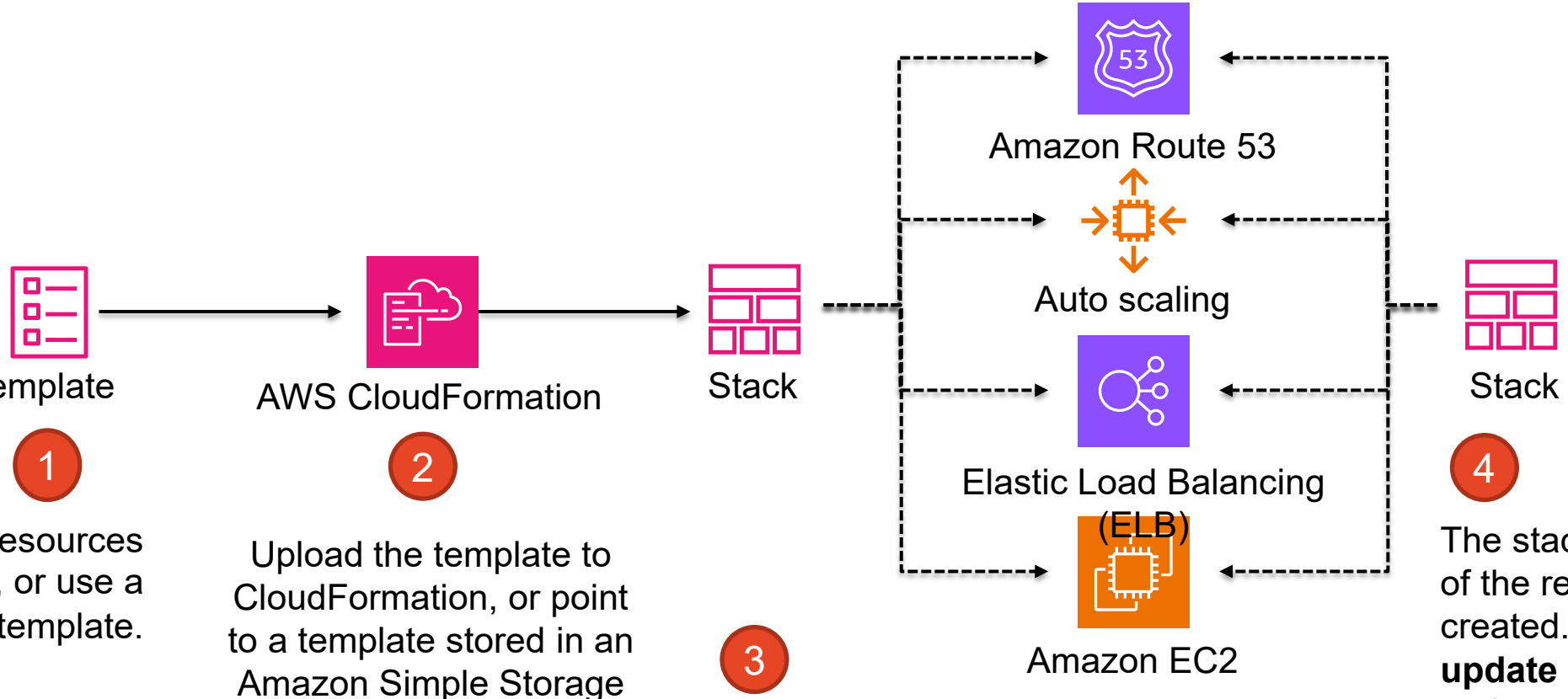


CloudFormation

CloudFormation

- Provides a simplified way to model, create, and manage a collection of AWS resources
  - A collection of resources is called a CloudFormation stack.
  - There is no extra charge (pay for only the resources that you create).
- Can create, update, and delete stacks
- Enables orderly and predictable provisioning and updating of resources
- Enables version control of AWS resource deployments

How CloudFormation works



1

Define your resources in a **template**, or use a prebuilt template.

2

Upload the template to CloudFormation, or point to a template stored in an Amazon Simple Storage Service (Amazon S3) bucket.
Or use aws cloudformation command

3

Run a **create stack** action. Resources are created across multiple services in your AWS account as a running environment.

4

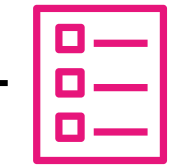
The stack retains control of the resources that are created. You can later **update stack**, **detect drift**, or **delete stack**.

Stacks

- A stack in AWS CloudFormation is a collection of AWS resources that can be managed as a single unit.
- Stacks are created based on CloudFormation templates
  - Think of template as a class and stacks as objects created based on that class
- There are different ways of grouping resources into a stack
  - Cohesion
  - Lifecycle
  - Security
  - Others
- A layer approach is typical: Network layer, Application Layer, Database Layer

CloudFormation Template

CloudFormation template syntax



Template

YAML example

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
 awsexamplebucket1:
   Type: AWS::S3::Bucket
```

JSON example

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Resources" : {
 “awsexamplebucket1" : {
 "Type" : "AWS::S3::Bucket"
 }
 }
}
```

YAML advantages

- Is optimized for readability
- Is less verbose
  - No brackets
  - quotes can be omitted most of the time
- Supports embedded comments

JSON advantages

- More widely used by other computer systems (for example, APIs)
- Usually less complex to generate and parse

Anatomy of a CloudFormation template

- When you write a CloudFormation template you have a choice of sections to include based on your workload.
- The only required section is **Resources**. Additional sections are optional.
- This example shows a YAML-formatted template fragment in the suggested order of sections.

```
---
AWSTemplateFormatVersion: "version date"
Description:
        String
Metadata:
        template metadata
Parameters:
        set of parameters
Rules:
        set of rules
Mappings:
        set of mappings
Conditions:
        set of conditions
Transform:
        set of transforms
Resources:
        set of resources
Outputs:
        set of outputs
```

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html

Resources

- The resources section contains multiple resources
- A resource has at least a name, a type, and some properties,
  - The property that can be included depends on the type

```
Resources:
  resourceName1:
    Type: AWS::ServiceName::ResourceType
     Properties:
        Propertyname1: PropertyValue1
        …
  resourceName2:
    Type: AWS::ServiceName::ReSourceType
    Properties:
        Propertyname1: PropertyValue1
        …
```

The name or logical ID of the resource is chosen by the user

The resource type is string with a given format

The properties can be included depends on the resource type

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resources-section-structure.html

Resources example: EC2 instance

```yaml
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance,
    Properties:
      ImageId: ami-9d23aeea
      InstanceType: m3.medium
      KeyName:
        Ref: KeyPair
```

```json
"Resources": {
   "Ec2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "ImageId": "ami-9d23aeea",
        "InstanceType": "m3.medium",
        "KeyName": {"Ref": "KeyPair "}
       }
   }
}
```

A name defined in other section

YAML format

JSON format

Parameters

- The Parameters section is optional but is a powerful way to customize the stacks created
  - Think of this as defining a parameterized constructor to allow you pass arguments to create different objects
  - Typically used to specify property values of stack resources

```
Parameters:
  ParameterLogicalID:
    Description: Information about the parameter
    Type: DataType
    Default: value
    AllowedValues:
      - value1
      - value2
```

String, Number, or an AWS-specific parameter type

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

Parameter and resources example

```yaml
Parameters:
  KeyPair:
    Description: SSH Key Pair
    Type: String
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance,
    Properties:
      ImageId: ami-9d23aeea
      InstanceType: m3.medium
      KeyName:
        Ref: KeyPair
```

```json
{
 "Parameters": {
    "KeyPair": {
       "Description": "SSH Key Pair",
       "Type": "String"}},
 "Resources": {
    "Ec2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
         "ImageId": "ami-9d23aeea",
         "InstanceType": "m3.medium",
         "KeyName": {"Ref": "KeyPair}
      }
    }
  }
```

We can also use **AWS::EC2::KeyPair::KeyName** as the type

Output

- It is an optional section declare the output values for the stack
  - Users define what data they want to pass as the output

```
Outputs:
  OutputLogicalID:
    Description:
    Value: Value to return
    Export:
      Name: Name of resource to export
```

Example intrinsic functions

```yaml
Parameters:
  KeyPair:
    Description: SSH Key Pair
    Type: String
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance,
    Properties:
      ImageId: ami-9d23aeea
      InstanceType: m3.medium
      KeyName:
        Ref: KeyPair
Outputs:
  ID:
      Value: !Ref Ec2Instance
      Description: ID of the EC2 instance created
  PublicName:
      Value: !GetAtt 'EC2Instance.PublicDnsName'
      Description: public DNS of the EC2
```

- Ref
  - Returns the value of a specified parameter, resource, or another intrinsic function.
  - commonly used to create references between resources within a CloudFormation template
- Fn::GetAtt (or !GetAtt for YAML)
  - Returns the value of an attribute from a resource.

Intrinsic Functions

- A list of built-in functions that can be used in the templates to assign values to properties that are not available until runtime.
- Basic Syntax
  - `Fn::`*`func_name inputs`*
  - Yaml short form: `!`*`func_name input(s)`*
- `Ref` is also an intrinsic function
  - `Ref` *`input`*
  - The returned value depends on the input specified
    - The value of the specified *parameter* or *resource*.
    - The output of a function

Output, parameter, resources example

```yaml
Parameters:
  KeyPair:
    Description: SSH Key Pair
    Type: String
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance,
    Properties:
      ImageId: ami-9d23aeea
      InstanceType: m3.medium
      KeyName:
        Ref: KeyPair
Outputs:
  ID:
      Value: !Ref Ec2Instance
      Description: ID of the EC2 instance created
  PublicName:
      Value: !GetAtt 'EC2Instance.PublicDnsName'
      Description: public DNS of the EC2
```

```json
{
 "Parameters": {
    "KeyPair": {
       "Description": "SSH Key Pair",
       "Type": "String"}},
 "Resources": {
    "Ec2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
         "ImageId": "ami-9d23aeea",
         "InstanceType": "m3.medium",
         "KeyName": {"Ref": "KeyPair}
        }
     }
  },
 "Outputs": {
    "InstanceId": {
        "Description": "InstanceId",
         "Value": {"Ref": "Ec2Instance"}
 }
```

CloudFormation template snippets

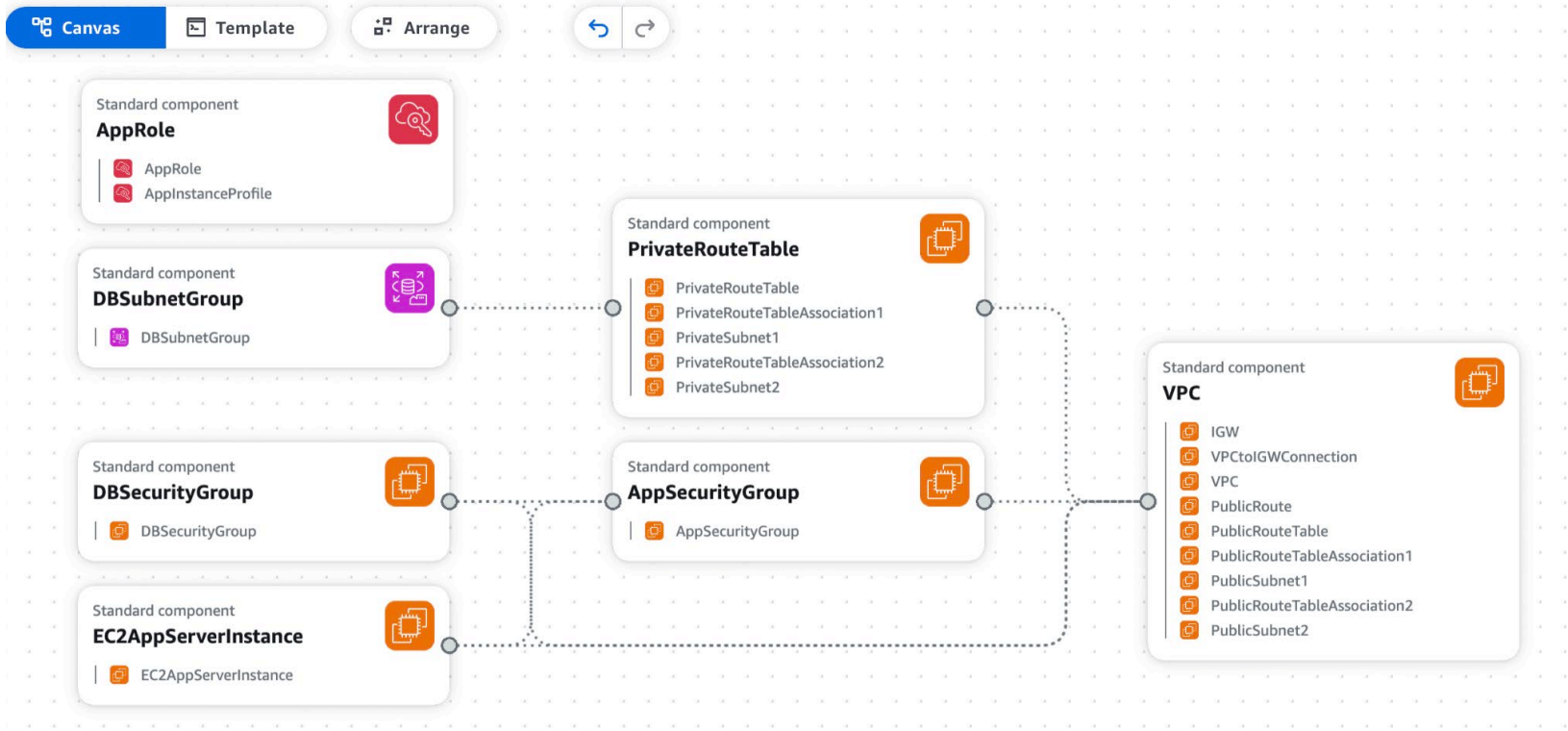
- AWS provide many snippets for various scenarios as starting point to create custom templates
  - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-snippets.html

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
```

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
    - Ref: MyDbSecurityByEC2SecurityGroup
    - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    MasterUsername: MyName
    ManageMasterUserPassword: true
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
  DeletionPolicy: Snapshot
```

Infrastructure Composer

- A visual way to work with CloudFormation template



Creating Stacks from AWS CLI

Creating Stack

```
aws cloudformation create-stack \
    --stack-name mybucket \
    --template-body file://my_bucket.yaml
```

```
// my_bucket.yaml
// creating S3 buket with default setting
AWSTemplateFormatVersion: "2010-09-09"
Description: This is my first bucket
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
```

Updating stack

```
aws cloudformation update-stack \
    --stack-name mybucket \
    --template-body file://my_bucket.yaml
```

```
// my_bucket.yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: This is my first bucket
Resources:
  MyBucket:
   Type:AWS::S3::Bucket
   Properties:
     AccessControl: PublicRead
```

Supplying parameters

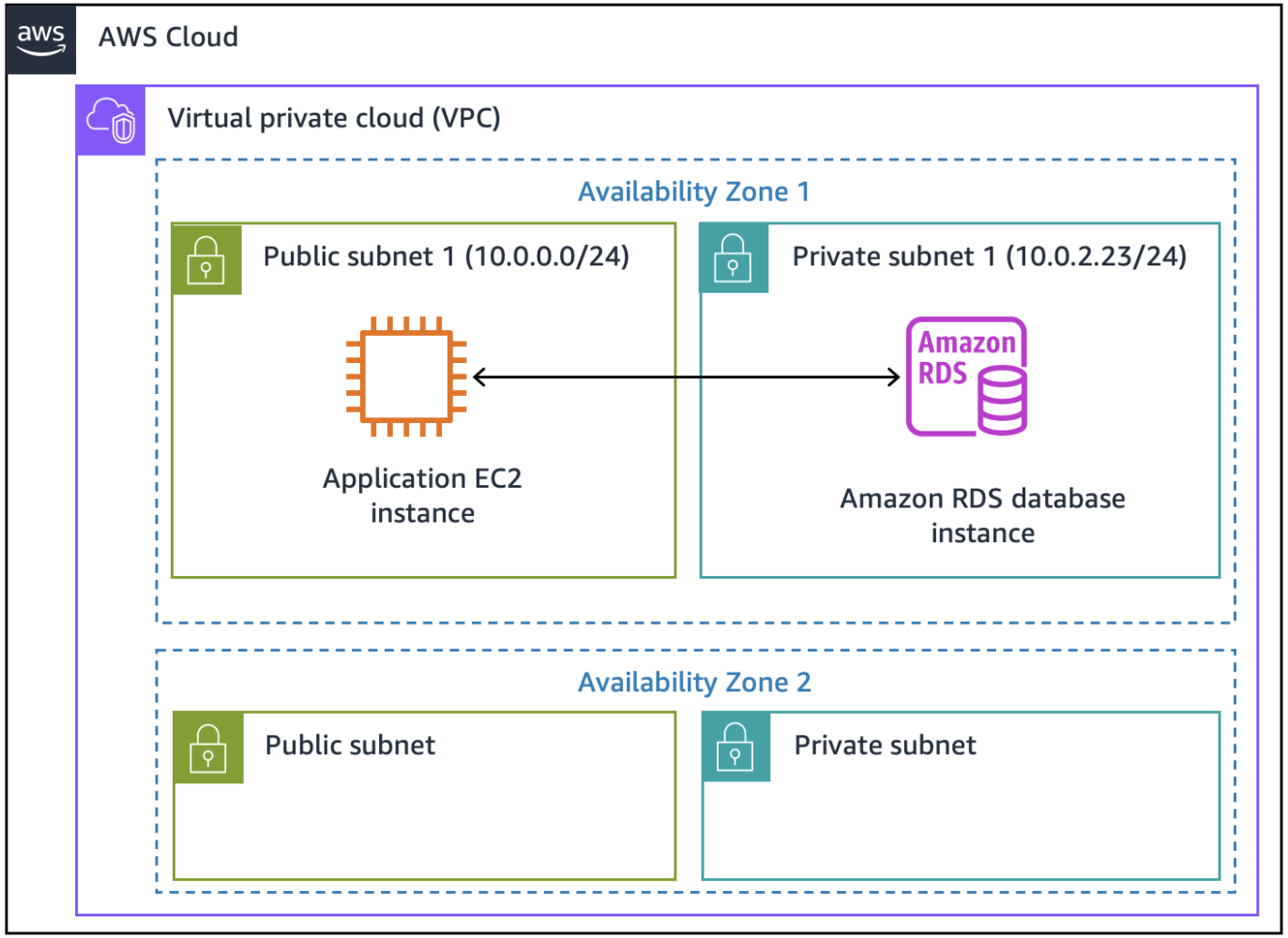
```
aws cloudformation create-stack \
    --stack-name my-s3-bucket-stack \
    --template-body file://s3-bucket.yaml \
    --parameters ParameterKey=BucketName,ParameterValue=my-unique-bucket-name
```

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Creates an S3 bucket with a user-provided name.

Parameters:
  BucketName:
    Type: String
    Description: The name of the S3 bucket to create.

Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref BucketName
```

CloudFormation Template Inspection

Module 6 Guided Lab: Creating an Amazon RDS Database

A template created most resources except the RDS



Resources created (1 of 2)

Resources (20)

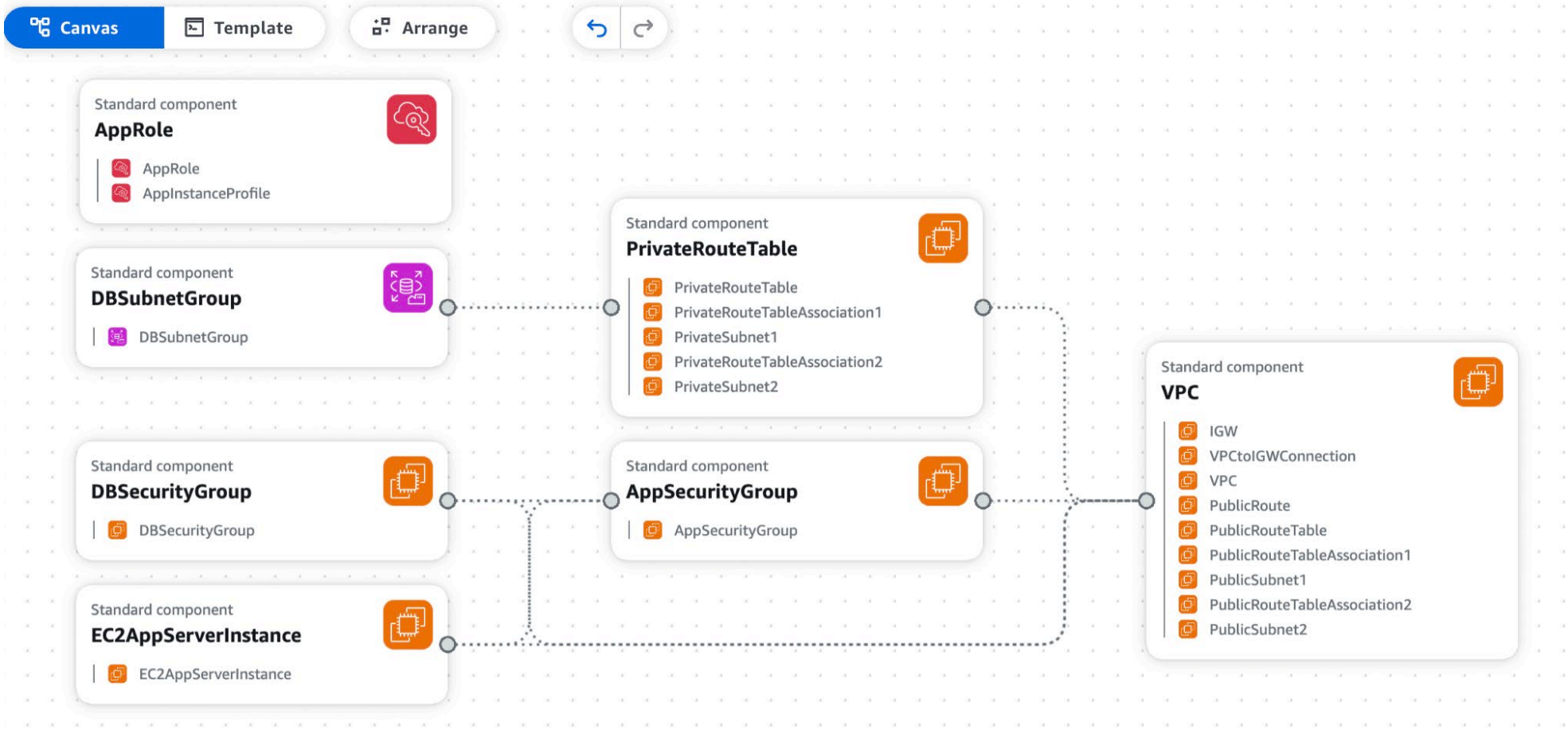
Search resources

Logical ID	Physical ID	Type
AppInstanceProfile	Inventory-App-Role	AWS::IAM::InstanceProfile
AppRole	Inventory-App-Role	AWS::IAM::Role
AppSecurityGroup	sg-06d1c3b6f67d70dec	AWS::EC2::SecurityGroup
DBSecurityGroup	sg-0da3a6026bc3d551c	AWS::EC2::SecurityGroup
DBSubnetGroup	lab-db-subnet-group	AWS::RDS::DBSubnetGroup
EC2AppServerInstance	i-063e5ff6407316ae3	AWS::EC2::Instance
IGW	igw-0a5b0ed5634c50c41	AWS::EC2::InternetGateway
PrivateRouteTable	rtb-015c53d5b3378bff8	AWS::EC2::RouteTable
PrivateRouteTableAssociation1	rtbassoc-015bf6d171b534823	AWS::EC2::SubnetRouteTableAssociation
PrivateRouteTableAssociation2	rtbassoc-017940f0cd265e4f4	AWS::EC2::SubnetRouteTableAssociation
PrivateSubnet1	subnet-02a1097bff3cae82c	AWS::EC2::Subnet
PrivateSubnet2	subnet-09e517a4e1be6d808	AWS::EC2::Subnet

Resources created (2 of 2)

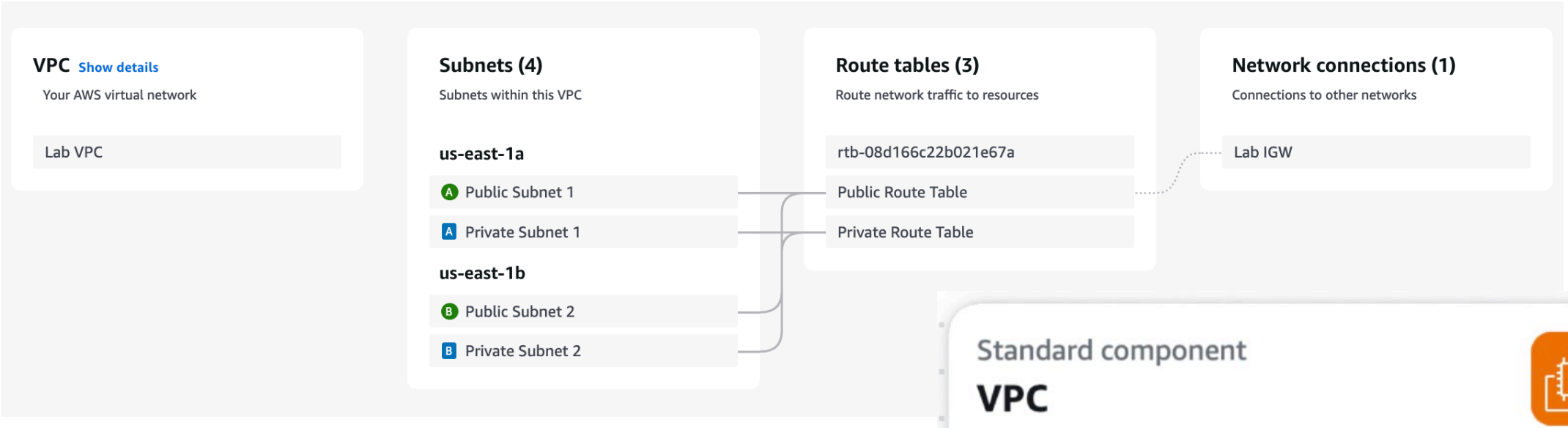
PublicRoute	rtb-0dfb7e297cd3309b8|0.0.0.0/0	AWS::EC2::Route
PublicRouteTable	rtb-0dfb7e297cd3309b8	AWS::EC2::RouteTable
PublicRouteTableAssociation1	rtbassoc-0746b9e3094f0fe72	AWS::EC2::SubnetRouteTableAssociation
PublicRouteTableAssociation2	rtbassoc-026a706210daf2e17	AWS::EC2::SubnetRouteTableAssociation
PublicSubnet1	subnet-06bc6d49352b441d6	AWS::EC2::Subnet
PublicSubnet2	subnet-053dbe33dc588c2d4	AWS::EC2::Subnet
VPC	vpc-0a77c8994f24e5d54	AWS::EC2::VPC
VPCtoIGWConnection	IGW|vpc-0a77c8994f24e5d54	AWS::EC2::VPCGatewayAttachment

The Infrastructure Composer view

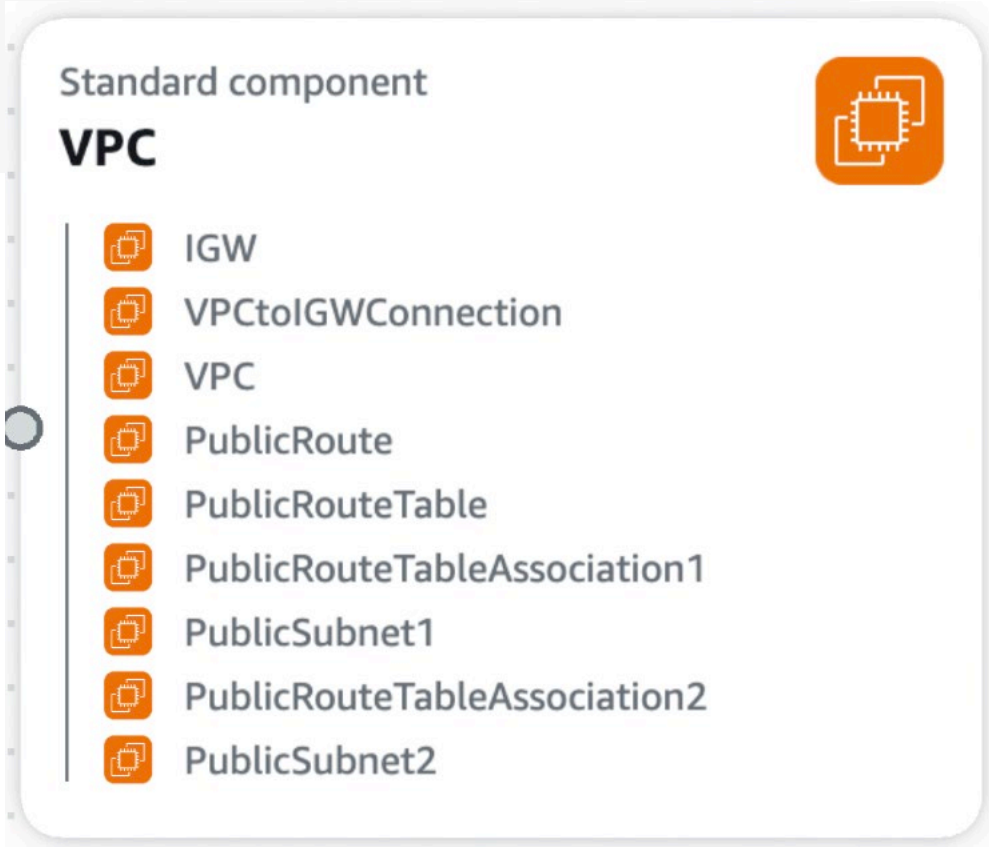


The Description and Parameters

```
AWSTemplateFormatVersion: 2010-09-09
Description: Lab template
# Lab VPC with 2 public + 2 private subnets (RDS requires 2 AZs) ‘
# DB Subnet Group across the 2 private subnets
# Role for EC2 instance to access RDS
Parameters:
  LatestAmiId:
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>’
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64’
  KeyName:
    Description: Name of an existing EC2 KeyPair
    Type: String
```



VPC resource map



Resources: VPC with Internet Gateway

DependsOn specifies creation order

Creation of this VPCtoIGWConnection resource should occur **after** the successful creation of the resources listed under it.

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-dependson.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpcgatewayattachment.html

```yaml
Resources:

###########
# VPC with Internet Gateway
###########

  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: Lab VPC

  IGW:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: Lab IGW

  VPCtoIGWConnection:
    Type: AWS::EC2::VPCGatewayAttachment
    DependsOn:
      - IGW
      - VPC
    Properties:
      InternetGatewayId: !Ref IGW
      VpcId: !Ref VPC
```

Resources: public route table

```yaml
###########
# Public Route Table
###########

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    DependsOn: VPC
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: Public Route Table

  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn:
      - PublicRouteTable
      - VPCtoIGWConnection
    Properties:
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref IGW
      RouteTableId: !Ref PublicRouteTable
```

rtb-0dfb7e297cd3309b8 / Public Route Table

Details Info

Route table ID	Main	Exp
rtb-0dfb7e297cd3309b8	No	2 s
VPC	**Owner ID**	
vpc-0a77c8994f24e5d54 | Lab VPC	258736218788	

Routes | Subnet associations | Edge associations | Route propagation | Tags

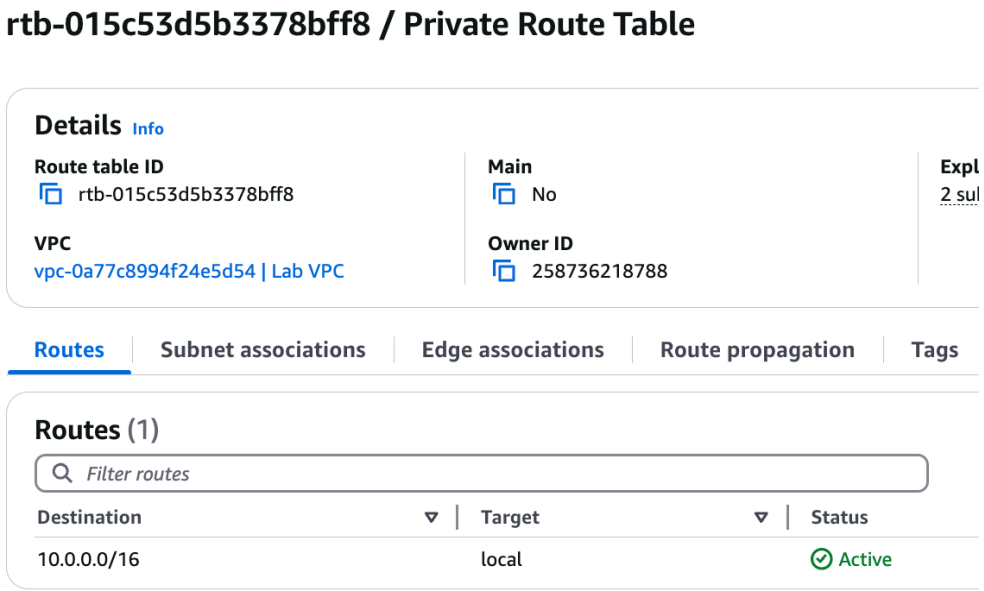
Routes (2)

Filter routes

Destination	Target	Status
0.0.0.0/0	igw-0a5b0ed5634c50c41	Active
10.0.0.0/16	local	Active

Resources: Private Route table

```yaml
###########
# Private Route Table
###########

  PrivateRouteTable:
    Type: AWS::EC2::RouteTable
    DependsOn: VPC
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: Private Route Table
```



Private route table only contains a default route, no route resource is needed

Resources: one public subnet (two in total)

```yaml
###########
# Public Subnets x 2
###########

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    DependsOn: VPC
    Properties:
      VpcId: !Ref VPC
      MapPublicIpOnLaunch: true
      CidrBlock: 10.0.0.0/24
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Public Subnet 1
```

Auto assignment of public IP

selects the first Availability Zone available in the current AWS Region.

```yaml
PublicRouteTableAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  DependsOn:
    - PublicRouteTable
    - PublicSubnet1
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1
```

Resources: one private subnet (two in total)

```yaml
###########
# Private Subnets x 2
###########

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    DependsOn: VPC
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.2.0/23
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Private Subnet 1
```

CIDR block

```yaml
PrivateRouteTableAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  DependsOn:
    - PrivateRouteTable
    - PrivateSubnet1
  Properties:
    RouteTableId: !Ref PrivateRouteTable
    SubnetId: !Ref PrivateSubnet1
```

Resources: DB Subnet group

```yaml
###########
# DB Subnet Group
###########

  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Lab-DB-Subnet-Group
      DBSubnetGroupName: Lab-DB-Subnet-Group
      SubnetIds:
        - !Ref PrivateSubnet1
        - !Ref PrivateSubnet2
      Tags:
        -
          Key: Name
          Value: DBSubnetGroup
```

This is a resource needed to create a DB instance

Resources: Security Group AppSG

```yaml
###########
# Security Group App-SG
###########

  AppSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: App-SG
      GroupName: App-SG
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: tcp
          FromPort: 80
          ToPort: 80
      VpcId: !Ref VPC
```

Inbound rules (1)

Manage tags | Edit inbound rules

Search

1

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
–	sgr-0b5d7819e15c47897	IPv4	HTTP	TCP	80	0.0.0.0/0

Resources: EC2

!Sub is a function for variable substitution, not very useful here

| means multiline string

```yaml
EC2AppServerInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref LatestAmiId
    IamInstanceProfile: Inventory-App-Role
    InstanceType: t2.micro
    KeyName: !Ref 'KeyName'
    SecurityGroupIds:
      - !Ref AppSecurityGroup
    SubnetId: !Ref PublicSubnet1
    UserData:
      Fn::Base64:
        !Sub |
          #!/bin/bash
          # Install Apache Web Server and PHP
          sudo dnf install -y httpd wget php-fpm php-mysqli php-json php
          sudo dnf install -y mariadb105-server
          # Download Lab files
          #sudo wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.c
          wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR
          sudo unzip inventory-app.zip -d /var/www/html/
          sudo wget https://docs.aws.amazon.com/aws-sdk-php/v3/download/a
          sudo unzip aws -d /var/www/html
          # Turn on web server
          sudo chkconfig httpd on
          sudo service httpd start
    Tags:
      - Key: Name
        Value: App Server
```

Resources: Outputs

```yaml
Outputs:

  AppServerPublicIP:
    Description: Public IP address of App server
    Value: !GetAtt
      - EC2AppServerInstance
      - PublicIp
```

Similar to
Value: !GetAtt 'EC2Instance.PublicIp'