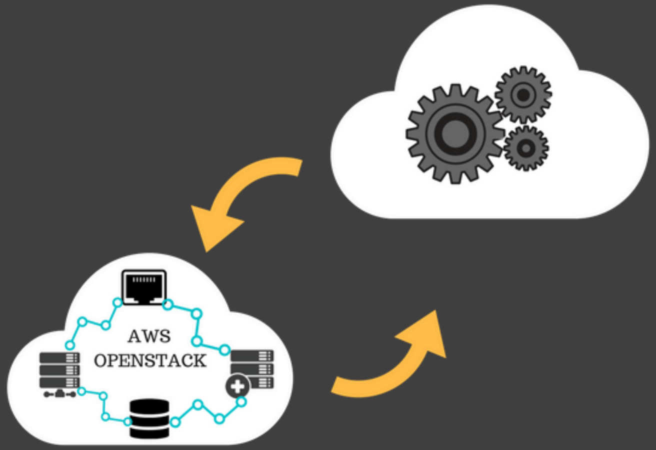
Course and Labs



AWS Design and Automation

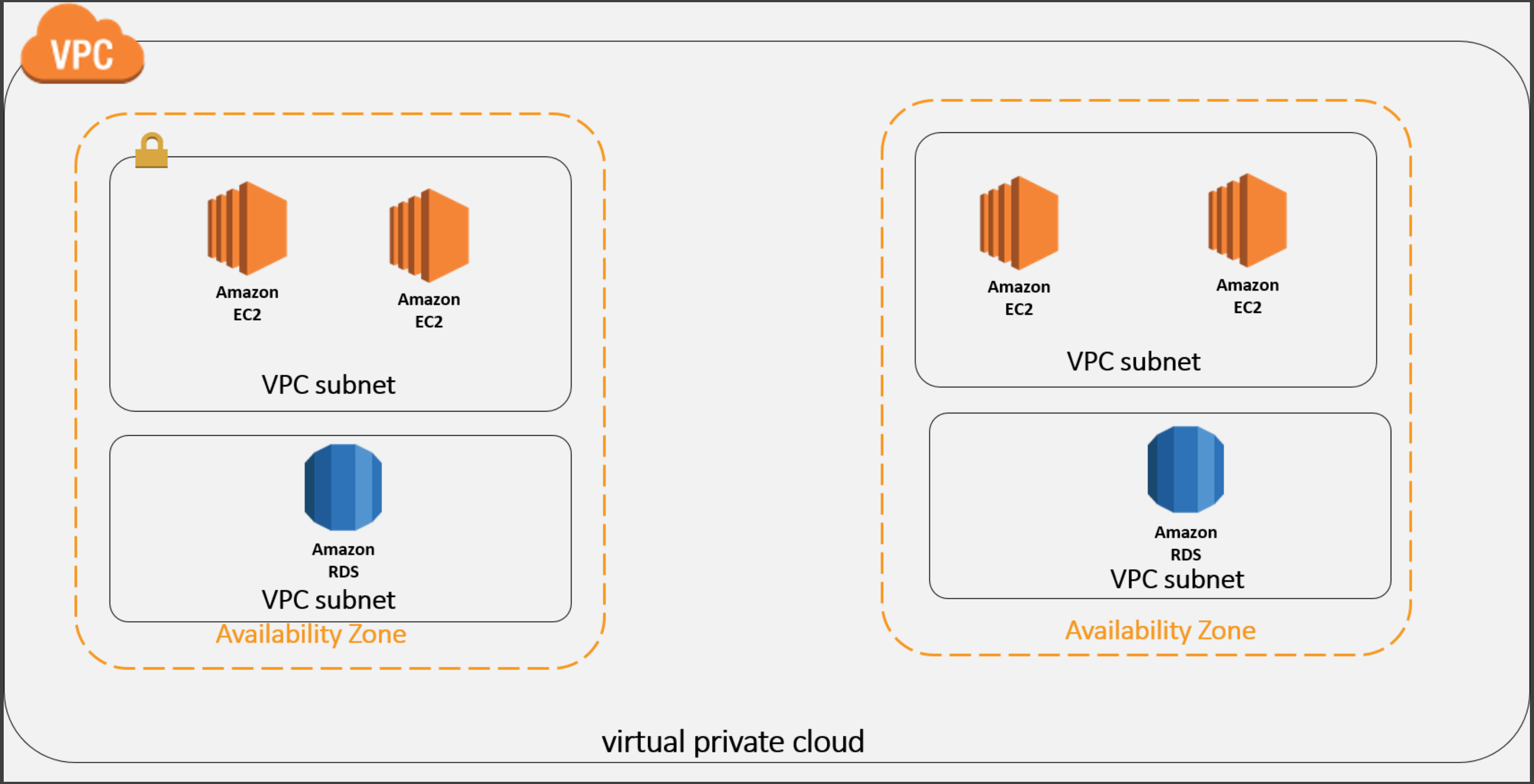
Module 2: Automate in AWS
Topic 1: Cloud Formation Overview

Mohanraj Shanmugam

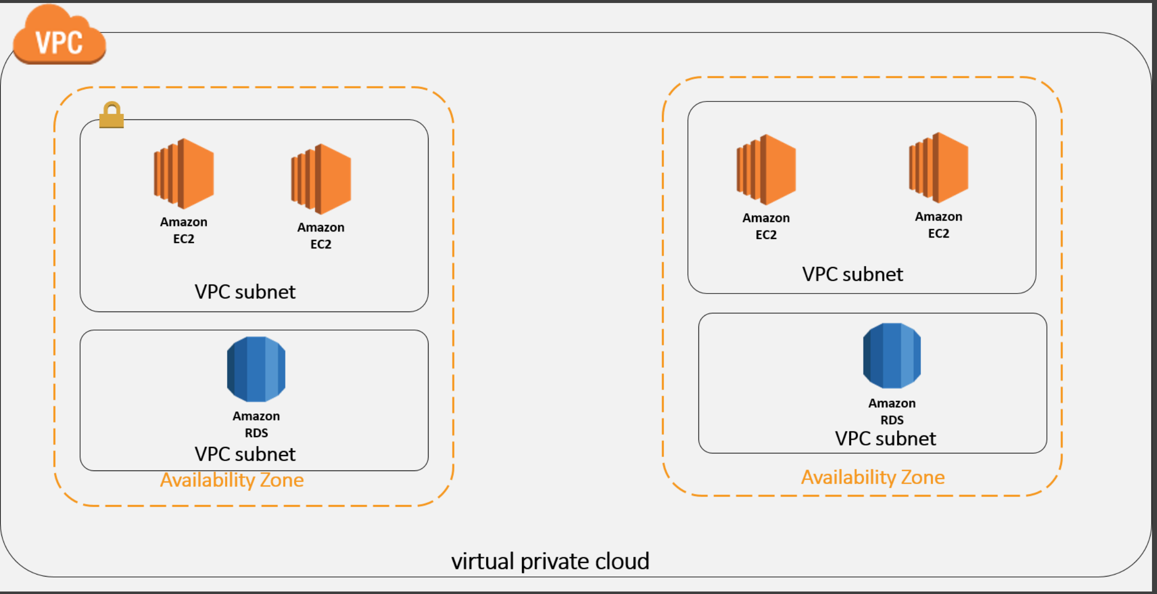
Course and Labs

AWS CloudFormation

Typical AWS Architecture

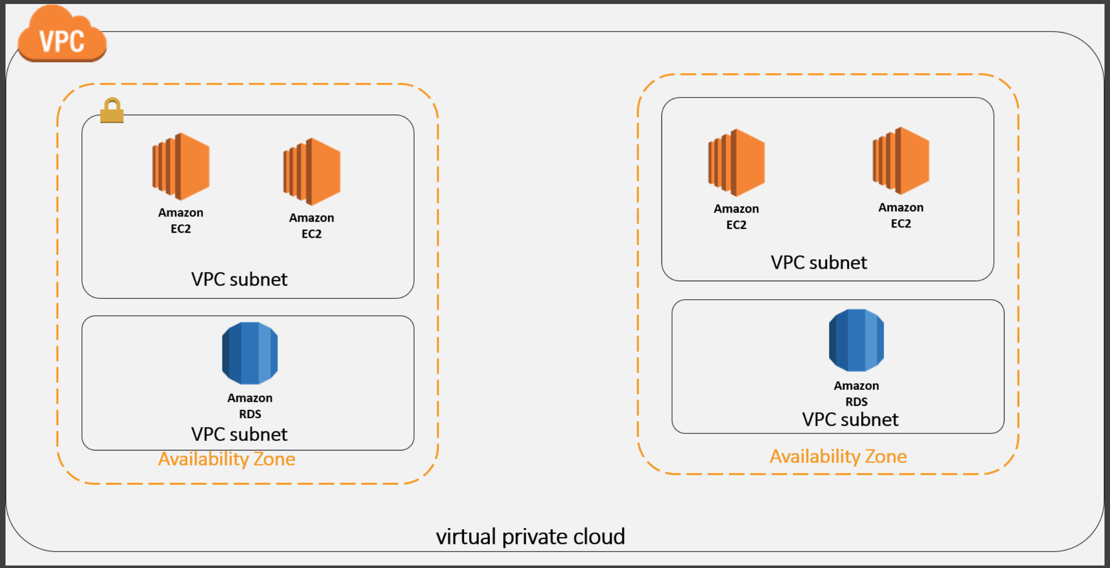


Cloud Formation Overview



- AWS CloudFormation gives developers and systems administrators an easy way to
  - Create/Provision,
  - Manage
  - Update
  - a collection of related AWS resources in an orderly and predictable fashion.

Cloud Formation Overview



- Create your own Blueprint to
  - describe the AWS resources
  - any associated dependencies or
  - runtime parameters required to run your application
- You don’t need to figure out the order for provisioning AWS services or the subtleties of making those dependencies work.
- CloudFormation takes care of this for you.

Concepts of Cloud Fromation

- Templates
  - Resource Definitions
  - Parameters

Course and Labs

AWS CloudFormation Templates

```
"myS3Bucket" : {
     "Type" : "AWS::S3::Bucket"
}
```

```
MyS3Bucket:
    Type: AWS::S3::Bucket
```

- **Templates**
  - An AWS CloudFormation template is a text file can be written in below Formats:
    - JSON
    - YAML
  - AWS CloudFormation uses these templates as blueprints for building your AWS resources.

Template Anatomy - JSON

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

Course and Labs

Template Anatomy - YAML

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Resources:
  set of resources

Outputs:
  set of outputs
```

Template Sections

- Templates include several major sections.
- The Resource section is the only required section.
- Some sections in a template can be in any order.
- However, as you build your template, it might be helpful to use the logical ordering as values in one section might refer to values from a previous section.

Course and Labs

Template Version Section

JSON

```
"AWSTemplateFormatVersion" :
"2010-09-09"
```

YAML

```
AWSTemplateFormatVersion:
"2010-09-09"
```

- Specifies the AWS CloudFormation template version that the template conforms to.
- The template format version is not the same as the API or WSDL version.
- If you don't specify a value, AWS CloudFormation assumes the latest template format version.

Course and Labs

Template Description Sections

JSON

```
"Description" : "Here are some details
about the template."
```

YAML

```
Description: >
Here are some details about the
template.
```

- The Description section (optional) enables you to include arbitrary comments about your template.
- The Description must follow the AWS Template Format Version section.
- The value for the description declaration must be a literal string that is between 0 and 1024 bytes in length.
- You cannot use a parameter or function to specify the description. The following snippet is an example of a description declaration:

Template Metadata Section

JSON

```
"Metadata" : {
        "Instances" : {"Description" : "Information about the
instances"},
        "Databases" : {"Description" : "Information about the
databases"} }
```

YAML

```
Metadata:
        Instances: Description: "Information about the instances"
        Databases: Description: "Information about the databases"
```

- You can use the optional metadata section that provide details about the Specific Resources.
- Some AWS CloudFormation features retrieve settings or configuration information that you define from the Metadata section.

Template Parameters Section

- Parameter section to pass values into your template when you create a stack. With parameters, you can create templates that are customized each time you create a stack.
- Each parameter must contain a value when you create a stack.
- You can specify a default value to make the parameter optional so that you don't need to pass in a value when creating a stack.

Template Parameters Section

JSON

```
"Parameters" : {
        "InstanceTypeParameter" : {
                "Type" : "String",
                "Default" : "t1.micro",
                "AllowedValues" : ["t1.micro", "m1.small", "m1.large"],
                "Description" : "Enter t1.micro, m1.small, or m1.large. Default is
t1.micro."
                }
        }
"Resources" : {
"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "InstanceType" : { "Ref" : "InstanceTypeParameter" },
    "ImageId" : "ami-2f726546"
  }
}
}
```

Template Parameters Section

```
YAML
Parameters:
  InstanceTypeParameter:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.
Resources:
   Ec2Instance:
     Type: AWS::EC2::Instance
     Properties:
       InstanceType:
         Ref: InstanceTypeParameter
       ImageId: ami-2f726546
```

Types

- Type (Mandatory) - The data type for the parameter (DataType).

String
A literal string.

For example, users could specify "MyUserName".

Number
An integer or float. AWS CloudFormation validates the parameter value as a number; however, when you use the parameter elsewhere in your template (for example, by using the Ref intrinsic function), the parameter value becomes a string.

For example, users could specify "8888".

Types

- Type (Mandatory) - The data type for the parameter (DataType).

List<Number>
An array of integers or floats that are separated by commas. AWS CloudFormation validates the parameter value as numbers; however, when you use the parameter elsewhere in your template (for example, by using the Ref intrinsic function), the parameter value becomes a list of strings.

For example, users could specify "80,20", and a Ref will result in ["80","20"].

CommaDelimitedList
An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas. Also, each member string is space trimmed.

For example, users could specify "test,dev,prod", and a Ref will result in ["test","dev","prod"].

Types

- **AWS-Specific Parameter Types:** specify existing AWS values that are in their account

Network Parameter Types

AWS::EC2::VPC::Id
A VPC ID, such as vpc-a123baa3.

List<AWS::EC2::VPC::Id>
An array of VPC IDs, such as vpc-a123baa3, vpc-b456baa3.

AWS::EC2::Subnet::Id
A subnet ID, such as subnet-123a351e.

List<AWS::EC2::Subnet::Id>
An array of subnet IDs, such as subnet-123a351e, subnet-456b351e.

Types

- **AWS-Specific Parameter Types:** specify existing AWS values that are in their account

Network Parameter Types

AWS::EC2::SecurityGroup::GroupName
An EC2-Classic or default VPC security group name, such as my-sg-abc.

List<AWS::EC2::SecurityGroup::GroupName>
An array of EC2-Classic or default VPC security group names, such as my-sg-abc, my-sg-def.

AWS::EC2::SecurityGroup::Id
A security group ID, such as sg-a123fd85.

List<AWS::EC2::SecurityGroup::Id>
An array of security group IDs, such as sg-a123fd85, sg-b456fd85.

Types

- **AWS-Specific Parameter Types:** specify existing AWS values that are in their account

Network Parameter Types

AWS::Route53::HostedZone::Id
An Amazon Route 53 hosted zone ID, such as Z23YXV4OVPL04A.

List<AWS::Route53::HostedZone::Id>
An array of Amazon Route 53 hosted zone IDs, such as Z23YXV4OVPL04A, Z23YXV4OVPL04B.

Types

- **AWS-Specific Parameter Types:** specify existing AWS values that are in their account

EC2 Parameter Types

AWS::EC2::AvailabilityZone::Name
An Availability Zone, such as us-west-2a.

List<AWS::EC2::AvailabilityZone::Name>
An array of Availability Zones for a region, such as us-west-2a, us-west-2b.

AWS::EC2::Image::Id
An Amazon EC2 image ID, such as ami-ff527ecf. Note that the AWS CloudFormation console won't show a drop-down list of values for this parameter type.

List<AWS::EC2::Image::Id>
An array of Amazon EC2 image IDs, such as ami-ff527ecf, ami-e7527ed7. Note that the AWS CloudFormation console won't show a drop-down list of values for this parameter type.

Course and Labs

Types

- **AWS-Specific Parameter Types:** specify existing AWS values that are in their account

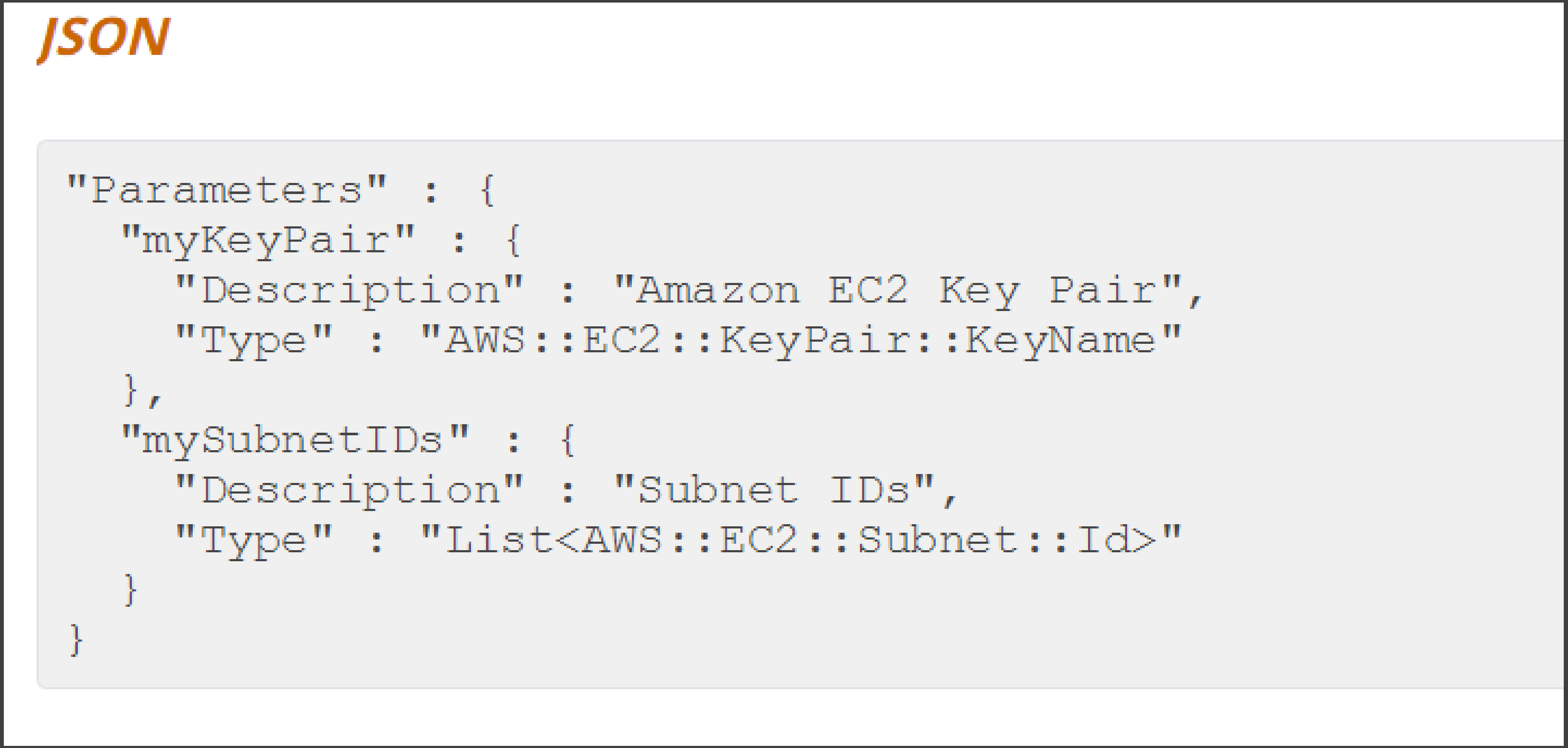
EC2 Parameter Types

AWS::EC2::Instance::Id
An Amazon EC2 instance ID, such as i-1e731a32.

List<AWS::EC2::Instance::Id>
An array of Amazon EC2 instance IDs, such as i-1e731a32, i-1e731a34.

AWS::EC2::KeyPair::KeyName
An Amazon EC2 key pair name.

AWS Specific Property



JSON

```
"Parameters" : {
  "myKeyPair" : {
    "Description" : "Amazon EC2 Key Pair",
    "Type" : "AWS::EC2::KeyPair::KeyName"
  },
  "mySubnetIDs" : {
    "Description" : "Subnet IDs",
    "Type" : "List<AWS::EC2::Subnet::Id>"
  }
}
```

Parameter String Constraints

- AllowedPattern
- A regular expression that represents the patterns you want to allow for String types.
- Example to allow the alpha Numerical "^[a-zA-Z0-9]*$"

- AllowedValues
- An array containing the list of values allowed for the parameter.
- Example: "AllowedValues" : ["t2.micro", "m1.small", "m1.large"]

Parameter String Constraints

- MaxLength
- An integer value that determines the largest number of characters you want to allow for String types.

- MinLength
- An integer value that determines the smallest number of characters you want to allow for String types.

- NoEcho
- Whether to mask the parameter value whenever anyone makes a call that describes the stack. If you set the value to true, the parameter value is masked with asterisks (*****).

Parameter Numeric Constraints

- MaxValue

A numeric value that determines the largest numeric value you want to allow for Number types.

- MinValue

A numeric value that determines the smallest numeric value you want to allow for Number

Parameter Numeric Constraints

- ConstraintDescription
- A string that explains the constraint when the constraint is violated. For example, without a constraint description, a parameter that has an allowed pattern of [A-Za-z0-9]+ displays the following error message when the user specifies an invalid value:

- Example : Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
- By adding a constraint description, such as must only contain upper- and lowercase letters, and numbers, you can display a customized error message:

A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value

Parameter Numeric Constraints

- Default
  - A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value that adheres to those constraints.
- Description
  - A string of up to 4000 characters that describes the parameter

AWS Specific Property

```
"Parameters" : {
 "DBPort" : {
  "Default" : "3306",
  "Description" : "TCP/IP port for the database",
  "Type" : "Number",
  "MinValue" : "1150",
  "MaxValue" : "65535"
 },
 "DBPwd" : {
  "NoEcho" : "true",
  "Description" : "The database admin account password",
  "Type" : "String",
  "MinLength" : "1",
  "MaxLength" : "41",
  "AllowedPattern" : "^[a-zA-Z0-9]*$"
 }
}
```

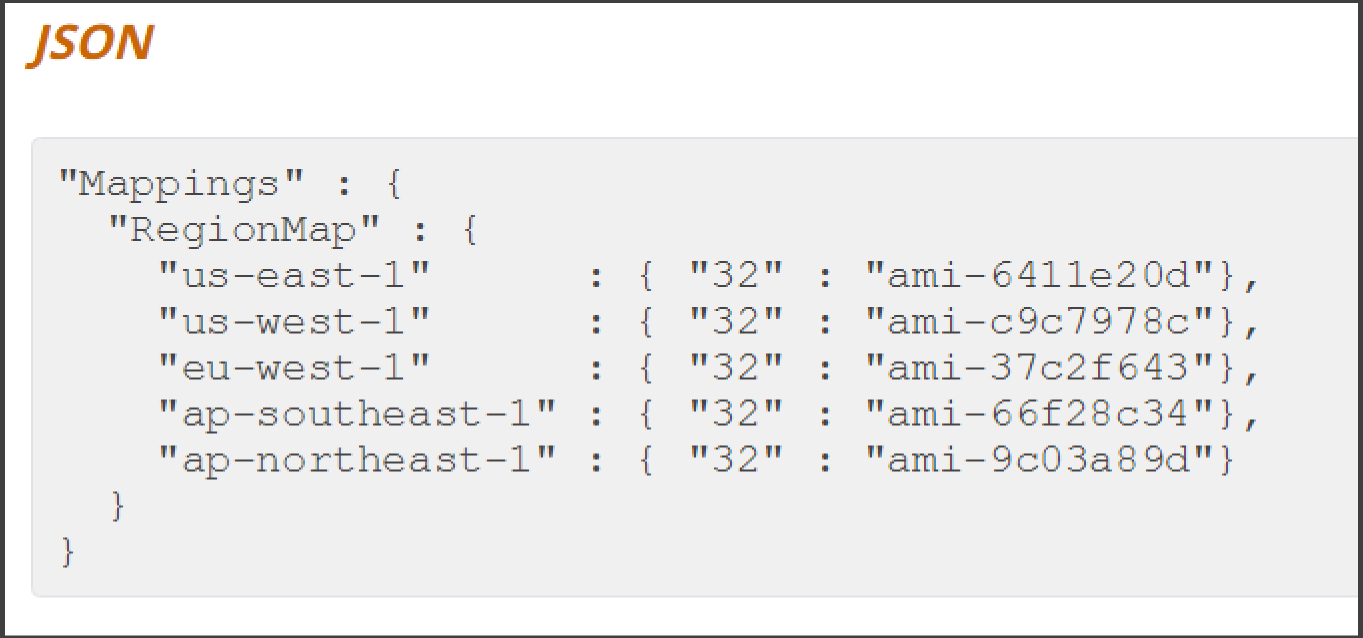
Course and Labs

AWS Specific Property

```
"Parameters" : {
 "myKeyPair" : {
   "Description" : "Amazon EC2 Key Pair",
   "Type" : "AWS::EC2::KeyPair::KeyName"
 },
 "mySubnetIDs" : {
   "Description" : "Subnet IDs",
   "Type" : "List<AWS::EC2::Subnet::Id>"
 }
}
```

Mappings

- Mappings section matches a key to a corresponding set of named values.
  - For example, if you want to set values based on a region, you can create a mapping that uses the region name as a key and contains the values you want to specify for each specific region.



JSON

```
"Mappings" : {
  "RegionMap" : {
    "us-east-1"       : { "32" : "ami-6411e20d"},
    "us-west-1"       : { "32" : "ami-c9c7978c"},
    "eu-west-1"       : { "32" : "ami-37c2f643"},
    "ap-southeast-1" : { "32" : "ami-66f28c34"},
    "ap-northeast-1" : { "32" : "ami-9c03a89d"}
  }
}
```

Multiple values

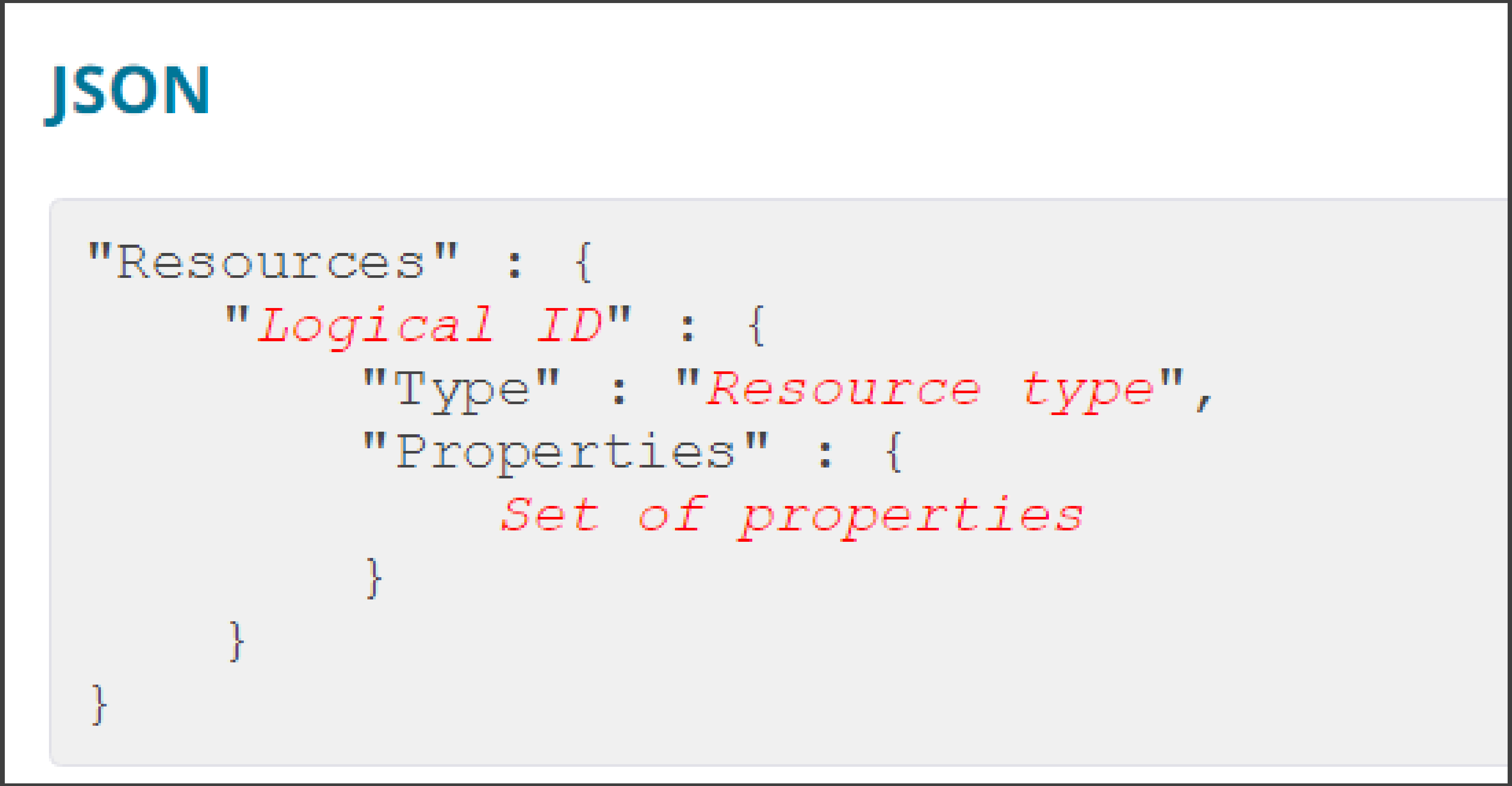
JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" },
      "ap-southeast-1" : { "32" : "ami-66f28c34", "64" : "ami-60f28c32" },
      "ap-northeast-1" : { "32" : "ami-9c03a89d", "64" : "ami-a003a8a1" }
    }
  },

  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "32"]},
        "InstanceType" : "m1.small"
      }
    }
  }
}
```

Resource Section

Course and Labs

- Resource Section declare the AWS resources that you want as part of your stack, such as an Amazon EC2 instance or an Amazon S3 bucket.



JSON

```
"Resources" : {
    "Logical ID" : {
        "Type" : "Resource type",
        "Properties" : {
            Set of properties
        }
    }
}
```

AWS Resource Types Reference

- http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html

Intrinsic Functions

- http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html

Course and Labs

Conditions Section

- Condition section includes statements that define when a resource is created or when a property is defined.
- For example, you can compare whether a value is equal to another value.
- Based on the result of that condition, you can conditionally create resources.

Conditions Section

- You can use the following intrinsic functions to define conditions:
  - Fn::And
  - Fn::Equals
  - Fn::If
  - Fn::Not
  - Fn::Or

How to Use Conditions Overview

- Parameter Section
  - Define the input values that you want to evaluate in your conditions. Conditions will result in true or false based on values from these input parameter.
- Condition Section
  - Define conditions by using the intrinsic condition functions. These conditions determine when AWS CloudFormation creates the associated resources.
- Resource Section
  - Associate conditions with the resources or outputs that you want to conditionally create. AWS CloudFormation creates entities that are associated with a true condition and ignores entities that are associated with a false condition.

Conditions Example

Course and Labs

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Mappings" : {
    "RegionMap" : {
      "us-east-1"      : { "AMI" : "ami-7f418316", "TestAz" : "us-east-1a" },
      "us-west-1"      : { "AMI" : "ami-951945d0", "TestAz" : "us-west-1a" },
      "us-west-2"      : { "AMI" : "ami-16fd7026", "TestAz" : "us-west-2a" },
      "eu-west-1"      : { "AMI" : "ami-24506250", "TestAz" : "eu-west-1a" },
      "sa-east-1"      : { "AMI" : "ami-3e3be423", "TestAz" : "sa-east-1a" },
      "ap-southeast-1" : { "AMI" : "ami-74dda626", "TestAz" : "ap-southeast-1a" },
      "ap-southeast-2" : { "AMI" : "ami-b3990e89", "TestAz" : "ap-southeast-2a" },
      "ap-northeast-1" : { "AMI" : "ami-dcfa4edd", "TestAz" : "ap-northeast-1a" }
    }
  },

  "Parameters" : {
    "EnvType" : {
      "Description" : "Environment type.",
      "Default" : "test",
      "Type" : "String",
      "AllowedValues" : ["prod", "test"],
      "ConstraintDescription" : "must specify prod or test."
    }
  },
```

Conditions Example

```
"Conditions" : {
  "CreateProdResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "prod"]}
},

"Resources" : {
  "EC2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]}
    }
  },

  "MountPoint" : {
    "Type" : "AWS::EC2::VolumeAttachment",
    "Condition" : "CreateProdResources",
    "Properties" : {
      "InstanceId" : { "Ref" : "EC2Instance" },
      "VolumeId"  : { "Ref" : "NewVolume" },
      "Device" : "/dev/sdh"
    }
  },

  "NewVolume" : {
    "Type" : "AWS::EC2::Volume",
    "Condition" : "CreateProdResources",
    "Properties" : {
      "Size" : "100",
      "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ]}
    }
  }
},
```

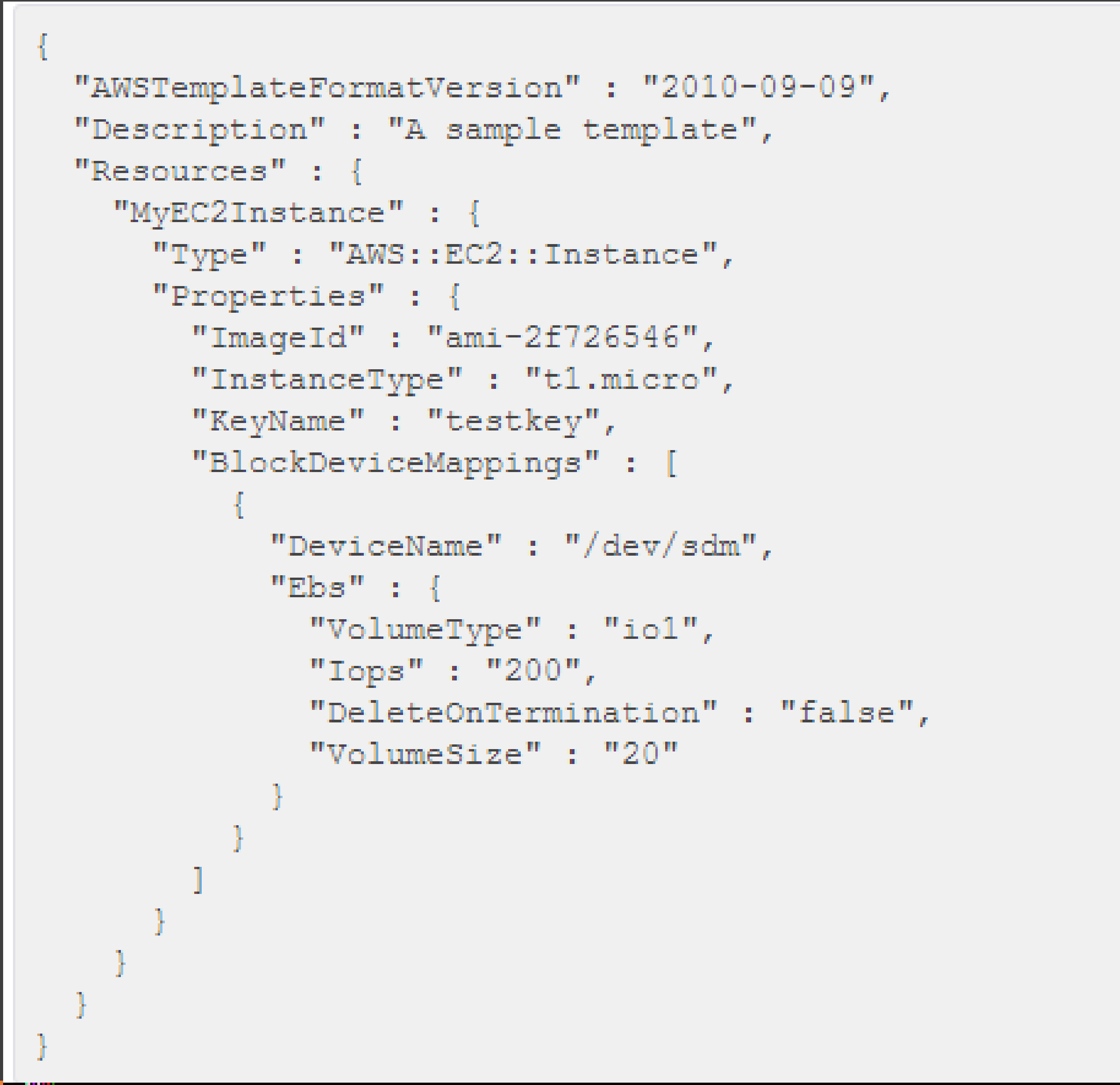
Output Section

```
"Outputs" : {
  "BackupLoadBalancerDNSName" : {
    "Description": "The DNSName of the backup load balancer",
    "Value" : { "Fn::GetAtt" : [ "BackupLoadBalancer",
"DNSName" ]},
    "Condition" : "CreateProdResources"
  },
  "InstanceID" : {
    "Description": "The Instance ID",
    "Value" : { "Ref" : "EC2Instance" }
  }
}
```

- The optional Outputs section declares output values that:
  - Return in response (to describe stack calls)
  - View on the AWS CloudFormation console.
  - Import into other stacks
- For example, you can output the S3 bucket name for a stack to make the bucket easier to find.

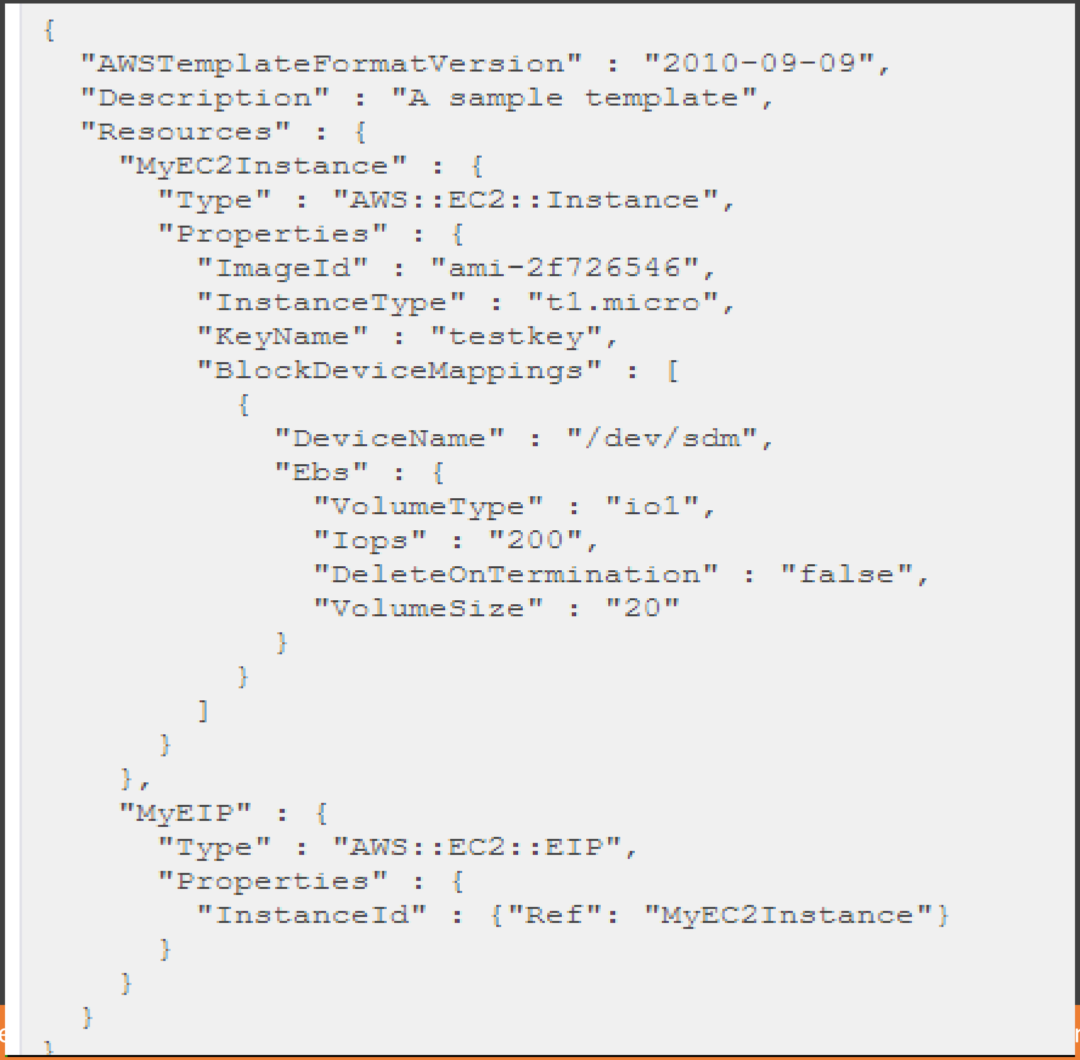
Templates

- For example, if you created a stack with the following template, AWS CloudFormation provisions an instance with an `ami-2f726546` AMI ID, `t1.micro` instance type, `testkey`key pair name, and an Amazon EBS volume.



```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample template",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-2f726546",
        "InstanceType" : "t1.micro",
        "KeyName" : "testkey",
        "BlockDeviceMappings" : [
          {
            "DeviceName" : "/dev/sdm",
            "Ebs" : {
              "VolumeType" : "io1",
              "Iops" : "200",
              "DeleteOnTermination" : "false",
              "VolumeSize" : "20"
            }
          }
        ]
      }
    }
  }
}
```

AWS CloudFormation Templates



```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample template",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-2f726546",
        "InstanceType" : "t1.micro",
        "KeyName" : "testkey",
        "BlockDeviceMappings" : [
          {
            "DeviceName" : "/dev/sdm",
            "Ebs" : {
              "VolumeType" : "io1",
              "Iops" : "200",
              "DeleteOnTermination" : "false",
              "VolumeSize" : "20"
            }
          }
        ]
      }
    },
    "MyEIP" : {
      "Type" : "AWS::EC2::EIP",
      "Properties" : {
        "InstanceId" : {"Ref": "MyEC2Instance"}
      }
    }
  }
}
```

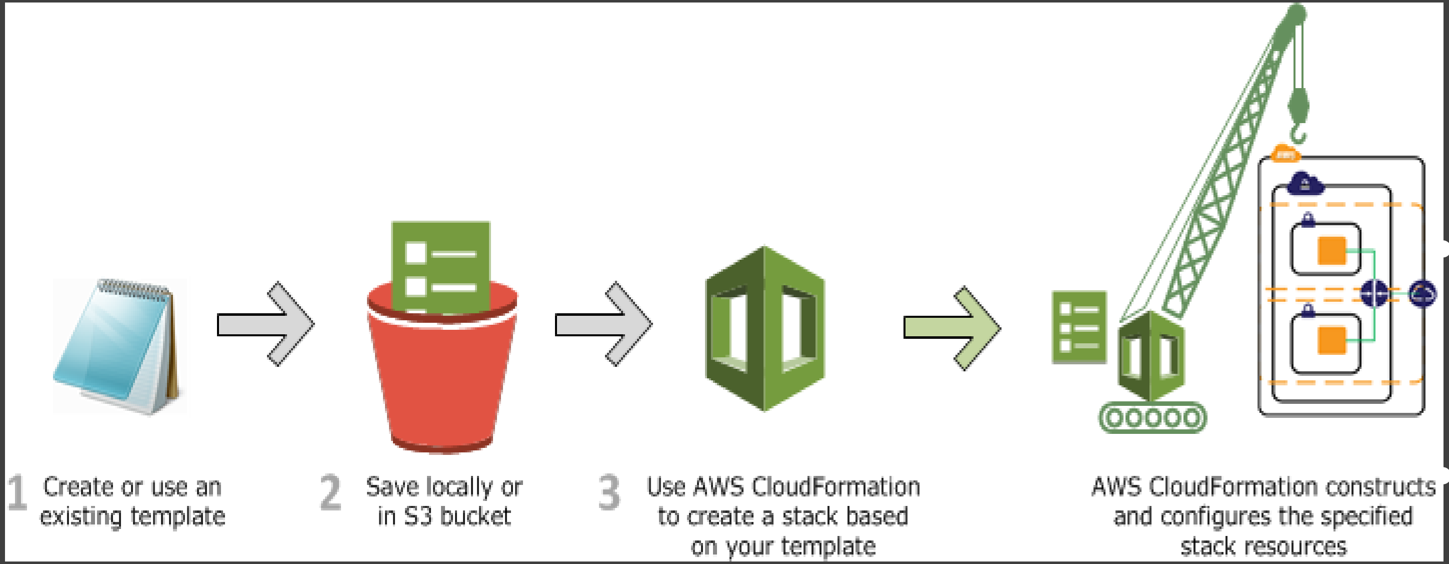
- AWS CloudFormation templates have additional capabilities that you can use to build complex sets of resources and reuse those templates in multiple contexts.
- you can specify a value like the instance type when you create a stack instead of when you create the template, making the template easier to reuse in different situations.

AWS CloudFormation Stacks

- **Stacks**
  - When you use AWS CloudFormation, you manage related resources as a single unit called a stack.
  - you create, update, and delete a collection of resources by creating, updating, and deleting stacks.
  - All the resources in a stack are defined by the stack's AWS CloudFormation template.
  - Suppose you created a template that includes an Auto Scaling group, Elastic Load Balancing load balancer, and an Amazon RDS database instance
  - To create those resources, you create a stack by submitting the template that you created, and AWS CloudFormation provisions all those resources for you.
  - To update resources, you first modify the original stack template and then update your stack by submitting the modified template.

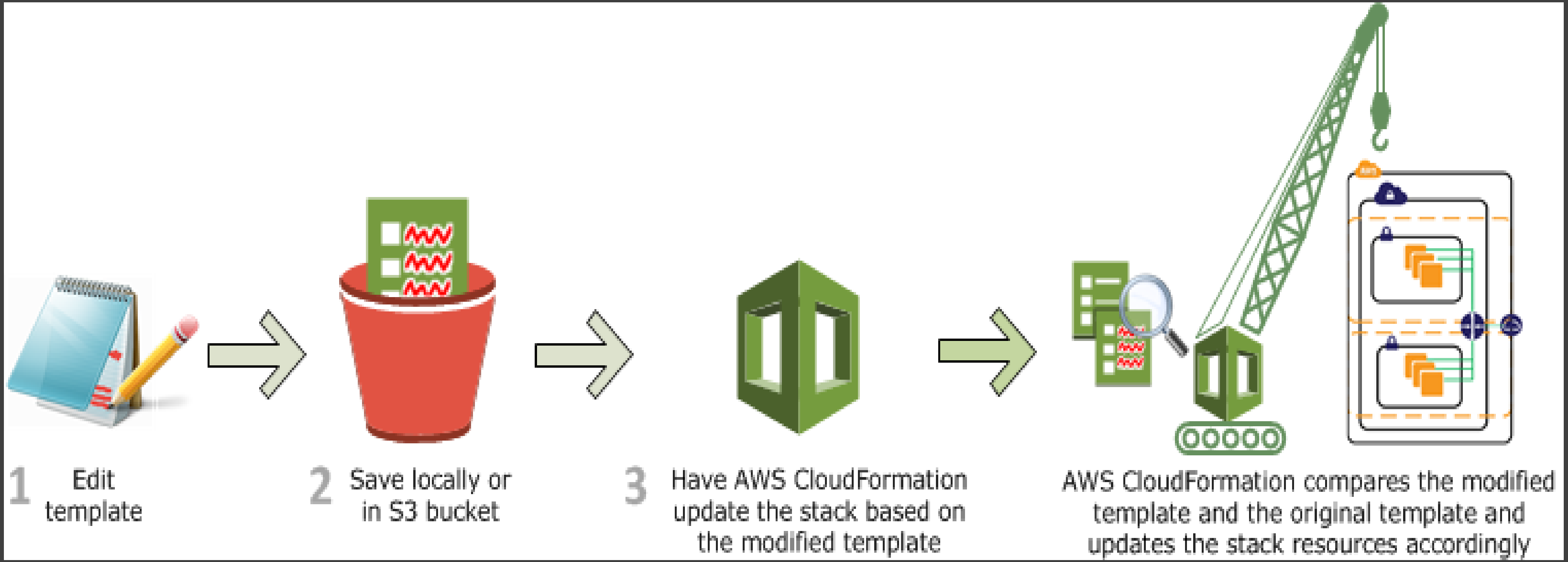
How Does AWS CloudFormation Work?

create stack workflow:


1 Create or use an existing template
2 Save locally or in S3 bucket
3 Use AWS CloudFormation to create a stack based on your template
AWS CloudFormation constructs and configures the specified stack resources

- Whenever you create a stack, AWS CloudFormation makes underlying service calls to AWS to provision and configure your resources.
- Note that AWS CloudFormation can only perform actions that you have permission to do.
- For example, to create Amazon EC2 instances by using AWS CloudFormation, you need permissions to create instances.

Update Stack Workflow


1 Edit template
2 Save locally or in S3 bucket
3 Have AWS CloudFormation update the stack based on the modified template
AWS CloudFormation compares the modified template and the original template and updates the stack resources accordingly

Delete Stack

- When you delete a stack, you specify the stack to delete, and AWS CloudFormation deletes the stack and all the resources in that stack.
- If you want to delete a stack but want to retain some resources in that stack, you can use adeletion policy to retain those resources.
- After all the resources have been deleted, AWS CloudFormation signals that your stack has been successfully deleted. If AWS CloudFormation cannot delete a resource, the stack will not be deleted. Any resources that haven't been deleted will remain until you can successfully delete the stack.

Course and Labs

Features and Benefits

- Supports a Wide Range of AWS Resources
  - AWS CloudFormation supports a wide range of AWS resources, allowing you to build a highly available, reliable, and scalable AWS infrastructure for your application needs.
- Easy to Use
  - CloudFormation makes it easy to organize and deploy a collection of AWS resources and lets you describe any dependencies or special parameters to pass in at runtime. You can use one of the many CloudFormation sample templates as a starting point.

Features and Benefits

- Declarative and Flexible
  - To create the infrastructure you want, you enumerate what AWS resources, configuration values, and interconnections you need in a template and then let AWS CloudFormation do the rest with a few simple clicks in the AWS Management Console, one command by using the AWS command line interface, or a single requests by calling the APIs.
- Infrastructure as Code
  - A template can be used repeatedly to create identical copies of the same stack (or to use as a foundation to start a new stack).
  - You can capture and control region-specific infrastructure variations such as Amazon EC2 AMIs, as well as Amazon EBS and Amazon RDS snapshot names.
  - Templates are simple JSON-formatted text files that can be placed under your normal source control mechanisms, stored in private or public locations such as Amazon S3, and exchanged via email.
  - With AWS CloudFormation, you can "open the hood," to see exactly which AWS resources make up a stack. You retain full control and have the ability to modify any of the AWS resources created as part of a stack.

Features and Benefits

- Visualize and Edit with Drag-and-Drop Interface
  - AWS Cloud Formation Designer provides a visual diagram of your template with icons representing your AWS resources and arrows showing their relationships.
  - You can build and edit templates using the drag-and-drop interface, then edit the template details using the integrated JSON text editor.
  - CloudFormation Designer allows you to spend more time designing your AWS infrastructure and less time manually coding your templates.

Features and Benefits

- Integration Ready
  - You can integrate AWS CloudFormation with the development and management tools of your choice.
  - AWS CloudFormation publishes progress events through the Amazon Simple Notification Service (SNS). With SNS, you can track stack creation and deletion progress via email and integrate with other processes programmatically.

Course and Labs

Features and Benefits

- Customized via Parameters
  - You can use parameters to customize aspects of your template at run time, when the stack is built.
  - For example, you can pass the RDS database size, EC2 instance types, database, and web server port numbers to AWS CloudFormation when you create a stack.
  - You can also use a parameterized template to create multiple stacks that may differ in a controlled way.
  - For example, your Amazon EC2 instance types, Amazon CloudWatch alarm thresholds, and Amazon RDS read-replica settings may differ among AWS regions if you receive more customer traffic in the US than in Europe.
  - You can use template parameters to tune the settings and thresholds in each region separately and still be sure that the application is deployed consistently across the regions.