AWS CloudFormation

What is AWS CloudFormation?

- AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS
- You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you
- You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that

Advantages of CloudFormation

- Simplify infrastructure management
- Quickly replicate your infrastructure
- Easily control and track changes to your infrastructure

CloudFormation Concepts

- Templates
  - A CloudFormation template is a JSON or YAML formatted text file
  - CloudFormation uses these templates as blueprints for building your AWS resources

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

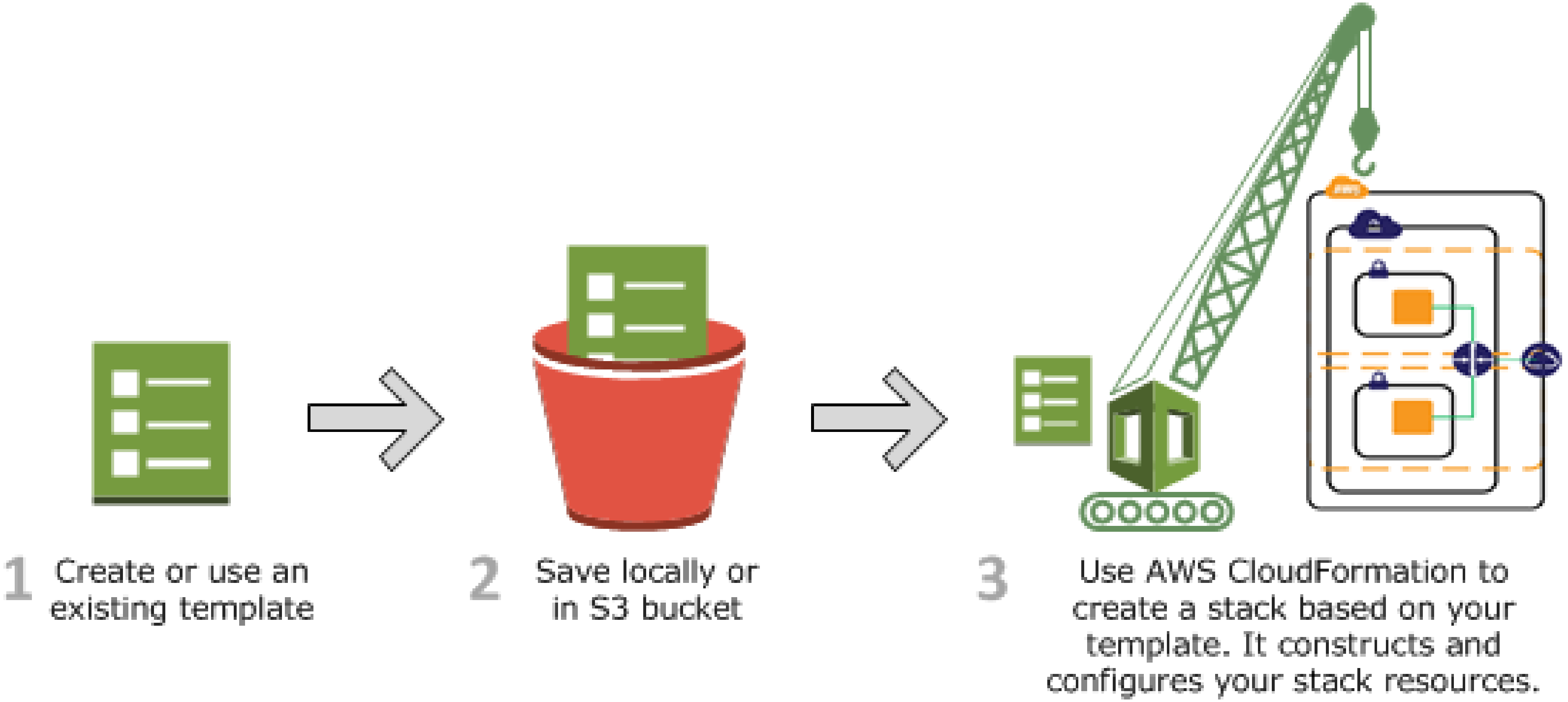
CloudFormation Concepts

- Stacks
  - When you use 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 CloudFormation template
  - Suppose you created a template that includes an Auto Scaling group, Elastic Load Balancing load balancer, and an Amazon Relational Database Service (Amazon RDS) database instance
  - To create those resources, you create a stack by submitting the template that you created, and CloudFormation provisions all those resources for you

CloudFormation Concepts

- Change Sets
  - If you need to make changes to the running resources in a stack, you update the stack
  - Before making changes to your resources, you can generate a change set, which is a summary of your proposed changes
  - Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them
  - For example, if you change the name of an Amazon RDS database instance, CloudFormation will create a new database and delete the old one. You will lose the data in the old database unless you've already backed it up
  - If you generate a change set, you will see that your change will cause your database to be replaced, and you will be able to plan accordingly before you update your stack

CloudFormation Concepts

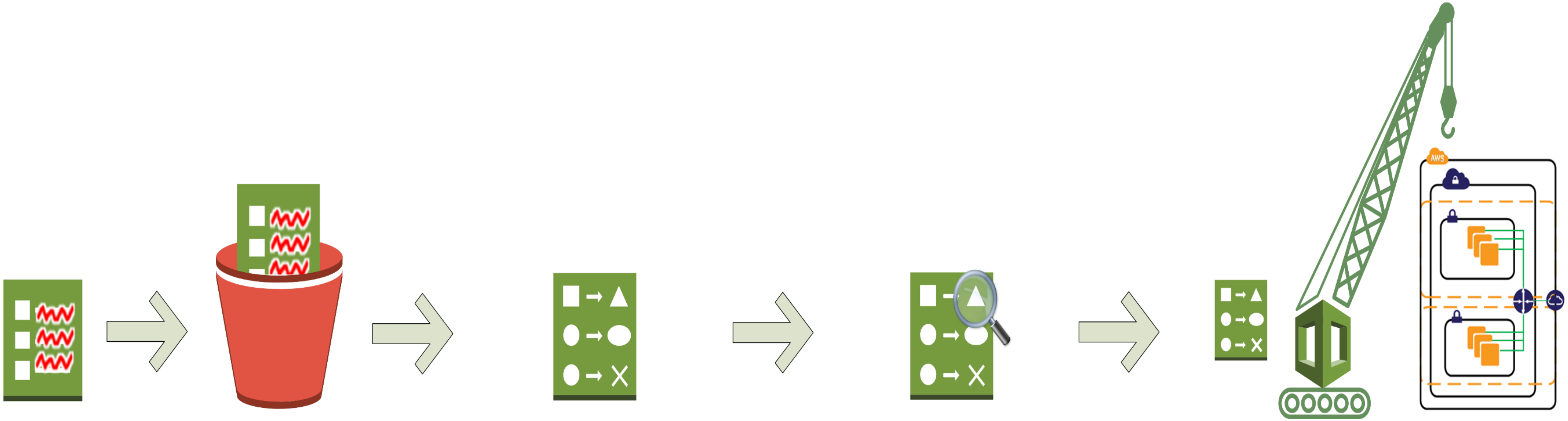


1 Create or use an existing template

2 Save locally or in S3 bucket

3 Use AWS CloudFormation to create a stack based on your template. It constructs and configures your stack resources.

CloudFormation Concepts



1 Edit template.

2 Save locally or in S3 bucket.

3 Use AWS CloudFormation to generate a change set based on your modified template and input parameter values.

4 View the change set, which describes the actions AWS CloudFormation performs if you execute it.

5 Execute the change set to update your stack. AWS CloudFormation performs all the changes described in the change set.

Template Anatomy

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

Description:

Metadata:

Parameters:

Rules:

Mappings:

Conditions:

Transform:

Resources:

Outputs:
```

Template Anatomy

- Format Version
- The latest template format version is 2010-09-09
- This is the only valid value

Template Anatomy

- Format Version
  - The latest template format version is 2010-09-09
  - This is the only valid value
  - AWSTemplateFormatVersion: "2010-09-09"

Template Anatomy

- Description
  - The Description section (optional) enables you to include comments about your template
  - The value for the description declaration must be a literal string that is between 0 and 1024 bytes in length
  - Description: > Here are some details about the template.

Template Anatomy

- Metadata
  - Optional Metadata section to include arbitrary JSON or YAML objects that provide details about the template.
  - For example, you can include template implementation details about specific resources, as shown in the following snippet:
  - Metadata: Instances: Description: "Information about the instances" Databases: Description: "Information about the databases"

Template Anatomy

- Parameters
- Parameters enable you to input custom values to your template each time you create or update a stack

```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.
```

Template Anatomy

- Parameters
- Referencing Parameters

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType:
      Ref: InstanceTypeParameter
    ImageId: ami-0ff8a91507f77f867
```

Template Anatomy

- Mappings

```
Mappings:
  RegionMap:
    us-east-1:
      "HVM64": "ami-0ff8a91507f77f867"
    us-west-1:
      "HVM64": "ami-0bdb828fd58c52235"
    eu-west-1:
      "HVM64": "ami-047bb4163c506cd98"
    ap-southeast-1:
      "HVM64": "ami-08569b978cc4dfa10"
    ap-northeast-1:
      "HVM64": "ami-06cd52961ce9f0d85"
```

Template Anatomy

- Conditions

```
Parameters:
  EnvType:
    Description: Environment type.
    Default: test
    Type: String
    AllowedValues:
      - prod
      - test
    ConstraintDescription: must specify prod or test.
Conditions:
  CreateProdResources: !Equals
    - !Ref EnvType
    - prod
```

Template Anatomy

- Resources

```
Resources:
  Logical ID:
    Type: Resource type
    Properties:
      Set of properties
```

```
Resources:
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-0ff8a91507f77f867"
```

Template Anatomy

- Outputs

```
Outputs:
  BackupLoadBalancerDNSName:
    Description: The DNSName of the backup load balancer
    Value: !GetAtt BackupLoadBalancer.DNSName
    Condition: CreateProdResources
  InstanceID:
    Description: The Instance ID
    Value: !Ref EC2Instance
```

Intrinsic Functions

- Fn::FindInMap

```
!FindInMap
        - 'InstanceConfiguration'
        - !Ref 'AWS::Region'
        - 'Type'
        - DefaultValue: m5.small
```

Intrinsic Functions

- Fn::GetAtt
- The Fn::GetAtt intrinsic function returns the value of an attribute from a resource in the template
- !GetAtt logicalNameOfResource.attributeName
- !GetAtt myELB.DNSName

Intrinsic Functions

- Fn::GetAZs
- The intrinsic function Fn::GetAZs returns an array that lists Availability Zones for a specified Region in alphabetical order
- !GetAZs region

```
mySubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId:
      !Ref VPC
    CidrBlock: 10.0.0.0/24
    AvailabilityZone:
      Fn::Select:
        - 0
        - Fn::GetAZs: ""
```