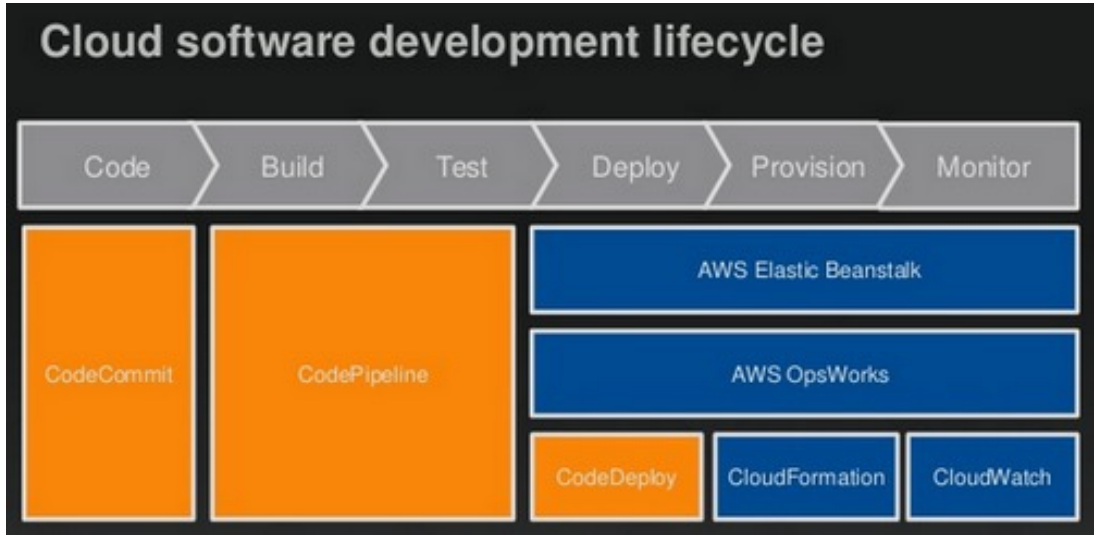
AWS CloudFormation

Tejas Parikh (t.parikh@northeastern.edu)
CSYE 6225
Northeastern University

What is AWS CloudFormation?

AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.



How CloudFormation Works

- Create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS 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; AWS CloudFormation handles all of that.

Advantages of CloudFormation

- Simplify Infrastructure Management
- Quickly Replicate Your Infrastructure
- Easily Control and Track Changes to Your Infrastructure
- It can handle dependencies
- It minimizes human failure
- It’s the documentation for your infrastructure

Simplify Infrastructure Management

- A web application may include a back-end database, an Auto Scaling group, an Elastic Load Balancing load balancer, and an Amazon Relational Database Service database instance. Normally, you might use each individual service to provision these resources. And after you create the resources, you would have to configure them to work together. All these tasks can add complexity and time before you even get your application up and running.
- Instead of manual provision, you can create AWS CloudFormation template that describes all the resources and their properties.
- When you use that template to create an AWS CloudFormation stack, AWS CloudFormation provisions all required resources such as Auto Scaling group, load balancer, and database. After the stack has been successfully created, your AWS resources are up and running. You can delete the stack just as easily, which deletes all the resources in the stack.
- **By using AWS CloudFormation, you easily manage a collection of resources as a single unit.**

Easily Control and Track Changes to Your Infrastructure

- Templates are regular files.
- Kept in version control system such as git.
- Track changes in your stack just like you track changes in your application.
- Reverse changes to your infrastructure by using previous version of your template

AWS CloudFormation Concepts

- **Templates**
  - An AWS CloudFormation template is a JSON or YAML formatted text file.
  - AWS CloudFormation uses these templates as blueprints for building your AWS resources.
- **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.
- **Change Sets**
  - Changes to the running resources in a stack are made by updating the stack.
  - Before making changes to your resources, you can generate a change set, which is 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.

Anatomy of a CloudFormation template

A basic CloudFormation template is structured into five parts (there are more than 5 parts though):

1. Format version (optional) - The latest template format version is 2010-09-09, and this is currently the only valid value. Specify this; the default is the latest version, which will cause problems if a new format version is introduced in the future.
2. Description (optional) —What is this template about?
3. Metadata (optional) - Objects that provide additional information about the template.
4. Parameters (optional) —Parameters are used to customize a template with values: for example, domain name, customer ID, and database password.
5. Rules (optional) - Validates a parameter or a combination of parameters passed to a template during a stack creation or stack update.
6. Mappings (optional) - A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function in the Resources and Outputs sections.
7. Conditions (optional) - Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.
8. Transform (optional) - For serverless applications (also referred to as Lambda-based applications), specifies the version of the AWS Serverless Application Model (AWS SAM) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it's processed.
9. Resources (required)—A resource is the smallest block you can describe. Examples are a virtual server, a load balancer, or an elastic IP address.
10. Outputs (optional)—An output is comparable to a parameter, but the other way around. An output returns something from your template, such as the public name of an EC2 server.

Anatomy of a CloudFormation template

JSON

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

  "Description" : "JSON 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
  }
}
```

YAML

```
---
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
```

Format Version (optional)

- The **AWSTemplateFormatVersion** section identifies the capabilities of the template.
- The latest template format version is **2010-09-09** and is currently the only valid value.

Description (optional)

- The Description section enables you to include comments about your template.

Metadata (optional)

- The optional Metadata section allows us to include arbitrary JSON or YAML objects that provide details about the template.

Parameters (optional)

- Specifies values that you can pass in to your template at runtime (when you create or update a stack).
- You can refer to parameters in the *Resources* and *Outputs* sections of the template.
- With parameters, you can create templates that are customized each time you create a stack.

Rules (optional)

- The Rules section validates a parameter or a combination of parameters passed to a template during a stack creation or stack update.
- To use template rules, explicitly declare Rules in your template followed by an assertion.
- Use the rules section to validate parameter values before creating or updating resources.

Mappings (optional)

- The Mappings section matches a key to a corresponding set of named values.
- Example:
  - The example shows a Mappings section with a map RegionMap, which contains five keys that map to name-value pairs containing single string values.
  - The keys are region names.
  - Each name-value pair is the AMI ID for the HVM64 AMI in the region represented by the key.

YAML

```yaml
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"
```

Conditions

- The Conditions section contains statements that define the circumstances under which entities are created or configured.

YAML

```yaml
Parameters:
  EnvType:
    Type: String
    AllowedValues:
      - prod
      - test
  BucketName:
    Default: ''
    Type: String
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
  CreateBucket: !Not
    - !Equals
      - !Ref BucketName
      - ''
  CreateBucketPolicy: !And
    - !Condition IsProduction
    - !Condition CreateBucket
Resources:
  Bucket:
    Type: 'AWS::S3::Bucket'
    Condition: CreateBucket
  Policy:
    Type: 'AWS::S3::BucketPolicy'
    Condition: CreateBucketPolicy
    Properties:
      Bucket: !Ref Bucket
      PolicyDocument: ...
```

Resources (required)

- Specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket.
- Resources can be referenced in the Resources and Outputs sections of the template.

Outputs (optional)

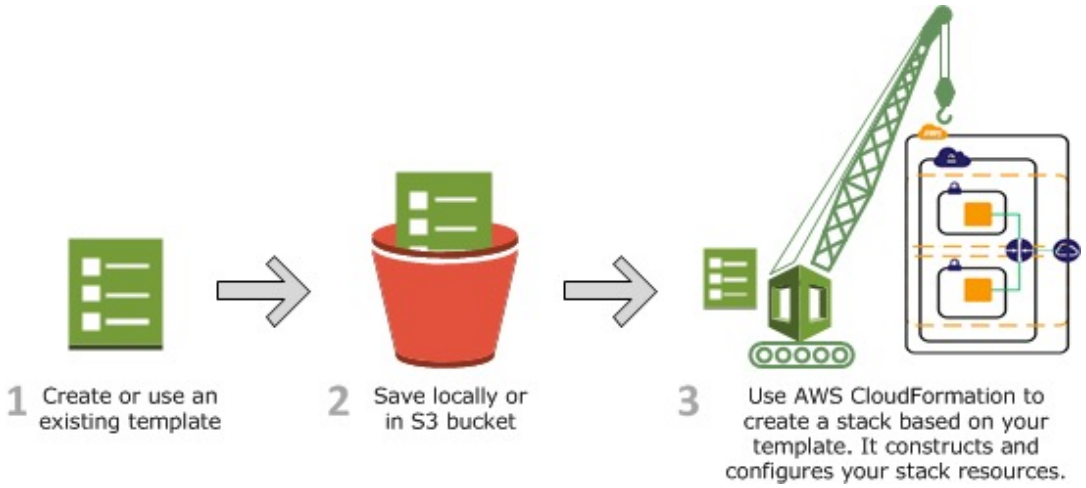
- Describes the values that are returned whenever you view your stack's properties.
- For example, you can declare an output for an S3 bucket name and then call the *aws cloudformation describe-stacks* AWS CLI command to view the name.

Intrinsic Function Reference

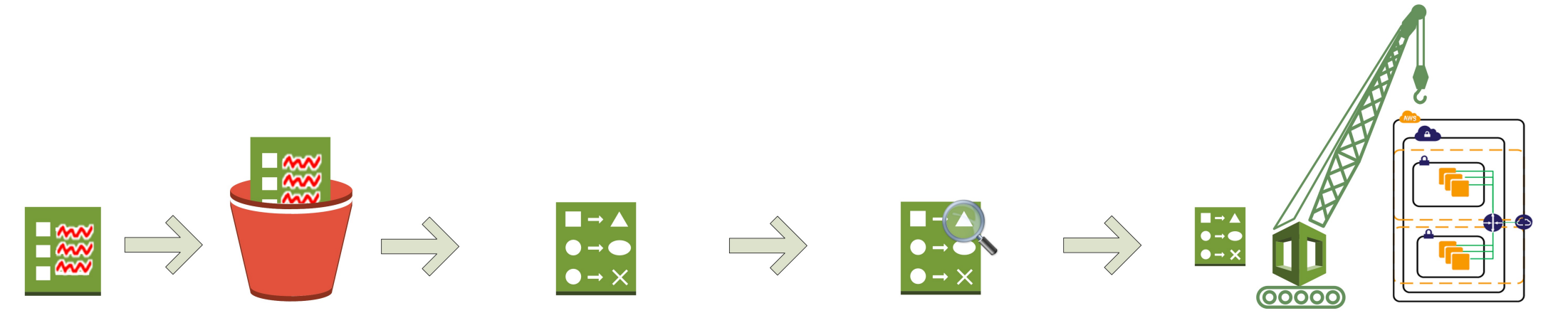
- AWS CloudFormation provides several built-in functions that help you manage your stacks. Use intrinsic functions in your templates to assign values to properties that are not available until runtime.
- *You can use intrinsic functions only in specific parts of a template. Currently, you can use intrinsic functions in resource properties, outputs, metadata attributes, and update policy attributes. You can also use intrinsic functions to conditionally create stack resources.*

Creating CloudFormation Stack

- To create a stack you run the *aws cloudformation create-stack* command.
- You must provide the stack name, the location of a valid template, and any input parameters.



Updating a Stack with Change Sets



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.

Delete Stack

- To delete a stack, you run the *aws cloudformation delete-stack* command.
- You must specify the name of the stack that you want to delete.
- When you delete a stack, **you delete the stack and all of its resources**.

Demo

	CloudFormation with a script on server startup	Elastic Beanstalk	OpsWorks
Configuration-management tool	All available tools	Proprietary	Chef
Supported platforms	Any	■ PHP ■ Node.js ■ IIS ■ Java/Tomcat ■ Python ■ Ruby ■ Docker	■ Ruby on Rails ■ Node.js ■ PHP ■ Java/Tomcat ■ Custom/any
Supported deployment artifacts	Any	Zip archive on Amazon S3	Git, SVN, archive (such as Zip)
Common use case	Complex and nonstandard environments	Common web application	Micro-services environment
Update without downtime	Possible	Yes	Yes
Vendor lock-in effect	Medium	High	Medium

Additional Resources

https://fall2018.csye6225.cloud/