

# AWS CloudFormation

Tejas Parikh ([t.parikh@northeastern.edu](mailto:t.parikh@northeastern.edu))

Spring 2018

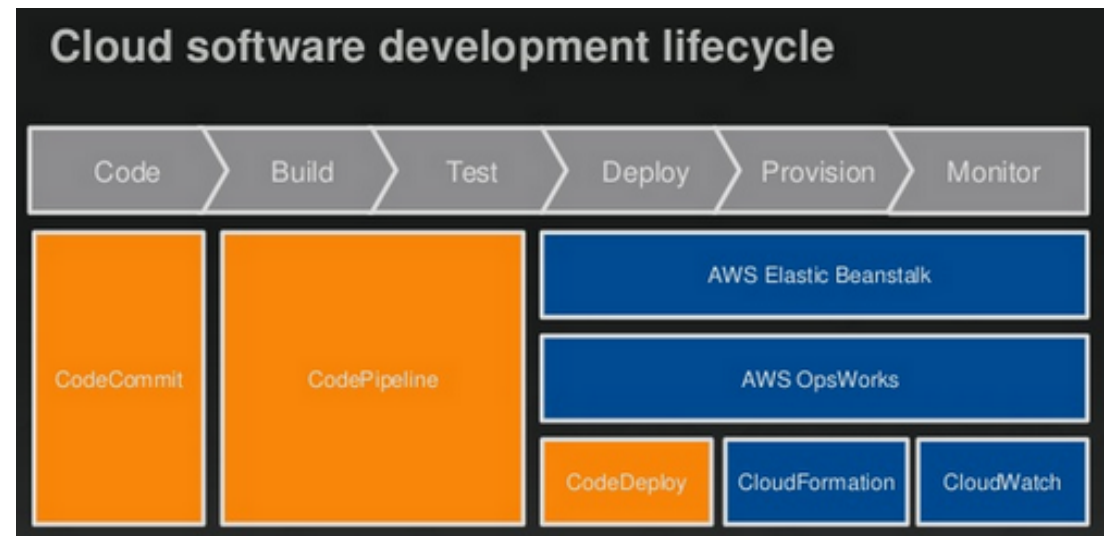
CSYE 6225

Northeastern University

<https://spring2018.csye6225.com>

# 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

- You 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. A template describes all of your 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.**

# Quickly Replicate Your Infrastructure

- Replicate your stack in multiple regions by reusing the template
- High availability

# 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** - 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 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. Parameters (optional) —Parameters are used to customize a template with values: for example, domain name, customer ID, and database password.
4. Resources (required)—A resource is the smallest block you can describe. Examples are a virtual server, a load balancer, or an elastic IP address.
5. 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.

```
{
  "AWSTemplateFormatVersion" : "version date",
  "Description" : "JSON string",
  "Metadata" : {
    template metadata
  },
  "Parameters" : {
    set of parameters
  },
  "Mappings" : {
    set of mappings
  },
  "Conditions" : {
    set of conditions
  },
  "Transform" : {
    set of transforms
  },
  "Resources" : {
    set of resources
  },
  "Outputs" : {
    set of outputs
  }
}
```

# 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.

# Resources (required)

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

# Outputs (optional)

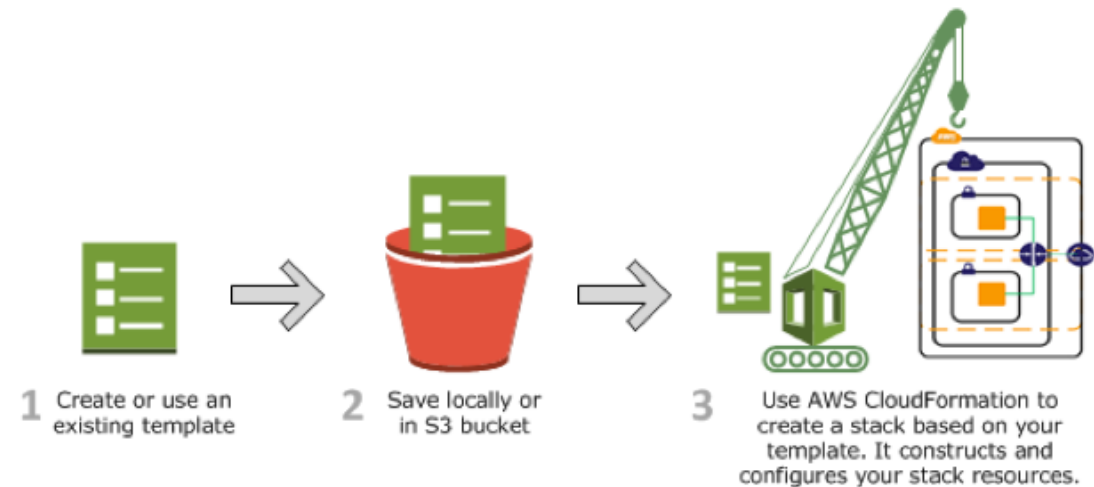
Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name and then call the *aws cloudformation describe-stacks* AWS CLI command to view the name.

# Intrinsic Function Reference

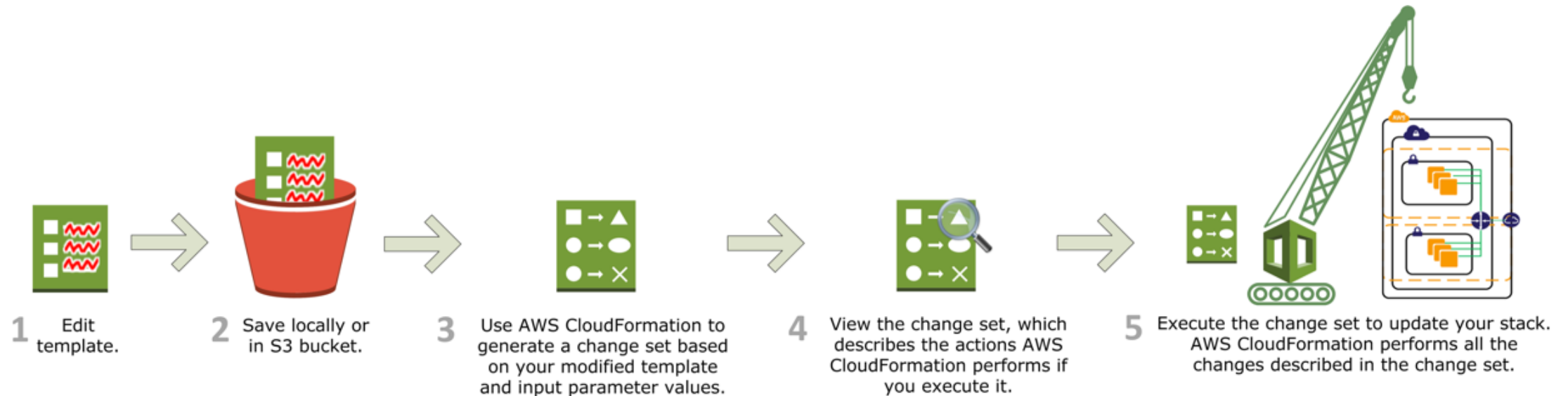
- AWS CloudFormation provides several built-in functions that help you manage your stacks. Use intrinsic functions in your templates to assign values to properties that are not available until runtime.
- *You can use intrinsic functions only in specific parts of a template.*  
*Currently, you can use 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



# 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	<ul style="list-style-type: none"> <li>■ PHP</li> <li>■ Node.js</li> <li>■ IIS</li> <li>■ Java/Tomcat</li> <li>■ Python</li> <li>■ Ruby</li> <li>■ Docker</li> </ul>	<ul style="list-style-type: none"> <li>■ Ruby on Rails</li> <li>■ Node.js</li> <li>■ PHP</li> <li>■ Java/Tomcat</li> <li>■ Custom/any</li> </ul>
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://spring2018.csye6225.com>