AWS on Steroids: CloudFormation Templates

Paul Marcelin
marcelin@alumni.cmu.edu

Northern California Oracle User Group

August 3, 2017

A link to the slides and examples will be provided at the end

“Infrastructure as Code” Goals

- Document:
  - Declare your system configuration.
  - Track changes in a source code control system like Git.
  - Keep declarations and infrastructure synchronized.
- Repeat:
  - Multiple people...
  - Can create consistent instances...
  - Serving particular purposes...
  - In multiple environments.

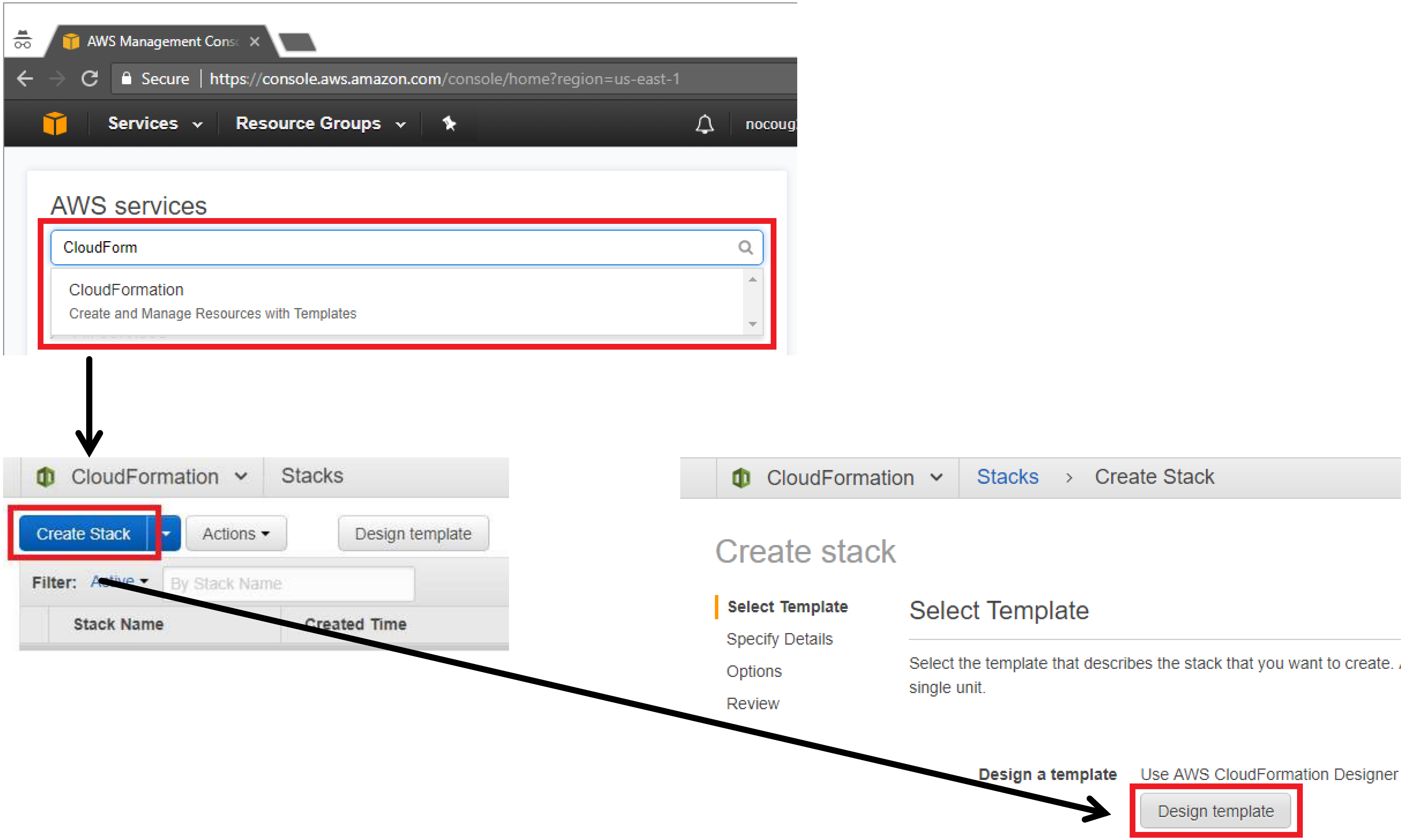
CloudFormation's Unique Role

- The Amazon Web Console:
  - Is fast for experimentation, but slow for repetition.
- The Amazon API:
  - Is imperative, not declarative. You cannot change infrastructure by editing the code used to create it.
- Generic tools (Chef, Puppet, Ansible, SaltCloud, Terraform):
  - May not support new or specialized Amazon resources and attributes.
  - Are perfect for "on-machine" configuration (operating system packages, user accounts, etc.), taking over where CloudFormation leaves off.

Choose YAML over JSON

- CloudFormation was originally JSON-only.
- YAML, supported since 2016,
  - Is easier to read, and
  - Permits comments.
- This presentation uses YAML.
- Most publicly-available templates were written in JSON.
- Convert with one click, using CloudFormation Designer in the Amazon Web Console…

Start CloudFormation Designer



AWS Management Cons

Secure | https://console.aws.amazon.com/console/home?region=us-east-1

AWS services

CloudForm

CloudFormation

Create and Manage Resources with Templates

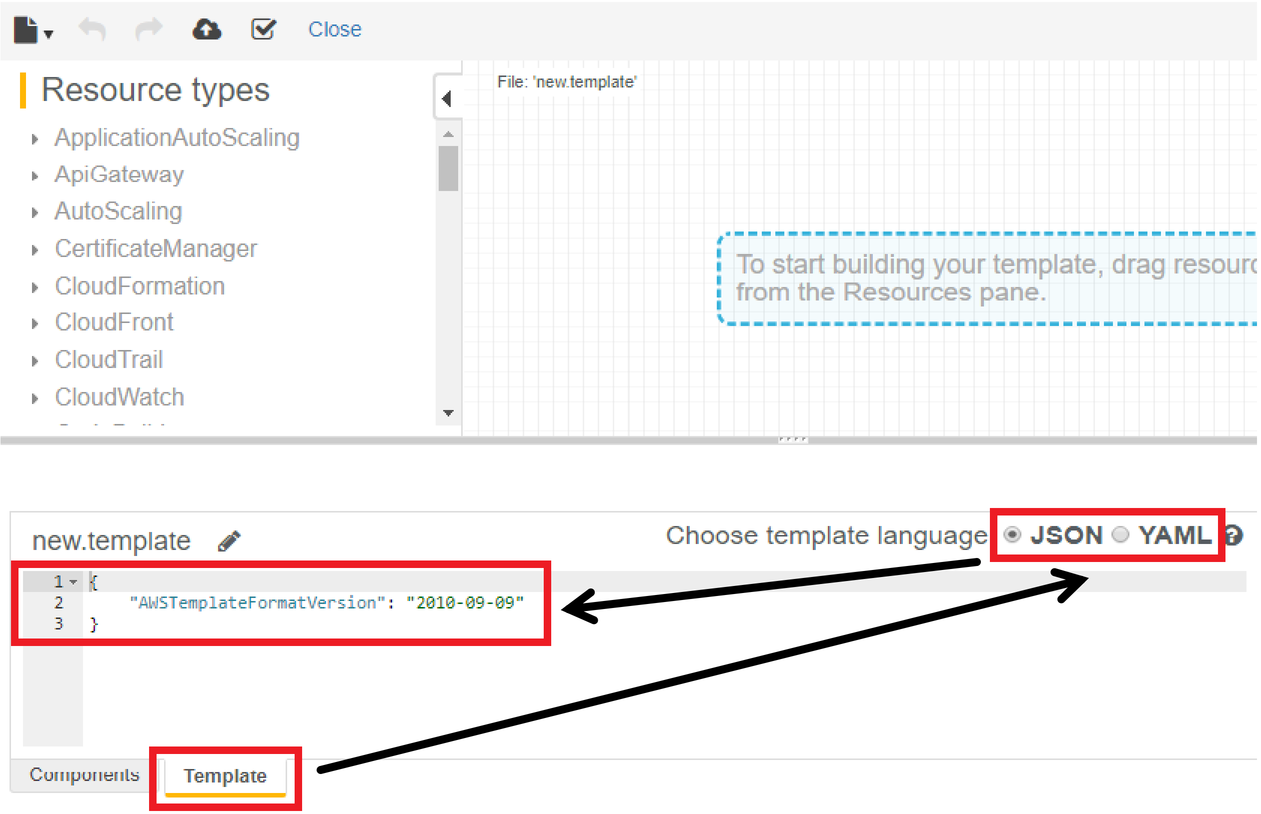
Create stack

Select Template

Select the template that describes the stack that you want to create.
single unit.

Design a template Use AWS CloudFormation Designer

Set Language and Select Default Template



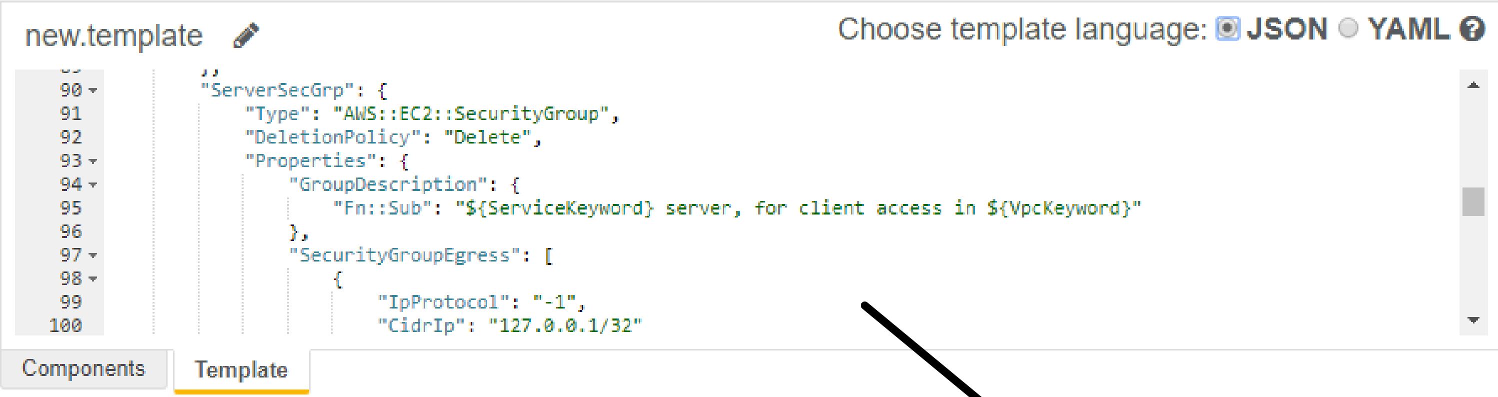
File: 'new.template'

To start building your template, drag resourc from the Resources pane.

Choose template language ● JSON ○ YAML

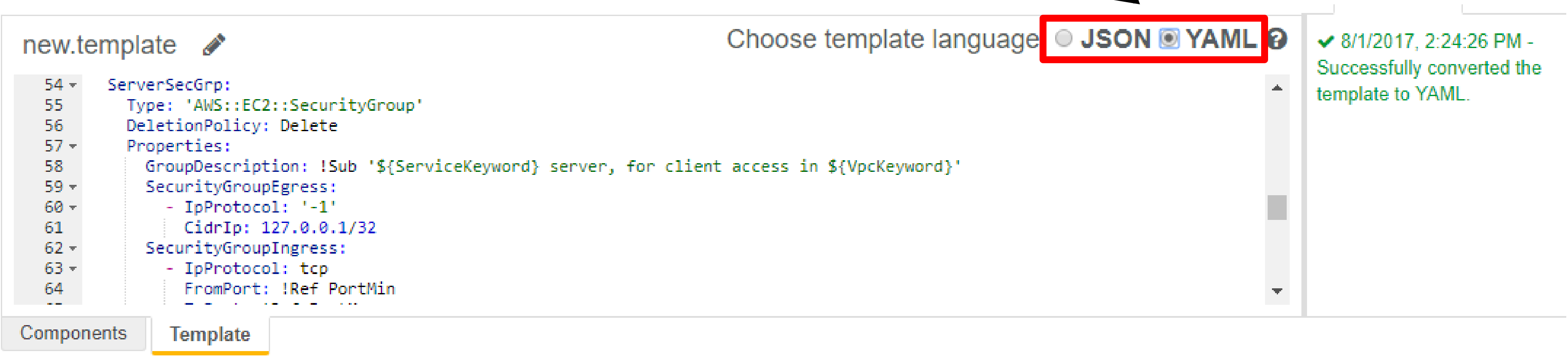
```
{
    "AWSTemplateFormatVersion": "2010-09-09"
}
```

Paste JSON Template and Convert to YAML



Choose template language: ● JSON ○ YAML

```
"ServerSecGrp": {
    "Type": "AWS::EC2::SecurityGroup",
    "DeletionPolicy": "Delete",
    "Properties": {
        "GroupDescription": {
            "Fn::Sub": "${ServiceKeyword} server, for client access in ${VpcKeyword}"
        },
        "SecurityGroupEgress": [
            {
                "IpProtocol": "-1",
                "CidrIp": "127.0.0.1/32"
```



Choose template language ○ JSON ● YAML

```
ServerSecGrp:
  Type: 'AWS::EC2::SecurityGroup'
  DeletionPolicy: Delete
  Properties:
    GroupDescription: !Sub '${ServiceKeyword} server, for client access in ${VpcKeyword}'
    SecurityGroupEgress:
      - IpProtocol: '-1'
        CidrIp: 127.0.0.1/32
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: !Ref PortMin
```

✓ 8/1/2017, 2:24:26 PM - Successfully converted the template to YAML.

Sections of a Template

```
---
AWSTemplateFormatVersion: "2010-09-09"
# Do not edit above this line!

Description: "Optional template summary string"

Parameters: # Optional section to prompt
# for new input values every time a stack
# is instantiated from the template

Resources:  # Required; core of the template

Outputs:    # Optional section to pass values
# to parent or peer templates
```

A Resource By Any Other Name...

- Every CloudFormation resource has a local name (“logical identifier”), used throughout the template of origin.
- Values passed to parent and peer templates are also named.
- Every stack (instance of a template) is uniquely named.
- Many resources also have names outside CloudFormation. Visible throughout the Web Console, these combine:
  - stack name,
  - local resource name from the template of origin, and
  - a random alphanumeric identifier.

Effective Naming Schemes:

- Take scope into account (1 template, 2 related templates, many templates, an entire AWS region, etc.).
- Use delimiters and case transitions advantageously.
- Place distinguishing details first, in case of truncation.

  ```
  ClientLogicalID
  ServerLogicalID
  ```

- *Or*, place categories first, so that related items sort together.

  ```
  SecurityGroup-Client
  SecurityGroup-Server
  EncryptionKey-Disk-Root
  EncryptionKey-Disk-Data
  ```

- Can accommodate new kinds of items.

Effective Names:

- Are short.
- Reveal purpose or application
  (`HRDB` instead of `OracleDB1` or `"Susan's DB"`).
- Distinguish resource types (`HRDB` and `HRServ`).
- Distinguish environments, availability zones, etc.
  (`HRDBProdA`, `HRDBProdB`, `HRDBDev`).
  - *When passing values to other stacks, build their names from parameters and constants. Do not hard-code!*
- Distinguish identifier types
  (reference `RootDiskKey` locally, but pass
  `RootDiskKeyID`, `RootDiskKeyAlias`, `RootDiskKeyARN`).
- Make sense to other people!

AWS Resources for an Oracle Instance

- Virtual Private Cloud (VPC) - basic network definition
- Database subnet group
  - We will use the default VPC and create a simplified subnet group; for production, use a model such as: *https://aws.amazon.com/quickstart/architecture/accelerator-uk-official/*
- Security groups - lists of firewall rules
- Key Management System (KMS) encryption key and alias
  - We will use the default key; for production, customize.
- Database option group - database engine configuration
- Database parameter group - database instance configuration
  - We will duplicate default options and parameters.

Security Group Pair: Parameters

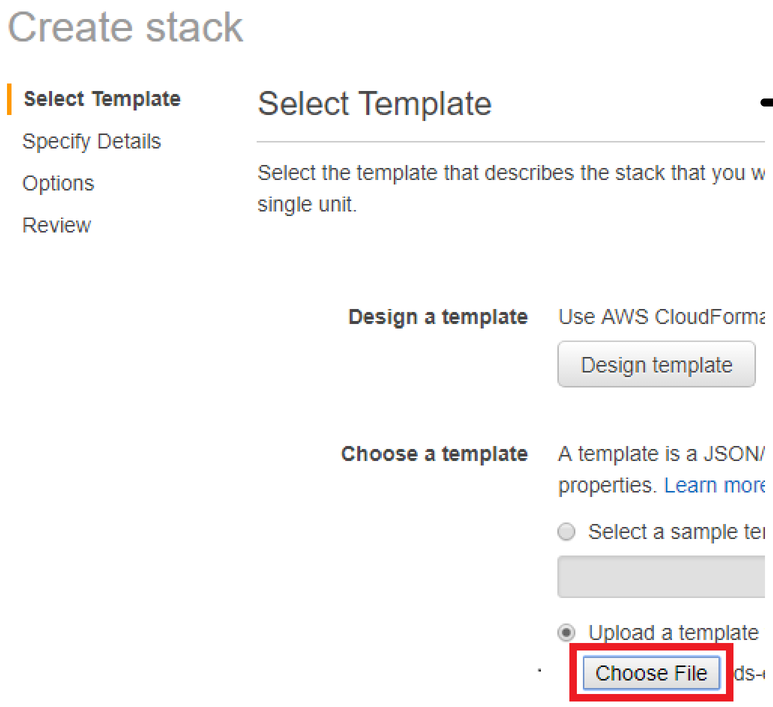
```
Parameters:

  VpcKeyword:
    Type: String
    Description: "Keyword to identify the VPC ..."
    Default: "DefaultVPC"

  VpcId:
    Type: "AWS::EC2::VPC::Id"
    Description: "Parent Virtual Private Cloud (VPC) ID ..."

  PortMin:
    Type: Number
    Description: "Service port range start"
    MinValue:     0
    MaxValue: 65535
    Default:   1521  # Oracle database

  ...
```

Security Group Pair: Parameter Metadata

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Port Range"
        Parameters:  # Preserves desired non-alphabetic order
          - PortMin
          - PortMax
```

YAML list elements start with hyphens, and may span multiple lines

Creating the Security Group Pair Stack



Create stack

- Select Template
- Specify Details
- Options
- Review

Select Template

Select the template that describes the stack that you w
single unit.

Design a template Use AWS CloudForma

Design template

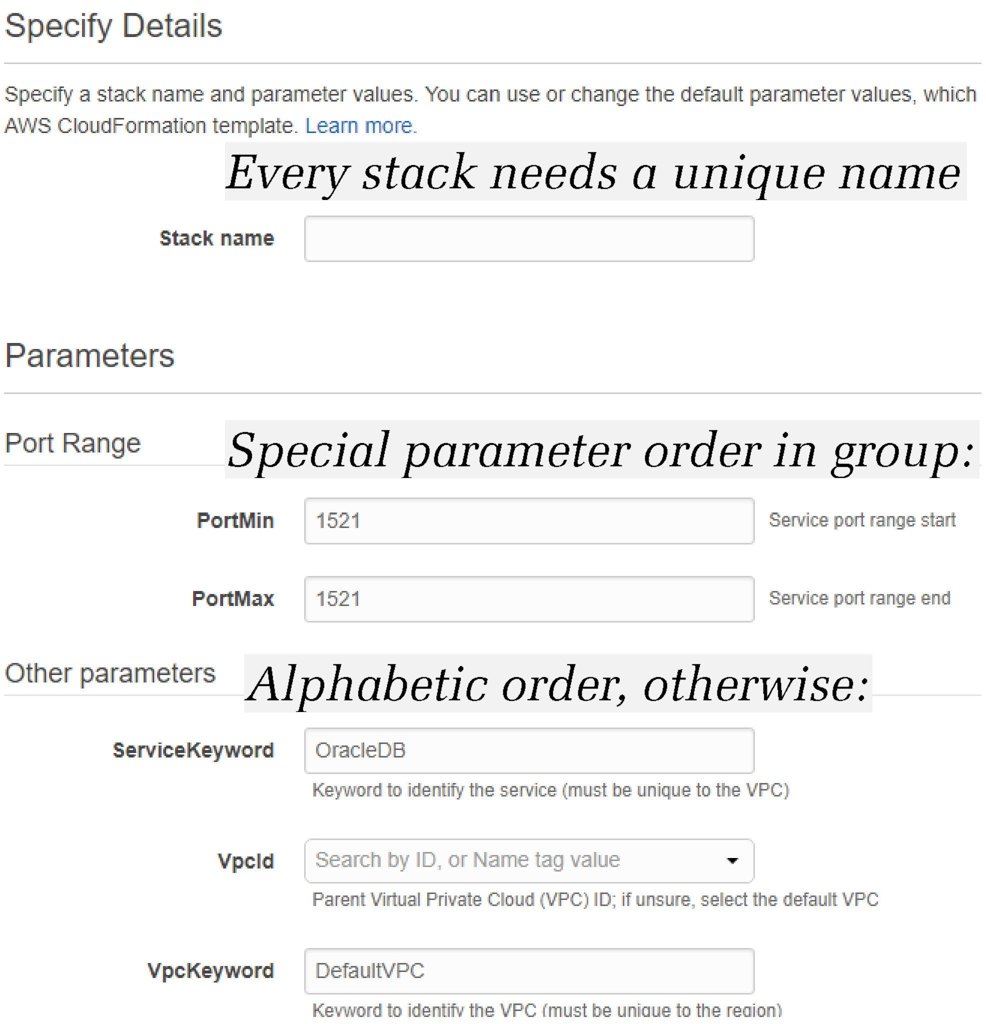
Choose a template A template is a JSON/
properties. Learn more

○ Select a sample te

● Upload a template

Choose File ds-

→



Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which
AWS CloudFormation template. Learn more.

Every stack needs a unique name

Stack name

Parameters

Port Range

Special parameter order in group:

PortMin 1521 Service port range start

PortMax 1521 Service port range end

Other parameters

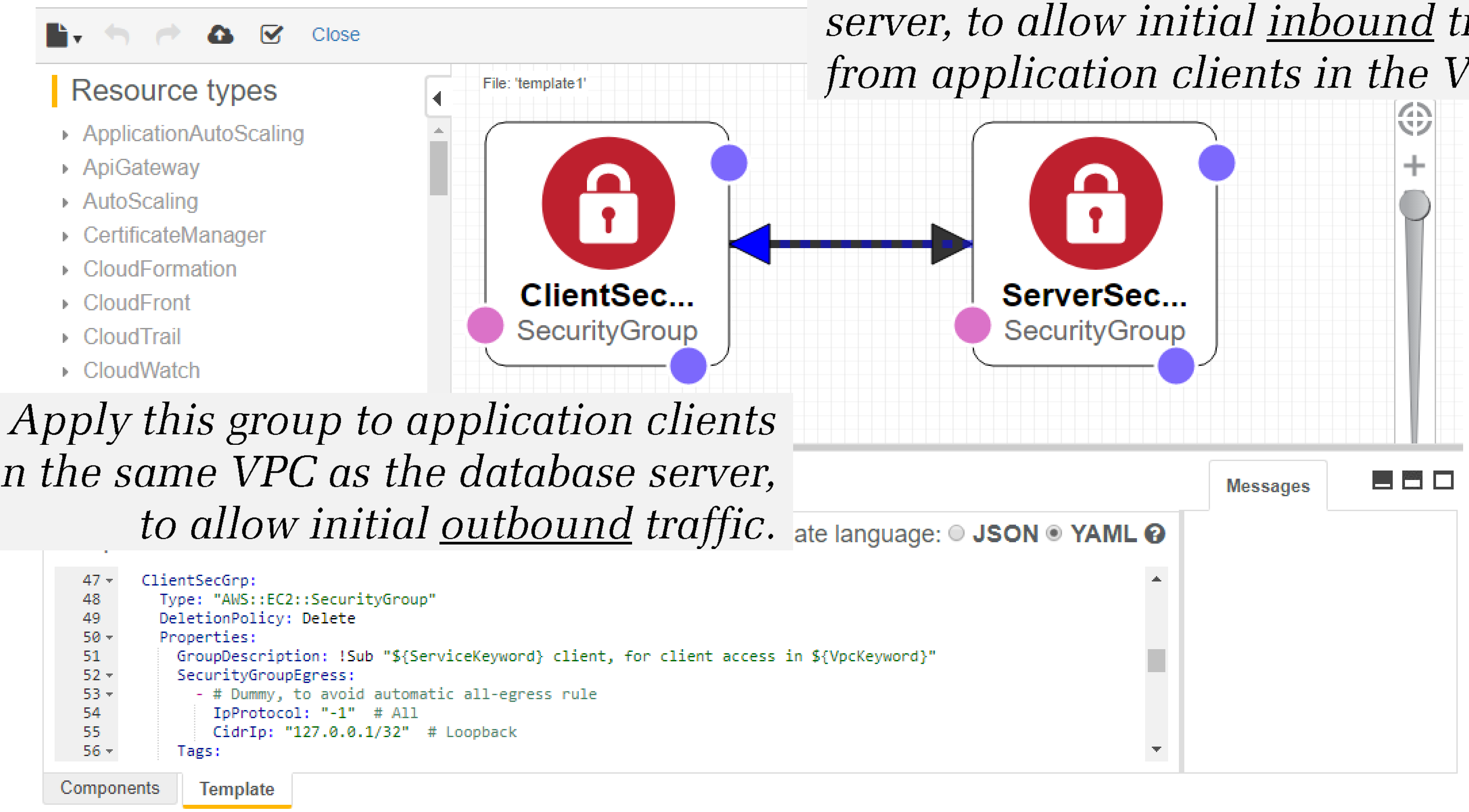
Alphabetic order, otherwise:

ServiceKeyword OracleDB
Keyword to identify the service (must be unique to the VPC)

VpcId Search by ID, or Name tag value
Parent Virtual Private Cloud (VPC) ID; if unsure, select the default VPC

VpcKeyword DefaultVPC
Keyword to identify the VPC (must be unique to the region)

Clarify Security Concept, in Designer



Apply this group to the database server, to allow initial inbound traffic from application clients in the VPC.

Apply this group to application clients in the same VPC as the database server, to allow initial outbound traffic.

ate language: ○ JSON ● YAML

```
ClientSecGrp:
  Type: "AWS::EC2::SecurityGroup"
  DeletionPolicy: Delete
  Properties:
    GroupDescription: !Sub "${ServiceKeyword} client, for client access in ${VpcKeyword}"
    SecurityGroupEgress:
      - # Dummy, to avoid automatic all-egress rule
        IpProtocol: "-1"  # All
        CidrIp: "127.0.0.1/32"  # Loopback
    Tags:
```

Resource Definition, with References

```
Resources:
```

Resource local name ("logical identifier")

```
  ClientSecGrp:
    ...
  ServerSecGrp:
    Type: "AWS::EC2::SecurityGroup"
    DeletionPolicy: Delete
    Properties:
```

Parameter

```
      GroupDescription: !Sub "${ServiceKeyword} server, ..."
      SecurityGroupIngress:
```

Also try "pseudo-parameter" constants!

```
        -
          IpProtocol: tcp
```

Parameter

```
          FromPort: !Ref PortMin
          ToPort: !Ref PortMax
```

Resource, defined above

```
          SourceSecurityGroupId: !Ref ClientSecGrp
      SecurityGroupEgress:
        ...
      VpcId: !Ref VpcId
```

Output to Parent, Export to Peers

```
Outputs:
```

Output name, for a parent template ("nested stacks")

```
  ClientSecGrpId:
    Value: !Ref ClientSecGrp
    Description: "Client security group ID"

    Export:
```

Export name, for peer templates ("cross-stack references")

```
      Name: !Sub "SecGrp-${VpcKeyword}-${ServiceKeyword}\
                  -Client-App-InVpc-Id"

  ServerSecGrpId:
    Value: !Ref ServerSecGrp
    Description: "Server security group ID"

    Export:
      Name: !Sub "SecGrp-${VpcKeyword}-${ServiceKeyword}\
                  -Server-App-InVpc-Id"
```

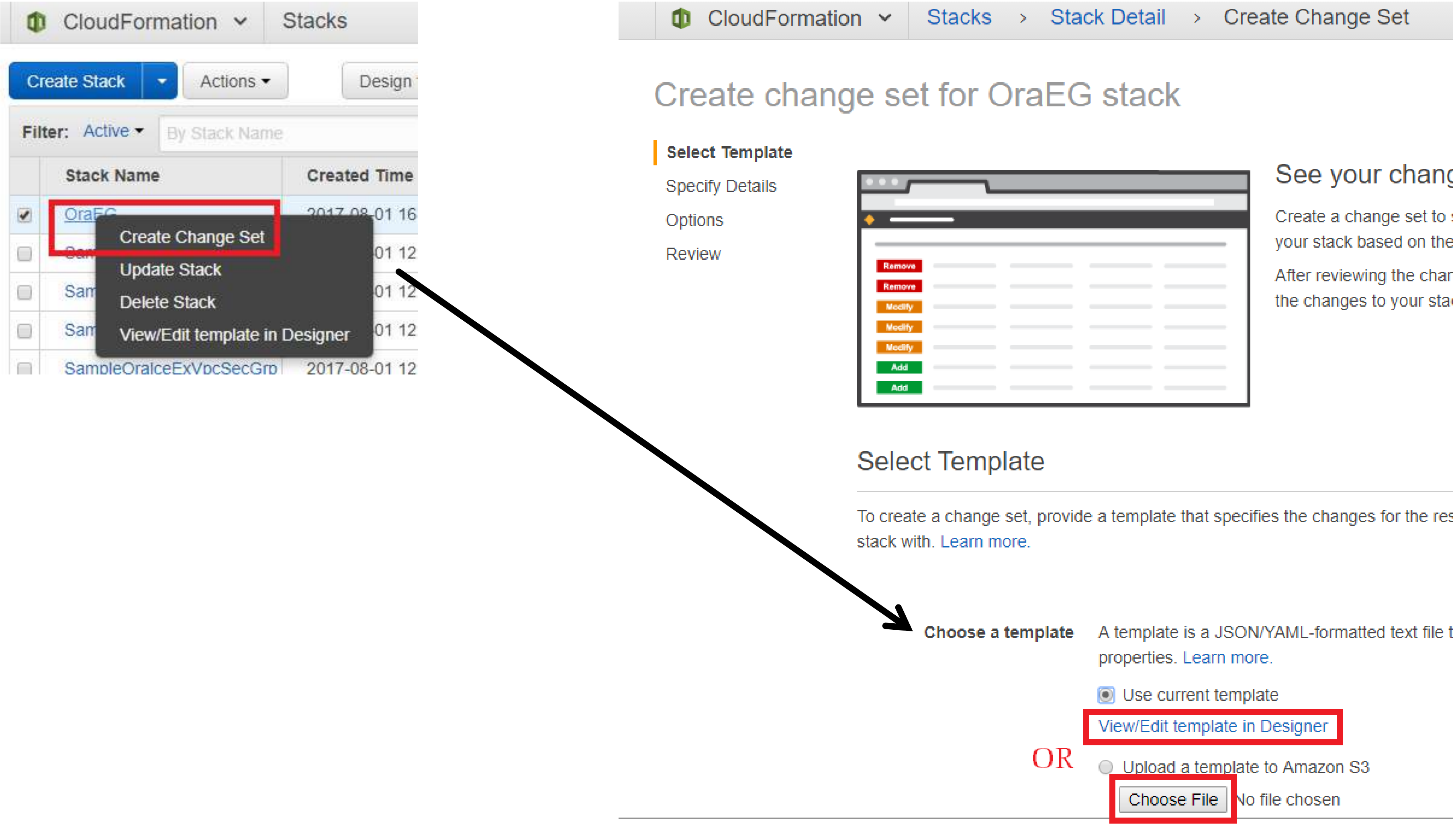
“Cross-Stack” Import from a Peer Template

```
Resources:

  SampleRdsInst:
    Type: "AWS::RDS::DBInstance"
    ...
    Properties:
      ...
      VPCSecurityGroups:
        - !ImportValue "SecGrp-DefaultVPC-OracleDB\
                        -Server-App-InVpc-Id"
        - !ImportValue "SecGrp-DefaultVPC-OracleDB\
                        -Server-App-ExVpc-Id"
      ...
```

Changes Involve Editing, Not Rewriting!

```
Resources:

  SampleRdsInst:
    Type: "AWS::RDS::DBInstance"
    ...
    Properties:
      ...
      AllocatedStorage: ~~20~~25  # GiB
      ...
```

Create a Change Set Reflecting the Edit



CloudFormation | Stacks

Create Stack | Actions | Design

Filter: Active | By Stack Name

	Stack Name	Created Time
☑	OraEG	2017-08-01 16
□	Sam	01 12
□	Sam	01 12
□	Sam	01 12
□	SampleOralceExVpcSecGrp	2017-08-01 12

CloudFormation | Stacks › Stack Detail › Create Change Set

Create change set for OraEG stack

See your chang

Create a change set to
your stack based on the

After reviewing the char
the changes to your sta

Select Template

To create a change set, provide a template that specifies the changes for the res
stack with. Learn more.

Choose a template: A template is a JSON/YAML-formatted text file t
properties. Learn more.

● Use current template

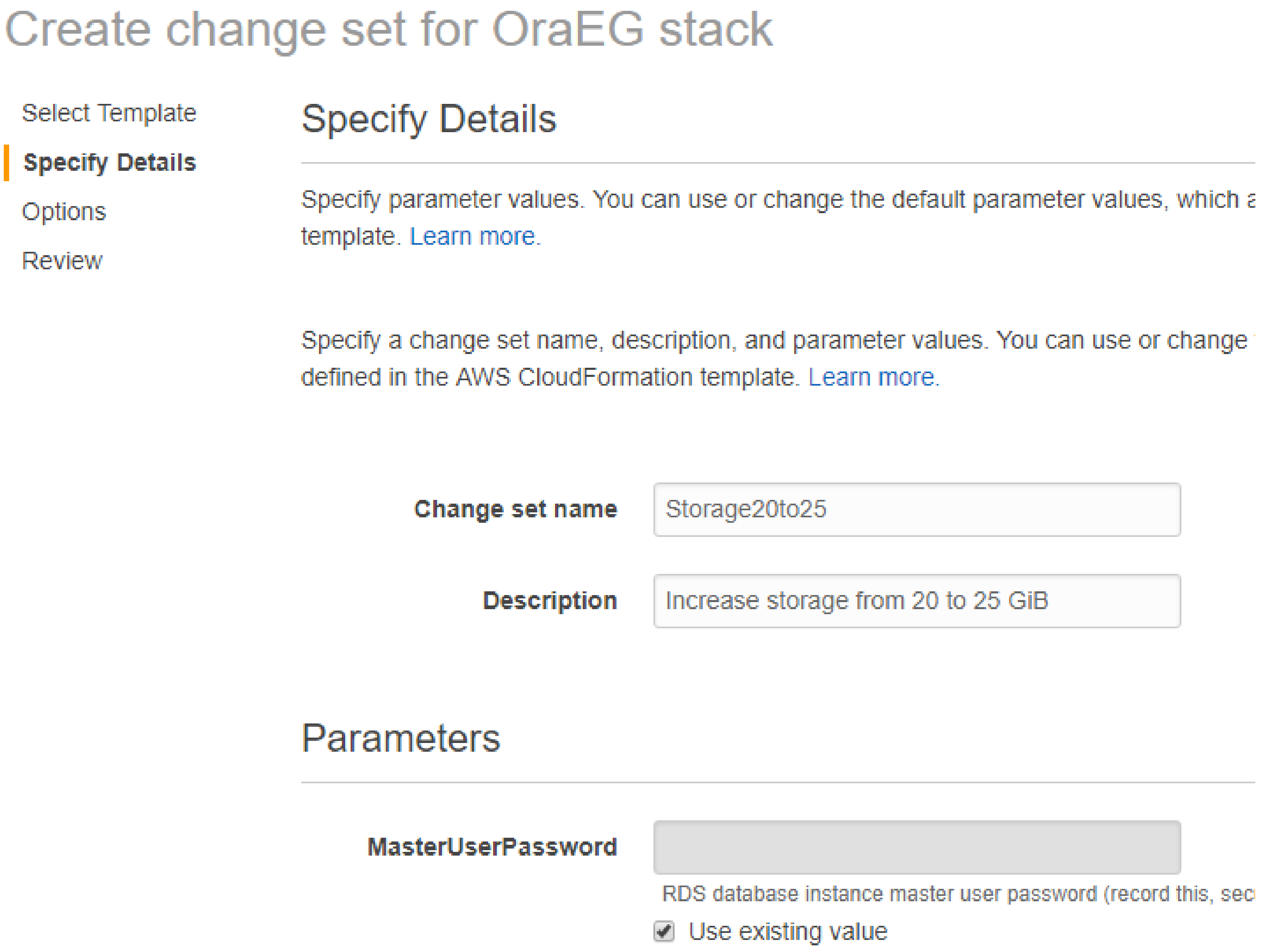
View/Edit template in Designer

OR

○ Upload a template to Amazon S3

Choose File No file chosen

Describe the Change Set



Create change set for OraEG stack

Select Template

Specify Details

Options

Review

Specify Details

Specify parameter values. You can use or change the default parameter values, which a template. Learn more.

Specify a change set name, description, and parameter values. You can use or change defined in the AWS CloudFormation template. Learn more.

Change set name Storage20to25

Description Increase storage from 20 to 25 GiB

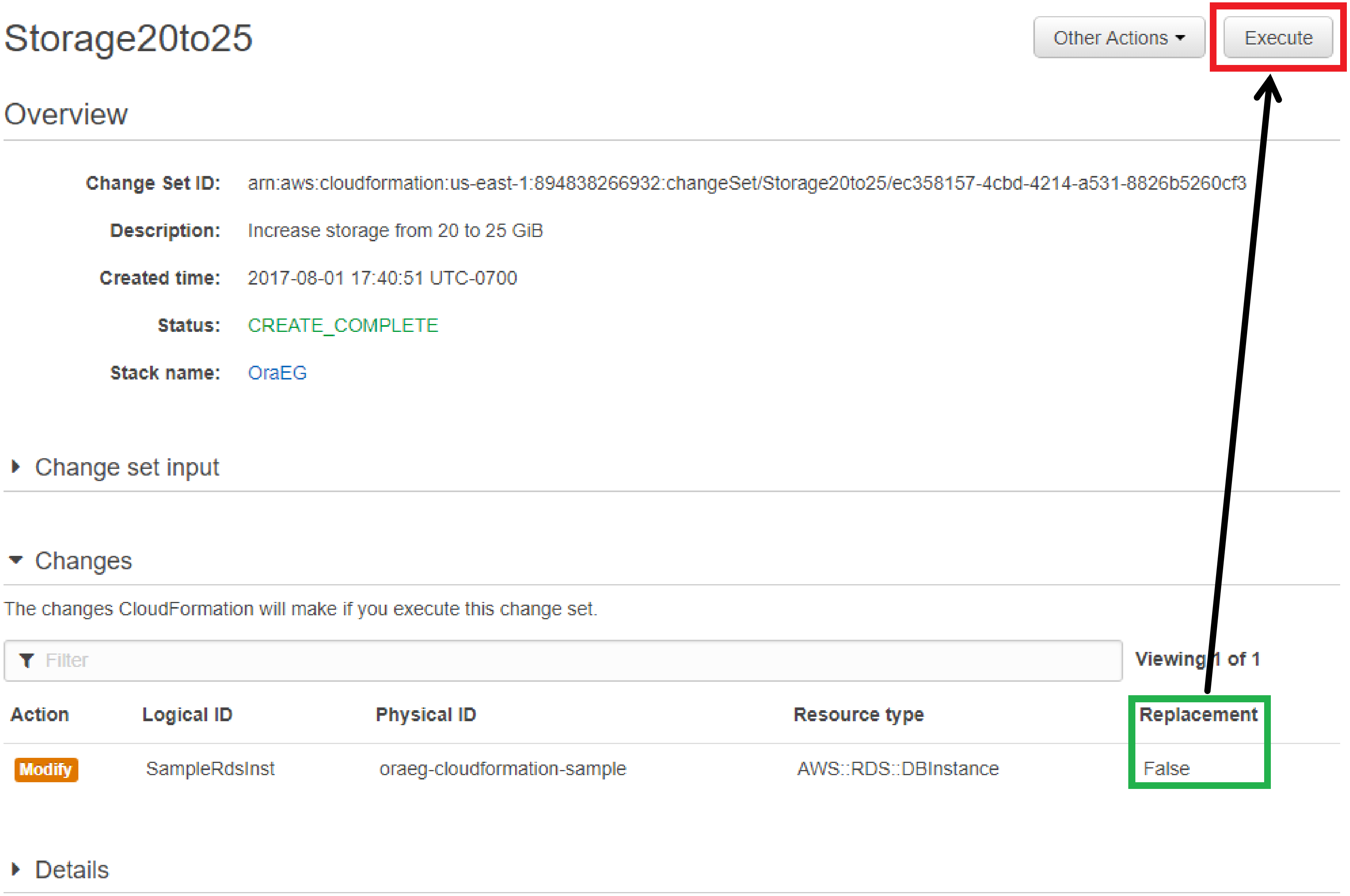
Parameters

MasterUserPassword

RDS database instance master user password (record this, sec

☑ Use existing value

Review Proposed Changes Before Executing



Storage20to25

Overview

Change Set ID: arn:aws:cloudformation:us-east-1:894838266932:changeSet/Storage20to25/ec358157-4cbd-4214-a531-8826b5260cf3

Description: Increase storage from 20 to 25 GiB

Created time: 2017-08-01 17:40:51 UTC-0700

Status: CREATE_COMPLETE

Stack name: OraEG

▸ Change set input

▾ Changes

The changes CloudFormation will make if you execute this change set.

Filter

Viewing 1 of 1

Action	Logical ID	Physical ID	Resource type	Replacement
Modify	SampleRdsInst	oraeg-cloudformation-sample	AWS::RDS::DBInstance	False

▸ Details

Summary: “Infrastructure as Code”

- With CloudFormation, you can…
  - Create resources declaratively.
  - Change resources by editing code instead of writing new code.
  - Track configuration in a source code control system.
  - Quickly spin up similar resources.
  - Share definitions with other people, so that they can launch resources on their own and you can move on to more interesting work.

Learning Resources

- CloudFormation user guide
  docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/

- AWS Loft - free training and advice
  https://aws.amazon.com/start-ups/loft/sf-loft/
  1446 Market Street (at Van Ness Avenue)
  San Francisco

- YAML guide for Ruby users
  yaml.org/YAML_for_ruby.html

- YAML guide for Python users
  https://docs.saltstack.com/en/latest/topics/yaml/

- Materials from this presentation
  bit.ly/nocougcfn1

Advanced Topics

- AWS IAM (Identity and Access Management) security roles
  docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
- Nested stacks and AWS S3 ("Simple Storage Service")
  docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-stack.html
- AWS command-line interface
  docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-cli.html
- Stack Sets for multiple AWS accounts (new in July, 2017!)
  https://aws.amazon.com/blogs/aws/use-cloudformation-stacksets-to-provision-resources-across-multiple-aws-accounts-and-regions/

AWS on Steroids: CloudFormation Templates

Paul Marcelin
marcelin@alumni.cmu.edu

Thank you for attending.
Questions and comments are appreciated.

bit.ly/nocougcfn1