

Hello,

Here are our given tasks.

(1) Create a Redis cluster with 1 master and 3 slaves * The cluster should have the capability to promote slaves to master automatically in the event of master going down for whatever reason.

(2) Once complete please create a cloud formation template to launch the entire infrastructure and automate the whole process using chef recipes.

(3) Upload code to github and add a README with clear instructions on how to setup etc. Once complete please send us the link.

=> First of all we will create four EC2 Instances and from that 1 will be Master and 3 will be Slave servers.

=> After that we will install Redis Cluster into them and then configure as a master and 3 slave servers by following steps from below link.

Step 1 — Install Redis

Starting with the Droplet that will host our **master server**, our first step is to install Redis. First we need to add Chris Lea's Redis repository (as always, take extreme caution when adding third party repositories; we are using this one because its maintainer is a reputable figure):

```
) sudo add-apt-repository ppa:chris-lea/redis-server
```

)

Press `ENTER` to accept the repository.

Run the following command to update our packages:

```
)      sudo apt-get update
)
```

Install the Redis server:

```
)      sudo apt-get install redis-server
)
```

Check that Redis is up and running:

```
)      redis-benchmark -q -n 1000 -c 10 -P 5
)
```

The above command is saying that we want `redis-benchmark` to run in quiet mode, with 1000 total requests, 10 parallel connections and pipeline 5 requests. For more information on running benchmarks for Redis, typing `redis-benchmark --help` in your terminal will print useful information with examples.

Let the benchmark run. After it's finished, you should see output similar to the following:

Output

```
PING_INLINE: 166666.67 requests per second
PING_BULK: 249999.98 requests per second
SET: 249999.98 requests per second
GET: 499999.97 requests per second
INCR: 333333.34 requests per second
LPUSH: 499999.97 requests per second
LPOP: 499999.97 requests per second
SADD: 499999.97 requests per second
SPOP: 499999.97 requests per second
LPUSH (needed to benchmark LRANGE): 499999.97 requests per second
LRANGE_100 (first 100 elements): 111111.12 requests per second
LRANGE_300 (first 300 elements): 27777.78 requests per second
```

```
LRANGE_500 (first 450 elements): 8333.33 requests per second
LRANGE_600 (first 600 elements): 6369.43 requests per second
MSET (10 keys): 142857.14 requests per second
```

Now repeat this section for the Redis **slave server**. If you are configuring more Droplets, you may set up as many slave servers as necessary.

At this point, Redis is installed and running on our two nodes. If the output of any node is not similar to what is shown above, repeat the setup process carefully and check that all prerequisites are met

Step 2 — Configure Redis Master

Now that Redis is up and running on our two-Droplet cluster, we have to edit their configuration files. As we will see, there are minor differences between configuring the master server and the slave.

Let's first start with our **master**.

Open `/etc/redis/redis.conf` with your favorite text editor:

```
)      sudo nano /etc/redis/redis.conf
)
```

Edit the following lines.

Set a sensible value to the keepalive timer for TCP:

```
/etc/redis/redis.conf
```

```
tcp-keepalive 60
```

Make the server accessible to anyone on the web by commenting out this line:

```
/etc/redis/redis.conf
```

```
#bind 127.0.0.1
```

Given the nature of Redis, and its very high speeds, an attacker may brute force the password without many issues. That is why we recommend uncommenting

the `requirepass` line and adding a complex password (or a complex passphrase, preferably):

```
/etc/redis/redis.conf
```

```
requirepass your_redis_master_password
```

Depending on your usage scenario, you may change the following line or not. For the purpose of this tutorial, we assume that no key deletion must occur. Uncomment this line and set it as follows:

```
/etc/redis/redis.conf
```

```
maxmemory-policy noeviction
```

Finally, we want to make the following changes, required for backing up data.

Uncomment and/or set these lines as shown:

```
/etc/redis/redis.conf
```

```
appendonly yes
```

```
appendfilename redis-staging-ao.aof
```

Save your changes.

Restart the Redis service to reload our configuration changes:

```
) sudo service redis-server restart
)
```

If you want to go the extra mile, you can add some unique content to the master database by following the **Redis Operations** sections in [this tutorial](#), so we can later see how it gets replicated to the slave server.

Now that we have the master server ready, let's move on to our slave machine.

Step 3 — Configure Redis Slave

We need to make some changes that allow our **slave server** to connect to our master instance:

Open `/etc/redis/redis.conf` with your favorite text editor:

```
)      sudo nano /etc/redis/redis.conf
)
```

Edit the following lines; some settings will be similar to the master's.

Make the server accessible to anyone on the web by commenting out this line:

```
/etc/redis/redis.conf
```

```
#bind 127.0.0.1
```

The slave server needs a password as well so we can give it commands (such as `INFO`). Uncomment this line and set a server password:

```
/etc/redis/redis.conf
```

```
requirepass your_redis_slave_password
```

Uncomment this line and indicate the IP address where the **master server** can be reached, followed by the port set on that machine. By default, the port is 6379:

```
/etc/redis/redis.conf
```

```
slaveof your_redis_master_ip 6379
```

Uncomment the `masterauth` line and provide the password/passphrase you set up earlier on the **master server**:

```
/etc/redis/redis.conf
```

```
masterauth your_redis_master_password
```

Now save these changes, and exit the file. Next, restart the service like we did on our master server:

```
)      sudo service redis-server restart
)
```

This will reinitialize Redis and load our modified files.

Connect to Redis:

```
) redis-cli -h 127.0.0.1 -p 6379
)
```

Authorize with the **slave server's password**:

```
) AUTH your_redis_slave_password
)
```

At this point we are running a functional master-slave Redis cluster, with both machines properly configured.

Step 4 — Verify the Master-Slave Replication

Testing our setup will allow us to better understand the behavior of our Redis Droplets, once we want to start scripting failover behavior. What we want to do now is make sure that our configuration is working correctly, and our master is talking with the slave Redis instances.

First, we connect to Redis via our terminal, on the **master server**:

First connect to the local instance, running by default on port 6379. In case you've changed the port, modify the command accordingly.

```
) redis-cli -h 127.0.0.1 -p 6379
)
```

Now authenticate with Redis with the password you set when configuring the master:

```
) AUTH your_redis_master_password
)
```

And you should get an **OK** as a response. Now, you only have to run:

```
) INFO
)
```

You will see everything you need to know about the master Redis server. We are especially interested in the `#Replication` section, which should look like the following output:

Output

. . .

```
# Replication
role:master
connected_slaves:1
slave0:ip=111.111.111.222,port=6379,state=online,offset=407,lag=1
master_repl_offset:407
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:2
repl_backlog_histlen:406
```

. . .

Notice the `connected_slaves:1` line, which indicates our other instance is talking with the master Droplet. You can also see that we get the slave IP address, along with port, state, and other info.

Let's now take a look at the `#Replication` section on our slave machine. The process is the same as for our master server. Log in to the Redis instance, issue the `INFO` command, and view the output:

Output

. . .

```
# Replication
role:slave
master_host:111.111.111.111
master_port:6379
master_link_status:up
master_last_io_seconds_ago:3
master_sync_in_progress:0
```

```
slave_repl_offset:1401
slave_priority:100
slave_read_only:1
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0

. . .
```

We can see that this machine has the role of slave, is communicating with the master Redis server, and has no slaves of its own.

Step 5 — Switch to the Slave

Building this architecture means that we also want failures to be handled in such a way that we ensure data integrity and as little downtime as possible for our application. Any slave can be promoted to be a master. First, let's test switching manually.

On a **slave machine**, we should connect to the Redis instance:

```
) redis-cli -h 127.0.0.1 -p 6379
)
```

Now authenticate with Redis with the password you set when configuring the slave

```
) AUTH your_redis_slave_password
)
```

Turn off slave behavior:

```
) SLAVEOF NO ONE
)
```

The response should be OK. Now type:

```
) INFO
```


)

Look for the `# Replication` section to find the following output:

Output

. . .


```
# Replication
role:master
connected_slaves:0
master_repl_offset:1737
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
```

. . .

As we expected, the slave has turned into a master, and is now ready to accept connections from other machines (if any). We can use it as a temporary backup while we debug our main master server.

=> Now we will create a CloudFormation template of whole Existing Infrastructure by using CloudFormer tool by following below link.

Go to the AWS Management console and select “CloudFormation service”.
[Click on the “Create Stack” button.](#)

 **AWS** ▾ **Services** ▾ **Edit** ▾

Create Stack

Update Stack

Delete Stack

Filter: **Active** ▾

By Name:

	Stack Name	Created Time	Status	Descr
<input type="checkbox"/>	CloudFormation	2015-03-18 14:05:42 UTC+0550	CREATE_COMPLETE	AWS
<input type="checkbox"/>	AWSCloudFormer	2015-03-18 13:58:18 UTC+0550	CREATE_COMPLETE	AWS
<input type="checkbox"/>	dnsoutput	2015-03-02 22:55:18 UTC+0550	CREATE_COMPLETE	AWS
<input type="checkbox"/>	instancetype	2015-03-02 22:34:21 UTC+0550	CREATE_COMPLETE	AWS
<input type="checkbox"/>	apachenew	2015-03-02 21:01:09 UTC+0550	CREATE_COMPLETE	Creat
<input type="checkbox"/>	demo	2015-03-02 18:13:35 UTC+0550	CREATE_COMPLETE	Bootc

Overview

Outputs

Resources

Events

Template

Parameters

Tags

Stack Policy

Select a sta

Select the sample template as “CloudFormer”.

Select Template

Specify Parameters

Options:

Review

Select Template

Specify a stack name and then select the template that describes the stack that you want to create.

Stack

An AWS CloudFormation stack is a collection of related resources that you provision and update as a single unit.

Name CloudFormer

Template

A template is a JSON-formatted text file that describes your stack's resources and their properties. AWS CloudFormation stores the stack's template in an Amazon S3 bucket. [Learn more.](#)

Source

☒ Select a sample template

CloudFormer

☐ Upload a template to Amazon S3

Choose File

No file chosen


☐ Specify an Amazon S3 template URL

<https://s3-external-1.amazonaws.com/cloudformation-templates-us->

Cancel

Next

Specify parameters as per requirement.

 AWS

Services

Edit

Admin@ind-arbit

N. Virginia

Support

Select Template

Specify Parameters

Options

Review

Specify Parameters

Specify values or use the default values for the parameters that are associated with your AWS CloudFormation template.

Parameters

AccessControl

3.0.0.0/0


The IP address range that can be used to access the CloudFormation tool. NOTE: We highly recommend that you specify a customized address range to lock down the tool.

Cancel

Previous

Next

Tag the template with proper naming convention(for ease of use).

 AWS

Services

Edit

Admin@ind-arbit

N. Virginia

Support

Select Template

Specify Parameters



Options

Review

Options

Tags

You can specify tags (key-value pairs) for resources in your stack. You can add up to 10 unique key-value pairs for each stack. [Learn more.](#)

	Key (127 characters maximum)	Value (255 characters maximum)	
1	Name	CloudFormer instance for creatingTemplate	
2			

Advanced

You can set additional options for your stack, like notification options and a stack policy. [Learn more.](#)

Cancel

Previous

Next

Finally, click on the “create” button & the stack is ready.

The screenshot shows the 'Review' step of the AWS CloudFormation console. On the left, a sidebar contains links for 'Select Template', 'Specify Parameters', 'Options', and 'Review' (which is highlighted). The main content area is titled 'Review' and contains several sections:

- Template:** Displays the template name 'CloudFormer', its URL, a description, and an estimated cost of '\$0.00'.
- Parameters:** Shows a parameter 'AccessControl' with a value of '0.0.0.0/0' and a checkbox 'Create IAM resources' which is checked.
- Options:** Includes a 'Tags' section with a tag 'Name' set to 'CloudFormer instance for creatingTemplate'.
- Advanced:** Shows 'Notification' set to 'none', 'Timeout' set to 'none', and 'Rollback on failure' set to 'Yes'.
- Capabilities:** A warning box states: 'The following resource(s) require capabilities: [AWS::IAM::Policy, AWS::IAM::InstanceProfile, AWS::IAM::Role]. This template might include identity and Access Management (IAM) resources, which can include groups, IAM users, and IAM roles with certain permissions. Ensure that the template you are using is from a trusted source. [Learn more.](#)' Below this, a checkbox is checked with the text 'I acknowledge that this template might cause AWS CloudFormation to create IAM resources.'

At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Create'. The 'Create' button is highlighted with a red rectangle.

The highlighted Stack is the one which we have just created. The “URL” is the link of the CloudFormer Tool (it is running on an t1.micro instance in our AWS Account.

Information Management Console - Google Chrome

New Customer Log... CloudFormation M... A must read story...

https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?filter=active&tab=outputs&stackId=arn:aws:cloudform...

AWS Services Edit

Alice @ ttd-ankit N. Virginia Support

Create Stack Update Stack Delete Stack

Filter: Active By Name: Showing 6 stacks

Stack Name	Created Time	Status	Description
CloudFormation	2015-03-18 14:05:42 UTC+0550	CREATE_COMPLETE	AWS CloudFormer Beta - template creation prototype application. This tool allows you to crea
AWSCloudFormer	2015-03-18 13:58:18 UTC+0550	CREATE_COMPLETE	AWS CloudFormer Beta - template creation prototype application. This tool allows you to crea
disouput	2015-03-02 22:55:18 UTC+0550	CREATE_COMPLETE	AWS CloudFormation Sample Template EC2InstanceWithSecurityGroupSample: Create an A
instancetype	2015-03-02 22:34:21 UTC+0550	CREATE_COMPLETE	AWS CloudFormation Sample Template EC2InstanceWithSecurityGroupSample: Create an A
apachenew	2015-03-02 21:01:09 UTC+0550	CREATE_COMPLETE	Create a load balanced sample web site. The AMI is chosen based on the region in which the
demo	2015-03-02 18:13:35 UTC+0550	CREATE_COMPLETE	Bootcamp Demo Application using CloudFormation

Overview Outputs Resources Events Template Parameters Tags Stack Policy

Key	Value	Description
URL	http://ec2-54-174-164-128.compute-1.amazonaws.com	AWS CloudFormer Prototype URL. Use this endpoint to create ...

Click on the “create Template button” and proceed further.



AWS

You have gone full screen.

[Exit full screen \(F11\)](#)

Welcome to the [AWS CloudFormation](#) template creation utility. This utility helps you to create a CloudFormation template from the AWS resources currently running in your account using a few simple steps. While the created template is complete and can be used to launch an AWS CloudFormation stack, it is a starting point for further customization. You should consider the following:

- Add Parameters to enable stacks to be customized at launch time.
- Add Mappings to allow the template to be customized to the specific environment.
- Replace static values with "Ref" and "Fn::GetAtt" functions to flow property data between resources where the value of one property is dependent on the value of a property from a different resource.
- Use CloudFormation metadata and on-host helper scripts to deploy files, packages and run commands on your Amazon EC2 instances.
- Customize your Amazon RDS DB instance database names and master passwords.
- Customize or add more Outputs to list important information needed by the stack user.

Select the AWS Region US East (Virginia)

When you press "Create Template" we will analyze all of the AWS resources in your account. This may take a little time.

[Create Template](#)

Known Issues

- Amazon RDS database instances in a VPC are not currently associated with VPC security groups. You will need to manually add these to your template once it is created.

For more information on how to build a template see the [AWS CloudFormation User Guide](#). You can also check out our [sample templates](#) demonstrating various template features.

By default, the account credentials will be used from the entries you typed in when AWS CloudFormation was created; however, they can be overridden by clicking [here](#).

You will see a "Analyzing your account" screen & this step will take some time (Be patient).



Analyzing your account

Give the template a Description (as per your convenience).

* Do not select any resource as of now.

AWS CloudFormer



Template Information

Select the AWS region to introspect. The description is optional but will be displayed in the AWS Management console v stack. You can optionally enter a filter for the resources. If you specify a filter, all resources with a name or a tag value th selected automatically. Note that the filter is a case-insensitive match.

Template Description

Template to create infra with ec2 instance with some pre-installed packages.

Resource Name Filter

☐ Select all resources in your account

At first, it will ask you to select the VPC. So, all the related subnets, network interface & security group will be automatically selected (although we can customize it).

1) VPC



Now just keep clicking on “Continue” button. CloudFormer takes care of the dependencies.

In subsequent steps, it will ask you to select the following:

- 2) VPC Network (VPC Subnets, Internet Gateways, Customer Gateways, DHCP options)
- 3) VPC Security (Network ACLs, Route Tables)
- 4) Network (ELB, Elastic IP , Network Interfaces)
- 5) Compute (Auto Scaling Groups, EC2 Instances)
- 6) Storage (EBS Volumes, RDS Instances, DynamoDB Tables, S3 Buckets)
- 7) Services (SQS, SNS Topics, SimpleDB Domains)
- 8) Config (Auto Scaling Launch Configurations, RDS Subnet groups, RDS Parameter Groups)
- 9) Security (EC2 Security Groups, RDS Security Groups, SQS Queue

Policies, SNS Topic Policies, S3 Bucket Policies)

10) Optional Resources (AutoScaling Policies, CloudWatch Alarms)

At last, you will see a summary page with all your selected settings.

AWS CloudFormer Region us-east-1

Intro DNS VPC VPC Subnets VPC Internet Gateways VPC DHCP Options VPC Network ACLs VPC Route Tables Network Compute Storage Services Config Security Operational **Summary** Done

Summary

You have selected the following resources. We have automatically generated logical resource names for the template; however, you can assign your own names by editing them if you prefer. You can also select any output values that you want to return when the stack is created. Output values are displayed in the AWS management console when you select the stack in the AWS CloudFormation tab. To edit the logical name or to select output parameters, click on the modify link for each resource you want to change.

[Back](#) [Cancel](#) [Continue](#)

Amazon Virtual Private Clouds (VPCs)

vpc-10790575 [Modify](#)

Amazon Virtual Private Cloud (VPC) Subnets

subnet-1e463cc4 [Modify](#)

Amazon Virtual Private Cloud (VPC) Internet Gateways

igw-9d0185f8 [Modify](#)

Amazon Virtual Private Cloud (VPC) DHCP Options

dopt-f3f71196 [Modify](#)

Amazon Virtual Private Cloud (VPC) Network ACLs

acl-3edaab5b [Modify](#)

Amazon Virtual Private Cloud (VPC) Route Tables

rth-01256e64 [Modify](#)

CloudFront will display your template, click on “Save Template”.

AWS CloudFormation Template

You can save the AWS CloudFormation template in an existing S3 bucket in your account by selecting a bucket and clicking **Save Template**. Alternatively, you can cut and paste the template content below and store it locally or in your source control repository. Note: If you save the template to an S3 bucket in a different AWS region from the one used to create the template, launching it in the new AWS region will likely use hardcoded values based on the original AWS region.

Template Name

S3 Bucket

Save Template

Cancel

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "vpc10790575": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "172.31.0.0/16",
        "InstanceTenancy": "default",
        "EnableDnsSupport": "true",
        "EnableDnsHostnames": "true"
      }
    },
    "subnetfe463cc4": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "CidrBlock": "172.31.0.0/20",
        "AvailabilityZone": "us-east-1e",
        "VpcId": {
          "Ref": "vpc10790575"
        }
      }
    }
  }
}
```

On the next screen you will see,

Congratulations!

You have created an AWS CloudFormation template and saved it to S3. You can now launch stacks using the template in the AWS Management Console. Note: if you launch the template, it will be launched in the same region as the S3 bucket in which you stored the template. If you want to launch the template in a different region, you must create a new template in that region.

Create Template

Launch Stack



=> Now at the end you will get CloudFormation Template saved in S3 bucket.

=> You have to change the permissions for the template by allocating policy " Anyone " to view/edit. So now anyone with that link can access that template. Here is the link.

<https://s3-us-west-2.amazonaws.com/testnclouds/cloudformer.template>

=> Now for automating whole process we will write Chef Recipe like below.
