Architect Proposal

# Analysis of requirements

At the beginning, they have not enough experiences and budget on AWS. Hourly downtime is acceptable at beginning.

They want Invest iteratively.

# How

I will propose a road map to resolve their problems. And the last 2 architecture diagram are the final solution.

If they has enough experiences and budget, then can skip all the steps and directly to the final.

My proposal is iteratively and automated.

Stage environment can be the same with production but use the smallest instances.

You can follow the instructions below to migrate to AWS step by step:

## Migrate to AWS with minimal workload and cost

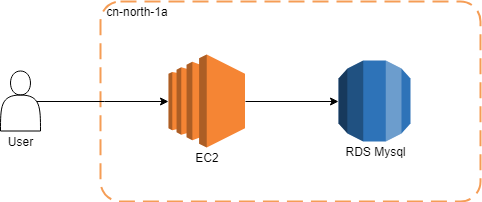
|  |  |  |
| --- | --- | --- |
| Options | Minimize | CDN powered |
| architecture diagram | C:\Users\DELL\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\97408B0B.tmp | C:\Users\DELL\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C221E0F0.tmp |
| Benefits | * Minimal workload and cost * Higher availability powered by AWS EC2 | * Very low latency by CloudFront(CDN) |

You can skip this step if you have enough experience and budget. Otherwise you can gather experiences at this step.

It is best to use [Elastic IP](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#Addresses:sort=PublicIp)(EIP) to do the DNS. You can have one Elastic IP (EIP) address associated with a running instance at no charge.

At this step, you can collect enough experience to the next step.

## Increase performance by decouple app server and DB server with RDS(will double the budget)



The [Amazon Relational Database Service](https://ap-southeast-1.console.aws.amazon.com/rds/home?region=ap-southeast-1) (RDS) offers ready-to-use relational databases. Automatically backup and updated database version, easily scale.

You don’t need Multi-AZ at this step because of it’s expensive and you can enable it at any time with a few mouse clicks. AWS will do the other works for you, like Master-slave backup.

### Steps:

1. Dump database to S3.
2. Create RDS instance.
3. Restore dumped data from S3 to RDS.
4. Create new EC2 by instructions below and config the database connection to RDS.
5. Switch DNS to new server.

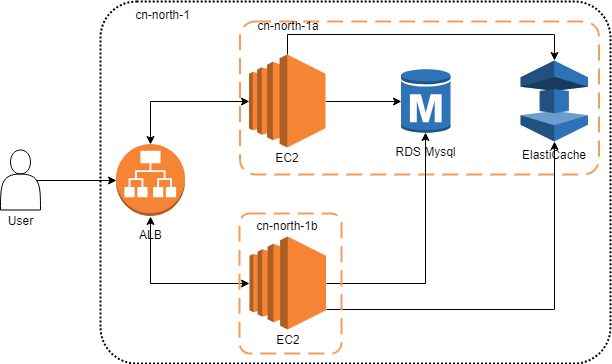
### You can automate configuration management for Disaster Recovery and productivity by [Elastic Beanstalk](https://ap-southeast-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-southeast-1#/welcome) or [OpsWorks Stacks](https://console.aws.amazon.com/opsworks/landing/choice?region=ap-southeast-1)

|  |  |  |
| --- | --- | --- |
| Options | [Elastic Beanstalk](https://ap-southeast-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-southeast-1#/welcome) | [OpsWorks Stacks](https://console.aws.amazon.com/opsworks/landing/choice?region=ap-southeast-1) |
|  | C:\Users\DELL\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DE945AA0.tmp | C:\Users\DELL\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\CA2762D.tmp |
| feature | Simplest | Flexible |
| Product intro | With [Elastic Beanstalk](https://ap-southeast-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-southeast-1#/welcome), you can **deploy**, **monitor**, and **scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business. | [AWS OpsWorks](https://console.aws.amazon.com/opsworks/landing/choice?region=ap-southeast-1) Stacks is a configuration management service that helps you build and operate highly dynamic applications and propagate changes instantly. |
| Use when | Usable | Elastic Beanstalk cannot fit your application because of some customization requirements. |

Both tools can be used on GUI, which have clearly guide and not need to read documents.

## Increase App server availability by load balancing

Powered by Elastic Beanstalk / OpsWorks, we can easily increase capacity and availability by adding an [ELB/ALB](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#LoadBalancers:sort=loadBalancerName) and a peer server:



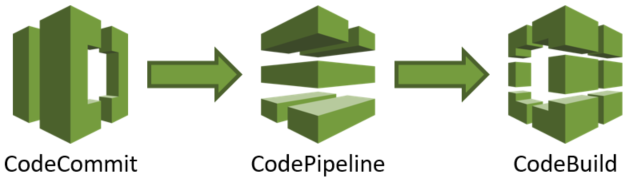
[Elastic Load Balancing](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#LoadBalancers:sort=loadBalancerName) can recreate your server by cooperate with Beanstalk/OpsWorks when the health check is failed (Disaster Recovery). In OpsWorks stack, if you want more flexibility you can try Time-based and Load-based Instances to manage load more flexible.

[ElastiCache](https://ap-southeast-1.console.aws.amazon.com/elasticache/home?region=ap-southeast-1) is a Redis web service. Refactor your application to stateless by persist states to [ElastiCache](https://ap-southeast-1.console.aws.amazon.com/elasticache/home?region=ap-southeast-1), and you can enable high availability by a backup server.

Attention:

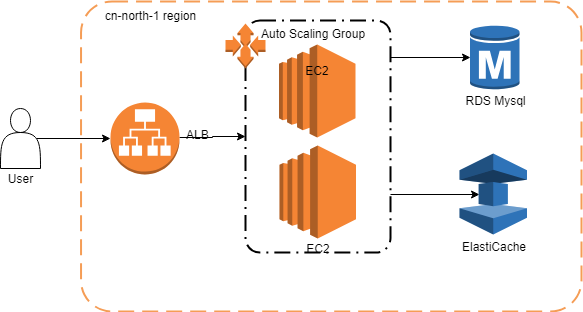
* We recommend to go with the modern Application(ALB) or Network Load Balancer because they are in most cases more cost efficient and more feature rich.
* But ALB may not well integration with the OpsWork stacks. You can use classic ELB.

## Continue delivery



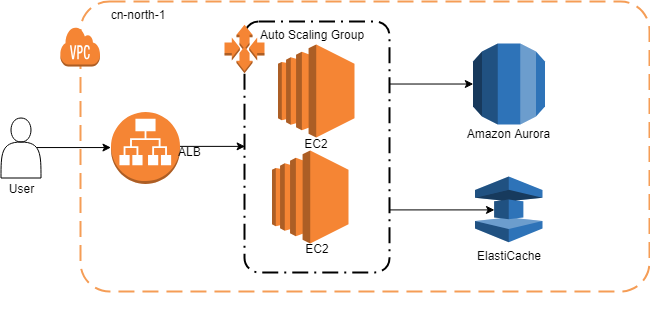
You can implement Continue Delivery pipeline by AWS developer tools like [CodeCommit](https://ap-southeast-1.console.aws.amazon.com/codecommit/home?region=ap-southeast-1#/introduction)(Git), [CodePipeline](https://ap-southeast-1.console.aws.amazon.com/codepipeline/home?region=ap-southeast-1)(CD), [CodeDeploy](https://ap-southeast-1.console.aws.amazon.com/codedeploy/home?region=ap-southeast-1), [CodeBuild](https://ap-southeast-1.console.aws.amazon.com/codebuild/home?region=ap-southeast-1)(CI).

## Increase app servers flexibility and fault-tolerance by auto scaling



[Auto Scaling](https://amazonaws-china.com/autoscaling/) ensures you have the correct number of EC2 instances available to handle your application load. You create collections of EC2 instances (called Auto Scaling groups), specify desired instance ranges for them, and create scaling policies that define when instances are provisioned or removed from the group. Auto-scaling is not by default available on OpsWorks, and there is no build in way to have an auto-scaling group associated with your OpsWorks stack, but it's possible with a bit of work. Read about it [here](https://aws.amazon.com/blogs/devops/auto-scaling-aws-opsworks-instances/).

## Increase Database flexibility and fault-tolerance by Aurora



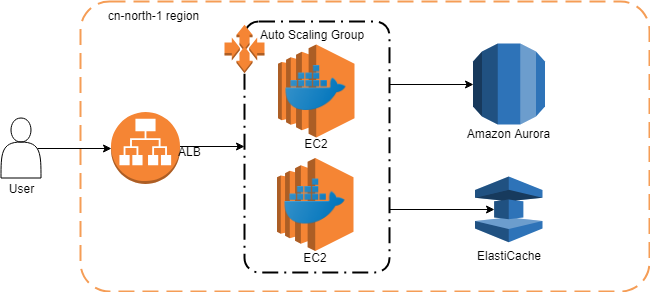
[Amazon Aurora](https://ap-southeast-1.console.aws.amazon.com/rds/home?region=ap-southeast-1) is a MySQL- and PostgreSQL-compatible enterprise-class database, starting at <$1/day. Aurora supports up to 64TB of auto-scaling storage capacity, 6-way replication across three availability zones, and 15 low-latency read replicas. It is less cost and complexity than multi-AZ RDS.

Enable Cluster and Multi-AZ enhance ElastiCache availability.

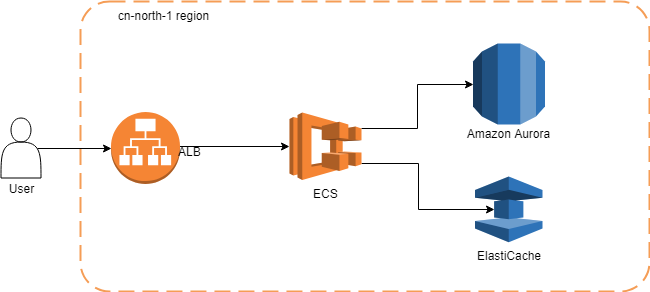
You can restore Aurora DB cluster from S3.

## Containerization by Docker

At first you can just docker your apps:

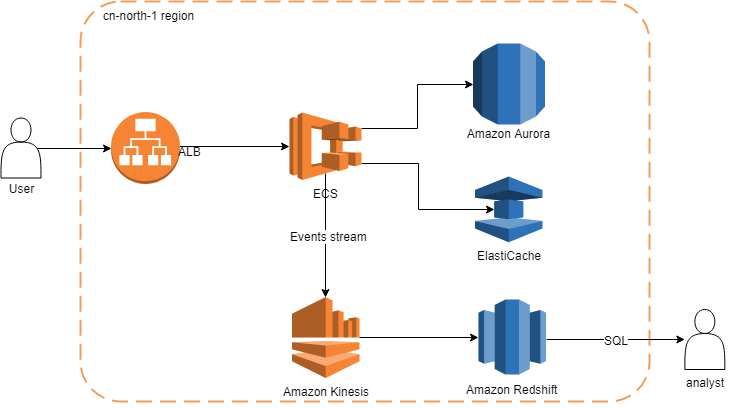


Next you can use Amazon Elastic Container Service ([ECS](https://ap-southeast-1.console.aws.amazon.com/ecs/home?region=ap-southeast-1#/clusters)) or Amazon Elastic Container Service for Kubernetes ([EKS)](https://amazonaws-china.com/cn/eks/) to increase resource utilization:



4-cores CPU EC2 is recommended.

## OLAP



A [Kinesis](https://ap-southeast-1.console.aws.amazon.com/kinesis/home?region=ap-southeast-1#/intro) stream is an ordered sequence of data records. To add data to a Kinesis stream, configure producers using the Streams PUT API or the Amazon Kinesis Producer Library (KPL).

You can use Lambda to do ETL if necessary.

Kinesis can be used for Asynchronous decoupling for your system, which is necessary for Microservices architecture.

## Enable microservice, Bigdata, Media, AI, IOT, AR & VR

With all preparations above. You can play microservice architecture with a lot more AWS services, like API Gateway, Lambda.