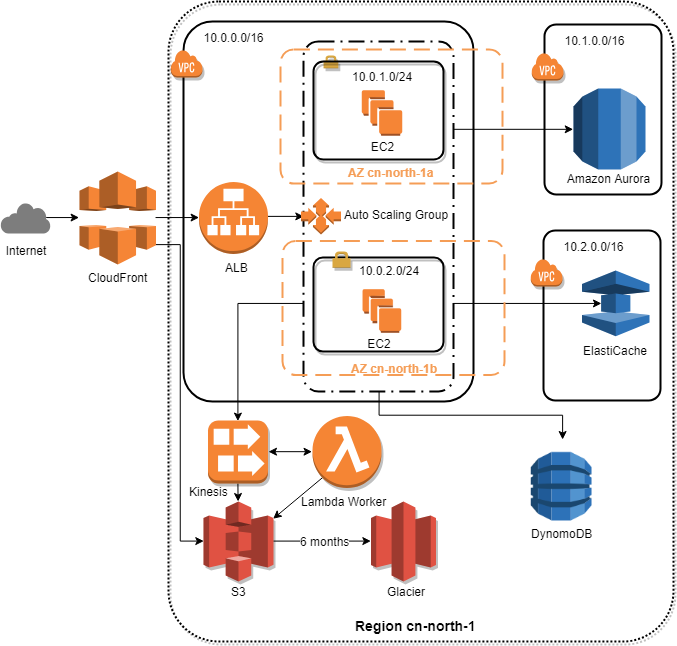
Architecture Proposal

Zhaolei 2018-05-28

# Architecture diagram



# Customer Concerns

## Scaling to meet the demand, but with uncertainty around when and how much this demand will be they are very concerned about buying too much infrastructure too soon or not enough too late

This is a cost concern.

[Auto Scaling](https://amazonaws-china.com/autoscaling/) and ALB can ensure the correct number of EC2 instances available to handle the application load. So, you don’t need to pay for unnecessary VMs.

With [ElastiCache](https://ap-southeast-1.console.aws.amazon.com/elasticache/home?region=ap-southeast-1), we can make the application stateless by storing the state data in cache, so you can distribute load to different EC2 instance and do not lost state data.

[Amazon Aurora](https://ap-southeast-1.console.aws.amazon.com/rds/home?region=ap-southeast-1) can ensures database always available and has enough storage and it is less cost and complexity than multi-AZ RDS. You only need to pay for your database by the hour with no long-term commitments or upfront fees.

The other services are also the same. You only need to pay for you used. And AWS has some free allowances per month which can cover a small amount of use.

## Disaster Recovery

Availability zones — Two are used. If one availability zone (AZ) suffers from an outage, we still have EC2 instances running in the other AZ.

Subnets — A subnet is tightly coupled to an AZ. Therefore, we need one subnet in each AZ.

EC2 instances — We have multi-redundancy for EC2 instances. We have multiple instances in one subnet (AZ), and we have instances in two subnets (AZs).

Regions — We can always deploy a same infrastructure on another region and stop all the instances in that region after we deployed new version of apps. And if the working region outage, we can start same stacks on the other region and switch the DNS. Because this is unlikely to happen at all, we can do this switch manually.

You can easily copy RDS snapshot to another region automatically by cli and daily task. DynamoDB snapshots should store to S3. S3 is cross region, highly available and highly durable naturally. With the backup on S3 you can recover from an unlikely region-wide outage.

Regularly switch region drill is recommended.

## Ability to configure database and data access layer for high performance and throughput

[Amazon Aurora](https://ap-southeast-1.console.aws.amazon.com/rds/home?region=ap-southeast-1) can ensures good throughput and performance by read replicas.

You can cache your query results in [ElastiCache](https://ap-southeast-1.console.aws.amazon.com/elasticache/home?region=ap-southeast-1), which can reduce replicated response time.

DynamoDB is a key-value store with document support (NoSQL). You can scale from one item to billions and from one request per second to tens of thousands of requests per second. To increase DB performance and throughput, you can make good use of NoSQL, for example store the frequently changed structure document data to DynamoDB. Then you can release relational DB (MySQL) to the areas where they are good at like ACID transaction.

## Making the user experience in the browser very low latency even though a large portion of their user base will be from far away

Using AWS CloudFront (CDN) helps reduce the load time for static web content.

You can cache replicated response in [ElastiCache](https://ap-southeast-1.console.aws.amazon.com/elasticache/home?region=ap-southeast-1) to reduce latency of replicated request.

Make good use of browser cache.

You can also add a SPA layer to reduce network traffic and increase app respond speed. But it may increase the initial load speed. You can use S3 to host the static web content and use CloudFront to deliver the content.

## Effective distribution of load

[Auto Scaling](https://amazonaws-china.com/autoscaling/) + ALB + CloudWatch + CloudFormation can ensure the load efficiency of your EC2 instances by Scale up and down. HealthCheckGracePeriod can be set to 5 minutes. You can use schedules to prepare instances for rush hours.

You can asynchronous decouple long-running operations with Kinesis and lambda. For example, image process. You can:

1. The user uploads an image to S3.
2. The web server puts a message onto Kinesis that contains a random ID and the URL.
3. The web server returns a link to the user where the PNG image will be found in the future. The link contains the random ID (e.g., http://$Bucket.s3-website-cn-north-1.amazonaws.com/$RandomId.png).
4. In the background, a lambda worker consumes the message from Kinesis, converts the content into a PNG, and uploads the image to S3.
5. At some point in time, the user tries to download the PNG at the known location. If the file is not found, the client retries in a few seconds.

## Self-healing infrastructure that recovers from failed service instances

If one of the EC2 instances crashes, the ALB stops to route requests to the crashed instances. The auto-scaling group replaces the crashed EC2 instance within minutes, and the ALB begins to route requests to the new instance.

## Security of data at rest and in transit

HTTPS can secure the data in transit. AWS CloudFront and ALB accepts HTTPS from the client and forwards the request to your HTTP only system.

RDS (Aurora) automated snapshots are created once a day. Manual snapshots can be triggered whenever needed.

In Aurora DB, data is protected by the database itself, security groups, and IAM. An IAM policy can denies all destructive actions to avoid data loss by human failure. Attaching the IAM policy to IAM users, groups, or roles controls which entity can use the policy to configure an RDS database. Only machines that really need to connect to the RDS database should be allowed to do so on the network level by security groups firewall rules.

DB engine access control should be used to:

* Limiting write access to a database to a few database users
* Limiting access to specific tables to a few users
* Limiting access to tables to isolate different applications

On-demand backups is recommended to create snapshots of your DynamoDB tables to be able to restore them later, if needed.

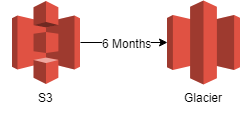
## Securing access to the environment as the delivery team expands

IAM Users for authentication, and groups to organize users’ authorization. Grant access only to the needed resources for each user.

You can also enable MFA for all users.

## Archival strategy for inactive objects greater than 6 months

Backup or structured data stored in S3 can be configured by a lifecycle rule to move objects to Glacier automatically:



|  |  |  |
| --- | --- | --- |
|  | S3 | Glacier |
| Storage Costs for a GB per month in US East (N. Virginia) | 0.023 USD | 0.004 USD |
| Costs for inserting data | Low | High |
| Costs for retrieving data | Low | High |
| Accessibility | Immediate upon request | One minute to twelve hours after request. Faster retrieval is more expensive. |
| Durability | Designed for annual durability of 99.999999999% | Designed for annual durability of 99.999999999% |

## Ability to easily manage and replicate multiple environments based on their blueprint architecture

OpsWorks and CloudFormation should be extensive used to manage configuration and infrastructures. With this work, your environments can be replicated automatically.

Also, the developer tools such as [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) should be used to achieve continue delivery your applications.

