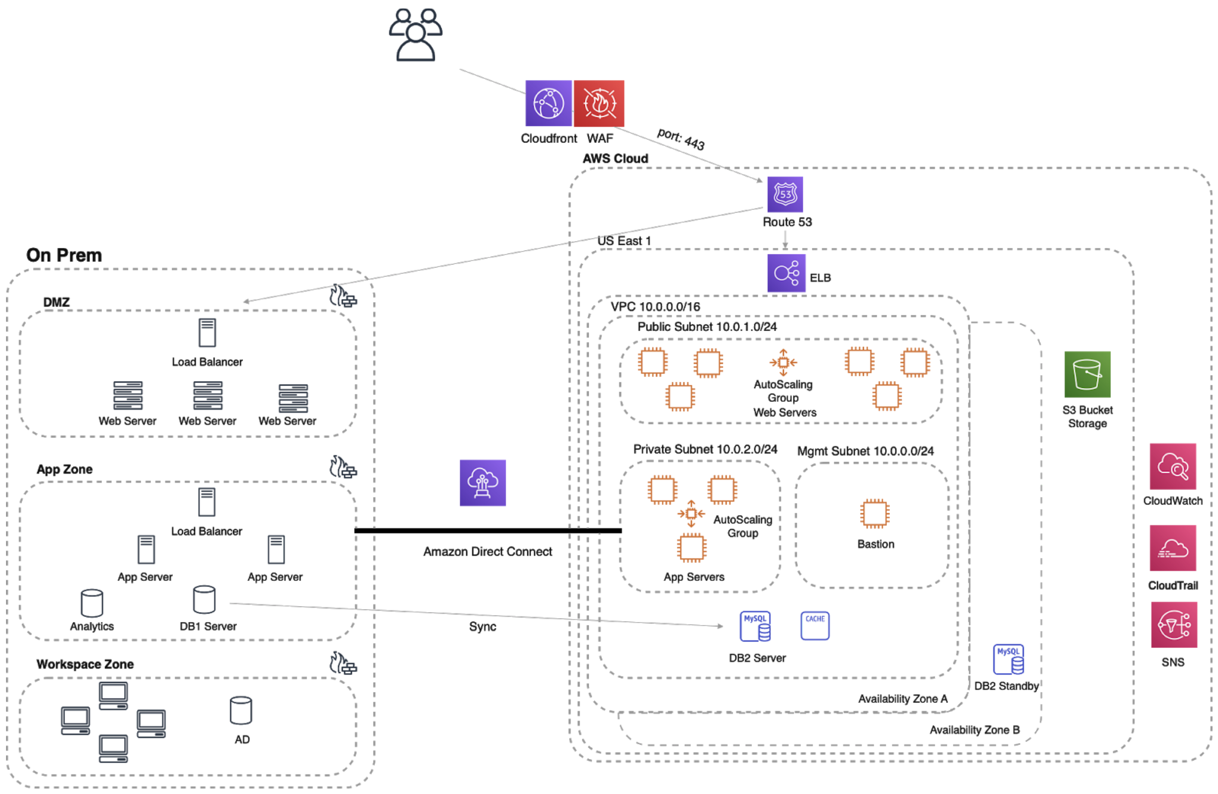
Reisz: Sr Infra Architect Architecture Design

Architecture



Key Assumptions

- Client wishes to maintain some of their DC operations with the ability to shift workload to the cloud “at their own pace”
- Client does NOT want a lift and shift solution in a short period of time. So we’ll move aspects of the architecture into production and start operating it. The goal is to gain confidence and experience along the way. This may lead to changes to the architecture.
- Client has DC operations experience and less on cloud. Training/Education/Gaining Confidence is a part of the success of the project
- Expecting “rapid growth in the next one to three months”
- No existing CDN/WAF/Bot Blocking
- Single DC (on prem today)
- Existing on prem architecture is NOT containerized uses physical machines or VMs
- Existing architecture is LAMP stack

Strategy

Most IT projects or cloud migrations that are not successful, fail because of people/organizational issues not technical ones. So the first step in this cloud migration strategy is to make sure leadership is onboard and questions answered. Quickly following that is to get the technical key influencers onboard. So we’ll spend time here on education/training.

In the writeup, it says “rapid growth in the next one to three months” First thing we want to do is get Route53 handling DNS. It will point right back to the existing A or CName record, but rather than through an existing registrar it will be through Route53. We do this so we can control where traffic is being sent and enable CloudFront/WAF/Bot Blocking.

In the absence of an existing CDN, CloudFront is an immediate and quick win. Depending on the read characteristics and update frequency we’ll see cache hit rates in the neighborhood of 80% (along with improving performance if we upgrade to HTTP/2, global presence, dedicated backbone, etc). CloudFront would fully front the existing site ***mostly*** as is. In addition, we can offer DDoS, Web Application Firewall (block IP ranges, ports, protect from Cross Site Scripting, OWASP Top10), and prevent some bad BOTS all with this step.

At this point, we have some options. I would engage leadership to decide what’s next. I would most likely setup the initial VPC/Subnets/Gateways/NAT/NACL/Security Groups so they can start to see the network topology (everything should be scripted into CloudFormation). The database replication/security is going to be a concern of their leadership (it would be for me if I didn’t have experience with it). I would go ahead and get it setup and replicating (on prem to the cloud). How we do this truly depends on the DB used and will take a bit of research. There are options that range from MySQL to full SQL service replication.

We use AWS Direct Connect to create a route between the on prem App Zone and the private subnet. We’ll use this same connection to enable RDS MySql replication. We’ll also need to failover of the DB instance (initially I have it shipping logs to a different availability zone in a warm backup but HA with Multi-AZ is a longer term goal). To answer the security question, Amazon RDS encrypts your db (data at rest) using keys management in KMS.

We can start to put objects into S3. They’ll be cached at CloudFront. Adapted services (later) would be replaced with services like SNS, CloudWatch, etc when it makes sense. We’ll use S3 in nightly batches to upload customer data. We’ll query it with Athena and write Lamda functions to process it on demand. We’ll store into an existing or a new datastore.

With the proven performance at the CDN/WAF and now a secured replicating DB, I’d begin to move their web and app tier into the cloud. The goal would be to integrate into their CI/CD pipeline and be able to push immutable images as deployable AMIs (longer term we might start the discussion of containers and EKS, but at this point we’re just moving onprem services/VMs to AWS). Once we have the images, we’ll set the web and app tiers up with autoscaling groups fronted by an ALB (Application Load Balancer). The ALB/Autoscalers will give us the ability to monitor our web/app tier and automatically adjust capacity to maintain steady, predictable performance at the lowest possible cost.

At this point, we would need to address any other services that the app requires (adapted services). I would either write code to point back to the on prem service or use an existing AWS service. The goal is to get a minimum viable cloud implementation up. Once we have that, we will go back to Route53 and push 1% of traffic through the cloud while keeping 99% of traffic going to on prem. We'll use this as a dial to control on-prem to cloud (we can eventually fully shift to cloud if desired).We'll need to setup equivalent logs/metics in CloudWatch so we can compare the performance of both solutions. We're looking for any issues.

Once we start to see load and ramp it up, we need to understand where the next constraints are (Theory of Constraint). If it's at the DB, we need to understand if it's write constrained, read constrained, or possibly just needs different resources. We can scale out RDS, use ElasticCache, or setup SQS to buffer writes. It depends on what we discover while operating the solution.

Discussion Questions:

How to scale their application to meet demand

Autoscalers and the ALB on the application side. ElasticCache, Read Replicas, and potentially SQS at the data tier.

How to address the lack of Disaster Recovery

RDS automatically backs up and stores to S3. On-prem to RDS replication setup and a standup in a different availability zone setup.

Effective distribution of load

Route53 and ALB/ELBs (used throughout).

How to architect for shared content storage, given the increased performance demands of the application?

CloudFront used for CDN services.

Latency of content served directly from S3 or storage attached to EC2

CloudFront used to cache large and small objects?

A self-healing infrastructure that recovers from failed service instances

AutoScalers, ALB/ELB. OnPrem/Cloud DB, backups, Hot/Warm DB (eventually HA with Multi-AZ), CloudWatch to observe, monitor, and alert.

Security of data at rest and in transit?

Move existing cert to Route53 and terminate AWS traffic routes there. For on prem, create a new cert to be used between to Route53 and the on prem gateway. ***Note: Longer term discussion about point to point encryption internal to the cloud.***

How to cost-effectively manage the ingestion of increased customer updates on AWS?

Depends on the update frequency, but a simple approach is to setup a nightly batch that updates CSV (even Apache Parquet in the future) into S3. Use Athena and Lambda functions to query and parse the data into MySQL or other datastore (such as DynamoDB).

How to connect their existing data center to the application on AWS, and then integrate this architecture with existing management systems?

We'll use Amazon Direct Connect to connect the two networks. A bastion as an alternative way to admin the private (and publics instances). Logs to CloudWatch