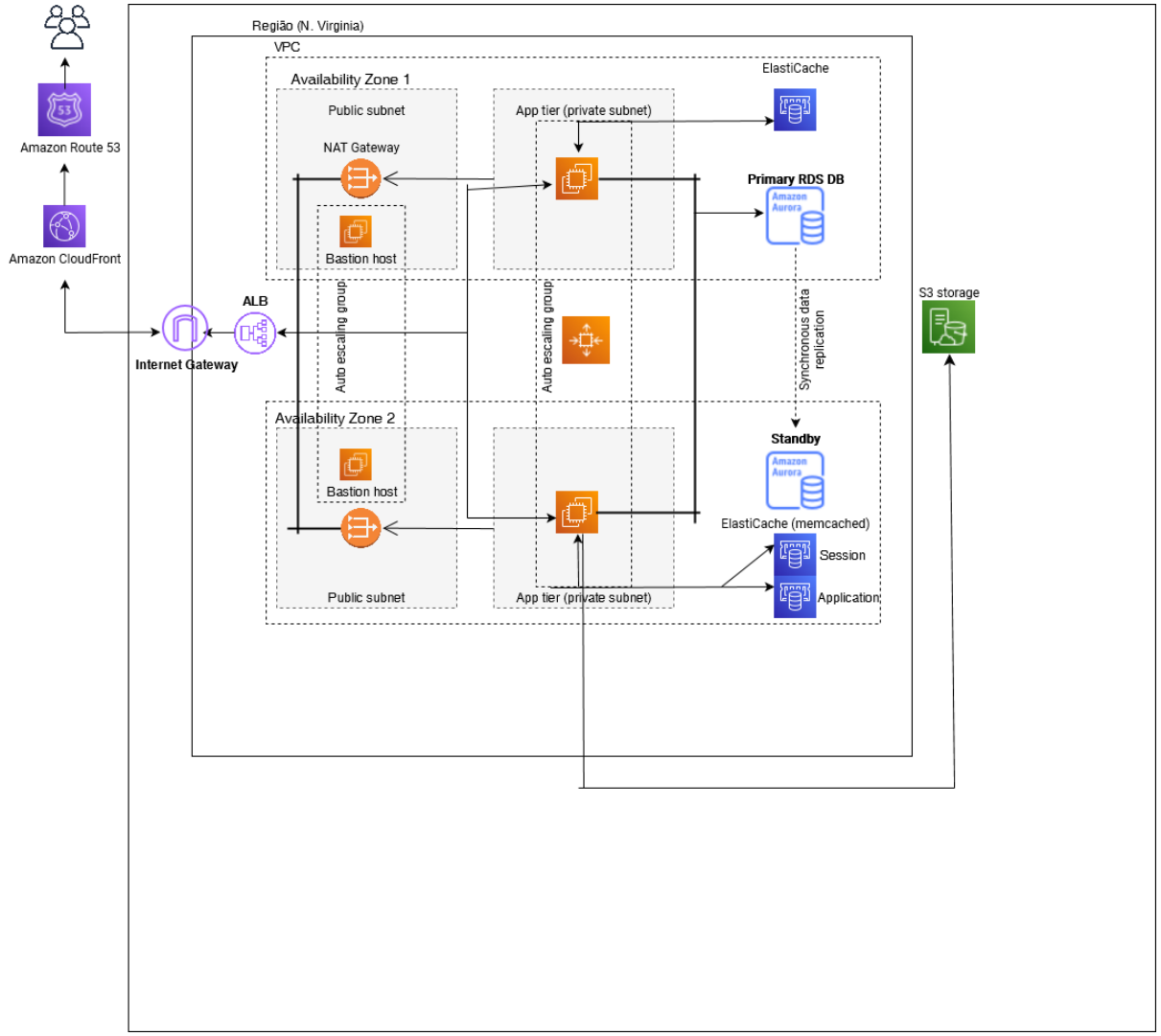

A instância escolhida para suportar o moodle de forma eficiente foi a c5.xlarge, que tem 8 Gib de RAM e 4 vCPU's com um SSD de uso geral de 50GB para arquivos de instalação do Moodle. Essa instância é a ideal considerando o custo-benefício, visto que ela é a opção mais barata ao mesmo tempo que oferece o processamento ideal para o contexto do cliente. Foi escolhido 50 Gib

De acordo com preços e orçamentos revisados, a escolha de plano mais coerente seria a combinação de On Demand e reservada, uma vez que o cliente tem planos de ficar por pelo menos 1 ano utilizando a AWS, seria prudente e mais barato usar um plano de reserva ao mesmo tempo que seja escalável para possíveis picos. Quando esses picos ocorrerem, o auto scaling da AWS agirá de forma eficiente aumentando

o número de instâncias conforme o uso, e quando os picos diminuírem o uso volta ao normal assim fazendo o custo diminuir bastante.

O plano também foi feito pensando no possível aumento de 50% dos alunos no segundo semestre do próximo ano, sendo que, de acordo com cálculos feitos (levando em conta os alunos perdidos e que a aplicação roda normalmente em 30% da capacidade máxima), serão necessárias em torno de no máximo 3 ou 4 instâncias quando a aplicação chegar ou exceder os 100% de capacidade, o que será feito automaticamente pelo auto scaling. A instância escolhida também dará conta do fluxo normalmente quando os 50% dos novos alunos chegarem, só precisando serem criadas mais instâncias quando a capacidade for bem maior que os usuais 30%.

Serviços presentes na arquitetura

De acordo com o desenho da arquitetura anexado, teremos vários tipos de serviços agindo em conjunto para tornar a aplicação sempre mais eficiente, levando em conta fatores como segurança, disponibilidade, escalabilidade e custos. Abaixo explicarei o papel de cada um:

Amazon Route 53: é um DNS na nuvem que converte nomes de sites (tais como www.example.com) para endereços IP numéricos que são lidos pelos computadores. No nosso contexto, será usado para direcionar os usuários para os pontos de menor latência, trazendo assim o mínimo atraso possível. Além disso, é possível criar domínios e também transferir registros DNS para domínios que existem. No nosso caso, ele será também o serviço que mapeará os domínios para o Amazon CloudFront (também é possível mapear diretamente para uma instância, para buckets do S3 Storage e outros tipos de serviços).

Amazon CloudFront: É um serviço que otimiza a entrega de conteúdo (arquivos html, CSS, Javascript e arquivos de imagem por exemplo). Contando com vários pontos de presença (PoP's), ele conecta o usuário ao ponto que oferecerá a maior agilidade no processamento, entregando assim o conteúdo da forma mais performática possível.

Internet Gateway: é basicamente o serviço que possibilita que o tráfego de rede entre na VPC. Sem um gateway como o Internet Gateway, a conexão via internet para a VPC não seria possível. Como será falado mais a frente, existe o NAT Gateway que é usado para conectar a VPC a serviços fora do seu escopo, porém não é algo mútuo, uma vez que ela não possibilita o inverso (agentes externos se conectarem a VPC) e é pra isso que serve o Internet Gateway.

ALB (Application Load Balancer): É o principal serviço que garantirá a disponibilidade e escalabilidade máxima na nossa aplicação (junto com os grupos de auto scaling). O Load Balancer distribui automaticamente o tráfego entre vários destinos (instâncias EC2, endereços IP e contêineres) entre uma ou mais Availability Zones (zonas de disponibilidade) tornando assim a aplicação mais eficiente, e também sempre usando Health Checks para garantir que o destino esteja pronto para receber aquela carga de trabalho. Há alguns anos, a Amazon lançou o ALB, que é um novo tipo de ELB com novas funcionalidades. Ela adiciona as várias funções já existentes o conceito de “target groups”, em que Listeners recebem solicitações e cargas de trabalho e decidem por meio de vários tipos de regras para qual target group eles enviarão aquela requisição. Os target groups então realizam o gerenciamento, dividindo o tráfego entre os destinos íntegros. Isso adiciona mais uma etapa e garante ainda mais a disponibilidade da aplicação.

Host bastions: É um serviço ou servidor que garantirá ainda mais segurança a aplicação em geral. O usuário se conectará ao host bastion, e então o bastian host se conectará de forma segura as redes privadas. Ele usa o protocolo SSH para conectar o usuário tanto na host bastion, quanto depois nas redes privadas, ou seja, uma dupla camada de proteção SSH. Fazendo uma analogia para melhor entendimento, é como se fosse um vigia de um prédio, analisando quem sai e quem entra do prédio. Ele fica na subnet pública, o que teoricamente seria algo perigoso, porém como vimos, nossa aplicação está bem robusta com vários serviços de segurança que impedirão um possível ataque de qualquer tipo.

NAT Gateway: serve pras redes privadas se conectarem a serviços fora do VPC (de forma segura, pois não é uma relação de troca, uma vez que serviços fora da VPC não podem se conectar de volta).

Auto scaling groups: são o outro serviço, além do ALB, que garantirão a escalabilidade da aplicação. Esse serviço escala a aplicação criando ou diminuindo a quantidade de servidores conforme necessário. Na criação dos auto scaling groups, você pode criar “Scaling Triggers”, onde você adiciona nessa parte as regras para criação de novos servidores ou diminuição dos existentes. Por exemplo, você pode dizer: se o uso das CPU’s ultrapassar os 80%, adicionar mais uma instância, e caso o uso seja menor que 20%, remover uma instância. Além da escalabilidade que o serviço garante, também há o fator econômico. Usar o auto scaling pode economizar muito dinheiro se usado corretamente. Ao colocar regras eficientes, o auto scaling usará a criação de servidores de forma econômica. Considerando que as instâncias são cobradas por hora, é muito importante esse uso consciente do serviço. Além disso, ele também age em conjunto com o ELB, uma vez que, ao serem criadas novas instâncias EC2 em um contexto que você criou, o ELB será responsável por distribuir essas novas instâncias.

Amazon Aurora (Banco de dados RDS): É o banco de dados escolhido para a aplicação do Moodle. Ele é compatível com o MySQL e com o PostgreSQL, tornando assim o uso de códigos usados nesses bancos totalmente utilizáveis no Aurora com quase nenhuma alteração. É possível migrar de e para o MySQL em uma hora facilmente. O Aurora oferece a performance e a disponibilidade de bancos de dados de nível comercial por um décimo do custo. O Aurora escala automaticamente se incrementando de 10 em 10 GB conforme demanda, podendo chegar até 128 TB. O replicamento do Amazon Aurora fornece alta disponibilidade, uma vez que milhares de réplicas são criadas a todo momento e podem ser transferidas para a instância principal caso haja alguma falha. Falando em falhas, ele possui um sistema de auto-reparação, que realiza inspeções a todo momento para diagnosticar e corrigir falhas.

Elasticache: é o sistema que “desafogará” o banco de dados. Ele armazenará dados passageiros na nossa aplicação, como sessões

ativas de alunos e professores. Usar o elasticache fará essas operações acontecerem em milissegundos, com uma latência ínfima. A escolha de dois memcached foi usada porque a documentação do Moodle considera que o Moodle trabalha bem melhor com um sistema de cache dedicado apenas para as sessões ativas e outra para a aplicação. (https://docs.moodle.org/39/en/Session_handling)

S3 (Sistema de armazenamento: o S3 foi escolhido como o sistema de armazenamento dentre vários outros (como EFS ou o próprio EBS) porque é o que mais se adequa ao contexto do Moodle. Uma vez que a aplicação possui algo em torno de 200GB de dados , sendo que 50GB são da aplicação em geral (que serão armazenados no EBS), os 150GB restantes são fotos e vídeos, o que torna o Amazon S3 a melhor opção. O S3 oferece disponibilidade, escalabilidade e segurança. Ele oferece diversos tipos de configuração, como por exemplo quatro camadas de acesso, sendo duas dessas contendo dados frequentemente usados e portanto são camadas com uma baixa latência e rápido acesso. As outras duas camadas podem ser de acesso assíncrono, para arquivos pouco acessados. Ele também tem opções de exclusão após certo tempo (exclusão de objetos que atingem sua “vida útil”). Oferece também uma camada de proteção a mais, uma vez que é possível configurar o bloqueio de objetos para alterações, tornando eles assim apenas legíveis e não alteráveis, além das proteções padrão, como bloqueio de acesso público aos buckets e objetos do S3 (caso você queira bloquear).

Especificidades do Moodle

Em alguns dos serviços, foram citadas algumas das especificidades para o funcionamento do Moodle na AWS, como por exemplo a recomendação do uso de um servidor compartilhado para o salvamento do cache de sessões. Além disso, também podemos citar o caso do armazenamento de dados e como a distribuição é feita para a melhor performance do Moodle, que não

trabalha bem normalmente com armazenamento compartilhado. Obs: o S3 foi escolhido ao invés do EFS porque os dois serviços podem armazenar o que é necessário e, após algumas análises de relatos constatei que o S3 pode ser mais eficiente que o EFS, que é muitas vezes relatado como “lento” pelos usuários. Abaixo está a distribuição que acarretará em mais eficiência:

$cfg->DirRoot = '/var/www/moodle/html' #Armazenado no volume raiz do EBS

$cfg->DataRoot = '/var/www/moodle/data' #Armazenado no sistema de arquivos S3

$cfg->cacheDir = '/var/www/moodle/cache' #Armazenado no sistema de arquivos S3

$cfg->TempDir = '/var/www/moodle/temp' #Armazenado no sistema de arquivos S3

$cfg->LocalCacheDir = '/var/www/moodle/local' #Armazenado no volume raiz do EBS

Essa configuração é importante porque alguns requisitos são necessários a respeito dos arquivos do Moodle, incluindo que devem estar em um ambiente fora do web server e deve ser criado um diretório vazio pra instalação do Moodle.

Ainda de acordo com a documentação do Moodle, quando ele é implantado como uma solução em cluster, o diretório **dirroot** deve sempre ser mantido no modo de leitura para o processo Apache, caso contrário, ao instalar ou desinstalar plugins ou **plugins**, os nós perderão a sincronização. Isso significa que os plug-ins não podem ser instalados em um cluster de servidor a partir da página do painel de administração.

Com essas coisas em mente, o Moodle operará de forma completamente performática e econômica na AWS.

Solução de monitoramento e deploy

O serviço de monitoramento escolhido para a aplicação foi o CloudWatch. O CloudWatch coleta dados operacionais e de monitoramento na forma de logs, métricas e eventos assim sendo possível analisar de forma minuciosa o que pode ser feito para aprimorar a arquitetura no geral. É possível definir métricas e alarmes para diversas situações, como quando sua aplicação bate algum valor específico (em dinheiro) e ele envia um email para você. Com milhares de dados por segundo, o CloudWatch correlaciona os mesmos e traz informações detalhadas. Ele tem integração com quase todas as soluções usadas na nossa aplicação, como o S3, o auto scaling e o EBS. Ele é capaz, por exemplo, de parar o auto scaling para reduzir custos no faturamento. Ele também pode agir com o S3 quando os arquivos passam de um determinado valor em GB. Também é possível observar dados entre regiões para saber se sua região está tendo um funcionamento satisfatório em relação as outras.

A solução escolhida para o deploy da aplicação é o Amazon CodeDeploy. O CodeDeploy é uma ferramenta de automatização das atualizações e lançamentos de novos recursos para a aplicação. O CodeDeploy conta com um fácil uso, tornando assim o ato de atualizar o software muito rápido. Ele automatiza tudo desde a atualização até possíveis erros, sendo possível observar tudo ao vivo através do console da AWS. Ele tem uma fácil integração com o Github, o que facilita muito também. Além disso, o mais interessante é a capacidade de atualizar sem precisar tornar a aplicação inativa

enquanto faz isso. Com o *In-place deployment,* o processo de atualização acontece de forma a atualizar as instâncias gradativamente e incrementando pouco a pouco rastreando a integridade dos aplicativos a cada mudança para garantir que a aplicação não terá erros pela atualização. Caso hajam erros, o CodeDeploy automaticamente reverte a atualização.