| General Information | |
|---|---|
| SoW created by | **Nour Samir** |
| Customer Name | **Appstract** |
| Project's Name | **Migration to AWS** |
| Infrastructure as code? | **Terraform** |

## The Main Purpose of the Project

1- Customer is migrating workload from Azure to AWS ( mainly 4 container services will be hosted on AWS ECS fargate and data from blob storage to AWS S3).
2- Customer is planning to migrate from MongoDB to AWS native service ( DocumentDB compatible with MongoDB)
3- Build CI/CD pipelines for auto-deployment
4- Cloudwatch for centralized logging and monitoring will be used to reduce ops overhead.
5- Apply best practices for running cloud-native workloads.

## Architecture diagrams

| Nr. | Diagram name | Link |
|---|---|---|
| 1 | Production environment | [Appstract - Architecture.png]() |

## Implementation tasks

| Nr. | Epic-level requirement | Sprint user stories | Time estimate | Description |
|---|---|---|---|---|
| 1 | Prod Account setup (In Scope) | | 1.5 | |
| 1 | | Enable access | 0.5 | Ask client to run Cloudvisor access CF template on their AWS account for this project<br>Enable access through Bitbucket automation |
| 2 | | Terraform bootstrap | 1 | Create S3 bucket for state and DynamoDB for state lock.<br>Create repository for Terraform code and ensure everything is ready for deployment. |
| 2 | Prod - Shared resources (In Scope) | | 4 | |
| 1 | | ECR | 2 | Creat ECR repositories for all 4 services<br>Enable scan on push on registry level<br><br>Check with the client:<br>- Name of each service |
| 2 | | IAM | 2 | Create IAM groups for users for client and share credentials.<br>Apply permissions trhough groups instead of attaching directly to users.<br>Enforce MFA on all users with access to console.<br><br>Check with the client:<br>- Name of users<br>- Minimum permissions for each user (ReadOnly, PowerUser, Admin)<br><br>Not needed in IaC |
| 3 | Prod - Networking (In Scope) | | 8 | |
| 1 | | VPC | 2 | - Create a new 3 tier VPC for Prod env.<br>- Deploy IGW, 2 NAT Gateway, Route Tables, and route associations. |
| 2 | | Bastion | 2 | Create a simple bastion host in public subnet<br>Create private key pair and save it to Secrets manager to allow client to connect<br>Enable SSH via Systems manager and Instance connect<br><br><br>Allow ingress on SSH from clients CIDRs" |

| | | | | |
|---|---|---|---|---|
| 3 | | Certificate | 2 | Create wildcard certificates and provide the validation requirements to the client.<br><br>Attach it to public endpoints - ALB. |
| 4 | | External DNS Integration | 2 | Provide customer with ALB DNS names and required CNAME record targets. Assist with creating or updating DNS entries in Cloudflare to point to AWS resources. Support ACM certificate validation steps if HTTPS is needed (via DNS or email validation). |
| 4 | Prod - Backend (In Scope) | | 22 | |
| 1 | | S3 - Static files | 2 | - Create 2-3 S3 buckets to migrate customers objects from Azure Blobstorage to AWS S3.<br>- Keep the block public access configuration enabled.<br>- Code changes on the app will be required here, point it out to the customer.<br><br>https://aws.amazon.com/blogs/storage/migrating-azure-blob-storage-to-amazon-s3-using-aws-datasync/<br>or<br>https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/migrate-data-from-microsoft-azure-blob-to-amazon-s3-by-using-rclone.html |
| 2 | | Secrets Manager | 2 | Store application secrets, database credentials, and API keys in AWS Secrets Manager.<br>Attach policies to allow ECS tasks access to specific secrets.<br><br>Check with the client:<br>- What should be stored in secrets |
| 3 | | ALB | 4 | Create Application Load Balancer with listeners for HTTP/HTTPS. Configure target groups for ECS services. Set up listener rules for routing to frontend and backend based on URL paths. Attach ALB to public subnets. Configure health checks.<br><br>Configure rules to:<br>- Return 403 by default<br>- Configure path-based routing rules.<br>- Add the newly created ACM certificate to the ALB.<br><br>Use ALB for: External-facing services, services needing health checks, path-based routing, SSL termination<br>Skip ALB for: Internal services, background workers, database containers, queue consumers |
| 4 | | ECS Cluster | 6 | Create a single ECS cluster with Fargate.<br>- Service will be in private subnet<br>- Container insights enabled<br>- Enable Cloudwatch logs agent<br>- Enable awslogs log driver |
| 5 | | ECS Service Configuration – Backend | 4 | - Create ECS task definitions for all Backend services (~4 services)<br>- Define CPU, memory, port mappings, environment variables, secrets integration.<br>- Deploy ECS services integrated with ALB target groups.<br>- private subnets and update ECS task definitions to connect to RDS via endpoint/Secrets Manager.<br>- ASG with sns alarms are set for the ECS services ( min =1, max=3)<br>- Check with customer the number of CPU and MEM required for each service, serivce ports, environment variables and logging confirgurations. |
| 6 | | DocumentDB | 4 | Create DocumentDB Instance-based cluster with 2 nodes and allow connectivity from private subnets.<br>Enable deleteion protection and performance insights.<br><br>Provide connection details to client and services through Secrets manager. Customer will start migrating their Data From MongoDB to DocumentDB<br><br>https://aws.amazon.com/tutorials/move-to-managed/migrate-mongodb-to-documentdb/ |
| 5 | Prod- CI/CD (In Scope) | | 2 | |
| 1 | | ECS tasks | 2 | Configure Azure DevOps pipelines to authenticate to ECR and push Docker images. Add deployment step using AWS CLI or ECS deploy tool to update ECS services. Use IAM user or OIDC for secure access. Document and test deployment.<br><br>Production Workflow: Triggered on merge to the master branch → pushes Docker image to production ECR repository.<br><br>Trigger automatically on merge to that branch. CI part should be handled by client (building the Docker image), we only deploy it to ECR. Deploy using 2 tags - 'latest' and Git commit hash.<br><br>This task will be done through guidance and customer will do the implmentation and we provide commands and guidance. |
| 52 | Monitoring Prod (guidance) | | 1.5 | |
| | | Central Cloudwatch dashboard | 0.5 | Setup dashboard for key infra metrics like ECS, ALB, DocumentDB,..etc . Implement from the console<br>Use Automatic dashboards. |

| | | | | |
|---|---|---|---|---|
| | | Clouwatch Alarms | 0.5 | Create basic alarms for the serivces used with sns notifications. Implement from the console |
| | | SNS Topic | 0.5 | Create SNS topic for the Cloudwatch alarms using terraform. |
| 53 | Handover | | 0 | |
| 1 | | Documentation | 0 | Baseline documentation in form of Terraform code is readily available. Additional documentation can be created on demand. |
| 2 | | Handover workshop | 0 | Organize a meeting with technical staff and go through AWS infrastructure and other resources. Answer any questions that may come up. |
| | | | | |
| | | **Project Total** | 39 | |
| | | **Total with buffer:** | 47 | |

**Out of scope**

| **Nr.** | **Item name** | **Comments** |
|---|---|---|
| 1 | DNS Change | Customer will handle DNS record changes in GoDaddy |
| 2 | Building Docker images | We expect CI pipeline to be built by the client<br>Customer said currently everything is Dockerized |
| 3 | WAF custom rules | The WAF will be deployed using the basic rules, any additional custom rules will be out of scope. |
| 4 | Secretes rotation | Secrets manager rotation policies will be handled by the customer. |
| 5 | Any custom ALB rules, only service routing is in scope | |
| 6 | Improving Application Code | Application code refactoring or debugging to match the aws services. |
| 7 | Manual Pipeline Adjustments | Customer is responsible for configuring BitBucket CI/CD logic (triggers, builds, test stages). We will provide deployment related guidance only. |
| 8 | Migrating data or database Schema Management | We create the infrastrucutre like s3 or RDS, guide on how to access but moving data there is customer responsibility. |
| 9 | Validating data migrated | Customer to verify that data is migrated correctly |
| 10 | Changes to existing production services | |

**Assumptions and constraints**

| **Nr.** | **Item name** | **Comments** |
|---|---|---|
| 1 | PROD is Frankfurt region | |
| 2 | No existing AWS accounts | We will create a dev and prod accounts for the customer |
| 4 | Customer will provide Dockerfiles and app code. | We will use the pipeline to push docker images to ECR and auto deploy to ECS |
| | Customer will provide us with the routing rules and domain for each ECS service | |
| 5 | Backend and Frontend Fully Functional and production ready | |
| 6 | Customer will provide all ECS services details like port number, health checks paths, env vars and secrets. | Customer will help us with the services and tasks details to deploy them using the right configurations for each service. |
| 7 | The provided CPU and RAM usage in the AWS Calculation is correct | Need to verify though |

**AWS Services Cost Breakdown**

| **Nr.** | **Environment** | **Estimated Monthly Costs** | **Estimated Annual Costs** | **Link to AWS calculator** |
|---|---|---|---|---|
| 1 | Staging | $1,106.33 | $13,275.96 | |
| 2 | Production | $446.60 | $5,359.20 | AWS Pricing Calculator |
| | **Total** | **$1,552.93** | **$18,635.16** | |

**AWS Services Cost Breakdown [ Internal Only ]**

| **Nr.** | **Environment** | **Estimated Monthly Costs** | **Estimated Annual Costs** | **Link to AWS calculator** |
|---|---|---|---|---|
| 1 | Staging | $590.73 | $7,088.76 | |
| 2 | Production - EU | $1,418.20 | $17,018.40 | AWS Pricing Calculator |
| | **Total** | **$2,008.93** | **$24,107.16** | |

**Risks**

| Nr. | Risk name | Mitigation actions required | Priority | Impact level | Probability level | Owner |
|---|---|---|---|---|---|---|
| 1 | AWS knowledge for handover | We can provide additional know during handover | High | medium | High | |
| 2 | Deploying 4 services to ECS | Customer guidance and allocate and debugging | High | High | High | |
| 3 | Some usage unknown | | Low | Low | Low | |
| 4 | Docker image issues | Adjust application for AWS | medium | medium | Low | |
| 5 | External DNS dependency (GoDaddy) | | medium | medium | Low | |
| 6 | Azure DevOps Pipelines | Cloudvisor provide the needed commands and guidance from aws side | medium | Low | medium | |