Migration Strategy

Migration Plan

- Organising Projects, Billing and Roles
- Setting up VPC
- Managed and Non managed Services
  - Database Migration / No-SQL (MongoDB) Migration
- VM Migration – Standalone and Managed Instances
- API Gateway / Endpoint
- Jobs – Adhoc and Event Based
- Hardening the Production setup – Load Balancer, Certificates, WAF etc
- Logging - Central Logging and Backups
- Monitoring – Setting up alerts on CPU usages etc
- CI/CD Pipeline - Continuous integration and deployment

Organising Projects, Billing and Roles

Plan & Execute

- Define an AWS Organization to centrally govern your environment
- Create a root organization and sub organizations to manage projects

Best practices

- Create multiple projects (i.e. development, testing, production) and client specific projects.
- Don’t allocate any resources to the root account
- Get visibility of billing per project and setup alerts for budget controls.
- Use Principle of Least Privilege with IAM (i.e. allocate minimal access to perform tasks)

Setup VPC

Plan & Execute

- Setup VPC for resource isolation
- Create unique subnets (i.e. CIDR range) for all the projects.

Best practices

- Setup VPC peering for sharing of resources across projects. (CIDR block cannot conflicts with peer VPC, so follow point 2 above as a best practice)
- Keep all VPC instances in private subnets and front end through a Load Balancer.
- Plan for at least 2 availability zones (and more based on your requirements) for fault tolerance/scalability.
- Setup network ACLs as firewalls to control inbound and outbound traffic at the subnet level.

Managed and Non Managed Services

Plan & Execute

- Plan for Database Migration – what versions of managed database (i.e. RDS) is available.
- Plan for non managed services – (i.e. Choosing MongoDB vs DynamoDB)

Best practices

- Setup up high availability for RDS and periodic database backups.
- Change default DB Parameter Groups group based on your RDS instance (i.e. logging query > 2 second etc) . This requires a DB restart.
- Setup monitoring alerts on DB (i.e. maximum db connection etc)
- Explore and leverage CloudFormation Templates for Setup (i.e. setting up a MongoDB production cluster) for non managed services.

VM Migration – Standalone and Managed Instances

Plan & Execute

- Plan for Standalone VM Migration – Exporting docker images to Amazon ECR registry
- Plan for Managed Instances – Create Auto Scaler Group, Launch configuration, Heath checks and Load Balancer configuration.

Best practices

- Create your Base AMI which can be used as a template for Auto-Scalable Launch configuration.
- Create Health configurations based on services (i.e. APIs, websites etc)
- Setup monitoring alerts for Auto Scaling (i.e. number of active instances, CPU usages etc)

API Gateway / Endpoint

Plan & Execute

- Migrate API gateway to AWS API Gateway - API Gateway supports importing and exporting APIs using the OpenAPI 2.0/3.0 API specification
- Create Routes and Integration targets (i.e. LoadBalancer).

Best practices

- Leverage CloudWatch metrics and CloudWatch Logs to monitor HTTP APIs (enable detail monitoring)

Jobs – Adhoc and Event Based

Plan & Execute

- Adhoc and Event based applications/Jobs – Leverage AWS Serverless Capabilities (Lambda, AWS Fargate etc)
- Create deployment package for Lambda. For containers, leverage AWS Fargate.
- Integrate Lambda functions for events (i.e. trigger Lambda functions on files uploads in AWS Bucket)

Best practices

- Package all of your dependencies (including SDK) with your deployment package.
- Understand how costs are calculated - (i.e. don't use recursions)
- Link to best practices - https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html

Hardening the Production Setup

Plan & Execute

- Understand what additional AWS services that you can leverage for hardening your production instance - Elastic Load Balancer, AWS Certificate Manager, WAF (Web application Firewall) etc
- AWS Certificate Manager provides free public certificates to be used with Load Balancers and API Gateway.

Best practices

- Run production in a separate sub organization/project. Restrict access to the sub organization (i.e. create policy and apply to suborganization to provide read only access)
- Configure multiple availability zones and corresponding public/private subnets.
- Enable CloudTrail for compliance and audits
- Create and review Security groups and Network ACLs.
- Use bastion host and ssh tunnelling for secure connectivity.
- Creating enough alarms events based on metrics (i.e. latency for APIs, 5xx errors, DB active connections, Lambda execution time etc)

Logging, Monitoring and CI/CD

Plan & Execute

- Plan if you need centralized logging .By default, AWS services (Lambda, RDS, API Gateway etc.) logs to AWS CloudWatch
- Application logs from VM can be logged via installing the CloudWatch agent. CloudWatch agent can be installed in your base AMI.
- Use your existing CI/CS tooling (Jenkins, GitHub etc) and push images to AWS ECR. Deploy images from ECR. If you are building from scratch, look at AWS Offerings like CodePipeline

Best practices

- Create CloudWatch alarms for sending alerts based on metrics (i.e. high/low CPU utilization, API latency etc).
- Check retention policy for AWS CloudWatch logs. Create lifecycle policy for logs archival.