Cloud Migration Principles

Recommendations & General Guidelines To Follow When Migrating To Cloud.

ATMECS™
Passionate Minds

Contents

1. Overview

This is a document which provides general guidelines to consider when migrating to Cloud. Although this document gives examples/notes that are relevant to AWS (Amazon Web Services), you should be able to apply these recommendations to other cloud services as well, such as GCP, Azure etc.

Also, some of the tools mentioned in this article belong to a generally used tool set. Please use the tools of your choice, based on your organization's needs and standards.

2. Standards Required: Naming Convention

Questions:

1) Having difficulty looking at AWS Console and trying to figure out who owns what resources (EC2, RDS, EBS, S3 etc.)? Need a better way to easily reconcile AWS resources with true owners?

2) Even if there is a way to figure out the owners of the AWS resources, it's all very manual. One would have to go look into each resource by name and then try to identify the owners by using their name etc. There is no consistency at all.

3) Would you like to know a better way of enforcing some rules in terms of creating and identifying AWS resources? Need a way to notify the users if they are not being compliant?

Aspect	Recommendation	Exception
Users, Groups, Roles etc.	● Use the organization's standard naming convention	No Exception
Name Tagging	● All resources must be tagged. Suggested tags: Name, Product/Group + more ● Resources include: EC2, S3, RDS ● The organization standards must be published for broader visibility	No Exception
Enforcement	● Develop an AWS CLI tool that scans and reports resources and objects that are Out of Compliance - monthly report to management ● Sending warning notes by the responsible person to the Product Manager. After two warnings, Ops team should lock-down the resource i.e., remove access	For Production instances, removing access might not be an option. The organization should have a compliance meeting where the responsible person must answer

Standard: Access & Cost Control

Questions:

1. Would you like to have a better command on the total AWS bill?
2. Would you like to have a better control on:
    a. Who is costing what?
    b. What are the $ expenses for Dev vs Prod etc.?
    c. How to have a mechanism to set some upper limits on the monthly cost and then get proactive notification if the cost is going to go over?
3. Would you want to have a clean and better control - to only let authorized Service Delivery personnel to access PROD resources. You don't want to mingle Dev and Prod - for both Security and Cost perspectives?
4. Have you seen instances where Developers, possibly inadvertently, spinning up AWS resources and not terminating them after usage? Need a better

control to limit and force some agreed resources upon limits? Need a better governance model?

5. Not all resources (especially EC2) are utilized 24x7. However, most/all of them left running all the time. Is there a better way to control this cost by running the EC2 only when needed?

Aspect	Recommendation	Exception
AWS Accounts	● Have two separate AWS Accounts: Dev/QA on one account, Staging/Prod on a separate AWS Account ● It would be easier that way to limit PROD access to a limited authorized personnel	No Exception
Develope r Limits	● Each Developer is allowed to certain limits per user. For example: ■ RAM – 10 GB total ■ Storage – 50 GB (EBS) ■ CPU – 8 vCPUs Develop an internal UI/App to let developers spin-up new instances. Not via AWS Console or AWS CLI. This would also help check/validate developer limits before allowing new instance spin-up	
Dev Instance Limits	● Developers who fall under this category SHOULD be limited to instances: ■ Size: mx. larger or lesser ■ RAM: 8 GB max ● Developers must be allowed to launch instances only in specific availability zones or regions. For example, us-east-1c, us-west-2b only. By doing so, automated scripts can manage the resources' up/down time based on the allowed guidelines. ■ For example: shutting down instances after 8 pm Pacific etc.	

EC2 instance types	● Dev – Use General Purpose ● Prod – Compute or Memory Optimized Enforcement: Generate a weekly report of all non-compliant instances and then have the responsible team to provide justification	Only servers used for specific reasons, such as Performance testing, which need to be modeled as close to PROD as possible

Standard: Security & Access Control

Questions:

1. Need a well-established Security compliance and governance model?
2. Allow only a limited number of authorized personnel to have view/edit privileges of global console access.
3. Need to make sure there's governance in AWS recommendations such as IAM, Keys rotation etc?
4. Need to make sure if any third-party app/vendor is brought in and whether their products/services are also compliant with the organization's standards?
5. Want to make sure there's a clear strategy in terms of Data Protection?
6. Want to make sure the governance and compliance are automated as much as possible by utilizing AWS tools and organization's tools? Do not rely on manual steps. If possible, eliminate those.

Aspect	Recommendation	Exception
AWS Metrics	• Define Metrics that are to be used for Operations and Developer debugging • Expose Metrics to the corresponding Dev Leads	
AWS Console Access	• PROD - NO AWS Console Access to anyone, except for authorized Service Delivery personnel • Dev – Limit AWS console access to only Tech Leads	PROD - Dev Leads to have Read-only access for debugging purpose. If this is to be done, this must be tied with existing process, such as User Access Management
Up Time	• Prod instances must be available 24x7. However, they must use Auto-scaling feature to scale down the number of running instances during off-peak business hours (when possible) • Dev instances should be automatically shut down via Autoscaling or AWS CLI. Suggested uptime is 8 am – 8 pm PDT • QA engineer servers should be automatically shut down, similar to Dev. However, QA servers that are running offline integration or long running tests need not be shutdown automatically. Use Tagging feature for machine classification. Enforcement: Ops team to send out warnings about non-compliant instances. After two warnings, remove the access privileges to the instances (via SGs)	Dev teams must account for saving their work. This is also a good way to develop "stateless" design.

AWS Keys	• Development teams MUST establish a standard practice to identify which key is used where. Use of naming convention itself might not be sufficient, if the developers are reusing the same key at different locations • Keys must be setup using external properties files, must not be hard-coded in the code base directly • Must rotate the AWS Keys every 3 months • For Prod, the keys must be configured at runtime. Do NOT expose the keys via properties files etc.	Maximum 2 months of additional time is allowed for key rotation This is to be enforced by Service Delivery personnel by providing one month notification to Dev teams
No Unwanted Access	• Check for Open SGs 0.0.0.0/xx • Check for services available in Public subnet • No public “S3” buckets, unless authorized • ADD: governance process to validate – (i.e.) weekly scans and reporting	
User Management	• Make sure the procedures are compliant in terms of users changing Dev groups, leaving the organization etc.	There MUST not be any public key that can be used to access AWS resources outside the organization network

Databases

Questions:

1) Would you want to move away from maintaining your own DB instances because it's too much of an admin and maintenance headache? Do you want to be relieved of this IT/DBA work, so you can focus more on other critical things?
2) Conglomeration of databases takes too much time to maintain/administer them and also demands constant training of IT Staff on these new database types. Can we enforce some of the organization's standards on the persistent data stores?
3) Backup and Restore – Do you want to establish a better backup strategy for your databases as well as an SLA to be able to restore the database, whenever needed?

Aspect	Recommendation	Exception
DB Type	• All RDBMS instances must use RDS except for the cases where approved databases are needed because of legacy needs.	Only when a particular DB is using fine-tuned features of underlying OS etc, that can't be supported by RDS. Even if it does, this must be reviewed by Architecture and Service Delivery teams and have a sign-off.
No SQL	• Use the organization-approved NoSQL DBs only. For example, support only DynamoDb, MongoDb	PoC for innovation/ideation
S3	• Define proper Bucket and Folder naming conventions. Use AWS CLI to report out non-compliant entities	Add S3 specific standards – security and operational

Backup and Restore	● Use AWS DB snapshots to take snapshots. Make sure the snapshot time is appropriate	

Service Delivery (SD)

Questions:

1. Repeatability through automation – Once a setup is done, do you want to make sure it can be replicated through automation by not relying on manual memory/steps?
2. As an SD person, one wants to make sure they can get good visibility and monitoring of the systems/resources that they are responsible to manage.
3. As an SD person, one wants to make sure they use as much as possible AWS tools/scripts as opposed to develop their own custom tooling.
4. Do you want to make sure you have a good control on Backup and Restore procedures?

Aspect	Recommendation	Exception
Instance	● Must use AMIs	
Pipeline	● Use CFT for setting up deployment pipeline for each product line – Dev, QA and PROD	
Monitoring	● Must use the organization monitoring tools + AWS services <List out standard tools used by the organization>	
AWS Services	● OpsWorks, Config, CloudWatch, CloudTrail	
Backup	● AMI – backups ● Databases – snapshots (daily) + read-replicas	

Restore	• Must exercise an end-to-end restoring and come up with a procedure document. Have a monthly practice to ensure it, unless a full DR process is in place	
CDN	• AWS CloudFront	
Consistency	• Must make sure the deployment models are consistent between Dev/QA/Prod environments. Either do the setup via scripting or use CFT	

Enterprise Logging

Questions:

1. Logging is a very important aspect of any enterprise system. Do you want to make sure you have good & established logging guidelines?
    a. Only use the organization's standard logging framework.
    b. Make sure the logging is setup automatically (reliability), as opposed to depending on humans to turn on/off logging.
2. Want to make sure the Logging infrastructure is enabled in such a way that it helps in the Monitoring and Debugging of any issues faced?

Aspect	Recommendation	Exception
Log Infrastructure	• ELK stack	
EC2 logging	• Each EC2 instance/process must setup logstash as part of the launch	

	process, so manual intervention is not needed	
Elastic Search	• Must use best practices in managing clusters and indexes. ▪ ES Domains ▪ Clustering ▪ Indexes ▪ Encryption – Data at Rest ▪ Index snapshots	
ES – AWS and the organization's DC	• Standard logging framework • Backup and reconciliation of Logging data between AWS and the organization's DC	• Make sure there are no version compatibility issues

Monitoring / Change Management

Questions:

1. Want to make sure there's uniform monitoring of the infrastructure, including different products/services? Must use the organization's standard monitoring tools
2. Individual product/services must expose/implement interfaces, so that they can be plugged into the monitoring framework.

Aspect	Recommendation	Exception
Uniform Monitoring	• Use the organization's standard tools: APM, NPM	None
Incident Management	• Must follow the organization's Incident Management process	None

Change Management	• Must follow the organization's Change Management policies.	None
Audit	• Must follow the organization's Audit policies.	None

Architecture

Aspect	Recommendation	Exception
Monolithic vs Distributed	• Product should NOT be built as monolithic. It should have different modules that can be deployed independently. This would provide the ability to scale if any module(s) becomes a bottleneck.	In case of product being sunset
Interface	• Must use RESTful Web Services	
RDBMS	• Must use RDS, except not supported by AWS	
NoSQL	• Only the following NoSQL are supported: For example: MongoDB, DynamoDB	
Scaling	• Product must be able to support scaling any module horizontally. Must not depend only on vertical scaling. • Must support AWS auto-scaling features	
Runtime	• Must be able to run in a Containerized environment.	

	• Support the organization’s Standard runtime environment – EKS, for example	
Performance	• Must use best practices in making sure the product performance is good • Use Asynchronous (message driven) architecture where applicable • Use Caching	
Deployment	• Must use CI/CD • Use Jenkins for CI/CD	

About the author

Vasu Gourabathina
Vice President, Engineering,
ATMECS, INC.

Vasu Gourabathina is a passionate software professional with 24+ years of experience in large-scale distributed software development. He has worked for startups (DISCERN, Augmentum, Vitria Technologies), mid-sized companies (Vitria Technologies, Formtek) and large corporations (Intergraph, Oracle and SAP Labs). Some of the areas include SaaS, Cloud, Big Data, EAI, ERP, BI, and Business Processing Modeling in Manufacturing, Healthcare, Finance, and High-Tech industries.

Vasu is very enthusiastic about the “data” space – data engineering, distributed data processing, big data, cloud computing, machine learning, artificial intelligence to name a few. He’s also versatile in dealing with Relational, No-SQL, and Graph databases.

Focusing on defining product strategies, delivering quality products through continuous iterations and improvements, handling operational challenges in a highly dynamic environment are some of the author’s strengths. And most importantly, he’s a hands-on leader who enjoys "rolling up the sleeves" when needed.

About the company

ATMECS is a result oriented full service engineering and R&D organization. We are **Technology Accelerators** bringing in visible transformation for our clients through automation, adoption of leading edge integrated development platforms, CI/CD, Dev Ops, Cloud, and Big Data . Several Fortune 500 customers and exciting next gen startup companies engage us to partner with them to solve critical business challenges.As **Innovation Catalysts** we help clients lead change through AI/ML, AR/VR, IOT, Conversational BOTs & Blockchain.

Thank You!
ATMECS
Passionate Minds