**Technical Challenge**

We understand your time is precious, but as discussed, we ask each applicant to take on our small challenges. There is no right or wrong approach and we're certainly not expecting war and peace 😊. We would expect no more than 30 minutes per challenge, if at the end of the time you haven’t completed don’t worry just send us what you have. (Uploaded to a public GIT repository – ensuring there is nothing KPMG sensitive in the test below there is no need for anything to be)

**Challenge #1**

**A 3-tier environment is a common setup. Use a tool of your choosing/familiarity creates these resources. Please remember we will not be judged on the outcome but more focusing on the approach, style and reproducibility.**

**Answer** ---- >

We can design a 3-Tier Environment in AWS using end to end DevOps automation as the backbone for slashing the manual efforts by half, invoking Agile Practices, faster value based deliverables in shorter span of time.

Below Points/Pre-Requisites can be taken in consideration while designing a 3-Tier Environment setup approach

* For End to End Automation we can apply DevOps, DevSecOps, Monitoring, Logging, and Governance Practices while setting up the Environment.
* For Infrastructure Automation we can apply IAC Services/Tools – Infra As Code practices by using AWS Cloudformation or Terraform
* For Infra-As-Code Yaml/Json templates scanning we can use some Tools like cfn-nag for Cloudformation, & tfsec,checkov etc for Terraform.
* For Code Release CI/CD Automation we can enable Automated Security testing like Unit testing, SCA – Secure Code Analysis like SonarQube for checking code bugs / vulnerabilities, duplicacy & coverage , SAST/DAST tools like Checkmarx for AppSec – Application Security Testing. We can set threshold / coverage parameters inside SonarQube dashboard. Incase a code doesn’t meet the standard’s while getting automatically scanned through CICD pipeline it will rollback automatically at any of the security testing stages.
* We can enable automated Monitoring/Governance as part CICD SDLC Lifecycle. We can use services like Cloudwatch, SNS, Approval Gates.
* We can create IAC & CICD Configuration automatically using YAML or JSON based script’s along with Paramaters JSON files which could be specific to Eviromennts like DEV, QA , UAT , PRE-PROD or PROD.
* We can deploy Infrastructure automatically using either Powershell tool with aws cli installed in it & running aws cloudformation cli commands or we can create a Infra based CICD Pipeline which can deploy our IAC YAML/JSON templates along with Parameter JSON files.
* We can create Master / Nested form of IAC CICD approach for deploying all the Templates in a single click deployment.
* We can use Source Version Controlled Repo/Tools like – AWS Codecommit, Github, Gitlab, Bitbucket etc for storing code files.. as part of Continuous Integration. The code files structure can have developer code & DevOps scripts – IAC & Code deployment CICD Pipeline scripts.
* We can apply Gitflow Branching Model with the Source Control Version for applying permissions/ownership for developers while they do parallel developments depending on the number of Environment’s. For Ex --- > DEV, QA & PROD could be permanents branches & then have IAC CICD & Release Deployment CICD pipelines. Feature , Bugfix, Hotfix could be temporary branches
* We need to make sure we all the required IAM roles / Policies created automated which can then be used with IAC & CICD Deployments.
* We need to make sure to remove the hardcoded values from YAML/JSON templates & use Paramater JSON files along with Templates to make them re-producable across multi-staged enviroments.
* We need to make sure if we have any credentials/secrerts to be used with YAML/JSON templets.. we need to store the secrets in secret store something like AWS Secrert Manager & use the ssm parameter path in YAML/JSON scripts.
* We can also make sure to only used \*wildcards if necessary & only in lower Environments – DEV , it’s a better practice to have only the specific Resource ARN values defined.
* We can also use Export /Import functions, Pseudo Functions inside the YAML/Json templates to make them more mature & refined. Also define Outputs sections so we can get the Output values for all the ARN of all the resources created which can be used other Inter Dependent templates stacks….

Some of Following Tools can be used for DevOps, DevSecOps , Monitoring & Governance while setting environments. It could be a combination of open source & Cloud Provider services or Just the Cloud Provider Services.

**IAC** -🡪 AWS Cloudformation, Terraform, Azure ARM, CDK – Cloud Development Kit

**CICD** -🡪 AWS CodePipeline, Azure DevOps CICD, Gitlab CICD, Jenkins CICD, AWS Codebuild, AWS Codedeploy, Azure Build, AWS Codestar

**SecOps** -🡪 Junit, Pytest, SonarQube – SCA, CheckMarx – SAST/DAST, Selenium – Load Testing

**Monitoring**/ **Governance** -- > Cloudwatch, Cloudtrail, AWS X-ray, Splunk, Prometheus, Grafana, ELK, AWS Event Bridge, Simple Notification Service, Codebuild – Approval Gates

**Source Control Version** -- > Github, Gitlab, AWS CodeCommit , Azure Repo, Bitbucket

**Containerization** & **Clustering** 🡪Docker/EKS/AKS/Kubernetes, AWS ECS/Fargate

**Database** -🡪Aurora MySQL & PostgreSQL RDS, Neptune GraphDB, DynamoDB

**Build/Code Framework** -🡪 Apache Maven, NPM, Python, Java, NodeJS/AngularJS

**Programming/Scripting** -🡪 Bash, PowerShell, AWS CLI, Azure CLI, EKSCLT, KUBECTL, AWS Cloudshell, Developer/IDE -🡪 Visual Studio Code, Putty, GitBash, MobaXterm

**RestAPI** -🡪 Swagger , Postman

***Sample 3-Tier Solution***

Below could be a sample 3-tier solution which we can design using DevOps automation techniques & keeping above pre-perquisites in mind.

**Designing a Serverless Framework 3-Tier Microservice Based Solution using AWS DevOps/DevSecOps practices.**

Presentation tier (front-end) – API Gateway

Application tier (back-end) – business logic layer – AWS Lambda

Data tier (database) – data storage layer – AWS Aurora, Dynamo DB, RDS etc.

**Scenario** -🡪 User requests comes to the API Gateway front-end. API forwards the requests to Lambda backend. Lambda is a function-as-a-service that allows you to deploy and run your code when invoked without managing servers. Lambda function then queries the database for all or specific data which is then sent back to the API Gateway and to the user.

We can deploy the Infrastructure components --- > VPC, Private/Public Subnets, NAT, IGW, Bastion Host for connecting to the Database, SonarQube, ChecMarx, API Gateway, AWS Lambda, Specific Database, Cloudwatch, AWS X-Ray, SSM Parameter Store, SNS, Required IAM roles/policies, S3 Buckets.

We can securely build, test & deploy the Serverless Code artifacts in an automated way which can be used with AWS Lambda functions.

We can have Infra CICD & Code Build, Deploy CICD Pipelines integrated with a Source Control which can first deploy the Infra-as-code scripts , then perform the Developer code testing, scanning, Build & Deployment.

We can use some tools / services like AWS SAM – Serverless Application Module which can help in deploying Serverless Components like AWS Lambda, Rest API Gateway, Dynamo DB etc…. We can deploy multiple lambda functions from one Single SAM File & then for future can also deploy enhanced feature for specific Lambda functions using the same SAM template.

We can use SWAGGER open source solution along with AWS SAM or Standard YAML/JSON to automate the API Gateway Documentation like Method Execution – PUT, GET, POST etc … Method Integrations like – AWS Lambda Functions ARN for backend invocation etc….

We can store the secrets for Database & Bastion Host in SSM & then use the SSM path inside the SAM, Bastion & Database YAML/JSON & Parameter JSON files templates for retrieving the values in an encrypted manner.. This helps in using the same scripts in higher environments as re-usable assets.

We can apply Branching strategies for Continuous / Parallel Deployments for Infra-as-code & SDLC & merge the code for deploying to higher environments automatically.

Using the Bastion Host UserData we can login to Database & update sample table having sample data which AWS Lambda can use in the backend & produce as to the Frontend calling API.

For any changes in development code or Infra-as-code changes we can apply triggers in CICD pipeline which will deploy the new changes & won’t affect the pre-existing Infrastructure.

Also we can configure Blue-Green/Canary deployment technique for Serverless Code for maintaining multiple versions & allowing shifting of certain % of user traffic to new version & rollback strategies to prevent any application downtime issues.

**Sample Other Scenarios** --- >> We can also deploy Microservices Containerized Applications using Services Like AWS ECS with EC2 , AWS Fargate – Serverless Containerization , Kubernetes platforms along with Load Balancing, Auto Scaling functionalities & using above pre-requisite though process for scalable re-producible deployments

**Challenge #2**

We need to write code that will query the metadata of an instance within AWS and provide a json formatted output. The choice of language and implementation is up to you.

**Answer** ---- >

We can retrieve/query the metadata of a running instance in AWS using one of the below methods.

***Option-1***

1. SSH/Login to the EC2 machine using Putty
2. Yum install curl package if not already installed
3. Install JQ utility, JQ is your JSON Query Processor
4. We can install JQ on Linux/Centos machine using below commands

* install EPEL Repository (yum install epel-release -y)
* yum install epel-release -y
* yum update -y
* yum install jq -y
* jq -Version
* curl http://169.254.169.254/latest/meta-data/ | jq
* curl <http://169.254.169.254/latest/meta-data/public-keys/> | jq

We can enable all these steps as part of bash script as well. make the sh file executable using chmod

***Option-2***

***Install python on EC2 instance***

yum install python3

***Install python virtual environment & dependencies***

pip3 install pipenv

pipenv install Pipfile

***Execute python code files for fetching the metadata & key (stored in git)***

python3 metadata.py

python3 key.py



**Bonus Points**

The code allows for a particular data key to be retrieved individually

Hints

·         Aws Documentation (<https://docs.aws.amazon.com/>)

·         Azure Documentation (<https://docs.microsoft.com/en-us/azure/?product=featured>)

·         Google Documentation (<https://cloud.google.com/docs>)

**Challenge #3**

We have a nested object, we would like a function that you pass in the object and a key and get back the value. How this is implemented is up to you.

Example Inputs

object = {“a”:{“b”:{“c”:”d”}}}

key = a/b/c

object = {“x”:{“y”:{“z”:”a”}}}

key = x/y/z

value = a

Hints:

*We would like to see some tests. A quick read to help you along the way*

*We would expect it in any other language apart from elixir.*

[*A quick read to help you along the way*](https://hexdocs.pm/elixir/master/Kernel.html#get_in/2)

***Answer*** *-- >*

you\_dictionary = {

new\_key: new\_val

for keys, new\_val in [(['y', 'q', 'x'], 40)

for new\_key in keys

}

print("Multiple keys same value:",you\_dictionary)

Once this has been completed please send us the output so we can get the ball rolling.