Mahmoud Zaky Fetoh

Phone: +201022381474
Email: zaky.fetoh@gmail.com
github.com/zaky-fetoh
linkedin.com/in/mahmoud-zaky-fetoh

Objective

I am a DevOps Engineer passionate about solving a real-world problem, working and collaborate with a great Engineers that I can learn from. I have mixed experience across various discipline such as full stack development, data Science, computer vision and mainly devops Engineer.

Education

2020 - 2023	**Faculty of Computers and Information, Menoufia university**, Egypt. M.Sc., doing computer vision research, Expected graduation 2023
2015 - 2019	**Faculty of Computers and Information, Menoufia university**, Egypt. B.Sc. Honors, I graduated with the Top, 1st, Grade, Excellent with honor GPA 3.5.

Work Experience

Dec 2023 - Present	**Cloud Consultant** @ **Bexprt**, KSA. Involved at architecting and implementing MLOps and DevOps projects for MENA customers. -*Technologies*: **Clould Provider**: AWS. **IaC**: Terraform. **CI/CD**: Github Actions, Atlantis, Terraform Cloud. **Container Orchestrators**: ECS. **ML Model Deployment**: SageMaker.
May 2023 - Dec 2023	**DevOps Engineer** @ **n-go**, KSA. Administrating kubernetes clusters for 1M user Application. -*Technologies*: **Clould Provider**: AWS, OCI, Cloudflare. **Container Orchestrators**: EKS, OKE. **VPN & ZTN**: Pritunl, Twingate. **CI/CD**: Jenkins, GitHub actions, ArgoCD. **kubernetes Autoscaler**: Cluster Autoscaler, karpenter. **Monitoring**: Prometheus, Grafana, Loki, ELK, Kiali, Jaeger. **IaC**: Terraform, Kustomize, Helm, Ansible. **Service Mesh**: Istio.
Jan 2023 - Apr 2023	**MLops Engineer** @ **Susoft**, Norway. Building microservice application for Training and deploying machine and deep learning models for Sales forecasting. Performing customer segmentation to direct marketing campaigns. -*Technologies*: **Model Training**: PyTorchForecasting, Pytorch, pandas, Prophet, NeuralProphet. **Model Monitoring**: Weight and biases, Prometheus, Grafana. **Model Serving**: torchScript, Docker, K8s. **Model Registry**: Minio. Asynchronous communication for issuing train request is done using RabbitMQ Gateway and load balancing services performed using NodeJS.

Frameworks & Technical Skills

Programming Languages:	Python, NodeJS, GoLang (learning)
HTTP Servers Frameworks:	ExpressJS, Flask.
Front-End Frameworks:	ReactJS, Angular (learning).
Version Control Tools:	Git, Github, Bitbucket.
Data Science:	PyTorch, Pandas, statsmodels, Prophet, Numpy, OpenCV, Plotly, SageMaker, LLAMA 2, PaddleOCR.

Client API:	REST, GraphQL, gRPC, SocketIO.
Security Utilities:	Joi, Jsonwebtoken (JWT), OAuth 2.0.
Databases:	MONGODB, MySQL, PostgreSQL, MariaDB.
Adminstration:	CCNA, MCSA, RHCSA, Kubernetes,
Clould Provider:	AWS, OCI, Cloudflare.
Testing & Monitoring:	Jest, Prometheus, Grafana, Loki, ELK, kiali, jaeger.
Container Orchestrators:	EKS, OKE, ECS, Docker-Compose.
Infrastucture as Code :	Terraform, Kustomize, Helm, Ansible.
CI/CD:	Atlantis, Terraform Cloud, Jenkins, Github Actions, ArgoCD .
VPN & ZTN:	Pritunl, Twingate.
Message Broker:	RabbitMQ, Kafka, SQS.
Documentation:	Swagger, LaTeX.
Caching:	Redis, MinIO.

Publications

2022 **Multiscale aware classification of COVID-19 from Chest X-Ray using a spatially weighted atrous spatial pyramid pooling CNN**[Link].
Mahmoud Z fetoh, Khalid M. Amin, Ahmed M. Hamad
In this paper I propose, scale invariant CNN architecture for COVID-19 classification. Proposed model based on building a scale space in each layer using Atrous spatial pyramid pooling then selecting a correct space to operate at using spatial attention module.

2021 **COVID-19 Detection Based on Chest X-Ray Image Classification using Tailored CNN Model**, [Link]. **Mahmoud Z fetoh**, Khalid M. Amin, Ahmed M. Hamad
In this paper I propose a very light-weight model as a consequence of using spatial separable kernel and depth-wise separable kernels for COVID-19 classification.
Published at: IJCI.

Project Experience

2023 **Internal Developer Platform**, [n-go].
Creating IDP that provide automate frequent devTeam tasks i.e) changing the Env variables of the running kubernetes Services. It secured with JWT as application level security. And behind Zero-Trust Network as infrastucture level security.
- Exposing NodeJS REST API as Backend.
- Simple UI using ReactJS for frequent tasks.
- IDP backend is Secured with JWT.

tools used: NodeJS, Python, bashScript, ReactJS.

2023 **Security Enhancement**, [n-go].
I helped my team deploying Zero-Trust network to allow DevTeam to access the Infrastucture securely.

2023 **Secret-free manifests**, [n-go].
For security concerns we removed all AWS secrets from manifests and replaced It with IRSA allowing k8s pods to assume an IAM role to access AWS directly. Also all non-aws secrets are moved to secret manager.
- IRSA for k8s pods.
- Crossplane for automatic secret creation at aws secret manager.
- Full integration with CD pipeline using crossplane.

tools used: Kustomize, awscli, External-Secrets-Operator, IRSA, bashScript, Reloader, crossplane.

2023 **Migrating of CI/CD**, [n-go].

Complete Pipeline Automation for service creating and update.

- Re-organizing all k8s manifests using Kustomize.
- Automating service creation and DNS record insertion.
- Configuring the App-of-apps as Helm chart to reduce duplication.
- Making all services follows ArgoCD app-of-apps pattern.
- Migration of all CI/CD pipeline from bitbuckets to github.
- Full integration with Jenkins pipeline.

tools used: Kustomize, Jenkins, argoCD, Python, bashScript, Reloader, Helm.

2023 **Remote Job Initiation**, [n-go].
Creating REST API to allow the dev team to remotely intiate K8s any job/cronjob (providing Image:tag for the container) on the cluster. This API responds with job status and the logs. Also configuring appropriate RBAC priveldge to the service.

- *tools used:* Kubectl, bashScript, NodeJS.

2023 **Monitoring & Alert manager**, [n-go].
configuring Prometheus-kube-stuck, loki, uptime-kuma, kiali, and jaeger to monitor the k8s cluster. Integrating Istio with Prometheus to collect service mesh metrics. Integrating With slack to notify a cretical ops isuees.

- *tools used:* Prometheus, Grafana, Loki, uptime-kuma, kiali, jaeger.

2023 **Complete Infrastucture Migrating from root AWS account to new AWS account**, [n-go].
Complate migration of dev (dev, testing, staging) to new aws account to allow devTeam to access separetly using SSO of G-suit.

- Migration of all AWS services (EKS, RDS, S3, Redshift, DMS, SQS) account to new AWS account.
- Configuring EKS and related nodeGroup, CSI driver, EC2 template.
- configuring Istio service mesh and integrating it with Prometheus.
- Configuring cert manager to stablish TLS connections.
- Reorganizing AWS VPC and producing terraform code for it.
- allow above is created with terraform.

tools used: Terraform, EKS, Cert-manager, Istio.

2023 **Personal Blog Site**, [Link].
This is a MERN stack project that manage personal blog and store it in MONGODB database. It performs caching for quary using redis. It's frontend is build using ReactJS and it state is managed using Redux-toolkit.

- *Built with:* NodeJS, ExpressJS, ReactJS, Crypto, Redis, MongoDB, Docker-Compose, Redux-toolkit, formik, yup.

2023 **Sales Forecasting** [Susoft]
I build a training pipeline and serving pipeline for TFT, prophet, and NeuralProphet models. These models are used to serve about 2k market chain. It is a microservice application with the following anatomy:

- **Model Trainer microservice**: It accepts trining requests throught a rabbitMQ queue (each model type has dedicated queue). Upon receiving the request it qaurry MariaDB for required data then perform data cleaning, Feature enrichment with whether data and model training. Upon model training completion it saves the trained model to model Registry(MinIO) and notifies the gateway model training success. It also logs the traning metrics to WandB for monitoring. TFT is saves to MinIO after converting it to torchScript format for performance.
  - *Built with:* python, pika, prophet, NueralProphet, PyTorchForecasting, pandas, mariadb.
- **Model Server microservice**: It registers itself to loadbalancer microservice to allow large scale serving. Upon receiving synchronous predict request it pulls appropriate model from Model Registry(MinIO) microservices and performs may also quarry required data from mariadb and performs data enrichment required. Then performs prediction and respond with results.
  - *Built with:* python, flask, pandas, prophet, NueralProphet, PyTorchForecasting.
- **LoadBalancer microservices**: It provide load balancing service for model Server service or any service that require load balancing such that each services periodically register itself every 1min.
  - *Built with:* NodeJS, Express
- **gateway**: It consolidates the entire application to single interface to interact with. When receiving train request it push request to appropriate RabbitMQ queue based on requested model type and when training completed it notifies the backend that it request model can serve. when receive a predict request to quarry the loadbalancer to available instance and forward the request to it.
  - *Built with:* NodeJS, Express, amqp.

2022 **Clinic Management System**, [Link].
A backend System for managing clinics, it's department, medical stuff, and patients. This System provides a RESTfull, and GraphQL for clients. Complex mongodb aggregation pipeline is implemented for extracting high level info.

- *Built with:* NodeJS, Express, Mongoose, graphql, Joi.

2022 **Image Manager**, [Link].
Microservices application for image storage, stored in plain form or encrypted form, with it's meta data and automatic tagging using Deep learning techniques.
It consists of three microservices as follow:

- **Image-Classification micro-service:** This micro-service is responsible for classifying images using ResNet18 other services can communicate with it using synchronous communication using by requesting it RESTfull API or through asynchronous communication using RabbitMQ.
  - *Built with:* Python, Pytorch, Flask, pika.
- **Image-Storage micro-service:** This micro-service is responsible for storing the images the users in either plain or encrypted, using AES, form other microservices can communicate with it using it's RESTfull API.
  - *Built with:* NodeJS, Express, Multer, crypto.
- **Image-manager micro-service:** This micro-service is responsible for 1) storing the meta-data, such as Image's owner and the image Tags, automatically added by Image Classification microservice, and the total number of views, 2) manage users, users credentials and authentication 3) communicate with Image-Classification microservice using RabbitMQ and With Image-Storage through it's RESTfull API.
  - *Built with:* NodeJS, Express, Mongoose, JWT, bcrypt, amqplib, Joi.
- **Route-Advertisement micro-service:** To avoid manually adjust API Gateway for adding public route route-advertiment (RA) manages all routes for the entire micro-service application. Any service wants to advertise a specific public route it can HTTP POST this route to RA through it's RESTfull API. RA assumes order of routes does not matter. RA allows dynamically adding routes to the API Gateway.
  - *Built with:* NodeJS, Express, Mongoose, axios, JWT.
- **API Gateway micro-service:** API Gateway (GW) micro-service periodically request new added routes to route-advertisement. Every GW is responsible of keep tracking it's state and request missed routes.
  - *Built with:* NodeJS, Express, Mongoose, axios, cron.