

I tried to automate all pieces, but couldn't finish up the workload pipelines, but created all the code looking at Azure's best practices and well architected frameworks. Given the intriguing project options available, I ventured to undertake the challenge of addressing each one. It is not a straightforward endeavor to devise a robust cloud architecture. The diagram above encapsulates a viable approach to production readiness (following thorough testing, of course). This approach involves organizing all resources within a dedicated resource group in Azure, encompassing cloud resources such as AKS clusters, Virtual Networks (VNETs), Subnets for AKS and Flexible MySQL Server instances, and ArgoCD GitOps workflows.

While I endeavored to automate all components, I was unable to complete the workload pipelines. However, I created all the code while adhering to Azure's best practices and well-architected frameworks.

Project A:

- Service to return POD IP and Version, from inside a container with Python Flask
- Created helm chart as well as pushing it to docker hub OCI registry along with container.

<https://hub.docker.com/repository/docker/sqapy/container-ip/general>

*** TODO: CI/CD workflows with github actions

How to use:

Installation can be done from a repository or directly pulling from DockerHub OCI registry

- Set your kube-context to your desired test cluster
- Go to following location:
<https://github.com/sqe/azure-terraform-aks/tree/main/container-ip/helm-chart/>
- Execute: `helm install container-ip "container-ip"`

Observe similar outcome:

```
sqapy@sqapy-T490s:~/aera/container-ip/k8s$ helm install container-ip "container-ip"
```

NAME: container-ip

LAST DEPLOYED: Fri Sep 13 07:39:13 2024

NAMESPACE: default

STATUS: deployed

REVISION: 1

NOTES:

1. Get the application URL by running these commands:

```
http://chart-example.local/
```

```
sqapy@sqapy-T490s:~/aera/container-ip/k8s$ kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
argocd	argocd-application-controller-57c654bb67-cxxrg	1/1	Running	0	4h49m
argocd	argocd-dex-server-7b557cf649-dkmzn	1/1	Running	0	4h49m
argocd	argocd-server-f8965847d-gzr57	1/1	Running	0	4h49m
default	container-ip-5dfff468d5-6kd96	0/1	ContainerCreating	0	23s
kube-system	csi-azuredisk-node-tcvn9	3/3	Running	0	4h57m
kube-system	csi-azurefile-node-d4s6r	3/3	Running	0	4h57m
kube-system	connectivity-agent-869d8bd998-sc5j9	1/1	Running	0	4h22m

```
sqapy@sqapy-T490s:~/aera/container-ip/k8s$ kubectl get pods -A
NAME                                READY    STATUS    RESTARTS   AGE
argocd-application-controller-57c654bb67-cxxrg  1/1      Running   0           4h49m
argocd-dex-server-7b557cf649-dkmzn             1/1      Running   0           4h49m
argocd-server-f8965847d-gzr57                  1/1      Running   0           4h49m
default/container-ip-5dfff468d5-6kd96           0/1      ContainerCreating 0           23s
kube-system/csi-azuredisk-node-tcvn9            3/3      Running   0           4h57m
kube-system/csi-azurefile-node-d4s6r           3/3      Running   0           4h57m
kube-system/connectivity-agent-869d8bd998-sc5j9  1/1      Running   0           4h22m

sqapy@sqapy-T490s:~/aera/container-ip/k8s$ kubectl get svc
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
container-ip                        NodePort    10.0.81.213    <none>          80:30463/TCP     3m7s
kubernetes                          ClusterIP   10.0.64.1     <none>          443/TCP          5h1m

sqapy@sqapy-T490s:~/aera/container-ip/k8s$ kubectl port-forward svc/container-ip 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
Handling connection for 8080
```

Project B:

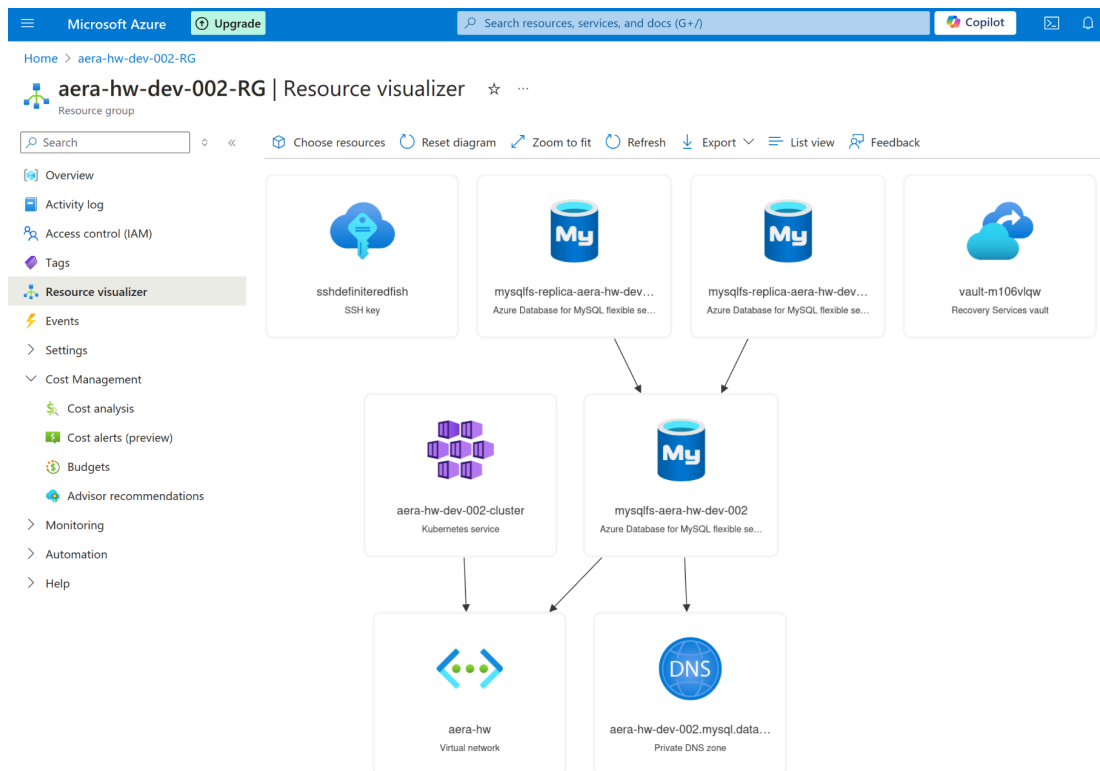
MySQL Flexible Server with replicas on the same location as primary, and reader replicas in more regions, configured via list named locations in terraform.tfvars

- Infrastructure provisioning:
 - *Export your Azure Service Principal credentials on shell*

- **`terraform apply -var-file=terraform.tfvars`**
- <https://github.com/sqe/azure-terraform-aks/tree/main/infra/environments/aera-hw-dev-001>
- Environment composition consist of VNet, Subnets, Storage, Permissions, AKS, ArgoCD, Helm bootstrap, Flexible Mysql Server
- Once infrastructure is provisioned the resource group map looks like as following:

***TODO: fix github actions for infra provisioning, the issue is setting up a Service Principal authentication which needs to be resolved

<https://github.com/sqe/azure-terraform-aks/actions/runs/10851039164/workflow>



Project C:

Containerized SFTP server using Azure BlobStore

- Created POD, PersistentVolumeClaim, Persistent Volume with blob.csi.azure.com driver
- <https://github.com/sqe/azure-terraform-aks/tree/main/sftp>

***TODO: Deploy SFTPGo container in AKS <https://sftpgo.com/>