SUMMIT

MILANO | 22 GIUGNO 2023

KUB201

Migrare un'applicazione su Kubernetes (e AWS) uno YAML alla volta: l'esperienza CRIF

Fabio Chiodini (he/him)
Principal Solutions Architect
AWS

Alex Landini (he/him)
Software Architect Tech Lead
CRIF

Gabriele Armao (he/him)
Senior System Advisor
CRIF

aws SUMMIT MILANO

Scarica la App «AWS Events» e accedi a tutte le informazioni dell'evento!

GET IT ON Google Play

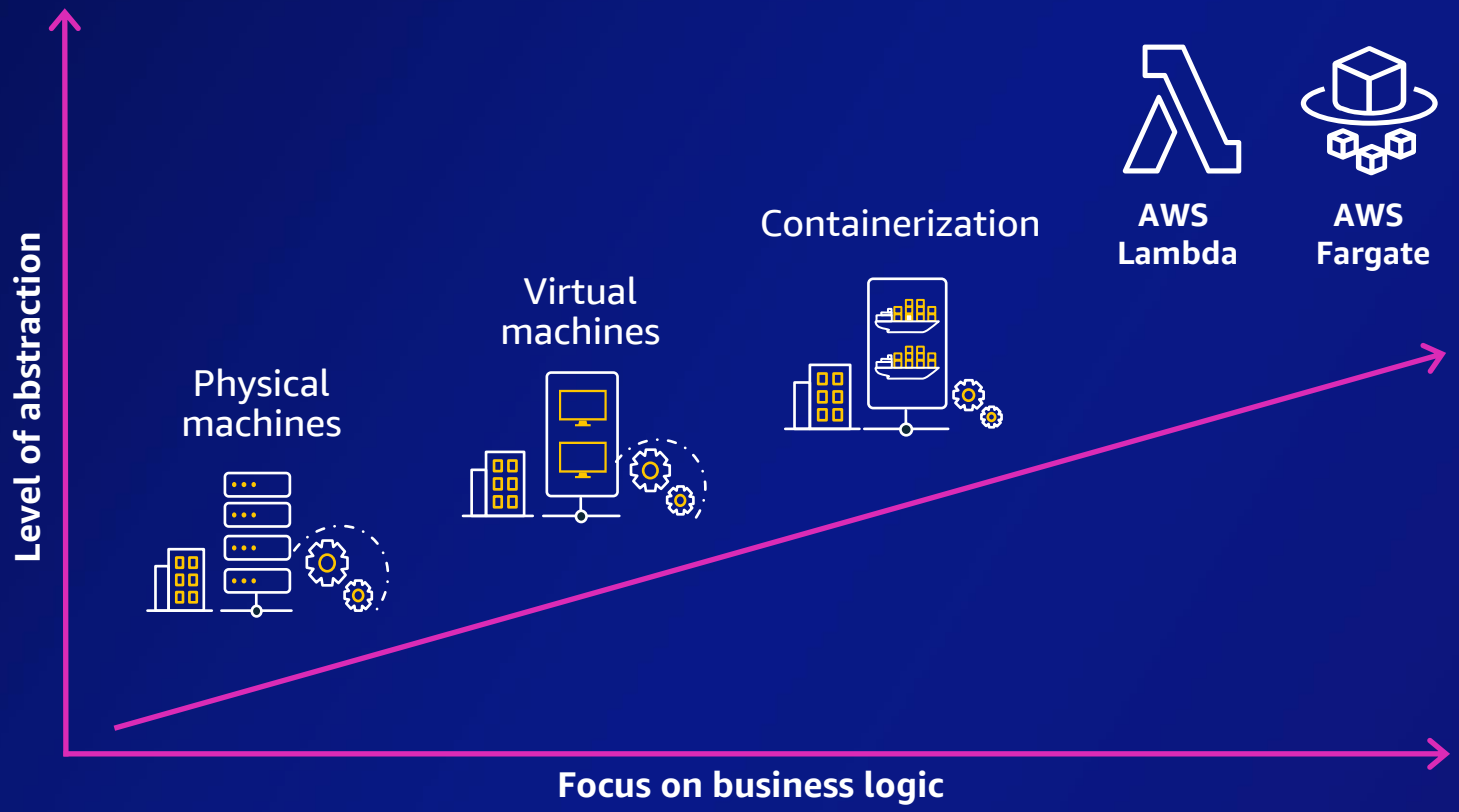
Download on the App Store

Agenda

- Containers and Kubernetes on AWS
- The CRIF experience with AWS
  - How it started (requirements)
  - First retro: Infra as Code and Traffic Management
  - Microservices Security and Async Messaging
  - High Availability and Persistent Data
  - CI/CD and Observability
  - A glimpse into our future
- Wrap Up

Containers and Kubernetes on AWS

Why our customers adopt containerization



Agility

Portability

Efficiency

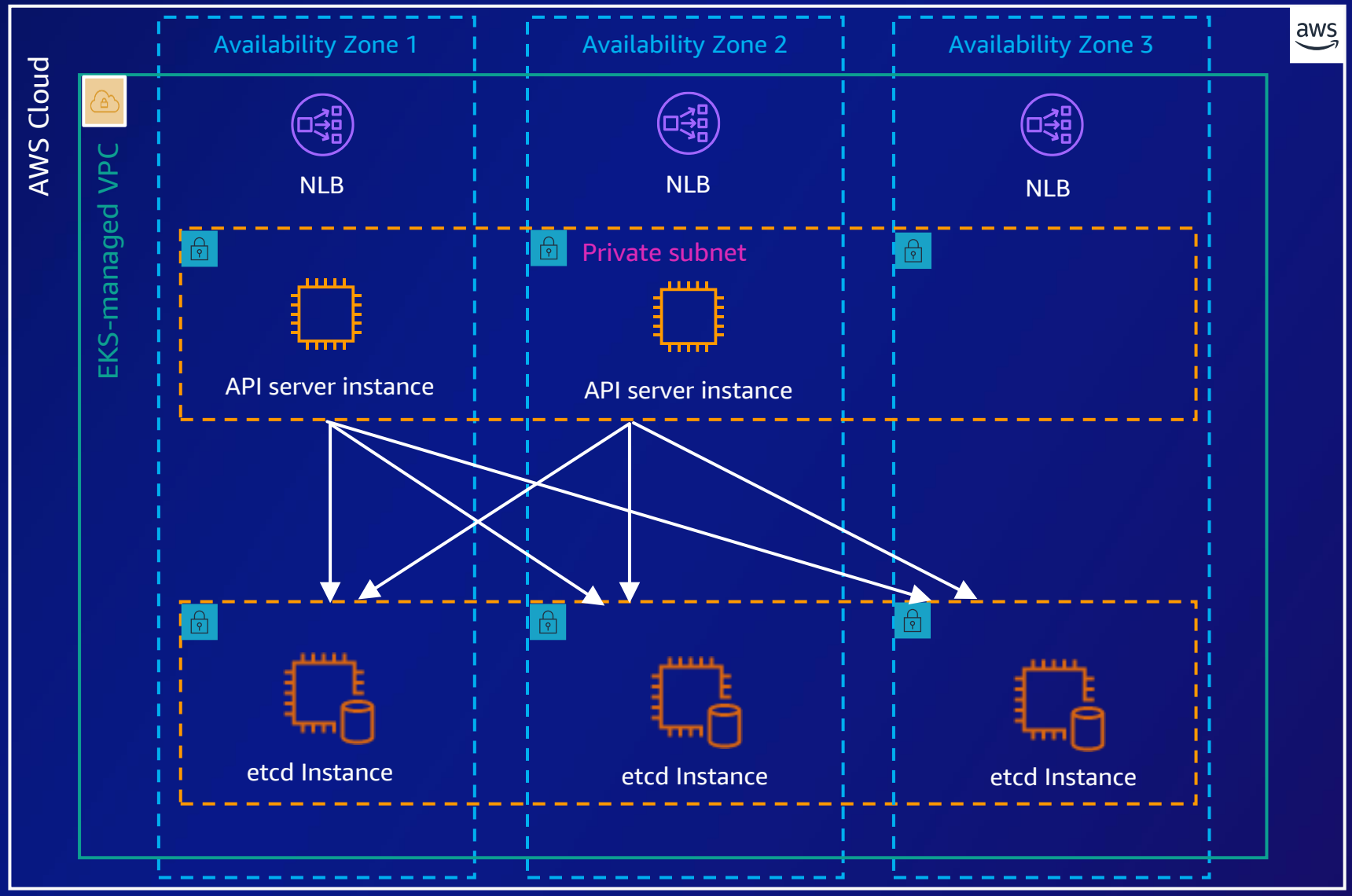
Amazon EKS is the most trusted and secure way to run Kubernetes

Amazon **EKS**

- Amazon EKS runs **vanilla Kubernetes**. Amazon EKS is upstream and **certified conformant version** of Kubernetes (with backported security fixes)
- Amazon EKS **supports four versions of Kubernetes**, giving customers time to test and roll out upgrades
- Amazon EKS provides a managed Kubernetes experience for **performant, reliable, and secure Kubernetes**
- Amazon EKS makes Kubernetes operations, administration, and management simple and boring

Amazon EKS enables you to build reliable, stable, and secure applications in any environment

Amazon EKS Control Plane Architecture



CRIF and AWS

OUR MISSION AND VISION

MISSION / The mission that drives CRIF is to create value and **new opportunities** for consumers and businesses by providing reliable information and solutions, allowing more powerful decisions and accelerating digital innovation.

VISION / Since 1988, we have been responsibly supporting our clients locally in their everyday financial journey, through **trusted information**, advanced **cutting-edge solutions, and unique expert knowledge.**

CRIF
Together to the next level

Where we started

System requirements

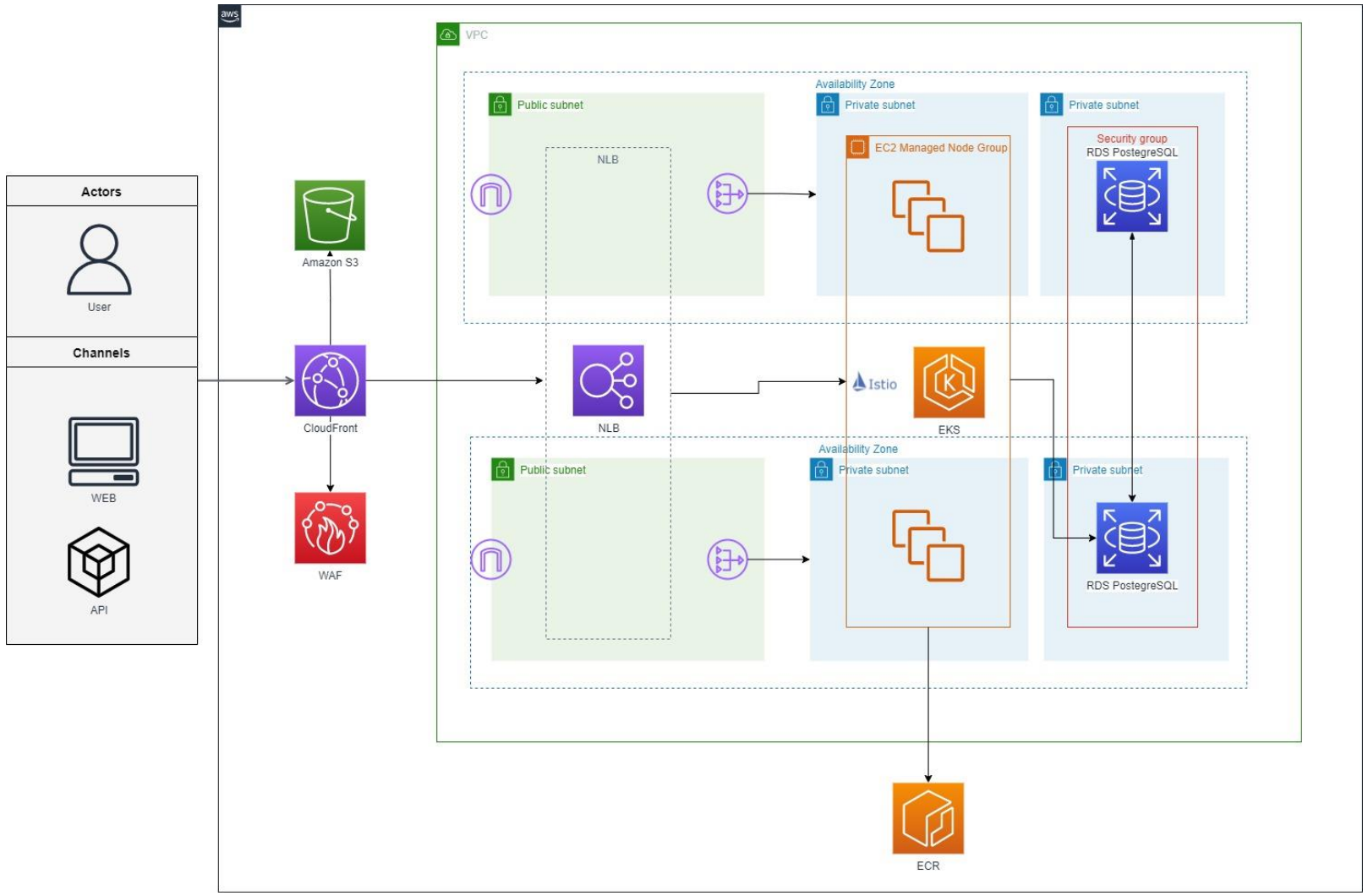
- **Portable –** deployable also on-premises
- **Cost Effective -** utilize AWS cloud services to architect a cost-effective cloud infrastructure
- **Private** - not freely accessible from the public Internet
- **Location constraint -** complying with data protection laws
- **High Available and Fault tolerant**
  - 24x7
  - 5K Transactions/day
  - About 3000 Single Users per day
- **Scalable and Responsive –** ensuring optimal performance and responsiveness during peak usage periods

Design Decisions

- **Portable –** EKS Kubernetes as a core of the system
- **Cost Effective**
  - RDS PostgreSQL
  - EKS EC2 Managed Node Group
- **Private –** AWS WAF, whitelist, packet filtering
- **Location constraint** – Infrastructure deployment in a compliant AWS Region
- **Fault Tolerant and Highly Available**
  - MultiAZ cloud architecture
  - Multiple Pod replicas
- **Scalable**
  - Kubernetes HPA
  - EC2 Autoscaling Group

How it started

First Cloud Architecture



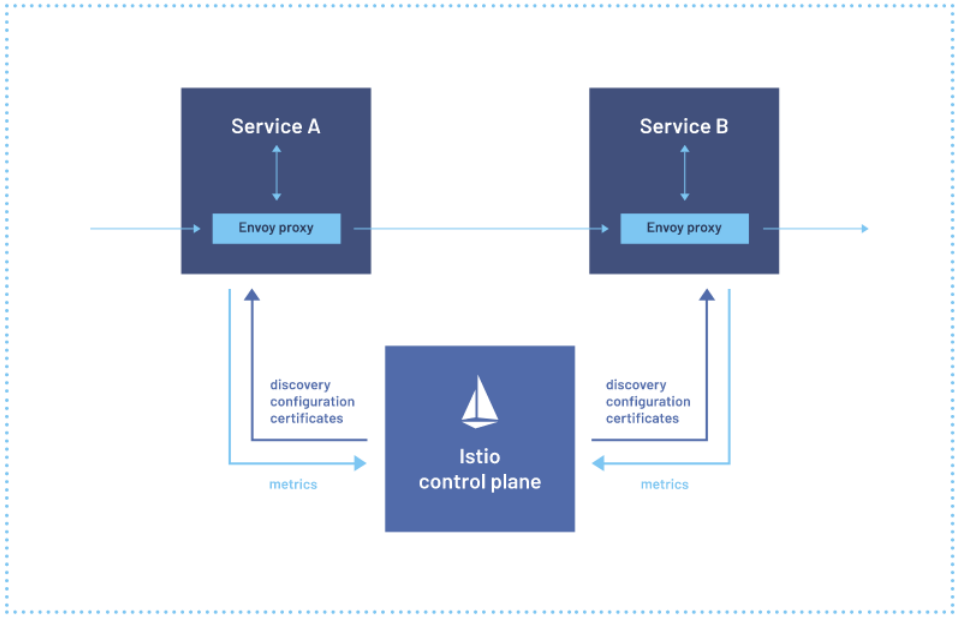
- **Web Tier**
  - CloudFront - static resource caching, decrease latency
  - S3 - Static web site
- **Application Tier**
  - Public Network Load Balancer
  - Elastic Kubernetes Service
  - Service Mesh - Istio
- **Data Tier**
  - Managed RDS for PostgreSQL
  - MultiAZ RDS setup

First Cloud Architecture – Service Mesh

- A service mesh is an infrastructural layer that:
  - Controls **how different microservices** composing an application **share data with one another**
  - Can be added to your applications without adding dependencies to application code, but it's deployed on top of the application services
  - Provides capabilities such as: observability, **traffic management**, and security

- **Istio** is one of the most popular service mesh
- Istio **traffic management features**:
  - Zone aware routing
  - Timeouts
  - Retries
  - Circuit Breakers

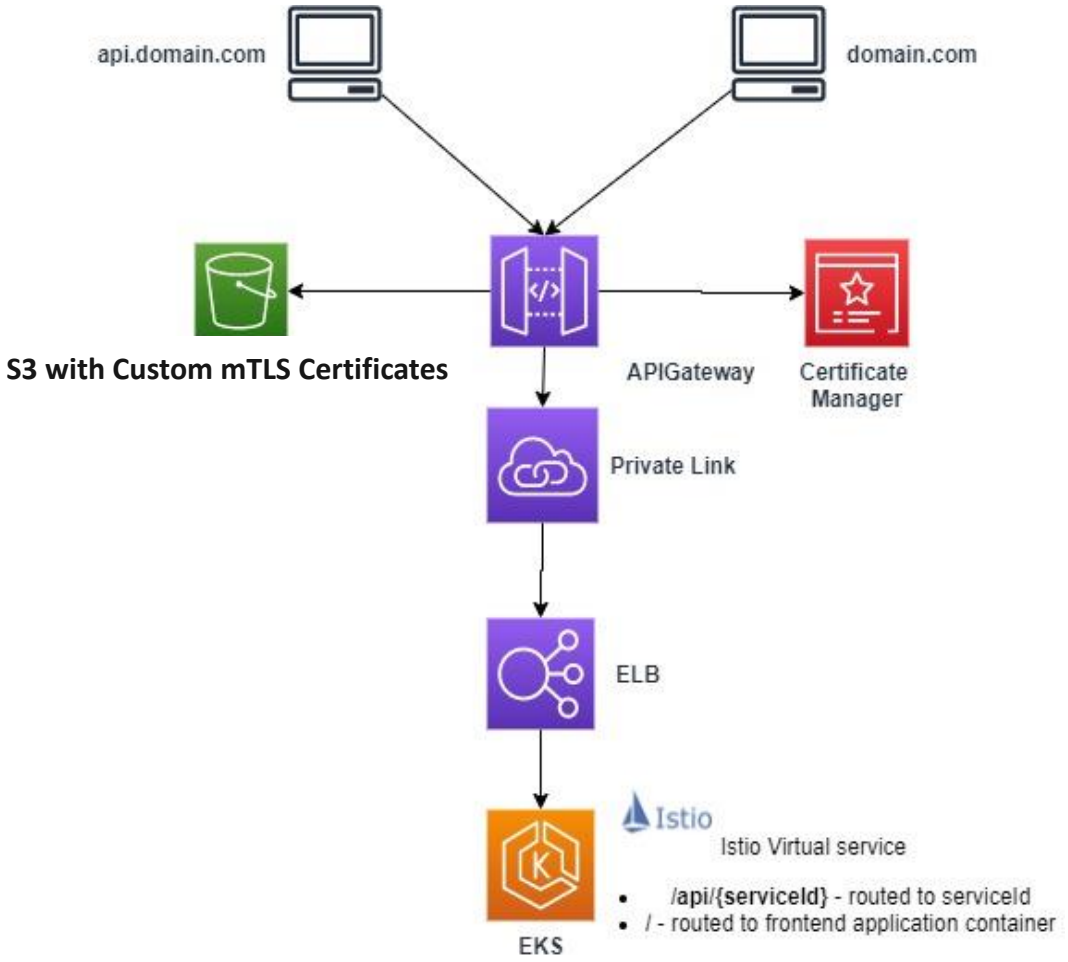
- Other Options
  - AWS service mesh **AppMesh**
  - **AWS VPC lattice** (was not available when we started the project)



First retrospective and Evolution

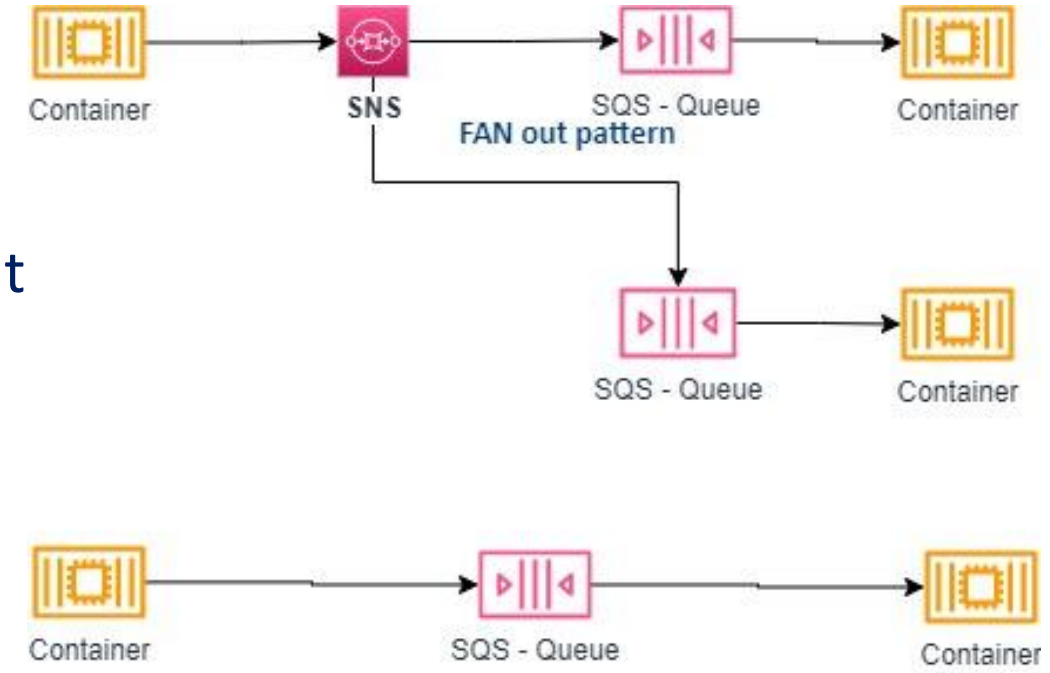
Cloud Architecture Evolution - mTLS for A2A (or STS) communication

- **Amazon API Gateway**
  - As system entry point
  - As HTTPS termination with AWS managed certificate
  - Enforce mTLS for STS requests
  - to expose web app through REST APIGateway we used **Custom domain name** to solve frontend application relative path issues



Cloud Architecture Evolution - Asynchronous messaging

- **SQS and SNS**
- SNS with FAN out pattern
- Setup IAM role for Kubernetes Service account using **IAM Roles for Service Accounts (IRSA)***
- AWS Java SDK v2
- **Apache Camel SQS/SNS** components**



* https://www.eksworkshop.com/docs/security/iam-roles-for-service-accounts/

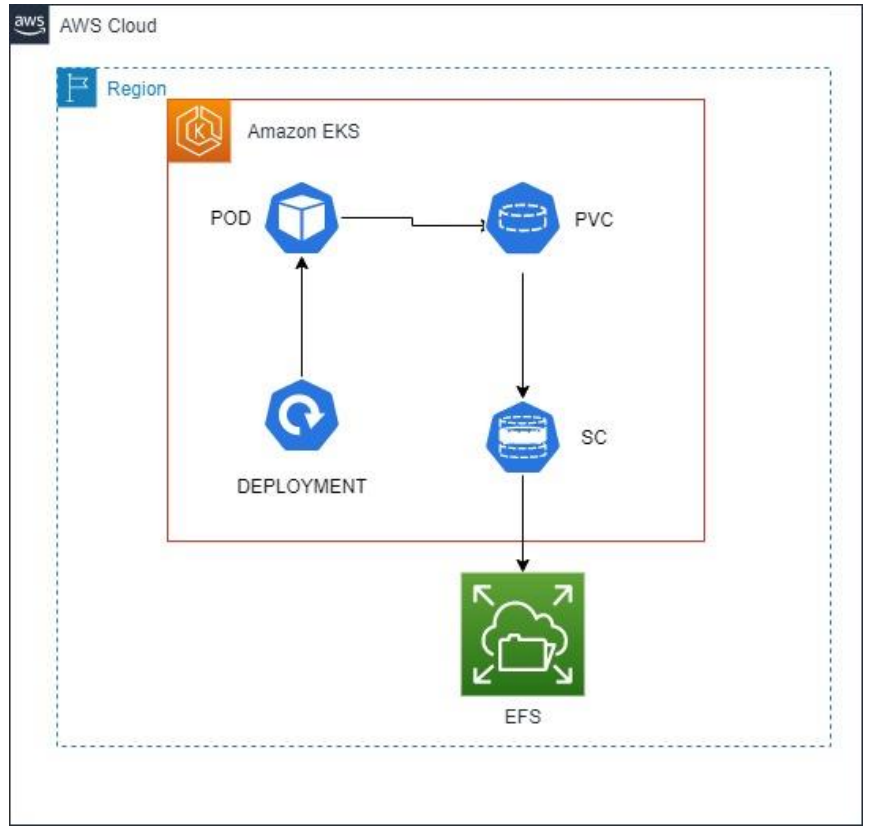
** https://camel.apache.org/components/3.18.x/aws2-sqs-component.html
https://camel.apache.org/components/3.18.x/aws2-sns-component.html

Cloud Architecture Evolution - Shared file storage for report files

Elastic File System (EFS)

It is a cloud storage service designed to provide scalable, elastic, concurrent file storage.

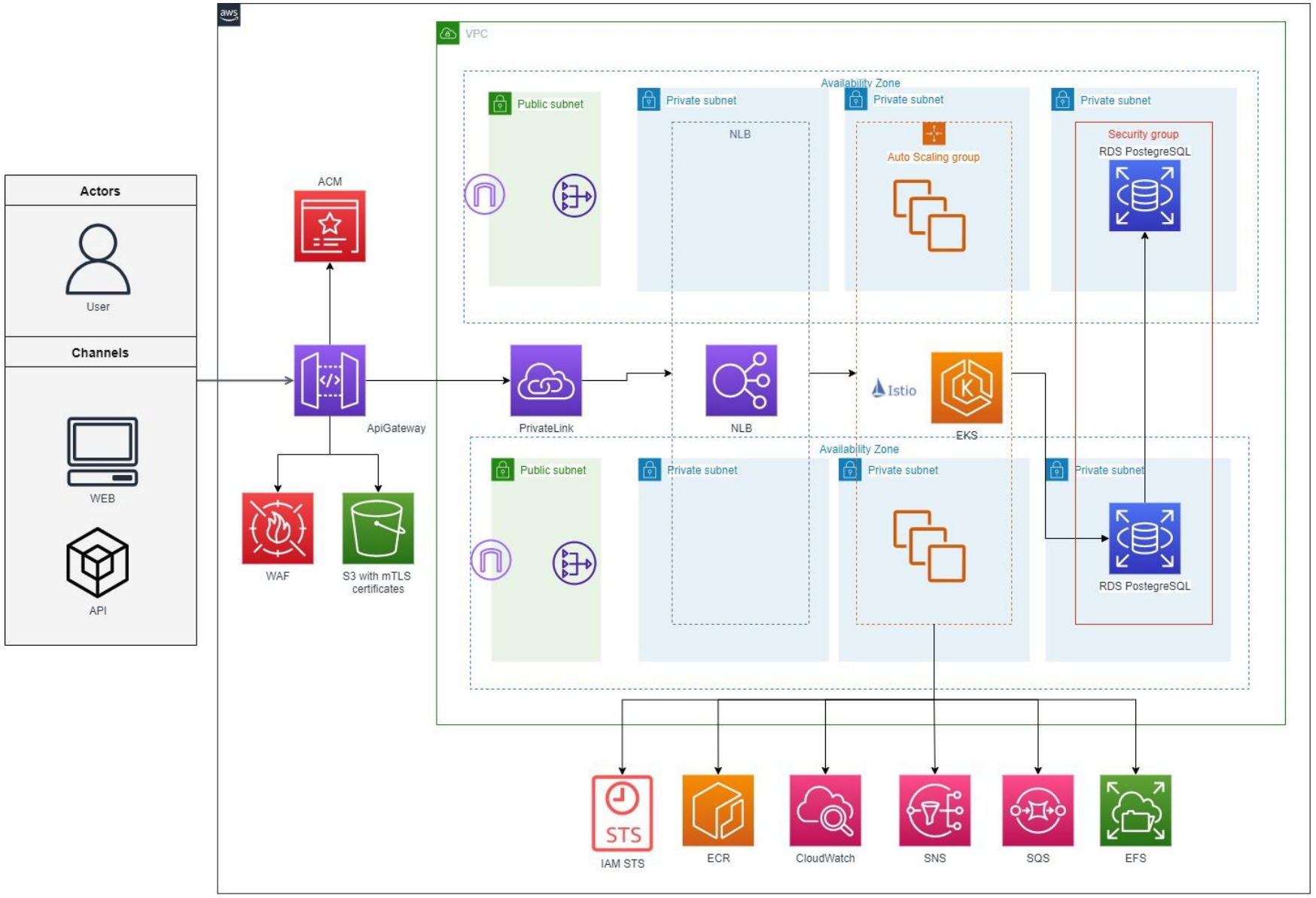
- EFS used via the EKS EFS CSI driver*
- IaC via EKS CSI terraform integration**



* https://github.com/kubernetes-sigs/aws-efs-csi-driver

** https://registry.terraform.io/modules/DNXLabs/eks-efs-csi-driver/aws/latest

Cloud Architecture - Final Evolution



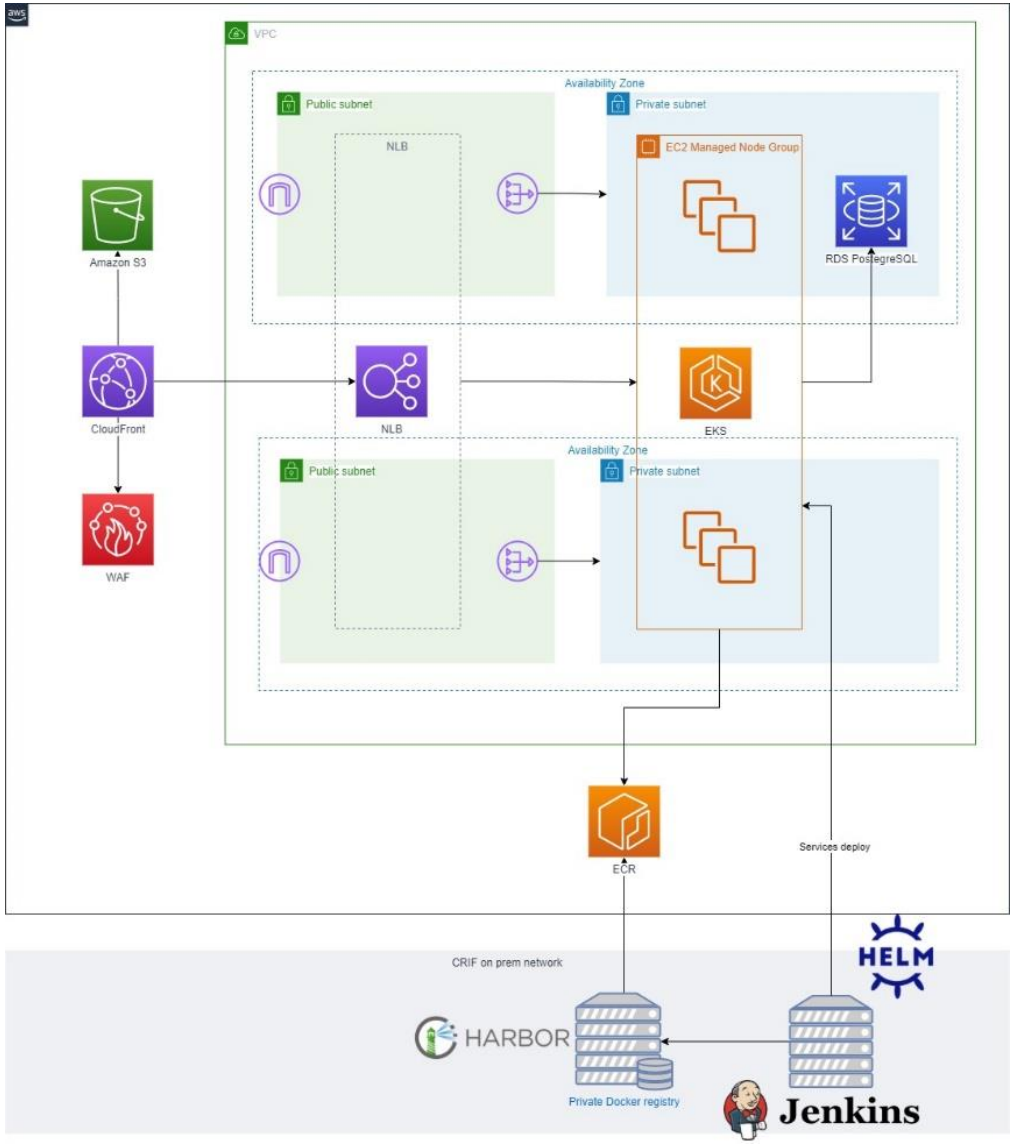
Let’s DevOps IT

First Cloud Architecture – CI/CD and Operations

- EKSCTL
  - Pros: allows easy an EKS cluster setup
  - Cons: it is not an IAC tool, limited only to EKS setup
- No Cloud Infrastructure Automation
  - Manual EKS cluster setup with eksctl tool
  - Manual S3 provision with aws console
  - Manual RDS provision with aws console
- No deployment pipeline
  - Manual upload to ECR of docker images
  - Manual Kubernetes resources deployment
- No monitoring
- No centralized logging

CI/CD

- **IaC** – created a **Terraform** script to setup the cloud architecture and Kubernetes infrastructural components (i.e. Istio)
- **Docker images** – docker images stored in Harbor (Corporate docker registry), mirrored to AWS ECR by Harbor
- **Helm** – adopted Helm as package manager for Kubernetes artifacts
- **Jenkins** – pipelines execution:
  - Pipeline to build docker images from source code
  - Pipeline to upload Helm charts to Harbor CRIF corporate docker registry
  - Secure Pipeline to deploy helm charts to EKS



High Availability

- **MultiAZ deployment** – spread the pod replicas across nodes in different AZ
  - https://aws.github.io/aws-eks-best-practices/reliability/docs/application/

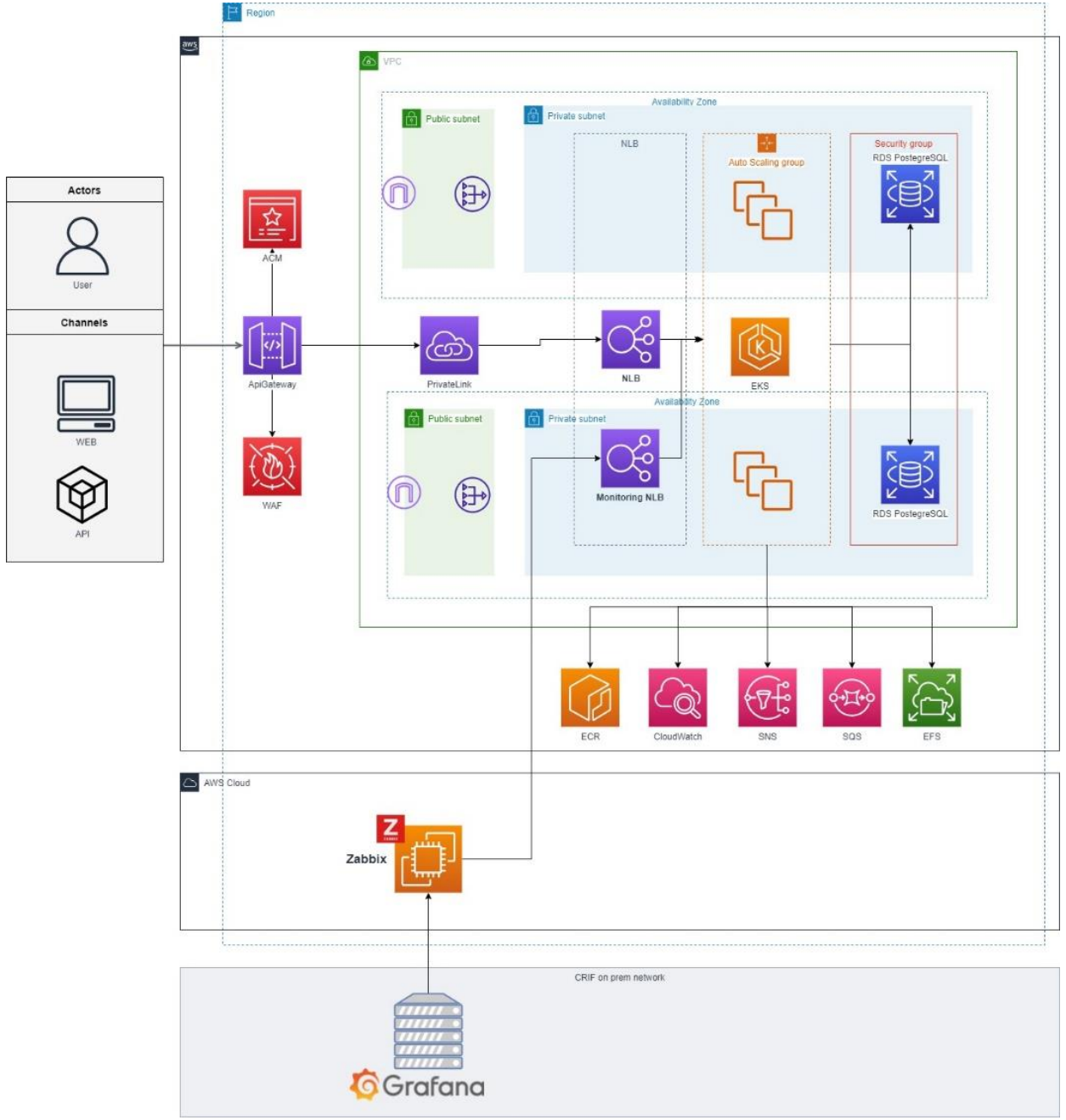
```yaml
affinity:
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - podAffinityTerm:
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - {{ include "app.name" . }}
        topologyKey: topology.kubernetes.io/zone
      weight: 100
    - podAffinityTerm:
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - {{ include "app.name" . }}
        topologyKey: kubernetes.io/hostname
      weight: 100
```

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: {{ include "app.name" . }}-hpa
  namespace: {{ .Release.Namespace }}
  labels:
{{ include "common.labels" . | indent 4 }}
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: {{ include "app.name" . }}
  minReplicas: {{ .Values.replicaCount }}
  maxReplicas: {{ .Values.maxReplicaCount }}
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 80
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300
    scaleUp:
      stabilizationWindowSeconds: 300
```

- HPA – **setup kubernetes horizontal autoscaler** on each deployment
  - https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
  - https://docs.aws.amazon.com/eks/latest/userguide/metrics-server.html

Monitoring

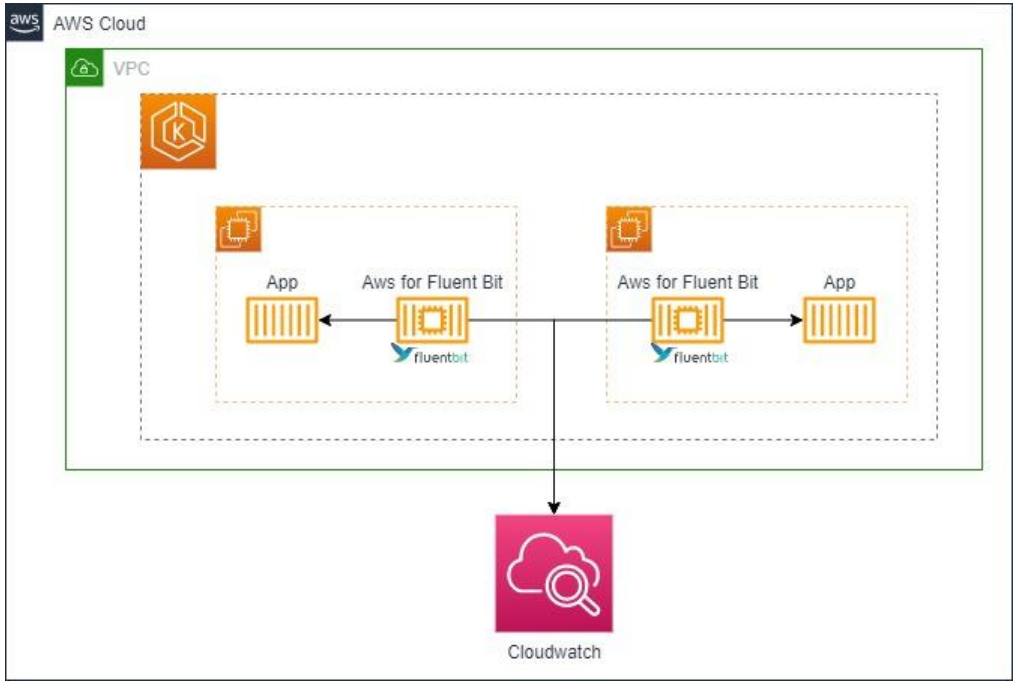
- **Requirements** – use **centralized on premises CRIF monitoring tools**
- Added a new Admin ELB – new private ELB to expose metrics to Zabbix Agent deployed on an EC2
- Deployed Kubernetes Kube State Metrics*
- Grafana - On premises **Grafana to display** containers metrics dashboards



* https://github.com/kubernetes/kube-state-metrics

Logging

- EKS and **Cloudwatch integration**
  - Seamless integration by using CloudWatch Container Insights*
  - **Fluent Bit** usage is well documented for K8s**
  - **IaC** -Helm chart – created a custom helm chart to **deploy FluentBit Cloudwatch integration with custom configuration**

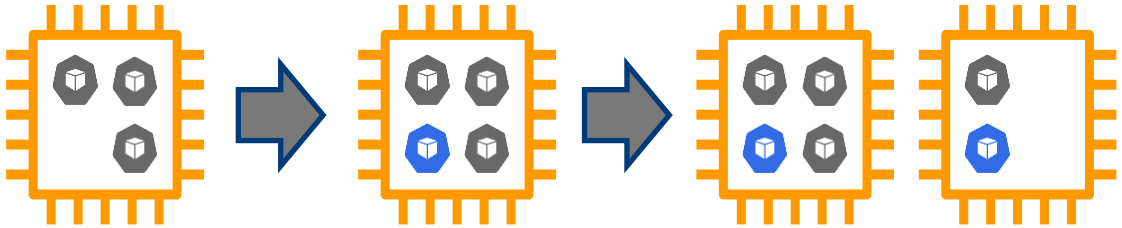


* https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Container-Insights-setup-logs-FluentBit.html

** https://docs.fluentbit.io/manual/installation/kubernetes

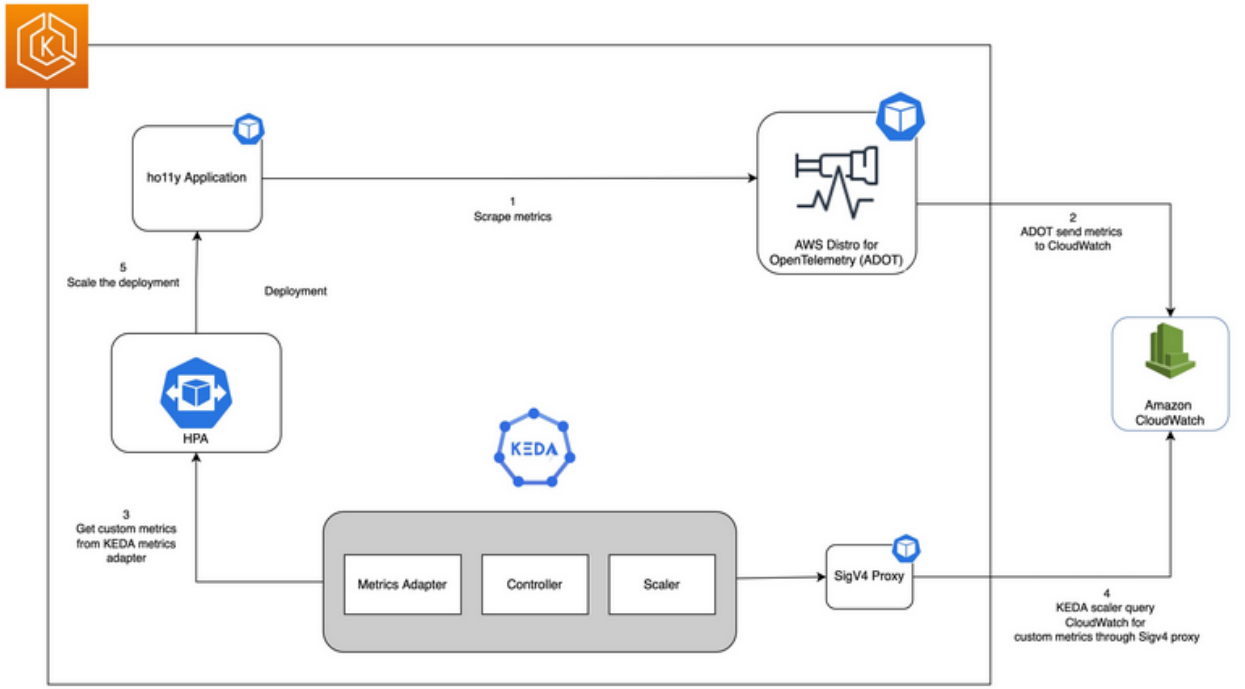
A glimpse into our Future

Future Improvements/Features



- **Autoscaling**
  - Pod scaling with Custom metrics via Horizontal Pod Autoscaler (HPA) + Keda*
  - Cluster Autoscaling with Karpenter (eventually Fargate)

- **Observability**
  - OpenTelemetry with AWS OTEL**
  - AWS X-Ray for Tracing***



* https://aws.amazon.com/it/blogs/mt/proactive-autoscaling-of-kubernetes-workloads-with-keda-using-metrics-ingested-into-amazon-cloudwatch

** https://aws-otel.github.io/

*** https://aws-otel.github.io/docs/getting-started/x-ray

CRIF GLOBAL TECHNOLOGIES

CRIF Together to the next level

Wrap Up

Key takeaways

Start small and iterate

Collaborate with other Teams

Don’t forget Observability

CI/CD and IaC are the key foundations for "doing K8s right"

Prefer managed services to maximize speed of delivery

Thank you!

Please complete the session survey in the mobile app

Fabio Chiodini
AWS

Alex Landini
CRIF

Gabriele Armao
CRIF

SUMMIT

MILANO | 22 GIUGNO 2023