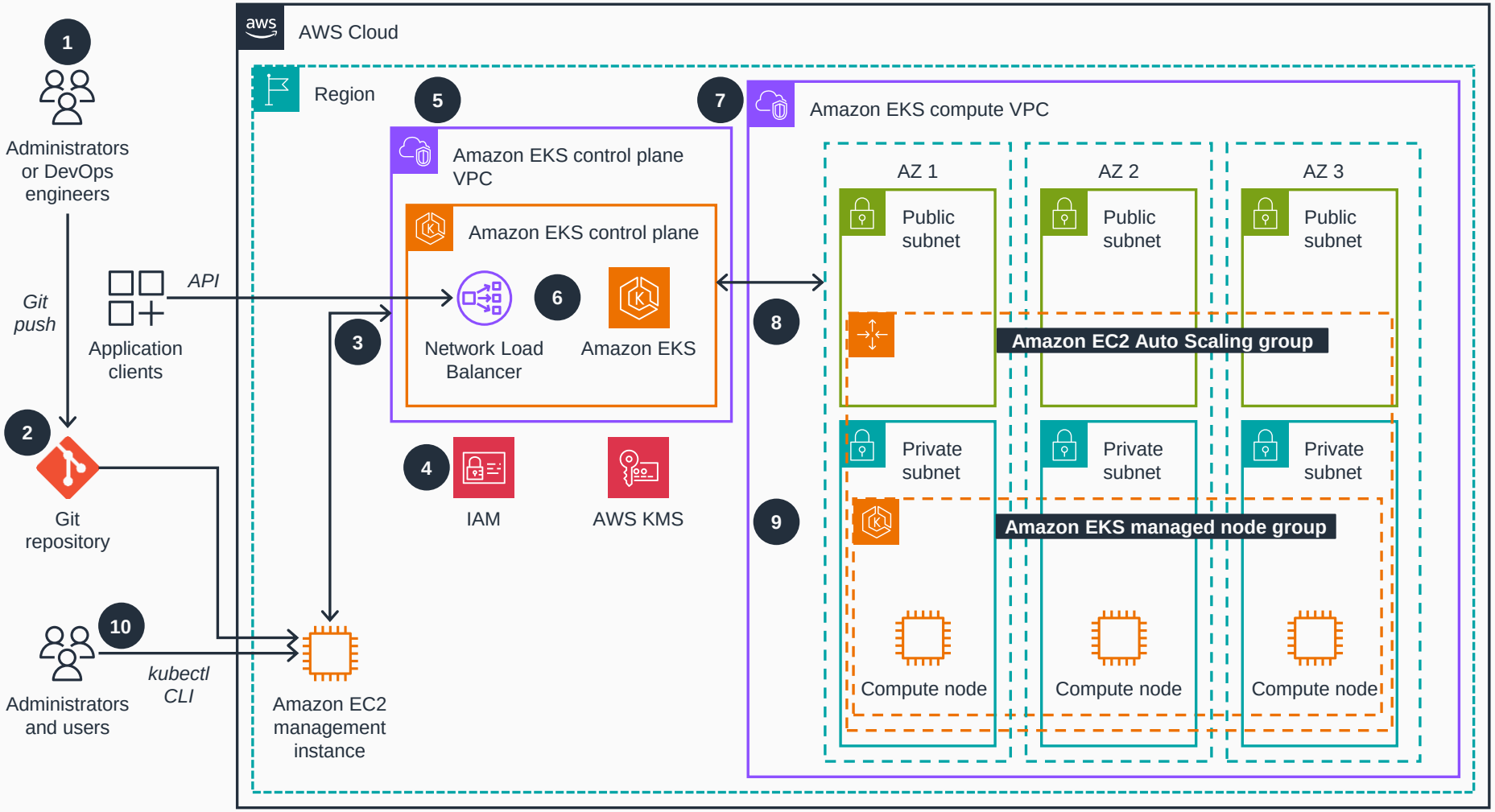
Guidance for Container Runtime Security Monitoring with CNCF Falco and AWS Security Hub (Optional)

This architecture diagram shows how to set up the Amazon Elastic Kubernetes Service (Amazon EKS) cluster needed for this Guidance.*



Optional: To deploy this Guidance, you need an **Amazon Elastic Kubernetes Service (Amazon EKS)** cluster provisioned. These steps show how to provision an **Amazon EKS** cluster using another project.

1. An administrator or DevOps user commits infrastructure as code (IaC) code changes, configured to **Amazon EKS** specification, into the Git repository.
2. **Amazon Elastic Compute Cloud (Amazon EC2)** management instance provisioning is started by the administrator or DevOps user by using **AWS CloudFormation** code obtained from the Git repository.
3. The management instance's *userData* script starts **Amazon EKS** cluster resource deployment processes against the target AWS environment (using *eksctl* command and cluster specification).
4. Required **AWS Identity and Access Management (IAM)** roles and polices and **AWS Key Management Service (AWS KMS)** keys are created.
5. The **Amazon Virtual Private Cloud (Amazon VPC)** for the **Amazon EKS** control plane component is deployed.
6. The **Amazon EKS** control plane components are deployed into the **Amazon EKS** VPC.
7. The **Amazon EKS** compute VPC is deployed for the **Amazon EKS** compute plane.
8. Public and private subnets and other networking components are deployed into the compute VPC.
9. The **Amazon EKS** node groups with compute plane nodes (**Amazon EC2** instances in an **Amazon EC2 Auto Scaling** group) are deployed into the compute VPC and join the **Amazon EKS** cluster.
10. The **Amazon EKS** cluster is available for application deployments. The Kubernetes API is accessible for command line interface (CLI) clients and applications through a Network Load Balancer.

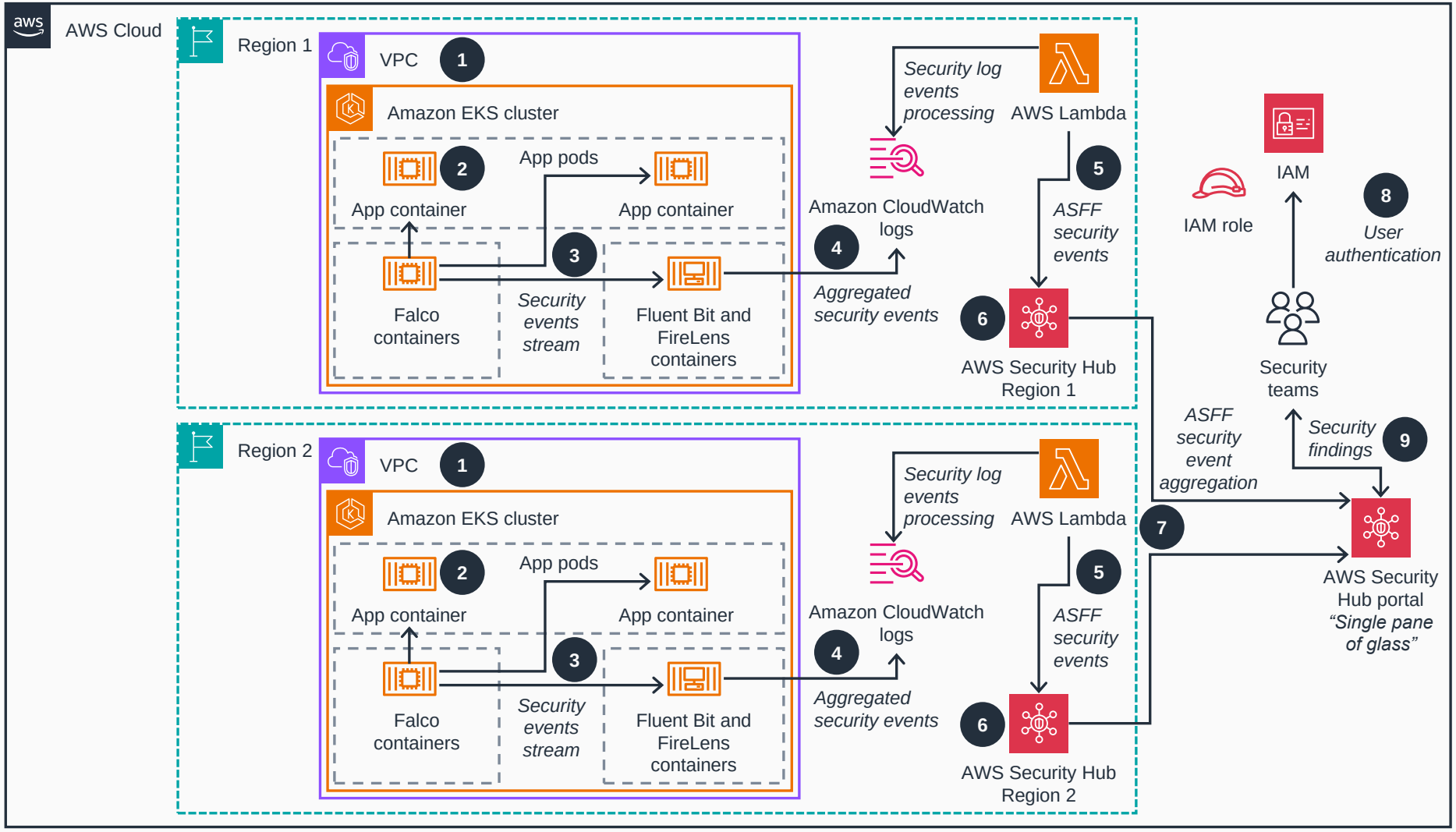
**Workflows that invoke Terraform commands are currently initiated manually and can be automated through GitHub events workflows in the future.*

Reviewed for technical accuracy June 11, 2024

AWS Reference Architecture

Guidance for Container Runtime Security Monitoring with CNCF Falco and AWS Security Hub

This architecture diagram shows how to detect runtime security events with CNCF Falco and send them to the AWS Security Hub portal for triage.



1. The Fluent Bit and FireLens log-event aggregation and the CNCF Falco security monitoring components are deployed into **Amazon EKS** clusters running in different AWS Regions.
2. CNCF Falco containers monitor application containers running on **Amazon EKS** cluster nodes for possible security incidents (based on defined rules) and generate security events.
3. The security events are streamed to the Fluent Bit and FireLens log-event aggregators.
4. Aggregated security events are imported into **Amazon CloudWatch** log streams.
5. **AWS Lambda** functions detect security events in the **CloudWatch** log stream, transform them into AWS Security Finding Format (ASFF) schema, and import them into regional hubs for **AWS Security Hub**.
6. Security findings in ASFF are available in regional **Security Hub** portals.
7. Security findings in ASFF are aggregated into a "single pane of glass" (or centralized) **Security Hub** portal, which includes the regional hubs as members. (This can be one of the regional **Security Hub** instances.)
8. Security team members authenticate into the central **Security Hub** portal using **IAM**, and access is granted according to **IAM** roles.
9. Aggregated security findings and remediation workflows are available in the "single pane of glass" **Security Hub** portal for acknowledgement and triage.

Reviewed for technical accuracy June 11, 2024

AWS Reference Architecture