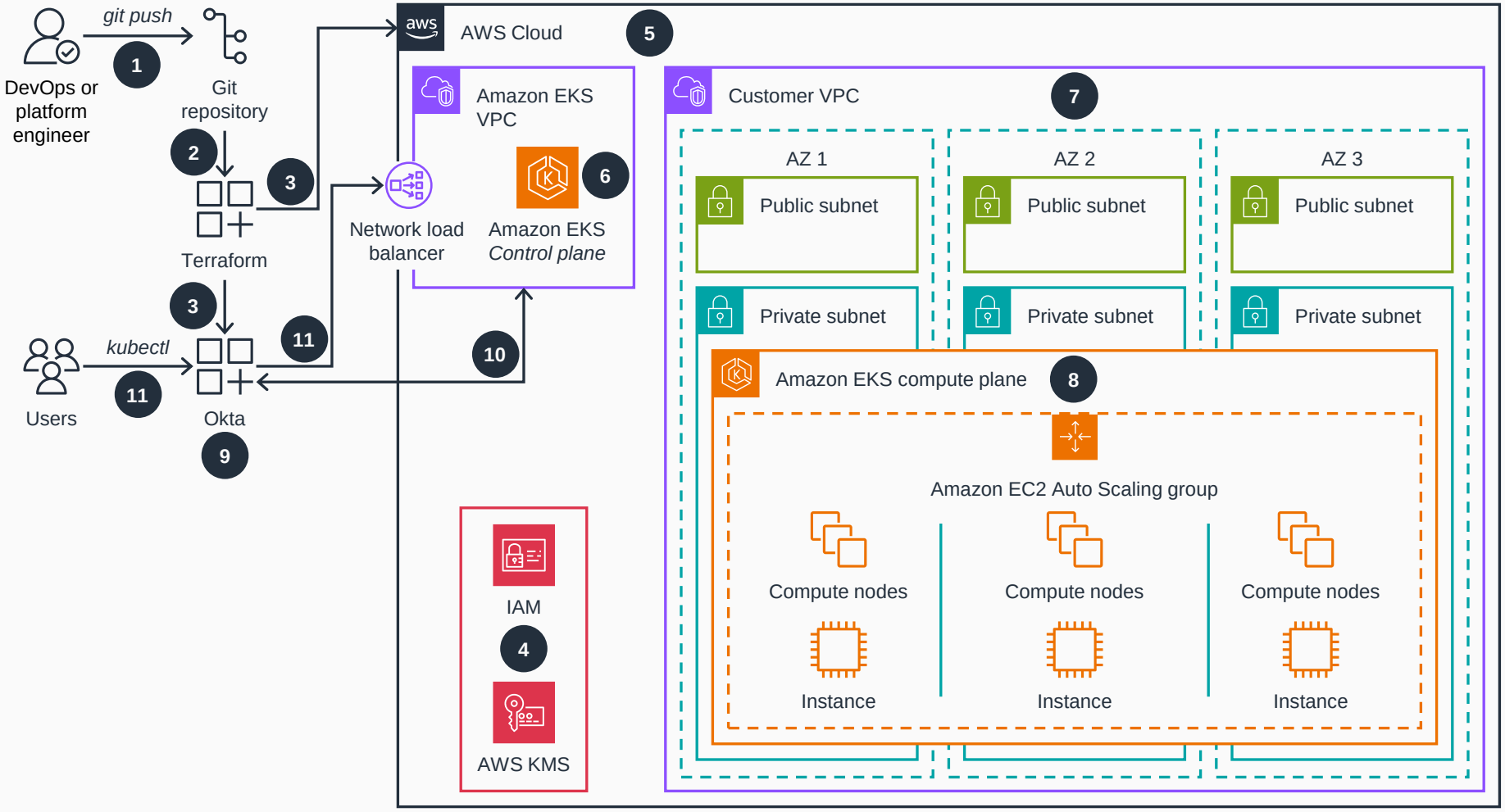
Guidance for Integrating External Single Sign-On Providers with Amazon EKS

This architecture diagram shows how to automate Amazon Elastic Kubernetes Service (Amazon EKS) cluster provisioning and integration with an single sing-on provider like Okta by using Terraform blueprints.



1. Your platform engineer commits and pushes Terraform IaC changes to the project's Git repository.
2. A Terraform infrastructure provisioning workflow is invoked by the code push to the Git repository or is initiated manually by your platform engineer.
3. The Terraform infrastructure provisioning workflow starts resource deployment processes, targeting AWS and Okta environments.
4. Required **AWS Identity and Access Management (IAM)** roles and polices and **AWS Key Management Service (AWS KMS)** keys are created.
5. **Amazon Virtual Private Cloud (Amazon VPC)** environments for **Amazon Elastic Kubernetes Service (Amazon EKS)** control plane and related networking components are deployed.
6. An **Amazon EKS** cluster control plane is deployed into the **Amazon EKS** virtual private cloud (VPC). The cluster control plane is provisioned across multiple Availability Zones (AZs) and is fronted by a Network Load Balancer.
7. Your VPC is deployed with public and private subnets and other networking components across multiple AZs.
8. An **Amazon EKS** compute plane, with managed node groups containing **Amazon Elastic Compute Cloud (Amazon EC2)** compute nodes, is deployed into your VPC.
9. Okta resources, an OAuth server, users, groups, and role assignments are created in the specified Okta organization.
10. An integration between **Amazon EKS** and Okta is created, along with required Kubernetes roles and role bindings.
11. The **Amazon EKS** cluster is available for applications and end users. The Kubernetes API is accessible using a Network Load Balancer with Okta single sign-on user authentication.

Reviewed for technical accuracy November 7, 2024

AWS Reference Architecture