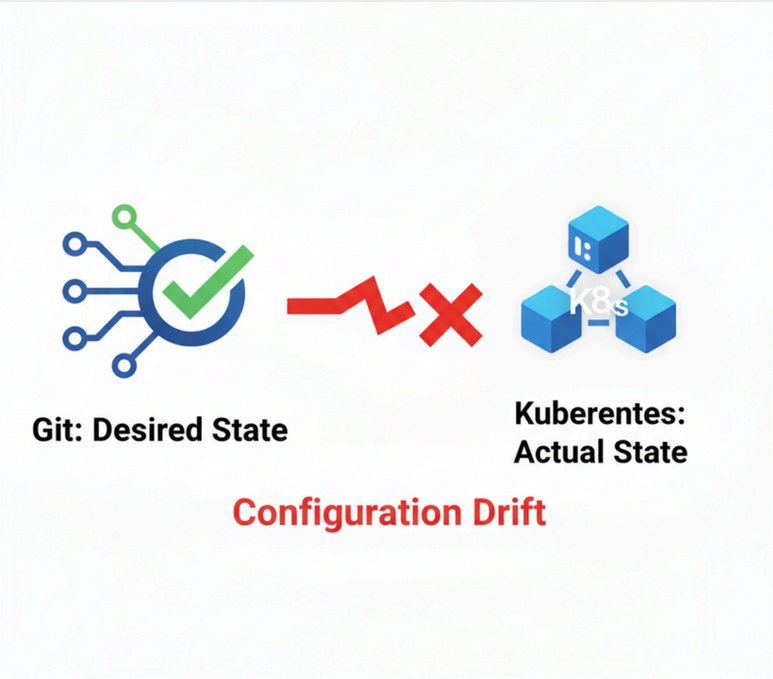
SOLVING KUBERNETES CONFIGURATION DRIFT WITH A SECURE GITOPS PIPELINE

An Implementation Using Argo CD, Jenkins, and SonarQube on AWS EKS

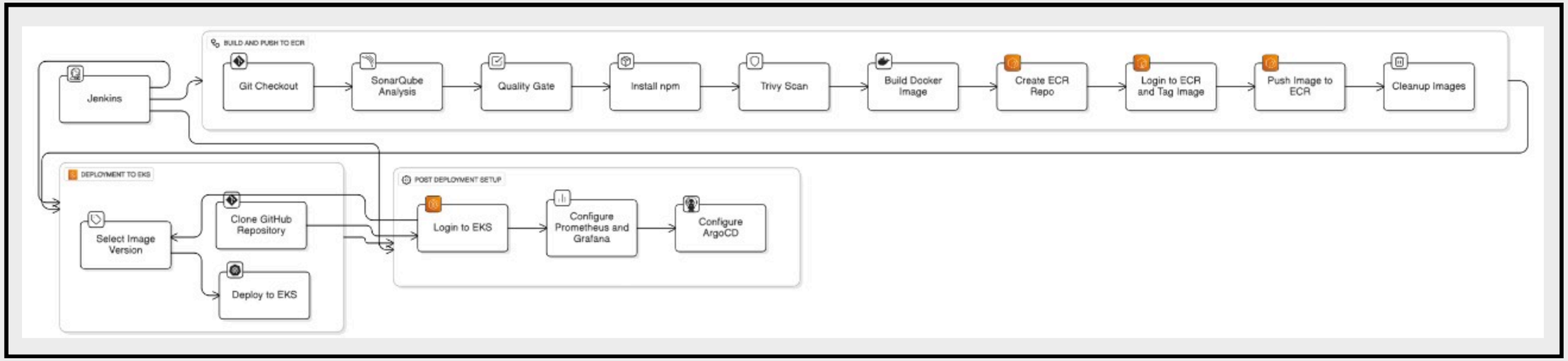
Presented by: Vasu Desai (22070122050), Dhaval Bhimani(22070122051), Dhruv Patel(22070122054), Harsh Kharwar(22070122074)

THE CORE PROBLEM: CONFIGURATION DRIFT



- What is it? Configuration drift occurs when the live state of a Kubernetes cluster no longer matches the intended configuration defined in its source files (e.g., YAMLs in Git).
- Why does it happen? It's often caused by urgent manual changes (kubectl apply -f), out-of-band updates, or inconsistent deployment practices.
- Why is it dangerous? It leads to unreliable deployments, failed rollbacks, security vulnerabilities, and makes it impossible to know the true state of your infrastructure.

THE SOLUTION: A SECURE GITOPS WORKFLOW



- The Principle: GitOps. We treat our Git repository as the single source of truth. The only way to change the infrastructure is to change the code in Git.
- The Reconciler: Argo CD. This tool continuously compares the live cluster state against the state defined in Git. If there's any drift, it automatically syncs the cluster to match the repository.
- The Quality Guard: SonarQube. Before any code is deployed, it first passes through a SonarQube Quality Gate to ensure it is secure and well-written.

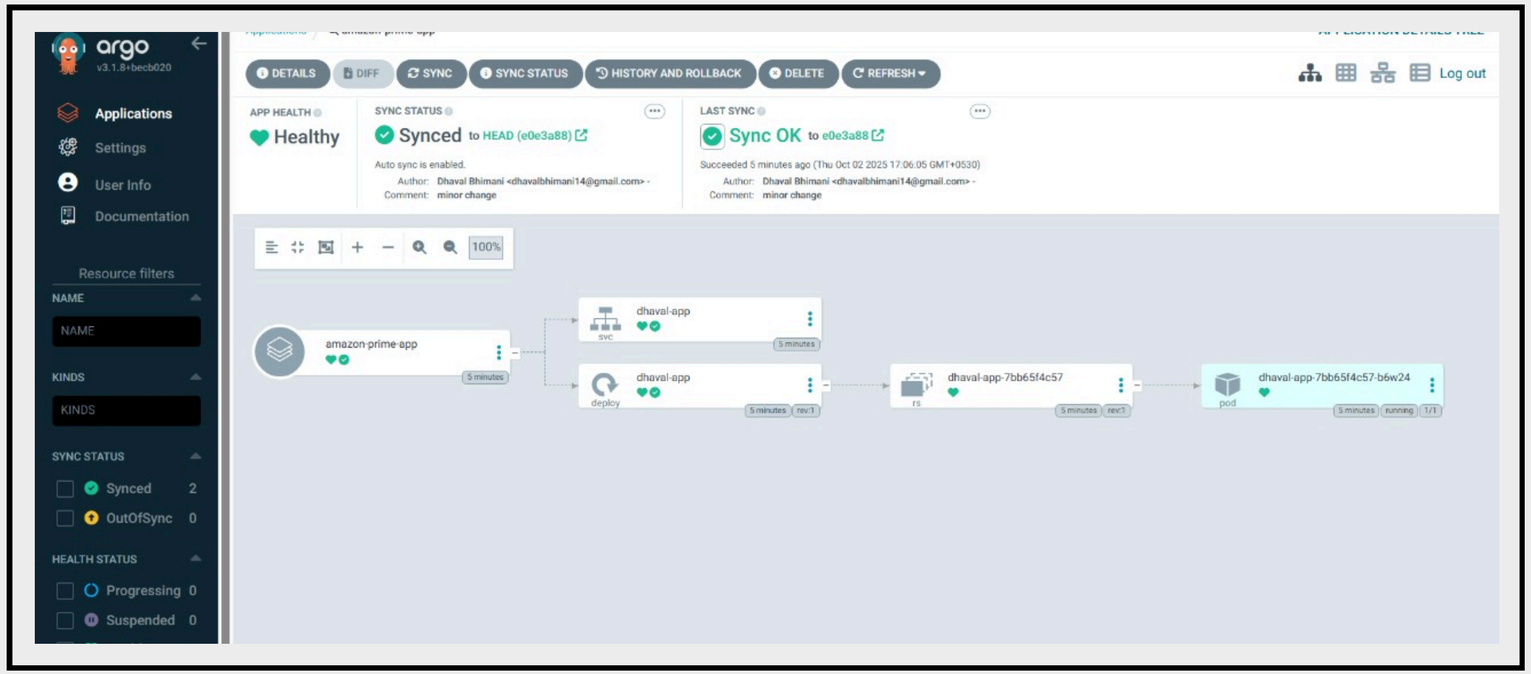
THE CI PHASE: PREPARING A SECURE ARTIFACT

Stage View

	Declarative: Tool Install	1. Git Checkout	2. SonarQube Analysis	3. Quality Gate	4. Install npm	5. Trivy Scan	6. Build Docker Image	7. Create ECR repo	8. Login to ECR & tag image	9. Push image to ECR	10. Cleanup Images
Average stage times: (full run time: ~9min 0s)	3s	3s	1min 22s	359ms	13s	11s	2min 53s	2s	666ms	22s	306ms
#3 Oct 02 15:46 1 commit	257ms	1s	2min 10s	261ms	6s	3s	5min 46s	4s	1s	44s	536ms
#2 Oct 02 No	5s	5s	34s	458ms	20s	19s	1s	120ms	79ms	74ms	76ms

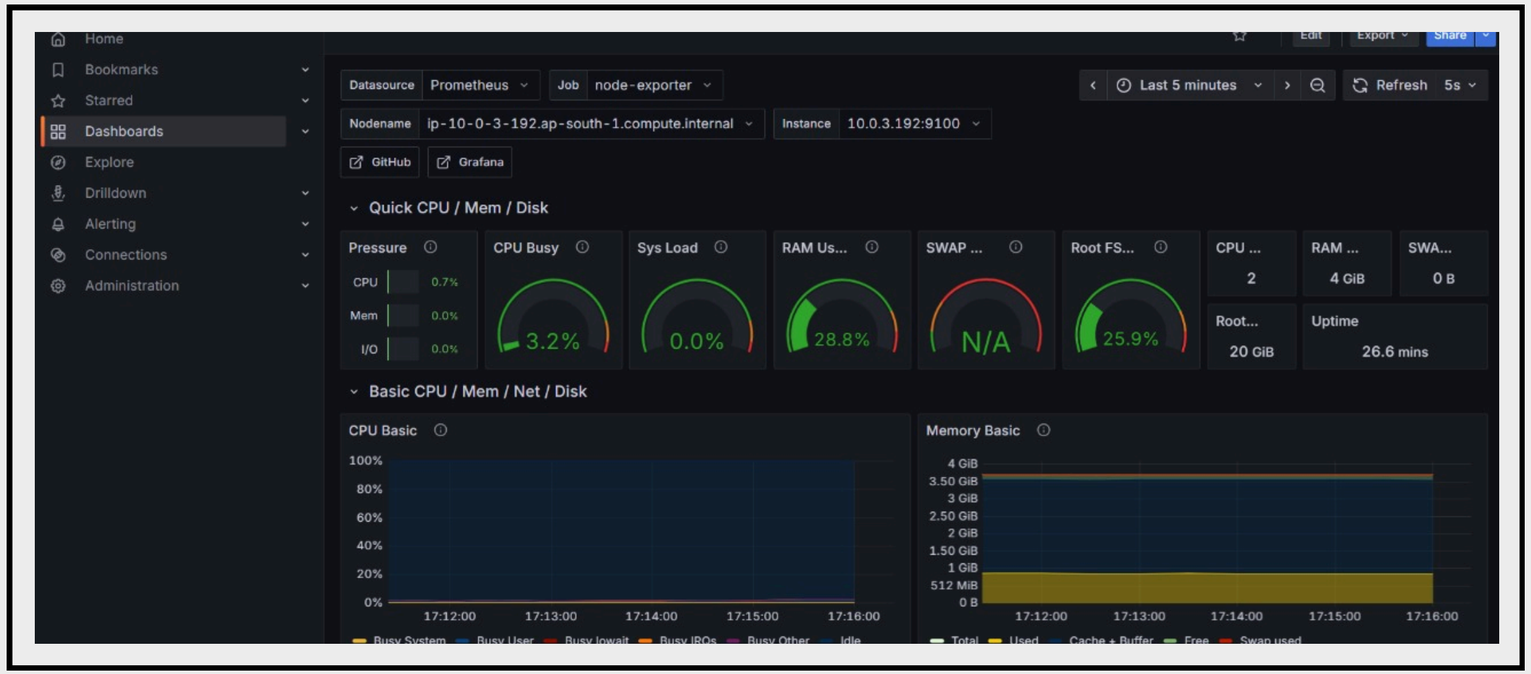
- Before GitOps can deploy anything, our CI pipeline, orchestrated by Jenkins, ensures the application is ready.
- SonarQube Analysis: Code is statically analyzed for bugs, vulnerabilities, and code smells.
- Quality Gate: This is a critical checkpoint. The pipeline stops if the code doesn't meet the defined quality and security standards.
- Build & Push: Only after passing the gate is the Docker image built and pushed to our Amazon ECR registry.

THE CD PHASE: ARGO CD IN ACTION



- This is the core of our solution to configuration drift.
- Monitoring: Argo CD monitors the Git repository that holds our Kubernetes deployment manifests.
- Reconciliation: It automatically detects any difference between the manifests in Git and the live state of the EKS cluster.
- Syncing: Argo CD automatically applies the necessary changes to the cluster to resolve the drift, ensuring the cluster state always matches the source of truth.

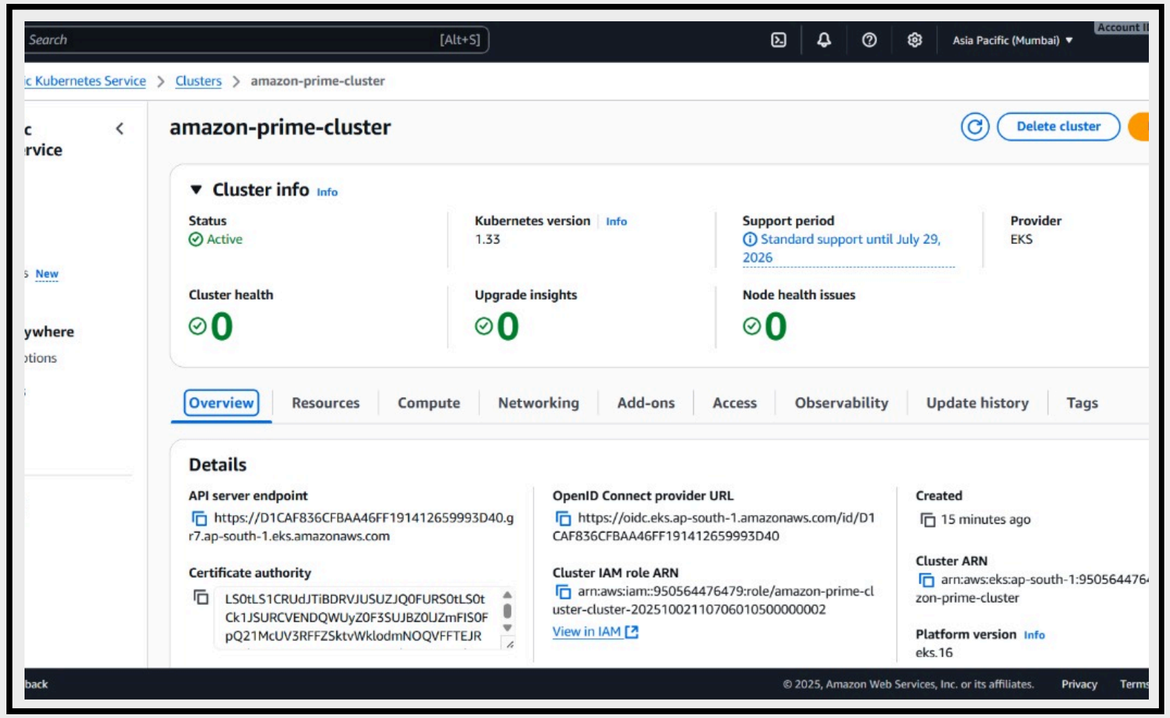
PROVING IT WORKS: A HEALTHY, MONITORED SYSTEM



- The result of this GitOps workflow is a consistently deployed and stable application.
- Because our deployments are reliable, we can trust the data from our monitoring tools.
- Using Prometheus and Grafana, we can observe the real-time health and performance of the application and its underlying nodes, confident that what we are seeing reflects our intended configuration.

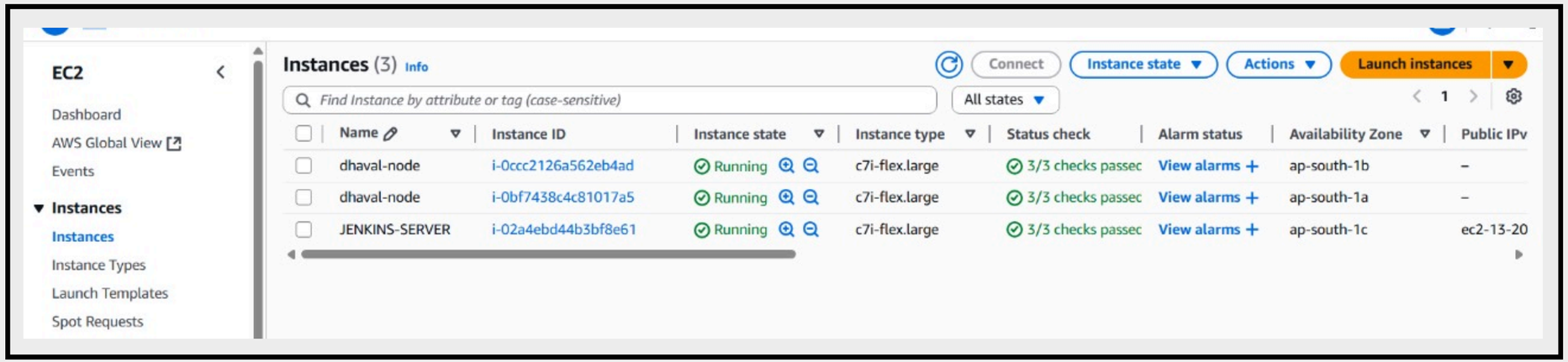
Grafana

THE ORCHESTRATION LAYER: AMAZON EKS



- The entire solution is built on a robust, scalable cloud infrastructure provided by AWS.
- Amazon EKS (Elastic Kubernetes Service) provides the managed Kubernetes control plane.
- It automates complex tasks like patching and updates, allowing us to focus on the application, not infrastructure management.
- Our cluster, amazon-prime-cluster, is the environment where Argo CD deploys and manages our application.

THE COMPUTE LAYER: EC2 WORKER NODES

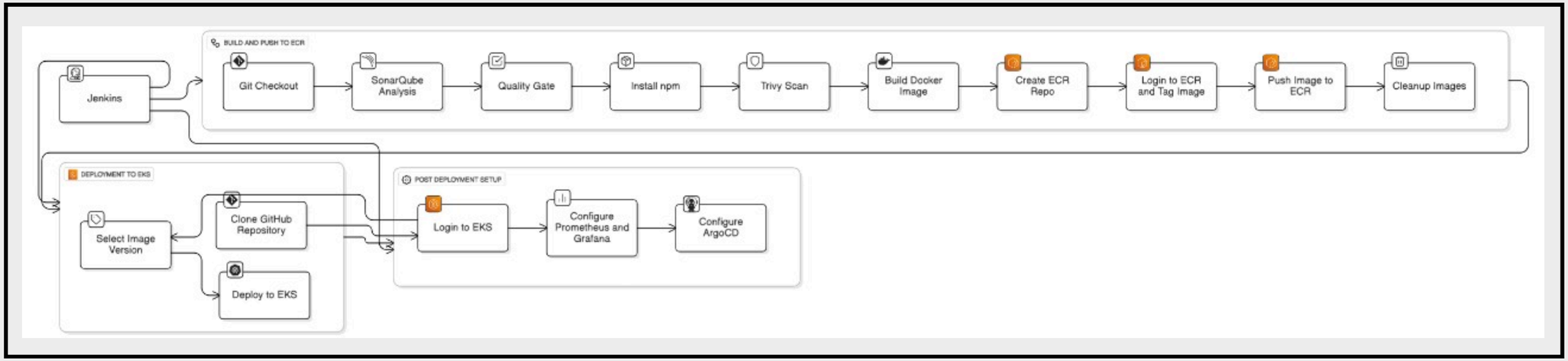


- The EKS cluster manages a group of worker machines, which are Amazon EC2 instances.
- These nodes are responsible for running the actual containerized application pods.
- Our setup includes a dedicated EC2 instance for the Jenkins server and a separate node group for the EKS worker nodes.
- This separation ensures our build processes and application workloads are isolated from each other.

CONCLUSION & KEY TAKEAWAYS

- Problem Solved: We eliminated manual deployments and the associated risk of configuration drift.
- Core Solution: By implementing a GitOps workflow with Argo CD, we made Git the single source of truth, leading to reliable, auditable, and automated deployments.
- Security Integrated: The CI pipeline, enforced by a SonarQube Quality Gate, ensures that only secure and high-quality code is deployed.
- Overall Benefit: The result is a modern, secure, and resilient application delivery process.

QUESTIONS?


BUILD AND PUSH TO ECR
Jenkins
Git Checkout
SonarQube Analysis
Quality Gate
Install npm
Trivy Scan
Build Docker Image
Create ECR Repo
Login to ECR and Tag Image
Push Image to ECR
Cleanup Images
DEPLOYMENT TO EKS
Select Image Version
Clone GitHub Repository
Deploy to EKS
POST DEPLOYMENT SETUP
Login to EKS
Configure Prometheus and Grafana
Configure ArgoCD