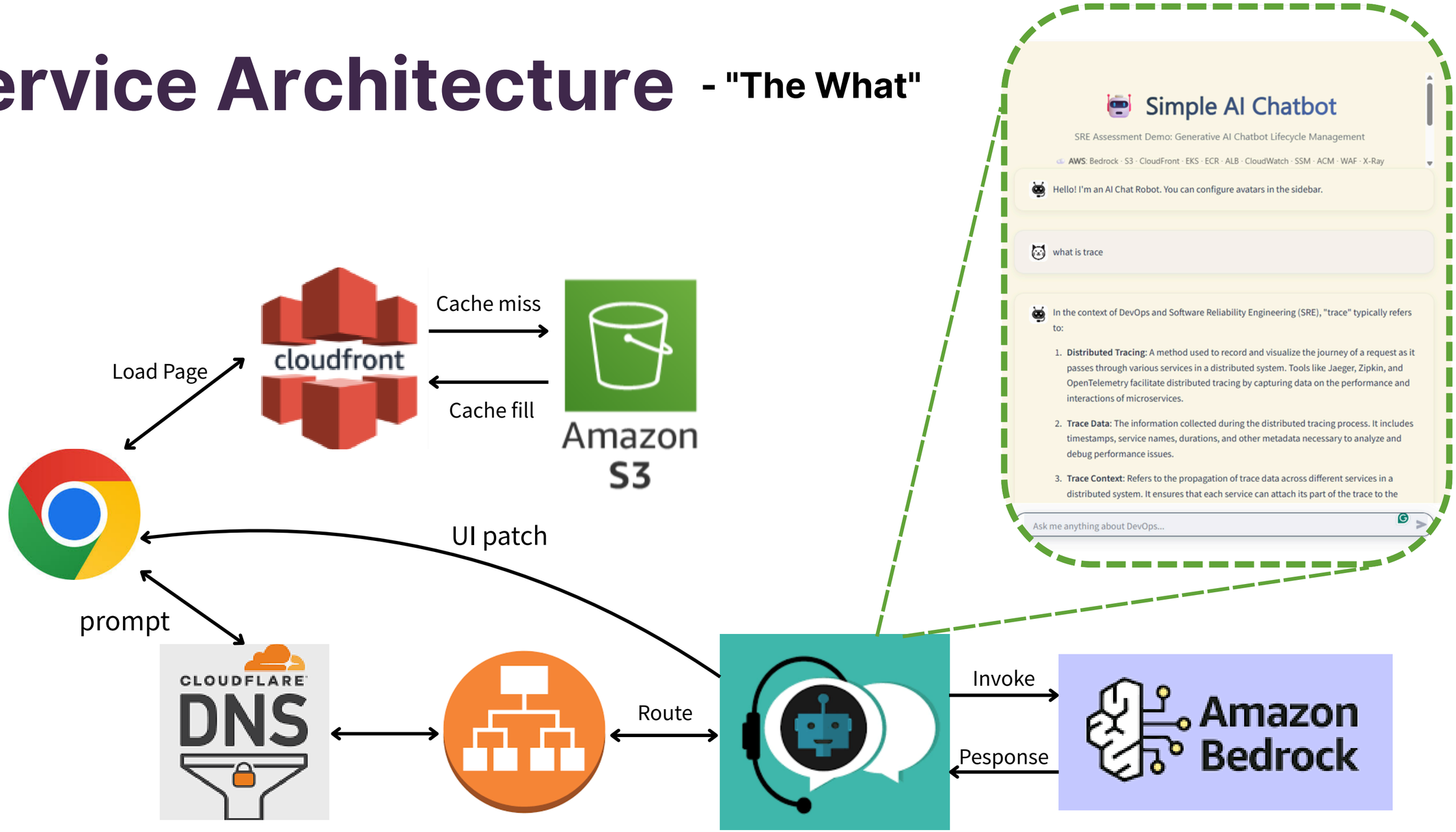
Project Goals & Architecture

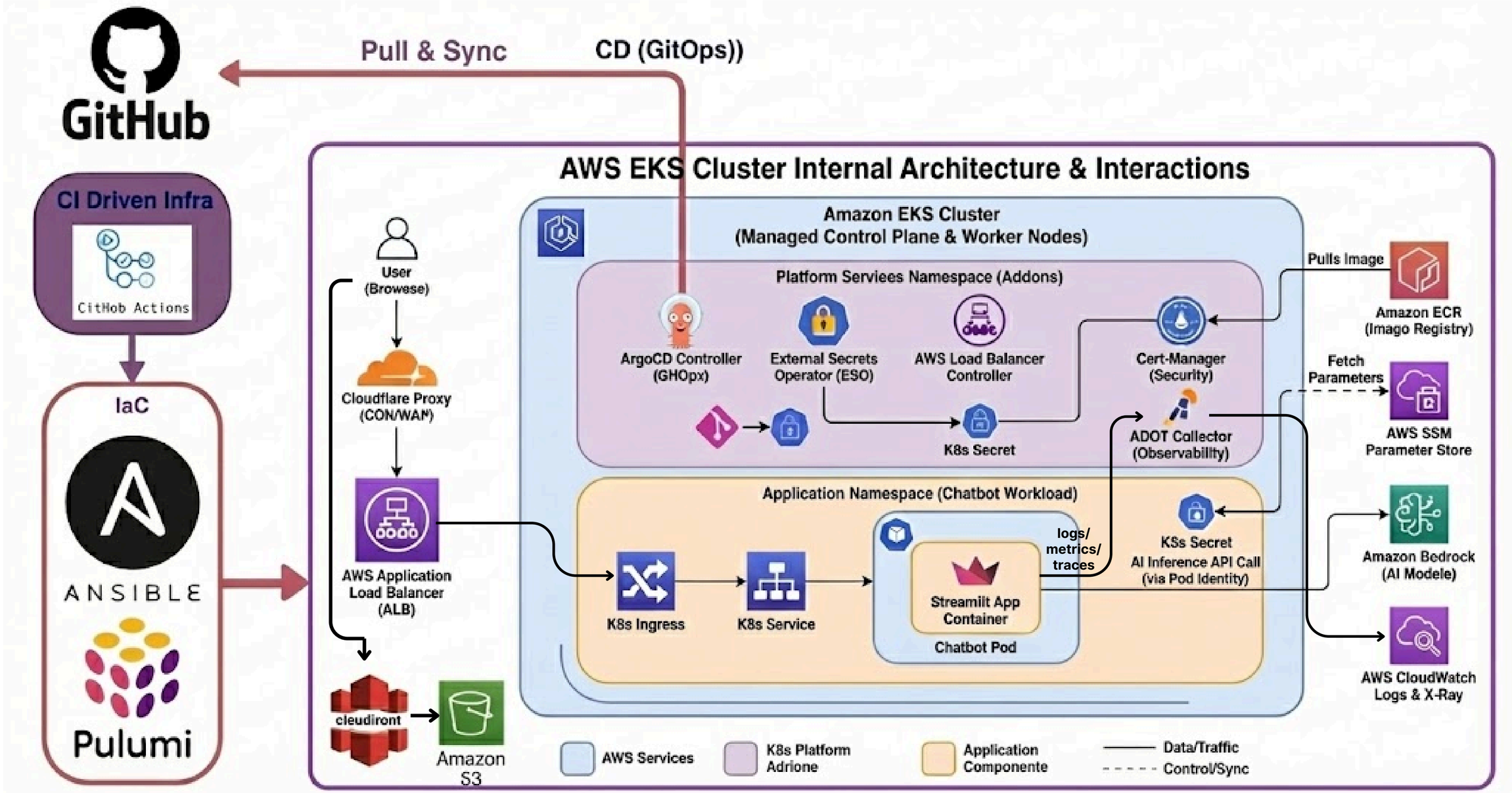
- **Objective:**
  Build an AI Q&A Chatbot platform on AWS with full lifecycle management.

- **Core Requirements:**
  - **Automation**: End-to-end infrastructure provisioning using Ansible & Pulumi.
  - **Modern Stack**: Containerized workload on EKS integrating with Amazon Bedrock.
  - **Focus Areas**: Prioritizing IaC, CI/CD, Observability, and Least-Privilege Security.
  - **Lifecycle**: "One-click" capability to Deploy, Update, and Destroy in a clean environment.

Service Architecture - "The What"



Platform Architecture - "The How"



Infrastructure as Code & Lifecycle Management

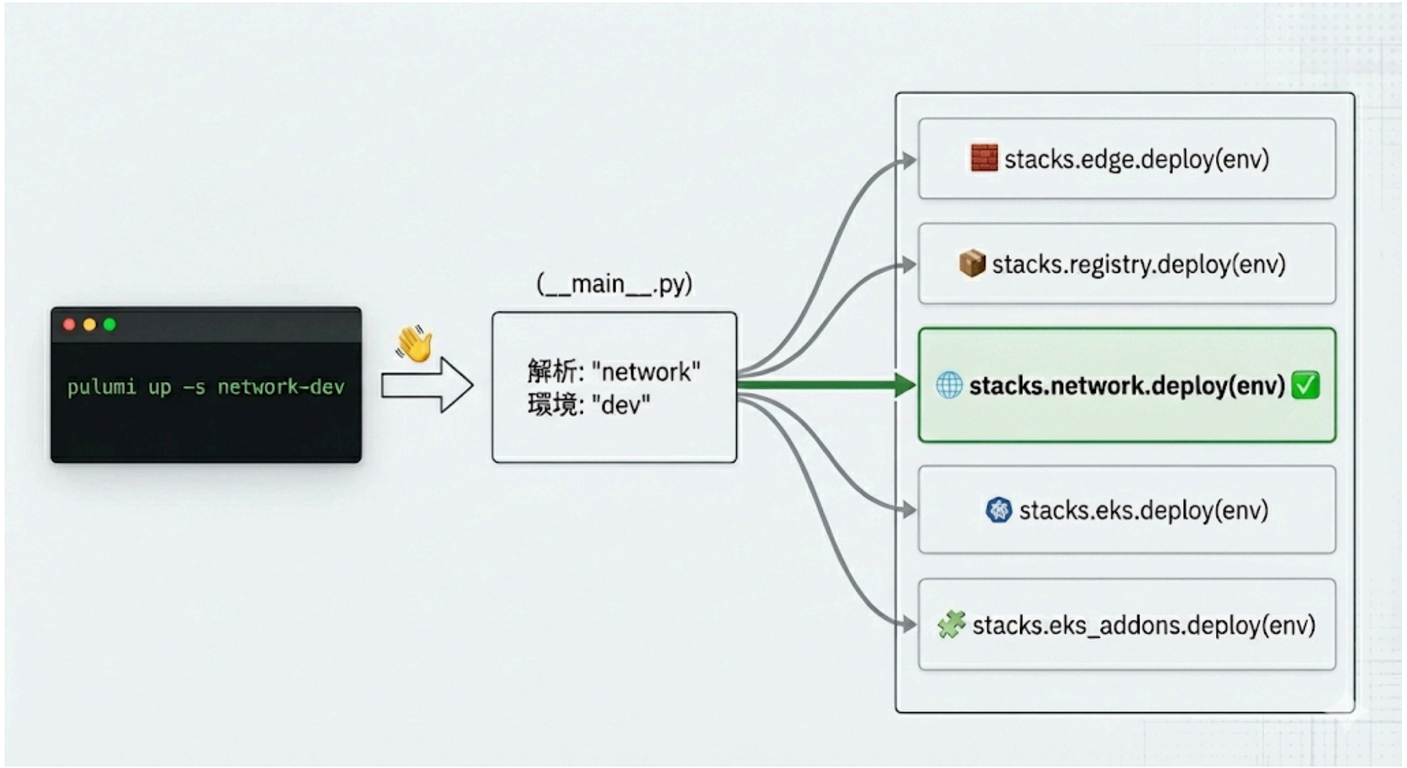
- **Core Strategy:**
  - **IaC Approach:**
    - **Ansible:** Acts as a unified control layer to wrap Pulumi execution, simplifying stack management and standardizing deployment workflows.
    - **Pulumi (Python):** Defines AWS resources (EKS, VPC, IAM) using standard Python objects and logic.
  - **Full Lifecycle Automation:**
    - **Bootstrap & Provision:** Setup of prerequisites and build a stack with dependency wiring
    - **Delivery**: Static asset upload & app deployment.
    - **Destroy:** Clean teardown logic to ensure no residual resources.
  - **Clean Account Guarantee:** Designed for ephemeral environments; verified by automated up and destroy cycles.

IaC Workflow: Infrastructure

Pulumi

As the Infrastructure Architect

- **Modular Design:** Decomposes infrastructure into decoupled Stacks
- **Programmatic Provisioning:** Uses Python to dynamically define AWS resources based on environment variables.
- **State Management:** Tracks resource state and dependencies.
- **Fine-Grained Security:** Automates the creation of IRSA and Pod Identity roles, rigorously enforcing the Principle of Least Privilege.

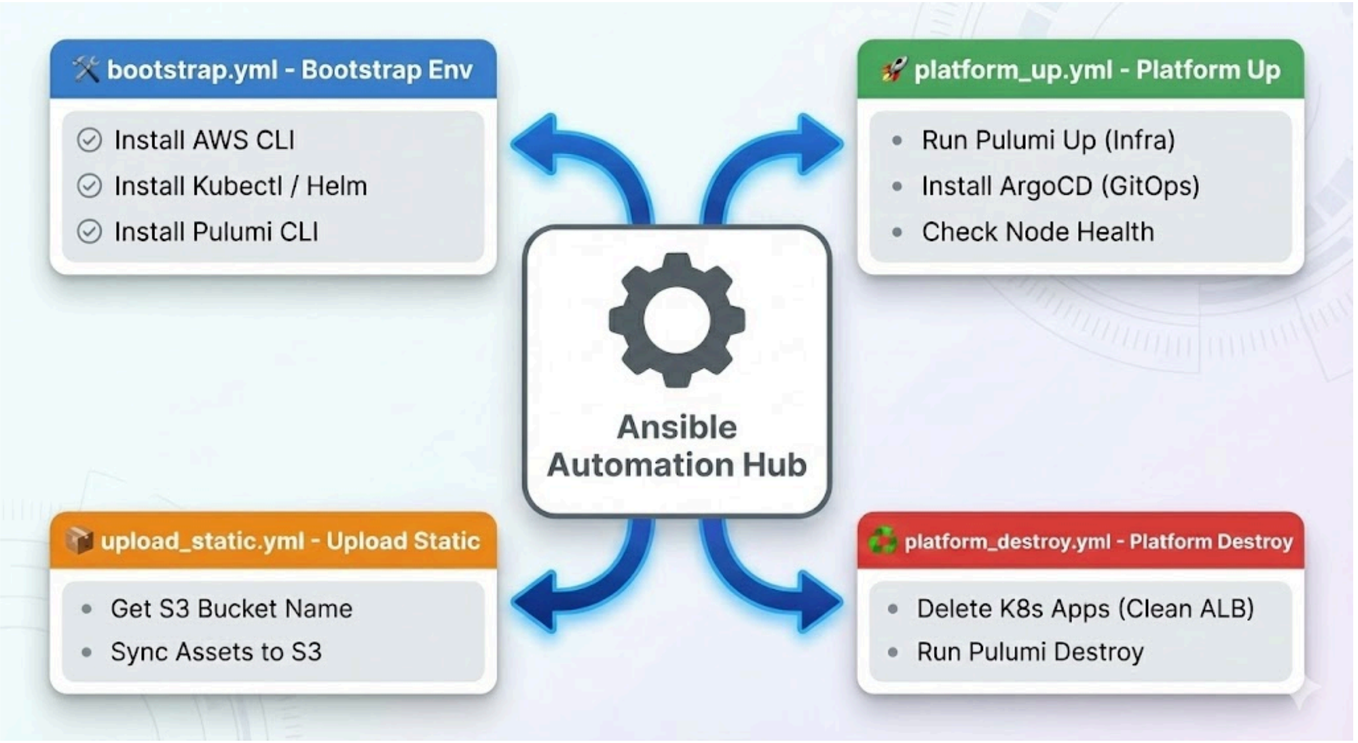


IaC Workflow: Automation

Ansible

As the Lifecycle Orchestrator

- **Environment Bootstrap:** Standardizes the execution environment.
- **IaC Driver:** Wraps Pulumi commands to provision infrastructure.
- **Platform Bootstrapping:** Initializes the EKS cluster, installs ArgoCD, and verifies node health before deployment.
- **Asset Delivery:** Synchronizes static frontend assets to S3
- **Graceful Teardown:** Manages strict dependency ordering during destruction to guarantee a Clean AWS Account.



CI/CD & GitOps

- **Hybrid Architecture:**
  - **CI (Push-Based):** GitHub Actions handles build, test, and infrastructure provisioning.
  - **CD (Pull-Based):** ArgoCD manages Kubernetes resource synchronization (GitOps).

- **Security-First CI Pipeline:**
  - **Keyless Authentication:** Uses OIDC (OpenID Connect) to authenticate with AWS (AssumeRole), eliminating long-term access keys.
  - **Ephemeral Runners:** All build steps run in isolated Docker containers defined by the pulumi-build-env image.

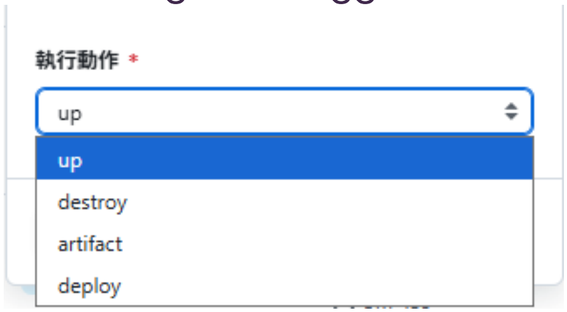
- **GitOps Implementation:**
  - **Single Source of Truth:** Cluster state is strictly defined in Git (k8s/ directory).
  - **App-of-Apps Pattern:** Uses a root Application to manage platform components and workloads hierarchically.
  - **Self-Healing:** ArgoCD automatically detects and corrects configuration drift.

From Code Push to Infra Lifecycle

GitHub Actions (CI)

Ops Mode (Manual Trigger):

Infrastructure Up
Build & Push
ArgoCD Trigger



- **On-Demand Infrastructure Ops:**
  - Uses workflow_dispatch for critical actions (up, destroy, deploy).
  - Provides a safe, manual gate for full-stack lifecycle management.

Dev Mode (Git Push):

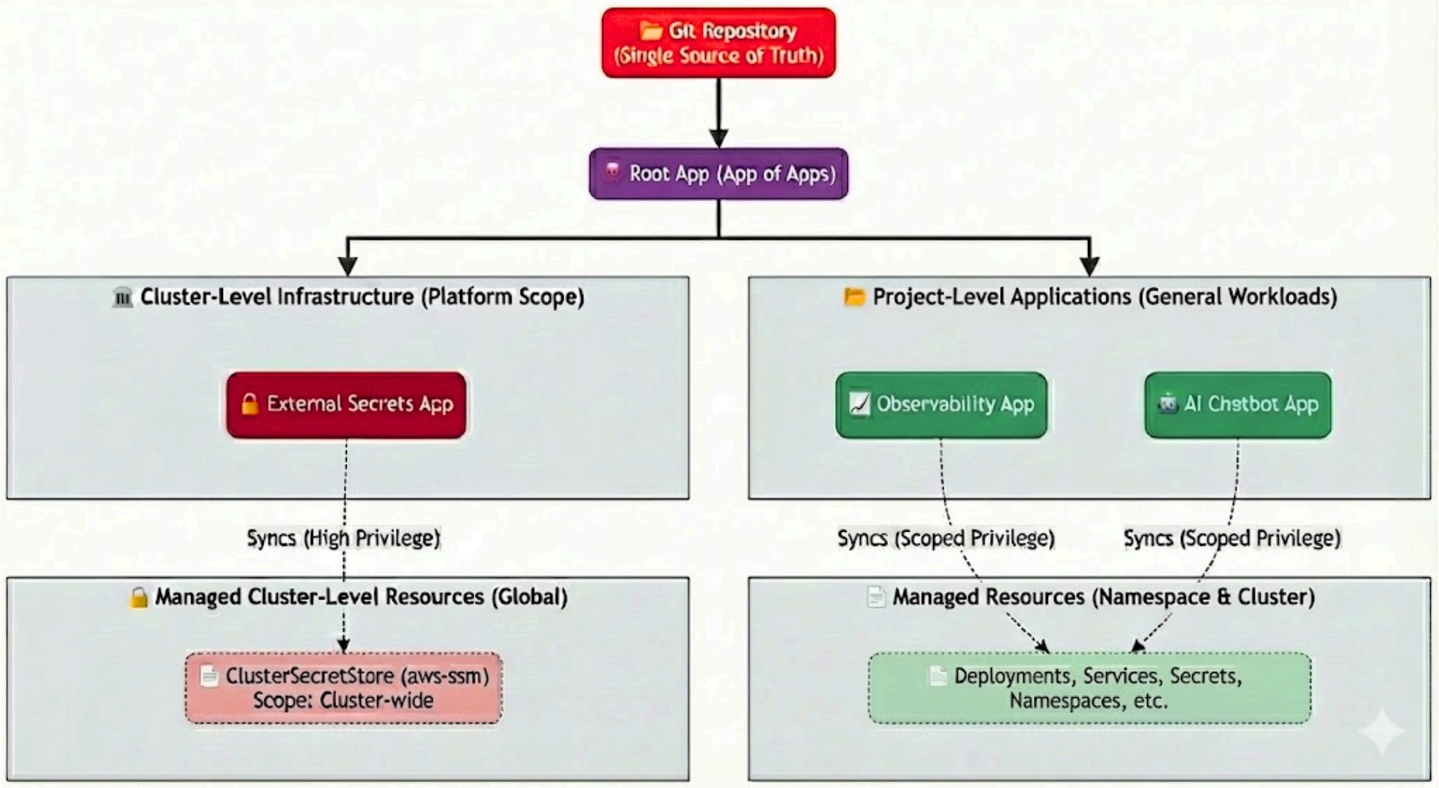
Build & Push

```
images:
  - name: 533267110761.dkr.ecr.ap-northeast-
    newTag: 5e37db8
```

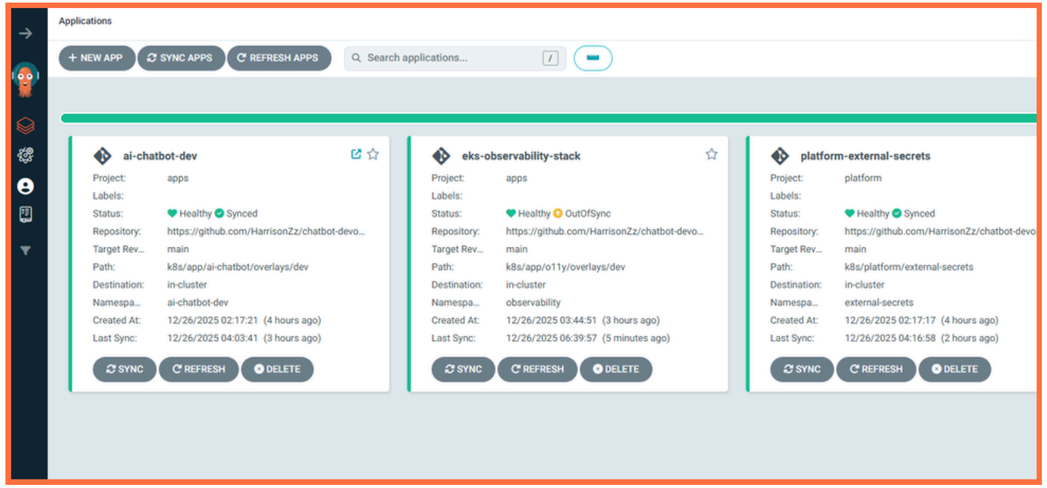
- **Automated Continuous Delivery:**
  - Triggered automatically on git push events.
  - Build & Tag: Builds Docker image and generates a unique hash.
  - Upload to ECR.
  - GitOps Integration: Automatically updates the kustomization.yaml with the newTag, triggering ArgoCD.

GitOps Architecture: Hierarchical Management

ArgoCD (CD)



1. **App-of-Apps Pattern:** Centralized management of all microservices and infrastructure components.
2. **Scope Isolation:** Strictly separates Cluster-Level resources (e.g., External Secrets) from Project-Level workloads.
3. **Security Governance:** Differentiates privileges—Admin access for Platform apps, scoped access for Business apps.



Security & Least Privilege

- **CI/CD Security: Keyless Authentication**

  GitHub Actions OIDC (OpenID Connect).

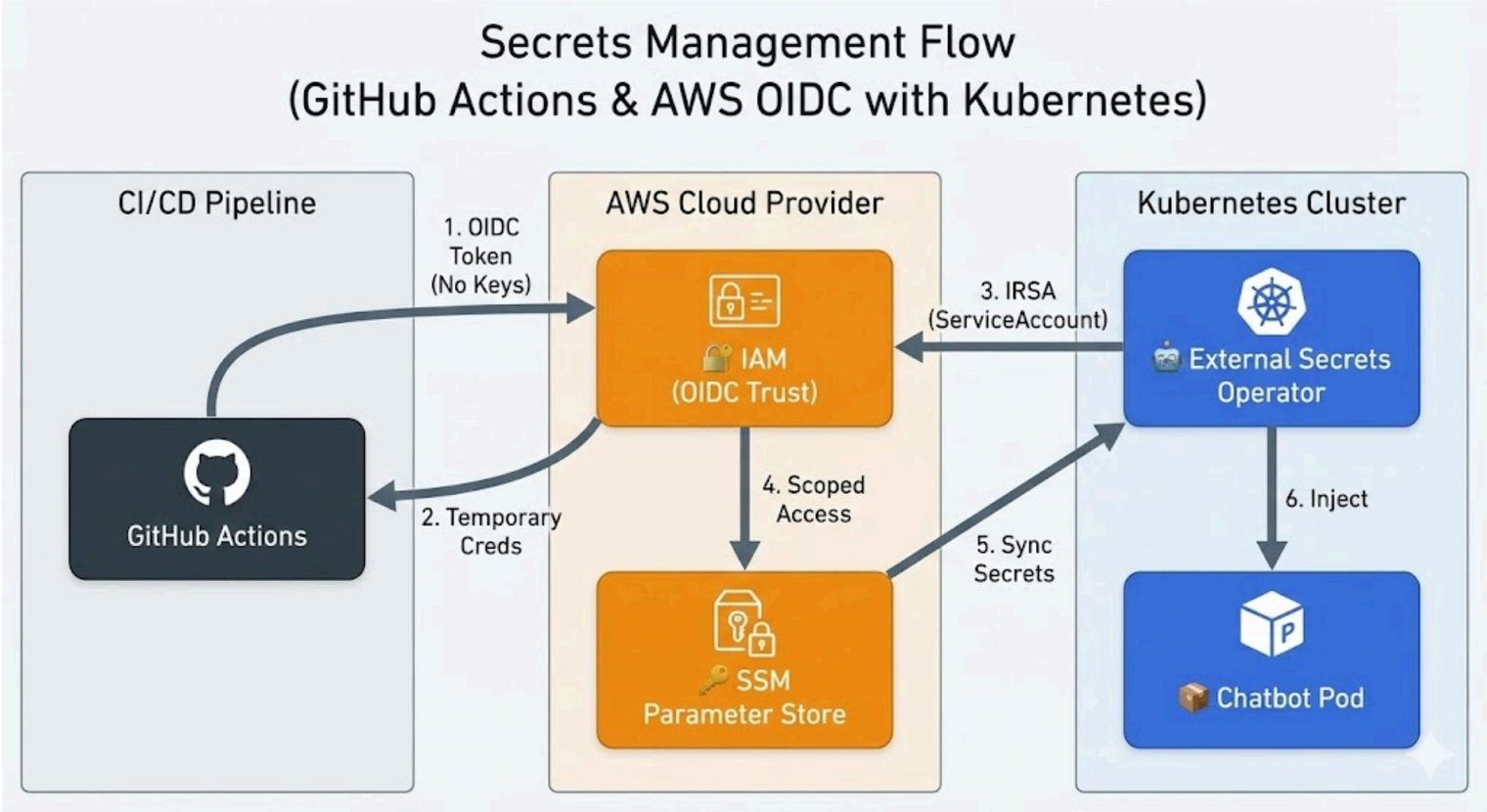
- **Secrets Management: The "Secret Zero" Solution**

  External Secrets Operator (ESO) + AWS SSM Parameter Store.

- **Workload Identity: IRSA + Pod Identity**

  IAM Roles for Service Accounts (IRSA).

IDENTITY & SECRET DATA FLOW




```yaml
permissions:
  id-token: write
  contents: write

- name: Configure AWS credentials (OIDC -> AssumeRole)
  uses: aws-actions/configure-aws-credentials@v4
  with:
    role-to-assume: ${{ env.ROLE_ARN }}
    aws-region: ${{ env.AWS_REGION }}
```

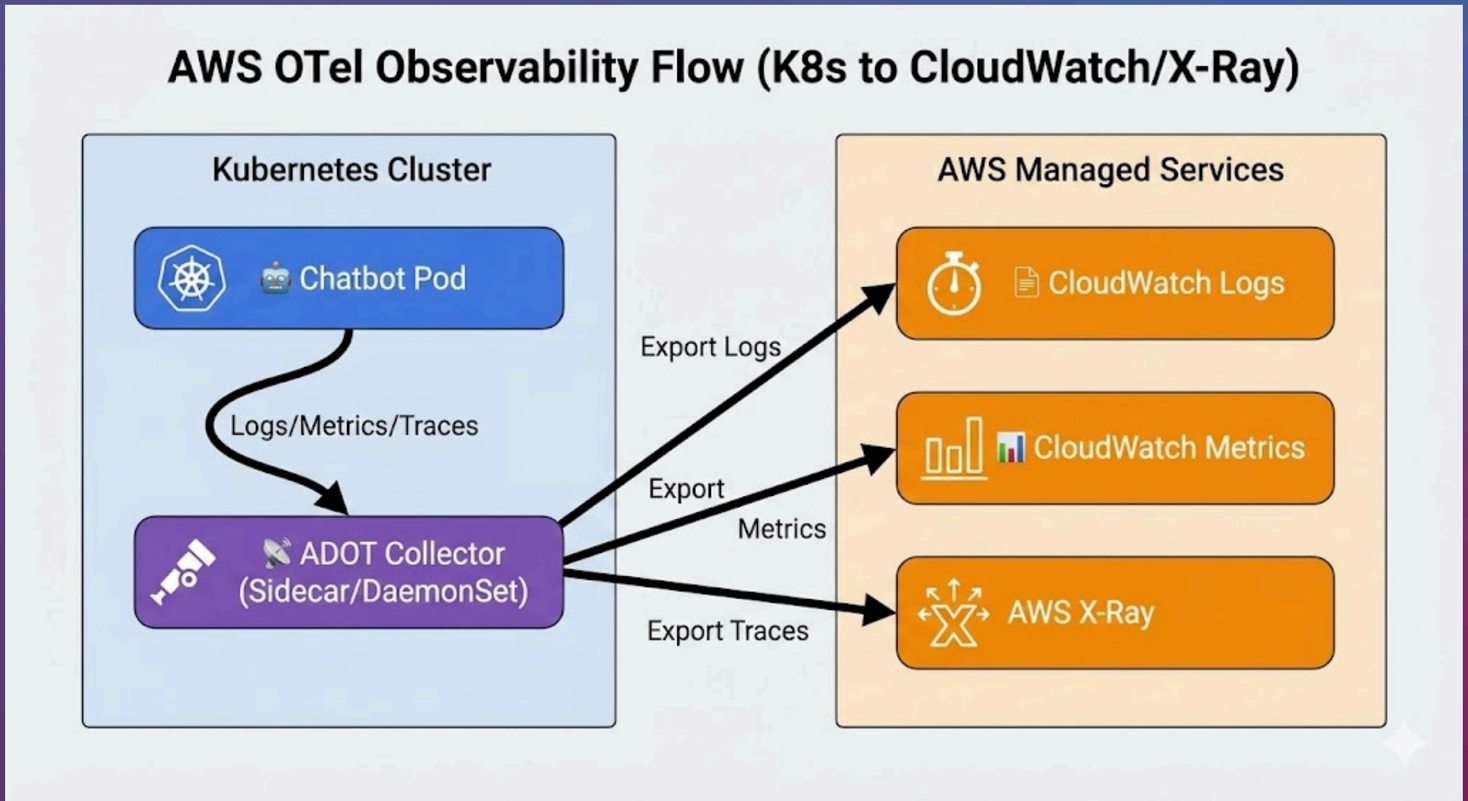
```yaml
spec:
  provider:
    aws:
      service: ParameterStore
      region: ap-northeast-1
      auth:
        jwt:
          serviceAccountRef:
            name: external-secrets-sa
            namespace: external-secrets
```

```python
obs_role_arn = self._create_irsa_role(
    role_name_part="adot-obs",
    namespace=namespace,
    sa_name=service_account,
    policy_json=obs_policy_json
)
```

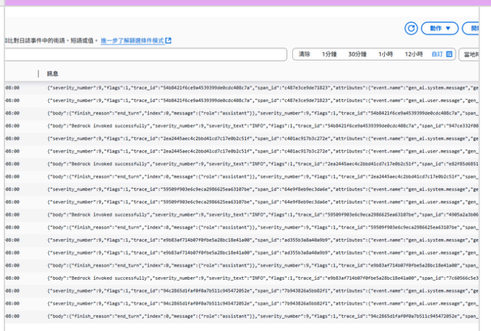
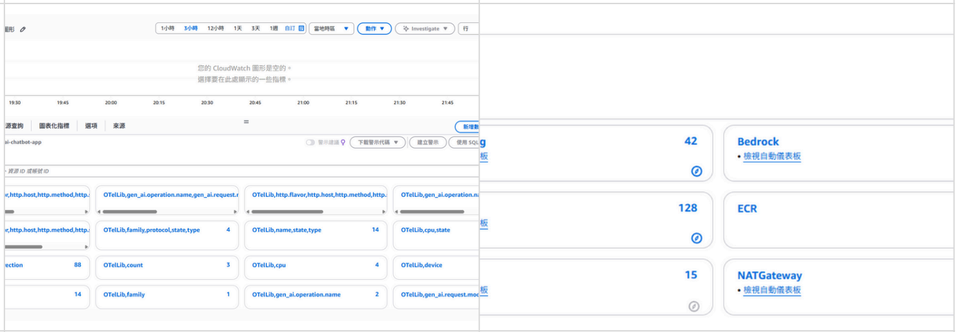
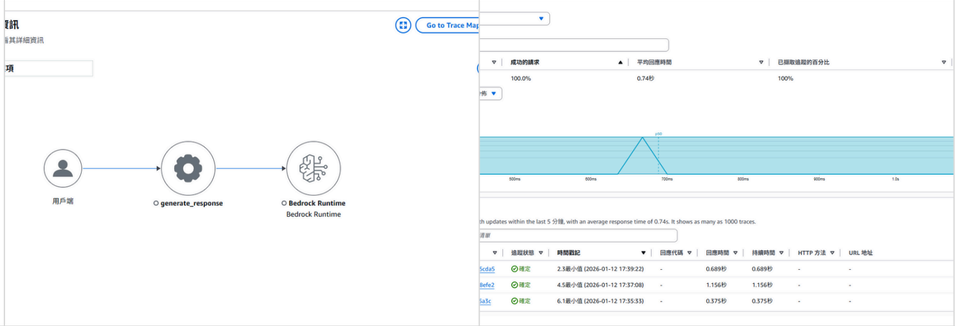
```python
pod_identity_assoc = aws.eks.PodIdentityAssociation(f"{
    cluster_name=self.cluster_name,
    namespace=namespace,
    service_account=service_account,
    role_arn=bedrock_role.arn,
    opts=ResourceOptions(parent=self)
)
```

Observability

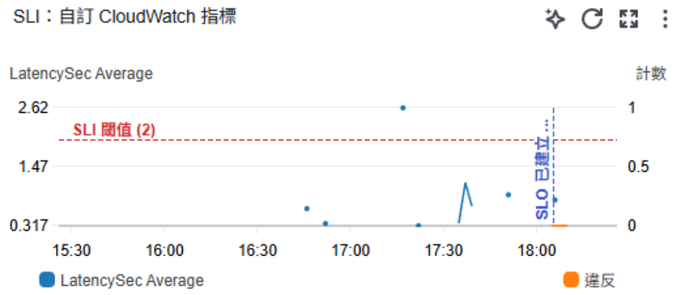
Full-Stack Visibility via OpenTelemetry



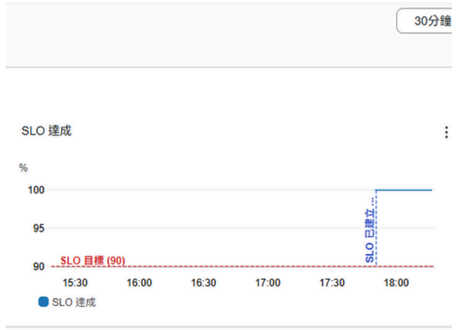
- **Standardization:**
  - Utilizes OpenTelemetry (OTel), the industry standard for observability.
- **Correlation:**
  - Unifies **Logs, Metrics, and Traces** (The Three Pillars) into a single collector.
  - Enables tracing a request from the **Ingress → Chatbot → Bedrock**.
- **Managed Service:**
  - Data is offloaded to **AWS CloudWatch & X-Ray**.
  - Zero maintenance overhead for storage or query engines (Serverless monitoring)

Pillar	Implementation	AWS Service	Demo
Logs	**ADOT Collector** (OTLP Receiver → CloudWatch Exporter)	**Amazon CloudWatch Logs**	
Metrics	**ADOT Collector** (OTLP Receiver → EMF Exporter)	**Amazon CloudWatch Metrics**	
Traces	**ADOT Collector** (OTLP Receiver → X-Ray Exporter)	**AWS X-Ray**	

SLI



SLO



- **Period: 2 hr**
- **Interval: 5 min**
- **SLO Goal: 90%**

延遲時間 < 2s

- **Period: 1 hr**
- **Interval: 5 min**
- **SLO Goal: 90%**

Error Budget

$$\frac{2 \times 60\ (\text{mins})}{5\ (\text{mins})} = 24$$

$$24 \times 0.1 = 2.4$$

$$\approx 2$$

$$\frac{60\ (\text{mins})}{5\ (\text{mins})} = 12$$

$$12 \times 0.1 = 1.2$$

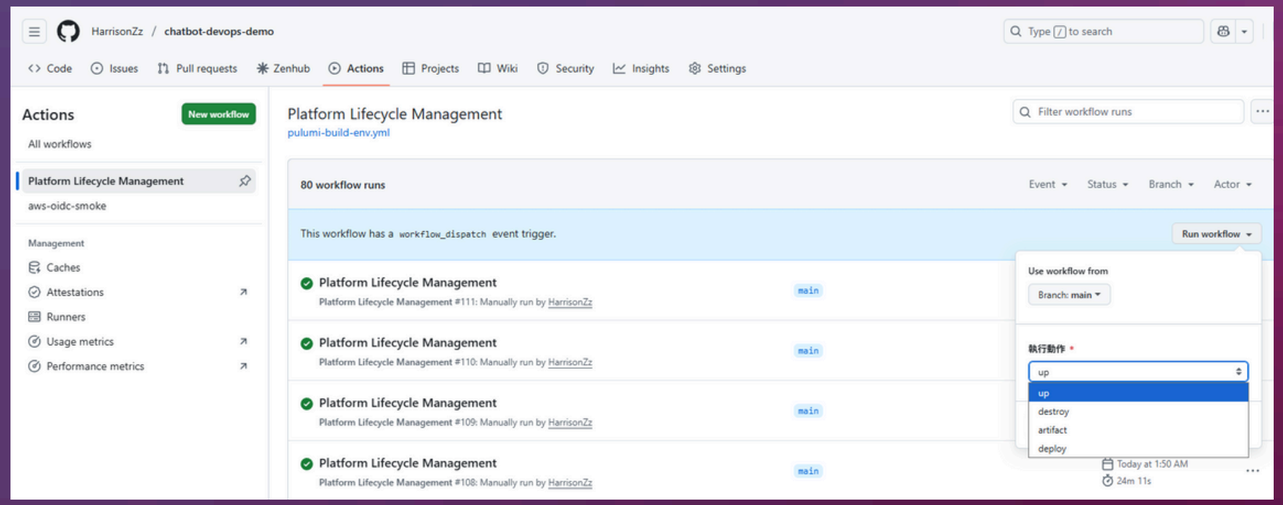
$$\approx 1$$

Live Demo & Conclusion

CLOUDFLARE

DNS Managed by Cloudflare

https://dev.hrscyj.uk



Q&A