Configuring policy enforcement in Cloud Deployer Pro

Overview

Cloud Deployer Pro includes a new policy engine for Advanced Multi-Cloud Orchestration. The policy engine enforces security and compliance rules directly in deployment pipelines across multiple clouds and connected on-premises environments.

The policy engine enables declarative, continuous enforcement of organizational rules during and after deployment. It blocks or warns on policy violations and flags configuration drift when changes occur outside the pipeline.

Advanced Multi-Cloud Orchestration with policy enforcement enabled consistently enforces policies across environments and reduces the risk of unauthorized or manual changes while supporting security and compliance requirements.

Policy enforcement follows a lifecycle of policy definition, policy integration, policy evaluation, enforcement, and drift detection. You define and integrate policies into deployment pipelines, configure access for policy evaluation, enforce policies during pipeline runs, and continuously monitor deployed resources for configuration drift.

Procedure: Define and integrate the policies

You can define and integrate policies into a pipeline by performing the following steps:

1. Open the *policy-manifest.yaml* file, and do the following:
    - Define policies in the YAML DSL format.
    - Define desired state for resources and configurations.
    - Define constraints and violations. For each violation, specify one of the following actions:
        - Deny the deployment.
        - Warn and log.
        - Auto-remediate.

        > **Note:** Auto-remediation is in beta stage for some resource types.

    - Add conditional blocks to apply policies based on your target environment or application type.
    - Define required resource tags, such as Owner, CostCenter, and Environment.
    - Save the *policy-manifest.yaml* file.
2. Open the *pipeline.yml* file.
3. Add a top-level `policies` block and reference the *policy-manifest.yaml* file.
4. Specify which policies apply at development, staging, and production stages.

   > **Note:** Override policy parameters per stage for environment-specific requirements or resource size limits.

5. Save the *pipeline.yml* file.

Procedure: Compose a policy set (optional)

Compose a policy set to reuse modular policies and manage complexity by performing the following steps:

1. Create modular policy files, such as *security-policies.yaml*, *compliance-policies.yaml*, and *cost-policies.yaml*.
2. Define the relevant policies in each modular policy file.
3. Create a *policy-set.yaml* file.
4. In the *policy-set.yaml* file, reference all the modular policy files.
5. Open the *pipeline.yml* file.
6. Reference the *policy-set.yaml* file instead of the *policy-manifest.yaml* file.

Creating separate modular policy files helps manage complexity and makes policies reusable across different deployments.

After defining the policies, the enforcement should work across various cloud platforms. To enforce policies, you must configure the Policy Engine IAM role to provide read-only access to the policy engine. The Policy Engine IAM role must be configured before deployment, separately from other deployment IAM roles.

Procedure: Evaluate and enforce policies

Configure the Policy Engine IAM role and evaluate and enforce policies by performing the following steps:

1. Open **Settings**, navigate to **Cloud Integrations**, and locate the **Policy Engine IAM Role** section.
2. Configure read-only access for the policy engine.

3. Configure cross-account and cross-cloud role assumption.
4. Open a terminal and do the following:
    - Run `cdp policy validate` to validate policy definitions and evaluate them without running a deployment.
    - Run `cdp pipeline run --policy-check` to run the pipeline and evaluate and enforce policies during deployment.
    - Run `cdp policy enforce` to enforce policies without running the deployment pipeline.

Drift detection continuously monitors deployed resources against the desired state defined in the *policy-manifest.yaml* file and the pipeline's declared configuration. If resources change outside the pipeline, such as manual modifications, the policy engine flags the changes as drift and generates alerts. Alerts can be delivered through webhooks or SNS or SQS integrations. Drift is typically resolved manually or by re-running the pipeline.

Procedure: Detect configuration drift

Detect configuration drift after a successful deployment by performing the following steps:

1. Open a terminal.
2. Run `cdp policy detect-drift --pipeline <id>` to compare deployed resources against the desired state and the pipeline configuration.
3. Review reported drift events when resources change outside the pipeline.
4. Configure alerts through webhooks, SNS, or SQS integrations.
5. Resolve drift events manually or by re-running the pipeline.