The Ultimate DevSecOps Framework: A Cross-Cloud Implementation Guide for Azure, AWS, and GCP

1. Introduction & Executive Summary

- **What is DevSecOps**: An operating model that embeds security into every phase of the SDLC (plan, develop, build/test, deploy, operate) via automation, policy as code, and shared accountability.
- **Core principles**
  - **Shift Left**: Threat modeling, secure design, and code scanning early.
  - **Continuous Security**: Automated checks across CI/CD and runtime.
  - **Automation**: Security as code; reproducible, testable controls.
  - **Collaboration**: Product, Dev, Sec, Ops, and Compliance aligned on risk and outcomes.
- **Benefits**
  - **Business**: Faster releases, fewer incidents, audit readiness, lower risk.
  - **Technical**: Repeatable hardened builds, deterministic infra, least privilege, continuous visibility.
- **Shared Responsibility Model**
  - Cloud providers secure the cloud; customers secure what they deploy in the cloud.
  - DevSecOps operationalizes customer responsibilities through automation and guardrails across identities, data, networks, apps, and workloads.

2. The DevSecOps Lifecycle: A Phase-by-Phase Breakdown

Phase 1: Plan & Design (Threat Modeling & Policy as Code)

Goal: Identify security requirements and threats before code is written.

Concept	Azure Service/Tool	AWS Service/Tool	GCP Service/Tool	Usage Instructions/Integ Steps
Threat modeling	Microsoft Threat Modeling Tool, STRIDE	AWS Well-Architected Tool (Security Pillar), Threat Composer	Google Cloud Architecture Framework (Security), Threat Modeling with STRIDE/LINDDUN	Run workshops pe capture mitigation user stories; valida against security architecture patte
Policy as Code (governance)	Azure Policy, Azure Landing Zones (ALZ), Microsoft Defender for Cloud recommendations	AWS Organizations SCPs, AWS Config, Service Control Policies, Security Hub controls	Organization Policy Service, Policy Controller (OPA/Gatekeeper), Assured Workloads (where applicable)	Encode guardrails centrally; enforce prod/prod baseline block drift via den policies; test polic with Conftest.
IaC scanning (pre-commit/CI)	Defender for DevOps (IaC), GitHub Advanced Security for IaC,	cfn-nag, CFN Guard, Checkov/Terrascan in CodeBuild, IAM Access Analyzer	Terraform Validator/OPA (Config Validator), Checkov/Terrascan in Cloud Build	Add IaC scanners commit and CI; fai builds on critical misconfigurations;

	Checkov/Terrascan in Azure Pipelines			generate SARIF for review.
Identity architecture	Microsoft Entra ID, PIM, Managed Identity	AWS IAM, IAM Identity Center, IAM Access Analyzer	Cloud IAM, Workload Identity Federation	Establish least-pri roles, role chaining break-glass; defin privilege elevation workflow and just-time access.
Secrets design	Azure Key Vault (HSM-backed), Managed Identity	AWS Secrets Manager, Parameter Store, KMS	Secret Manager, CMEK/KMS	Decide secret sou standardize SDKs; forbid inline secret rotate automatical
Data protection	Azure Information Protection, Disk Encryption, Purview (governance)	KMS envelope encryption, Macie (discovery)	KMS/CMEK, DLP API, Data Catalog	Classify data, set encryption requirements, defi rotation and dual-control procedure

The Ultimate DevSecOps Framework: A Cross-Cloud Implementation Guide for Azure, AWS, and GCP (contd.)

Phase 2: Develop (SAST, SCA, Pre-commit Hooks)

Goal: Find and fix vulnerabilities in the code during development.

Concept	Azure Service/Tool	AWS Service/Tool	GCP Service/Tool	Usage Instructions
SAST	GitHub Advanced Security (CodeQL), SonarQube on Azure	Amazon CodeGuru (code quality), CodeWhisperer security scans, SonarQube on EC2	Cloud Build triggers with SAST runners, SonarQube on GCE	Enable SAST on PRs with SARIF reporting; enforce severity thresholds; auto-create fix tickets.
SCA/Dependency scanning	GitHub Dependabot, Defender for DevOps (OSS), Snyk	AWS CodeArtifact + SCA (Snyk/Mend), Amazon Inspector SBOM ingestion	Artifact Registry + On-Demand Scanning, Snyk	Generate SBOM (CycloneDX/Syft); break builds on critical CVEs; pin versions; cache allowlist.
Secrets detection	GitHub secret scanning, Gitleaks in Azure Pipelines	git-secrets, TruffleHog in CodeBuild/CodePipeline	Secret detection via Gitleaks in Cloud Build	Add pre-commit hooks and CI jobs; revoke on detection; rotate automatically.

Pre-commit quality gates	Pre-commit framework, ESLint/flake8, Commit signing (Sigstore keyless)	Pre-commit, commit signing with GPG/Sigstore	Pre-commit, Sigstore cosign keyless for commits	Mandate passing hooks; enforce DCO/signoff; verify provenance.

Phase 3: Build & Test (CI/CD Security, DAST, Container Scanning)

Goal: Automate security checks within the CI/CD pipeline.

Concept	Azure Service/Tool	AWS Service/Tool	GCP Service/Tool	Usage Instructions
CI/CD hardening	Azure Pipelines protected environments, service connections with least privilege	CodeBuild/CodePipeline with scoped IAM roles, OIDC to GitHub	Cloud Build with Worker Pools, restricted service accounts	Use OIDC short-lived creds; isolate runners; sign artifacts; enforce required reviews.
Container image scanning	ACR + Defender for Cloud, Trivy task	ECR + Amazon Inspector, Trivy in CodeBuild	Artifact Registry + Container Analysis (on-push/on-demand), Trivy in Cloud Build	Scan on build and on-push; fail on criticals; export SARIF; attach attestations.
DAST	OWASP ZAP in Azure Pipelines	OWASP ZAP container in CodeBuild/CodePipeline	ZAP in Cloud Build/Cloud Run job	Spin ephemeral test env; run auth'd ZAP scans; publish HTML/SARIF; gate on risk budget.
Supply chain integrity	Azure Attestation, Notary v2 (ORAS), cosign signatures	ECR with image signing (cosign), SLSA provenance in CodeBuild	Binary Authorization, cosign, SLSA provenance with Cloud Build	Sign images and SBOMs; enforce Binary Auth (GCP) or admission policies (OPA).
Test data protection	Azure Dev/Test Labs, synthetic data pipelines	AWS DMS/Glue masking, synthetic datasets	DLP + masked test datasets	Automate data masking; forbid prod data in test; monitor policy violations.

Phase 4: Deploy (Infrastructure Security, Secrets Management)

Goal: Securely deploy infrastructure and applications with managed secrets.

Concept	Azure Service/Tool	AWS Service/Tool	GCP Service/Tool	Usage Instructions
IaC deployment	ARM/Bicep, Terraform with AzureRM	CloudFormation/CDK, Terraform with AWS	Terraform with Google provider, Config Controller/Config Connector (KRM)	Use CI to plan/apply with approvals; store state securely; drift detection via policy.
Secrets at deploy/runtime	Azure Key Vault + Managed Identity	Secrets Manager/Parameter Store + IAM roles	Secret Manager + Workload Identity	Inject secrets vi native SDKs or CSI; disallow env var persistence; rotate and test.
Network/security perimeters	Azure Firewall, Private Link, NSGs, WAF on Front Door/AppGW	VPC, PrivateLink, Security Groups, AWS WAF	VPC Service Controls, Private Service Connect, Cloud Armor	Enforce private build→registry paths; restrict egress; apply WAF managed rules.
Admission/guardrails	Azure Policy for AKS, Defender for Containers	OPA/Gatekeeper on EKS, Inspector ECS/EKS	Policy Controller, Binary Authorization	Block non-signed images; enforce PSP/PSS; ensure rootless/readonl FS.

Phase 5: Operate & Monitor (CSPM, CWPP, SIEM, Incident Response)

Goal: Continuously monitor and protect running workloads.

Concept	Azure Service/Tool	AWS Service/Tool	GCP Service/Tool	Usage Instruction
CSPM posture	Microsoft Defender for Cloud	AWS Security Hub + Config	Security Command Center (SCC)	Enable org-w map to CIS/N set remediati workflows.
CWPP workload protection	Defender for Servers/Containers	Amazon Inspector (EC2/ECR/ECS/EKS), GuardDuty EKS Runtime	Container Threat Detection, VM Threat Detection	Deploy agents/agent coverage; tria criticals; auto quarantine options.

Threat detection	Microsoft Sentinel + Defender for Cloud Apps	GuardDuty, Detective, Security Lake + Athena/QuickSight	Chronicle SIEM, Event Threat Detection	Ingest logs at scale; detecti as code; threa intel feeds; response playbooks.
Logging/telemetry	Azure Monitor, Log Analytics, Application Insights	CloudWatch, VPC Flow Logs, CloudTrail	Cloud Logging, Cloud Monitoring, VPC Flow Logs, Audit Logs	Set retention data class; in exclusion for cost; field-lev parsing/labels
Incident response	Azure Automation/Functions, Sentinel SOAR	Systems Manager, Step Functions, Lambda SOAR	Cloud Functions/Run + Chronicle SOAR	Runbooks wit least privilege simulate regularly; evidence preservation (forensics).
Vulnerability mgmt	Defender for Cloud assessments	Inspector/SSM Patch Manager	OS patch management, OS Config	Patch SLAs; maintenance windows; exception process.

3. Cross-Cloud Use Case Studies

Use Case A: Securing a Serverless Application (API)

- Architecture goals: authenticated API, least-privilege execution, signed artifacts, private networking, centralized logging.
- Azure
    - API: Azure API Management (APIM) + Azure Functions (Premium/Isolated), Azure Front Door + WAF.
    - AuthN/Z: Entra ID (OAuth2/JWT validation at APIM), Managed Identity for Functions.
    - Secrets: Azure Key Vault with Key Vault References or SDK.
    - Build/Deploy: Azure Pipelines/GitHub Actions, Bicep for infra, Functions deploy with slots.
    - Security: Defender for Cloud (recommendations), Defender for App Service/Functions, App Insights for tracing.
    - Steps:
        1. Create APIM policy to validate JWT and rate-limit.
        2. Build Functions with SAST/SCA; sign artifact (cosign) and publish to ACR (if containerized).
        3. Provision with Bicep; assign system-assigned managed identity with least privilege.
        4. Configure Key Vault access policies/role assignments; rotate secrets.

    5. Enable Defender plans; stream logs to Log Analytics and Sentinel; add detection rules.

- AWS
    - API: Amazon API Gateway + AWS Lambda, AWS WAF (managed rules).
    - AuthN/Z: Amazon Cognito JWT validation; Lambda execution role with least-privilege policies.
    - Secrets: AWS Secrets Manager or SSM Parameter Store.
    - Build/Deploy: CodePipeline/CodeBuild or GitHub Actions OIDC; SAM/Serverless Framework; artifact signing with cosign.
    - Security: GuardDuty, CloudTrail, Security Hub; Inspector for Lambda code scanning via SBOM ingestion.
    - Steps:
        1. Define SAM template with API Gateway + Lambda + WAF.
        2. Enable OIDC for pipeline; run SAST/SCA; sign image if using container Lambda.
        3. Attach minimal IAM role to Lambda; VPC integration if accessing private resources.
        4. Fetch secrets at runtime; enable rotation lambda for Secrets Manager.
        5. Centralize logs in CloudWatch; enable Security Hub; add EventBridge rules for alerts.

- GCP
    - API: API Gateway or Cloud Endpoints + Cloud Functions/Cloud Run; Cloud Armor for WAF (when using HTTPS Load Balancer).
    - AuthN/Z: Cloud IAP (for HTTPS LB) or JWT validation at API Gateway; Workload Identity for service-to-service.
    - Secrets: Secret Manager with automatic rotation (via Cloud Functions/Run).
    - Build/Deploy: Cloud Build, Artifact Registry, Binary Authorization (Cloud Run supports attestations).
    - Security: SCC Premium, Event Threat Detection, Cloud Audit Logs; Container/On-Demand Scanning.
    - Steps:
        1. Provision API Gateway + Cloud Run with Terraform; enable JWT auth.
        2. Build with Cloud Build; run SAST/SCA; generate SBOM; sign image (cosign).
        3. Enforce Binary Authorization policy or deploy attested images.
        4. Grant minimal IAM to service account; access secrets via Secret Manager API/CSI.
        5. Route logs to Cloud Logging; create Chronicle/SCC detections and alerting.

Use Case B: Securing a Containerized Application (Kubernetes)

- Common controls: image scanning, signature enforcement, admission control (OPA), network policies, least-privilege, secrets CSI, RBAC, audit logs.

- Azure (AKS)
    - Build to ACR with Trivy/Defender; sign with cosign.
    - Enforce Azure Policy for AKS (deny privileged, require signed images).
    - Use Managed Identity/Workload Identity for pods; CSI Secrets Store for Key Vault.
    - Enable Defender for Containers (agent/agentless), Azure Monitor for containers.
    - Private cluster, Azure Firewall, egress lockdown; Calico/Cilium network policies.

- AWS (EKS)

    - Build to ECR; scan via Amazon Inspector.
    - Gatekeeper (OPA) for Pod Security, image provenance; IRSA for pod IAM.
    - Secrets via Secrets Manager CSI driver; encrypt etcd with KMS.
    - GuardDuty EKS Runtime, CloudWatch Container Insights; VPC CNI with restricted egress.
- GCP (GKE)
    - Build to Artifact Registry; Container Analysis scanning; sign with cosign.
    - Binary Authorization to enforce signatures/attestations.
    - Workload Identity for GSA↔KSA; Secret Manager CSI driver.
    - GKE Dataplane V2 network policies; Container Threat Detection; SCC findings.
- Side-by-side steps
    1. CI: run SAST/SCA; build image; SBOM (Syft); scan (Trivy/registry); cosign sign.
    2. Configure admission: Azure Policy for AKS / OPA on EKS / Binary Authorization on GKE.
    3. Deploy manifests with `imagePullPolicy: Always`, drop capabilities, runAsNonRoot, readOnlyRootFilesystem.
    4. Apply NetworkPolicies; restrict egress; enforce internal registries via imagePolicyWebhook/BinAuthz.
    5. Enable runtime sensors and send logs to SIEM; create automated quarantine playbooks.

Use Case C: Securing a Virtual Machine-Based Workload

- Azure
    - Images: Azure Marketplace CIS images or Packer; Azure Compute Gallery.
    - Patching: Azure Update Manager; Defender for Servers.
    - EDR: Microsoft Defender for Endpoint; AMA/OmsAgent for logs.
    - Encryption: Azure Disk Encryption with platform-managed or customer-managed keys.
    - Access: Just-in-time VM Access, Bastion; disable SSH password; use AAD login.
    - IR: Snapshot policies, Azure Automation runbooks, Sentinel forensics workbooks.
- AWS
    - Images: EC2 Image Builder, Marketplace CIS AMIs.
    - Patching: Systems Manager Patch Manager; Inspector for vulnerabilities.
    - EDR: GuardDuty Malware Protection; integrate CrowdStrike/Defender if needed.
    - Encryption: EBS with KMS; enforce via SCP/Config rules.
    - Access: SSM Session Manager (no SSH), least-privilege instance profiles, IAM Access Analyzer.
    - IR: Snapshot volumes; isolate in quarantine SG; Forensics account via AWS Organizations.
- GCP
    - Images: Shielded VM images, OS Config for compliance; Packer via Compute Engine.
    - Patching: OS Patch Management; VM Manager; vulnerability reports.
    - EDR: Chronicle integrations, third-party agents; VM Threat Detection.
    - Encryption: CMEK per disk; uniform bucket-level access for artifacts.
    - Access: OS Login/2FA, IAP for SSH/RDP (no public IP); least-privilege service accounts.
    - IR: Create disk clones; block egress with firewall; analyze with Compute Engine forensics VM.

4. Example Software Tools & Integration Catalog

- SAST
  - Snyk Code: fast PR feedback; SARIF output; IDE integrations.
  - Checkmarx: deep coverage for enterprise; policy-driven gates.
  - SonarQube: code quality + security hotspots; branch analysis.
  - Example (CI step):

    ```
    snyk code test --severity-threshold=high --sarif > snyk-code.sarif
    ```

- SCA / Dependency
  - Snyk Open Source: tests manifests and transitive deps.
  - Mend (WhiteSource): license compliance + risk scoring.
  - Dependabot/Renovate: automated PRs for updates.
  - Example:

    ```
    snyk test --all-projects --severity-threshold=high --sarif > snyk-deps.sarif
    ```

- IaC Scanning
  - Checkov, tfsec (in Trivy), Terrascan, CFN Guard, cfn-nag.
  - Example:

    ```
    checkov -d infra/ --framework terraform,cloudformation,kubernetes --quiet --soft-fail=false
    ```

- Container Scanning
  - Trivy, Grype/Anchore, Clair.
  - Example:

    ```
    trivy image --scanners vuln,secret,misconfig --exit-code 1 --severity CRITICAL,HIGH $IMAGE_REF
    ```

- DAST
  - OWASP ZAP, Burp Suite Enterprise.
  - Example (ZAP baseline):

    ```
    zap-baseline.py -t https://staging.example.com -r zap.html -J zap.sarif -m 5
    ```

- Secrets Detection
  - Gitleaks, TruffleHog, git-secrets.
  - Example:

```
gitleaks detect --no-banner --report-format sarif --report-path
gitleaks.sarif
```

- SBOM and Provenance
    - Syft (SBOM), CycloneDX, in-toto attestations, SLSA framework, cosign.
    - Example:

    ```
    syft packages dir:. -o cyclonedx-json > sbom.json
    cosign sign --keyless $IMAGE_REF
    cosign attest --predicate sbom.json --type cyclonedx $IMAGE_REF
    ```

- Policy as Code
    - OPA/Gatekeeper, Conftest, Kyverno (K8s).
    - Example:

    ```
    conftest test infra/ --policy policy/
    ```

- CI/CD Platforms
    - GitHub Actions, Azure Pipelines, GitLab CI, Jenkins, CircleCI.
    - Generic pipeline snippet (build, scan, sign, push):

    ```
    docker build -t $IMAGE_REF .
    syft dir:. -o cyclonedx-json > sbom.json
    trivy image --exit-code 1 --severity CRITICAL,HIGH $IMAGE_REF
    docker push $IMAGE_REF
    cosign sign --keyless $IMAGE_REF
    cosign attest --predicate sbom.json --type cyclonedx $IMAGE_REF
    ```

- CSPM/CNAPP
    - Wiz, Lacework, Prisma Cloud, Orca: unified posture + workload/runtime.
    - Integration: org-wide read roles; connect to cloud APIs; CI controls for shift-left checks; unified alerting to SIEM/SOAR.

5. Conclusion & Best Practices Summary

- Key takeaways
    - Treat security as code: policies, controls, and tests versioned and automated.
    - Enforce least privilege and short-lived credentials via OIDC and workload identities.
    - Secure the software supply chain: SBOMs, signatures, provenance, and admission enforcement.
    - Continuously validate posture and runtime with CSPM/CWPP and robust telemetry.
- Multi-Cloud Getting Started Checklist
    - Define threat model and risk acceptance criteria per product.

    - Establish org-wide guardrails: Azure Policy, AWS SCP/Config, GCP Org Policy/Policy Controller.
    - Standardize IaC (Terraform/Bicep/CDK/KRM) and add IaC scanning in pre-commit and CI.
    - Enable SAST, SCA, secret scanning on PRs with severity gates and SARIF reporting.
    - Implement CI hardening: isolated runners, OIDC to cloud, artifact signing, SBOMs.
    - Choose registry and enable native scanning: ACR + Defender, ECR + Inspector, Artifact Registry + Container Analysis.
    - Enforce runtime policies: AKS Policy, OPA/Gatekeeper for EKS, Binary Authorization for GKE.
    - Centralize logging and detections in Sentinel, Security Lake + analytics, or Chronicle.
    - Automate patching and vulnerability management for VMs and containers.
    - Drill incident response with SOAR runbooks; keep forensics playbooks tested.
    - Implement secrets management everywhere; remove plaintext secrets from repos and pipelines.
- Common pitfalls to avoid
    - Relying solely on cloud defaults without organization guardrails.
    - Allowing broad, long-lived credentials or static keys in CI.
    - Skipping provenance/signing leading to untrusted artifacts in prod.
    - Inconsistent environments across clouds; drift between policy and runtime.
    - Alert fatigue without ownership, triage workflows, and auto-remediation.
    - Using legacy/unsupported services (e.g., avoid new deployments on GCP Deployment Manager; prefer Terraform/KRM).

Visual Workflows (Mermaid Diagrams)

Phase 1: Plan & Design – Threat Modeling & Policy as Code

```
flowchart TD
  A[Business Objectives] --> B[Threat Modeling \n STRIDE/LINDDUN]
  B --> C[Security Requirements \n User Stories/Acceptance Criteria]
  C --> D[Policy as Code \n Azure Policy / AWS SCP+Config / GCP Org Policy]
  D --> E[IaC Baselines \n Terraform/Bicep/CDK/KRM]
  E --> F[Pre-commit & CI IaC Scanning \n Checkov/Terrascan/OPA]
  F --> G[Design Review Gate]
```

Phase 2: Develop – SAST, SCA, Secrets & Hooks

```
flowchart TD
  A[Developer IDE] --> B[Pre-commit Hooks \n lint/tests/secret-scan]
  B --> C[Branch Push / PR]
  C --> D[SAST (CodeQL/SonarQube)]
  C --> E[SCA & SBOM (Snyk/Mend/Syft)]
  C --> F[Secrets Scan (Gitleaks/TruffleHog)]
  D & E & F --> G[PR Checks & Severity Gates]
  G -->|Pass| H[Approve & Merge]
  G -->|Fail| I[Fix Findings]
```

Phase 3: Build & Test – CI/CD Security, DAST, Container Scanning

```
flowchart TD
  A[Source Merge] --> B[Build Container/Artifact]
  B --> C[SBOM Generate (Syft/CycloneDX)]
  B --> D[Image Scan (Trivy/Registry Scan)]
  B --> E[Sign & Attest (cosign/SLSA)]
  D --> F{Vuln Policy}
  F -->|Pass| G[Ephemeral Env Deploy]
  F -->|Fail| H[Block & Create Issue]
  G --> I[DAST (ZAP)]
  I --> J{Risk Budget Gate}
  J -->|Pass| K[Promote to Deploy Stage]
  J -->|Fail| H
```

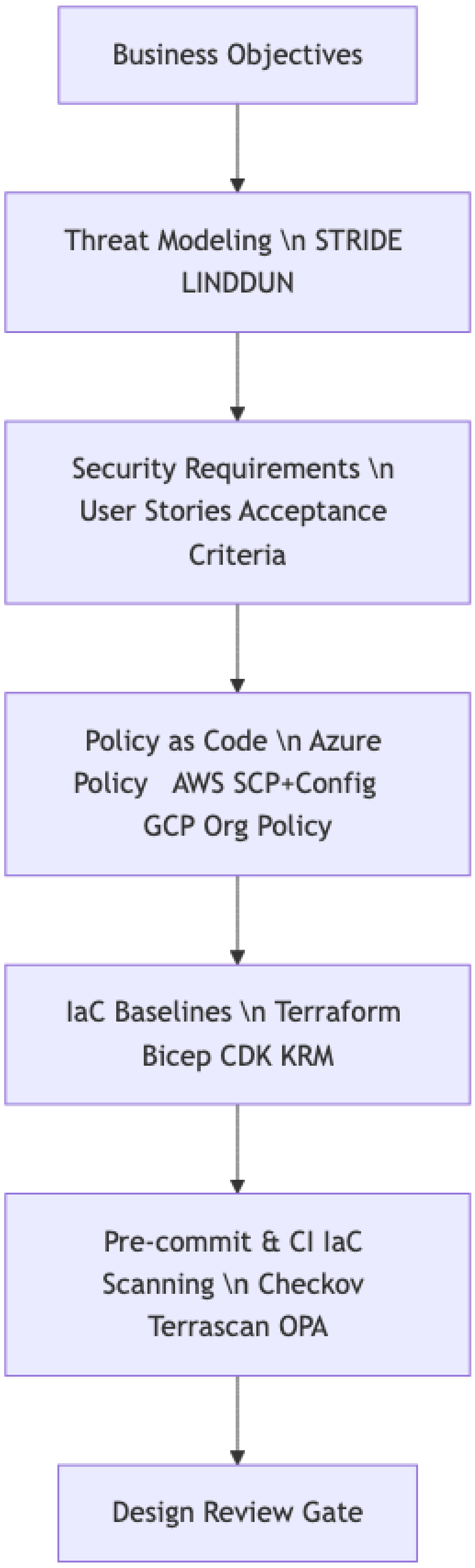
Phase 4: Deploy – Infra Security & Secrets Management

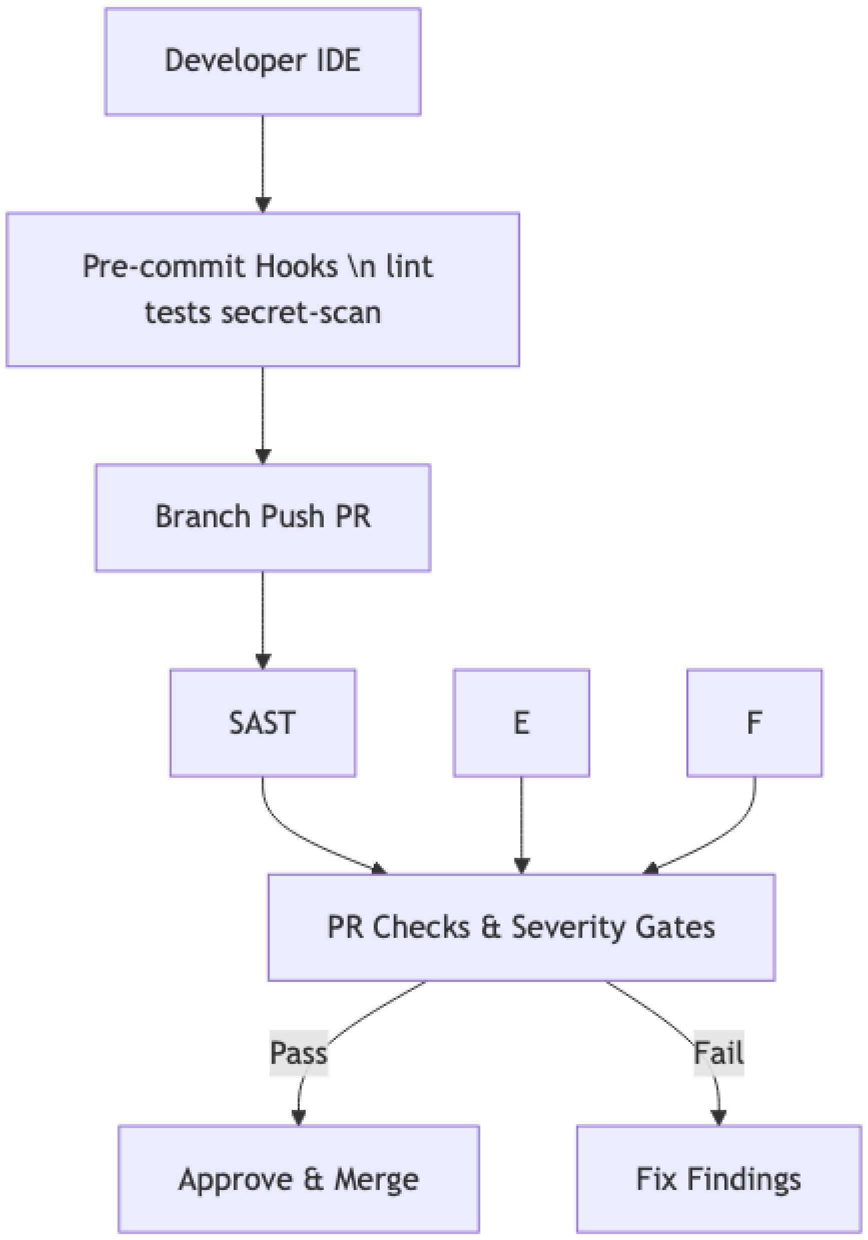
```
flowchart TD
  A[Release Approval] --> B[IaC Apply \n Terraform/Bicep/CDK/KRM]
  B --> C[Guardrails \n Azure Policy / OPA / BinAuthz]
  C --> D[Runtime Identities \n Managed Identity / IRSA / Workload Identity]
  D --> E[Secrets via Vaults \n Key Vault / Secrets Manager / Secret Manager]
  E --> F[Network Controls \n WAF, Private Links, Firewall, VPC SC]
  F --> G[Deploy to Target Env]
```

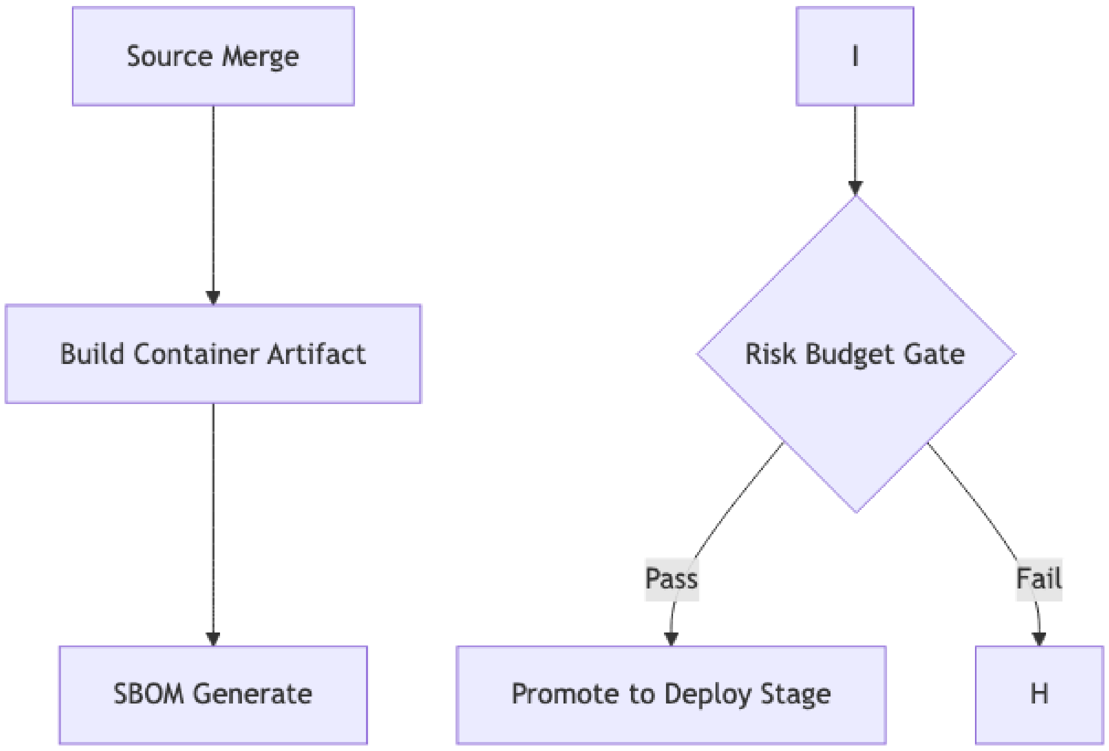
Phase 5: Operate & Monitor – CSPM, CWPP, SIEM, IR

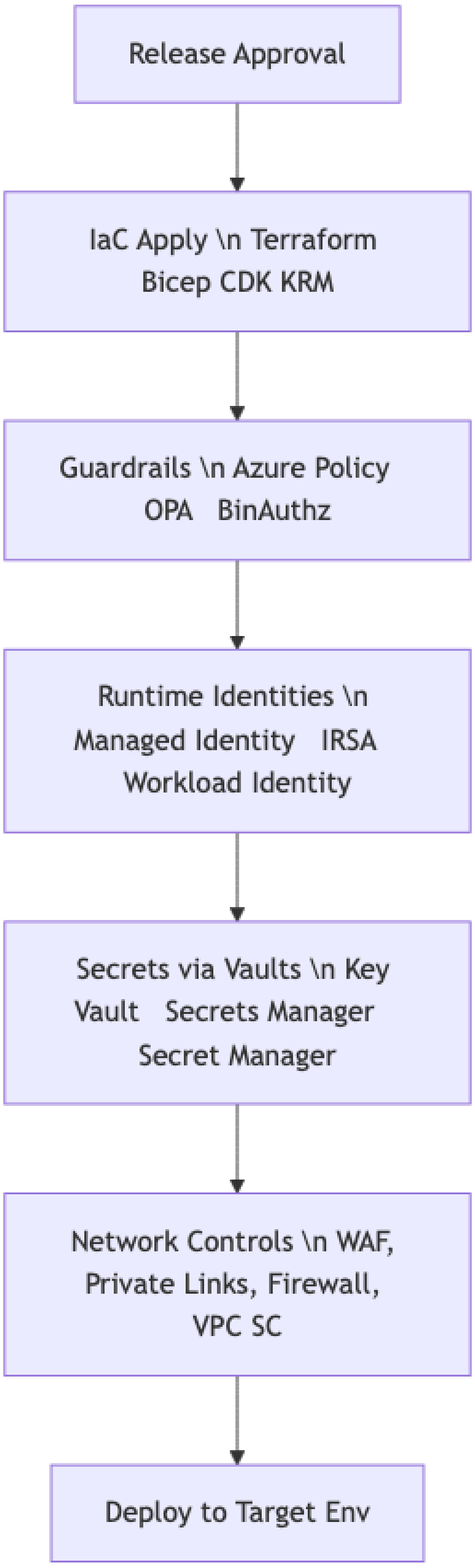
```
flowchart TD
  A[Workloads Running] --> B[CSPM Posture \n Defender for Cloud / Sec Hub / SCC]
  A --> C[CWPP Runtime \n Defender / Inspector / CTD]
  A --> D[Telemetry \n Monitor/CloudWatch/Cloud Logging]
  D --> E[SIEM \n Sentinel / Security Lake / Chronicle]
  E --> F[Detections-as-Code + SOAR]
  F --> G[Containment & Eradication]
  G --> H[Lessons Learned & Policy Update]
  H --> B
```

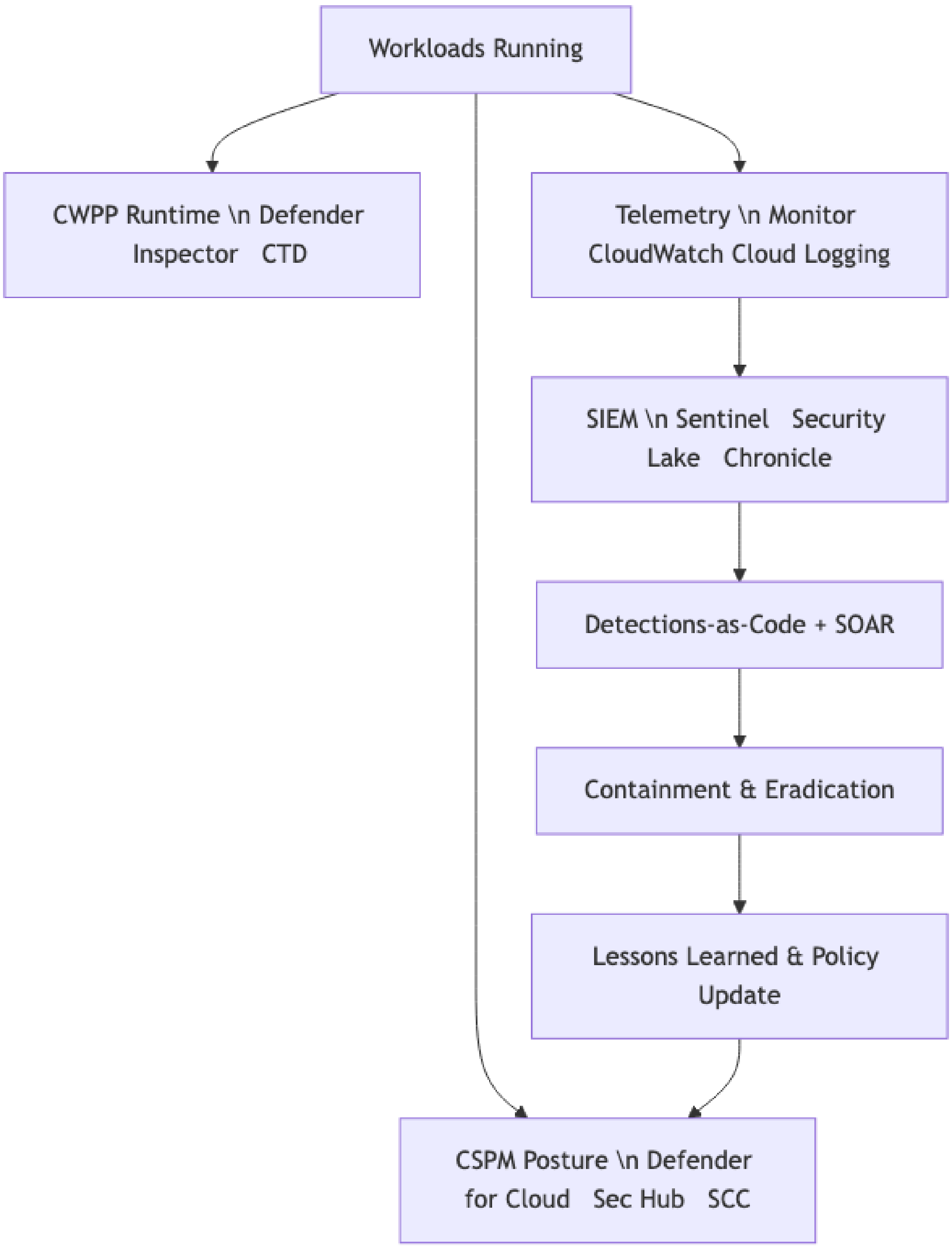
Inline PNGs


Business Objectives
Threat Modeling \n STRIDE LINDDUN
Security Requirements \n User Stories Acceptance Criteria
Policy as Code \n Azure Policy   AWS SCP+Config GCP Org Policy
IaC Baselines \n Terraform Bicep CDK KRM
Pre-commit & CI IaC Scanning \n Checkov Terrascan OPA
Design Review Gate


Developer IDE
Pre-commit Hooks \n lint tests secret-scan
Branch Push PR
SAST
E
F
PR Checks & Severity Gates
Pass
Fail
Approve & Merge
Fix Findings


Source Merge
Build Container Artifact
SBOM Generate
I
Risk Budget Gate
Pass
Fail
Promote to Deploy Stage
H


Release Approval
IaC Apply \n Terraform Bicep CDK KRM
Guardrails \n Azure Policy OPA BinAuthz
Runtime Identities \n Managed Identity IRSA Workload Identity
Secrets via Vaults \n Key Vault Secrets Manager Secret Manager
Network Controls \n WAF, Private Links, Firewall, VPC SC
Deploy to Target Env


Workloads Running
CWPP Runtime \n Defender Inspector CTD
Telemetry \n Monitor CloudWatch Cloud Logging
SIEM \n Sentinel Security Lake Chronicle
Detections-as-Code + SOAR
Containment & Eradication
Lessons Learned & Policy Update
CSPM Posture \n Defender for Cloud Sec Hub SCC