National College of Ireland

Healthcare-Grade Cross-Cloud Policy Delivery: Signed OPA Bundles as OCI Artifacts on AWS, Azure & GCP

MSc Research Project
MSc in Cloud Computing

Lakshmi Prasanna Maddipoti
Student ID: 23409029

School of Computing
National College of Ireland

Supervisor: Sean Heneey

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Lakshmi Prasanna Maddipoti
Student ID:	23409029
Programme:	MSc in Cloud Computing **Year:** 2025
Module:	Research in Project
Supervisor:	Sean Heneey
Submission Due Date:	28/01/2026
Project Title:	Healthcare-Grade Cross-Cloud Policy Delivery: Signed OPA Bundles as OCI Artifacts on AWS, Azure & GCP
Word Count:	1265 **Page Count:** 13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: M.L.Prasanna

Date: 28/01/2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Lakshmi Prasanna Maddipoti
23409029

1. Introduction

Provide a complete, operator-focused configuration manual for building, signing, publishing, discovering, verifying, and enforcing OPA policy bundles delivered as OCI artifacts across AWS, Azure, and GCP, with verify-before-use gating and audit-ready evidence outputs.

2. Requirements

2.1. Hardware

- Local workstation (Windows/Linux/macOS)
    - CPU: 2+ cores (4+ recommended for faster bundle builds and repeated verify/test runs)
    - RAM: 4 GB minimum (8 GB recommended)
    - Disk: 1–2 GB free for tools + artifacts + evidence (more if you retain long evidence history)
- Cloud shell / CI runner
    - Any standard Linux shell environment with outbound internet (HTTPS/443) and enough disk to download binaries and store a few bundles/evidence files (typically < 500 MB is sufficient for MVP-scale runs)
- Network Outbound HTTPS (TCP 443) access to:
    - OCI registries (ECR/ACR/GAR/GHCR as used)
    - Cloud identity endpoints used by your auth flow (AWS/Azure/GCP)
    - Optional Sigstore services only if you use keyless signing (otherwise not required for offline/keyed verification)

2.2. Software

Core toolchain is as follows:

- Open Policy Agent (OPA)
- ORAS CLI
- Cosign (Sigstore)

Imports

```python
import os, shutil, subprocess
from pathlib import Path

os.environ["PATH"] += r";C:\Tools"
os.environ.setdefault("COSIGN_PASSWORD", "1")

COSIGN = r"C:\Tools\cosign.exe" if Path(r"C:\Tools\cosign.exe").exists() else shutil.which("cosign")
if not COSIGN or not Path(COSIGN).exists():
    raise SystemExit("Cosign not found at C:\\Tools\\cosign.exe (or on PATH).")

def run(cmd, *, check=True, capture=True, env=None, cwd=None):
    p = subprocess.run(
        cmd, text=True, capture_output=capture, check=False,
        env=(env or os.environ.copy()), cwd=cwd
    )
    if check and p.returncode != 0:
        raise RuntimeError(
            f"Command failed ({p.returncode}): {' '.join(cmd)}\n"
            f"--- stdout ---\n{p.stdout}\n--- stderr ---\n{p.stderr}"
        )
    return p

keys_dir = Path("keys"); keys_dir.mkdir(exist_ok=True)
priv = keys_dir / "cosign.key"
pub  = keys_dir / "cosign.pub"
derived_pub = keys_dir / "from_priv.pub"

print("Using cosign at:", COSIGN)
print("Keys dir:", keys_dir.resolve())
```

```
Using cosign at: C:\Tools\cosign.exe
Keys dir: C:\Users\pc\Desktop\Prasanna Code\keys
```

- Cloud CLIs (for registry creation/auth where applicable)
    - AWS CLI (ECR)
    - Azure CLI (ACR)
    - Google Cloud SDK (gcloud) (Artifact Registry)
- Utilities (recommended)
    - jq for extracting digests/fields from JSON outputs
    - sha256sum (or equivalent) for file hashing
    - tar (or a deterministic packer implementation if not using opa build)

Keys

```
if not priv.exists() or not pub.exists():
    print("Keys missing -> generating key pair in current directory...")
    for stray in [Path("cosign.key"), Path("cosign.pub")]:
        if stray.exists():
            stray.unlink()

    run([COSIGN, "generate-key-pair"], check=True, capture=True)

    src_priv = Path("cosign.key")
    src_pub  = Path("cosign.pub")
    if not src_priv.exists() or not src_pub.exists():
        raise SystemExit("cosign did not produce cosign.key/cosign.pub in the current directory.")
    src_priv.replace(priv)
    src_pub.replace(pub)
    print("Keys created and moved to ./keys")

p = run([COSIGN, "public-key", "--key", str(priv)], check=True, capture=True)
derived_pub.write_text(p.stdout, encoding="utf-8")

print("Wrote:", derived_pub)
print("Same as keys/cosign.pub? ->", pub.read_text(encoding="utf-8") == derived_pub.read_text(encoding="utf-8"))
```

```
Wrote: keys\from_priv.pub
Same as keys/cosign.pub? -> True
```

- If using a Python CLI wrapper
    - Python 3.10+ recommended (3.8+ acceptable if dependencies permit)
    - Ability to run subprocess tools and write evidence JSON to disk
- Accounts, credentials, and permissions
    - A registry namespace/repository in each target cloud registry (ECR/ACR/GAR), plus credentials that permit:
    - Publishers: push artifact manifests/blobs and publish associated signature artifacts (referrers)
    - Consumers/verifiers: pull by digest and discover associated artifacts (referrers) as required
    - Public key distribution (cosign.pub) to all verifier environments
    - Private key protection (file-backed MVP keys stored securely; never committed)

3. Definitions and Conventions

Following are some of the important definitions:

- Bundle: OPA bundle archive (e.g., bundle.tar.gz) containing Rego + data.
- OCI artifact: Content stored in an OCI registry using an OCI manifest/config/layers (not necessarily a container image).
- Tag: Mutable reference (e.g., :v1, :latest). Not trusted as a security anchor.
- Digest pin: Immutable reference @sha256:<digest>; unit of trust.
- Subject: The artifact being signed (the OCI manifest identified by digest).

- Referrer: An OCI associated artifact that references a subject (e.g., signatures, attestations).
- Verify-before-use gate: Hard requirement: if verify fails or required signature is missing → do not execute OPA.

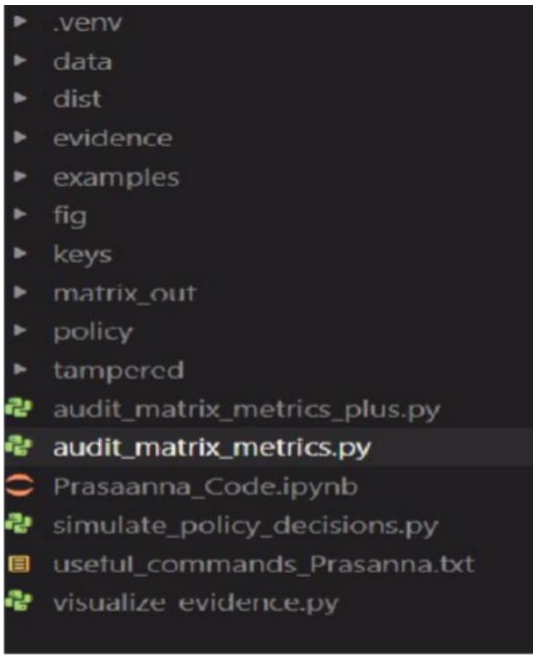
Following are some of the important notations:

- REGISTRY_HOST examples:
- ECR: <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com
- ACR: <ACR_NAME>.azurecr.io
- GAR: <LOCATION>-docker.pkg.dev
- REPO example: medpolicy/policies
- REF_TAG example: prod-2025-12-31
- SUBJECT example: ${REGISTRY_HOST}/${REPO}@sha256:${DIGEST}

4. Repository Layout (Operational Standard)

Recommended layout:

- policy/ Rego + data inputs (source of truth)
- dist/ built bundle outputs (e.g., dist/bundle.tar.gz)
- keys/ public key only in repo; private key stored securely outside repo
- evidence/ JSON evidence per run
- tampered/ tamper-lab variants (optional for experiments)



Security rule: Never commit cosign.key (private key) to GitHub or shared storage.

5. Deterministic Build Configuration

You may use either:

- OPA bundle build
  - mkdir -p dist
  - opa build -b ./policy -o ./dist/bundle.tar.gz

Building

```
import os, subprocess
from pathlib import Path
```

```
os.environ["PATH"] += r";C:\Tools"
os.environ["COSIGN_PASSWORD"] = os.environ.get("COSIGN_PASSWORD", "1")

need("cosign"); need("opa")

src = Path("policy")
dist = Path("dist"); dist.mkdir(exist_ok=True)
bundle_path = (dist / "bundle.tar.gz").resolve()
sig_out = Path(str(bundle_path) + ".sig").resolve()
pub = (Path("keys")/"cosign.pub").resolve()
priv = (Path("keys")/"cosign.key").resolve()


for p in [bundle_path, sig_out]:
    if p.exists():
        p.unlink()


make_repro_tar(src, bundle_path)
```

```
WindowsPath('C:/Users/pc/Desktop/Prasanna Code/dist/bundle.tar.gz')
```

- Deterministic packer
  - If using a custom tar packer, enforce:
  - Sorted file order

    - Fixed UID/GID
    - Fixed mtime
    - Consistent permissions normalization (or preserved if required, but stable)
- Compute bundle digest (local evidence)
    - sha256sum ./dist/bundle.tar.gz | tee ./evidence/bundle.sha256.txt

6. OCI Artifact Packaging Standard

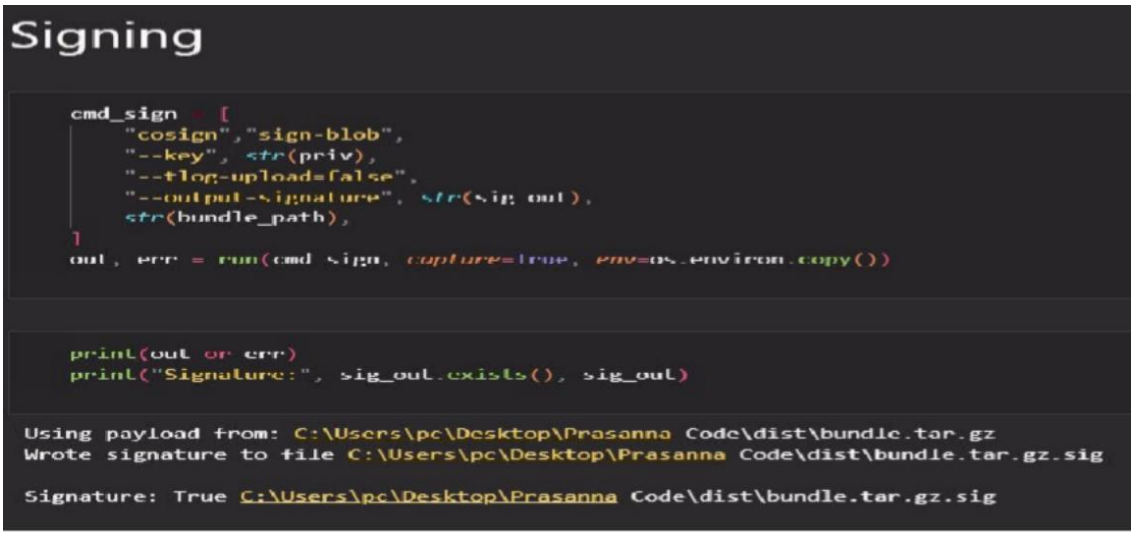
- Bundle layer: application/vnd.oci.image.layer.v1.tar+gzip
- Config blob: a small JSON config with a custom type
  eg. application/vnd.medpolicy.config.v1+json

```
cat > config.json <<'JSON'
{
  "artifact": "opa-bundle",
  "name": "medpolicy",
  "version": "1"
}
JSON
```

7. Signing Configuration (Cosign)

Generate keys (store private key securely outside repo; keep public key distributed to verifiers):

- cosign generate-key-pair



Sign the subject digest:

- cosign sign --key ./cosign.key ${SUBJECT}

Veryfiying

```
cmd_verify = [
    "cosign","verify-blob",
    "--key", str(pub),
    "--signature", str(sig_out),
    "--insecure-ignore-tlog",
    str(bundle_path),
]
run(cmd_verify, capture=False, env=os.environ.copy())
print("Verify OK and Done")
```

```
Verify OK and Done
```

Verification (required gate)

- cosign verify --key ./cosign.pub ${SUBJECT} | tee evidence/cosign-verify.json

Evaluating

```
need("opa")
import tarfile, tempfile
from pathlib import Path
def eval_with_input(bundle_path: Path, query: str, input_path: Path):
    with tempfile.TemporaryDirectory() as td:
        td = Path(td)
        with tarfile.open(bundle_path, "r:gz") as tar:
            tar.extractall(td)
        cmd = ["opa","eval","-i", str(input_path), "-d", str(td), query, "--format","pretty"]
        out, err = run(cmd, capture=True)
        return out.strip()

res_allow = eval_with_input(bundle_path, "data.med.allow", Path("examples/input_allow.json"))
res_deny  = eval_with_input(bundle_path, "data.med.allow", Path("examples/input_deny.json"))
```

- Gate rule (fail closed):
    - If cosign verify exits non-zero → stop pipeline immediately.
    - No OPA evaluation is permitted after a verification failure.

8. Referrers Discovery (Signatures)

Discover referrers for the subject digest:

- oras discover ${SUBJECT} -o json | tee evidence/referrers.json

Operational checks:

- Referrers output must include at least one signature artifact for regulated mode.
- If referrers are empty, treat them as non-compliant unless your policy explicitly allows offline signature distribution.

9. Registry Configuration by Cloud

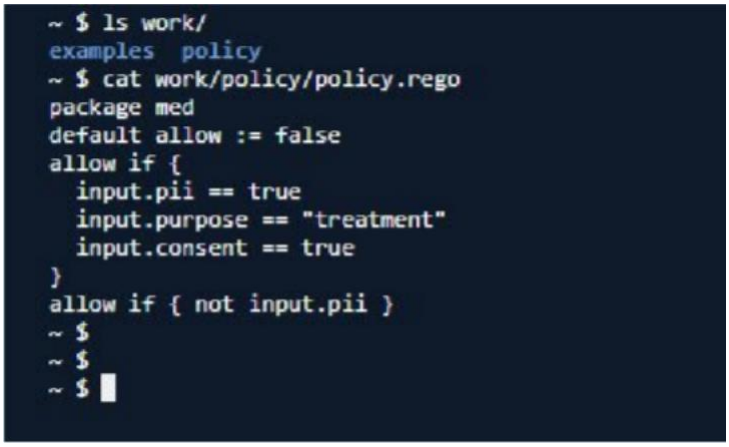
The artifact shape and commands remain cloud-neutral and the cloud differences are primarily authentication and IAM/RBAC.

AWS ECR (Elastic Container Registry)

- Create repository
    - aws ecr create-repository --repository-name ${REPO} --region <REGION>
- Login for ORAS
    - aws ecr get-login-password --region <REGION> \
    - | oras login ${REGISTRY_HOST} --username AWS --password-stdin

Minimum IAM guidance (high level)

- Publishers: permissions to create repo (optional), push blobs/manifests, and publish associated artifacts.
- Consumers/verifiers: permissions to pull blobs/manifests and query associated metadata/referrers (if supported in your environment).



Azure ACR (Azure Container Registry)

- Create registry
  - az group create -n <RG> -l <LOCATION>
  - az acr create -n <ACR_NAME> -g <RG> --sku Basic

Headless token login (recommended for Cloud Shell / CI)

- TOKEN=$(az acr login -n <ACR_NAME> --expose-token --output tsv --query accessToken)
- echo "$TOKEN" | oras login <ACR_NAME>.azurecr.io \
- --username 00000000-0000-0000-0000-000000000000 \
- --password-stdin

RBAC

- Publishers: role equivalent to push/upload to the registry.
- Consumers/verifiers: pull permissions plus metadata discovery as required.

Google Artifact Registry (GAR)

- Enable API and create Docker-format repository
    - gcloud services enable artifactregistry.googleapis.com
    - gcloud artifacts repositories create <REPO_NAME> \
    - --repository-format=docker \
    - --location=<LOCATION>

ORAS login using access token

- gcloud auth print-access-token| oras login <LOCATION>-docker.pkg.dev --username oauth2accesstoken --password-stdin

IAM

- Publishers: writer role to push artifacts.
- Consumers/verifiers: reader role to pull and (where needed) query attachments/metadata.

10. Runtime Consumption (OPA) Configuration

Verify-before-use enforcement sequence

- Resolve digest → construct SUBJECT
- cosign verify (gate)
- Pull bundle by digest (or use local bundle) only after verify
- opa eval

11. Evidence Configuration

Evidence files per run (minimum set) store under evidence/<run_id>/:

- bundle.sha256.txt
- descriptor.json (subject descriptor with digest)
- referrers.json
- cosign-verify.json
- opa-eval.json (OPA decision outputs)
- run.json (summary record)

```
run,status,evidence_completeness_score,has_timestamp,has_bundle_digest,has_signature_path,has_public_key_path,
1765808422,WARN,0.8571428571428571,True,True,True,True,True,False,,True,True,True
1765808424,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766466705,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766466708,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766472201,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766472204,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766473787,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766473790,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766473795,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766473798,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766473802,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766473804,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766474049,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766474052,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
1766474119,WARN,0.8571428571428571,True,True,True,True,True,True,False,,True,True,True
1766474121,FAIL,0.7142857142857143,True,True,False,True,False,,True,True,True,True,True
```

Evidence schema (recommended) run.json should include at least:

- Timestamp (UTC)
- Tool versions (opa/oras/cosign + cloud CLI used)
- Bundle digest (local file digest)
- Registry subject digest (manifest digest)
- Signature verification outcome (pass/fail + signer info if available)
- Referrers discovered (count + identifiers)
- OPA evaluation query, inputs (hashes), outputs (decisions)
- Exit codes and stderr summaries on failure

audit_summary.json

matrix_out > audit_summary.json > ...


```json
{
  "runs_total": 16,
  "status_counts": {
    "WARN": 8,
    "FAIL": 8
  },
  "pass_rate": 0.0,
  "warn_rate": 0.5,
  "fail_rate": 0.5,
  "unique_bundle_digests": 8,
  "bundle_digest_counts": {
    "sha256:8c771596433efbe8c2881d9124d082a00ae7b91964e8bf26bae625d673dfe18b": 2,
    "sha256:fe1426e9a9af1fb6212635bedc008ee3ae1e4bbd28edb1c39c688783f7f9e107": 2,
    "sha256:fe44551208e35cbae419092240a417f95436a601451b6655c59d3ed2a903f61d": 2,
    "sha256:f86869a2d6be238a0ee87dd72f52b43aed0ac37b6cbf0c93f22123f82c745f7c": 2,
    "sha256:105ac2968ca2e1c1692ace621e91a6cc170504f2ad03f9284da800364973c566": 2,
    "sha256:8a81afc8df5f3b8a64ea1756b7f37fa44d6ff116e948308bf73ff6ec100fba74": 2,
    "sha256:065c50f32ee49c1a97316a17d366d58f639a24d71a39cc956e8917a79e7786c7": 2,
    "sha256:b889187eb7a0797a13a71e7a7bf4e1e552ad72f2a2bcacc496083f5d8d9959b0": 2
  },
  "avg_evidence_completeness": 0.7857142857142857,
  "tamper_expected_rate": 1.0,
  "opa_sanity_pass_rate": 1.0,
  "verify_ok_rate": 1.0,
  "private_key_path_logged_rate": 1.0
}
```


12. References

- Open Policy Agent Documentation Bundles (management bundles) and OCI registry usage
- Open Policy Agent Documentation OPA configuration example using services: type: oci for downloading bundles from OCI repositories
- ORAS Documentation oras discover command (referrers discovery; documented as preview/under development)
- Microsoft Learn (Azure Container Registry) Managing OCI/supply-chain artifacts with ORAS; explains referrers API usage and oras discover for reference graphs
- Open Containers Initiative OCI Image & Distribution Specs v1.1 release notes; describes the referrers API endpoint and the subject/artifactType model
- Open Containers Initiative OCI Distribution Specification project (registry API standardization)
- Sigstore Cosign Cosign overview and offline verification notes (bundle materials /annotation behavior)
- Sigstore Docs Cosign "Signing Other Types"; signatures stored as OCI artifacts in registries

- AWS Open Source Blog OCI 1.1 support in Amazon ECR; explains subject field and the referrers API in the registry context
- Red Hat Blog OCI 1.1 Referrers API overview and interoperability/security benefits