PLOT – Secure App Launch Sandbox (Self-Built Cloud) v1.0

Zweck: Eine universelle, sichere Plattform, auf der **Programme/Software/Applikationen** in einer **selbstgebauten Cloud-Umgebung** gestartet, getestet und betrieben werden können – individuell konfigurierbar, aber mit **wiederverwendbaren Standards**.

---

1) Leitbild & Architekturprinzipien

- **Universalität**: Container (OCI), Functions/Serverless, Batch/Jobs, ggf. **WASM (WASI)** und **Micro-VMs (Firecracker/Kata)**.
- **Sicherheit zuerst**: Zero-Trust-Netz, mTLS, signierte Images, Politik-Durchsetzung (Admission), isolierte Mandanten.
- **Selbstbedienung mit Guardrails**: Dev-Teams starten Apps selbst; Plattform setzt Grenzen (Quotas, Policies, Budgets).
- **Reproduzierbarkeit**: IaC, deklarativ (GitOps), Ephemeral Environments, deterministische Seeds.
- **EU-Konform**: Datenresidenz, DSGVO, Audit-Trail.

Ebenen 1. **Control Plane** (Admin): Identity/RBAC, Policy, Observability, Catalog. 2. **Data Plane** (Workloads): Kubernetes-Cluster + Add-ons (oder Nomad), Isolationszonen, Objekt-Storage, Message-Bus. 3. **Developer UX**: Self-Service Portal/CLI/API, Templates, One-Click-Sandboxes.

---

2) Runtimes & Start-Modi (universell)

- **Container**: Deployment/StatefulSet, CronJob, Argo Workflows für Pipelines.
- **Serverless**: Knative/Functions – schnelle Starts, Auto-Scale to 0.
- **WASM**: Spin/Kwasm-Runtime für ultra-leichte Services.
- **Micro-VMs**: Firecracker/Kata für starke Isolierung spezieller Workloads.

Start-API (Skizze)

```
POST /api/launch/run
{
  "runtime": "container|function|wasm|microvm",
  "artifact": "registry.example/app:1.2.3",
  "env": {"LOG_LEVEL":"info"},
  "resources": {"cpu":"500m","memory":"512Mi"},
  "exposure": {"ingress": true, "auth": "oidc"},
  "tenant": "ten-acme", "project": "prj-42"
}
```

---

3) Sicherheit & Isolierung

Netz & Identitäten - **Service Mesh (mTLS)**: automatische Identität pro Pod (SPIFFE/SPIRE), Traffic-Verschlüsselung. - **NetworkPolicies** (Cilium/Calico): Default-Deny, nur erlaubte egress/ingress.

Supply-Chain & Admission - **Image-Signaturen** (Sigstore/Cosign), **SBOM** (Syft), **Vuln-Scanning** (Trivy/Grype). - **Admission-Regeln** (Kyverno/Gatekeeper): nur signierte, nicht-kritische CVEs, Root-Verbot.

Laufzeit-Sicherheit - **PodSecurity**: `restricted` Profile, seccomp, readOnlyRootFS, Drop CAPs. - **gVisor/Kata** für feinkörnige Sandboxen.

Secrets - External Secrets Operator (ESO) + **Vault**/KMS, kurzlebige Token, Rotationen, kein Secret im Repo.

Beispiel Kyverno-Policy (signierte Images)

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata: {name: require-signed-images}
spec:
  validationFailureAction: enforce
  rules:
  - name: verify-signature
    match: {resources: {kinds: [Pod]}}
    verifyImages:
    - image: "registry.example/*"
      key: "cosign.pub"
```

---

4) Mandanten, Projekte & Guardrails

- **Tenant-Factory**: `Namespace + ResourceQuota + LimitRange + NetworkPolicy` je Team/Projekt.
- **Rollen**: `OWNER`, `DEV`, `OPS`, `AUDITOR`; feingranular via OPA/ABAC (Projekt-Scopes).
- **Kosten-Leitplanken**: Quotas, Horizontal/Vertical Autoscaling-Grenzen, Idle Kill für Sandboxes.

Namespace-Bootstrap (Snippet)

```
apiVersion: v1
kind: Namespace
metadata: {name: prj-42}
---
apiVersion: v1
kind: ResourceQuota
metadata: {name: prj-42-quota, namespace: prj-42}
spec: {hard: {requests.cpu: "8", requests.memory: "16Gi", pods: "50"}}
```

---

5) Developer Experience (DX)

- **Template-Katalog**: App-Blueprints (Web API, Worker, Batch, Realtime) als Helm/ArgoCD AppSets.
- **Backstage-Portal**: Create-Plugin „New App → Sandbox“.
- **CLI**: `plot sandbox up`, `plot app launch`, `plot app logs`, `plot app down`.
- **GitOps**: ArgoCD/Flux; alle Änderungen deklarativ, auditierbar.

Makefile (Auszug)

```
sandbox:
    terraform -chdir=infra/envs/sandbox apply -auto-approve
    helmfile apply

launch-api:
    curl -H "Authorization: Bearer $$TOKEN"
     -X POST $(PLOT_API)/launch/run -d @specs/launch.json
```

---

6) CI/CD & Promotions

- **Supply-Chain**: Build → SBOM → Scan → Sign → Push → Deploy (GitOps PR).
- **Gates**: CVE-Schwellen, Tests, Policy-Checks, manuelles 4-Augen-Gate für Prod.
- **Canary/Blue-Green** per Mesh/Ingress; **Ephemeral Envs** pro PR.

GitHub Actions (Signieren)

```
- name: Build
  run: docker build -t registry.example/app:${{ github.sha }} .
- name: SBOM
  run: syft packages registry.example/app:${{ github.sha }} -o spdx-json >
sbom.json
- name: Scan
  run: trivy image --exit-code 1 --severity HIGH,CRITICAL registry.example/
app:${{ github.sha }}
- name: Sign
  run: cosign sign --key cosign.key registry.example/app:${{ github.sha }}
```

---

7) Observability & Betrieb

- **Tracing** (OpenTelemetry), **Logs** (strukturierte JSON), **Metriken** (Prom/Grafana), SLO/SLI je App.
- **Runbooks**: Standard-Störungen (CrashLoop, OOM, PodBudget, RateLimits) + Entscheidungsbäume.
- **Health**: `/readyz`, `/livez`, synthetische Probes; Uptime-Probe extern.

---

8) Starten einer App – End-to-End

1) **Template klonen**: `apps/webapi/` → `prj-42`. 2) **Secrets binden**: `ExternalSecret` → Vault-Pfad. 3) **Build & Sign**: CI produziert Image + SBOM + Signatur. 4) **Launch PR**: ArgoCD tracked `env/sandbox/prj-42/app.yaml`. 5) **Admission** prüft Signatur & Policies, **Mesh** vergibt mTLS-Identität. 6) **Expose**: Ingress + OIDC-Auth; Rate Limit & WAF aktiv. 7) **Telemetry**: Logs/Traces/Metriken sichtbar; KPIs im Dashboard.

App-Manifest (hardening-Snippet)

```
apiVersion: apps/v1
kind: Deployment
metadata: {name: webapi, namespace: prj-42}
spec:
  template:
    spec:
      securityContext: {runAsNonRoot: true}
      containers:
      - name: web
        image: registry.example/webapi:1.2.3
        ports: [{containerPort: 8080}]
        securityContext:
          allowPrivilegeEscalation: false
          readOnlyRootFilesystem: true
          capabilities: {drop: ["ALL"]}
```

---

9) Compliance & Audit

- **Audit-Trail**: API/Launch-Events signiert, Hash-Kette, Export.
- **Datenschutz**: Pseudonymisierte Logs, PII-Filter, Data Residency.
- **Least-Privilege**: IAM-Boundaries, Short-Lived Tokens, Just-In-Time-Zugriff.

---

10) Universelle Erweiterbarkeit

- **Runtimes als Plugins** (Container, WASM, Micro-VM, Batch).
- **Provisioning**: Neue Sandbox-Cluster per Modul (Terraform), identische Add-ons.
- **Policies als Code**: Repos für Kyverno/OPA, getestete Policy-Releases.

---

11) „Sandbox-Ready“ Checkliste

- [ ] IaC deployt Control & Data Plane (EU-Region) in <30 min.
- [ ] Launch-API nimmt Container & Functions an; Policies greifen.
- [ ] Logs/Traces/Metriken pro App sichtbar; Alarmrouten definiert.
- [ ] Onboarding-Template & Backstage-Wizard funktionieren.

- [ ] Audit-Export & Signaturprüfung verifiziert.

---

12) Nächste Schritte

1) Backstage-Template & Argo AppSet für **Web-API** und **Worker** bereitstellen. 2) Kyverno-Bundle (signierte Images, Root-Verbot, Ressourcengrenzen) aktivieren. 3) Vault/ESO-Pfadstruktur definieren; Beispiel-Secret. 4) Mesh mTLS & OIDC-Ingress aktivieren; Beispielservice veröffentlichen. 5) Runbooks & „Day-2“-Guides ins Portal stellen.

> Ergebnis: Eine **individuell konfigurierbare**, **universell einsetzbare** Startplattform, die in deiner selbstgebauten Cloud sicher läuft – vom Ein-Klick-Sandbox bis zum gehärteten Dauerläufer.