From Incident to Innovation

Security automation and zero-downtime for Azure secrets

Press Space for the next page →

Sponsors

Microsoft

AGIC

present
exprivia group

BizzyNow
make yourself better

Agenda

- The incident pattern and why it repeats
- Design goals: security, auditability, zero-downtime
- Event-driven architecture on Azure
- Business impact: estimated hours saved per month
- Deep dive: Event Grid trigger and rotation
- Operations: SharePoint sync and backups
- IaC + delivery: Bicep, GitHub Actions, Dependabot
- Copilot support for implementation and documentation
- Future improvements: AI-assisted operations
- Takeaways and practical next steps

Why this story

- Secrets expire on weekends and holidays too
- Manual rotation = security gaps and audit pain
- **Downtime** is the most expensive kind of learning

What changed

- Event Grid + Functions replace manual scripts
- Jira + SharePoint become the system of record
- Backups and replay for forensic confidence
- A reusable blueprint for secret hygiene

The real cost of downtime

Human impact

- On-call stress and weekend firefighting
- Context switching derails planned work
- Manual effort repeated every single month
- Shared secrets erode hygiene and team trust

Business impact

We manage **150 app registrations**.
One failing secret on a critical integration:

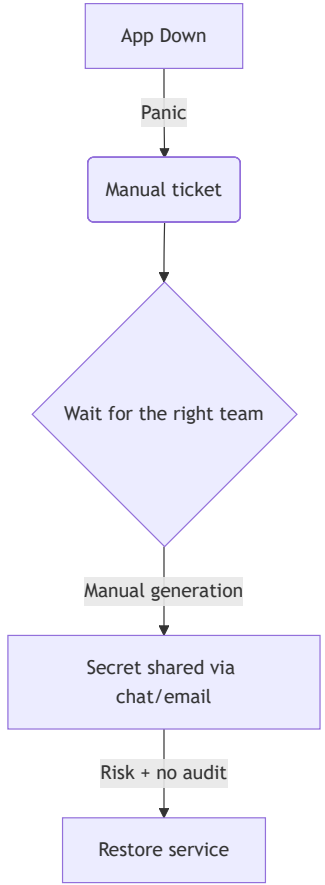
Client annual revenue	€1,000,000,000
Working hours/day	12
Cost per working hour	**~€250,000**

> Even a short outage on the wrong app
> can cost **hundreds of thousands of euros**.

The incident pattern we inherited

What actually breaks

- Critical apps go down with no warning
- Logs show: "client secret expired"
- Escalations, manual firefighting, long MTTR
- This stems from good intentions: "secrets should expire every 6 months"
- **Security debt**: secrets shared via chat or email, no audit trail



The legacy detection approach

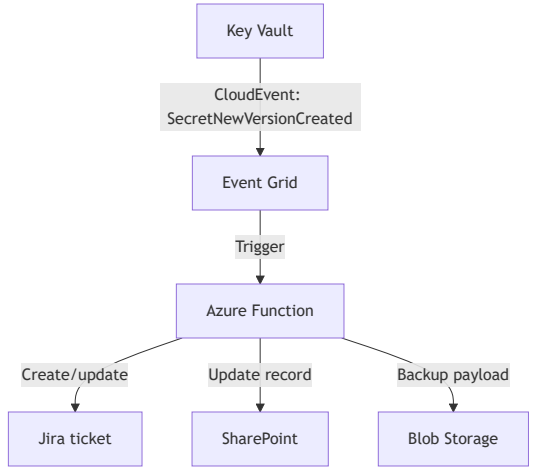
- Someone runs scripts "every now and then" to discover what is expiring...
- ❌ Reactive, slow, and brittle
- ❌ No context or audit trail for operations
- ❌❌❌❌ I'm the one manually doing it, each month, through manually created tickets ❌❌❌❌

The target design

Turn incidents into a **proactive, event-driven** workflow.

Key components

- **Entra ID + Key Vault**: system of record for secrets
- **Event Grid system topics**: CloudEvents on secret changes
- **Azure Functions**: serverless workflow + integrations
- **Jira + SharePoint**: ticketing and tracking
- **Blob Storage**: immutable event replay and backups

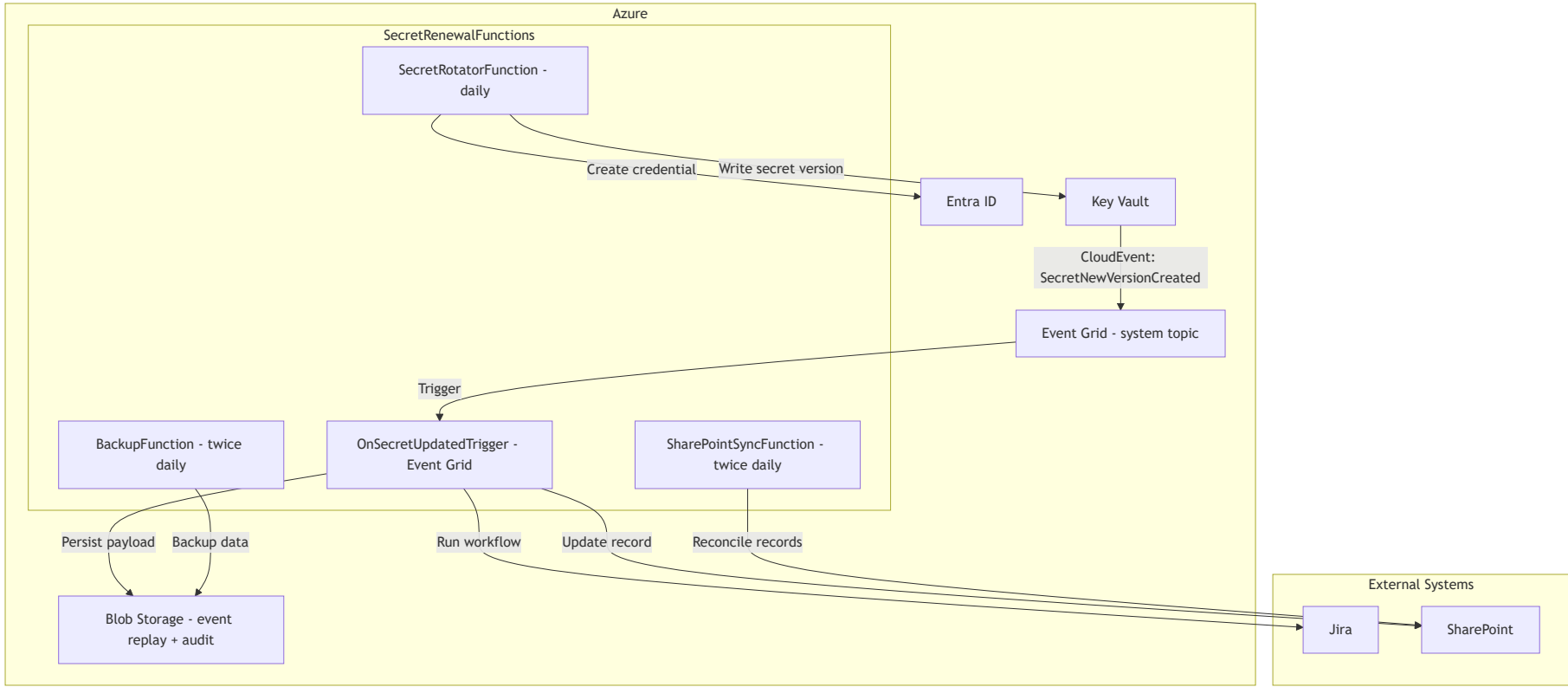


"I'm wicked smart, and with AI I'll be done in 5 minutes"

More problems

- What about resiliency
- What about division of responsibilities and least privilege?
- What about monitoring, alerting, and auditing?
- What about backups and replay for recovery and compliance?
- What about...?

Full architecture: close the loop



Business value: estimated monthly hours saved

For 150 app registrations

- Manual model (monitoring + coordination + updates): about 35 min per app per month
- Automated model (exceptions + approvals + review): about 8 min per app per month
- Estimated saving: about 27 min per app per month

Monthly impact

- 150 x 27 min = 4,050 min per month
- About 67.5 hours saved every month
- At 200 apps, about 90 hours saved per month

Why this matters across teams

- Fewer cross-team status pings and handoffs
- Less waiting between Wuerth teams and client teams
- Faster renewals with a shared system of record (Jira + SharePoint)
- Time shifts from firefighting to hardening and onboarding

Estimate based on... ai

I just made it up, since sometimes it was 4 days of actual work per month, sometimes just 2 hours

Please don't tell everyone that I pulled these numbers out of thin air 🙈

Deep Dive: Event Grid → Function (CloudEvents)

What the trigger does (by design):

- Logs the CloudEvent envelope for traceability
- Persists the raw payload to Blob Storage (`event-grid-payloads`) for replay/audit
- Renames the blob with the secret name for fast filtering
- Validates `SecretNewVersionCreated` and ignores non-AppId secret names
- Runs the shared workflow and **rethrows on failure** to enable Event Grid retries

Function(nameof(OnSecretUpdatedTrigger))

```
public async Task Run([EventGridTrigger] CloudEvent cloudEvent)
{
  var containerClient = _blobServiceClient.GetBlobContainerClient("event-grid-payloads");
  await containerClient.CreateIfNotExistsAsync();
  var blobClient = containerClient.GetBlobClient($"{cloudEvent.Id}.json");
  await blobClient.UploadAsync(new BinaryData(JsonSerializer.Serialize(cloudEvent)), overwrite: true);
}
```

Deep Dive: Proactive rotation (timer)

The rotator is intentionally conservative:

- Daily schedule (6 AM) and a next-month expiry threshold
- Skips apps with a newer credential (avoid double-rotation)
- Validates Key Vault references and URI structure before writing
- Creates a 6-month credential, writes to Key Vault, then relies on Event Grid

Function(nameof(SecretRotatorFunction))

```
public async Task Run([TimerTrigger("0 0 6 * * *")] TimerInfo myTimer)
{
  var records = (await _sharePointService.GetAllRenewalRecordsAsync())
    .OrderBy(r => r.NextExpiryDateTime)
    .ToList();
  var thresholdDate = DateTime.UtcNow.AddMonths(1).AddDays(1).Date;
  var expiringRecords = records.Where(r =>
    r.NextExpiryDateTime != null &&
    r.NextExpiryDateTime <= thresholdDate &&
    r.NextExpiryDateTime > DateTime.UtcNow.AddMonths(-1)).ToList();
}
```

Deep Dive: Automatic validation (what used to be manual)

The twice-daily sync validates each SharePoint row and repairs metadata:

- Normalize Key Vault references and flag invalid formats
- Ensure the App ID still exists in Entra ID
- Verify owners in Jira and Entra; enforce account ownership patterns
- Confirm Key Vault secret exists and assign Secrets User when needed
- Validate Jira ticket URL and creation window vs. expiry dates

IaC with Bicep

- Event Grid **system topic** sourced from Key Vault
- Delivery tuned for reliability: 3 attempts, 24h TTL

```
param includedEventTypes array = [
  'Microsoft.KeyVault.SecretNewVersionCreated'
  'Microsoft.KeyVault.SecretNearExpiry'
  'Microsoft.KeyVault.SecretExpired'
]

properties: {
  eventDeliverySchema: 'CloudEventSchemaV1_0'
  retryPolicy: {
    maxDeliveryAttempts: 3
    eventTimeToLiveInMinutes: 1440
  }
}
```

CI/CD pipeline

- Onboarding workflow generates parameter files and opens a PR per client
- **What-if** validates infra changes before any deployment
- **Deploy** wires Event Grid subscriptions only after what-if passes

```yaml
what_if:
  needs: create_pr
  environment: production
  steps:
    - uses: azure/login@v2
      with:
        client-id: ${{ secrets.AZURE_CLIENT_ID }}
        tenant-id: ${{ secrets.AZURE_TENANT_ID }}
        subscription-id: ${{ inputs.subscriptionId }}
    - name: What-if deployment
      run: |
        az deployment sub what-if \
          --subscription "${{ inputs.subscriptionId }}" \
          --location "${{ inputs.location }}" \
          --template-file infra/main.subscription.bicep \
          --parameters @infra/parameters/kv-eg-sub-${{ need
```

GitHub as force multiplier

Delivery and platform workflows

- GitHub Actions standardizes build, test, publish, and deployment gates
- Onboarding workflows generate repeatable infra changes per client
- Pull requests become a shared operating model across teams

Dependency hygiene

- Dependabot keeps NuGet and action versions current
- Security and maintenance updates are visible, reviewable, and trackable

GitHub Copilot impact

- Speeds up implementation drafts and refactors
- Helps produce clearer docs and runbooks while coding
- Accelerates this Slidev presentation creation and iteration
- Frees engineering time for architecture and reliability decisions

Who observes the observer?

Every function, workflow, and integration reports to **Application Insights** — so nothing fails silently.

Structured logging

- All functions use `ILogger<T>` wired to Application Insights via the SDK
- Correlation IDs propagate through Event Grid → Function chains
- Tracing... just works?

Automatic alerting

- Simple detection: exceptions > 0 = email with critical alert to team...
- ...30 days to fix though

Vibe coders after sending
AI code to production

Takeaways

- ✅ **Zero downtime**: proactive secret rotation
- 🔒 **Security-first operations**: no secrets shared manually outside secure channels
- ⚡ **Serverless integration**: low ops overhead, scalable by default
- 📃 **Auditable by design**: event payloads and backups for replay
- 🧘 **Less toil**: fewer late-night surprises from expiry-driven outages

Future improvements

- Add self-service dashboards for teams and clients with explainable status
- Auto-generate owner-facing summaries and action checklists per app, possibly using AI (clustering)
- Expand policy-as-code checks before deployment and before rotation
- Improve app reg provisioning and deprovisioning workflows
- Self-service secret rotation trigger for teams and clients, with explainable status and audit trails
- Self-service app registration creation or onboarding for existing apps
- More tests???

Rate this talk

You can find me @ WÜRTH | IT

Q&A

Share your Feedback