

# From Incident to Innovation

Security automation and zero-downtime for Azure secrets

Press Space for the next page →



# Sponsors



# Why this story

## Reality check

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

## What changed

- Event Grid + Functions replace "best effort" scripts
- Jira + SharePoint become the system of record
- Backups and replay give us forensic confidence
- A proven event-driven pattern for secret hygiene
- Practical guardrails for auditability and resilience
- A blueprint you can reuse across teams

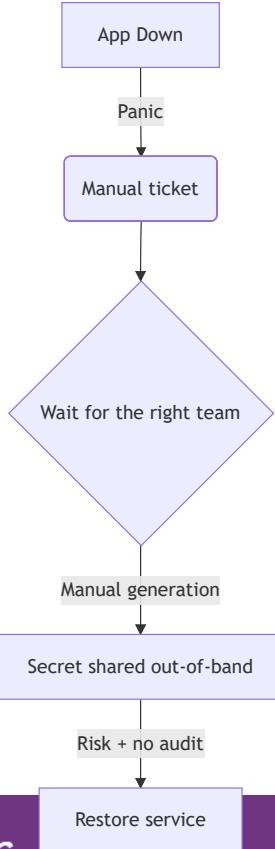
# Agenda

- 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

# 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
- Security debt: secrets shared out-of-band,  
weak audit trail



# The legacy detection approach

Someone runs scripts "every now and then" to discover what is expiring...

TODO insert image meme here

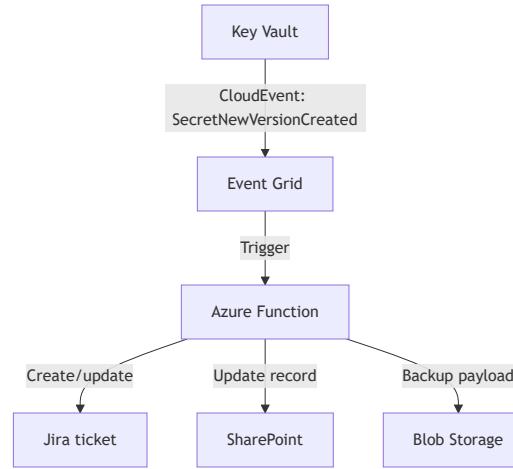
✗ Reactive, slow, and brittle

# The target design

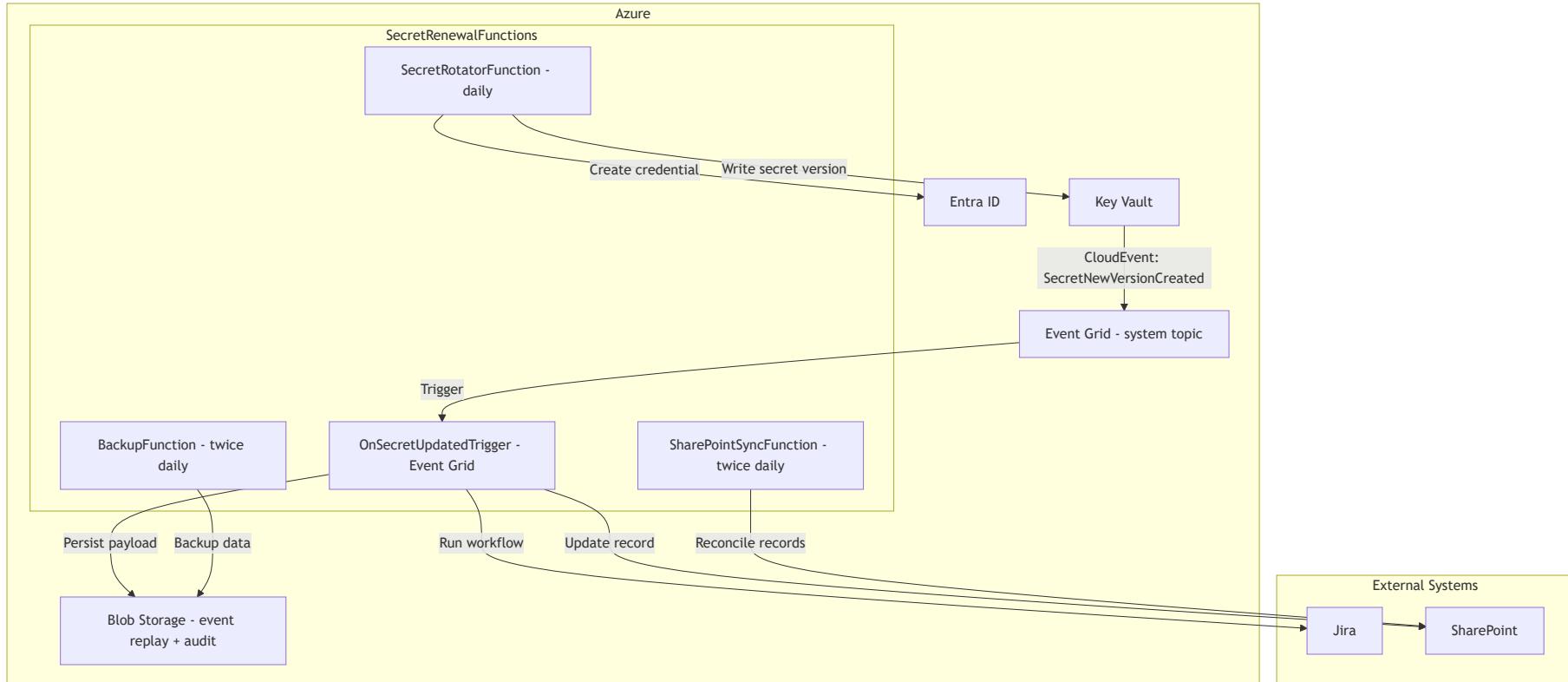
Turn incidents into a proactive, event-driven workflow.

## 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



# Full architecture: close the loop



# Business value: estimated monthly hours saved

## Why this matters across teams

### 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

- $150 \times 27 \text{ min} = 4,050 \text{ min per month}$
- About 67.5 hours saved every month
- At 200 apps, about 90 hours saved per month

- 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 recurring coordination-heavy operations across multiple teams.

# 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

# Infrastructure and delivery

## IaC with Bicep

- Event Grid system topic sourced from Key Vault
- CloudEvents schema, retry policy, filtered event types
- Subscription-scope `main.subscription.bicep` for per-client rollouts
- Event types include `SecretNewVersionCreated`,  
`SecretNearExpiry`, `SecretExpired`
- Delivery tuned for reliability: 3 attempts, 24h TTL, 1 event per batch

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

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

## CI/CD (GitHub Actions)

- Restore → build → test → publish → zip
- Deploy via `Azure/functions-action`
- Onboarding workflow generates parameter files and PRs per client
- What-if and deploy stages for Event Grid wiring

```
- name: Build SecretRenewalFunctions
  run: dotnet build --configuration Release --no-restore

- name: Test
  run: dotnet test --configuration Release --no-build --log

- name: Publish SecretRenewalFunctions
  run: dotnet publish src/SecretRenewalFunctions/SecretRenewalFunctions.csproj --configuration Release --output ./dist
```

# GitHub as force multiplier GitHub Copilot impact

## 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

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

## Dependency hygiene

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

# Takeaways

- Zero downtime: proactive secret rotation
- Security-first operations: no secret sharing out-of-band
- 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

Continuous Innovation = Reliable Ops

# Future improvements

- Predict upcoming risk windows using historical rotation and incident data
- Use AI to propose rotation timing with lowest business impact
- Auto-generate owner-facing summaries and action checklists per app
- Detect anomalous secret activity patterns earlier with assisted triage
- Expand policy-as-code checks before deployment and before rotation
- Add self-service dashboards for teams and clients with explainable status

# Rate this talk

---



You can find me @  WÜRTH|IT



## Q&A

Share your Feedback

