```
# 📊 Synthetic Analytics & Alerting

> **Series:** SYNTH | **Notebook:** 6 of 6 | **Created:** December 2025

## Dashboards, SLOs, and Alerting Strategies

This notebook covers advanced analytics for synthetic monitoring, including
building dashboards, configuring SLOs, and implementing effective alerting
strategies using the latest Dynatrace platform.

---

## Table of Contents

1. Analytics Overview
2. Availability Analysis
3. Performance Analysis
4. Location Comparison
5. SLO Configuration
6. Alerting Strategies
7. Dashboard Building


## Prerequisites

- ✅ Access to a Dynatrace environment with Synthetic Monitoring
- ✅ Completed SYNTH-01 through SYNTH-05
- ✅ Active synthetic monitors generating data




## 1. Analytics Overview

### Key Metrics for Synthetic Monitoring

| Metric Category | Metrics | Use Case |
|-----------------|---------|----------|
| **Availability** | Success rate, failure count | SLA compliance |
| **Performance** | Response time, TTFB, DNS | User experience |
| **Reliability** | Consecutive failures, MTTR | Stability |
| **Geographic** | Per-location metrics | Regional issues |

### Data Sources

| Source | Description | Best For |
|--------|-------------|----------|
```

````
| `bizevents` | Execution results | Real-time analysis |
| `dt.entity.synthetic_test` | Monitor definitions | Configuration |
| `dt.entity.synthetic_location` | Location info | Geographic analysis |
| `metrics` | Aggregated metrics | Long-term trends |

## 2. Availability Analysis

### Calculating Availability

```
Availability = (Successful Executions / Total Executions) × 100%

Example SLA Targets:
- 99.9% = max 8.76 hours downtime/year
- 99.5% = max 43.8 hours downtime/year
- 99.0% = max 87.6 hours downtime/year
```

```dql
// Overall synthetic availability (last 7 days)
fetch bizevents, from: now() - 7d
| filter event.provider == "dynatrace.synthetic"
| summarize {
    total_executions = count(),
    successful = countIf(synthetic.availability == true),
    failed = countIf(synthetic.availability == false)
  }
| fieldsAdd availability_pct = round((successful * 100.0) / total_executions,
decimals: 3)
| fieldsAdd downtime_minutes = round((failed * 5.0), decimals: 0)  //
Assuming 5-min frequency
```

```dql
// Availability by monitor (last 7 days)
fetch bizevents, from: now() - 7d
| filter event.provider == "dynatrace.synthetic"
| summarize {
    total = count(),
    successful = countIf(synthetic.availability == true),
    failed = countIf(synthetic.availability == false)
  }, by: {dt.entity.synthetic_test}
| fieldsAdd availability_pct = round((successful * 100.0) / total, decimals:
3)
| sort availability_pct asc
| limit 30
```
````

```dql
// Availability trend over time
fetch bizevents, from: now() - 7d
| filter event.provider == "dynatrace.synthetic"
| fieldsAdd hour_bucket = bin(timestamp, 1h)
| summarize {
    success_count = countIf(synthetic.availability == true),
    total_count = count()
  }, by: {hour_bucket}
| fieldsAdd availability_pct = round((success_count * 100.0) / total_count,
decimals: 2)
| sort hour_bucket desc
```

```dql
// Daily availability report
fetch bizevents, from: now() - 30d
| filter event.provider == "dynatrace.synthetic"
| fieldsAdd day = formatTimestamp(timestamp, format: "yyyy-MM-dd")
| summarize {
    total = count(),
    successful = countIf(synthetic.availability == true),
    failed = countIf(synthetic.availability == false)
  }, by: {day}
| fieldsAdd availability_pct = round((successful * 100.0) / total, decimals:
3)
| sort day desc
| limit 30
```

3. Performance Analysis

Performance Metrics

Metric	Description	Good	Warning	Critical
Response Time	Total execution time	< 2s	2-5s	> 5s
DNS	DNS resolution	< 50ms	50-200ms	> 200ms
Connect	TCP connection	< 100ms	100-300ms	> 300ms
TTFB	Time to first byte	< 500ms	500ms-1s	> 1s

```dql
// Response time percentiles by monitor
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    p50_ms = percentile(toDouble(synthetic.response_time), 50),
```

```
    p75_ms = percentile(toDouble(synthetic.response_time), 75),
    p90_ms = percentile(toDouble(synthetic.response_time), 90),
    p95_ms = percentile(toDouble(synthetic.response_time), 95),
    p99_ms = percentile(toDouble(synthetic.response_time), 99),
    executions = count()
  }, by: {dt.entity.synthetic_test}
| sort p95_ms desc
| limit 20
```

```dql
// Performance trend comparison (this week vs last week)
fetch bizevents, from: now() - 14d
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| fieldsAdd week = if(timestamp > now() - 7d, "This Week", else: "Last Week")
| summarize {
    avg_response_ms = avg(toDouble(synthetic.response_time)),
    p95_response_ms = percentile(toDouble(synthetic.response_time), 95),
    executions = count()
  }, by: {dt.entity.synthetic_test, week}
| sort dt.entity.synthetic_test, week
```

```dql
// Timing breakdown analysis
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    avg_dns_ms = avg(toDouble(synthetic.dns_time)),
    avg_connect_ms = avg(toDouble(synthetic.connect_time)),
    avg_ssl_ms = avg(toDouble(synthetic.ssl_time)),
    avg_ttfb_ms = avg(toDouble(synthetic.time_to_first_byte)),
    avg_download_ms = avg(toDouble(synthetic.download_time)),
    avg_total_ms = avg(toDouble(synthetic.response_time))
  }, by: {dt.entity.synthetic_test}
| fieldsAdd other_ms = avg_total_ms - avg_dns_ms - avg_connect_ms -
avg_ssl_ms - avg_ttfb_ms - avg_download_ms
| sort avg_total_ms desc
| limit 20
```

```dql
// Performance anomalies (response time > 2x average)
// First calculate average per monitor, then find executions above threshold
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
```

```
| filter synthetic.availability == true
| fieldsAdd response_ms = toDouble(synthetic.response_time)
| summarize {
    avg_response = avg(response_ms),
    max_response = max(response_ms),
    min_response = min(response_ms),
    executions = count()
  }, by: {dt.entity.synthetic_test}
| fieldsAdd deviation_factor = round(max_response / avg_response, decimals:
1)
| filter deviation_factor > 2
| sort deviation_factor desc
| limit 50
```

4. Location Comparison

Geographic Analysis

Compare synthetic results across locations to:
- Identify regional performance issues
- Detect CDN or DNS problems
- Validate global availability
- Optimize user experience by region

```dql
// Availability by location
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| summarize {
    total = count(),
    successful = countIf(synthetic.availability == true),
    failed = countIf(synthetic.availability == false)
  }, by: {dt.entity.synthetic_location}
| fieldsAdd availability_pct = round((successful * 100.0) / total, decimals:
2)
| sort availability_pct asc
| limit 30
```

```dql
// Performance by location
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    avg_response_ms = avg(toDouble(synthetic.response_time)),
    p50_ms = percentile(toDouble(synthetic.response_time), 50),
```

````
    p95_ms = percentile(toDouble(synthetic.response_time), 95),
    executions = count()
  }, by: {dt.entity.synthetic_location}
| sort avg_response_ms desc
| limit 30
```

```dql
// Location performance heatmap (monitor x location)
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    avg_response_ms = avg(toDouble(synthetic.response_time)),
    availability_pct = countIf(synthetic.availability == true) * 100.0 /
count()
  }, by: {dt.entity.synthetic_test, dt.entity.synthetic_location}
| sort dt.entity.synthetic_test, avg_response_ms desc
```

```dql
// Location-specific failures
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == false
| summarize {
    failure_count = count(),
    unique_monitors = countDistinct(dt.entity.synthetic_test),
    error_types = collectDistinct(synthetic.error_message)
  }, by: {dt.entity.synthetic_location}
| sort failure_count desc
| limit 20
```

## 5. SLO Configuration

### Service Level Objectives for Synthetic

| SLO Type | Metric | Example Target |
|----------|--------|----------------|
| **Availability** | Success rate | 99.9% |
| **Performance** | Response time P95 | < 3000ms |
| **Error Budget** | Allowed failures | 0.1% of executions |

### Creating Synthetic SLOs

1. **Navigate to**: Settings → Service-level objectives
2. **Create SLO**: Name, description, timeframe
````
````

````
3. **Define metric**: Use synthetic availability or response time
4. **Set target**: e.g., 99.9% availability
5. **Configure alerting**: Error budget alerts

```dql
// SLO calculation - Availability (30-day window)
fetch bizevents, from: now() - 30d
| filter event.provider == "dynatrace.synthetic"
| summarize {
    total_executions = count(),
    successful = countIf(synthetic.availability == true),
    failed = countIf(synthetic.availability == false)
  }, by: {dt.entity.synthetic_test}
| fieldsAdd sli_availability = round((successful * 100.0) / total_executions,
decimals: 4)
| fieldsAdd slo_target = 99.9
| fieldsAdd error_budget_total = round(total_executions * 0.001, decimals: 0)
// 0.1% budget
| fieldsAdd error_budget_remaining = round(error_budget_total - failed,
decimals: 0)
| fieldsAdd error_budget_pct = round((error_budget_remaining * 100.0) /
error_budget_total, decimals: 1)
| fieldsAdd slo_status = if(sli_availability >= slo_target, "✅ Met", else:
"❌ Breached")
| sort sli_availability asc
| limit 30
```

```dql
// SLO calculation - Performance (P95 < 3000ms)
fetch bizevents, from: now() - 30d
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    executions = count(),
    p95_response_ms = percentile(toDouble(synthetic.response_time), 95),
    within_target = countIf(toDouble(synthetic.response_time) < 3000)
  }, by: {dt.entity.synthetic_test}
| fieldsAdd sli_performance = round((within_target * 100.0) / executions,
decimals: 2)
| fieldsAdd slo_target = 95.0  // 95% of requests < 3s
| fieldsAdd slo_status = if(sli_performance >= slo_target, "✅ Met", else:
"❌ Breached")
| sort sli_performance asc
| limit 30
```

```dql
````
````

```
// Error budget burn rate (last 7 days)
fetch bizevents, from: now() - 7d
| filter event.provider == "dynatrace.synthetic"
| fieldsAdd day = formatTimestamp(timestamp, format: "yyyy-MM-dd")
| summarize {
    total = count(),
    failed = countIf(synthetic.availability == false)
  }, by: {dt.entity.synthetic_test, day}
| fieldsAdd daily_error_rate = round((failed * 100.0) / total, decimals: 3)
| fieldsAdd budget_consumed = round(failed * 100.0 / (total * 0.001),
decimals: 1)  // vs 0.1% budget
| sort dt.entity.synthetic_test, day desc
```

6. Alerting Strategies

Alert Types

Alert Type	Trigger	Use Case
Availability	Consecutive failures	Outage detection
Performance	Response time threshold	Degradation
SLO	Error budget consumption	Proactive warning
SSL	Certificate expiration	Security

Alert Configuration Best Practices

Setting	Recommendation	Reason
Consecutive failures	2-3	Avoid false positives
Location threshold	2+ locations	Confirm not local issue
Alert delay	5-10 minutes	Allow for transients
Auto-resolve	After 2 successes	Clear resolved issues

```dql
// Monitors with recent failures (potential outages)
// Check for monitors that have failed multiple times recently
fetch bizevents, from: now() - 1h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == false
| summarize {
    failure_count = count(),
    last_failure = max(timestamp),
    first_failure = min(timestamp)
  }, by: {dt.entity.synthetic_test, dt.entity.synthetic_location}
| filter failure_count >= 2
| sort failure_count desc
| limit 20
```

```

```dql
// Performance degradation alerts (P95 > threshold)
fetch bizevents, from: now() - 1h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    avg_response_ms = avg(toDouble(synthetic.response_time)),
    p95_response_ms = percentile(toDouble(synthetic.response_time), 95),
    executions = count()
  }, by: {dt.entity.synthetic_test}
| filter p95_response_ms > 5000  // > 5 seconds threshold
| sort p95_response_ms desc
```

```dql
// Multi-location failure detection (global outage indicator)
fetch bizevents, from: now() - 30m
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == false
| summarize {
    failing_locations = countDistinct(dt.entity.synthetic_location),
    failure_count = count(),
    locations = collectDistinct(dt.entity.synthetic_location)
  }, by: {dt.entity.synthetic_test}
| filter failing_locations >= 2
| fieldsAdd severity = if(failing_locations >= 3, "CRITICAL", else:
"WARNING")
| sort failing_locations desc
```

7. Dashboard Building

Recommended Dashboard Tiles

Tile Type	Content	Visualization
Overall Health	Availability %	Single value
Availability Trend	Hourly availability	Line chart
Response Time	P95 by monitor	Bar chart
Location Map	Geographic availability	World map
Failures Table	Recent failures	Table
SLO Status	Error budget	Gauge

Dashboard Layout
```

(
ciIHZpZXdCb3g9IjAgMCA4MDAgMzYwIj4KICA8ZGVmcz4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0
idGlsZUdyYWQxIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAgICAgPHN0
b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzEwYjk4MTtzdG9wLW9wYWNpdHk6MSIgL
z4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMDU5NjY5O3N0b3
Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmFkaWVudCB
pZD0idGlsZUdyYWQyIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAgICAg
PHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzNiODJmNjtzdG9wLW9wYWNpdHk6M
SIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMjU2M2ViO3
N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmFkaWV
udCBpZD0idGlsZUdyYWQzIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAg
ICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6I2VmNDQ0NDtzdG9wLW9wYWNpd
Hk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZGMyNj
I2O3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmF
kaWVudCBpZD0idGlsZUdyYWQ0IiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4K
ICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6I2Y1OWUwYjtzdG9wLW9wY
WNpdHk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZD
k3NzA2O3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxmaWx0ZXI
gaWQ9ImRhc2hTaGFkb3ciPgogICAgICA8ZmVEcm9wU2hhZG93IGR4PSIxIiBkeT0iMSIgc3RkRGV2
aWF0aW9uPSIyIiBmbG9vZC1vcGFjaXR5PSIwLjE1Ii8+CiAgICA8L2ZpbHRlcj4KICA8L2RlZnM+C
gogIDwhLS0gQmFja2dyb3VuZCAtLT4KICA8cmVjdCB3aWR0aD0iODAwIiBoZWlnaHQ9IjM2MCIgZm
lsbD0iI2Y4ZjlmYSIgcng9IjEwIi8+CgogIDwhLS0gVGl0bGUgLS0+CiAgPHRleHQgeD0iNDAwIiB
5PSIyOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjE4IiBmb250
LXdlaWdodD0iYm9sZCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+U3ludGhldGljI
E1vbml0b3JpbmcgRGFzaGJvYXJkIExheW91dDwvdGV4dD4KICA8dGV4dCB4PSI0MDAiIHk9IjQ4Ii
Bmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiM2NjY
iIHRleHQtYW5jaG9yPSJtaWRkbGUiPlJlY29tbWVuZGVkIHRpbGUgb3JnYW5pemF0aW9uIGZvciBv
cGVyYXRpb25hbCB2aXNpYmlsaXR5PC90ZXh0PgoKICA8IS0tIERhc2hib2FyZCBjb250YWluZXIgL
S0+CiAgPHJlY3QgeD0iMzAiIHk9IjY1IiB3aWR0aD0iNzQwIiBoZWlnaHQ9IjI4MCIgcng9IjEwIi
BmaWxsPSIjZmZmIiBzdHJva2U9IiNlMmU4ZjAiIHN0cm9rZS13aWR0aD0iMiIvPgoKICA8IS0tIFJ
vdyAxOiBLUEkgVGlsZXMgLS0+CiAgPHRleHQgeD0iNTAiIHk9IjkwIiBmb250LWZhbWlseT0iQXJp
YWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjN
jQ3NDhiIj5LRVkgTUVUUklDUzwvdGV4dD4KCiAgPCEtLSBBdmFpbGFiaWxpdHkgLS0+CiAgPHJlY3
QgeD0iNTAiIHk9IjEwMCIgd2lkdGg9IjE2MCIgaGVpZ2h0PSI3MCIgcng9IjgiIGZpbGw9InVybCg
jdGlsZUdyYWQxKSIgZmlsdGVyPSJ1cmwoI2Rhc2hTaGFkb3cpIi8+CiAgPHRleHQgeD0iMTMwIiB5
PSIxMjUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsb
D0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5PdmVyYWxsIEF2YW
lsYWJpbGl0eTwvdGV4dD4KICA8dGV4dCB4PSIxMzAiIHk9IjE1NSIgZm9udC1mYW1pbHk9IkFyaWF
sLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjI0IiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hp
dGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPjk5LjglPC90ZXh0PgoKICA8IS0tIEFjdGl2ZSBNb25pd
G9ycyAtLT4KICA8cmVjdCB4PSIyMjUiIHk9IjEwMCIgd2lkdGg9IjE2MCIgaGVpZ2h0PSI3MCIgcn
g9IjgiIGZpbGw9InVybCgjdGlsZUdyYWQyKSIgZmlsdGVyPSJ1cmwoI2Rhc2hTaGFkb3cpIi8+CiA
gPHRleHQgeD0iMzA1IiB5PSIxMjUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9u
dC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZ
GxlIj5BY3RpdmUgTW9uaXRvcnM8L3RleHQ+CiAgPHRleHQgeD0iMzA1IiB5PSIxNTUiIGZvbnQtZm
FtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIyNCIgZm9udC13ZWlnaHQ9ImJvbGQ
iIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj40NzwvdGV4dD4KCiAgPCEtLSBGYWls
ZWQgTW9uaXRvcnMgLS0+CiAgPHJlY3QgeD0iNDAwIiB5PSIxMDAiIHdpZHRoPSIxNjAiIGhlaWdod

D0iNzAiIHJ4PSI4IiBmaWxsPSJ1cmwoI3RpbGVHcmFkMykiIGZpbHRlcj0idXJsKCNkYXNoU2hhZG
93KSIvPgogIDx0ZXh0IHg9IjQ4MCIgeT0iMTI1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2V
yaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNo
b3I9Im1pZGRsZSI+RmFpbGVkIE1vbml0b3JzPC90ZXh0PgogIDx0ZXh0IHg9IjQ4MCIgeT0iMTU1I
iBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMjQiIGZvbnQtd2VpZ2
h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+MjwvdGV4dD4KCiAgPCE
tLSBTTE8gU3RhdHVzIC0tPgogIDxyZWN0IHg9IjU3NSIgeT0iMTAwIiB3aWR0aD0iMTgwIiBoZWln
aHQ9IjcwIiByeD0iOCIgZmlsbD0idXJsKCN0aWxlR3JhZDQpIiBmaWx0ZXI9InVybCgjZGFzaFNoY
WRvdykiLz4KICA8dGV4dCB4PSI2NjUiIHk9IjEyNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLX
NlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5
jaG9yPSJtaWRkbGUiPlNMTyBUYXJnZXQgTWV0PC90ZXh0PgogIDx0ZXh0IHg9IjY2NSIgeT0iMTU1
IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMjQiIGZvbnQtd2VpZ
2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+ODUlPC90ZXh0PgoKIC
A8IS0tIFJvdyAyOiBDaGFydHMgLS0+CiAgPHRleHQgeD0iNTAiIHk9IjE5NSIgZm9udC1mYW1pbHk
9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmb250LXdlaWdodD0iYm9sZCIgZmls
bD0iIzY0NzQ4YiI+VFJFTkRTPC90ZXh0PgoKICA8IS0tIEF2YWlsYWJpbGl0eSBUcmVuZCAtLT4KI
CA8cmVjdCB4PSI1MCIgeT0iMjA1IiB3aWR0aD0iMzQ1IiBoZWlnaHQ9IjY1IiByeD0iNiIgZmlsbD
0iI2YxZjVmOSIgc3Ryb2tlPSIjZTJlOGYwIiBzdHJva2Utd2lkdGg9IjEiLz4KICA8dGV4dCB4PSI
2NSIgeT0iMjI1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAi
IGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjMzMzIj5BdmFpbGFiaWxpdHkgVHJlbmQgKDcgZGF5c
yk8L3RleHQ+CiAgPCEtLSBTcGFya2xpbmUgc2ltdWxhdGlvbiAtLT4KICA8cG9seWxpbmUgcG9pbn
RzPSI3MCwyNTUgMTIwLDI1MCAxNzAsMjUyIDIyMCwyNDggMjcwLDI1MSAzMjAsMjQ1IDM3MCwyNTA
iIHN0cm9rZT0iIzEwYjk4MSIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub25lIi8+CiAgPGxpbmUg
eDE9IjcwIiB5MT0iMjYwIiB4Mj0iMzcwIiB5Mj0iMjYwIiBzdHJva2U9IiNlMmU4ZjAiIHN0cm9rZ
S13aWR0aD0iMSIvPgoKICA8IS0tIFJlc3BvbnNlIFRpbWUgRGlzdHJpYnV0aW9uIC0tPgogIDxyZW
N0IHg9IjQxMCIgeT0iMjA1IiB3aWR0aD0iMzQ1IiBoZWlnaHQ9IjY1IiByeD0iNiIgZmlsbD0iI2Y
xZjVmOSIgc3Ryb2tlPSIjZTJlOGYwIiBzdHJva2Utd2lkdGg9IjEiLz4KICA8dGV4dCB4PSI0MjUi
IHk9IjIyNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmb
250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzMzMyI+UmVzcG9uc2UgVGltZSBieSBNb25pdG9yIFR5cG
U8L3RleHQ+CiAgPCEtLSBCYXIgY2hhcnQgc2ltdWxhdGlvbiAtLT4KICA8cmVjdCB4PSI0NDAiIHk
9IjI0MCIgd2lkdGg9IjYwIiBoZWlnaHQ9IjI1IiByeD0iMiIgZmlsbD0iIzNiODJmNiIvPgogIDxy
ZWN0IHg9IjUxMCIgeT0iMjQ4IiB3aWR0aD0iNjAiIGhlaWdodD0iMTciIHJ4PSIyIiBmaWxsPSIjM
TBiOTgxIi8+CiAgPHJlY3QgeD0iNTgwIiB5PSIyNTIiIHdpZHRoPSI2MCIgaGVpZ2h0PSIxMyIgcn
g9IjIiIGZpbGw9IiNmNTllMGIiLz4KICA8cmVjdCB4PSI2NTAiIHk9IjI0NSIgd2lkdGg9IjYwIiB
oZWlnaHQ9IjIwIiByeD0iMiIgZmlsbD0iIzhiNWNmNiIvPgoKICA8IS0tIFJvdyAzOiBUYWJsZSAt
LT4KICA8dGV4dCB4PSI1MCIgeT0iMjkwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiI
GZvbnQtc2l6ZT0iMTAiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjNjQ3NDhiIj5SRUNFTlQgRk
FJTFVSRVM8L3RleHQ+CgogIDxyZWN0IHg9IjUwIiB5PSIzMDAiIHdpZHRoPSI3MDUiIGhlaWdodD0
iMzUiIHJ4PSI0IiBmaWxsPSIjZmVmMmYyIiBzdHJva2U9IiNlZjQ0NDQiIHN0cm9rZS13aWR0aD0i
MSIvPgogIDx0ZXh0IHg9IjY1IiB5PSIzMTgiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZ
iIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyI+TW9uaXRvcjwvdGV4dD4KICA8dGV4dCB4PSIyMD
AiIHk9IjMxOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiB
maWxsPSIjMzMzIj5Mb2NhdGlvbjwvdGV4dD4KICA8dGV4dCB4PSIzMjAiIHk9IjMxOCIgZm9udC1m
YW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5FcnJvc
jwvdGV4dD4KICA8dGV4dCB4PSI1MDAiIHk9IjMxOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLX
NlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5UaW1lPC90ZXh0PgogIDx0ZXh0IHg9IjY
yMCIgeT0iMzE4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAi
IGZpbGw9IiMzMzMiPkR1cmF0aW9uPC90ZXh0PgogIDx0ZXh0IHg9IjcwMCIgeT0iMzE4IiBmb250L

WZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPlN0YX
R1czwvdGV4dD4KCiAgPHRleHQgeD0iNjUiIHk9IjMyOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5
zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjQ3NDhiIj5jaGVja291dC1mbG93PC90ZXh0
PgogIDx0ZXh0IHg9IjIwMCIgeT0iMzI4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiI
GZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPlVTLUVhc3Q8L3RleHQ+CiAgPHRleHQgeD0iMz
IwIiB5PSIzMjgiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCI
gZmlsbD0iIzY0NzQ4YiI+VGltZW91dCBhZnRlciA2MHM8L3RleHQ+CiAgPHRleHQgeD0iNTAwIiB5
PSIzMjgiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsb
D0iIzY0NzQ4YiI+MTA6NDUgQU08L3RleHQ+CiAgPHRleHQgeD0iNjIwIiB5PSIzMjgiIGZvbnQtZm
FtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY0NzQ4YiI+NSB
taW48L3RleHQ+CiAgPHJlY3QgeD0iNjk1IiB5PSIzMTUiIHdpZHRoPSI1MCIgaGVpZ2h0PSIxNiIg
cng9IjQiIGZpbGw9IiNlZjQ0NDQiLz4KICA8dGV4dCB4PSI3MjAiIHk9IjMyNyIgZm9udC1mYW1pb
Hk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbm
Nob3I9Im1pZGRsZSI+T1BFTjwvdGV4dD4KPC9zdmc+Cg==)

```dql
// Dashboard tile: Overall availability (single value)
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| summarize {
    availability_pct = round(countIf(synthetic.availability == true) * 100.0
/ count(), decimals: 2)
  }
```

```dql
// Dashboard tile: Active vs failed monitors
fetch bizevents, from: now() - 1h
| filter event.provider == "dynatrace.synthetic"
| summarize {
    latest_status = takeFirst(synthetic.availability)
  }, by: {dt.entity.synthetic_test}
| summarize {
    total_monitors = count(),
    healthy = countIf(latest_status == true),
    failing = countIf(latest_status == false)
  }
```

```dql
// Dashboard tile: Response time by monitor (bar chart)
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == true
| summarize {
    p95_response_ms = percentile(toDouble(synthetic.response_time), 95)
  }, by: {dt.entity.synthetic_test}
```

```
| sort p95_response_ms desc
| limit 10
```

```dql
// Dashboard tile: Recent failures table
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| filter synthetic.availability == false
| fields timestamp,
         monitor = dt.entity.synthetic_test,
         location = dt.entity.synthetic_location,
         error = synthetic.error_message
| sort timestamp desc
| limit 20
```

```dql
// Dashboard tile: Hourly availability trend (line chart)
fetch bizevents, from: now() - 24h
| filter event.provider == "dynatrace.synthetic"
| fieldsAdd hour_bucket = bin(timestamp, 1h)
| summarize {
    success_count = countIf(synthetic.availability == true),
    total_executions = count(),
    failures = countIf(synthetic.availability == false)
  }, by: {hour_bucket}
| fieldsAdd availability_pct = round((success_count * 100.0) /
total_executions, decimals: 2)
| sort hour_bucket asc
```

Summary

In this notebook, you learned:

✅ **Availability analysis** - Calculating and trending availability
✅ **Performance analysis** - Percentiles, timing breakdown, anomalies
✅ **Location comparison** - Geographic performance analysis
✅ **SLO configuration** - Service level objectives and error budgets
✅ **Alerting strategies** - Outage detection, performance alerts
✅ **Dashboard building** - Key tiles and layouts

Series Complete!

```
You have completed the Synthetic Monitoring Best Practices series:

1. ✅ **SYNTH-01**: Fundamentals
2. ✅ **SYNTH-02**: Browser Monitors
3. ✅ **SYNTH-03**: HTTP Monitors
4. ✅ **SYNTH-04**: Private Locations
5. ✅ **SYNTH-05**: Network Monitoring
6. ✅ **SYNTH-06**: Analytics & Alerting

---

## References

- [Synthetic Monitoring Overview](https://docs.dynatrace.com/docs/platform-modules/digital-experience/synthetic-monitoring)
- [Analyze Synthetic Monitors](https://docs.dynatrace.com/docs/platform-modules/digital-experience/synthetic-monitoring/analysis-and-alerting/analyze-synthetic-monitors)
- [Service Level Objectives](https://docs.dynatrace.com/docs/platform-modules/automations/service-level-objectives)
- [Alerting Profiles](https://docs.dynatrace.com/docs/platform-modules/automations/alerting)
- [Dashboards](https://docs.dynatrace.com/docs/observe-and-explore/dashboards-and-notebooks/dashboards-new)
```