Visual Studio LIVE! | AUSTIN
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

I don't feel so well

Integrating health checks in your .NET solutions

Alex Thissen
Cloud architect

@alexthissen

Xpirit
Think ahead. Act now.

MVP Microsoft® Most Valuable Professional

Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for “Converge360 Events” in your app store
- Find this session on the Agenda tab
- Click “Session Evaluation”
- Thank you!

Challenges for large-scale distributed systems

Keeping entire system running

Determine state of entire system and intervene

How to know health status of individual services?

Collecting/correlating performance and health data

Events, metrics, telemetry, logs, traces

Usually centralized in a distributed landscape, e.g. micro-services


Raygun.io


NewRelic

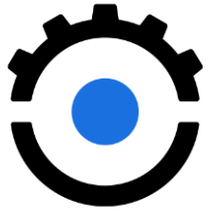

AlertSite


AppMetrics


Azure Monitor


DataDog


Sentry.io


Runscope

Traditional medicine and health

Centralized

Single point that knows how to assess health

Challenging

Combining measurements to health information

Based on generic types of measured values

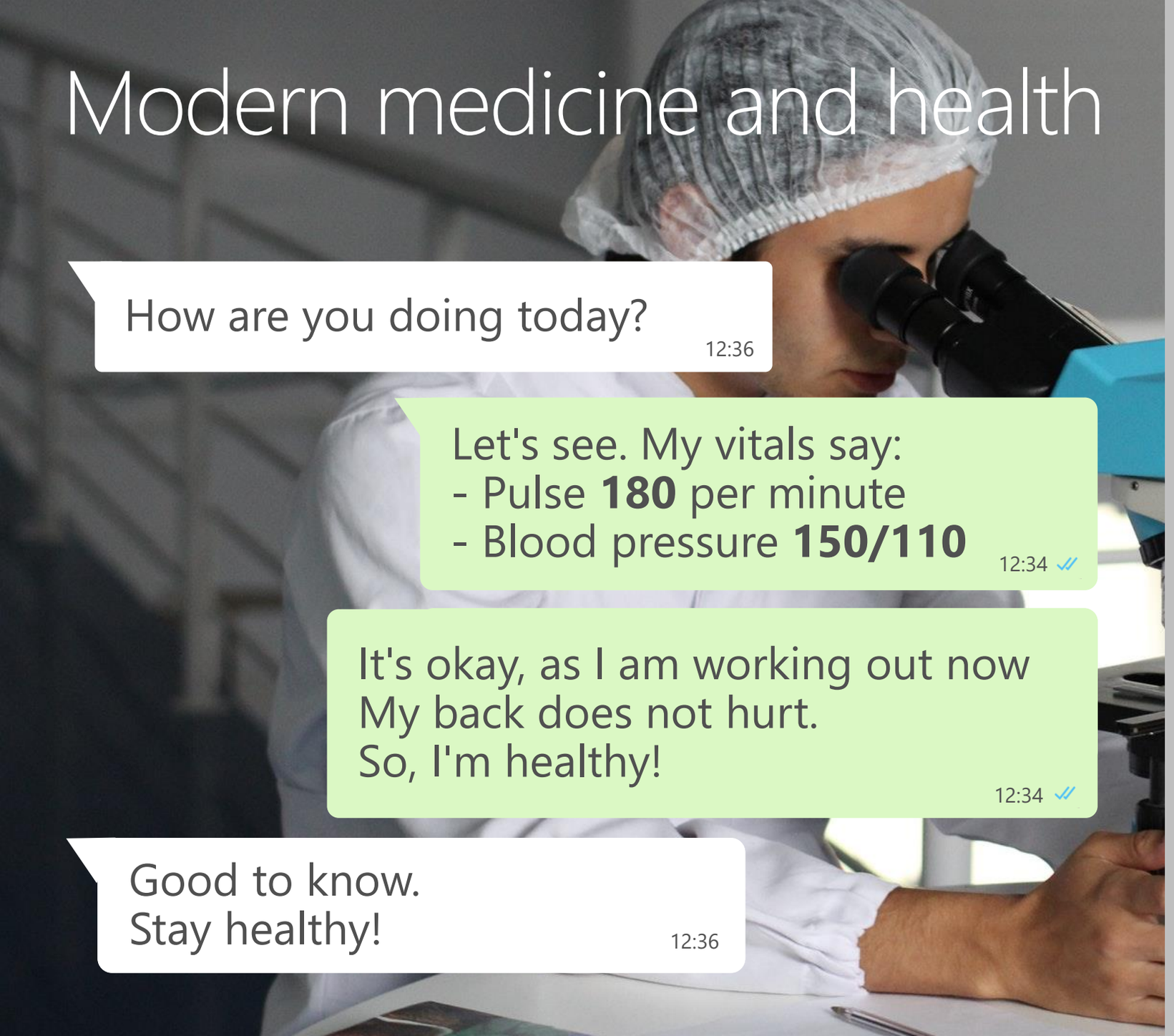
Absence of measurements

Differences in behavior from person to person

Unknown internals

Multiple places to access health

Modern medicine and health



Self-assessment

Determing your own health status

Know what defines healthy and unhealthy

Context matters

Measurements might need to be interpreted differently

Depending on:

- Situation
- Circumstances
- Unmeasurable values

You know best

Difference between metrics and health info

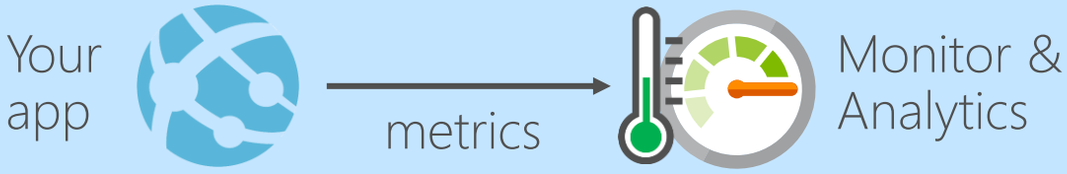
Metrics

Many individual measured values and counts of events

Watch performance and trends

Useful for diagnostics and troubleshooting

Logic external to origin



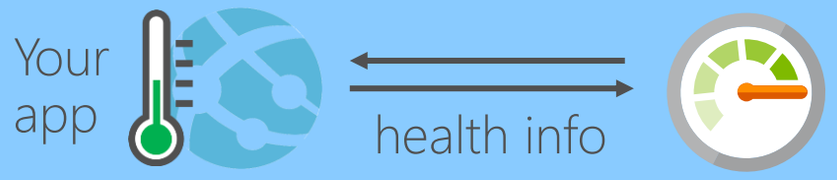
Health

Intrinsic knowledge of implementation required

DevOps mindset:

Logic to determine health is part of origin

Deployed together, good for autonomy



Levels of health

Simple

Availability

Any response
Status code indication
Formal endpoints

Latency

Time to respond

Internals

Memory
Disk space

Advanced

External dependencies

- URL endpoints (e.g. Web API or CDN)
- Databases
- Service bus or queue
- Storage

Readiness & liveliness

Distinguish startup and normal operation
Good for external lifetime management

Preventive

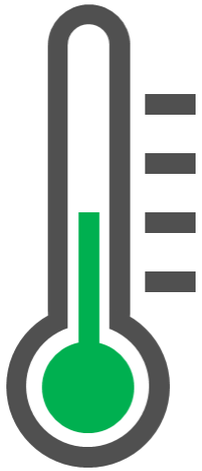
Predicting

- Indication of impending failure
- Interesting with AI and ML

Examples

- Expiring certificates
- Trends in memory pressure
- Failing resiliency countermeasures

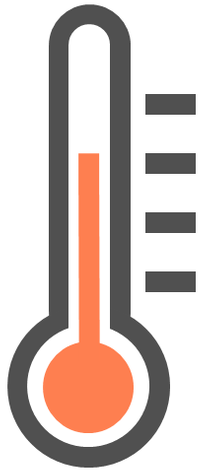
Health status



Healthy

200 OK

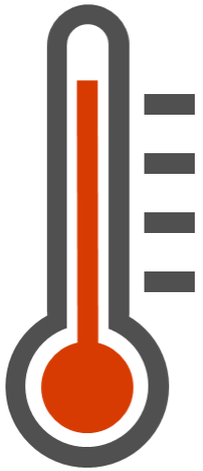
"Everything is fine"



Degraded

200 OK

"Could be doing better or about to become unhealthy"



Unhealthy

503 Service Unavailable

"Not able to perform"

Integrating health checks

New since .NET Core 2.2

Available to all .NET applications
Plugs deep into ASP.NET Core

Microsoft.Extensions.Diagnostics.HealthChecks
.Abstractions
.EntityFramework

Microsoft.AspNetCore.Diagnostics.HealthChecks

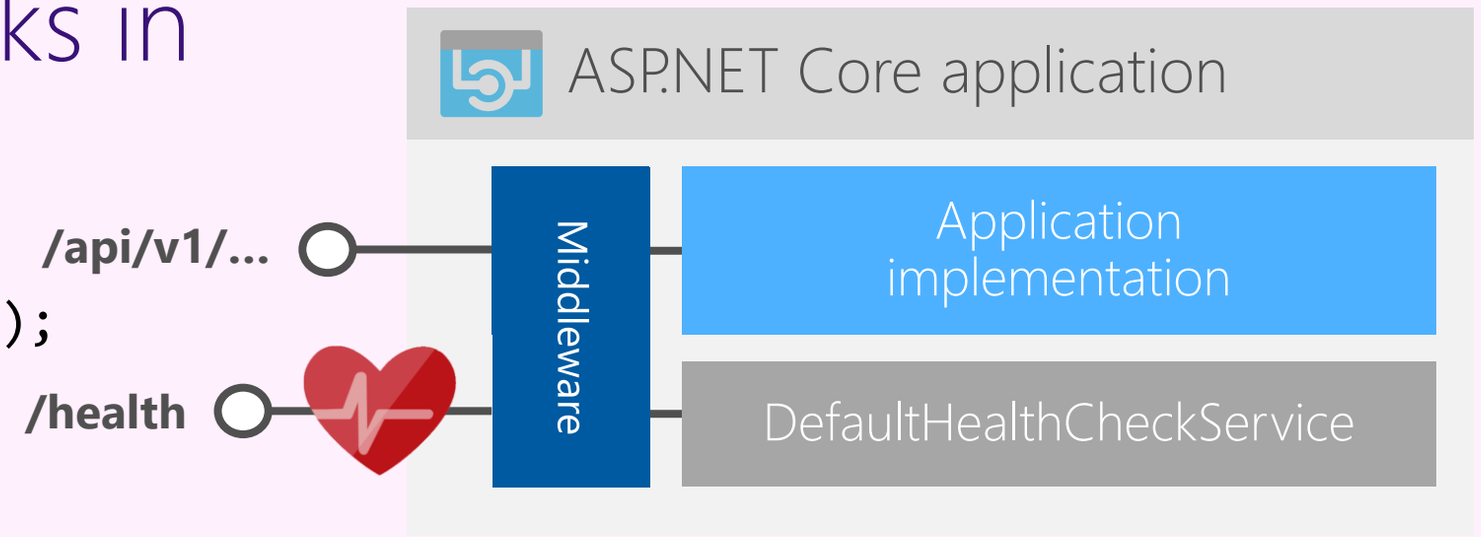
Bootstrap health checks in ASP.NET Core app

Dependency injection

```
builder.Services.AddHealthChecks();
```

ASP.NET Core middleware routing

```
app.MapHealthChecks("/health");
```



Using health checks

What?

```
public interface IHealthCheck
{
  Task<HealthCheckResult> CheckHealthAsync(
    HealthCheckContext context,
    CancellationToken cancellationToken = default);
}
```

When?

On demand from endpoints

Periodically by publishers

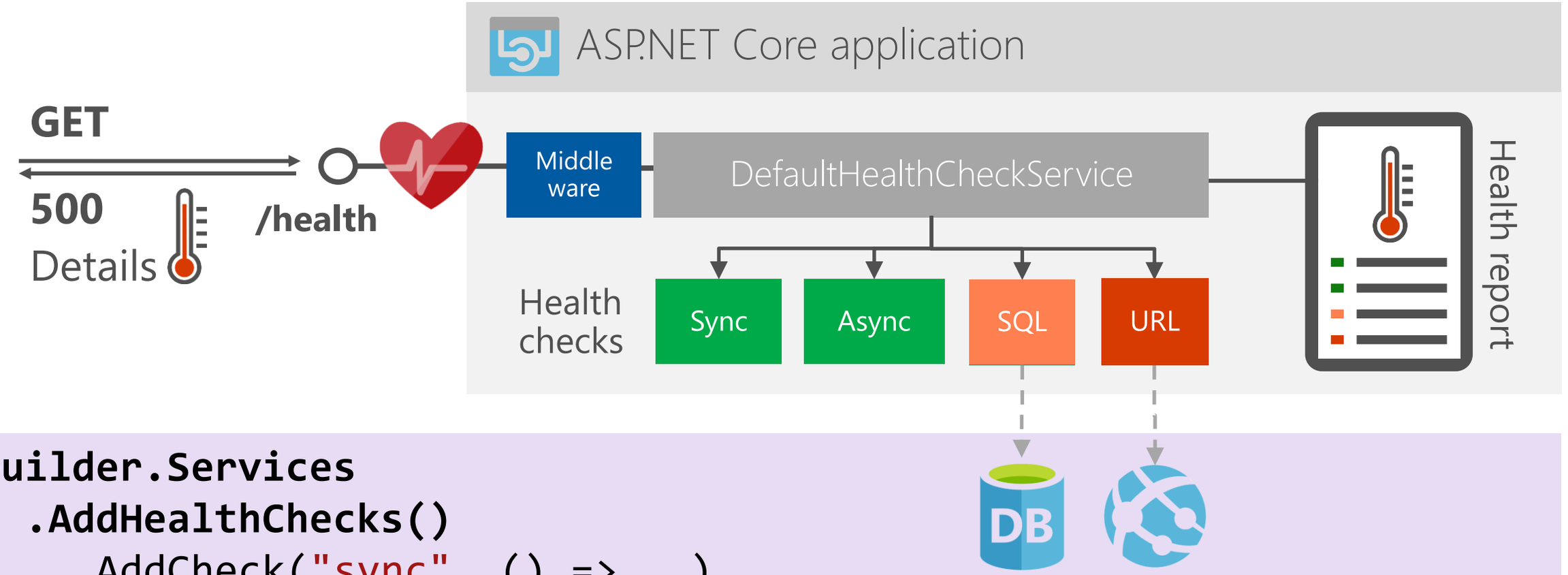
How?

Iterating over health check registrations

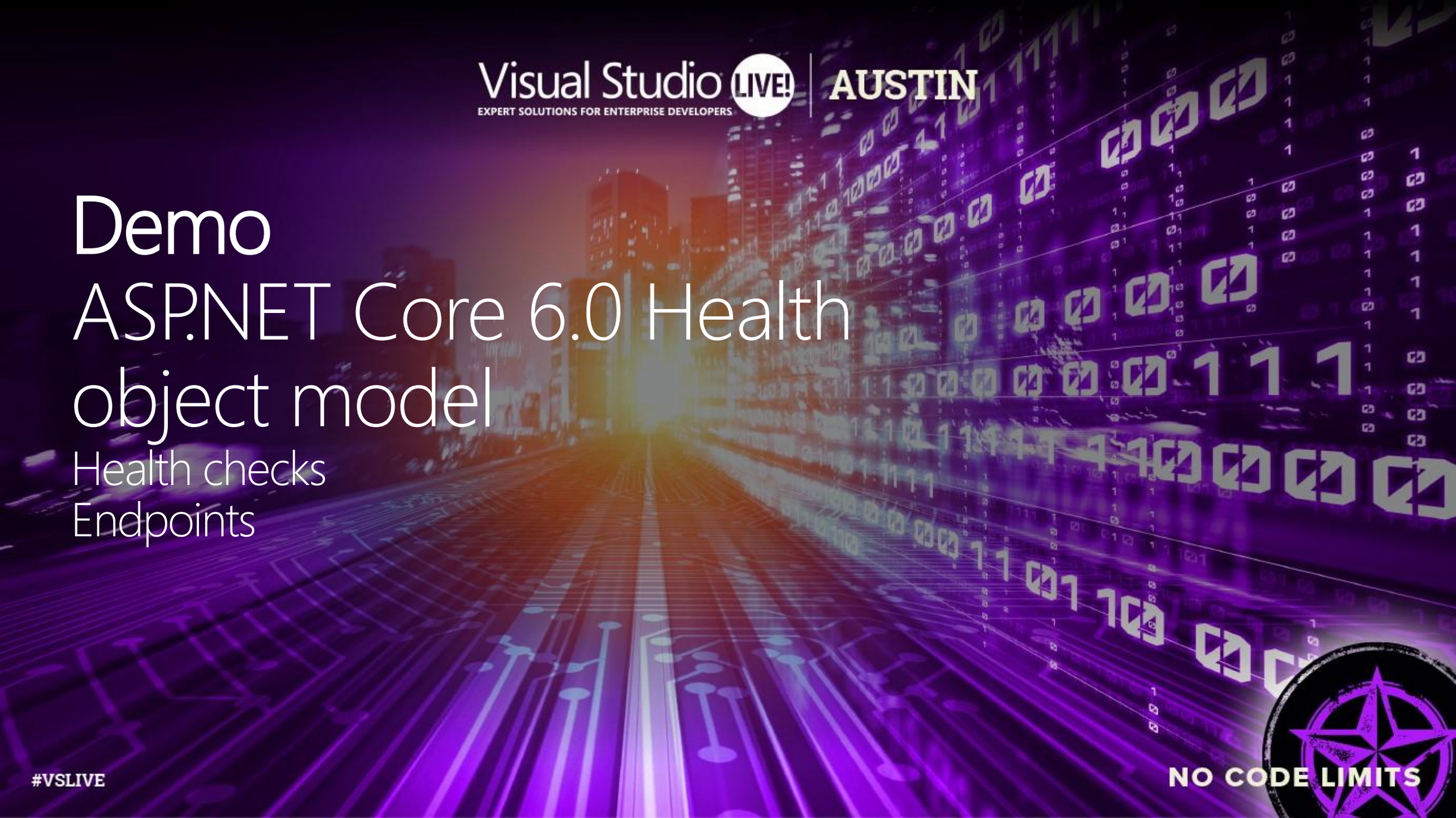
```
if (currentValue == HealthStatus.Failed)
{
    // Game over, man! Game over!
    // (We hit the worst possible status, so
    return currentValue;
}
```

From: https://github.com/aspnet/Diagnostics/blob/master/src/Microsoft.Extensions.Diagnostics.HealthChecks.Abstractions/HealthReport.cs

Integrating health checks




```
builder.Services
   .AddHealthChecks()
      .AddCheck("sync", () => … )
      .AddAsyncCheck("async", async () => … )
      .AddCheck<SqlConnectionHealthCheck>("SQL")
      .AddCheck<UrlHealthCheck>("URL");
```


Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
AUSTIN
Demo
ASP.NET Core 6.0 Health object model
Health checks
Endpoints
#VSLIVE
NO CODE LIMITS

Custom health checks

Only 1 out-of-box check

Entity Framework `DbContext`

- Microsoft.Extensions.Diagnostics.
  HealthChecks.EntityFrameworkCore

```
services.AddHealthChecks()
    .AddDbContextCheck<GamingDbContext>("EF")
```

Build your own

1. Delegate for sync or async factory
2. Implementation of `IHealthCheck`

Community packages

- AspNetCore.Diagnostics.HealthChecks.*

Xabaril/BeatPulse

System (Disk Storage, Memory)

Network (Tcp, Ftp, Sftp, Imap, Smtp, Dns resolve)

Azure Storage (Blobs, Tables and Queues)

Azure Service Bus (Event Hub, Service Bus queues and topics)

RabbitMQ

Kafka

Redis

Elasticsearch

EventStore

Identity Server

AWS DynamoDB

SqlServer

MongoDb

Oracle

DocumentDb

MySQL

SqLite

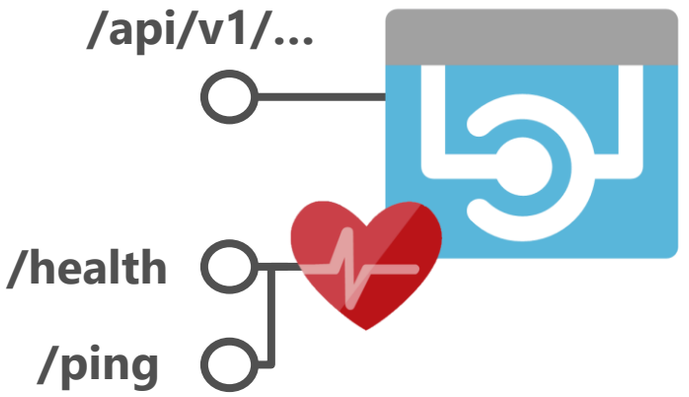
Postgress Sql

Yours?

Beyond the basics

Register multiple health endpoints

Order of registrations matters



Middleware options

Change HTTP status codes per health result
Allow client-side caching
Change response writing
Predicate for filtering health checks to evaluate

Register custom health check as singleton

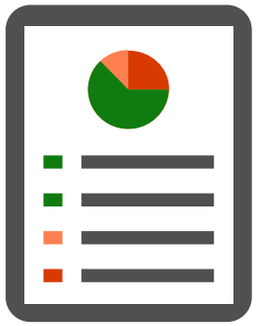
```
builder.Services.AddSingleton<KafkaHealthCheck>());
builder.Services.AddSingleton(new SqlConnectionHealthCheck(
  new SqlConnection(Configuration.GetConnectionString("TestDB"))));
```

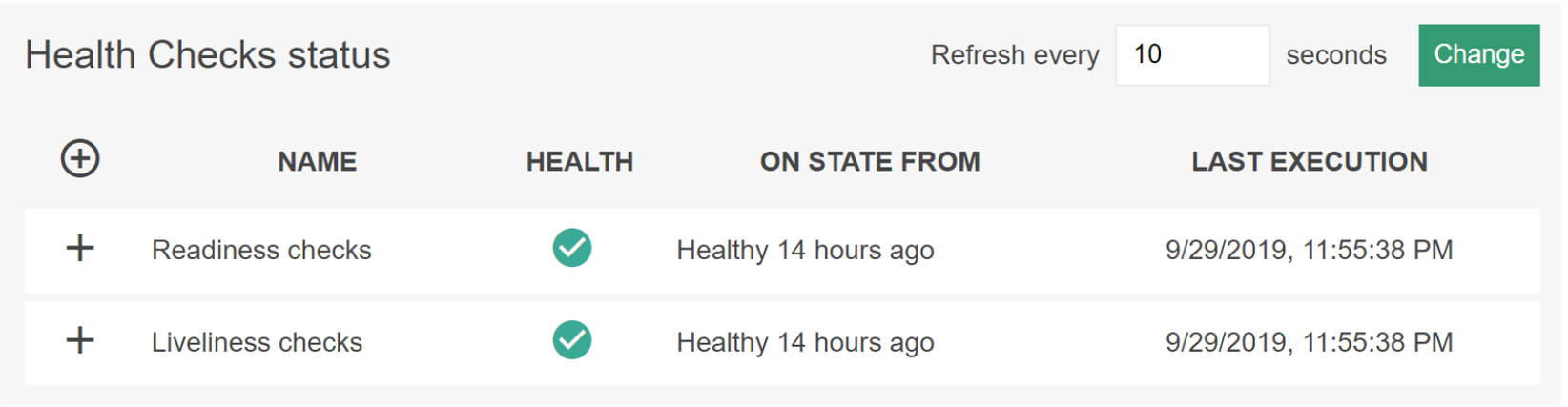
Visualizing health checks

1. Customize health endpoint output for more details
   Specify delegate from HealthCheckOptions.ResponseWriter
2. Query endpoint(s)
3. Build user interface

Xabaril BeatPulse AspNetCore.HealthChecks.UI

- Host in ASP.NET Core application
- Run from Docker container






Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
AUSTIN
Demo
A bit more advanced healthchecks
#VSLIVE
NO CODE LIMITS

Monitoring health

Endpoints

Frequency

Locations

Alerts

AVAILABILITY TEST	20 MIN	AVAILABILITY
Overall	**0.00%**	**0.00%**
Retro Gaming Web API Health check	0.00%	0.00%
Central US	0.00%	0.00%
East US	0.00%	0.00%
North Central US	0.00%	0.00%
South Central US	0.00%	0.00%
West US	0.00%	0.00%

Alert activated 9:31 AM

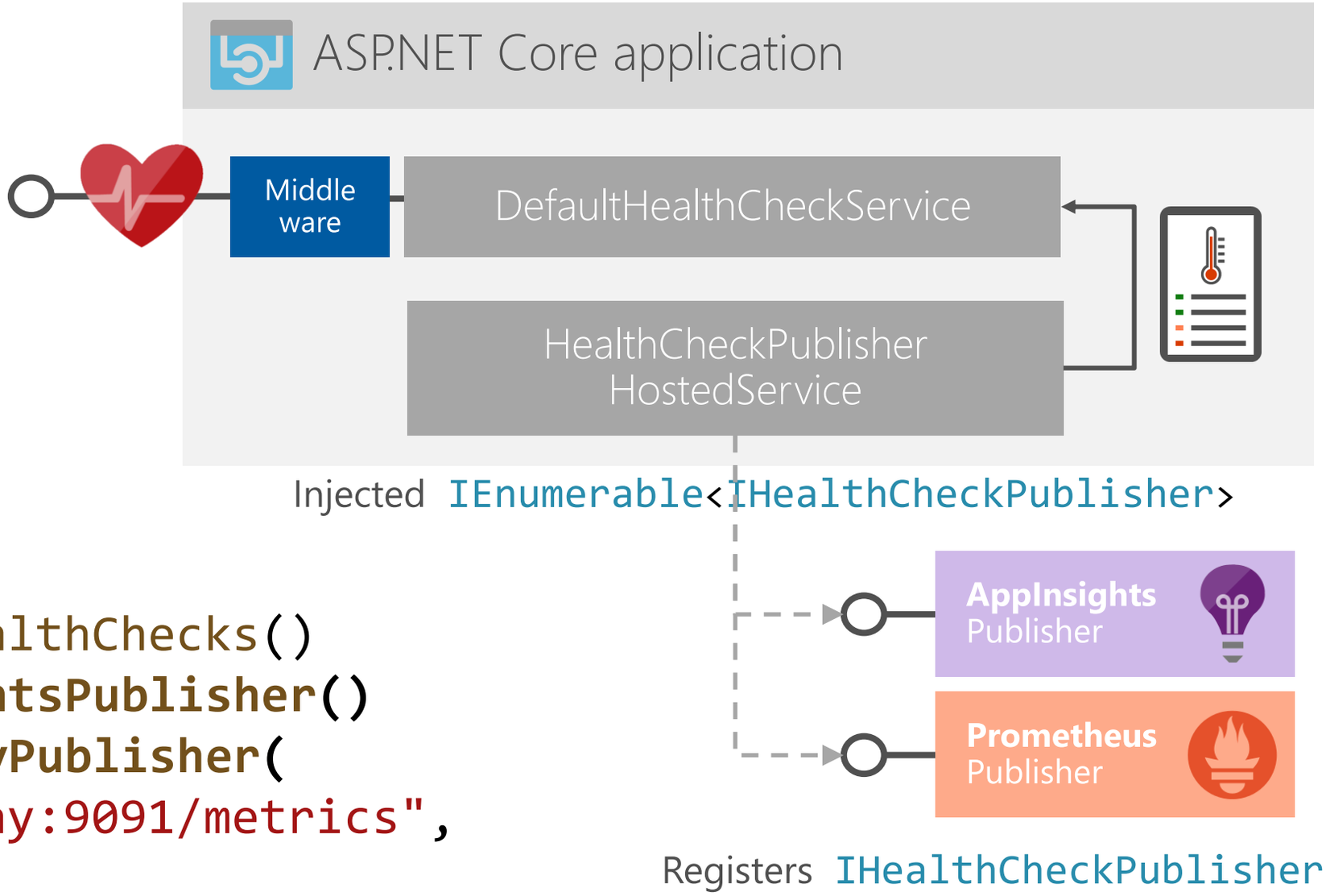
RetroGaming2019ApplicationInsights: availability test retro gaming web api health check-retrogaming2019applicationinsights crossed the configured threshold of failed locations

Health check publishers

Pushes out health info periodically

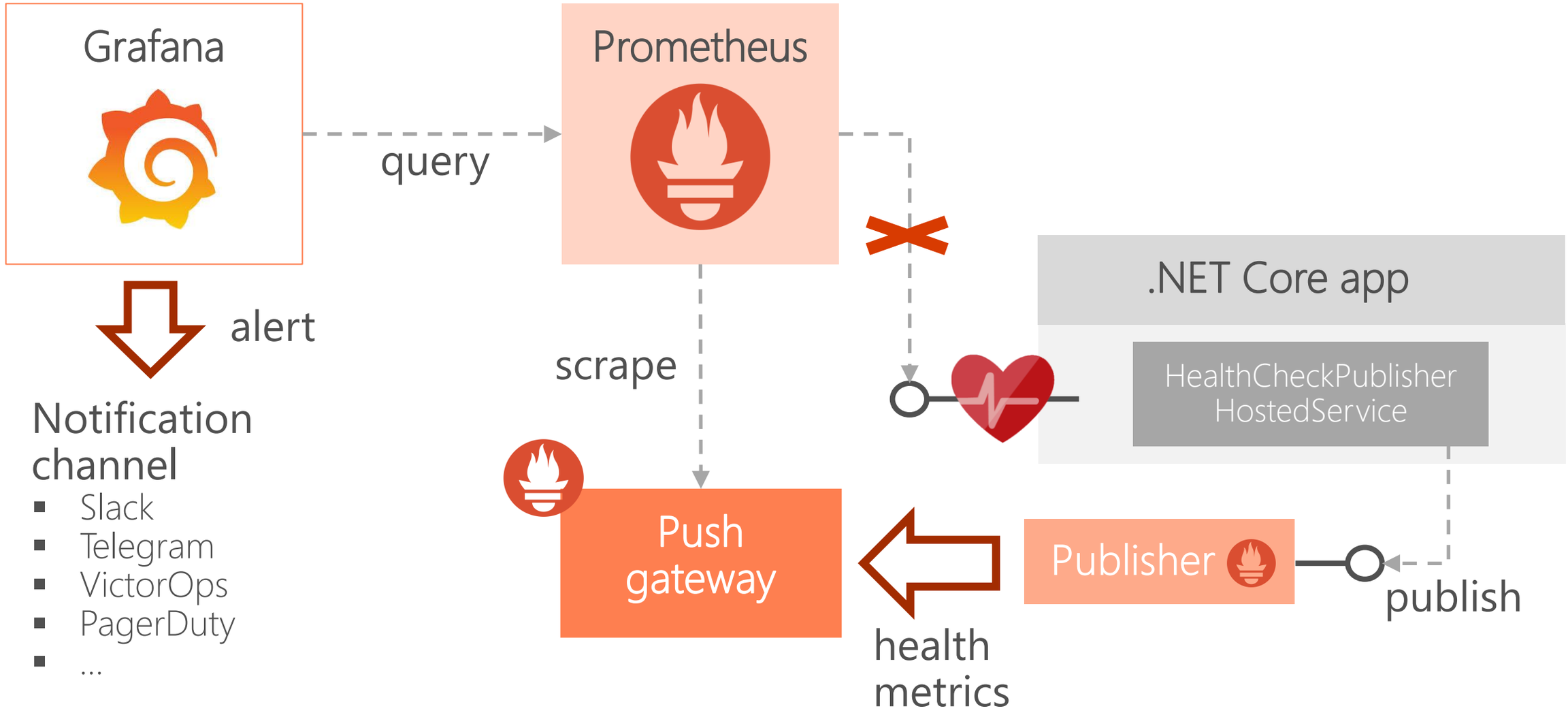
Options

Timeout: max time to execute check
Delay: time to wait after startup
Period: period of execution
Predicate: Filter for checks to execute




```
builder.Services.AddHealthChecks()
   .AddApplicationInsightsPublisher()
   .AddPrometheusGatewayPublisher(
      "http://pushgateway:9091/metrics",
      "pushgateway")
```

Prometheus and Grafana




Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
AUSTIN
Demo
Health publishers
Prometheus
Grafana
#VSLIVE
NO CODE LIMITS

Resilient and self-healing applications

Resiliency

Use cloud patterns:

- Circuit Breaker
- Timeout
- Retry

Performance

Metrics

Instrumentation

Availability

Zero-downtime upgrades

Readiness

Liveliness

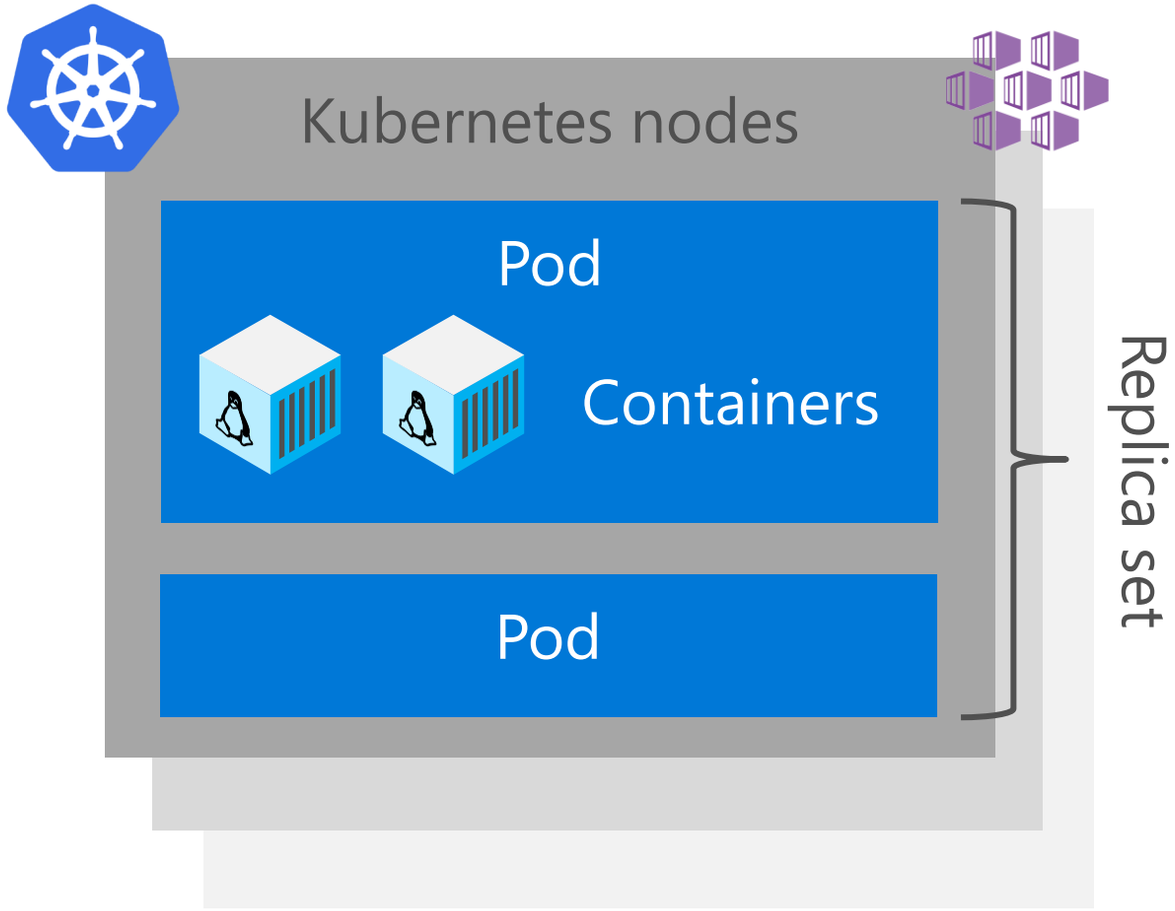
Monitoring

Health endpoint monitoring

Alerts

Readiness and liveness

Probing containers to check for availability and health



k8s-deployment.yaml

```yaml
readinessProbe:
  httpGet:
    path: /health/ready
    port: 8080
  initialDelaySeconds: 20
  periodSeconds: 10
  timeoutSeconds: 10
  failureThreshold: 3

livenessProbe:
  httpGet:
    path: /health/lively
    port: 8080
```

Readiness

Ready to receive incoming traffic

Not ready: remove container from load balancer

Liveliness

Indicates when to restart a container

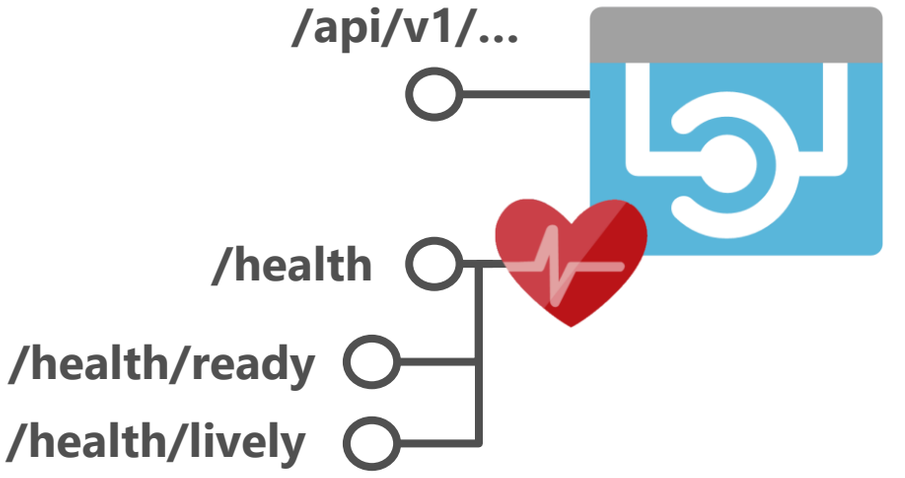
Implementing readiness and liveliness

1. Add health checks with tags

```
services.AddHealthChecks()
  .AddCheck<CircuitBreakerHealthCheck>(
    "circuitbreakers",
    tags: new string[] { "ready" });
```

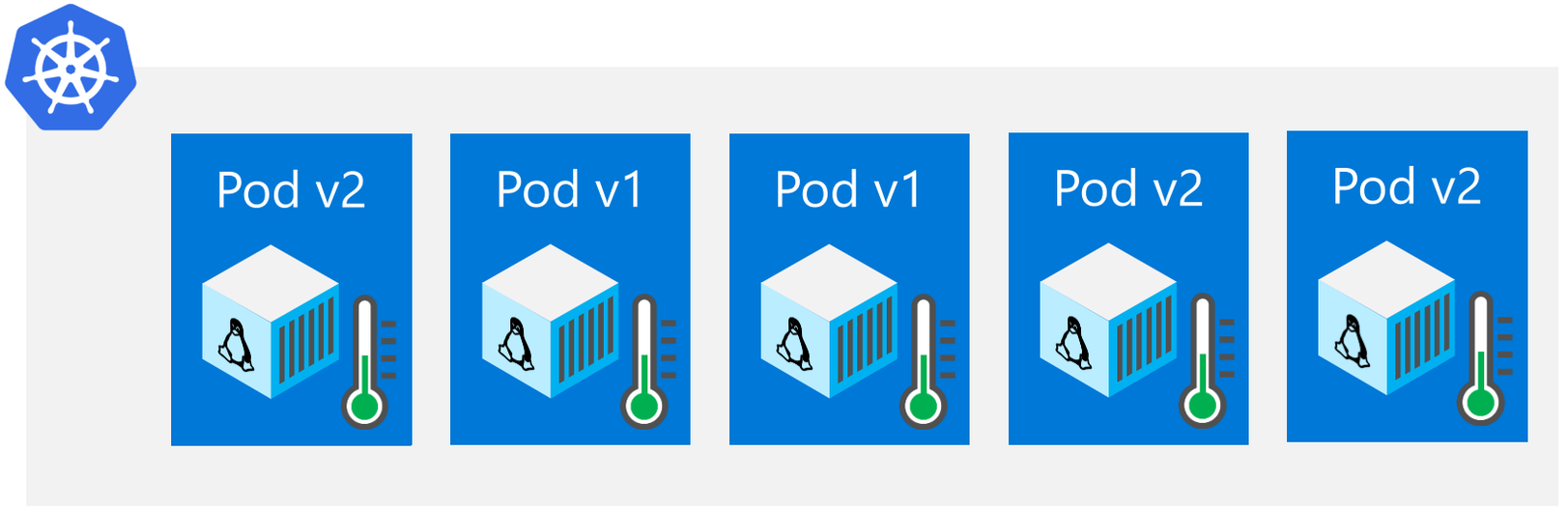
2. Register multiple endpoints with filter using Options predicate

```
app.UseHealthChecks("/health/ready",
  new HealthCheckOptions() {
    Predicate = reg => reg.Tags.Contains("ready")
});
```



Remember:
Order of registration matters

Zero downtime deployments




```
spec:
  replicas: 3
  revisionHistoryLimit: 0
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2
      maxUnavailable: 0
```

Original pods only taken offline after new healthy one is up

Allows roll forward upgrades: Never roll back to previous version


Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
AUSTIN
Demo
Readiness and liveliness probes
Docker containers
Kubernetes
#VSLIVE
NO CODE LIMITS

Securing

Expose as little detail as possible

Use different internal port

Inside a cluster ports are not exposed by default

Leverage notion of a management port

Add authentication using middleware

```
app.UseWhen(
  ctx => ctx.User.Identity.IsAuthenticated,
  a => a.UseHealthChecks("/securehealth")
);
```

Publish instead of endpoint

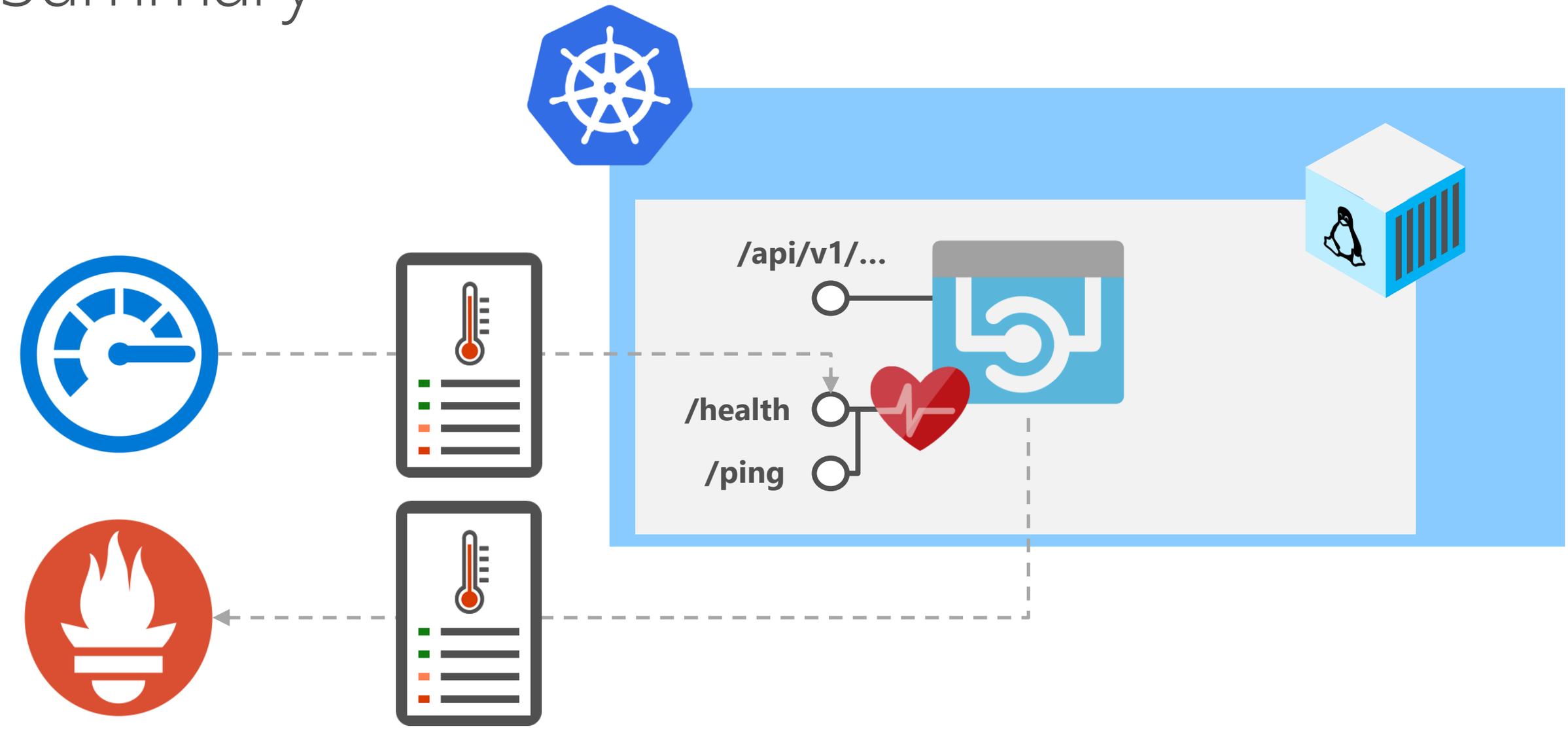
Best practices

1. Assume degraded state
2. Set short timeouts on checks

Inside health checks and for publishers
For example, when connecting to external dependencies

3. Avoid complicated health checks
4. Register health check as singletons in DI

Summary



Questions and Answers

Maybe later?

@alexthissen
athissen@xpirit.com

Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for “Converge360 Events” in your app store
- Find this session on the Agenda tab
- Click “Session Evaluation”
- Thank you!

Resources

ASP.NET Core Health monitoring

https://docs.microsoft.com/en-us/azure/architecture/patterns/health-endpoint-monitoring
https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/health-checks
https://github.com/aspnet/Diagnostics/tree/master/src

Kubernetes

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/

BeatPulse Xalabril

https://github.com/Xabaril/AspNetCore.Diagnostics.HealthChecks

Demo source code

https://github.com/alexthissen/healthmonitoring