Når millisekunder teller

Hva det kreves av Yr for å levere værdata for hele verden

Hva skjer hos Yr og hva er egentlig en utfordning

8-10 000 000 / uke

3 000 000 /dag

400 000 /time (08:00 – 10:00)

107 000 rpm

Kilde: GA og NewRelic

Hvor ligger grensen mellom premature optimisation og det man må gjøre?

Hva er egentlig god nok responstid?

To problematiske områder

Geografisk søk

- Begrenset mulighet for cache
- Mange kall

Importen av data

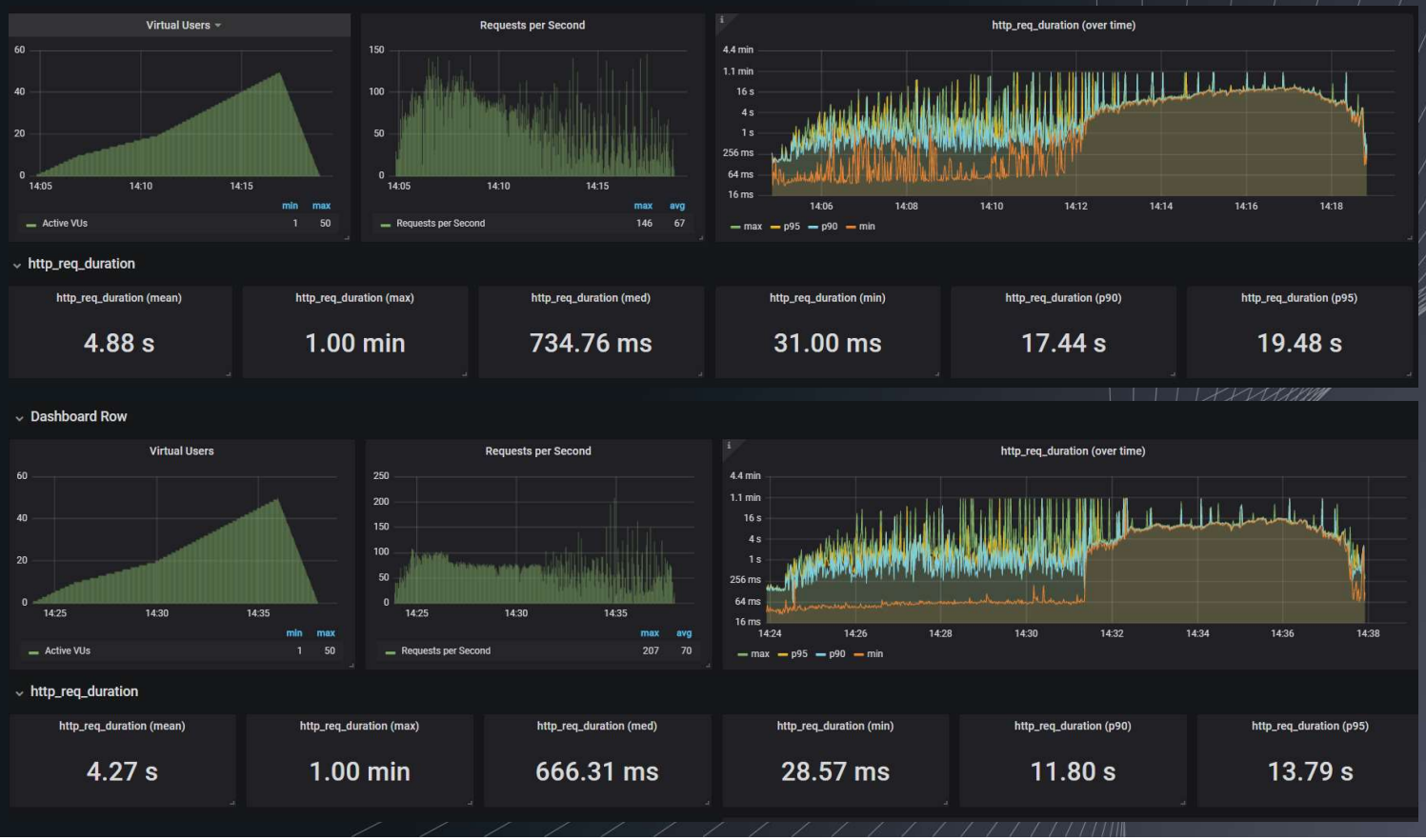
- Ca 13 000 000 objekter
- Masse metadata
- Endres hele tiden

1. Få mest mulig fra API

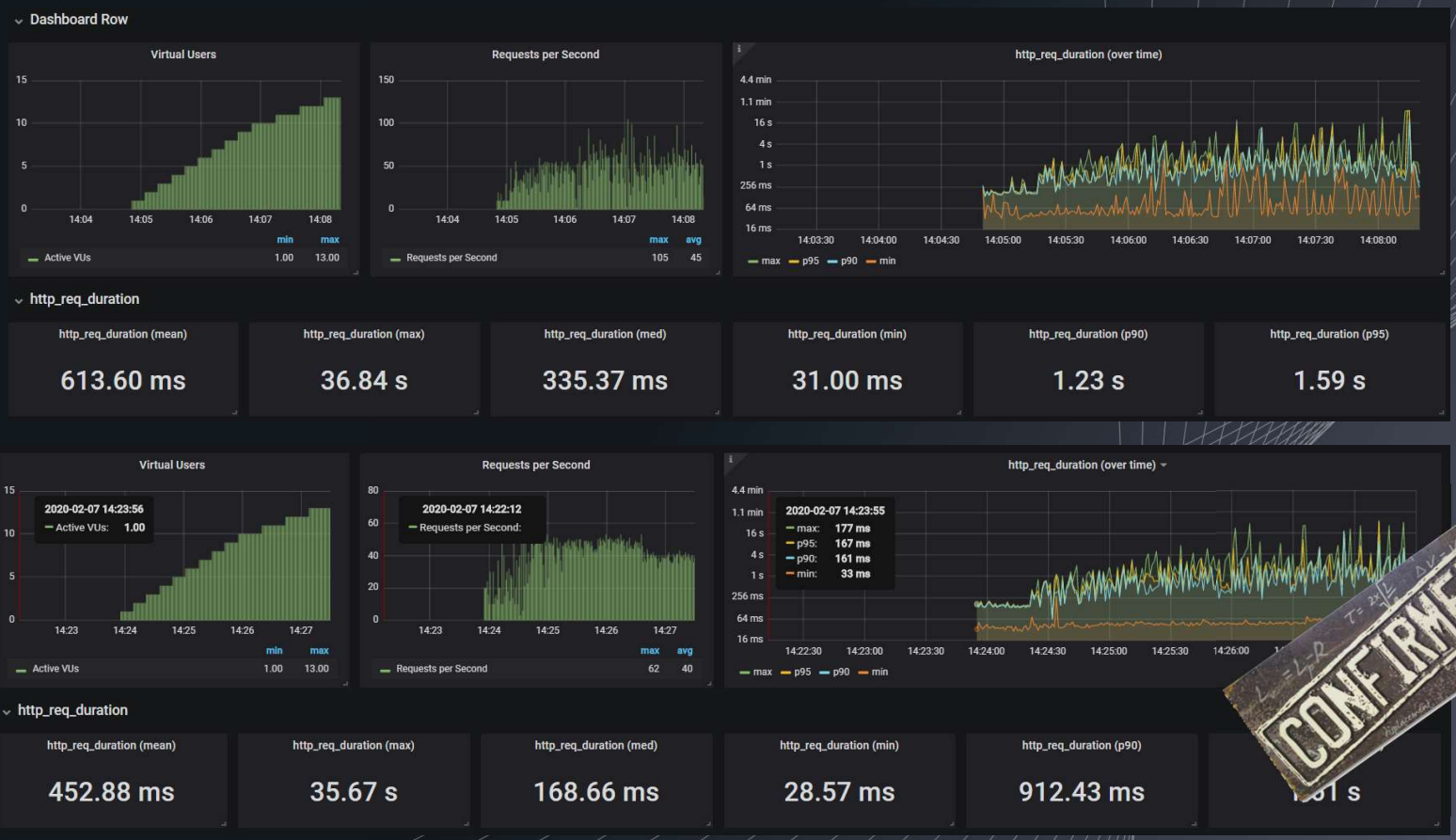
Hypoteser

- .Net Core er kjappere enn .Net, og .Net Core 3.* er kjappere enn 2.*
- System.Json er kjappere enn Newtonsoft
- Linux er bedre enn Windows

Er .Net Core 3.* bedre enn 2.*?



Er .Net Core 3.* bedre enn 2.*?

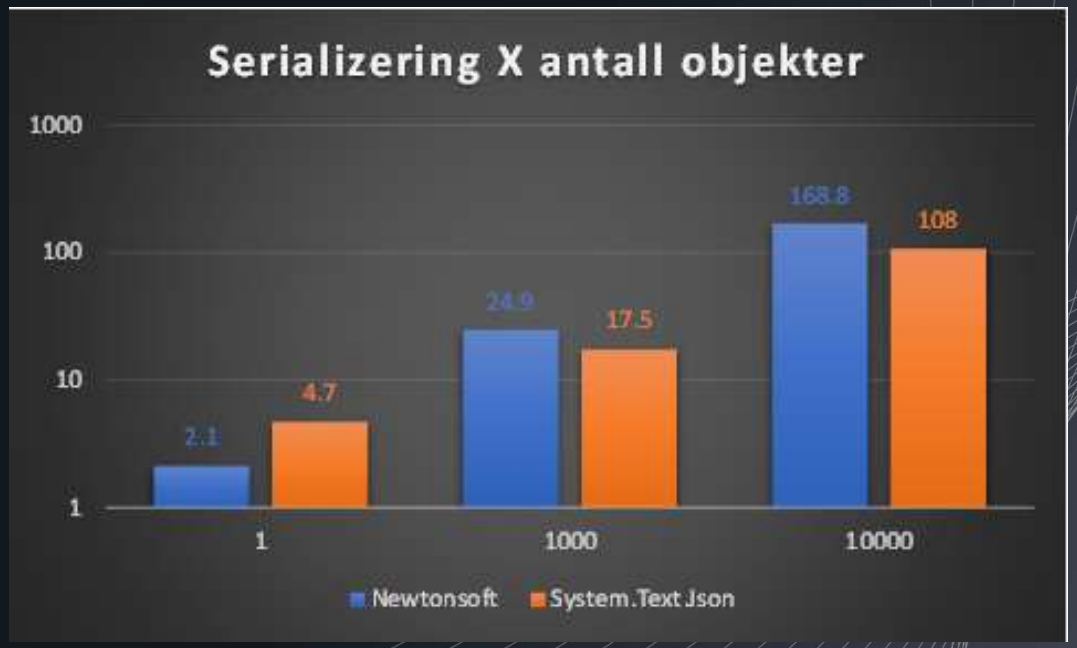


Er .Net Core 3 Json bedre enn Newtonsoft?

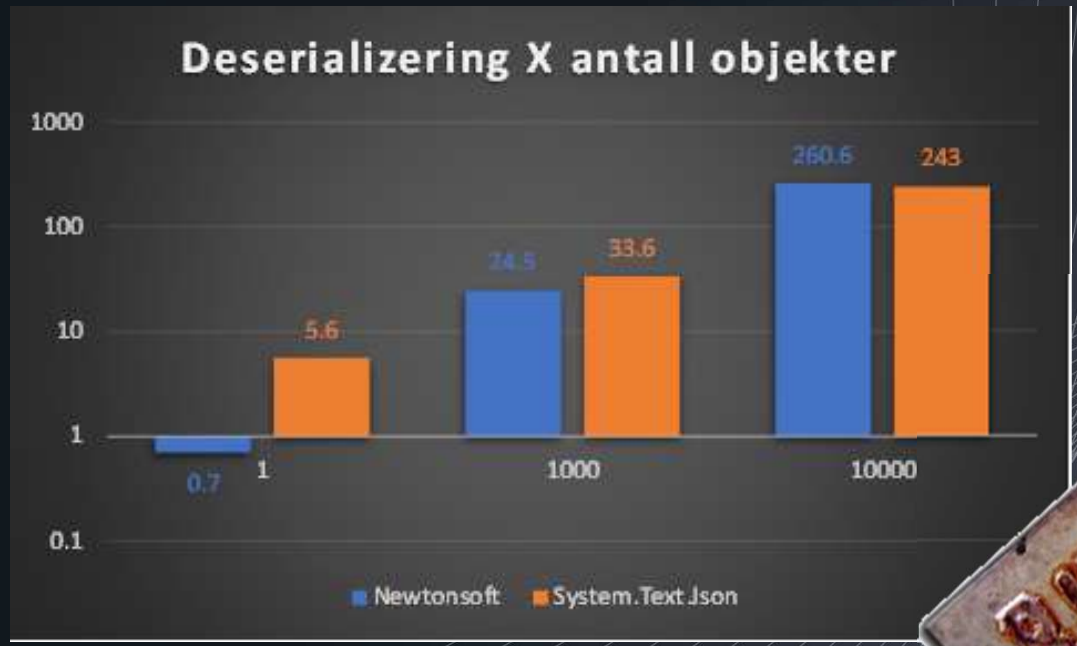
Test objekt:

```
public class LocationBase
{

    5 references | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public int Id { get; set; }
    1 reference | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public string RegionId { get; set; }
    1 reference | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public string CategoryId { get; set; }

    1 reference | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public double Lat { get; set; }
    1 reference | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public double Lon { get; set; }
    1 reference | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public long Altitude { get; set; }
    1 reference | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public int AltitudeType { get; set; }
    0 references | Dmitry Konovalov, 81 days ago | 1 author, 1 change
    public string ExternalData { get; set; }

    1 reference | 0 changes | 0 authors, 0 changes
    public string Timezone { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string Geometry { get; set; }
}
```

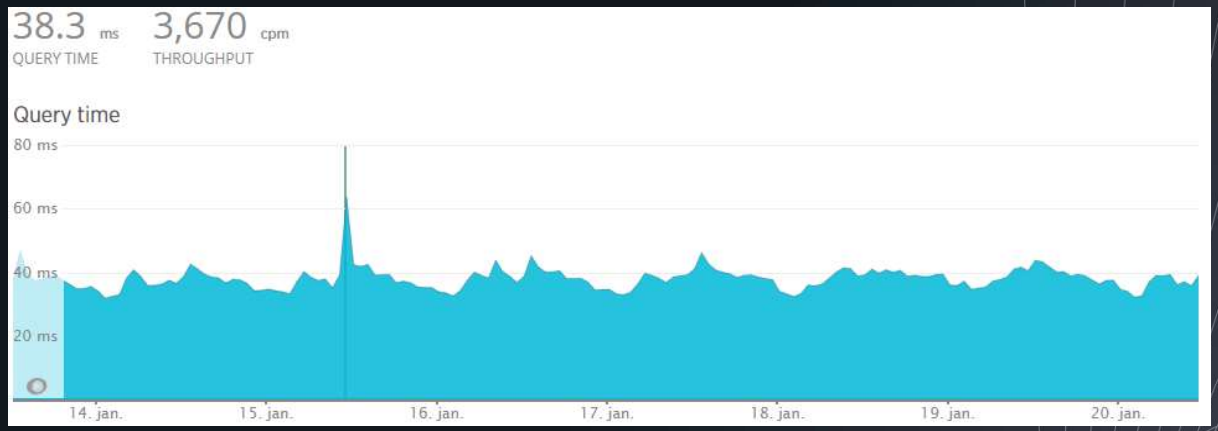
Er .Net Core 3 Json bedre enn Newtonsoft?

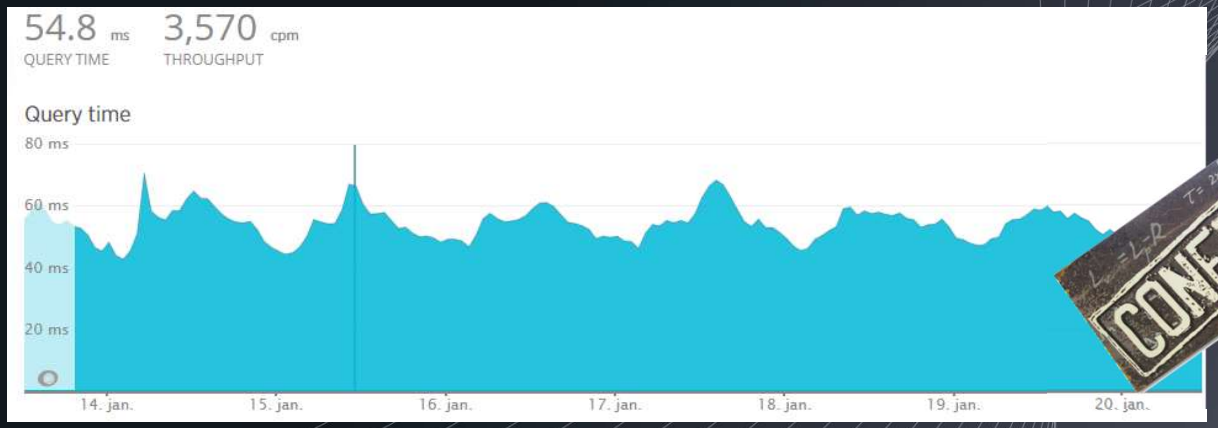


Er .Net Core 3 Json bedre enn Newtonsoft?



Er Linux bedre enn Windows?





2. DB relaterte optimaliseringer

Velg riktig database

```
SELECT *, st_distance(coordinates,
st_SetSrid(st_MakePoint(@lat, @lon), 4326)) AS distance,
FROM observationstation
WHERE distance < 1000
ORDER BY distance ASC
```

MS SQL

1000+ ms

Cosmos DB

300 ms

Postgre SQL

40-250 ms

Bruk riktige indekser og bygg riktige spørringer

- SELECT TOP 100 * FROM place WHERE id > 42
- SELECT TOP 100 *FROM place WHERE **St_intersects**(**St_geogfromtext**('SRID=4326; POLYGON(-179.9 0,0 0,0 85.06,-179.9 85.06,-179.9 0)'), coordinates)
  ORDER BY weight
- Velg bare den data du skal ha (ikke bruk SELECT * rundt omkring)
- Unngå felter som er vanskelig å indeksere og spørre (json i textkolonne osv)

Glem Entity Framework

Dapper skjult trobbel

```
public void UpdateRowsTest2(List<TestData> data)
{
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Execute(sql: "INSERT into test_data(Id, Name, Number) VALUES (@Id, @Name, @Number)", data);
    }
}
```

```
10:12:40 INF] Process started
10:12:44 INF] Dapper insert took 4285 ms
10:12:44 INF] Bulk insert took 174 ms
```

```
public void UpdateRowsTest1(List<TestData> data)
{
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();

        var bulkCopy = new SqlBulkCopy(connection);
        bulkCopy.DestinationTableName = "test_data";
        bulkCopy.BatchSize = data.Count;
        var copyParameters :string[] = new[]
        {
            nameof(TestData.Id),
            nameof(TestData.Number),
            nameof(TestData.Name)
        };

        using (var reader = ObjectReader.Create(data, copyParameters))
        {
            bulkCopy.WriteToServer(reader);
        }
    }
}
```

3. Algoritmiske feil

Linq og lesbarhet

Filtrer bort det som er allerede prosessert eller «Finn alle elementer i en list som eksisterer ikke i en annen»

```
var ids :IEnumerable<string> = await _importerService.GetSourceIdsForCountry(countrySpec.Key, CurrentSource);

foreach (var namesPortion :List<NameUpdateModel> in _reader.ReadAlternateNames( file: "alternateNamesV2"))
{
    var countrySpecific :List<NameUpdateModel> = namesPortion.Join(ids, name => name.SourceId, id :string => id, (name, id :string) => name).ToList();
```

380ms

```
var ids :IEnumerable<string> = await _importerService.GetSourceIdsForCountry(countrySpec.Key, CurrentSource);
var hash = new HashSet<string>(ids);

foreach (var namesPortion :List<NameUpdateModel> in _reader.ReadAlternateNames( file: "alternateNamesV2"))
{
    var countrySpecific :IEnumerable<NameUpdateModel> = namesPortion.Where(n :NameUpdateModel => hash.Contains(n.SourceId));
```

10ms

Linq og lesbarhet

Noen kanskje liker ikke kode som dette (DRY, osv):

```
entries = new[]
{
    new DbValueEntry {Entry = place.Status, Name = "Status", Type = NpgsqlDbType.Smallint},
    new DbValueEntry {Entry = place.Altitude, Name = "Altitude", Type = NpgsqlDbType.Integer},
    new DbValueEntry {Entry = place.Name, Name = "Name", Type = NpgsqlDbType.Varchar}
};
```

1ms

Da ble det sånn:

```
var entries = new []
{
    new DbValueEntry(expression:value :dynamic => place.Status),
    new DbValueEntry(expression:value :dynamic => place.Altitude),
    new DbValueEntry(expression:p :dynamic => place.Name),
};
```

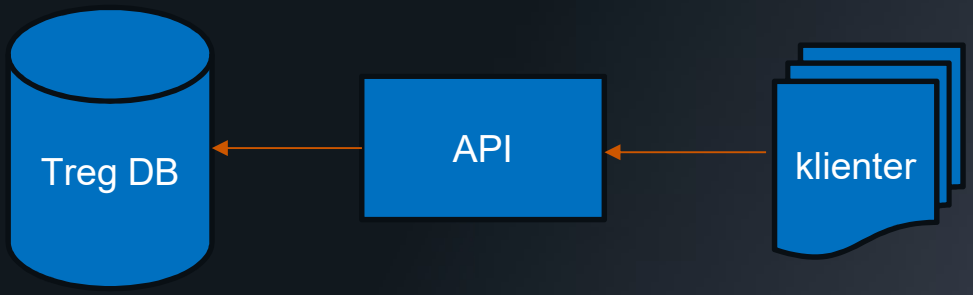
139ms

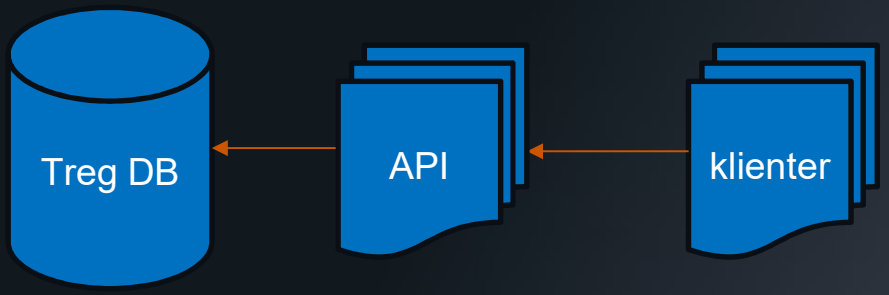
Reflection is evil

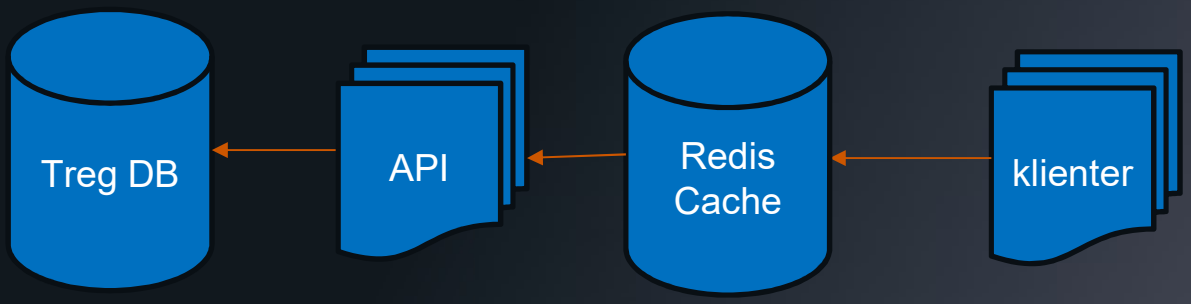
Dette står inni i det godt lesbart metode:

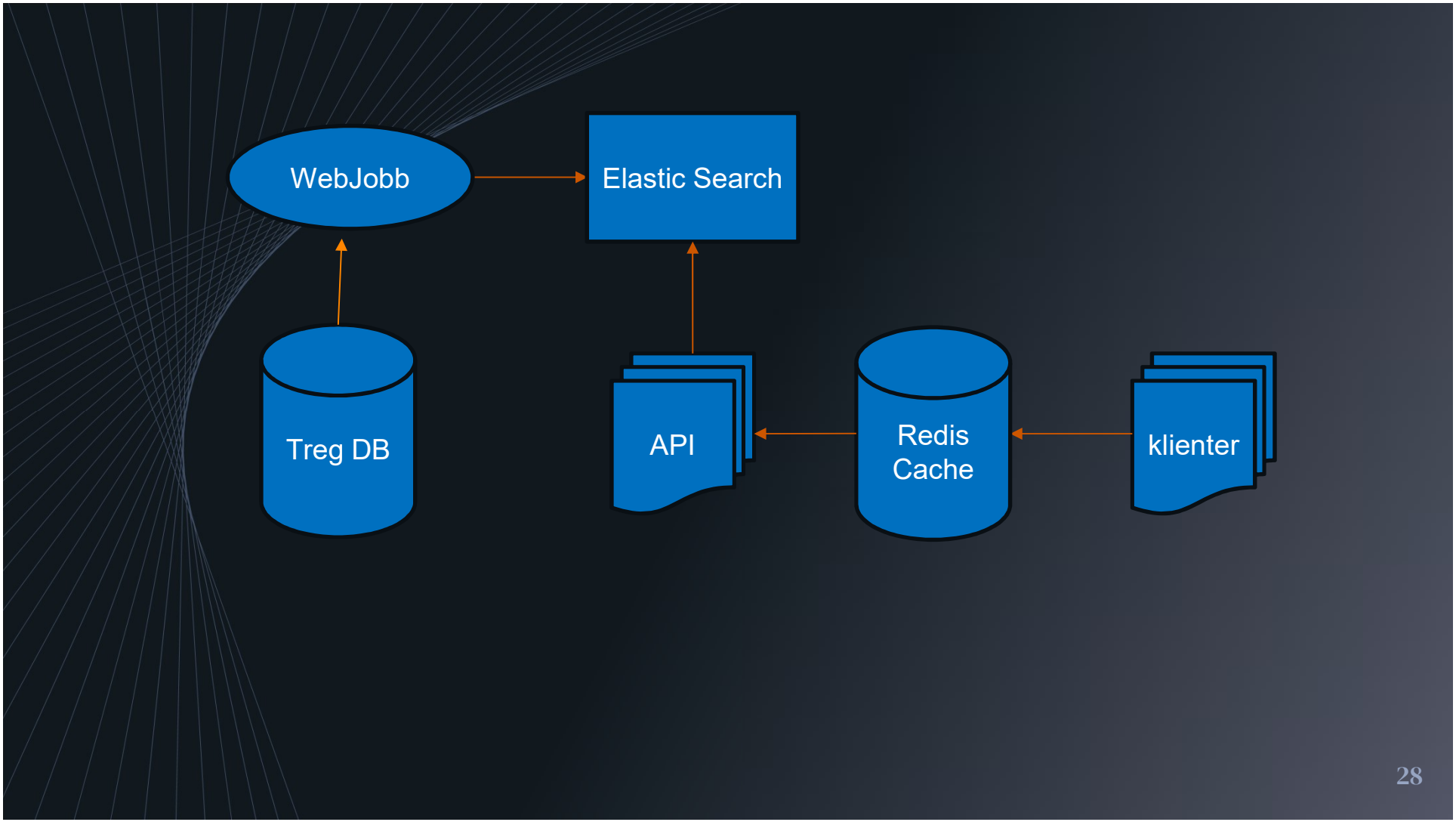
```
public DbValueEntry(Expression<Func<dynamic, d
{

    var memberExpression = expression.Body as
    Type sourceType = null;
    if (memberExpression == null)
    {
        var unaryExpression = expression.Body
        if (unaryExpression != null)
            memberExpression = unaryExpression

        var newExpression = expression.Body as
        if (newExpression != null)
        {
            var src MemberInfo = newExpression.Me
            if (src != null)
            {
                Name = src.Name.ToUnderscoreCase();
                sourceType = ((PropertyInfo)src).PropertyType;
                Type = MapToPgType(sourceType);
                var getter MethodInfo? = ((PropertyInfo)src).GetGetMethod();
                if (getter != null)
                {
                    Entry = getter.Invoke( obj: expression.Compile().Invoke( arg: null), parameters: null);
                }

            }     ...
```

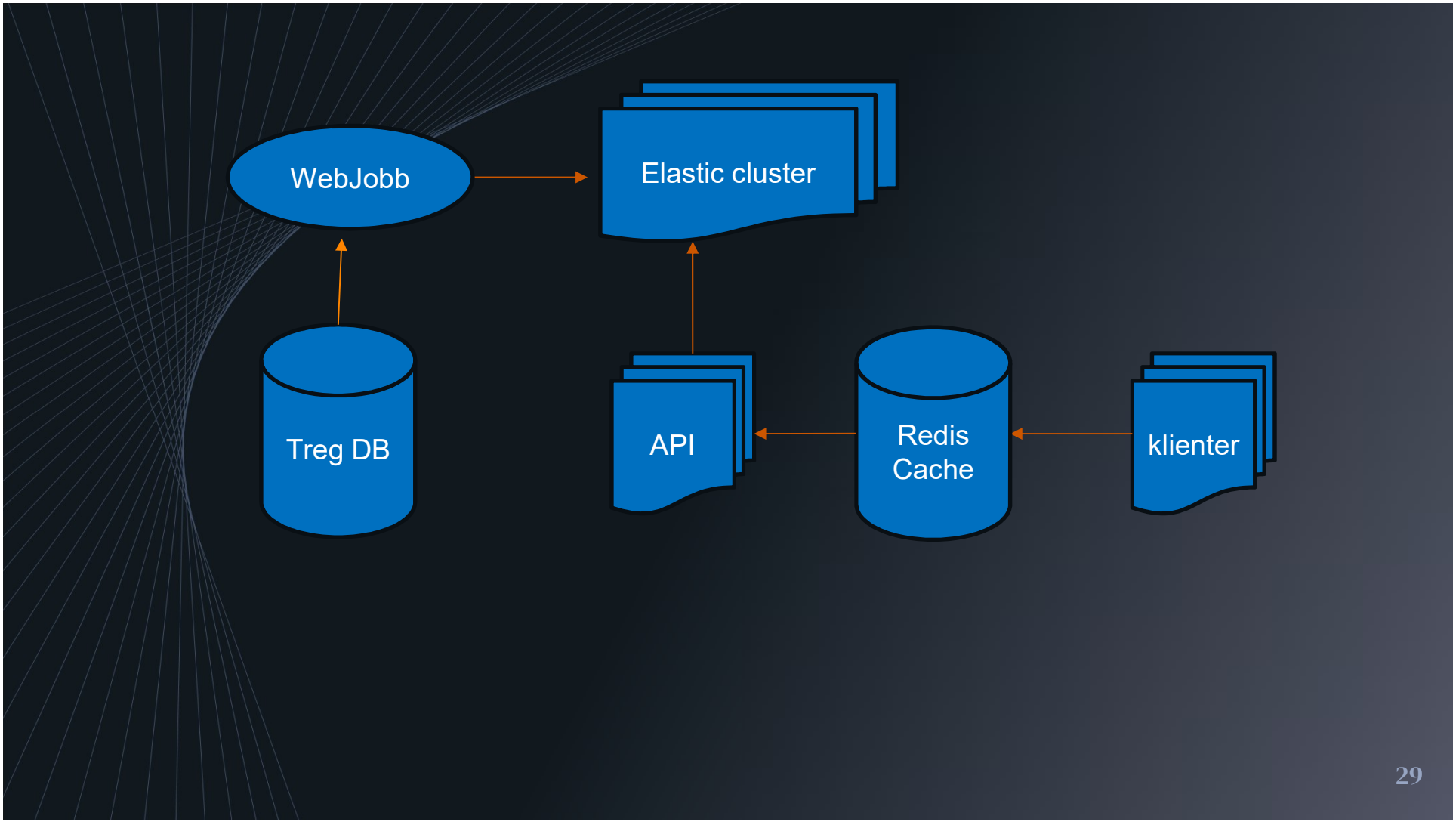
Her må det komme «Vi klarte å få ned respomnstid fra XXX til X ms og alt flyr

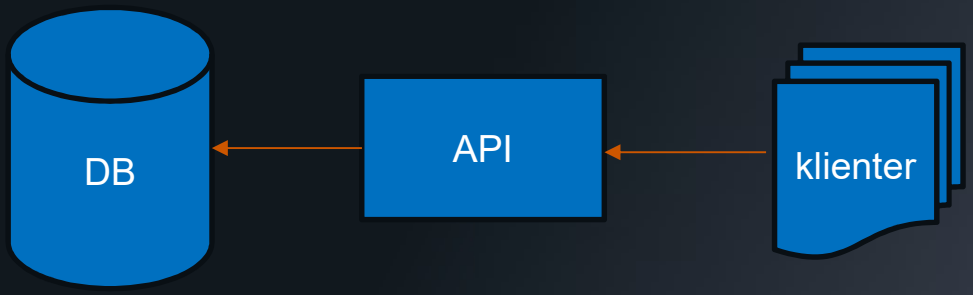

Treg DB
API
klienter


Treg DB
API
klienter


Treg DB
API
Redis
Cache
klienter


WebJobb
Elastic Search
Treg DB
API
Redis
Cache
klienter


WebJobb
Elastic cluster
Treg DB
API
Redis
Cache
klienter


DB
API
klienter

Oppsummering

- Tenk på plattform
- Valider «kjente» fakta
- Kode som er pent er ikke garantert godt
- Kjør ytelsestest med k6 og for validere før produksjon
- Kjør enkelt profilerings sesjon for å finne flaskehalser
- Pass på det som kjøres mange ganger eller mot store datasett