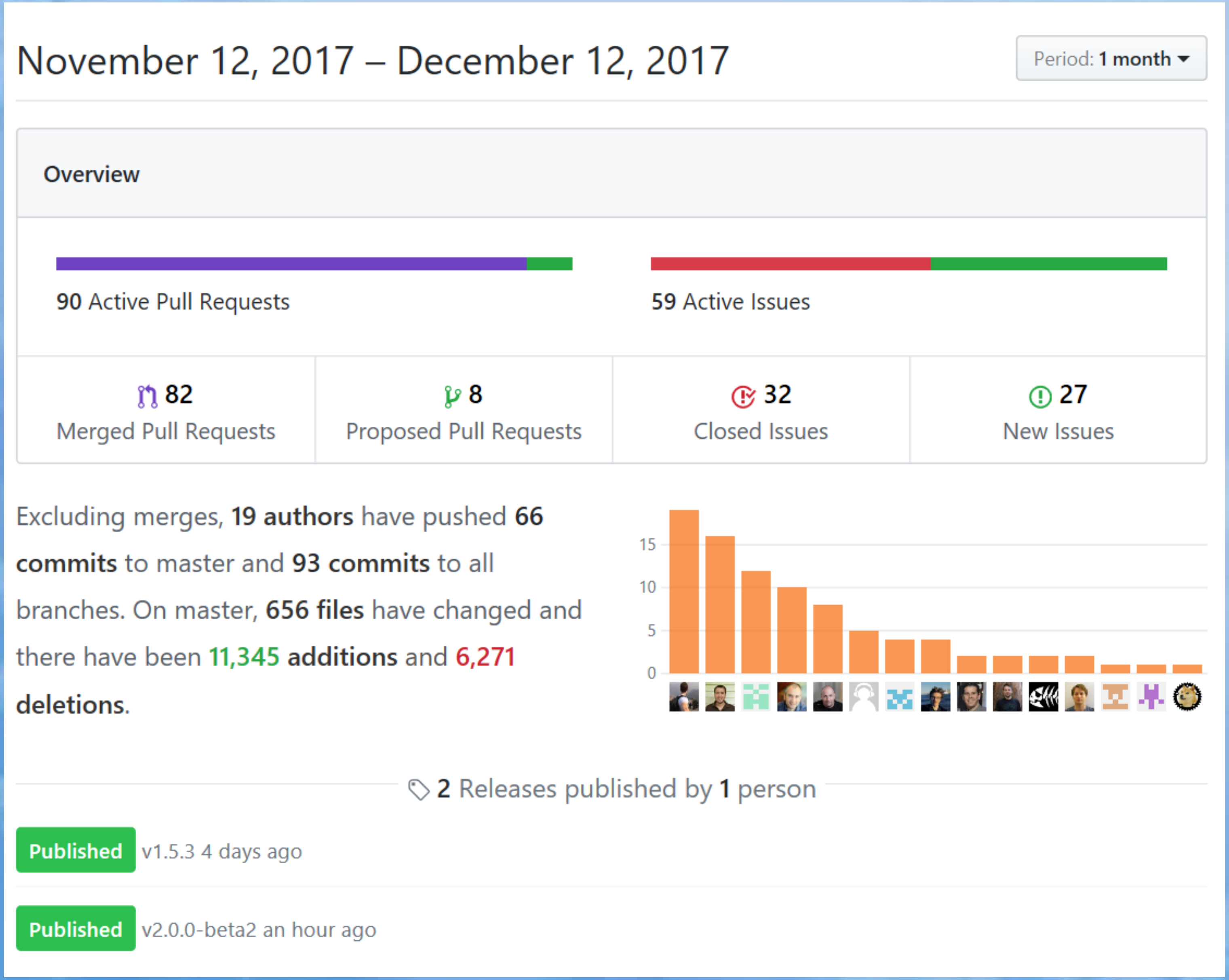
The Long and Winding Road to Orleans 2.0

Community Meetup #15
December 13th 2017

THANK YOU!



November 12, 2017 – December 12, 2017

Period: **1 month**

Overview

90 Active Pull Requests

59 Active Issues

82 Merged Pull Requests	**8** Proposed Pull Requests	**32** Closed Issues	**27** New Issues

Excluding merges, **19 authors** have pushed **66 commits** to master and **93 commits** to all branches. On master, **656 files** have changed and there have been **11,345 additions** and **6,271 deletions**.

2 Releases published by **1** person

Published v1.5.3 4 days ago

Published v2.0.0-beta2 an hour ago

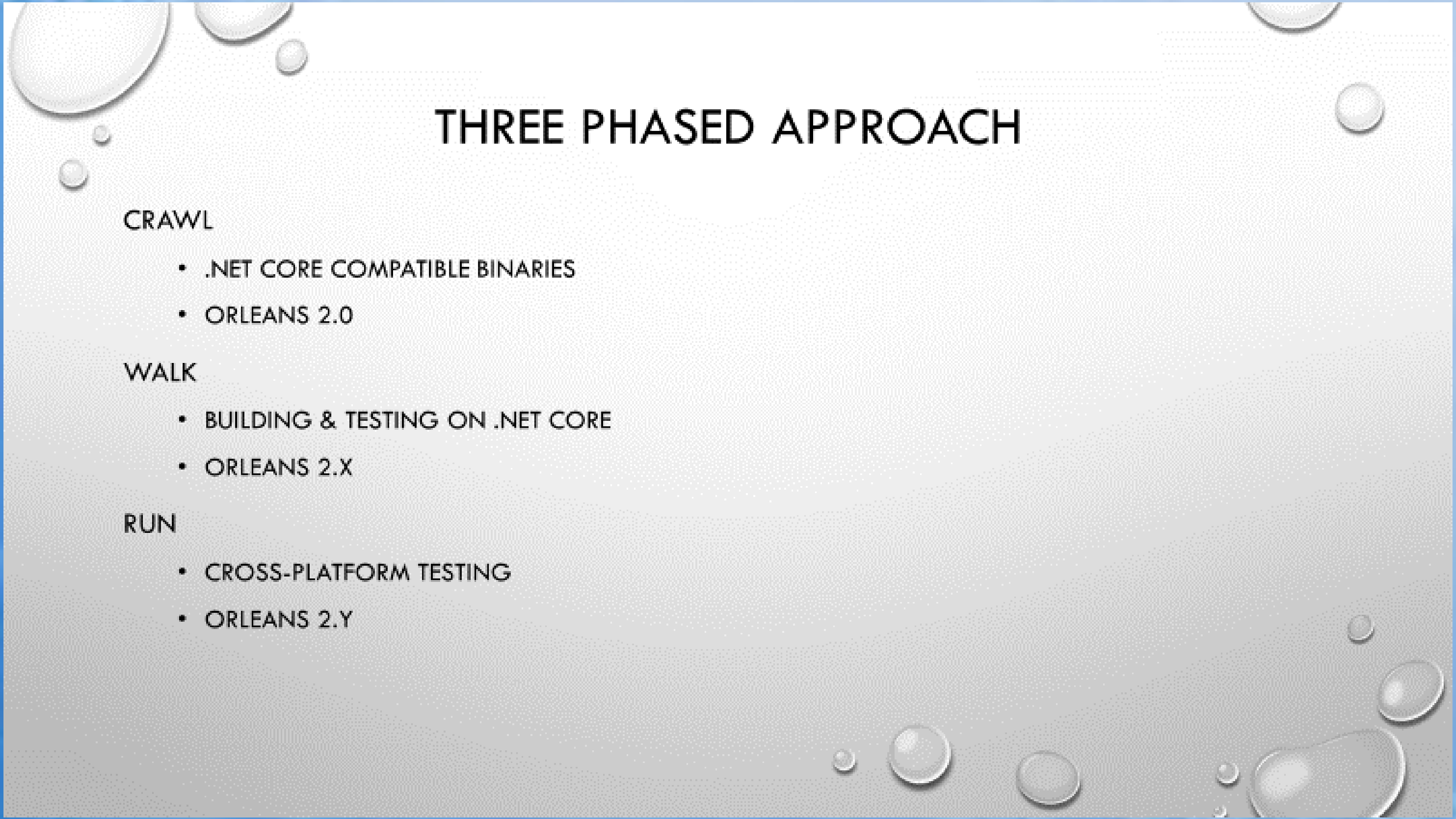
Key Goals of Orleans 2.0

- Compatibility with .NET Core
  - Via .NET Standard 2.0
  - Enable cross-platform
- Modernization of codebase
  - Fluid programmatic configuration
  - Extensibility via DI
  - Restructuring and re-componentization
  - Leverage new capabilities of the platform

Added Goals

- Support for ACID cross-grain transactions
  - In beta
- New composition model (facets)
  - Move away from base classes
- Containers
  - Configuration and shutdown challenges

Still True



THREE PHASED APPROACH

CRAWL

- .NET CORE COMPATIBLE BINARIES
- ORLEANS 2.0

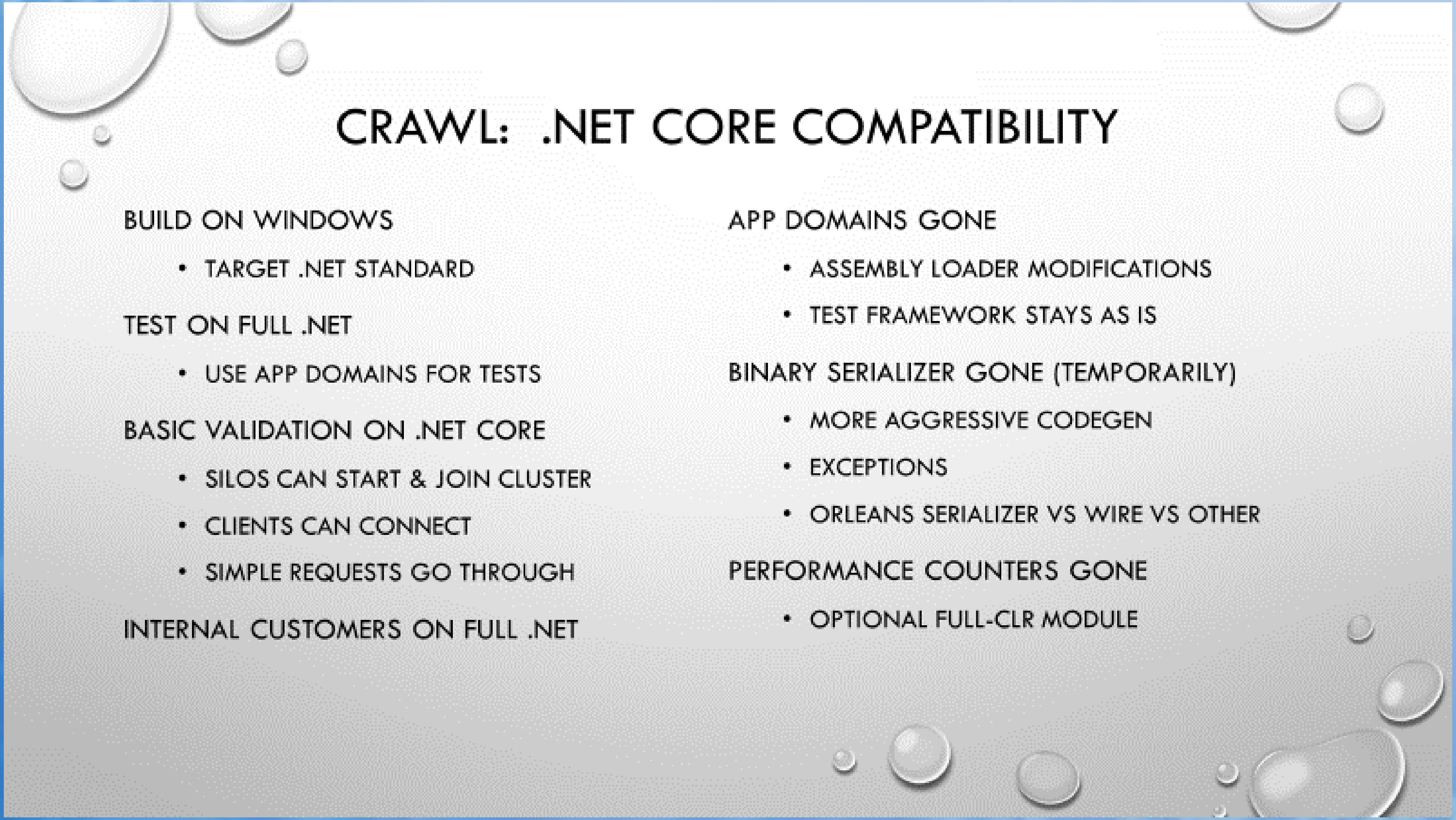
WALK

- BUILDING & TESTING ON .NET CORE
- ORLEANS 2.X

RUN

- CROSS-PLATFORM TESTING
- ORLEANS 2.Y

Are We There Yet?



CRAWL: .NET CORE COMPATIBILITY

BUILD ON WINDOWS

- TARGET .NET STANDARD

TEST ON FULL .NET

- USE APP DOMAINS FOR TESTS

BASIC VALIDATION ON .NET CORE

- SILOS CAN START & JOIN CLUSTER
- CLIENTS CAN CONNECT
- SIMPLE REQUESTS GO THROUGH

INTERNAL CUSTOMERS ON FULL .NET

APP DOMAINS GONE

- ASSEMBLY LOADER MODIFICATIONS
- TEST FRAMEWORK STAYS AS IS

BINARY SERIALIZER GONE (TEMPORARILY)

- MORE AGGRESSIVE CODEGEN
- EXCEPTIONS
- ORLEANS SERIALIZER VS WIRE VS OTHER

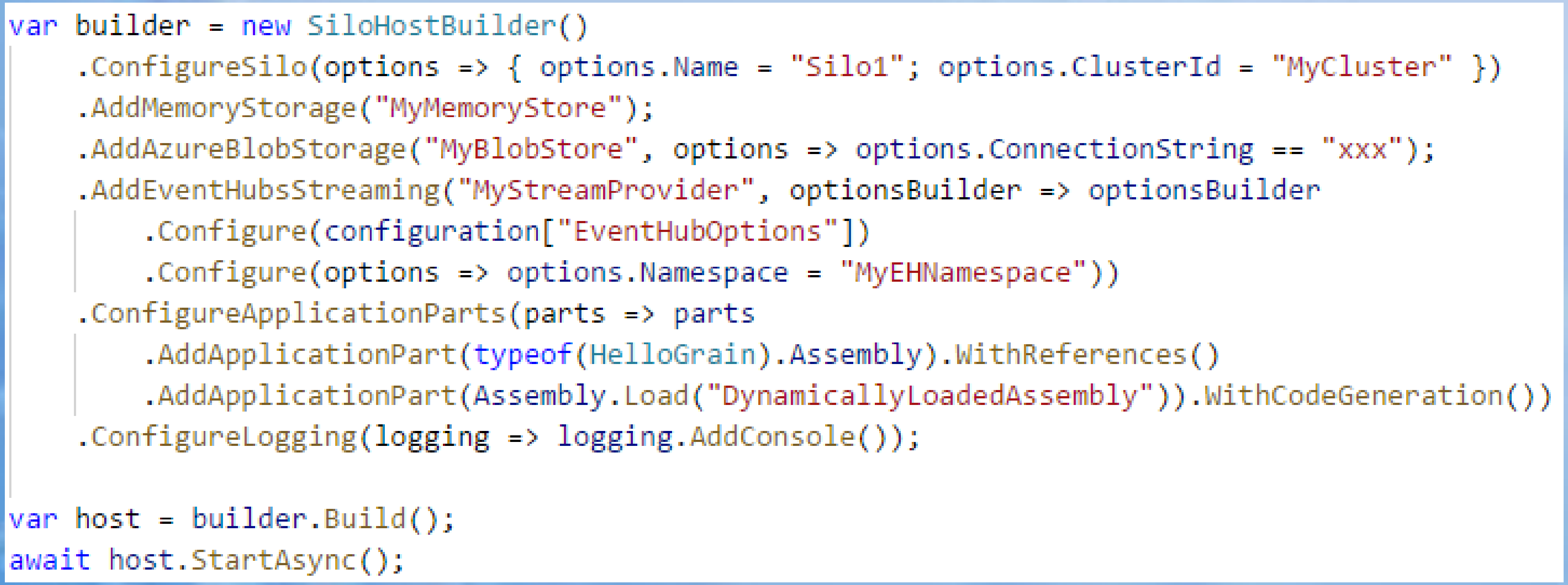
PERFORMANCE COUNTERS GONE

- OPTIONAL FULL-CLR MODULE

Changes to Configuration

- ClientBuilder and SiloHostBuilder
  - Modern fluid API
  - No statics
  - Everything via DI
- Explicit loading of application code
- Leverage ASP.NET Core components and principles

Example of configuration (tentative)



```
var builder = new SiloHostBuilder()
    .ConfigureSilo(options => { options.Name = "Silo1"; options.ClusterId = "MyCluster" })
    .AddMemoryStorage("MyMemoryStore");
    .AddAzureBlobStorage("MyBlobStore", options => options.ConnectionString == "xxx");
    .AddEventHubsStreaming("MyStreamProvider", optionsBuilder => optionsBuilder
        .Configure(configuration["EventHubOptions"])
        .Configure(options => options.Namespace = "MyEHNamespace"))
    .ConfigureApplicationParts(parts => parts
        .AddApplicationPart(typeof(HelloGrain).Assembly).WithReferences()
        .AddApplicationPart(Assembly.Load("DynamicallyLoadedAssembly")).WithCodeGeneration())
    .ConfigureLogging(logging => logging.AddConsole());

var host = builder.Build();
await host.StartAsync();
```

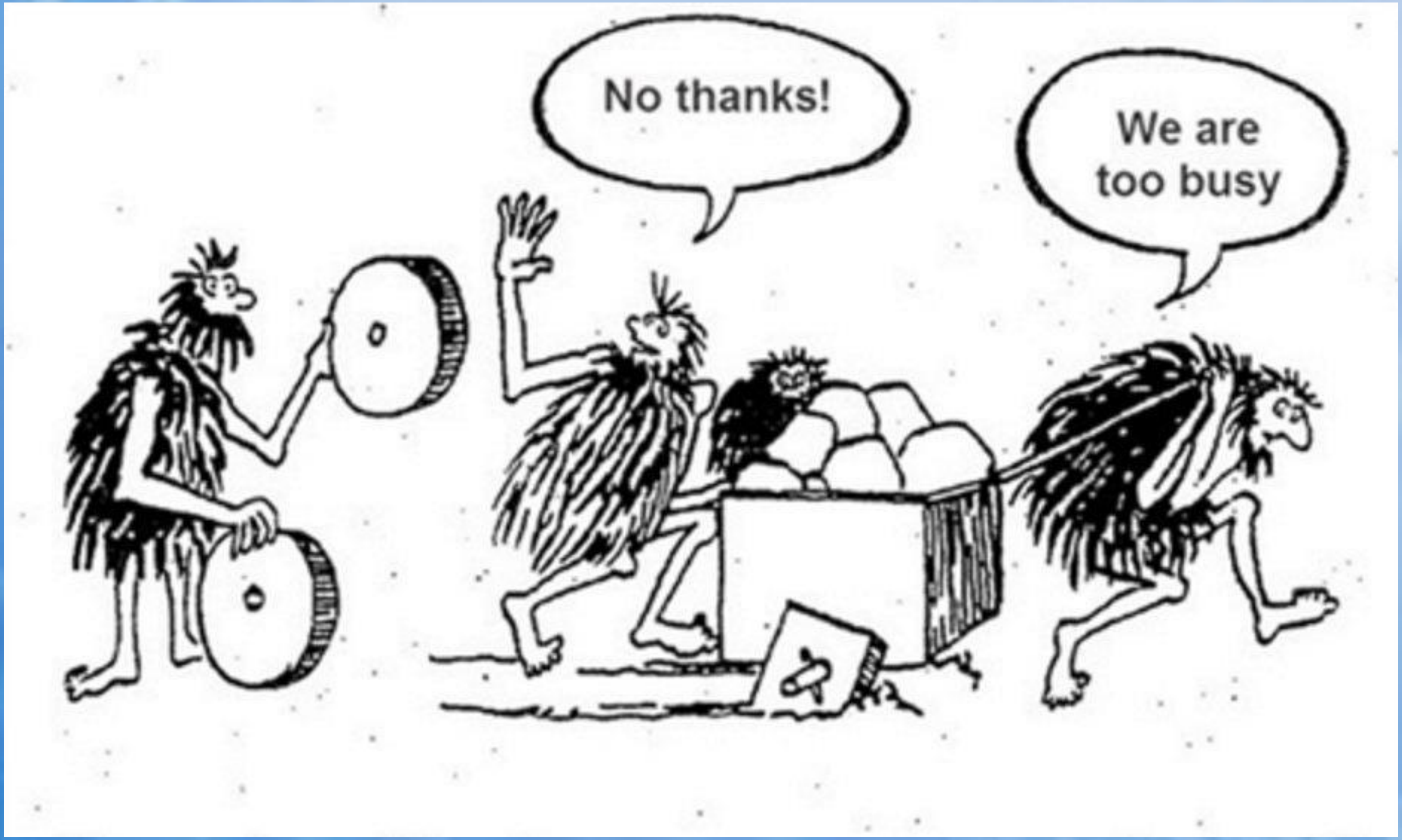
Restructuring

- Microsoft.Orleans.Core.Abstractions
  - Abstraction package for grain development
- Microsoft.Orleans.CodeGeneration.Build
  - Build time codegen
- Most app projects should only need these 2
- Should rarely need an upgrade

Restructuring cont'd

- From Microsoft.Orleans.OrleansAzureUtils to:
  - Orleans.Clustering.AzureStorage
  - Orleans.Persistence.AzureStorage
  - Orleans.Reminders.AzureStorage
  - Orleans.Statistics.AzureStorage
  - Orleans.Streaming.AzureStorage
  - Orleans.Hosting.AzureCloudServices
- From Microsoft.Orleans.OrleansAWSUtils to:
  - Microsoft.Orleans.Clustering.DynamoDB
  - Microsoft.Orleans.Persistence.DynamoDB
  - Microsoft.Orleans.Reminders.DynamoDB
  - Microsoft.Orleans.Streaming.SQS

Leverage platform capabilities



Leverage platform capabilities cont’d

- Logging
- DI
- Generic host
- Options

Transactions

- Definitions
  - Grain – Owner of persisted state
  - Transaction – Operation across a set of grains in which all state changes are either successful or aborted together.
- Scenarios & goals
  - Performant distributed transactions across arbitrary number of grains.
- Beta
  - Transactions involving up to 8 single state grains.
  - Correctness, performance, recoverability testing.
- Plans for release
  - Orleans 2.0 – Beta Orleans Transactions
  - Orleans 2.2 – Orleans Transaction release

http://dotnet.github.io/orleans/Documentation/2.0/Transactions.html

Please Help Us Test 2.0!

- Running on .NET Core
  - Windows
  - Linux
- Various application environments
- Serialization, especially for F# types
- Docker
  - Windows/Linux containers, K8s, SF
- Transactions