Microsoft

Azure

Scaling the Monolith

Daniel Larsen

Technical Evangelist

2019

It’s a Trilogy!

Scaling the monolith

Refactoring the monolith

Strangling the monolith

Scalability

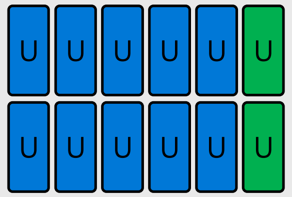
Scale up

Scale up App Services
Scale up VMs
Scale up SQL DB
More cores, more RAM

Scale out

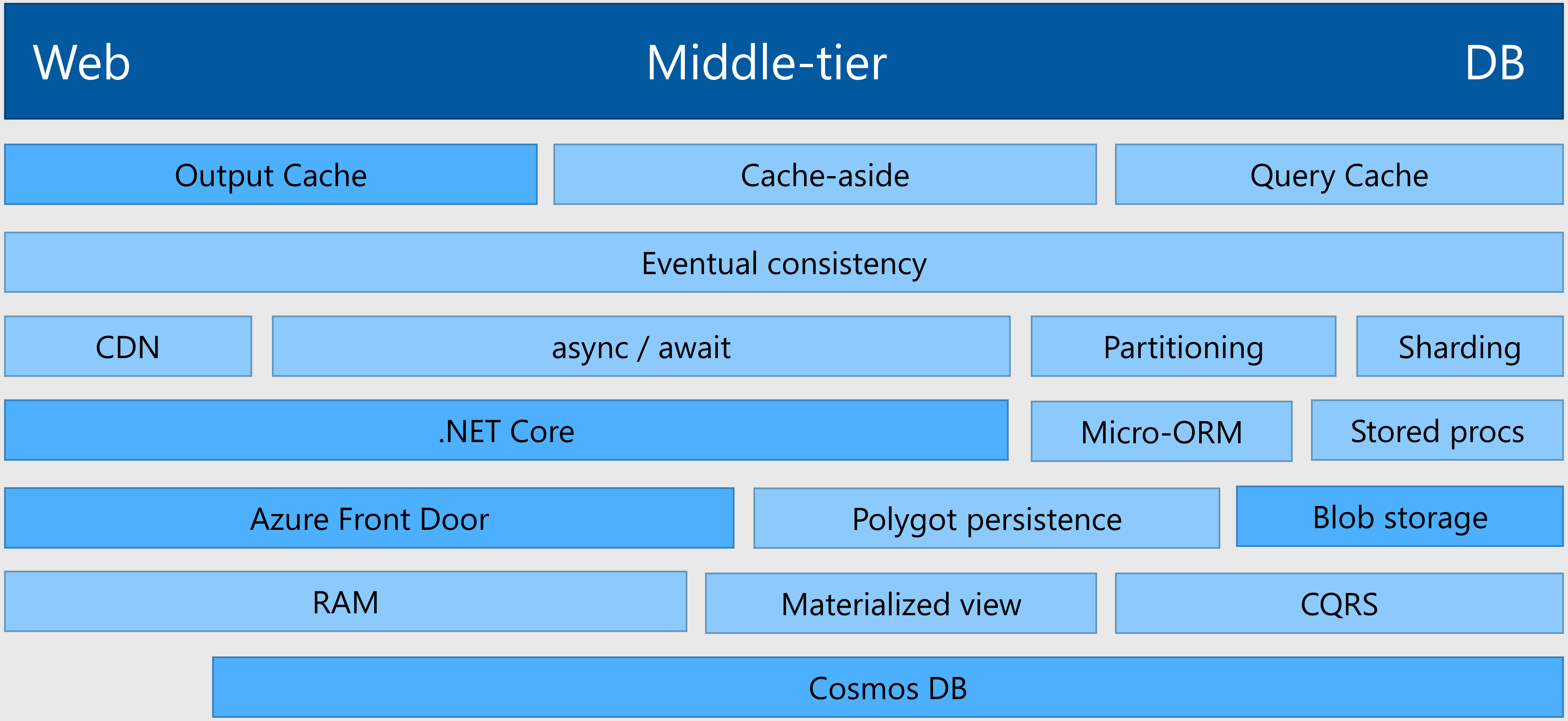
Scale out App Services
Scale out VM Scale Sets
Auto-scale
Web farm + Load balancing
SQL Clustering

Optimize

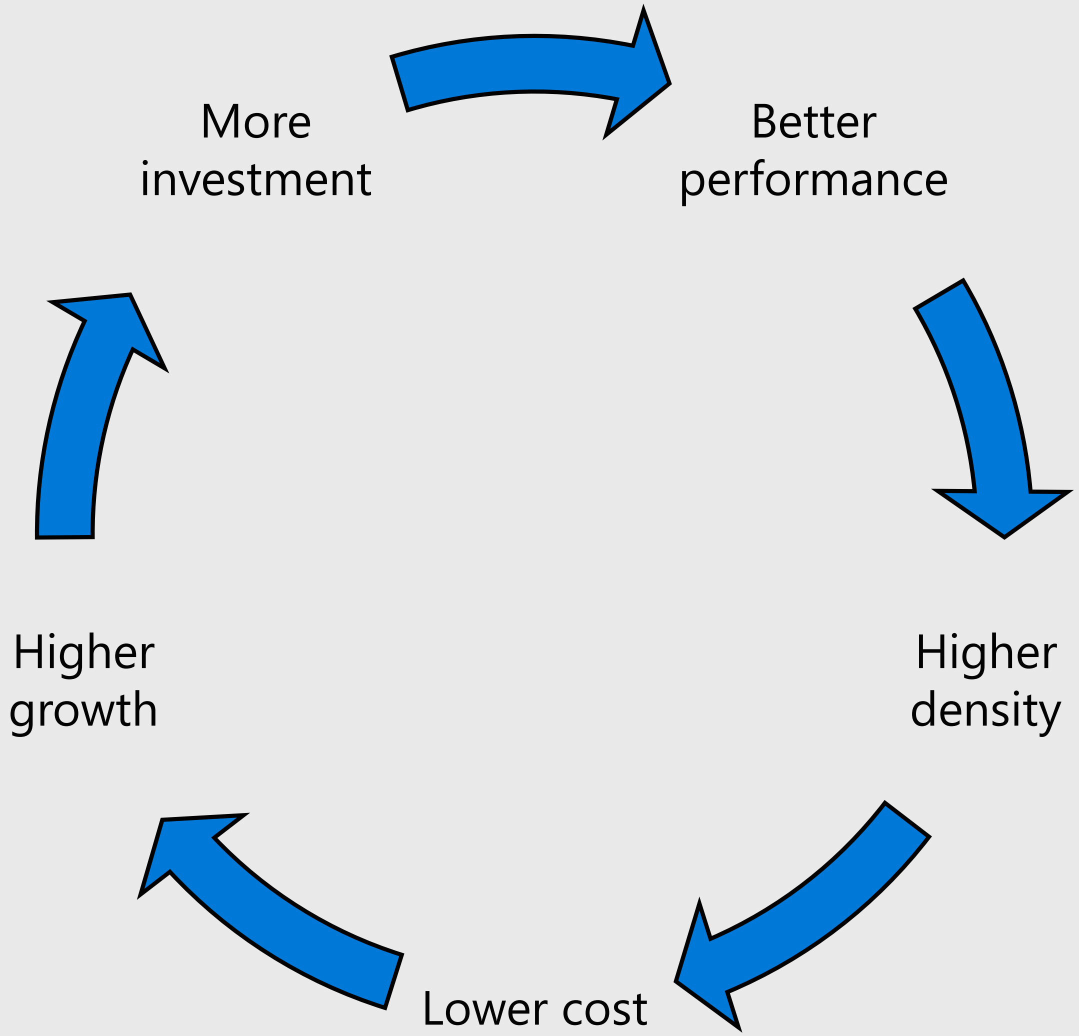


Scalability of Software Architecture
Optimization of Software (code)
Optimization of Platform services
Cloud design patterns

Patterns and practices for Scalability of software

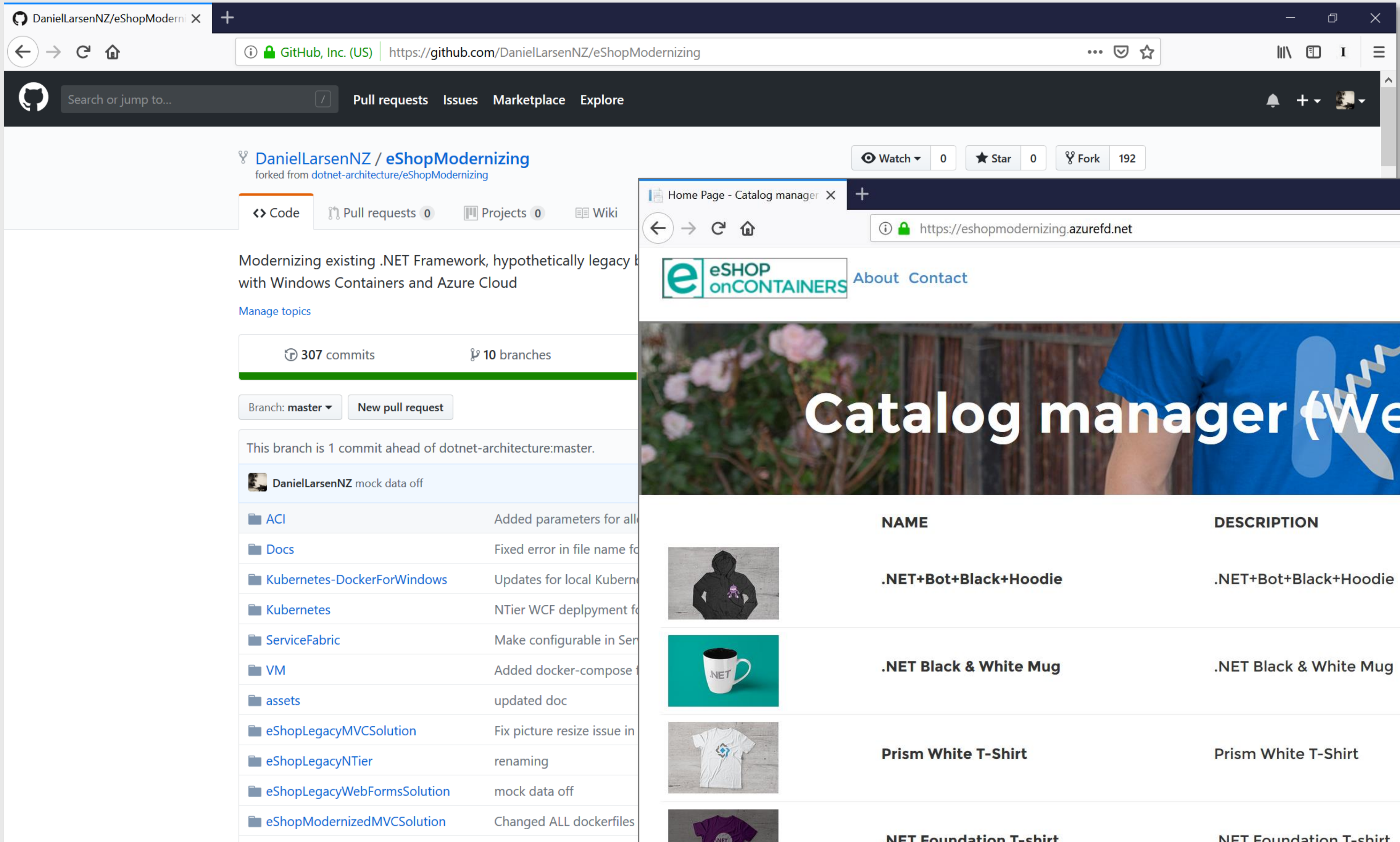


Performance engineering



eShopModernizing

https://github.com/DanielLarsenNZ/eShopModernizing



DanielLarsenNZ/eShopModernizing
forked from dotnet-architecture/eShopModernizing

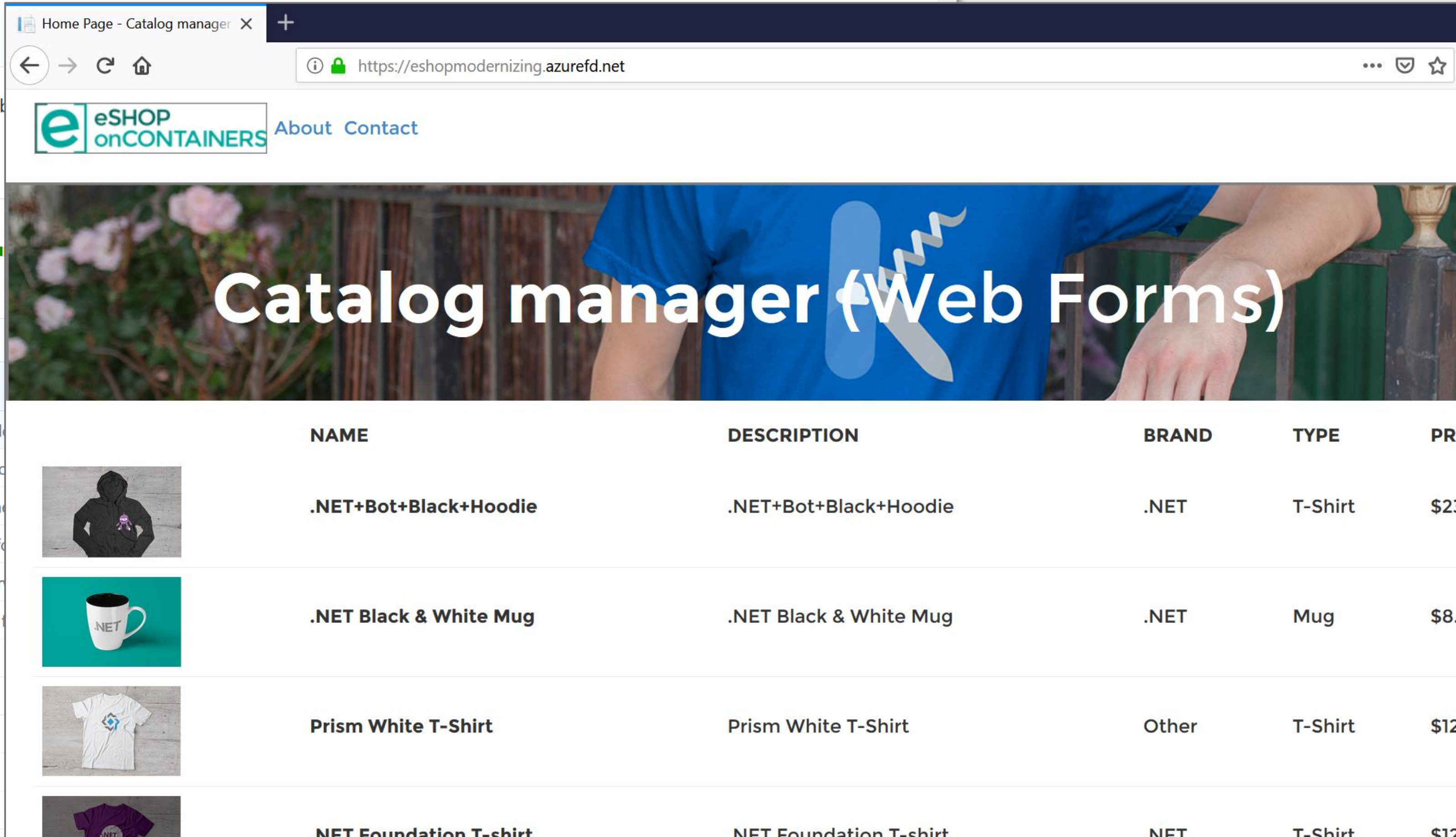
Modernizing existing .NET Framework, hypothetically legacy
with Windows Containers and Azure Cloud

307 commits · 10 branches

This branch is 1 commit ahead of dotnet-architecture:master.

DanielLarsenNZ mock data off

ACI	Added parameters for all
Docs	Fixed error in file name fo
Kubernetes-DockerForWindows	Updates for local Kuberne
Kubernetes	NTier WCF deplpyment fo
ServiceFabric	Make configurable in Ser
VM	Added docker-compose f
assets	updated doc
eShopLegacyMVCSolution	Fix picture resize issue in
eShopLegacyNTier	renaming
eShopLegacyWebFormsSolution	mock data off
eShopModernizedMVCSolution	Changed ALL dockerfiles



https://eshopmodernizing.azurefd.net

Catalog manager (Web Forms)

	NAME	DESCRIPTION	BRAND	TYPE	PR
	.NET+Bot+Black+Hoodie	.NET+Bot+Black+Hoodie	.NET	T-Shirt	$2
	.NET Black & White Mug	.NET Black & White Mug	.NET	Mug	$8
	Prism White T-Shirt	Prism White T-Shirt	Other	T-Shirt	$12
	.NET Foundation T-shirt	.NET Foundation T-shirt	.NET	T-Shirt	$1

Key performance indicators

↓10ms
server response

↓1s
page load

↓100
dtu max

↑10,000
rpm / core

↑25:1
web : db calls

Demo

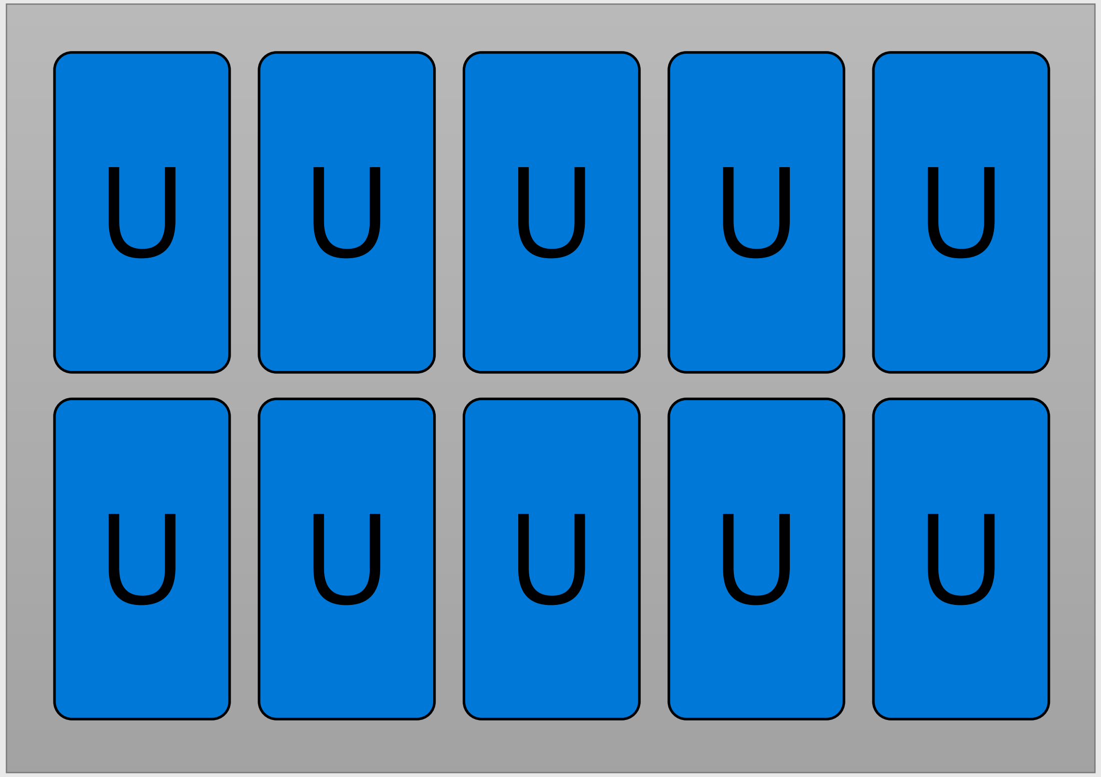
Azure DevOps Load Tests

https://eshopmodernizingpv2-baseline.azurewebsites.net/

Cloud economics primer

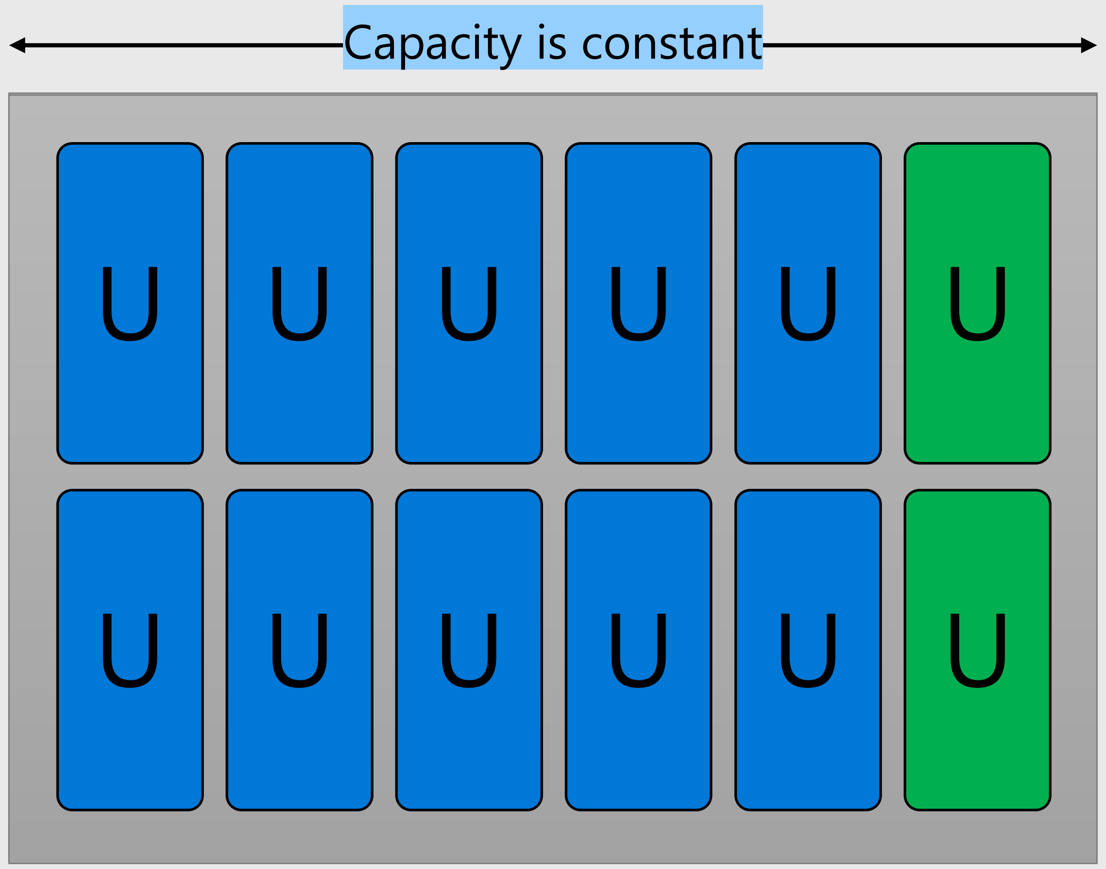
Density

Cost to serve: e.g. Cost per *n* users / tenants / transactions



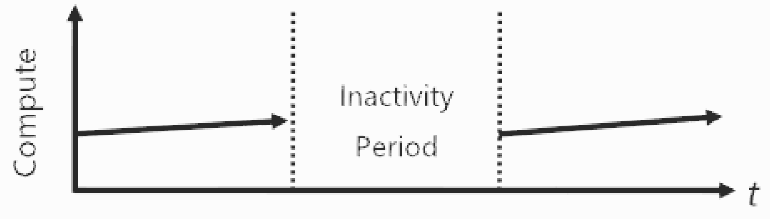
Density

Cost to serve: e.g. Cost per *n* users / tenants / transactions



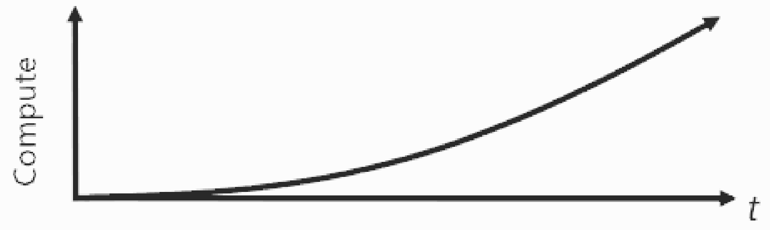
20% higher density = 20% reduction in cost to serve

Cloud Computing Patterns



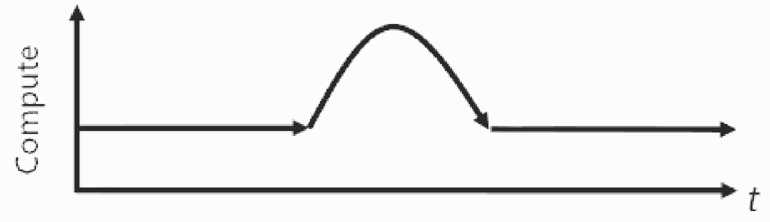
On and Off

On & off workloads (e.g. batch job)
Over provisioned capacity is wasted
Time to market can be cumbersome



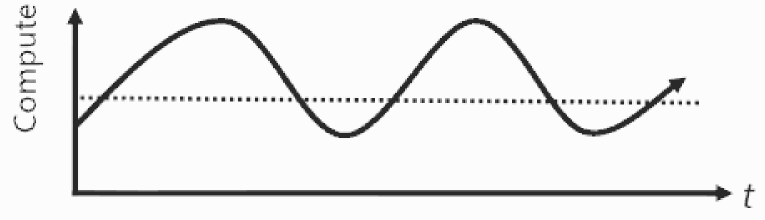
Growing Fast

Successful services needs to grow/scale
Keeping up w/ growth is big IT challenge
Cannot provision hardware fast enough



Unpredictable Bursting

Unexpected/unplanned peak in demand
Sudden spike impacts performance
Can't over provision for extreme cases

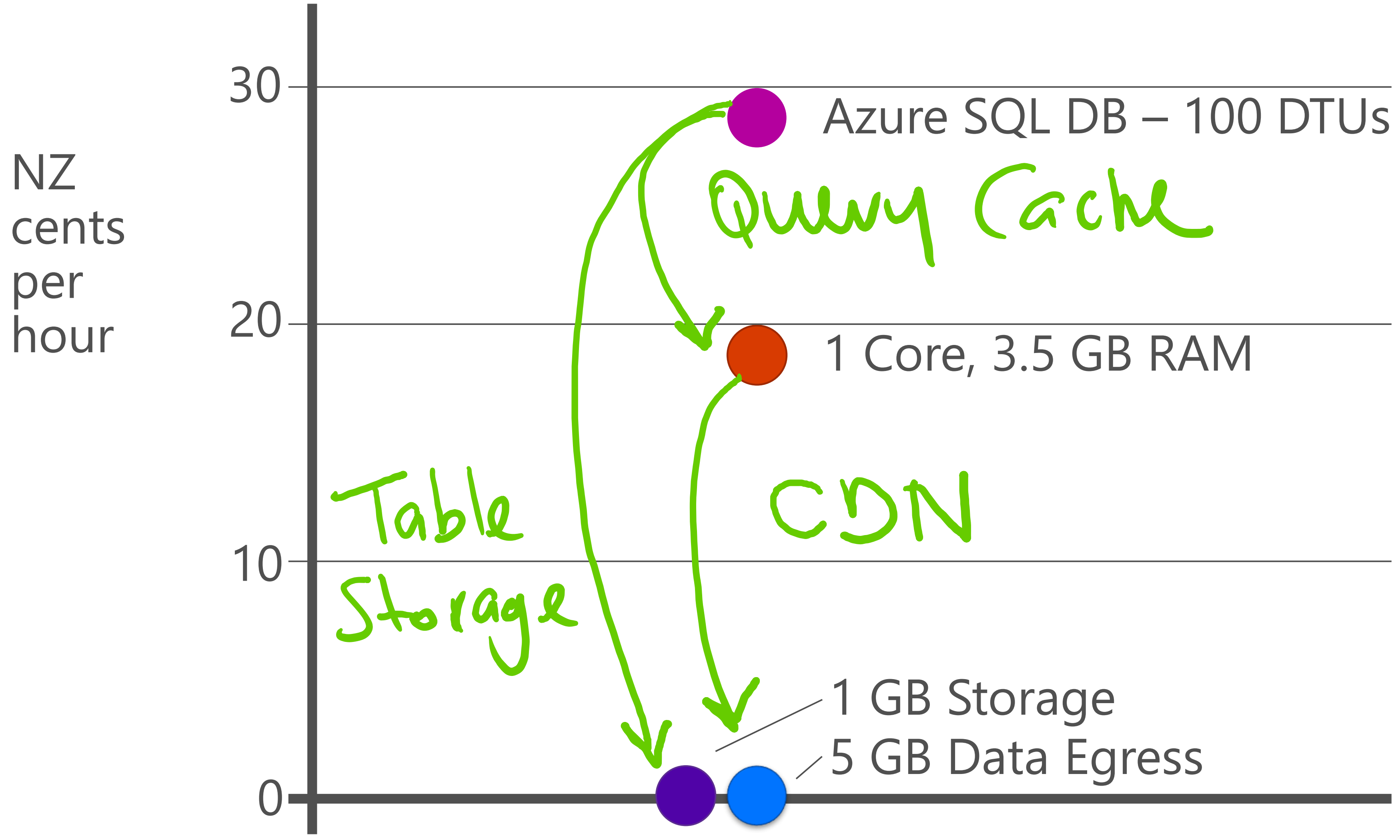


Predictable Bursting

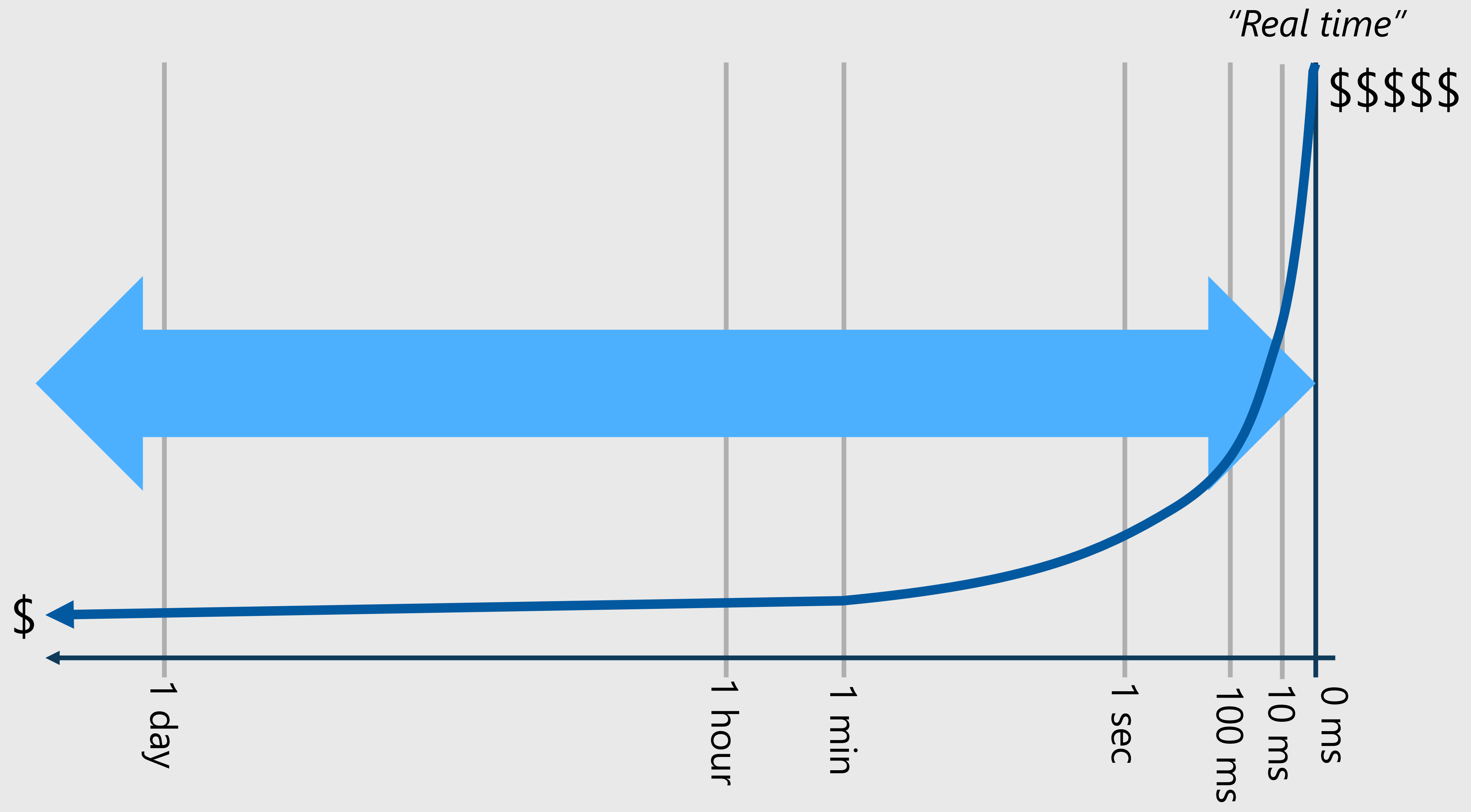
Services with micro seasonality trends
Peaks due to periodic increased demand
IT complexity and wasted capacity

Optimising costs

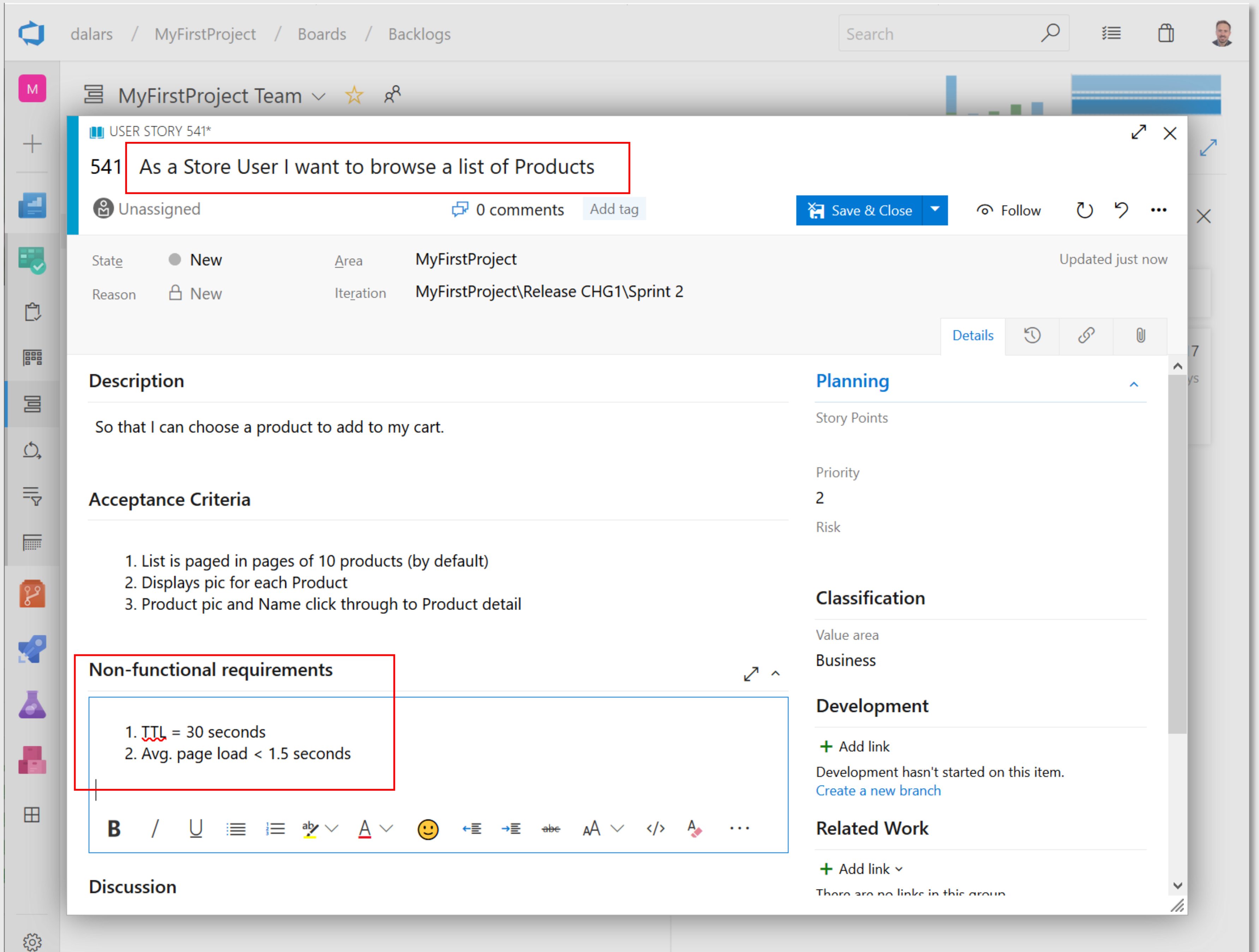
Polygot Persistence



Eventual consistency



As a User



USER STORY 541*

541 As a Store User I want to browse a list of Products

Unassigned
0 comments
Add tag
Save & Close
Follow

State: New
Reason: New
Area: MyFirstProject
Iteration: MyFirstProject\Release CHG1\Sprint 2
Updated just now

Details

Description

So that I can choose a product to add to my cart.

Acceptance Criteria

1. List is paged in pages of 10 products (by default)
2. Displays pic for each Product
3. Product pic and Name click through to Product detail

Non-functional requirements

1. TTL = 30 seconds
2. Avg. page load < 1.5 seconds

Discussion

Planning

Story Points

Priority
2

Risk

Classification

Value area
Business

Development

Add link
Development hasn't started on this item.
Create a new branch

Related Work

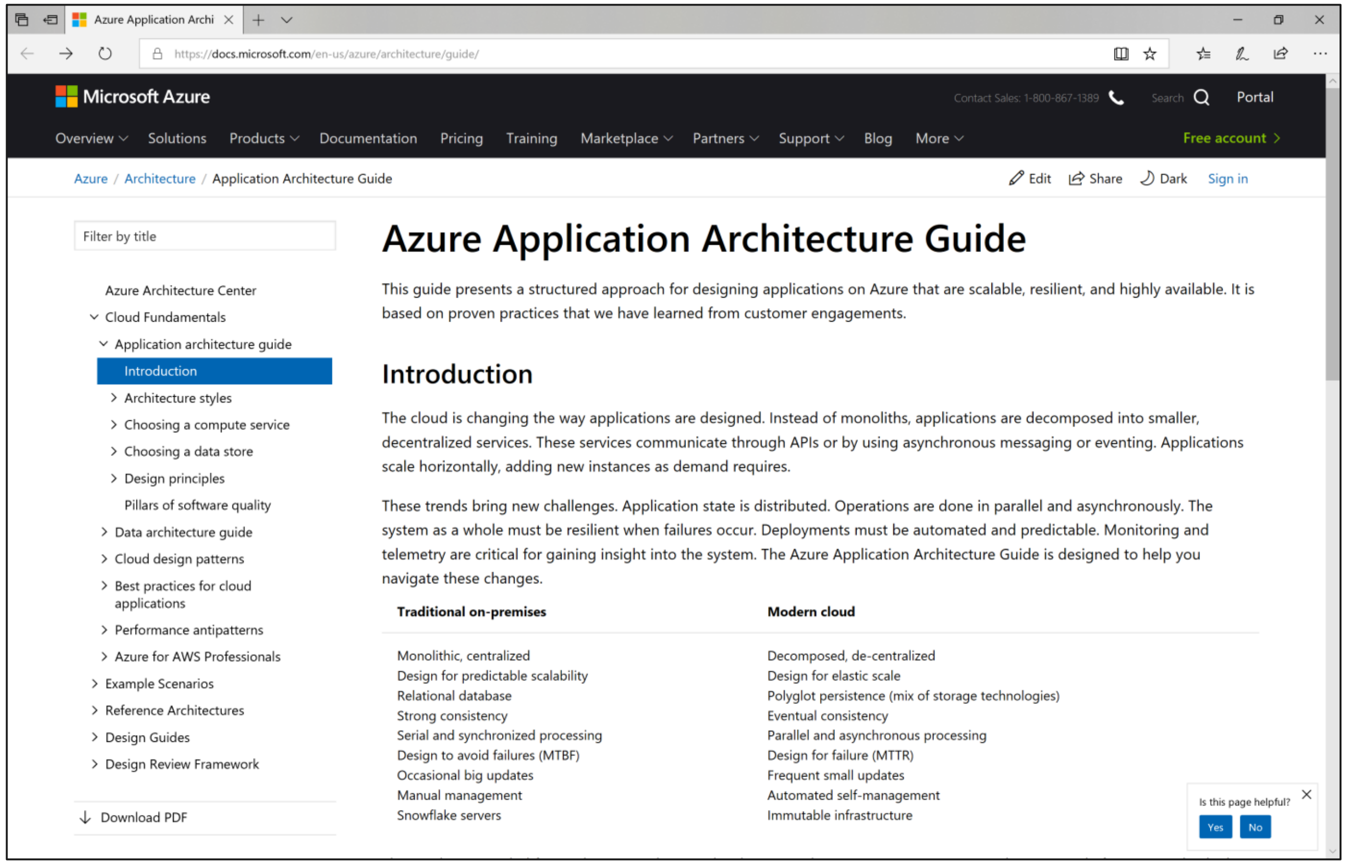
Add link
There are no links in this group

Demo

Output Cache

Scalability checklist

Azure Application Architecture Guide



Azure Application Architecture Guide

This guide presents a structured approach for designing applications on Azure that are scalable, resilient, and highly available. It is based on proven practices that we have learned from customer engagements.

Introduction

The cloud is changing the way applications are designed. Instead of monoliths, applications are decomposed into smaller, decentralized services. These services communicate through APIs or by using asynchronous messaging or eventing. Applications scale horizontally, adding new instances as demand requires.

These trends bring new challenges. Application state is distributed. Operations are done in parallel and asynchronously. The system as a whole must be resilient when failures occur. Deployments must be automated and predictable. Monitoring and telemetry are critical for gaining insight into the system. The Azure Application Architecture Guide is designed to help you navigate these changes.

Traditional on-premises	Modern cloud
Monolithic, centralized	Decomposed, de-centralized
Design for predictable scalability	Design for elastic scale
Relational database	Polyglot persistence (mix of storage technologies)
Strong consistency	Eventual consistency
Serial and synchronized processing	Parallel and asynchronous processing
Design to avoid failures (MTBF)	Design for failure (MTTR)
Occasional big updates	Frequent small updates
Manual management	Automated self-management
Snowflake servers	Immutable infrastructure

https://aka.ms/architecture

Scalability checklist

The **Scalability checklist** contains 30+ recommendations. 10 are listed here.

	Recommendation	**Patterns & guidance**
1	Review the performance antipatterns	Performance antipatterns
2	Use asynchronous calls	Asynchronous programming
3	Design for eventual consistency	Data Consistency Primer
4	Use data partitioning, Consider de-normalizing data	Data partitioning, Materialized View
5	Minimize load on the data store, Minimize the volume of data retrieved, Optimize and tune SQL queries and indexes	Automatic tuning in Azure SQL Database
6	Partition the workload, design for scaling, scale as a unit	Microservices architecture
7	Aggressively cache, Use output cache, Enable client caching	Caching Guidance
8	Offload and distribute intensive CPU/IO tasks	Background jobs, Competing Consumers
9	Use queues to level load	Queue-Based Load Leveling Pattern
10	Carry out performance profiling and load testing	Testing cloud service performance

https://docs.microsoft.com/en-us/azure/architecture/checklist/scalability

Performance antipatterns for cloud applications

A *performance antipattern* is a common practice that is likely to cause scalability problems when an application is under pressure.

Busy Database	Offloading too much processing to a data store
Busy Front End	Moving resource-intensive tasks onto background threads
Chatty I/O	The cumulative effect of a large number of I/O requests
Extraneous Fetching	Retrieving more data than is needed, resulting in unnecessary I/O
Improper Instantiation	Repeatedly creating and destroying objects that are designed to be reused
Monolithic Persistence	Using the same data store for data with very different usage patterns
No Caching	Failing to cache data
Synchronous I/O	Blocking the calling thread while I/O completes

https://docs.microsoft.com/en-us/azure/architecture/antipatterns/index

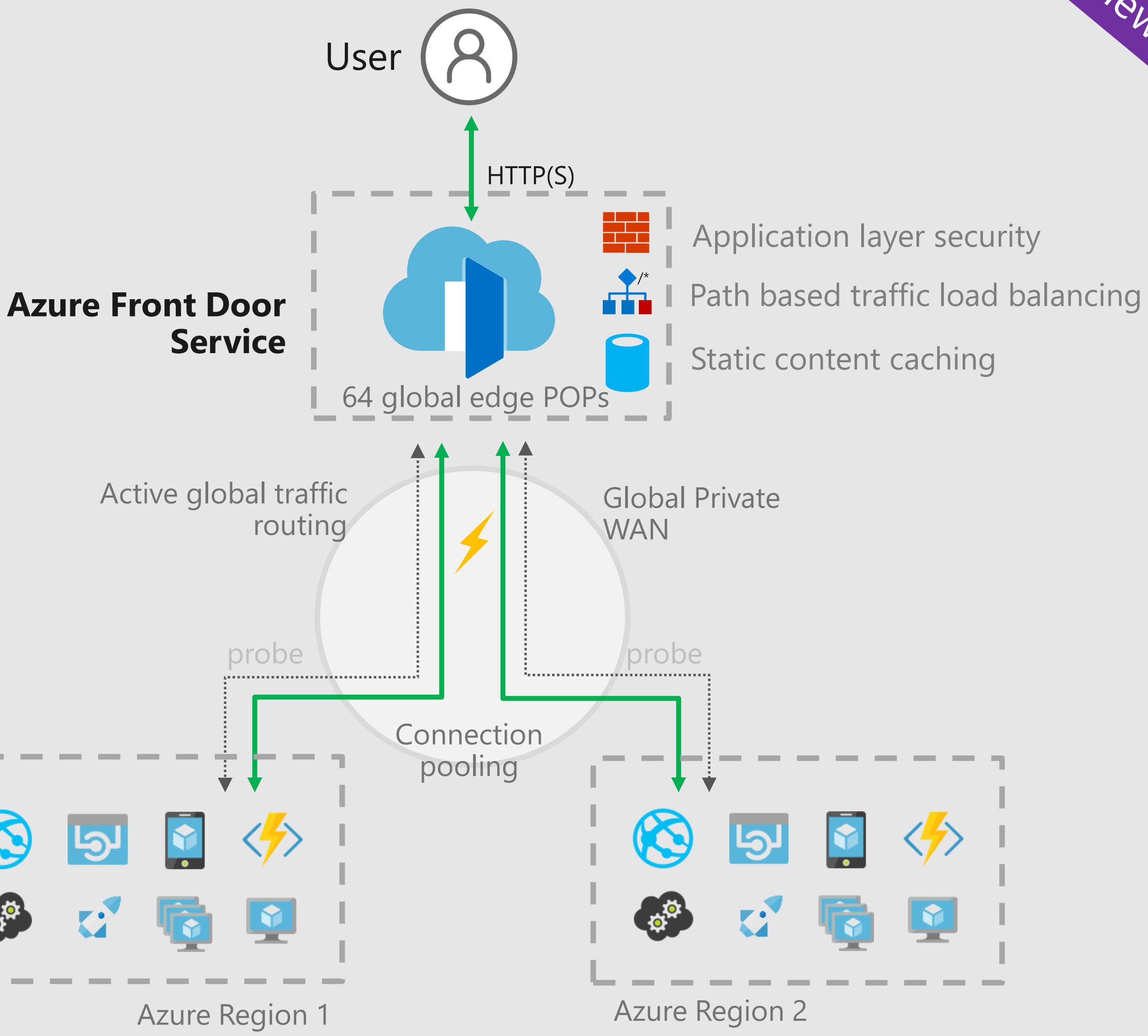
Azure Front Door Service

Your secure entry point for delivering globally performant hyperscale apps.

- ✓ Application acceleration at Microsoft's edge
- ✓ Integration with App Services
- ✓ Global HTTP load balancing with instant failover
- ✓ Massive SSL offload
- ✓ Integrated static content caching
- ✓ Central application traffic dashboard

Office 365 Azure Skype Bing
Azure DevOps MSN OneDrive
Xbox Cortana Windows Teams

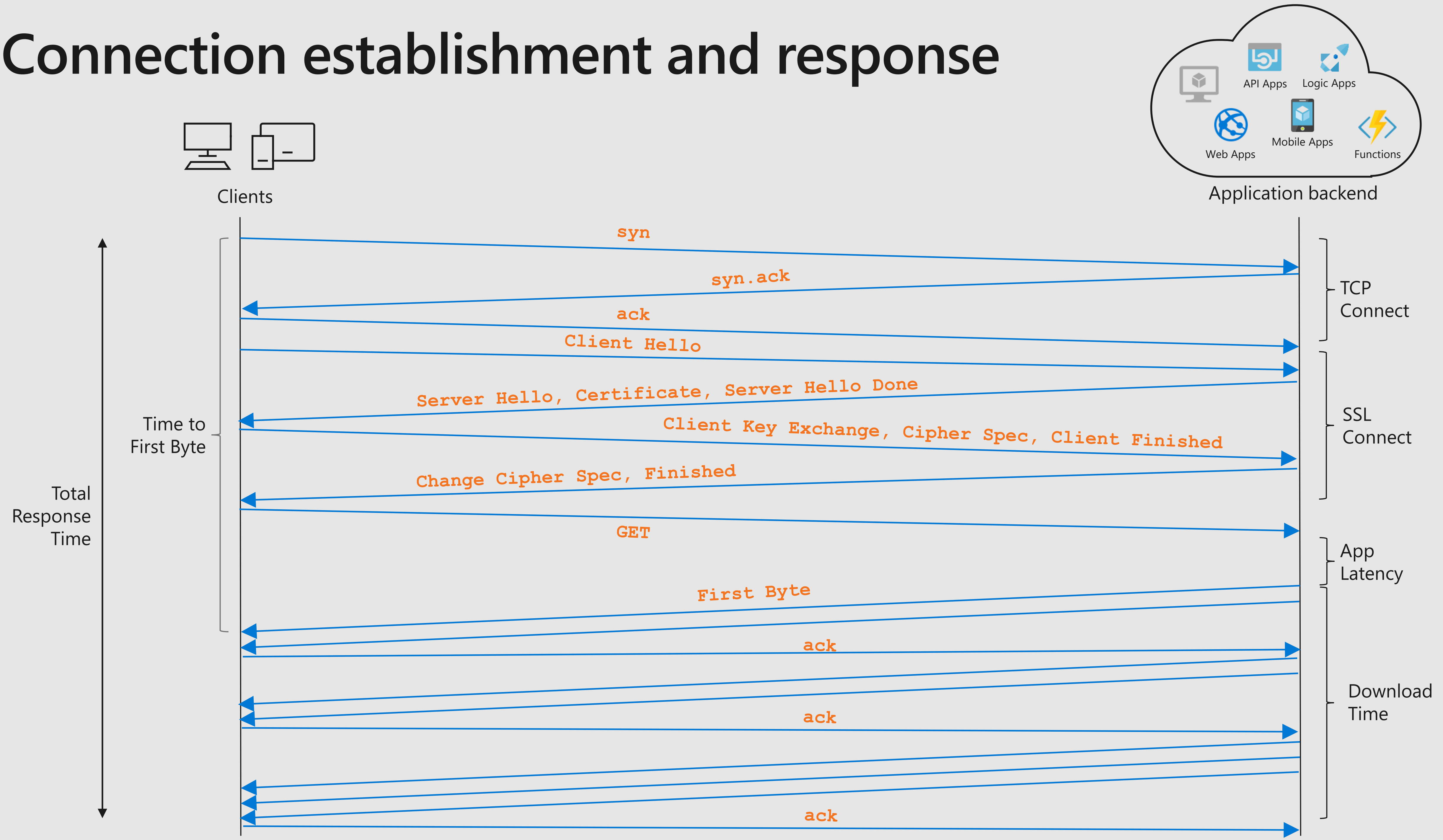
Microsoft



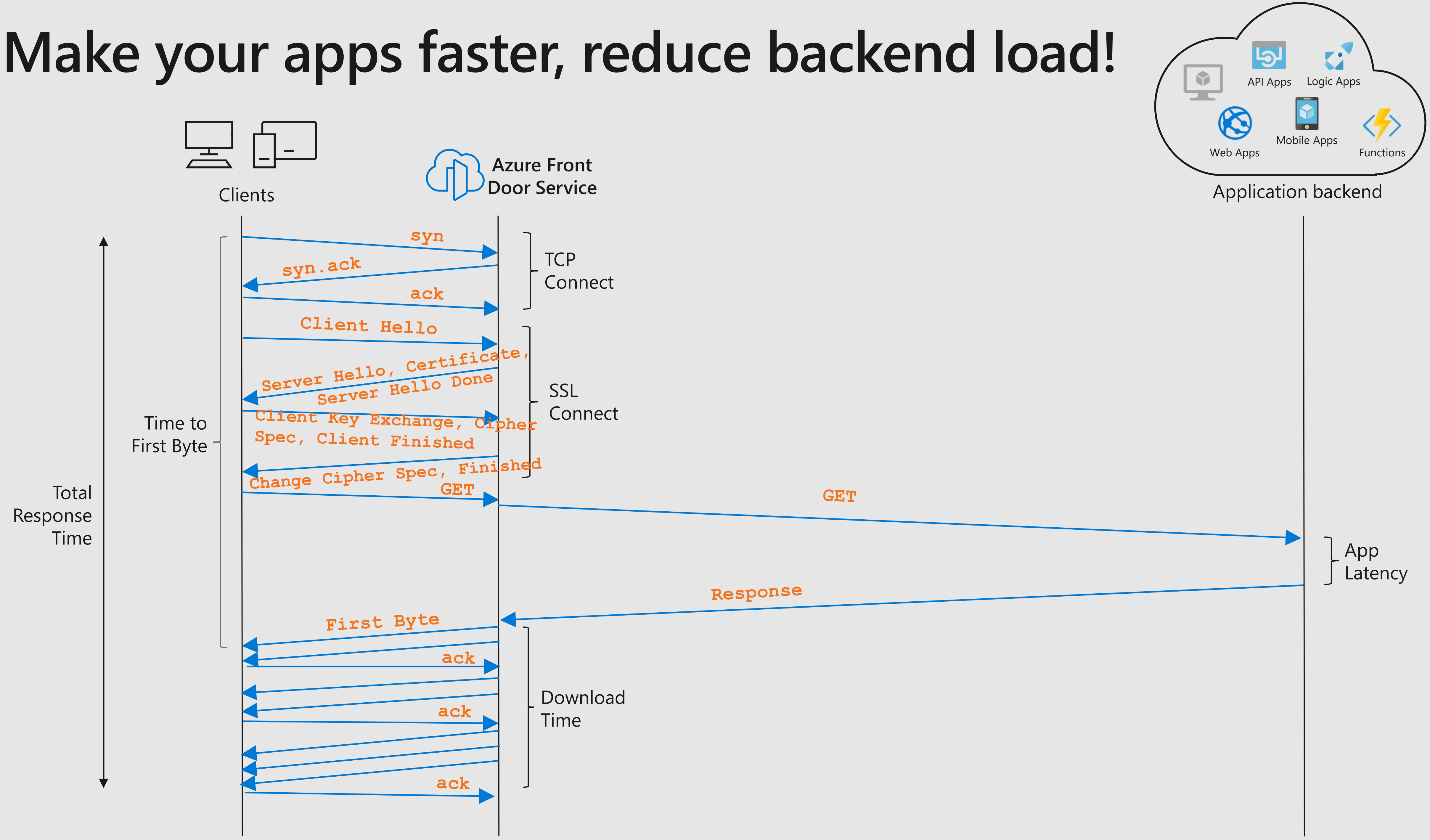
Preview

https://azure.com/frontdoor

Connection establishment and response



Make your apps faster, reduce backend load!



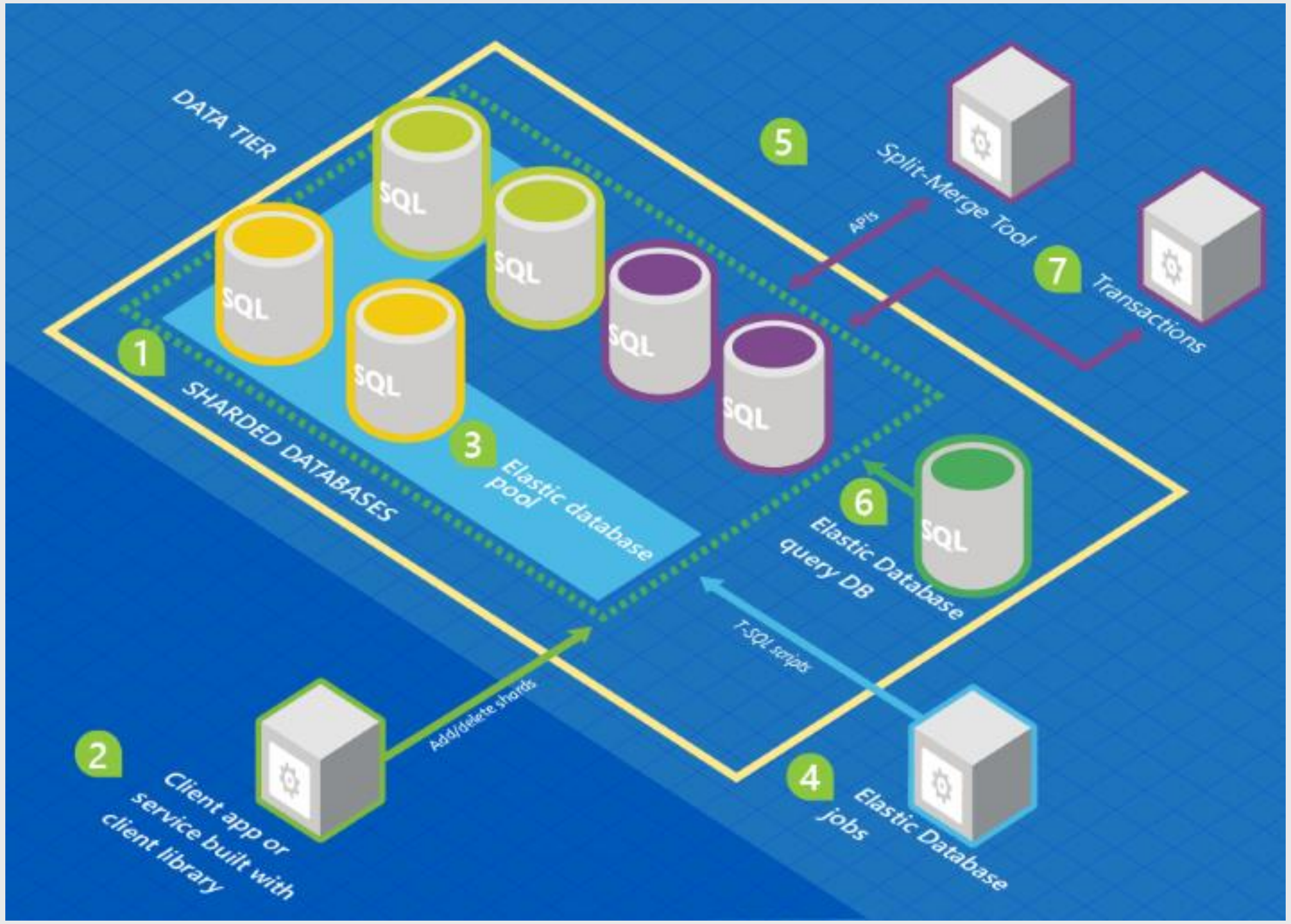
Demo

Azure Front Door

Scaling out with Azure SQL Database

Elastic Database Tools

1. Azure SQL DBs – Shard set
2. Elastic Database client library
3. Subset of DBs can be put into an Elastic pool
4. Elastic Database job runs scheduled or ad hoc T-SQL scripts against all databases
5. Split-merge tool to move data from one shard to another
6. Elastic Database query to write a query that spans all databases in the shard set
7. Elastic transactions to run transactions that span several databases

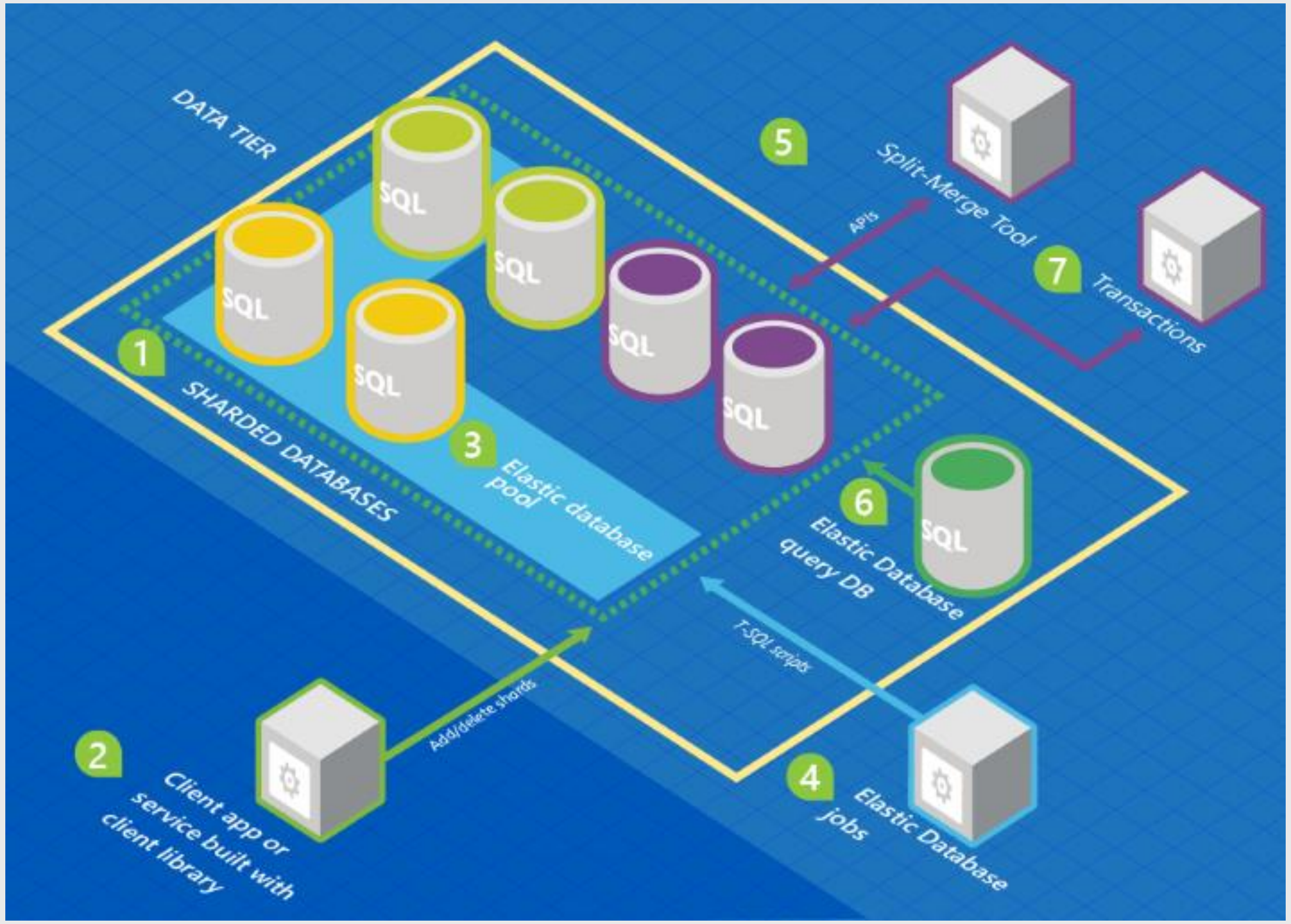


https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-scale-introduction

Scaling out with Azure SQL Database

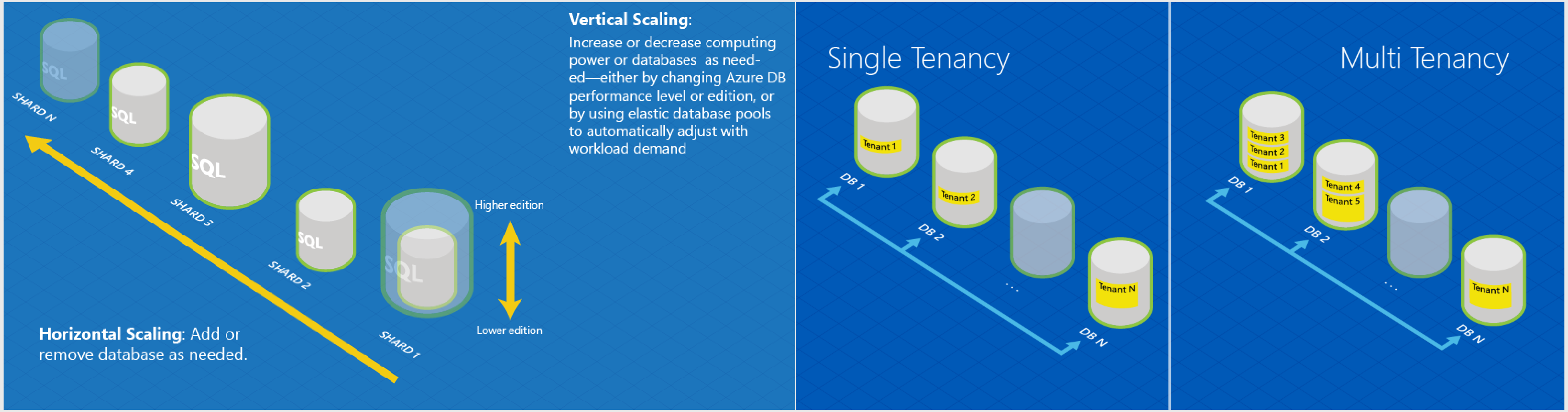
Elastic Database Tools

1. Azure SQL DBs – Shard set
2. Elastic Database client library
3. Subset of DBs can be put into an Elastic pool
4. Elastic Database job runs scheduled or ad hoc T-SQL scripts against all databases
5. Split-merge tool to move data from one shard to another
6. Elastic Database query to write a query that spans all databases in the shard set
7. Elastic transactions to run transactions that span several databases



https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-scale-introduction

Sharding



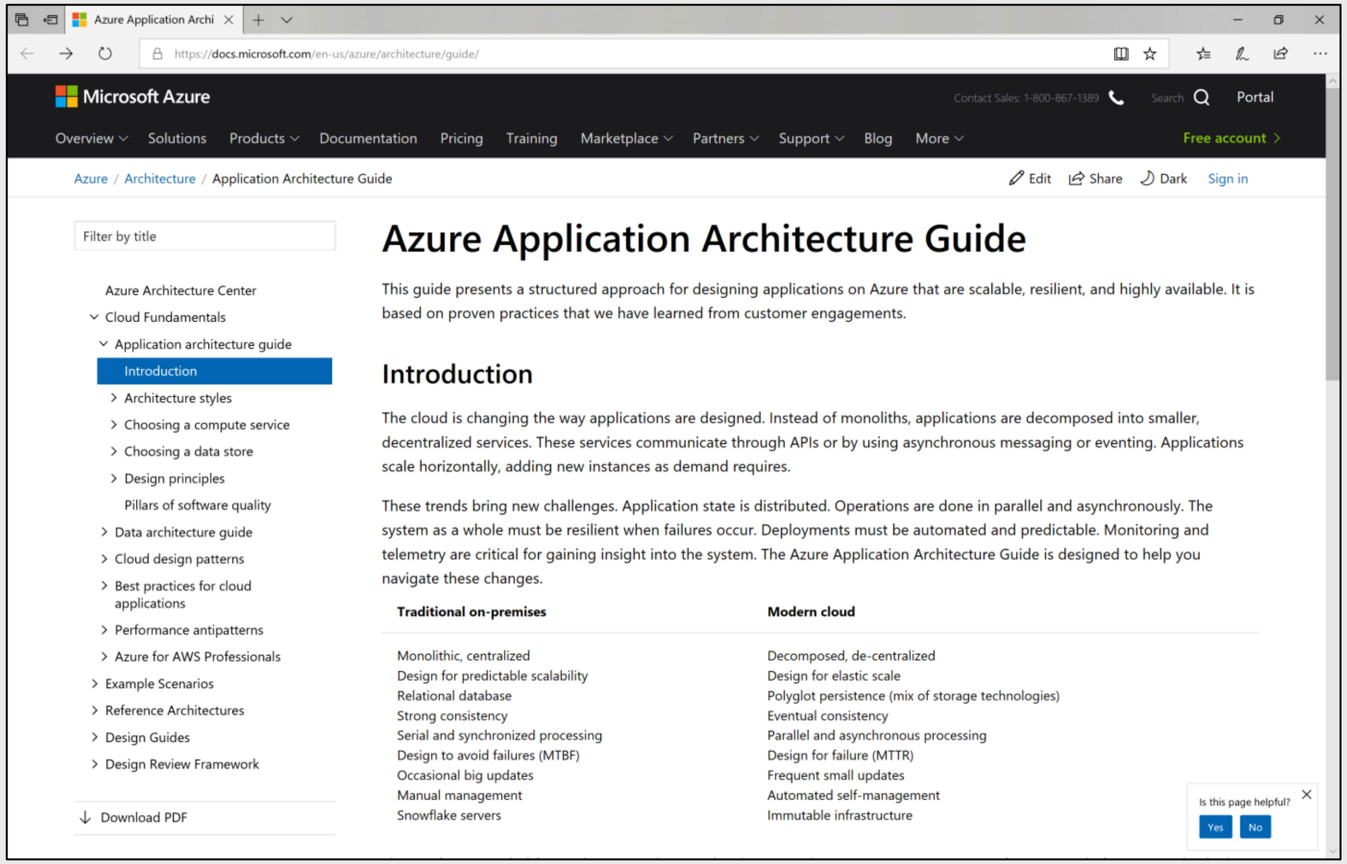
Vertical Scaling: Increase or decrease computing power or databases as needed—either by changing Azure DB performance level or edition, or by using elastic database pools to automatically adjust with workload demand

Horizontal Scaling: Add or remove database as needed.

https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-scale-introduction

Wrapping up

Azure Application Architecture Guide



Azure Application Architecture Guide

This guide presents a structured approach for designing applications on Azure that are scalable, resilient, and highly available. It is based on proven practices that we have learned from customer engagements.

Introduction

The cloud is changing the way applications are designed. Instead of monoliths, applications are decomposed into smaller, decentralized services. These services communicate through APIs or by using asynchronous messaging or eventing. Applications scale horizontally, adding new instances as demand requires.

These trends bring new challenges. Application state is distributed. Operations are done in parallel and asynchronously. The system as a whole must be resilient when failures occur. Deployments must be automated and predictable. Monitoring and telemetry are critical for gaining insight into the system. The Azure Application Architecture Guide is designed to help you navigate these changes.

Traditional on-premises	Modern cloud
Monolithic, centralized	Decomposed, de-centralized
Design for predictable scalability	Design for elastic scale
Relational database	Polyglot persistence (mix of storage technologies)
Strong consistency	Eventual consistency
Serial and synchronized processing	Parallel and asynchronous processing
Design to avoid failures (MTBF)	Design for failure (MTTR)
Occasional big updates	Frequent small updates
Manual management	Automated self-management
Snowflake servers	Immutable infrastructure

Microsoft
Azure
Cloud Application Architecture Guide

https://aka.ms/architecture

Thank you!

Microsoft

https://github.com/DanielLarsenNZ/talks

Image credits

https://commons.wikimedia.org/wiki/File:Glorious-blue-mountain-range.jpg