

Microsoft Azure Microservices/ Service Fabric



Authorized & published by Summitworks Technologies Inc

Agenda

- **Microservices**
 - Designing, building, and operating microservices on Azure
 - Introduction to Microservices
 - Domain analysis
 - Identifying microservice boundaries
 - Data considerations
 - Interservice communication
 - API design
 - API gateways
 - Logging and monitoring
 - CI/CD
- **Azure services**
 - Compute services
 - Azure Service Fabric

Microservices

What is a Microservice?

Microservice is a service-based application development methodology. In this methodology, **big applications will be divided into smallest independent service units**. Microservice is the process of implementing Service-oriented Architecture (SOA) by dividing the entire application as a collection of interconnected services, **where each service will serve only one business need**.

Following are some rules that we need to keep in mind while developing a Microservice-oriented application.

- **Independent** – **Each microservice should be independently deployable.**
- **Coupling** – All microservices should be loosely coupled with one another such that **changes in one will not affect the other.**
- **Business Goal** – **Each service unit of the entire application should be the smallest and capable of delivering one specific business goal.**

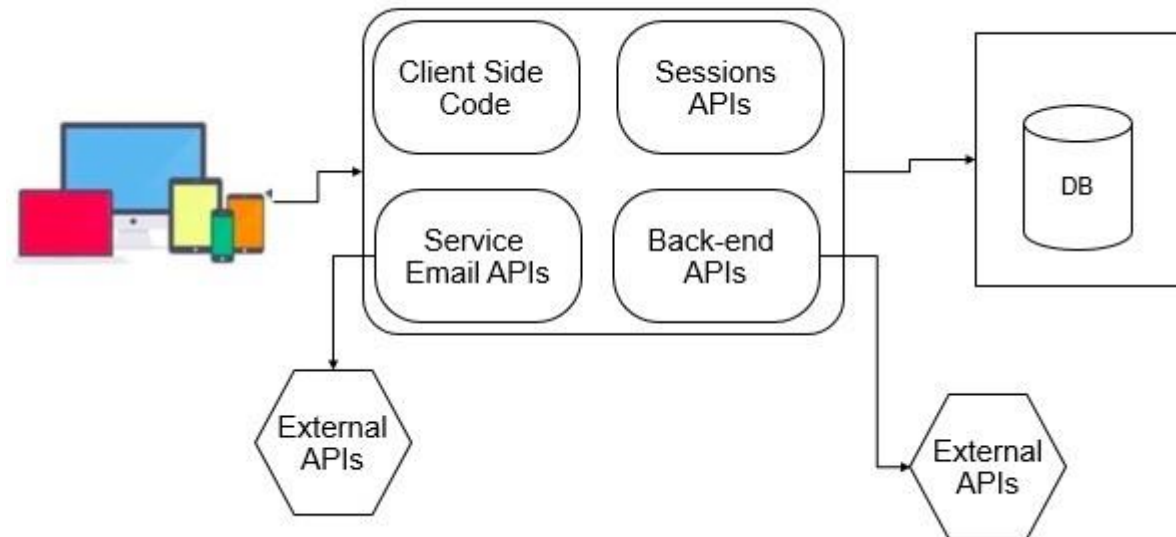
What is a Microservice?

Let us consider an example of **online shopping portal** to understand microservice in depth. Now, let us break this entire **E-commerce portal into small business units** such as user management, order management, check-in, payment management, delivery management, etc. One successful order needs to proceed through all of these modules within a specific time frame. **Following is the consolidated image of different business units associated with one electronic commerce system.**



What is a Microservice?

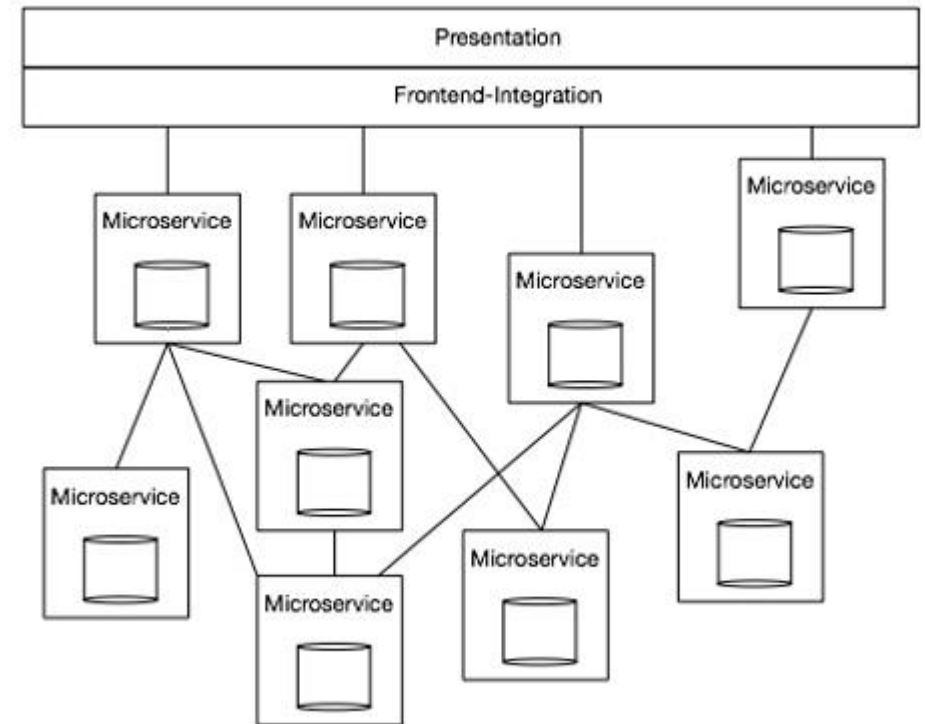
In this example image, you can see different modules such as Database for storing different users and business data. At the front-end, we have different device where we usually render user or business data to use. In the middle, we have one package that can be a deployable EAR or WAR file that accepts request from the users end, processes it with the help of the resources, and renders it back to the users. Everything will be fine until business wants any changes in the above example.



What is a Microservice?

In this architecture, we are dividing different parts of the software by exposing them as a service. Any part of the software can easily communicate with each other by consuming respective services. That's how microservice plays a great role in modern web application.

Let us compare our shopping cart example in the line of microservice. We can break down our shopping cart in the different modules such as “Search”, “Filter”, “Checkout”, “Cart”, “Recommendation”, etc.



Microservices Advantages

Small in size

Basically, a service should not perform more than one business task, hence it will be obviously small in size and easy to maintain.

Focused

Each microservice is designed to deliver only one business task. While designing a microservice, the architect should be concerned about the focal point of the service, which is its deliverable.

Autonomous

Each microservice should be an autonomous business unit of the entire application. Hence, the application becomes more loosely coupled, which helps to reduce the maintenance cost.

Microservices Advantages

Technology heterogeneity

Microservices supports different technologies to communicate with each other in one business unit, which helps the developers to use the correct technology at the correct place. By implementing a heterogeneous system, one can obtain maximum security, speed and a scalable system.

Resilience

Microservices follows high level of resilience in building methodology, hence whenever one unit fails it does not impact the entire business. Resilience is another property which implements highly scalable and less coupled system.

Ease of deployment

As the entire application is sub-divided into small piece of units, all of them can be deployed in any environment very easily with less time complexity.

Microservices Disadvantages

Distributed system

Different technologies will be used to develop different parts of a microservice. A huge set of skilled professionals are required to support this big heterogeneous distributed software.

Cost

Microservice is costly, as you have to maintain different server space for different business tasks.

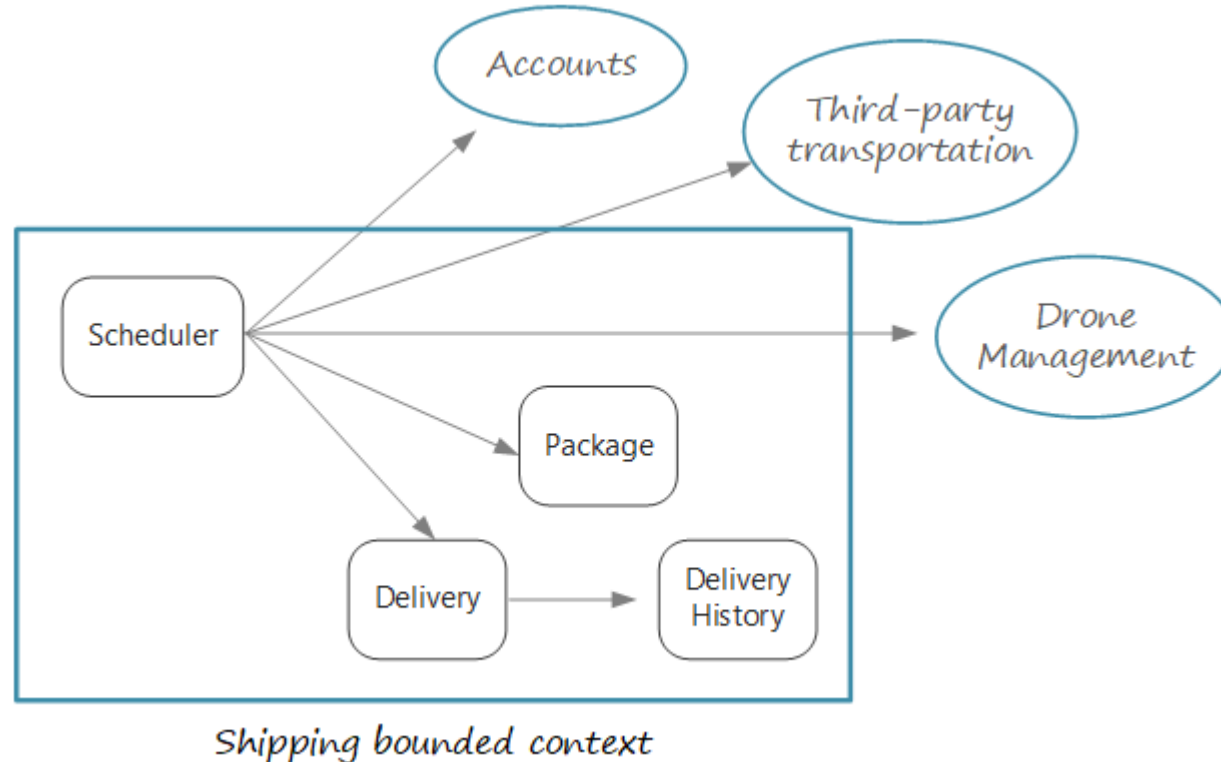
Enterprise readiness

Microservice architecture can be considered as a conglomerate of different technologies, as technology is evolving day-by-day. Hence, it is quite difficult to make a microservice application enterprise ready to compare to conventional software development model.

Identifying Microservices boundaries

Microservices boundaries

What is the right size for a microservice? You often hear something to the effect of, "not too big and not too small" — and while that's certainly correct, it's not very helpful in practice. But **if you start from a carefully designed domain model, it's much easier to reason about microservices.**



Microservices boundaries

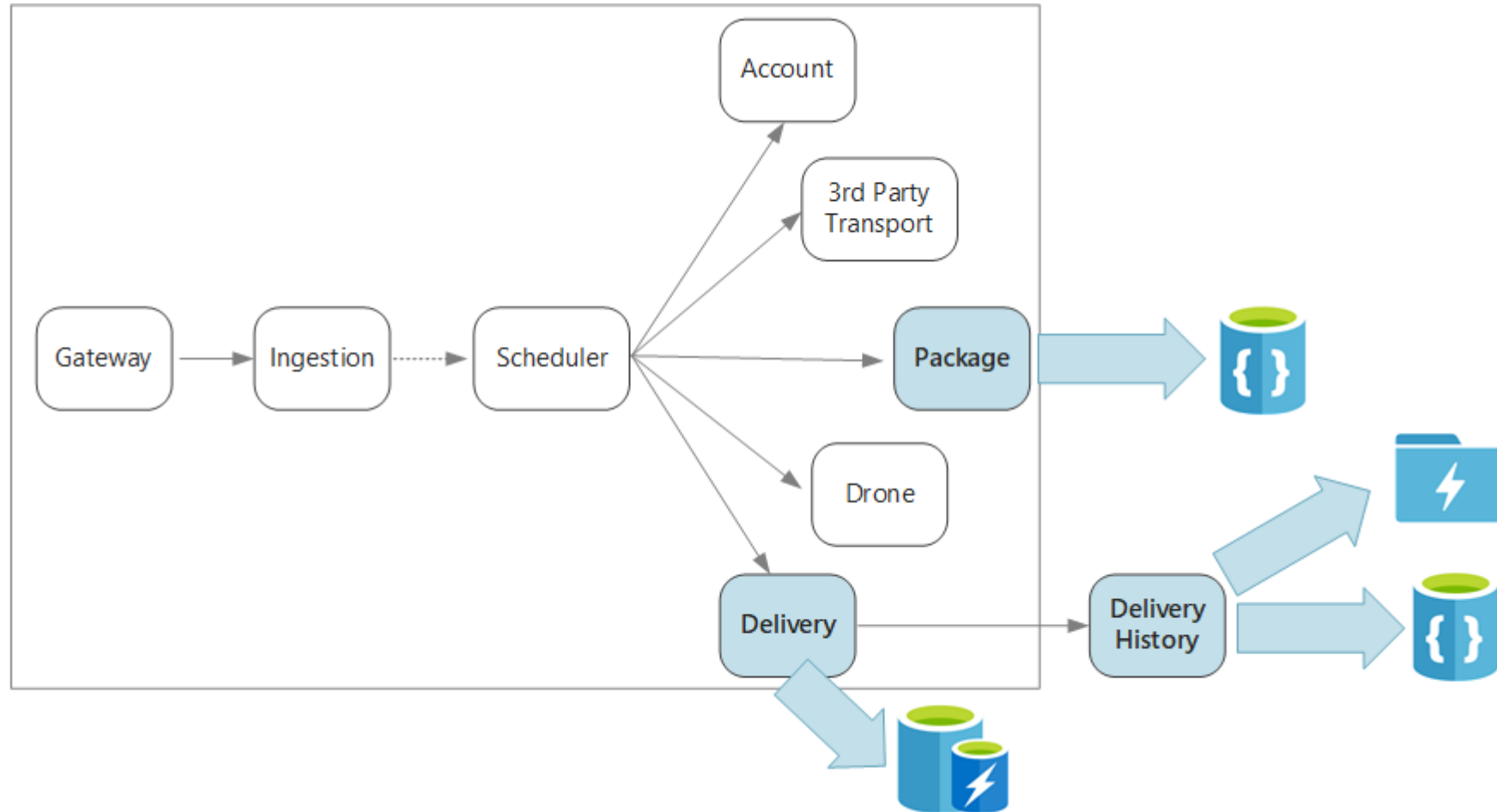
After you identify the microservices in your application, validate your design against the following criteria:

- Each service has a single responsibility.
- There are no chatty calls between services. If splitting functionality into two services causes them to be overly chatty, it may be a symptom that these functions belong in the same service.
- Each service is small enough that it can be built by a small team working independently.

Data Considerations

Data Considerations

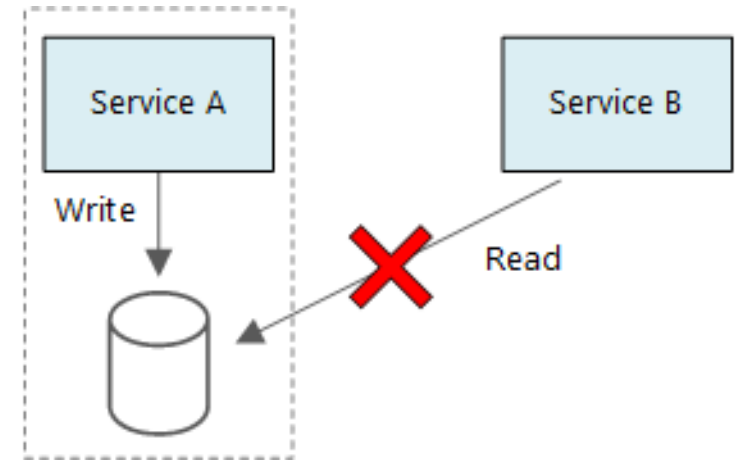
Because every microservice manages its own data, data integrity and data consistency are critical challenges.



Data Considerations

A basic principle of microservices is that each service manages its own data. Two services should not share a data store. Instead, each service is responsible for its own private data store, which other services cannot access directly.

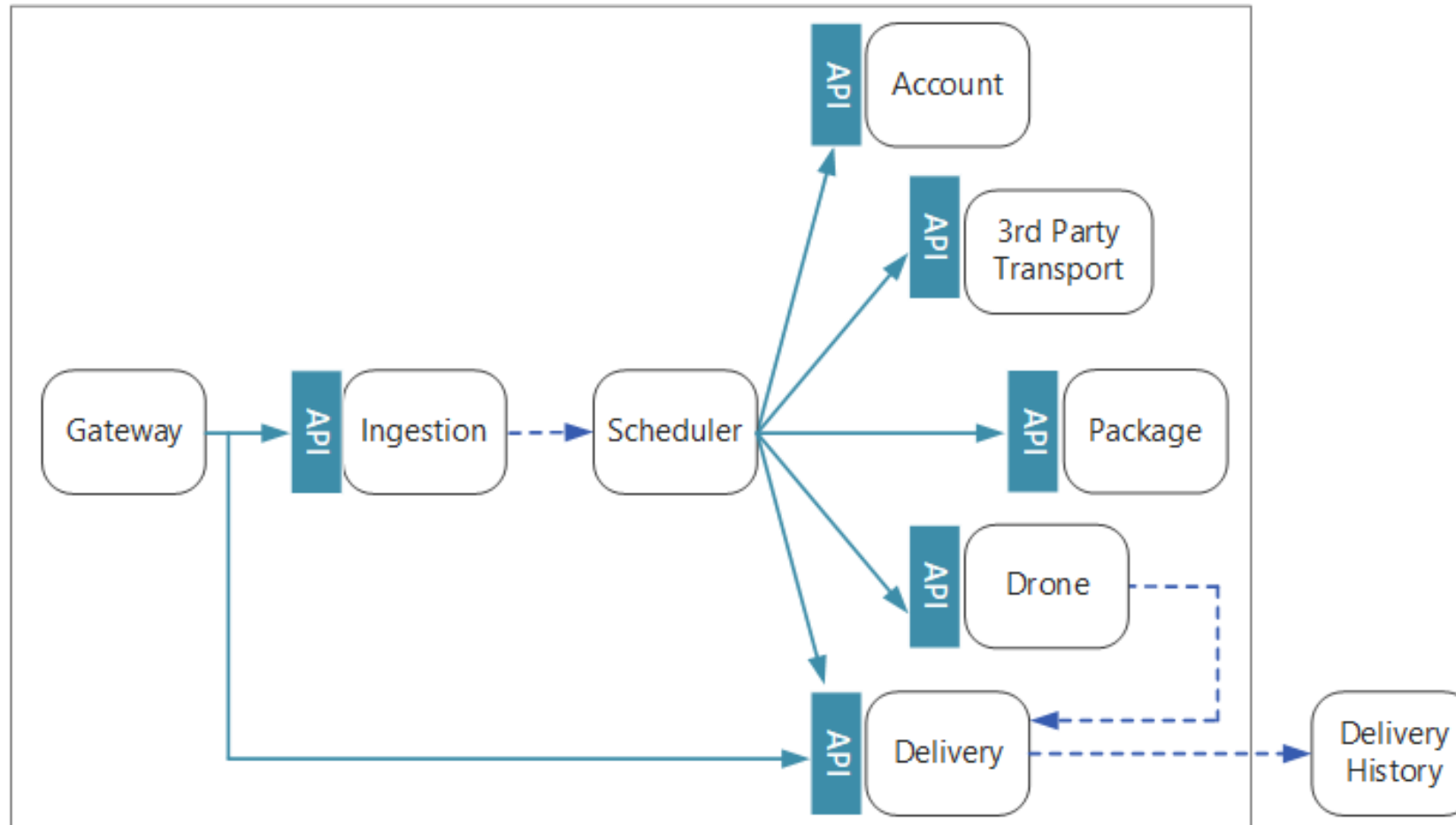
The reason for this rule is to avoid unintentional coupling between services, which can result if services share the same underlying data schemas. If there is a change to the data schema, the change must be coordinated across every service that relies on that database. Each microservice may have its own data models, queries, or read/write patterns. Using a shared data store limits each team's ability to optimize data storage for their particular service.



Interservice communication

Interservice communication

Communication between microservices must be efficient and robust. With lots of small services interacting to complete a single transaction, this can be a challenge.



Interservice communication

Synchronous versus asynchronous messaging

There are two basic messaging patterns that microservices can use to communicate with other microservices.

- **Synchronous communication**

In this pattern, a service calls an API that another service exposes, using a protocol such as HTTP or gRPC. This option is a synchronous messaging pattern because the caller waits for a response from the receiver.

- **Asynchronous message passing.**

In this pattern, a service sends message without waiting for a response, and one or more services process the message asynchronously.

API Gateways

API Gateways

What is an API gateway?

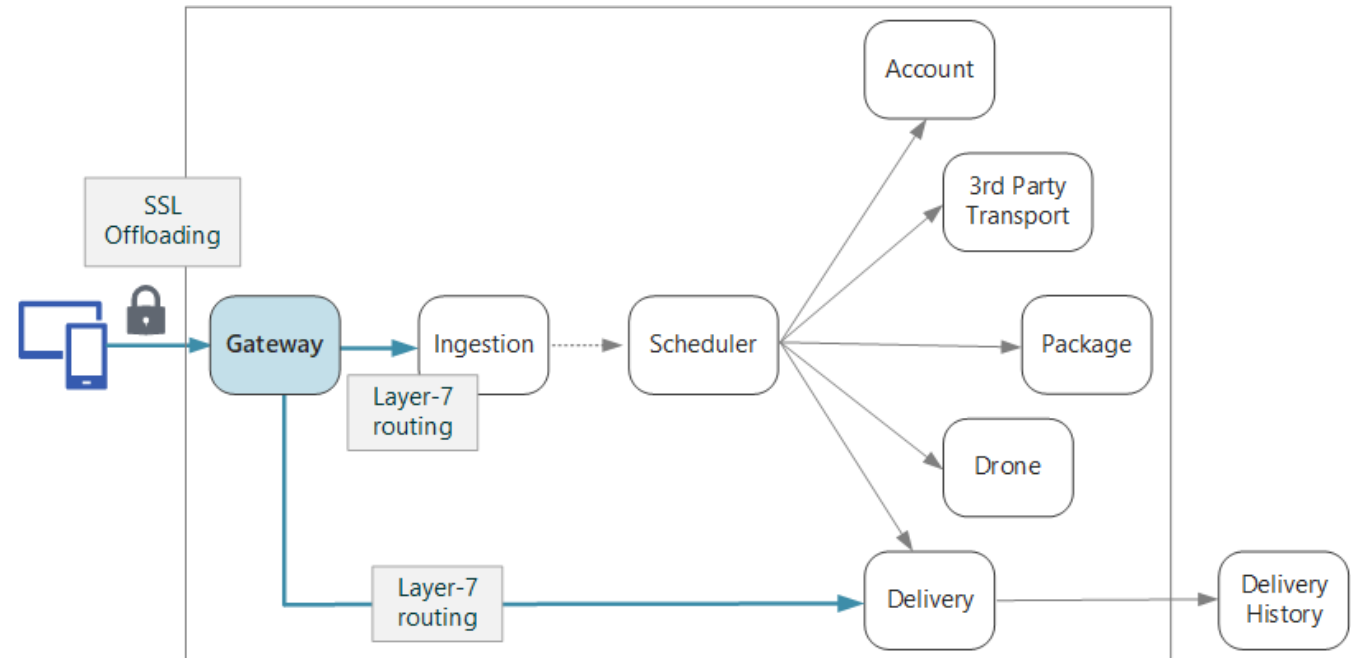
In a microservices architecture, a client might interact with more than one front-end service. Given this fact, how does a client know what endpoints to call? What happens when new services are introduced, or existing services are refactored? How do services handle SSL termination, authentication, and other concerns? An *API gateway* can help to address these challenges.

An API gateway sits between clients and services. It acts as a reverse proxy, routing requests from clients to services. It may also perform various cross-cutting tasks such as authentication, SSL termination, and rate limiting. If you don't deploy a gateway, clients must send requests directly to front-end services.

API Gateways

Here are some examples of functionality that could be offloaded to a gateway:

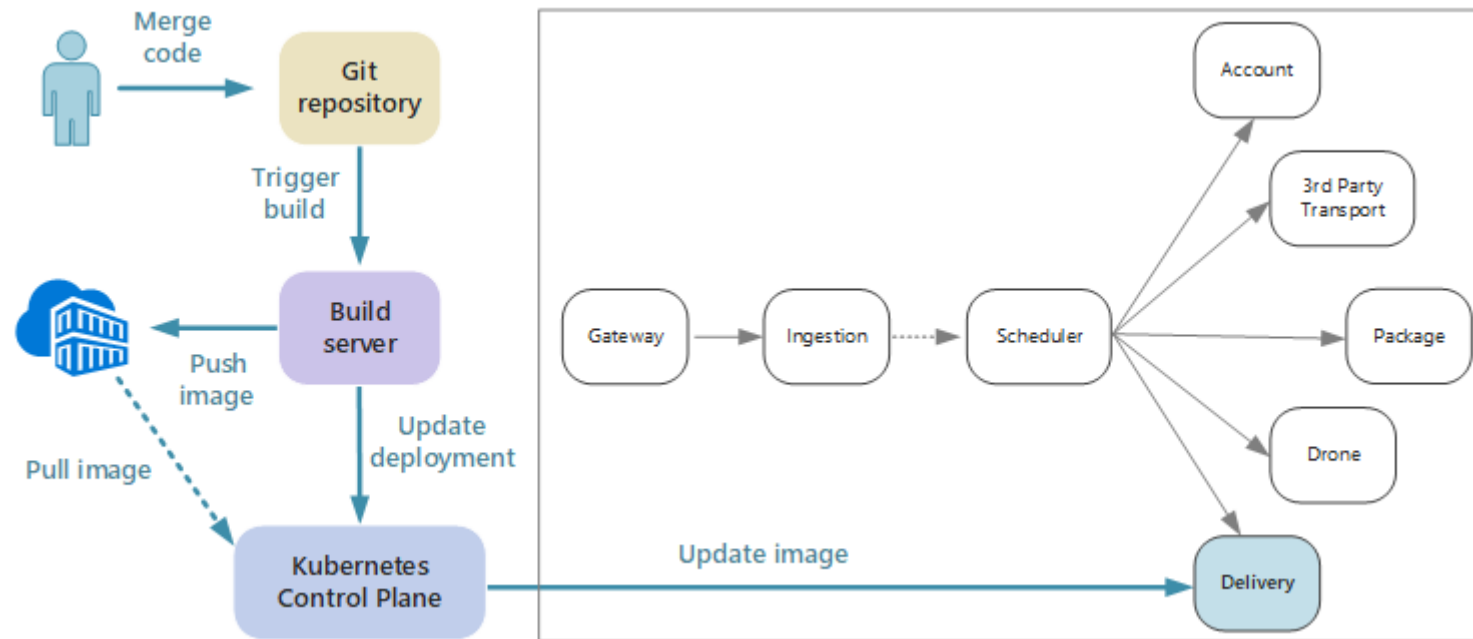
- SSL termination
- Authentication
- IP whitelisting
- Client rate limiting (throttling)
- Logging and monitoring
- Response caching
- Web application firewall
- GZIP compression
- Servicing static content



Continuous Integration / Continuous Delivery (CI/CD)

CI/CD

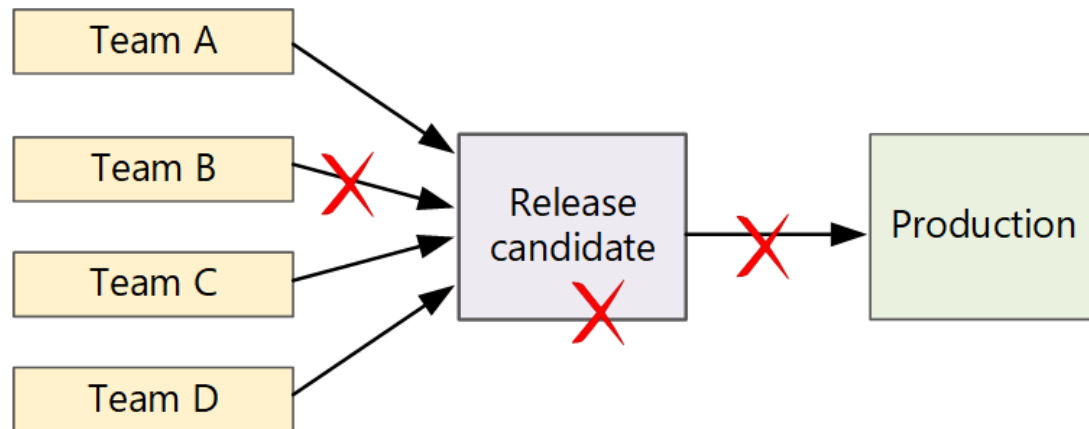
Continuous integration and continuous delivery (CI/CD) are a key requirement for achieving success with microservices. Without a good CI/CD process, you will not achieve the agility that microservices promise. Some of the CI/CD challenges for microservices arise from having multiple code bases and heterogenous build environments for the various services.



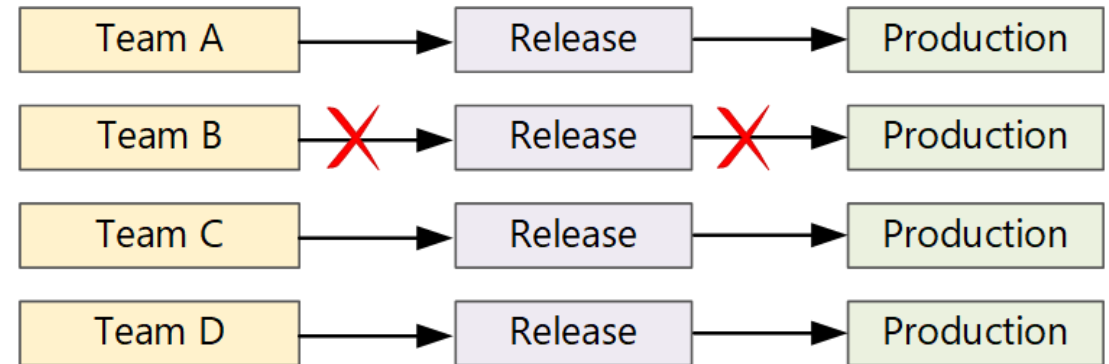
Faster release cycles are one of the biggest reasons to adopt a microservices architecture.

CI/CD

Following the microservices philosophy, there should never be a long release train where every team has to get in line. **The team that builds service "A" can release an update at any time, without waiting for changes in service "B" to be merged, tested, and deployed.** The CI/CD process is critical to making this possible. Your release pipeline must be automated and highly reliable, so that the risks of deploying updates are minimized. If you are releasing to production daily or multiple times a day, regressions or service disruptions must be very rare. **At the same time, if a bad update does get deployed, you must have a reliable way to quickly roll back or roll forward to a previous version of a service.**



Monolith



Microservices

CI/CD

When we talk about CI/CD, we are really talking about several related processes: Continuous integration, continuous delivery, and continuous deployment.

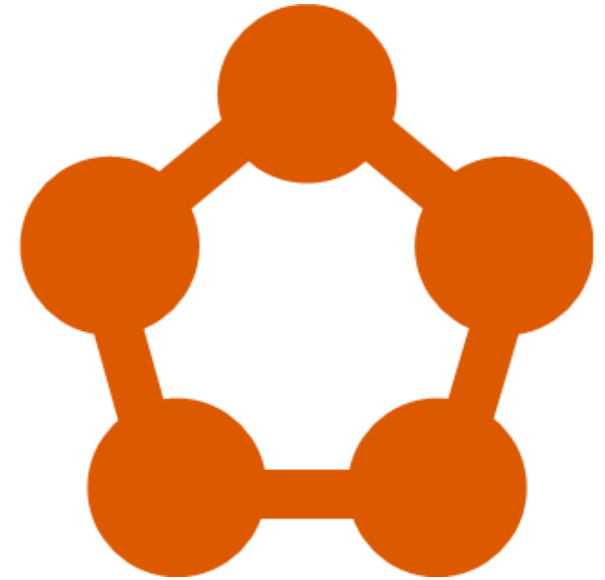
- Continuous integration means that code changes are frequently merged into the main branch, using automated build and test processes to ensure that code in the main branch is always production-quality.
- Continuous delivery means that code changes that pass the CI process are automatically published to a production-like environment. Deployment into the live production environment may require manual approval, but is otherwise automated. The goal is that your code should always be *ready* to deploy into production.
- Continuous deployment means that code changes that pass the CI/CD process are automatically deployed into production.

Azure Compute Services - Service Fabric

Azure Service Fabric

Azure Service Fabric is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices and containers. Service Fabric also addresses the significant challenges in developing and managing cloud native applications. Developers and administrators can avoid complex infrastructure problems and focus on implementing mission-critical, demanding workloads that are scalable, reliable, and manageable. Service Fabric represents the next-generation platform for building and managing these enterprise-class, tier-1, cloud-scale applications running in containers.

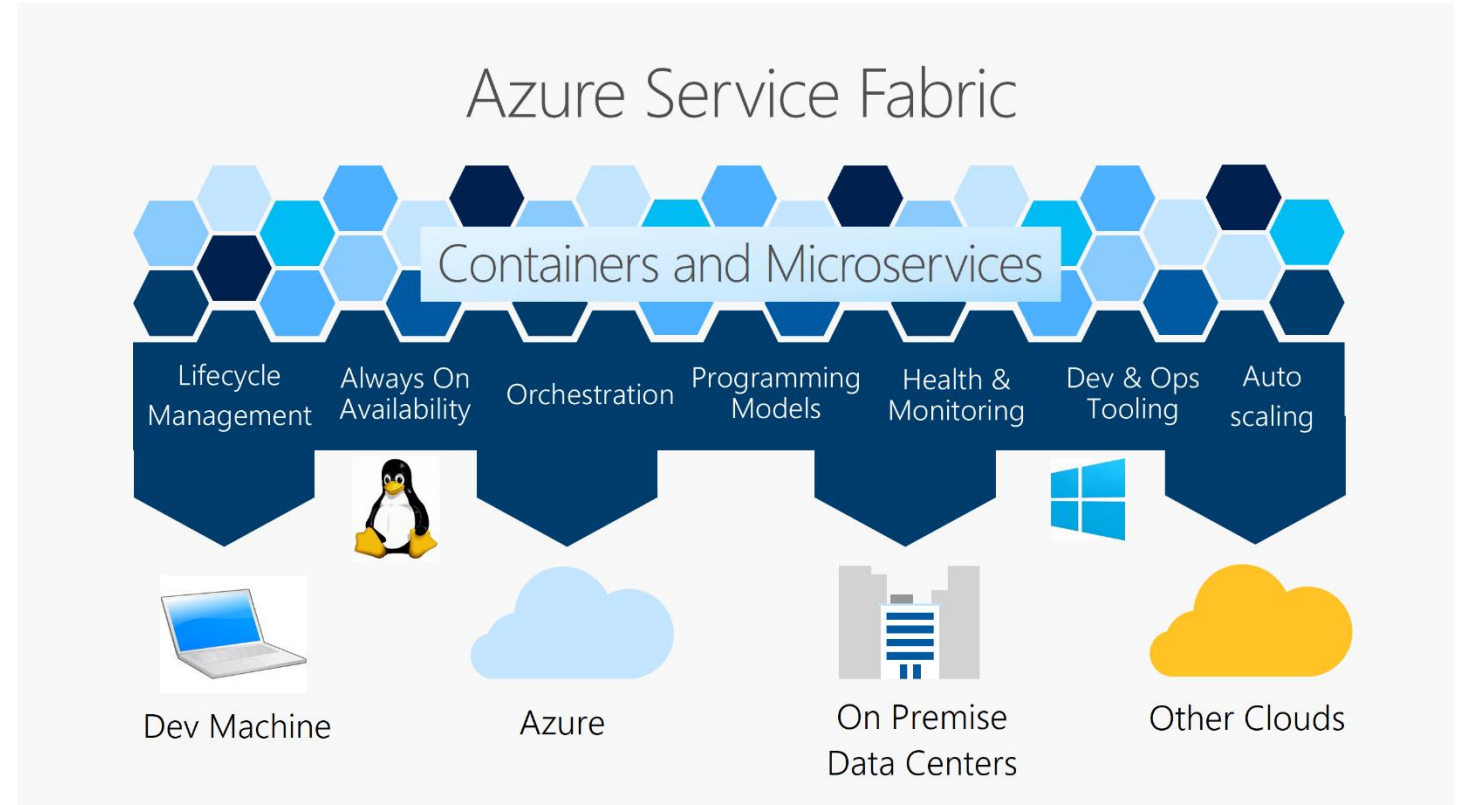
Service Fabric powers many Microsoft services today, including *Azure SQL Database, Azure Cosmos DB, Cortana, Microsoft Power BI, Microsoft Intune, Azure Event Hubs, Azure IoT Hub, Dynamics 365, Skype for Business, and many core Azure services.*



Azure Service Fabric

Any OS, any cloud

Service Fabric runs everywhere. You can create clusters for Service Fabric in many environments, including Azure or on premises, on Windows Server, or on Linux. You can even create clusters on other public clouds. In addition, the development environment in the SDK is identical to the production environment, with no emulators involved. In other words, what runs on your local development cluster deploys to the clusters in other environments.



Azure Service Fabric

Key capabilities

- Deploy to Azure or to on-premises datacenters that run Windows or Linux with zero code changes. Write once, and then deploy anywhere to any Service Fabric cluster.
- Develop scalable applications that are composed of microservices by using the Service Fabric programming models, containers, or any code.
- Develop highly reliable stateless and stateful microservices. Simplify the design of your application by using stateful microservices.
- Deploy applications in seconds, at high density with hundreds or thousands of applications or containers per machine.

Azure Service Fabric

Key capabilities

- Deploy different versions of the same application side by side, and upgrade each application independently.
- Scale out or scale in the number of nodes in a cluster. As you scale nodes, your applications automatically scale.
- Monitor and diagnose the health of your applications and set policies for performing automatic repairs.

Deploy a .NET app to Service Fabric

Azure Service Fabric is a distributed systems platform for deploying and managing scalable and reliable microservices and containers.

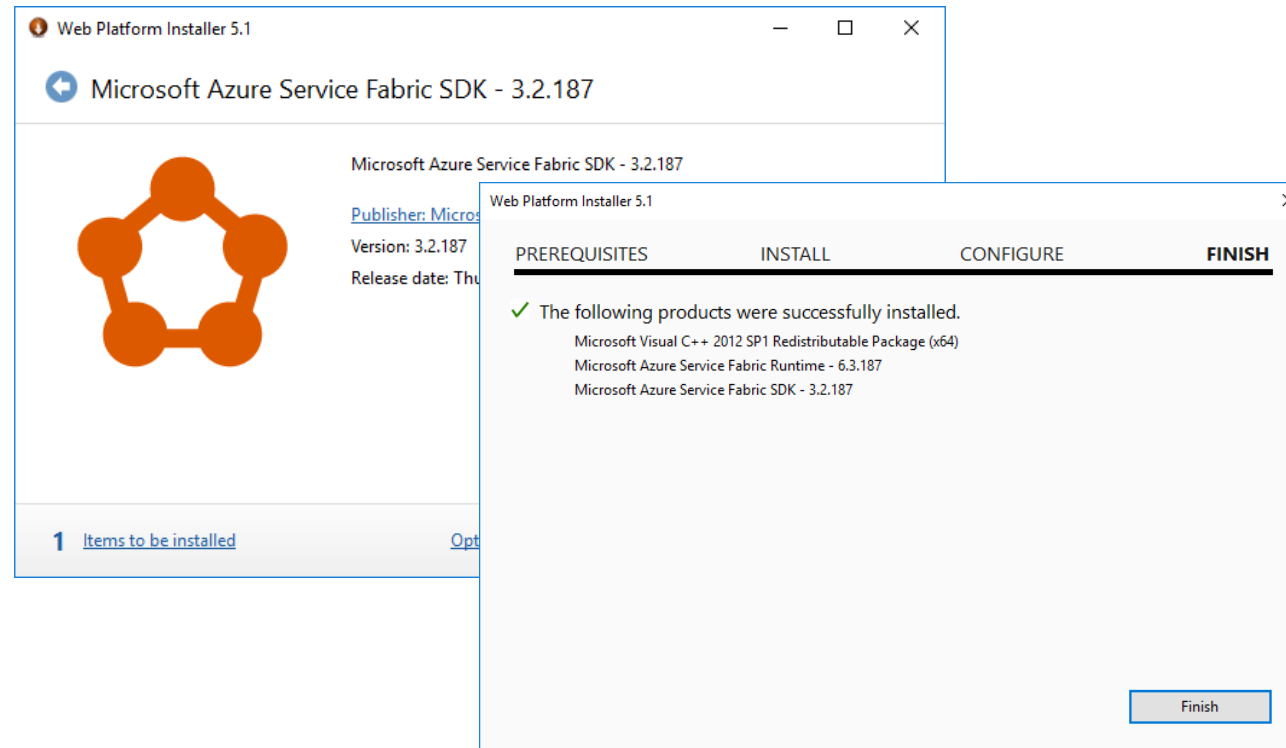
Using this application you learn how to:

- Create an application using .NET and Service Fabric
- Use ASP.NET core as a web front-end
- Store application data in a stateful service
- Debug your application locally
- Deploy the application to a cluster in Azure
- Scale-out the application across multiple nodes
- Perform a rolling application upgrade

Deploy a .NET app to Service Fabric

Install the Microsoft Azure Service Fabric SDK

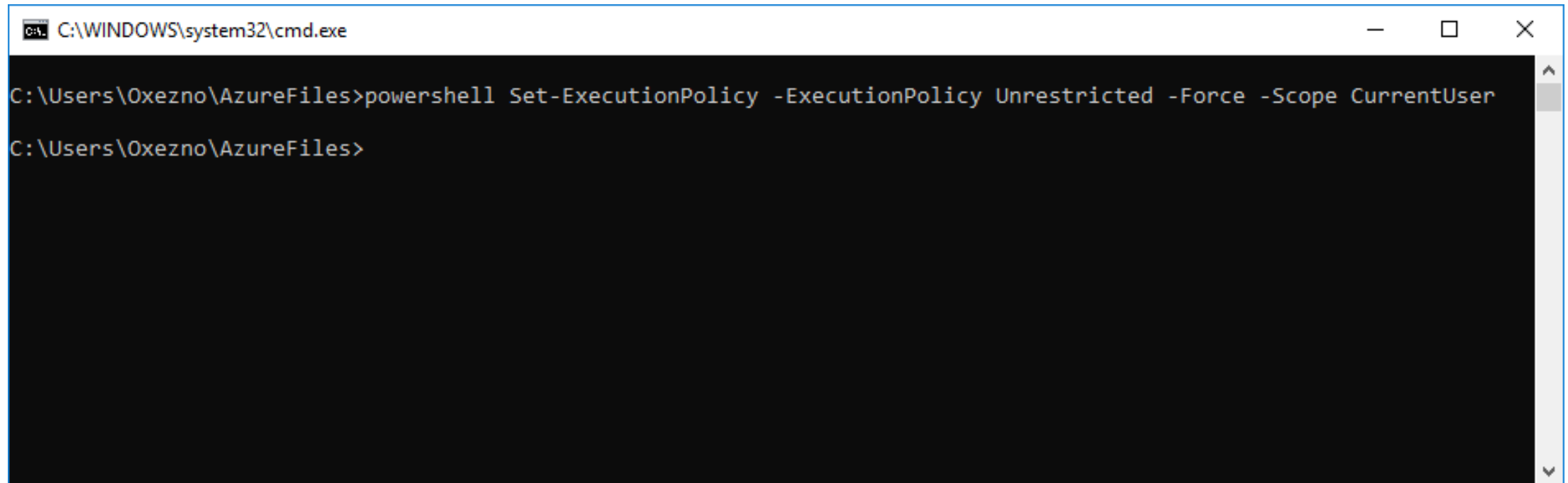
<https://www.microsoft.com/web/handlers/webpi.ashx?command=getinstallerredirect&appid=MicrosoftAzure-ServiceFabric-CoreSDK>



Deploy a .NET app to Service Fabric

Run the following command to enable Visual Studio to deploy to the local Service Fabric cluster:

`powershell Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser`



A screenshot of a Windows Command Prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The command prompt shows the current directory as 'C:\Users\Oxezno\AzureFiles'. The user has entered the command 'powershell Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser'. The prompt is now waiting for the next command.

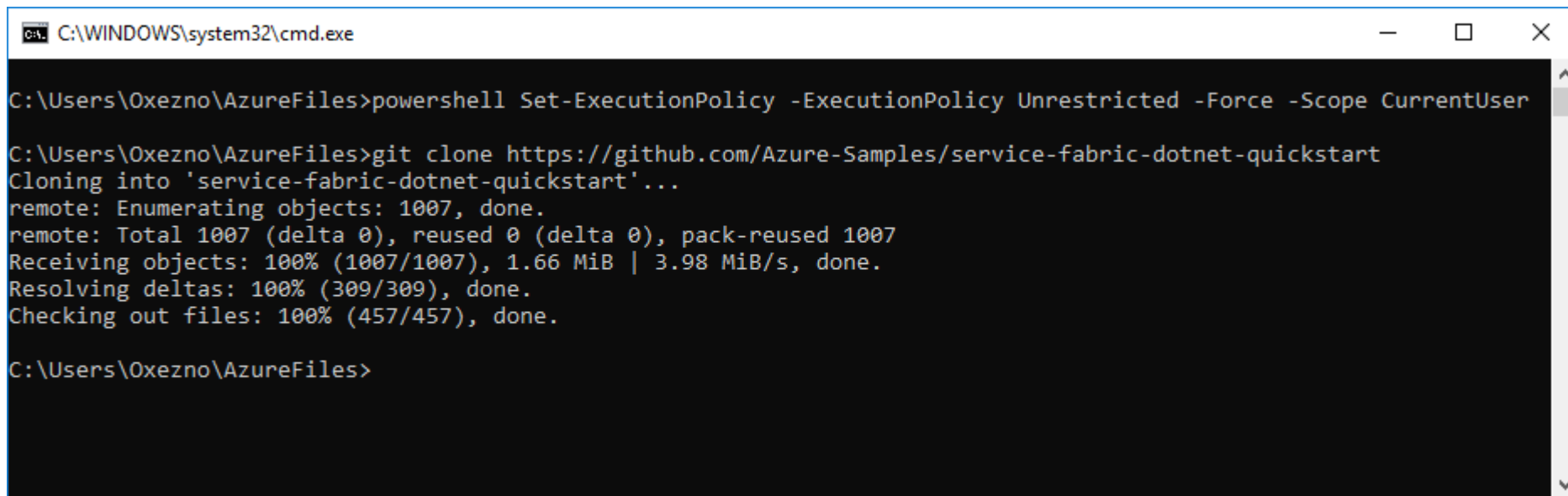
```
C:\WINDOWS\system32\cmd.exe
C:\Users\Oxezno\AzureFiles>powershell Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser
C:\Users\Oxezno\AzureFiles>
```

Deploy a .NET app to Service Fabric

Download the sample

In a command window, run the following command to clone the sample app repository to your local machine.

```
git clone https://github.com/Azure-Samples/service-fabric-dotnet-quickstart
```

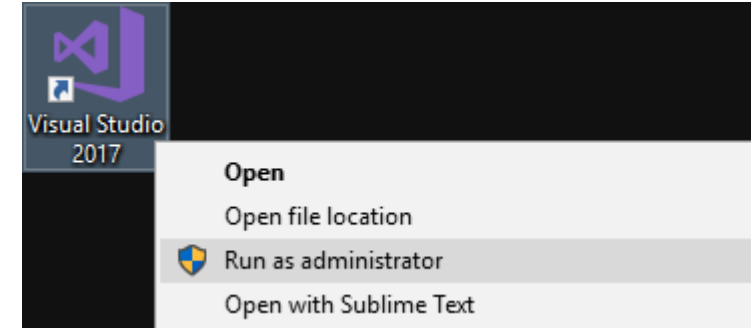
A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the execution of a git clone command. The prompt is "C:\Users\Oxezno\AzureFiles>". The command entered is "powershell Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser". The output shows the command being executed successfully. The prompt is "C:\Users\Oxezno\AzureFiles>". The command entered is "git clone https://github.com/Azure-Samples/service-fabric-dotnet-quickstart". The output shows the cloning process: "Cloning into 'service-fabric-dotnet-quickstart'...", "remote: Enumerating objects: 1007, done.", "remote: Total 1007 (delta 0), reused 0 (delta 0), pack-reused 1007", "Receiving objects: 100% (1007/1007), 1.66 MiB | 3.98 MiB/s, done.", "Resolving deltas: 100% (309/309), done.", "Checking out files: 100% (457/457), done.", "C:\Users\Oxezno\AzureFiles>".

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Oxezno\AzureFiles>powershell Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser
C:\Users\Oxezno\AzureFiles>git clone https://github.com/Azure-Samples/service-fabric-dotnet-quickstart
Cloning into 'service-fabric-dotnet-quickstart'...
remote: Enumerating objects: 1007, done.
remote: Total 1007 (delta 0), reused 0 (delta 0), pack-reused 1007
Receiving objects: 100% (1007/1007), 1.66 MiB | 3.98 MiB/s, done.
Resolving deltas: 100% (309/309), done.
Checking out files: 100% (457/457), done.
C:\Users\Oxezno\AzureFiles>
```

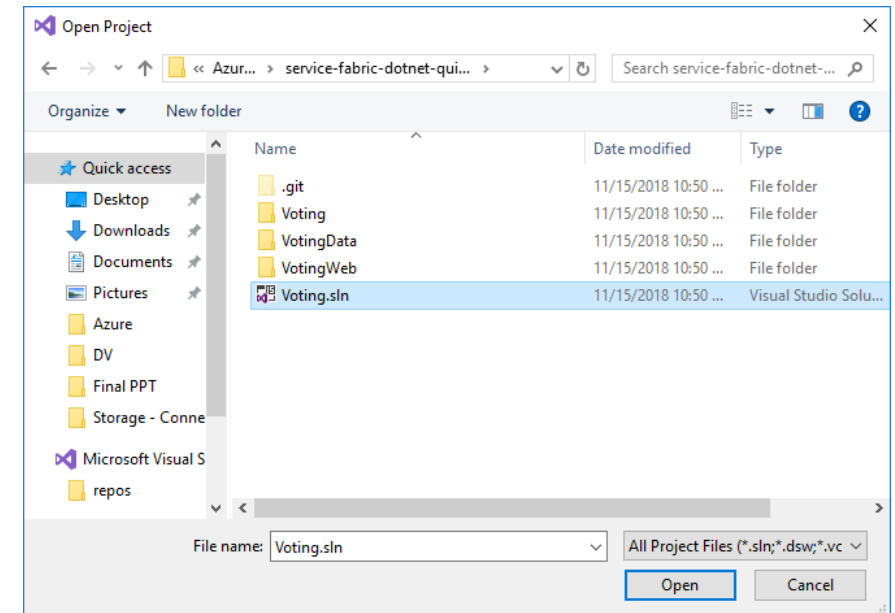
Deploy a .NET app to Service Fabric

Run the application locally

Right-click the Visual Studio icon in the Start Menu and choose **Run as administrator**. In order to attach the debugger to your services, you need to run Visual Studio as administrator.



Open the **Voting.sln** Visual Studio solution from the repository you cloned.



Deploy a .NET app to Service Fabric

By default, the Voting application is set to listen on port 8080. The application port is set in the */VotingWeb/PackageRoot/ServiceManifest.xml* file. You can change the application port by updating the **Port** attribute of the **Endpoint** element. To deploy and run the application locally, the application port must be open and available on your computer. If you change the application port, substitute the new application port value for "8080" throughout this article.

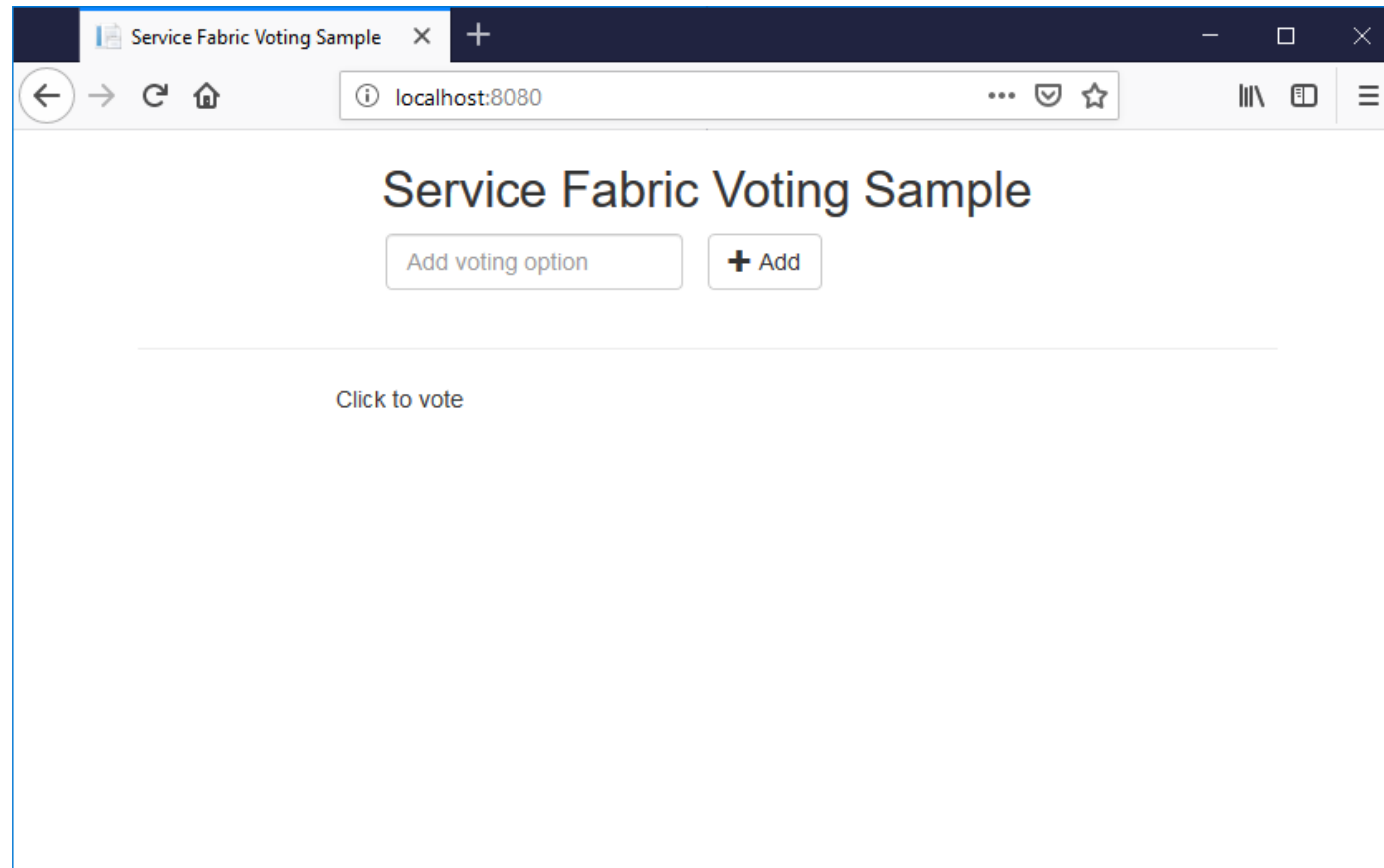
To deploy the application, press **F5**.

Note

The first time you run and deploy the application, Visual Studio creates a local cluster for debugging. This operation may take some time. The cluster creation status is displayed in the Visual Studio output window. In the output, you see the message "The application URL is not set or is not an HTTP/HTTPS URL so the browser will not be opened to the application." This message does not indicate an error, but that a browser will not auto-launch.

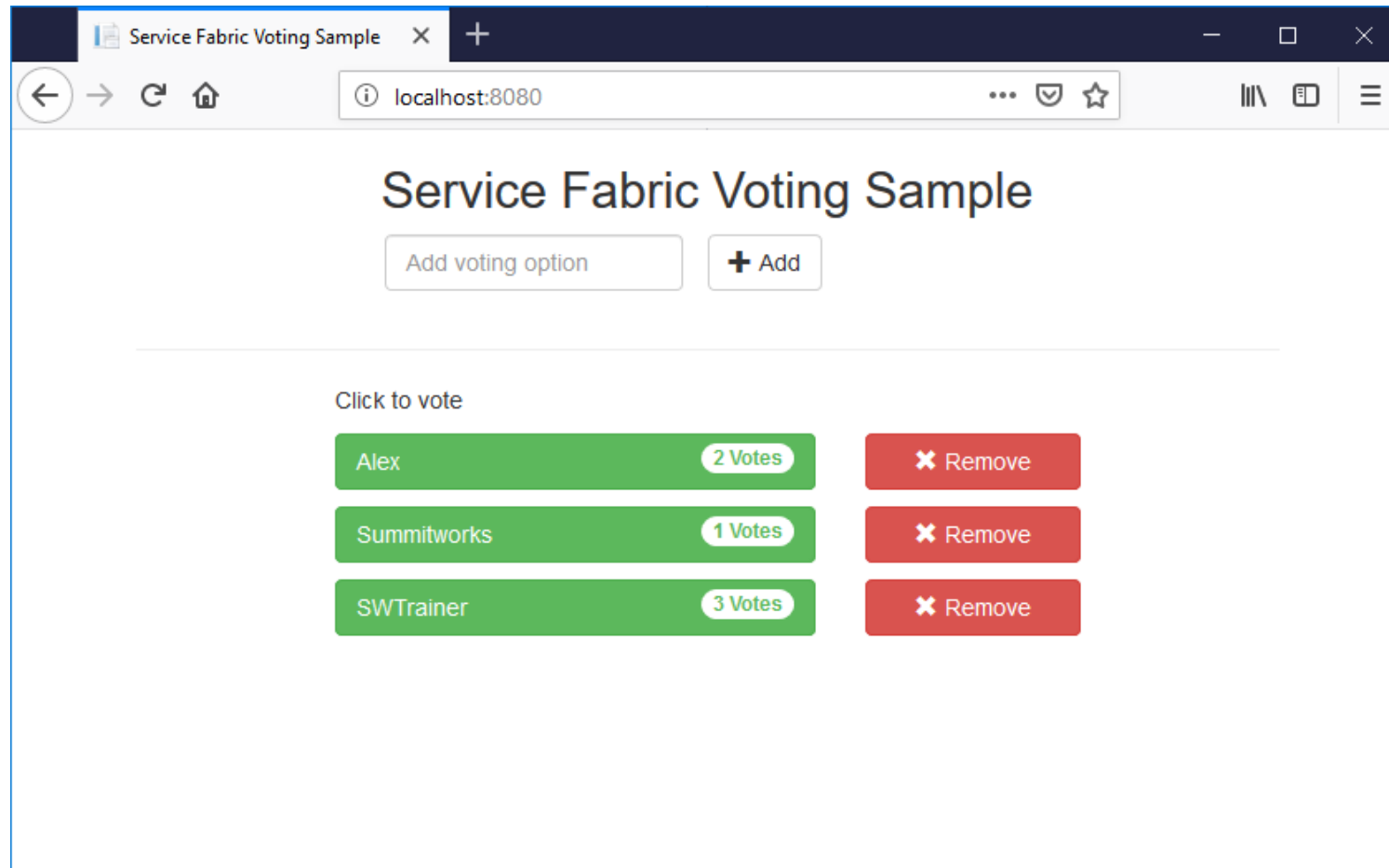
Deploy a .NET app to Service Fabric

When the deployment is complete, launch a browser and open this page: *http://localhost:8080* - the web front end of the application.



Deploy a .NET app to Service Fabric

You can now add a set of voting options, and start taking votes. The application runs and stores all data in your Service Fabric cluster, without the need for a separate database.

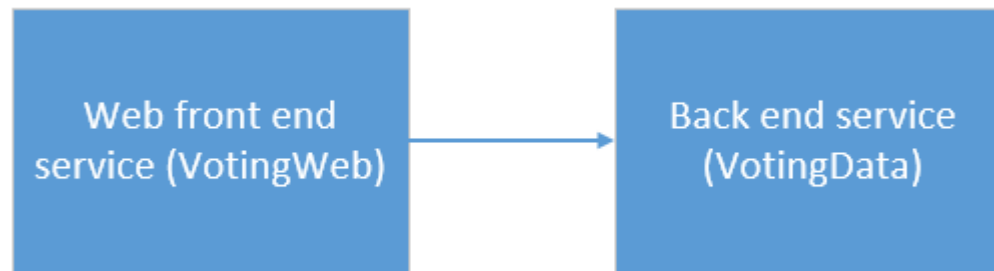


Deploy a .NET app to Service Fabric

Walk through the voting sample application

The voting application consists of two services:

- Web front-end service (VotingWeb)- An ASP.NET Core web front-end service, which serves the web page and exposes web APIs to communicate with the backend service.
- Back-end service (VotingData)- An ASP.NET Core web service, which exposes an API to store the vote results in a reliable dictionary persisted on disk.



Deploy a .NET app to Service Fabric

When you vote in the application the following events occur:

1. A JavaScript sends the vote request to the web API in the web front-end service as an HTTP PUT request.
2. The web front-end service uses a proxy to locate and forward an HTTP PUT request to the back-end service.
3. The back-end service takes the incoming request, and stores the updated result in a reliable dictionary, which gets replicated to multiple nodes within the cluster and persisted on disk. All the application's data is stored in the cluster, so no database is needed.

Deploy a .NET app to Service Fabric

Deploy the application to Azure

To deploy the application to Azure, you need a Service Fabric cluster which runs the application.

Join a Party cluster

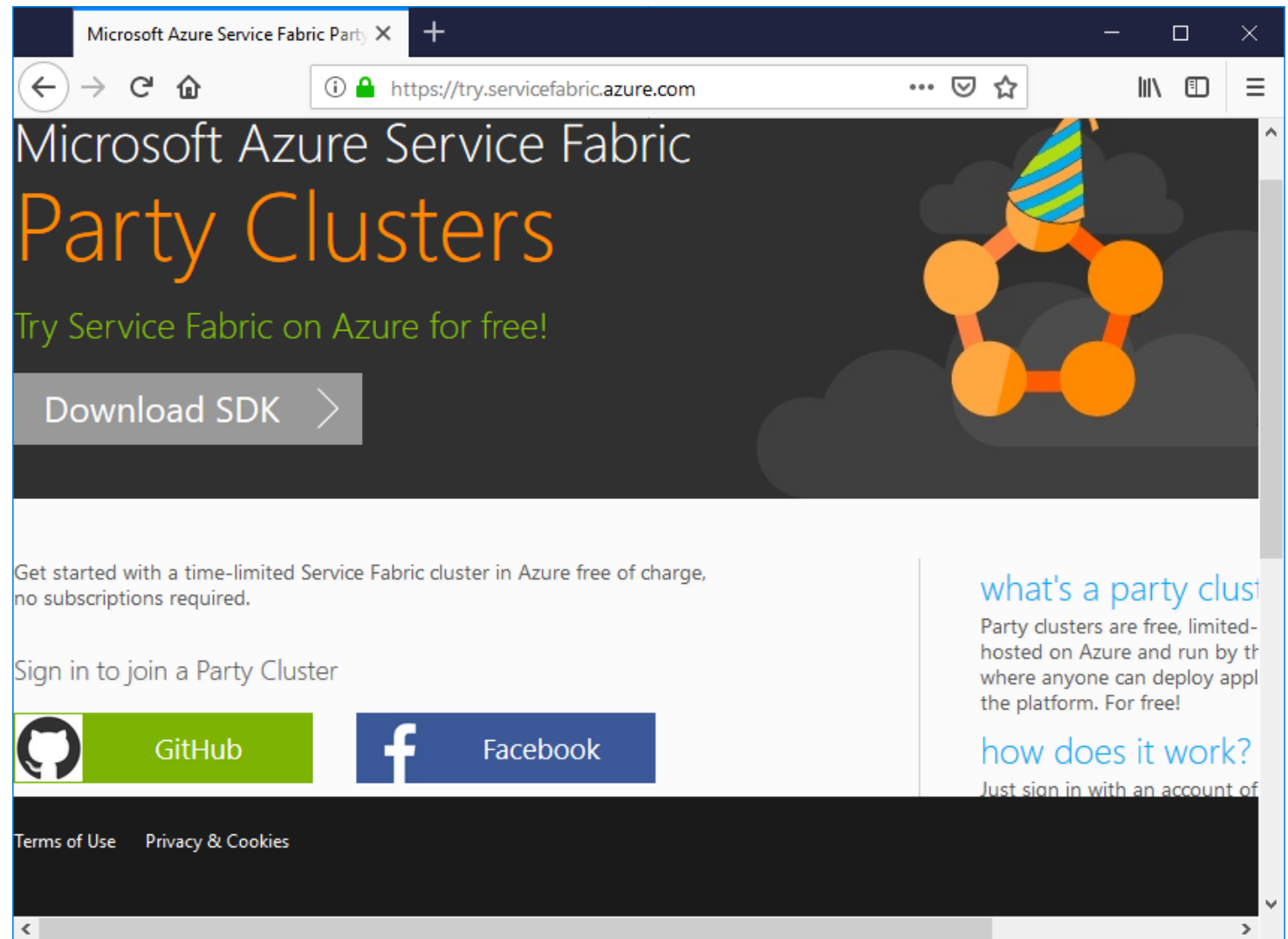
Party clusters are free, limited-time Service Fabric clusters hosted on Azure and run by the Service Fabric team where anyone can deploy applications and learn about the platform. The cluster uses a single self-signed certificate for node-to-node as well as client-to-node security.

Sign in and join a Windows cluster.

[*https://try.servicefabric.azure.com/*](https://try.servicefabric.azure.com/)

Deploy a .NET app to Service Fabric

Download the PFX certificate to your computer by clicking the **PFX** link. Click the **How to connect to a secure Party cluster?** link and copy the certificate password. The certificate, certificate password, and the **Connection endpoint** value are used in following steps.



Deploy a .NET app to Service Fabric

Download the PFX certificate to your computer by clicking the **PFX** link. Click the **How to connect to a secure Party cluster?** link and copy the certificate password. The certificate, certificate password, and the **Connection endpoint** value are used in following steps.

Note

There are a limited number of Party clusters available per hour. If you get an error when you try to sign up for a Party cluster, you can wait for a period and try again,

Your secure Service Fabric cluster is ready!

ReadMe: [How to connect to a secure Party cluster?](#)

Certificate required to connect:

PFX

Service Fabric Explorer

<https://win2430dp0wl3t.westus.cloudapp.azure.com:19080/Explorer/index.html>

Connection endpoint

win2430dp0wl3t.westus.cloudapp.azure.com:19000

Expires on:

November 16 at 19:43:16 UTC

Time remaining

0 hours, 37 minutes, and 9 seconds

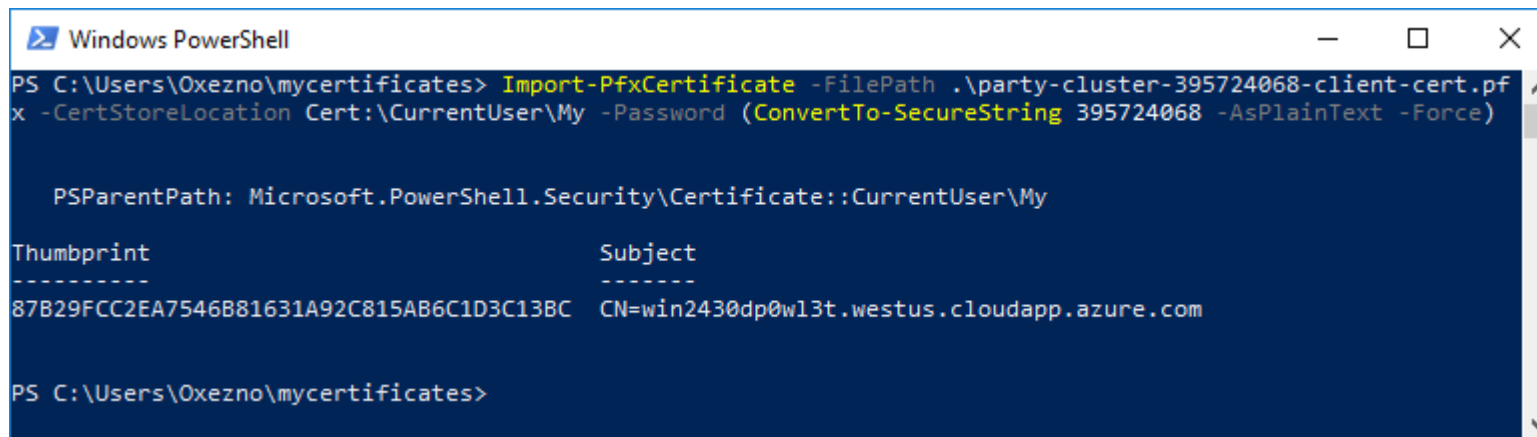
Available ports

80, 8081, 8080, 20000, 20001, 20002, 20003, 20004, 20005

Deploy a .NET app to Service Fabric

On your Windows machine, install the PFX in *CurrentUser\My* certificate store.

- *mkdir mycertificates*
- *Import-PfxCertificate -FilePath .\party-cluster-873689604-client-cert.pfx -CertStoreLocation Cert:\CurrentUser\My -Password (ConvertTo-SecureString 873689604 -AsPlainText -Force)*



```
Windows PowerShell
PS C:\Users\Oxezno\mycertificates> Import-PfxCertificate -FilePath .\party-cluster-395724068-client-cert.pfx -CertStoreLocation Cert:\CurrentUser\My -Password (ConvertTo-SecureString 395724068 -AsPlainText -Force)

PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My

Thumbprint                               Subject
-----
87B29FCC2EA7546B81631A92C815AB6C1D3C13BC  CN=win2430dp0w13t.westus.cloudapp.azure.com

PS C:\Users\Oxezno\mycertificates>
```

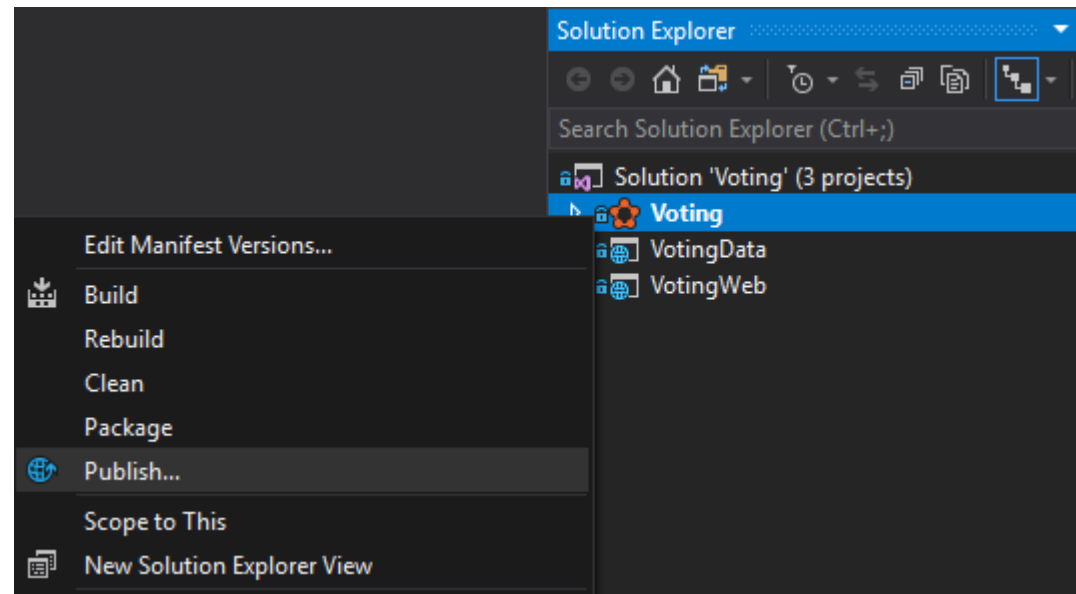
Remember the thumbprint for a following step.

Deploy a .NET app to Service Fabric

Deploy the application using Visual Studio

Now that the application is ready, you can deploy it to a cluster directly from Visual Studio.

1. Right-click Voting in the Solution Explorer and choose Publish. The Publish dialog appears.



Deploy a .NET app to Service Fabric

- Copy the Connection Endpoint from the Party cluster page into the Connection Endpoint field. For example, *zwin7fh14scd.westus.cloudapp.azure.com:19000*. Click Advanced Connection Parameters and ensure that the FindValue and ServerCertThumbprint values match the thumbprint of the certificate installed in a previous step.

- Click **Publish**.

Publish Service Fabric Application

Target profile:
PublishProfiles\Cloud.xml

Save Profile

Microsoft account
dubanand@outlook.com

Connection Endpoint:
win2430dp0wl3t.westus.cloudapp.azure.com:19000

Advanced Connection Parameters

Name	Value
X509Credential	true
ServerCertThumbprint	87B29FCC2EA7546B81631A92C815AB6C1D3C13BC
FindType	FindByThumbprint
FindValue	87B29FCC2EA7546B81631A92C815AB6C1D3C13BC
StoreLocation	CurrentUser
StoreName	My
Add a parameter	

Remove

How to configure secure connections

Application Parameters File:
ApplicationParameters\Cloud.xml

Edit...

☐ Upgrade the Application

Configure Upgrade Settings

Manifest Versions...

Publish Cancel

Deploy a .NET app to Service Fabric

4. Open a browser and type in the cluster address followed by ':8080' to get to the application in the cluster - for example,

- <http://zwin7fh14scd.westus.cloudapp.azure.com:8080>.

You should now see the application running in the cluster in Azure.

