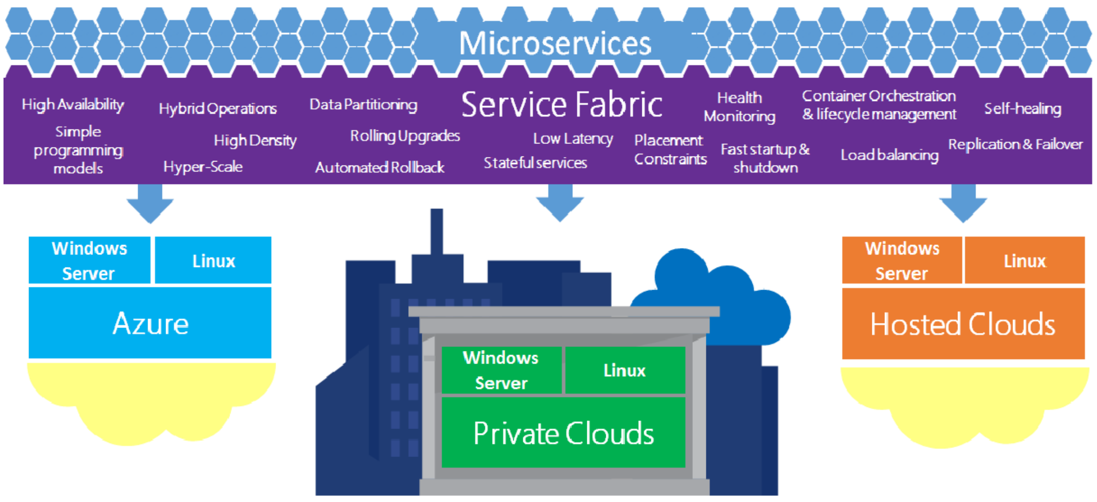
What is Azure Service Fabric?

Service Fabric is a distributed systems platform used to build scalable, reliable, and easily-managed applications for the cloud. Service Fabric addresses the significant challenges in developing and managing cloud applications.
Service Fabric provides comprehensive runtime and lifecycle management capabilities to applications composed of these microservices. It hosts microservices inside containers that are deployed and activated across the Service Fabric cluster.

Key concepts:

- Applications on service fabric are designed to be composed of microservices
- Stateless or stateful Service Fabric microservices are basic unit
- Application lifecycle management is managed by service fabric.



Azure Service Fabric addresses issues in PaaS V1 as below:

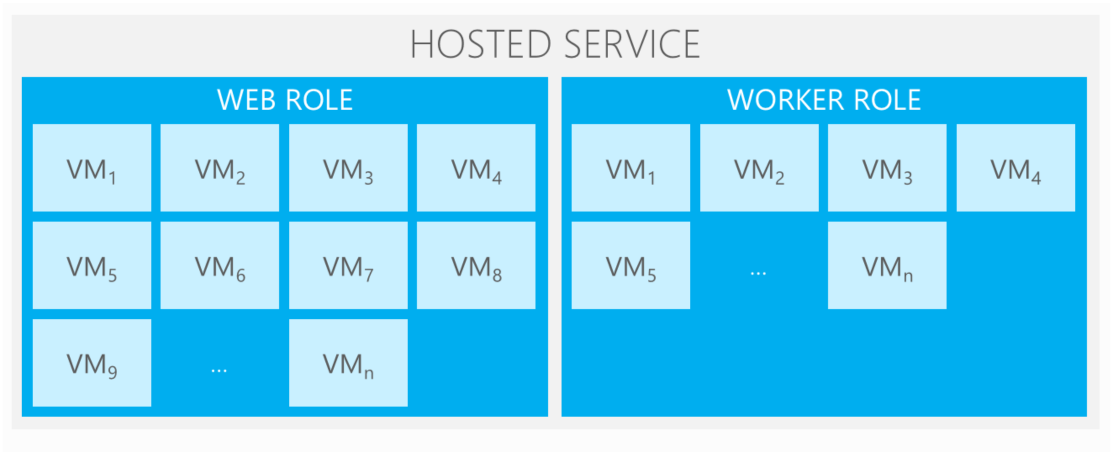
- Lack flexibility (Win Server): → service fabric will support windows and Linux in future.
- Low VM resource utilization: → Service Fabric will organize resource in smart unit like docker container.
- Hard to implement distributed system
  - Replication
  - Load balancing

  → Service fabric manage load balancing, replication framework, leadership election (HA), partitioning (high performance), distributed collections and key/value store, etc.

Reference:
https://azure.microsoft.com/en-us/documentation/articles/service-fabric-overview/
http://channel9.msdn.com/Events/Build/2015/3-618

Comparing PaaS V1 and PaaS V2

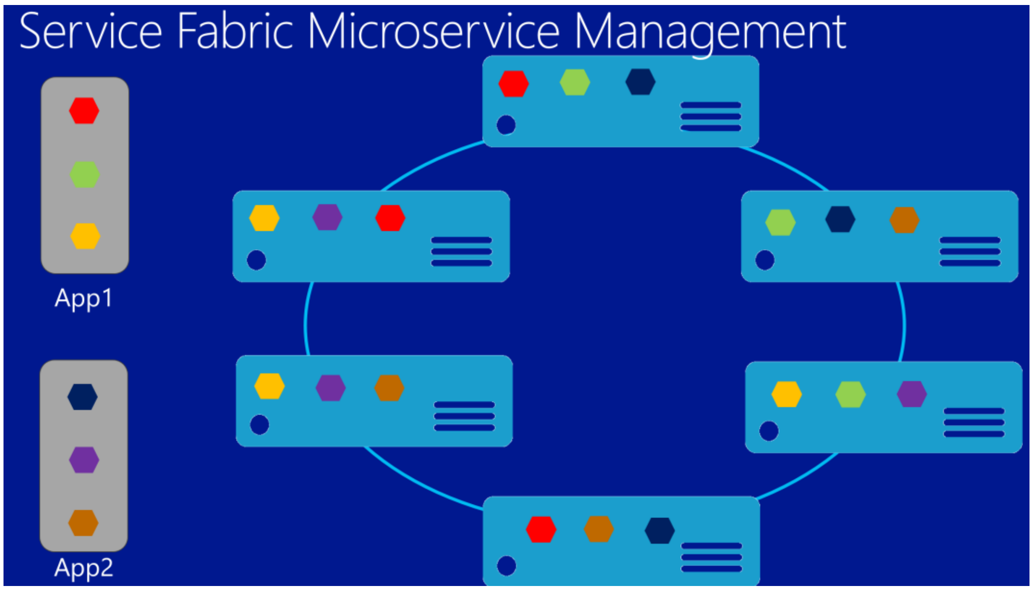
➢ PaaS V1:



Key Points:

- A service is designed into multi web/worker/etc. roles.
- Each role is initialized into multi instances, each instance stands for a managed virtual machine in azure datacenter.
- Each role instance exclusively use one virtual machine.
- All role instances are required to be stateless.
- Communications between instances are implemented by external queue/storage services, concurrency/transaction/consistency of the communication should be handled by developer.
- Auto healing is realized by replacing problem instance with new VM, no state consistency.
- …

➢ PaaS V1:



Key Points:

- A service is designed into multi microservices.
- Each micro service is initialized into multi instances, each instance stands for a managed compute unit like docker container in azure machines.
- Each azure machine can be used for multi microservice instances.
- Instances of microservices can be either stateless or stateful.
- Each instance is provisioned with replicas and auto syncing channels.
- For stateless microservice instances, communications between them can also be implemented by external queue/storage services, concurrency/transaction/consistency of the communication should be handled by developer.
- For stateful microservices instances, their state info is managed/replicated/partitioned by service fabric
- Auto healing is realized by replacing problem instance with instance replica, state consistency is guaranteed.
- …

Choosing frameworks to design

Service Fabric offers two high-level frameworks for building services: the Reliable Actors APIs and the Reliable Services APIs. While both are built on the same Service Fabric core, they make different tradeoffs between simplicity and flexibility in terms of concurrency, partitioning, and communication. It is useful to understand both models so that you can choose the appropriate framework for a particular service within your application.

RELIABLE ACTORS APIS	RELIABLE SERVICES APIS
Your problem space involves many small independent units of state and logic	You need to maintain logic across multiple components
You want to work with single-threaded objects while still being able to scale and maintain consistency	You want to use reliable collections (like .NET Dictionary and Queue) to store and manage your state
You want the framework to manage the concurrency and granularity of state	You want to control the granularity and concurrency of your state
You want the platform to manage communication for you	You want to manage the communication and control the partitioning scheme for your service
Can be used for: IoT Distributed computing. Social Networking of internet.	Can be used for: Website, web service, socket server and more. Can act as "stateful + stateless" worker/web role.

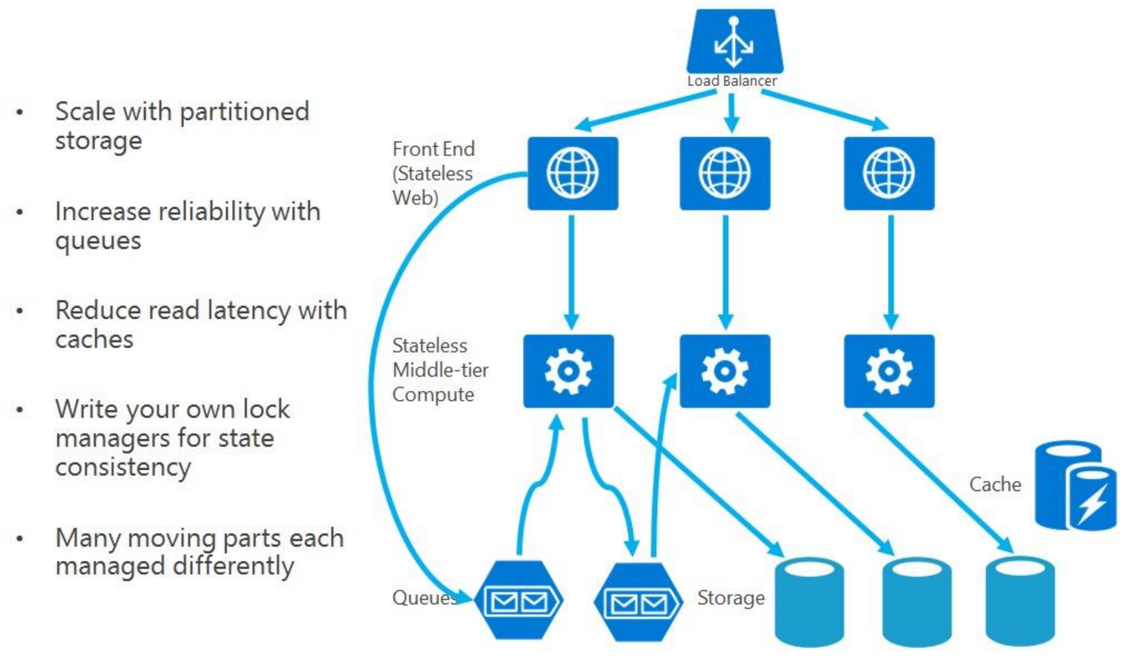
Currently these 2 programming models only support C# language (windows), but it is promised to support other popular languages and operation systems.

Reference about reliable service API and reliable actor API:
https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-actors-introduction/
https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-services-introduction/

Azure Service Fabric Scenarios and IoT reference

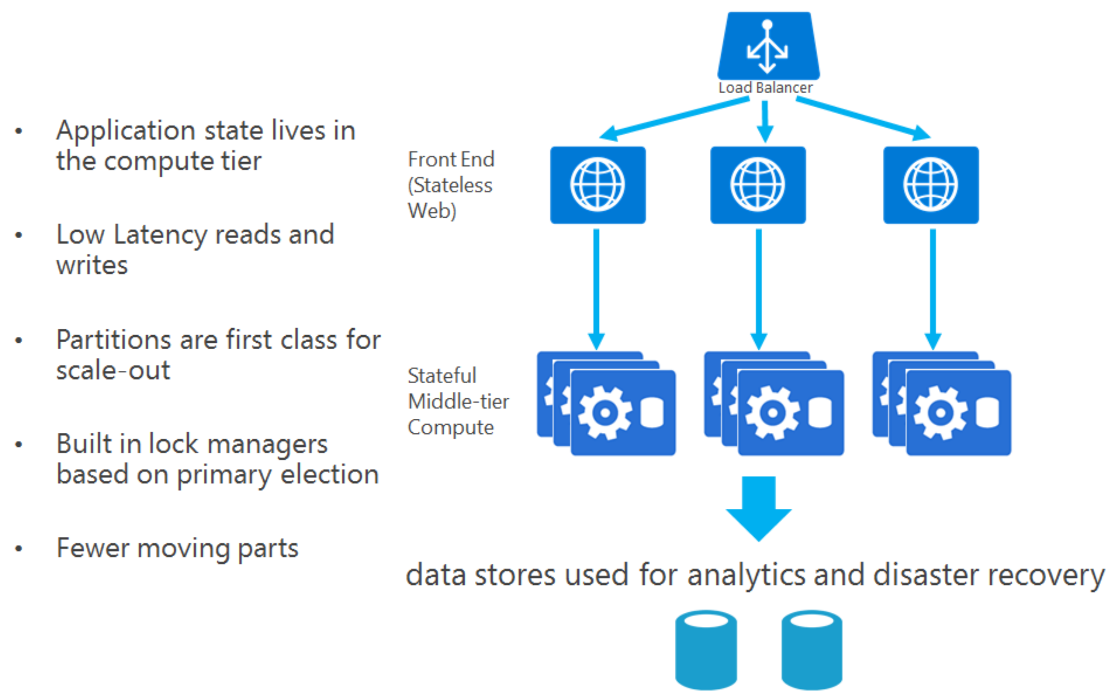
Basic implementation of stateless and stateful services
Reference:
https://azure.microsoft.com/en-us/documentation/articles/service-fabric-application-scenarios/
https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-actors-patterns-introduction/

- Stateless services



- Scale with partitioned storage
- Increase reliability with queues
- Reduce read latency with caches
- Write your own lock managers for state consistency
- Many moving parts each managed differently

- Stateful services

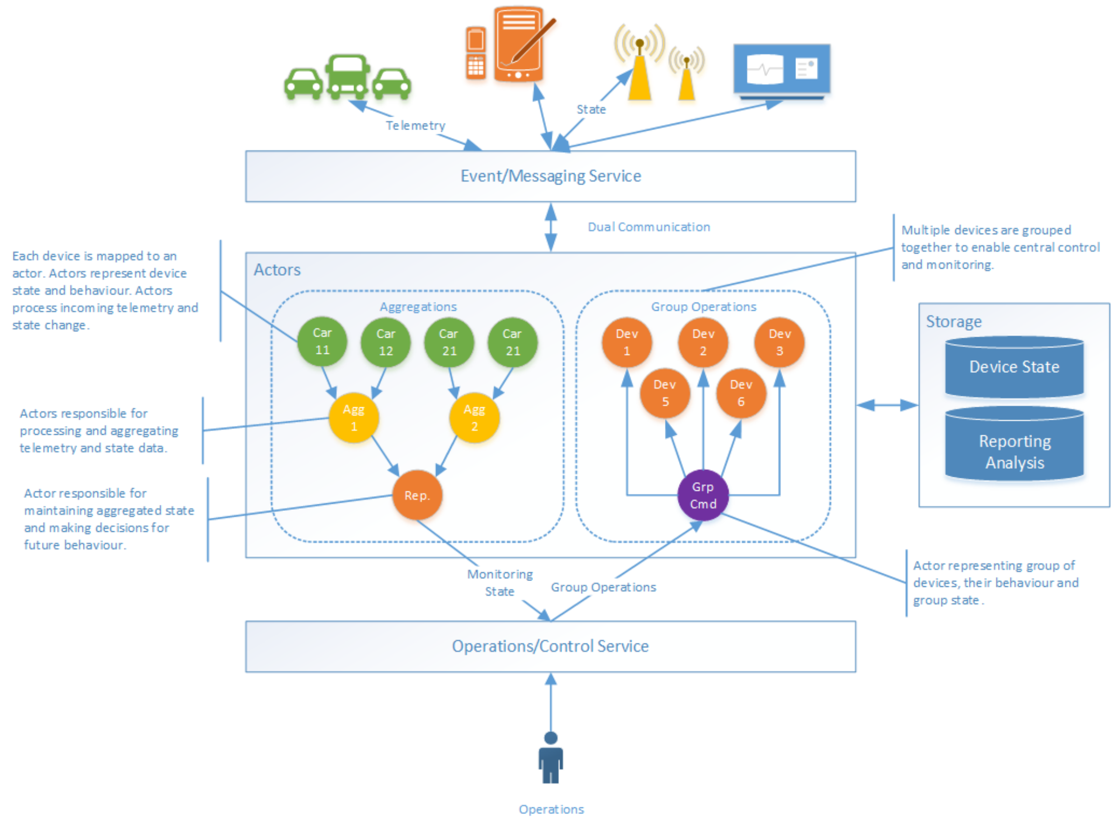


- Application state lives in the compute tier
- Low Latency reads and writes
- Partitions are first class for scale-out
- Built in lock managers based on primary election
- Fewer moving parts

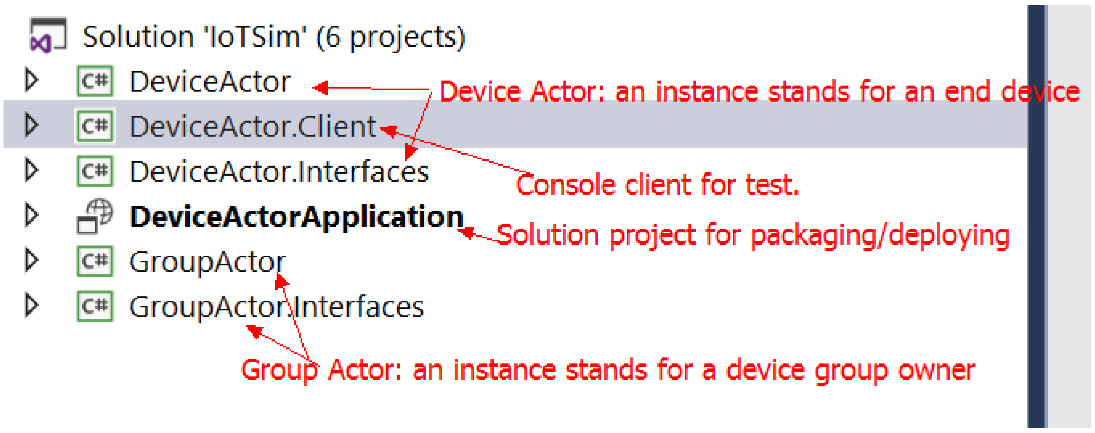
Load Balancer
Front End (Stateless Web)
Stateful Middle-tier Compute
data stores used for analytics and disaster recovery

➢ IoT Reference

Here is a common IoT scenario, end devices report their status to event service, then in cloud, dumped microservice instances are generated/updated to reflect the real complex end surrounding, including device status and their connection.


Telemetry
State
Event/Messaging Service
Dual Communication
Each device is mapped to an actor. Actors represent device state and behaviour. Actors process incoming telemetry and state change.
Multiple devices are grouped together to enable central control and monitoring.
Actors
Aggregations
Car 11
Car 12
Car 21
Car 21
Agg 1
Agg 2
Rep.
Group Operations
Dev 1
Dev 2
Dev 3
Dev 5
Dev 6
Grp Cmd
Storage
Device State
Reporting Analysis
Actors responsible for processing and aggregating telemetry and state data.
Actor responsible for maintaining aggregated state and making decisions for future behaviour.
Actor representing group of devices, their behaviour and group state.
Monitoring State
Group Operations
Operations/Control Service
Operations

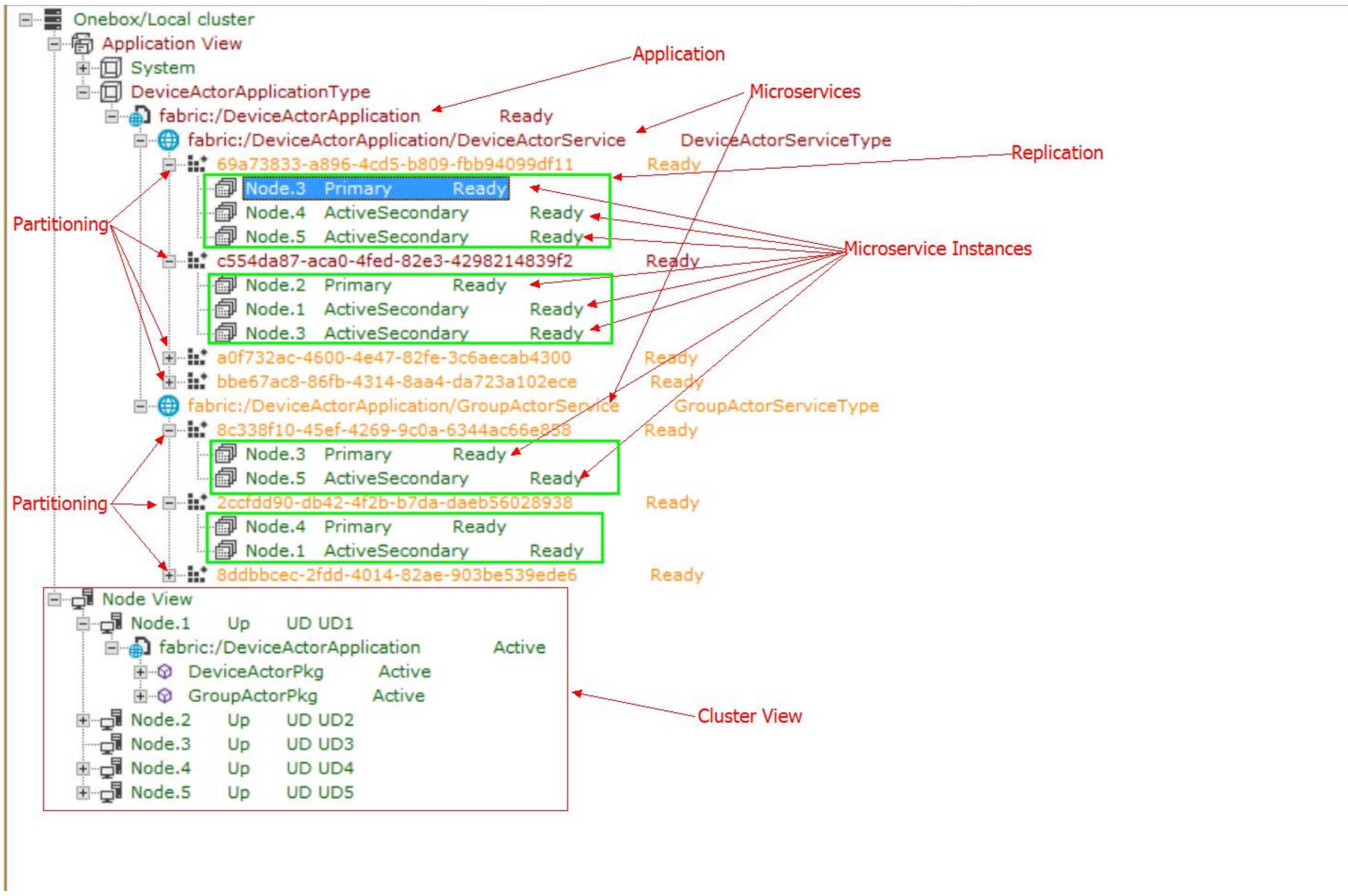
- Project Structure(Demo):



```
Solution 'IoTSim' (6 projects)
  DeviceActor
  DeviceActor.Client
  DeviceActor.Interfaces
  DeviceActorApplication
  GroupActor
  GroupActor.Interfaces
```

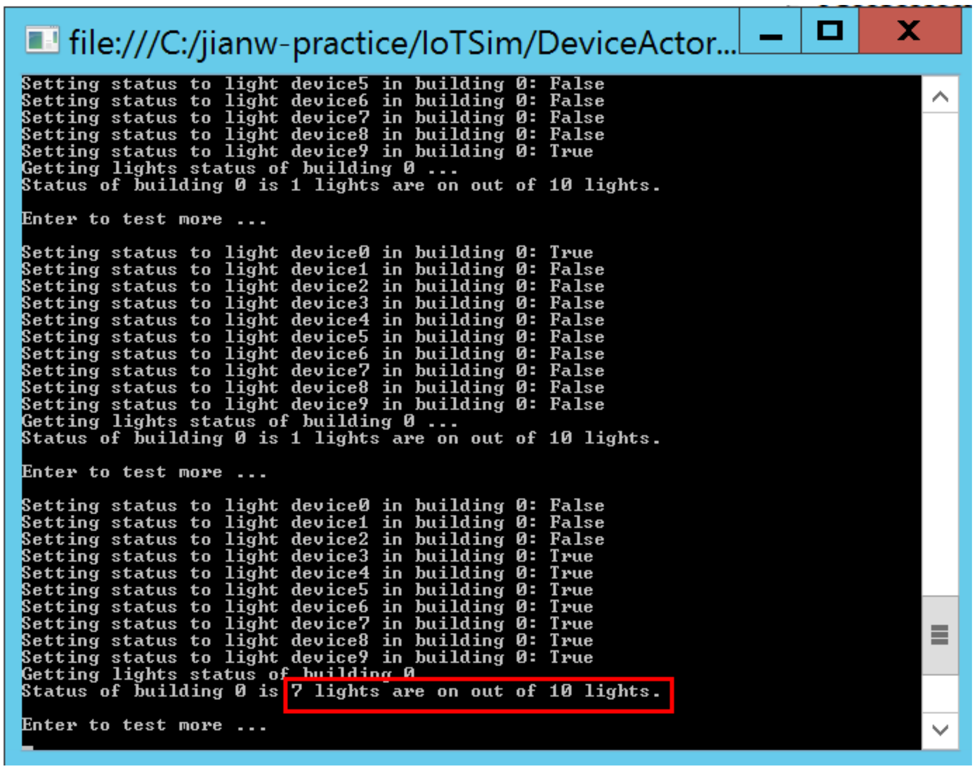
Device Actor: an instance stands for an end device
Console client for test.
Solution project for packaging/deploying
Group Actor: an instance stands for a device group owner

- Deployment Structure:



```
Onebox/Local cluster
  Application View
    System
    DeviceActorApplicationType
      fabric:/DeviceActorApplication                      Ready
        fabric:/DeviceActorApplication/DeviceActorService      DeviceActorServiceType
          69a73833-a896-4cd5-b809-fbb94099df11            Ready
            Node.3   Primary           Ready
            Node.4   ActiveSecondary         Ready
            Node.5   ActiveSecondary         Ready
          c554da87-aca0-4fed-82e3-4298214839f2            Ready
            Node.2   Primary           Ready
            Node.1   ActiveSecondary         Ready
            Node.3   ActiveSecondary         Ready
          a0f732ac-4600-4e47-82fe-3c6aecab4300            Ready
          bbe67ac8-86fb-4314-8aa4-da723a102ece            Ready
        fabric:/DeviceActorApplication/GroupActorService      GroupActorServiceType
          8c338f10-45ef-4269-9c0a-6344ac66e858            Ready
            Node.3   Primary           Ready
            Node.5   ActiveSecondary         Ready
          2ccfdd90-db42-4f2b-b7da-daeb56028938            Ready
            Node.4   Primary           Ready
            Node.1   ActiveSecondary         Ready
          8ddbbcec-2fdd-4014-82ae-903be539ede6            Ready
  Node View
    Node.1     Up     UD UD1
      fabric:/DeviceActorApplication           Active
        DeviceActorPkg          Active
        GroupActorPkg           Active
    Node.2     Up     UD UD2
    Node.3     Up     UD UD3
    Node.4     Up     UD UD4
    Node.5     Up     UD UD5
```

- Demo:



```
Setting status to light device5 in building 0: False
Setting status to light device6 in building 0: False
Setting status to light device7 in building 0: False
Setting status to light device8 in building 0: False
Setting status to light device9 in building 0: True
Getting lights status of building 0 ...
Status of building 0 is 1 lights are on out of 10 lights.

Enter to test more ...

Setting status to light device0 in building 0: True
Setting status to light device1 in building 0: False
Setting status to light device2 in building 0: False
Setting status to light device3 in building 0: False
Setting status to light device4 in building 0: False
Setting status to light device5 in building 0: False
Setting status to light device6 in building 0: False
Setting status to light device7 in building 0: False
Setting status to light device8 in building 0: False
Setting status to light device9 in building 0: False
Getting lights status of building 0 ...
Status of building 0 is 1 lights are on out of 10 lights.

Enter to test more ...

Setting status to light device0 in building 0: False
Setting status to light device1 in building 0: False
Setting status to light device2 in building 0: False
Setting status to light device3 in building 0: True
Setting status to light device4 in building 0: True
Setting status to light device5 in building 0: True
Setting status to light device6 in building 0: True
Setting status to light device7 in building 0: True
Setting status to light device8 in building 0: True
Setting status to light device9 in building 0: True
Getting lights status of building 0
Status of building 0 is 7 lights are on out of 10 lights.

Enter to test more ...
```

Windows Fabric VS. Azure Service Fabric

The Service Fabric team is pleased to announce the availability of *Service Fabric v4.0 CU1 release*. Microsoft Azure Service Fabric is the official public name of the technology known internally as ***Windows Fabric***.

As we transition to a public platform, we will continue to develop and release the Windows Fabric runtime and SDK to internal partners who are already consuming it. In addition, we are providing the new Service Fabric SDK preview with the Reliable Services and Reliable Actor programming models. For more information about the differences in these packages and their roadmap, please see our Frequently Asked Questions.

Summary

1. Azure Service Fabric is new branding of pervious Windows Fabric, it is now set up in Azure public datacenters and its goal to provide the fabric service to public customers for deploying/hosting hyper-scale, complex, high availability, high performance, mission critical compute application.
2. Applications on service fabric are composed of microservices (stateful or stateless). In future, a microservice can be any c++/c#/java/php/etc. code execution or other executable like .exe.
3. Azure Service Fabric leverage smart container rather than current virtual machine in PaaS V1 to host/manage microservice instance.
4. Azure service fabric is responsible for load balancing, replication, consistency, data syncing and more for microservice instances, developers only need to focus on application design.
5. Azure service fabric currently provide 2 programming models(actor API and service API).
6. With actor API, real scenario with thousands of objects like IoT can be dumped into compute model and is ready for any quick processing (like alert, reporting.).
7. Azure Service Fabric provide thorough interfaces for developers to specify how an application is hosted in target fabric (including replication, load balancing, partitioning, consistency, garbage collection, concurrency, and all requirements in distributed computing).
8. Azure service fabric is with same core as previous windows fabric, so applications running on azure service fabric can work on local windows fabric seamlessly.

More

Dev Environment: https://azure.microsoft.com/en-us/documentation/articles/service-fabric-get-started/
Document: https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-actors-introduction/
More: https://azure.microsoft.com/en-us/documentation/articles/service-fabric-overview/