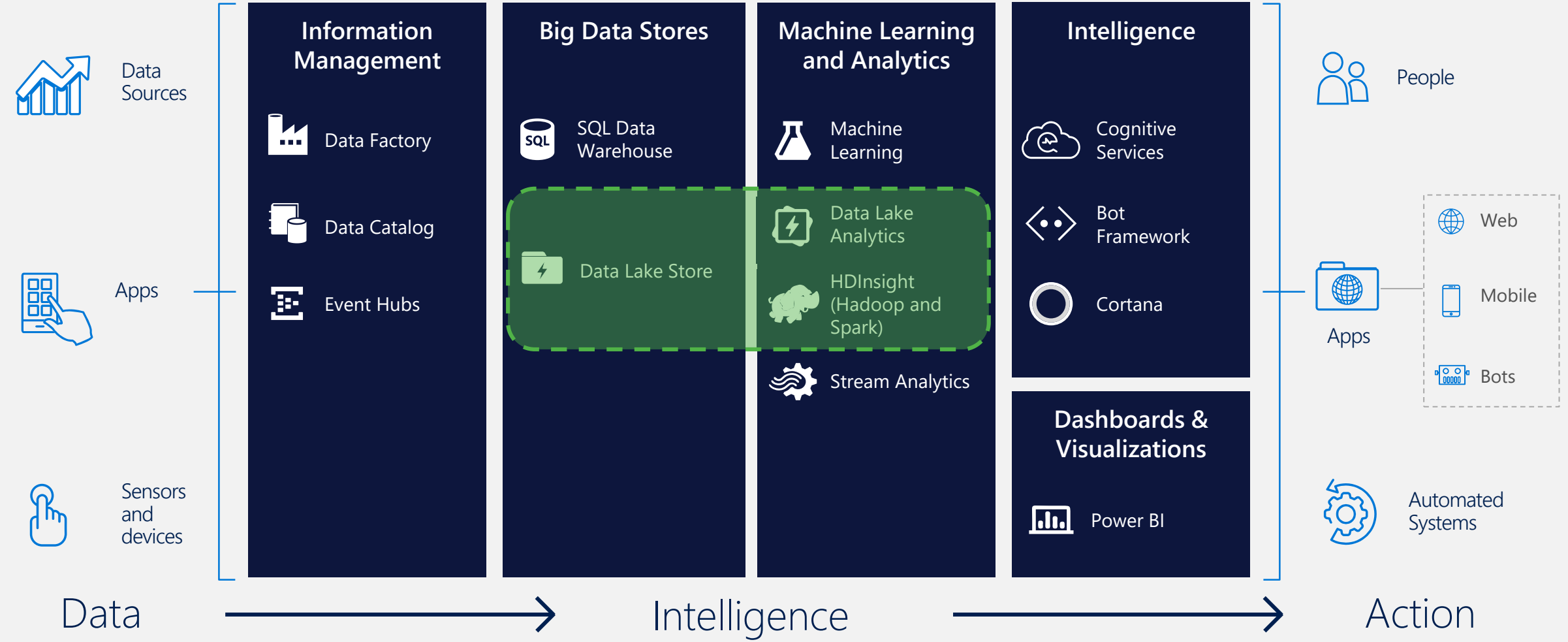
Azure Data Lake Analytics

Manjunath Suryanarayana

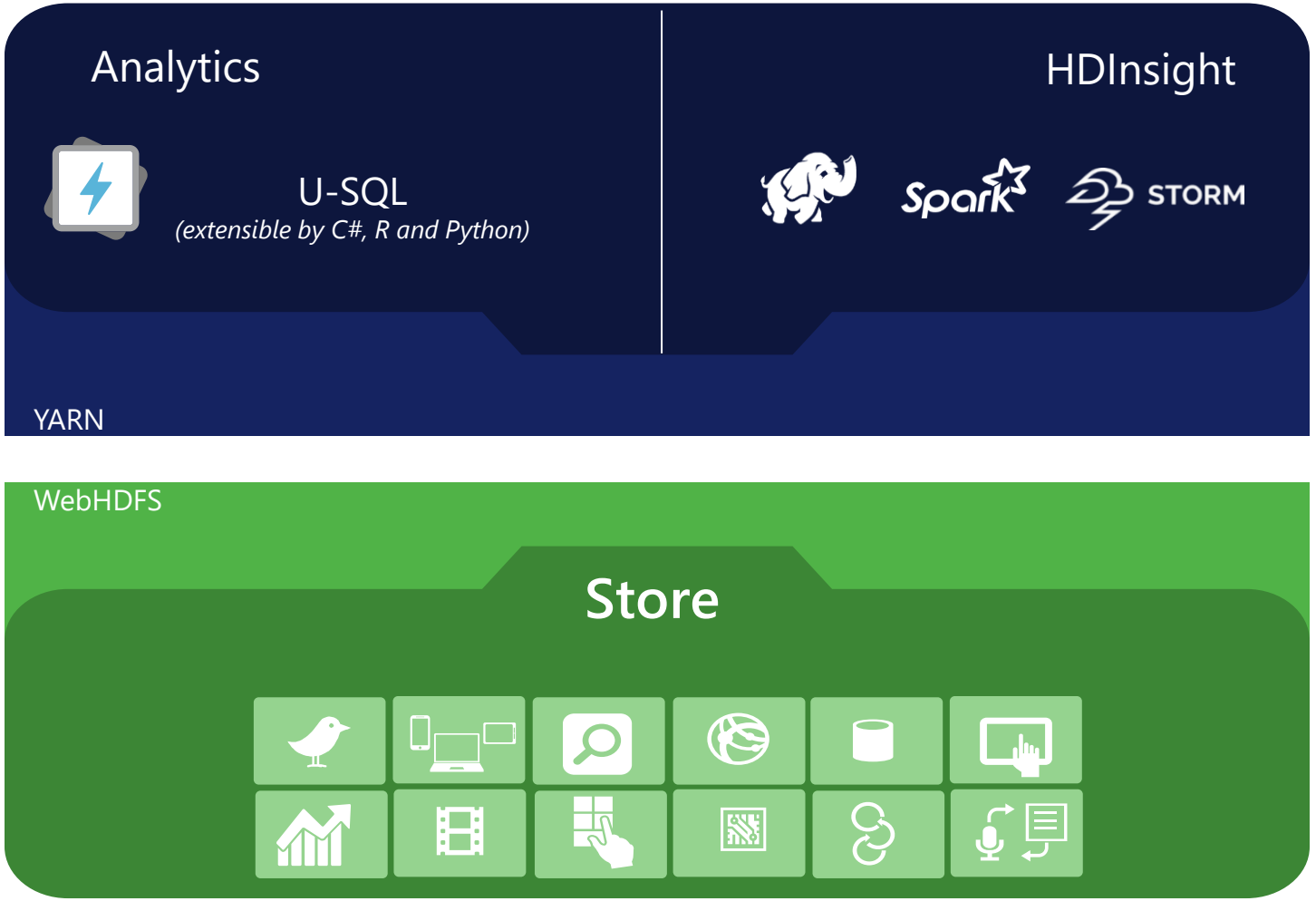
Microsoft

Azure Data Lake

as part of Cortana Intelligence Suite



Azure Data Lake

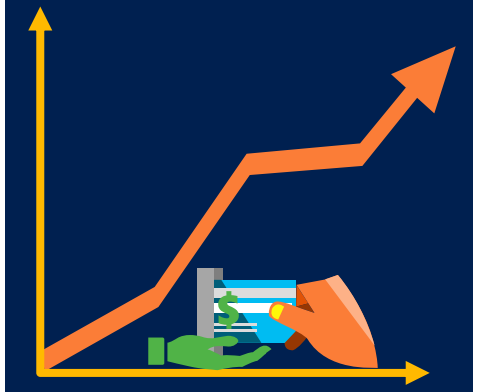


Azure Data Lake Analytics

Microsoft

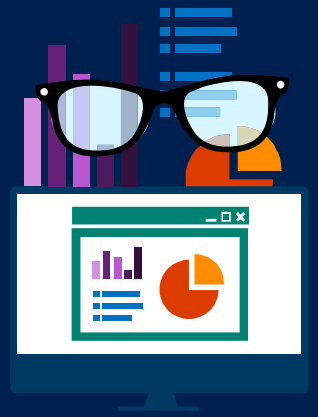
Azure Data Lake Analytics

Start in seconds
Scale instantly
Pay per job



Develop massively parallel programs with simplicity

Debug and optimize your Big Data programs with ease



Virtualize your analytics

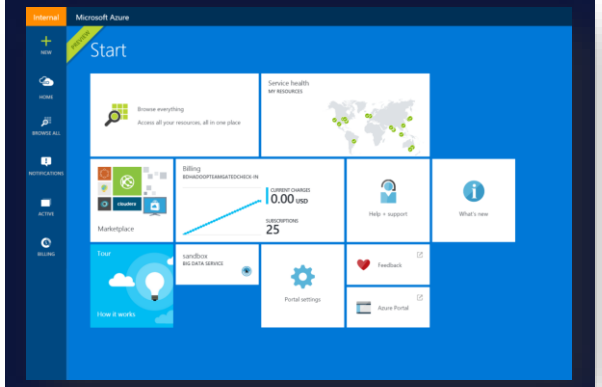
Enterprise-grade security, auditing and support

Microsoft

Get started

1

Log in to Azure



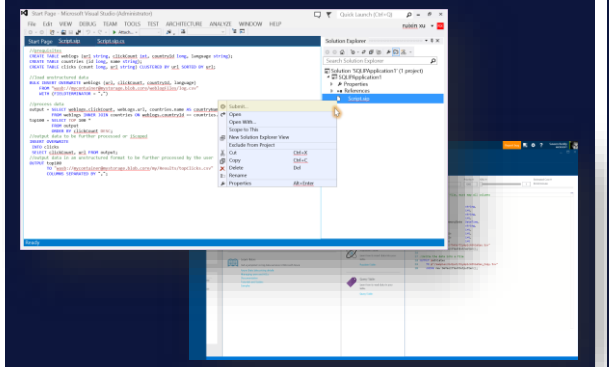
2

Create an ADLA account

30 seconds

3

Write and submit an ADLA job with U-SQL



4

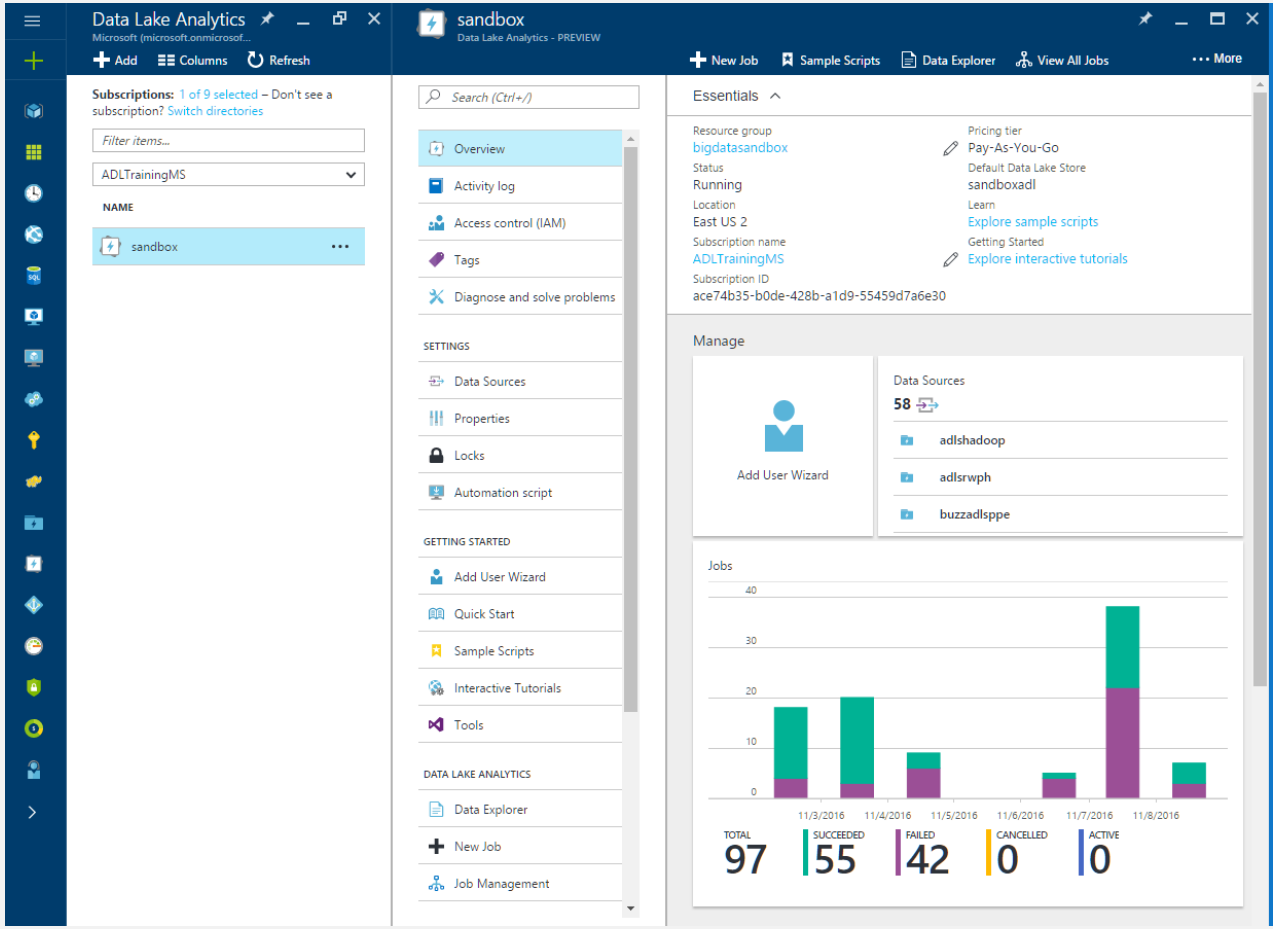
The job reads and writes data from storage

ADLS
Azure Blobs
Azure DB
…

Microsoft

What can you do in the Azure Portal?

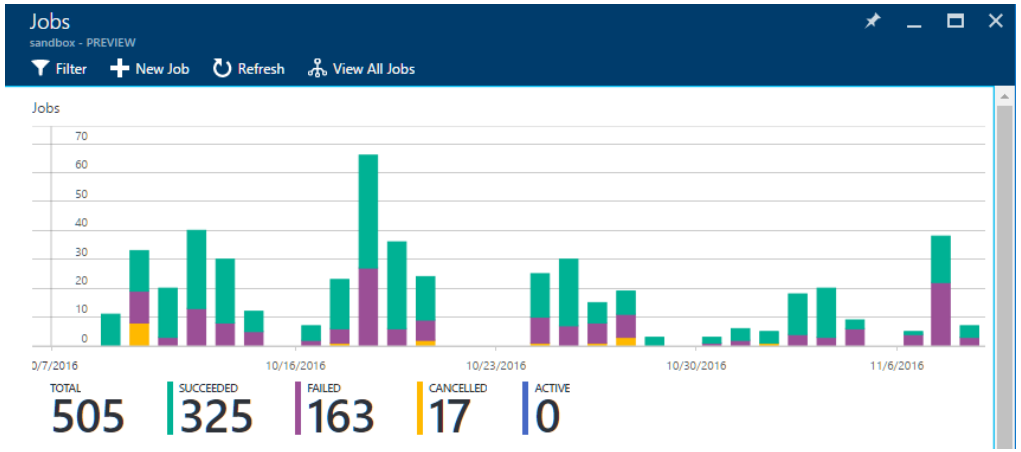
- Create a new Data Lake Analytics account
- Author U-SQL scripts
- Submit U-SQL jobs
- Cancel running jobs
- Provision users who can submit jobs
- Visualize usage stats (compute hours)
- Visualize job management chart

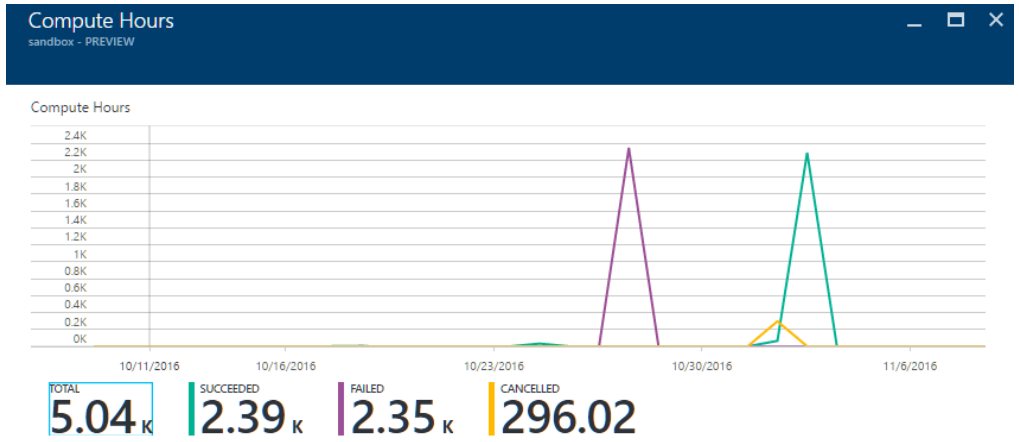


Visualizing ADL Analytics monitoring charts

1. **Job management:**
   The total number of jobs submitted as well as the number that succeeded, failed or were cancelled

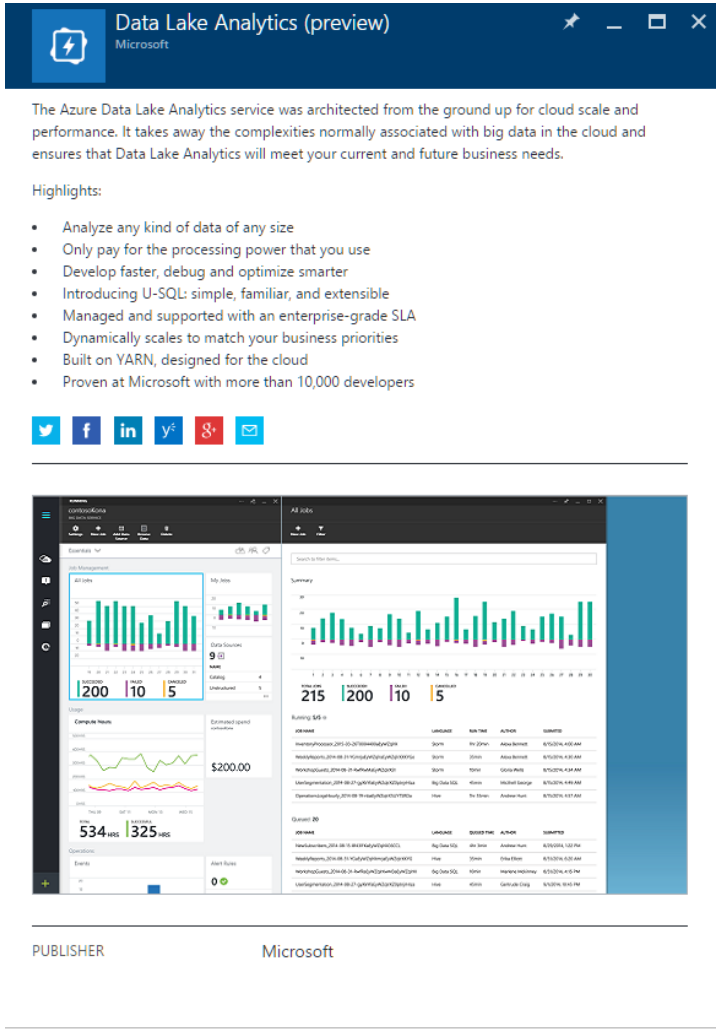
2. **Usage:**
   The number of compute hours consumed by the jobs

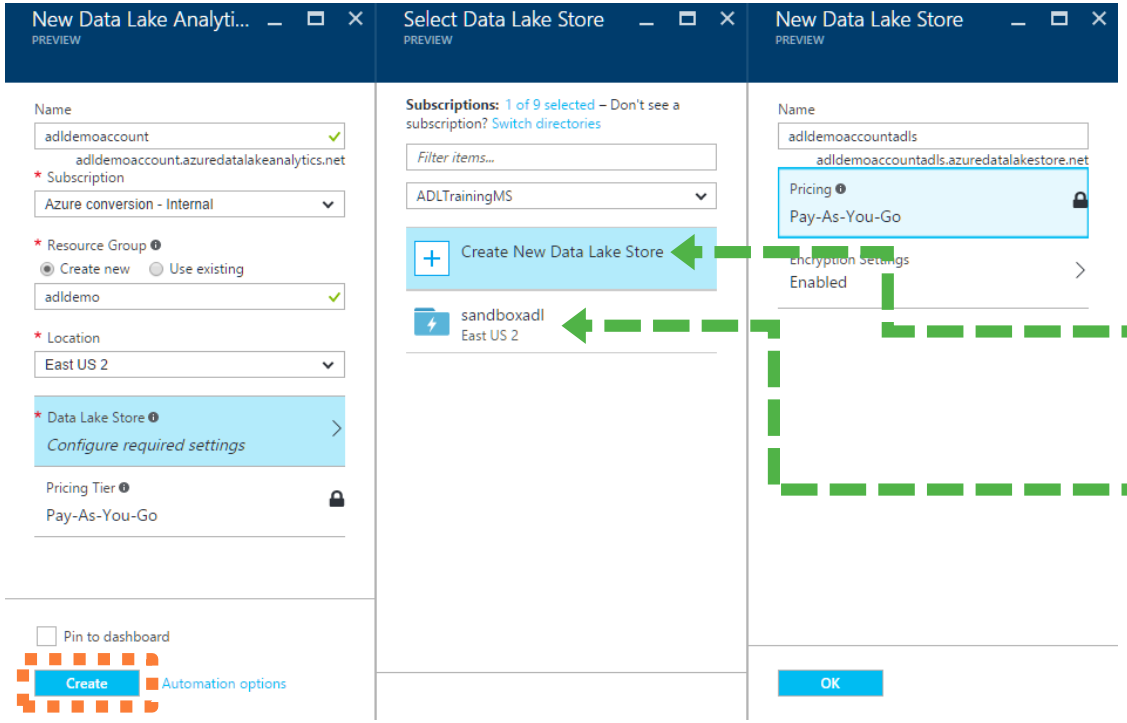




Microsoft

Creating an ADL Analytics account



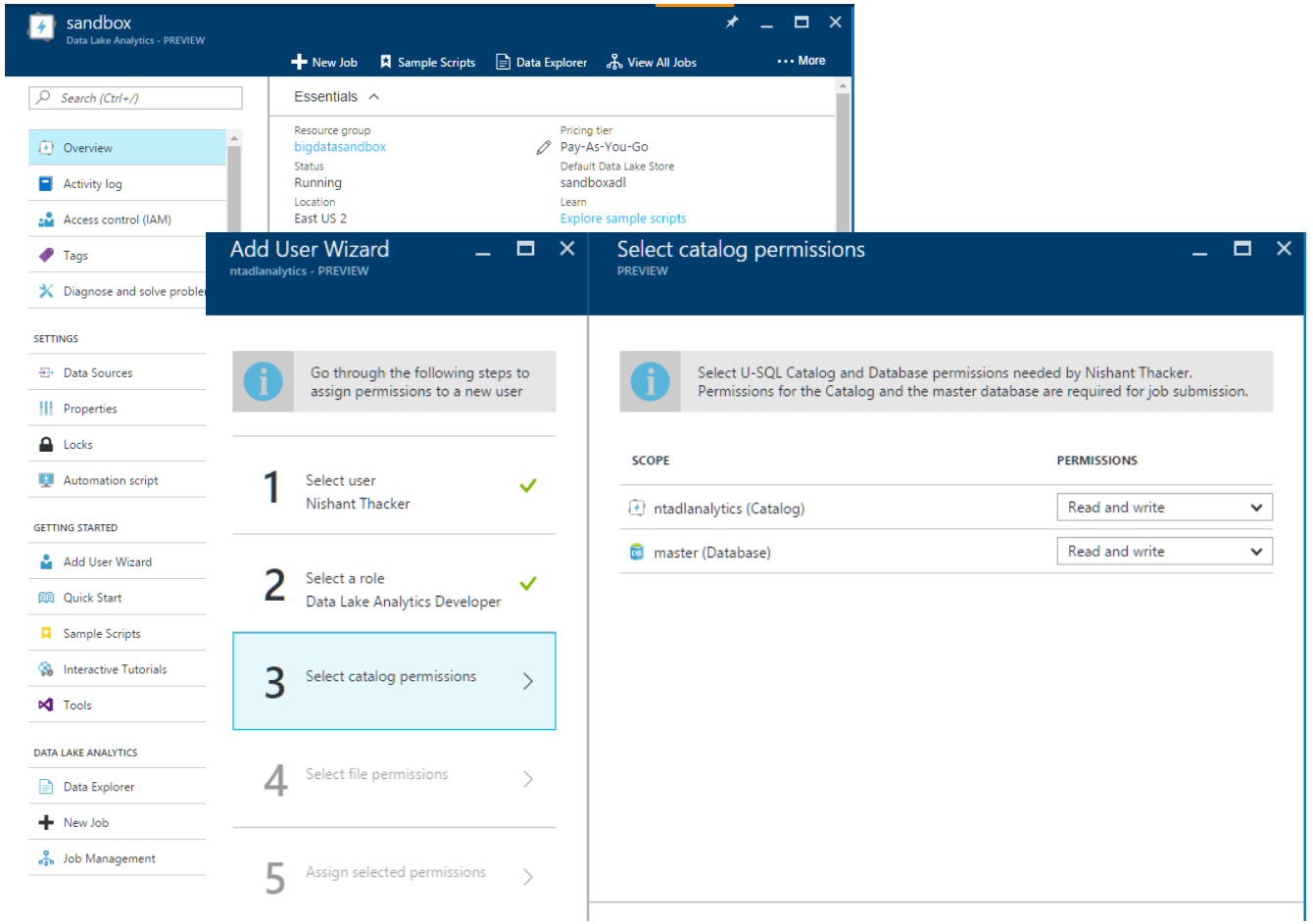


- Create a new ADL store default account, or
- Associate with an existing ADL store account (with the right permissions)

Deploying Azure Data Lake Analytics

Provisioning users for Data Lake Analytics

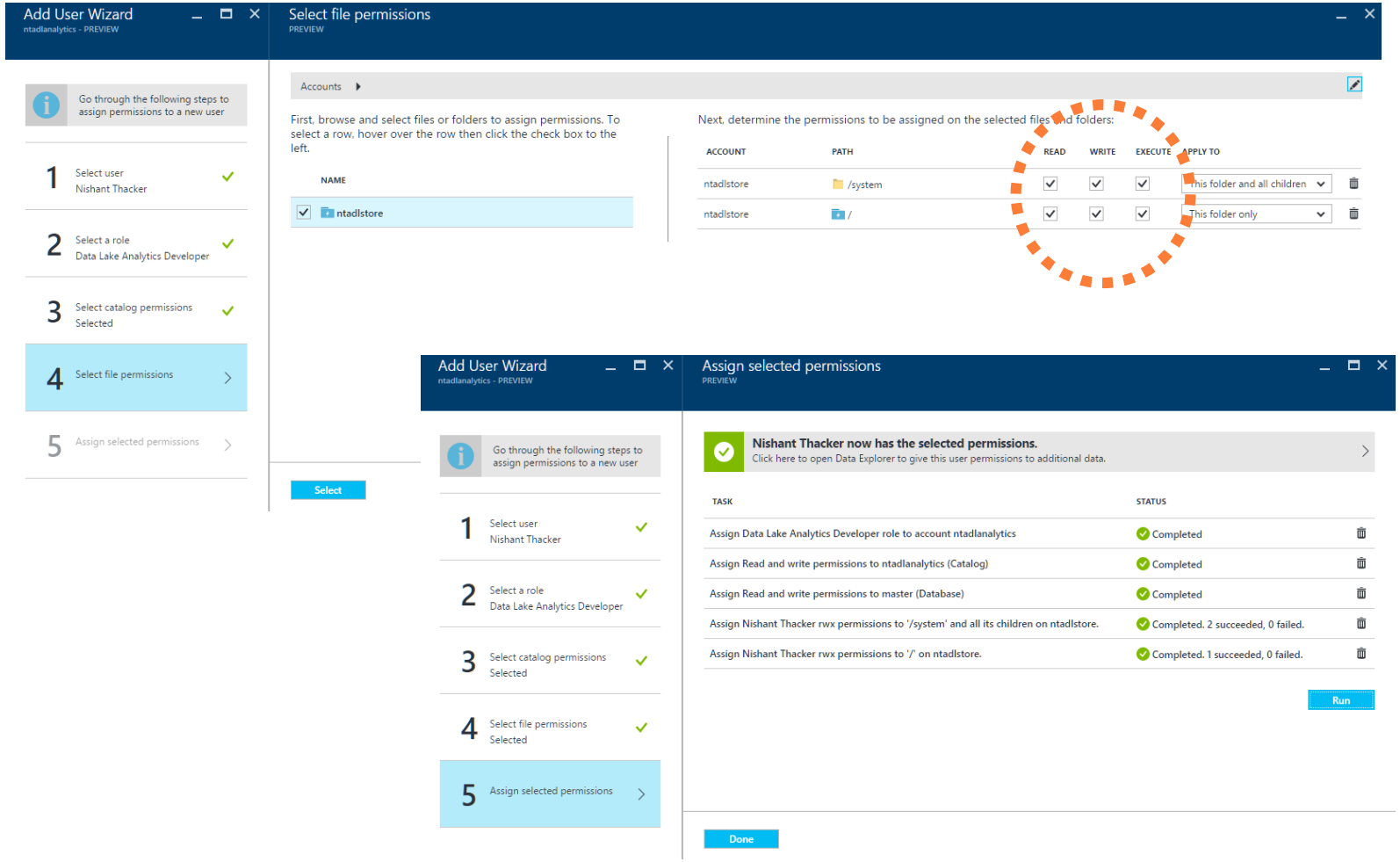
Users have to be explicitly provisioned for both the Big Data Lake account as well as for the associated ADL Store account



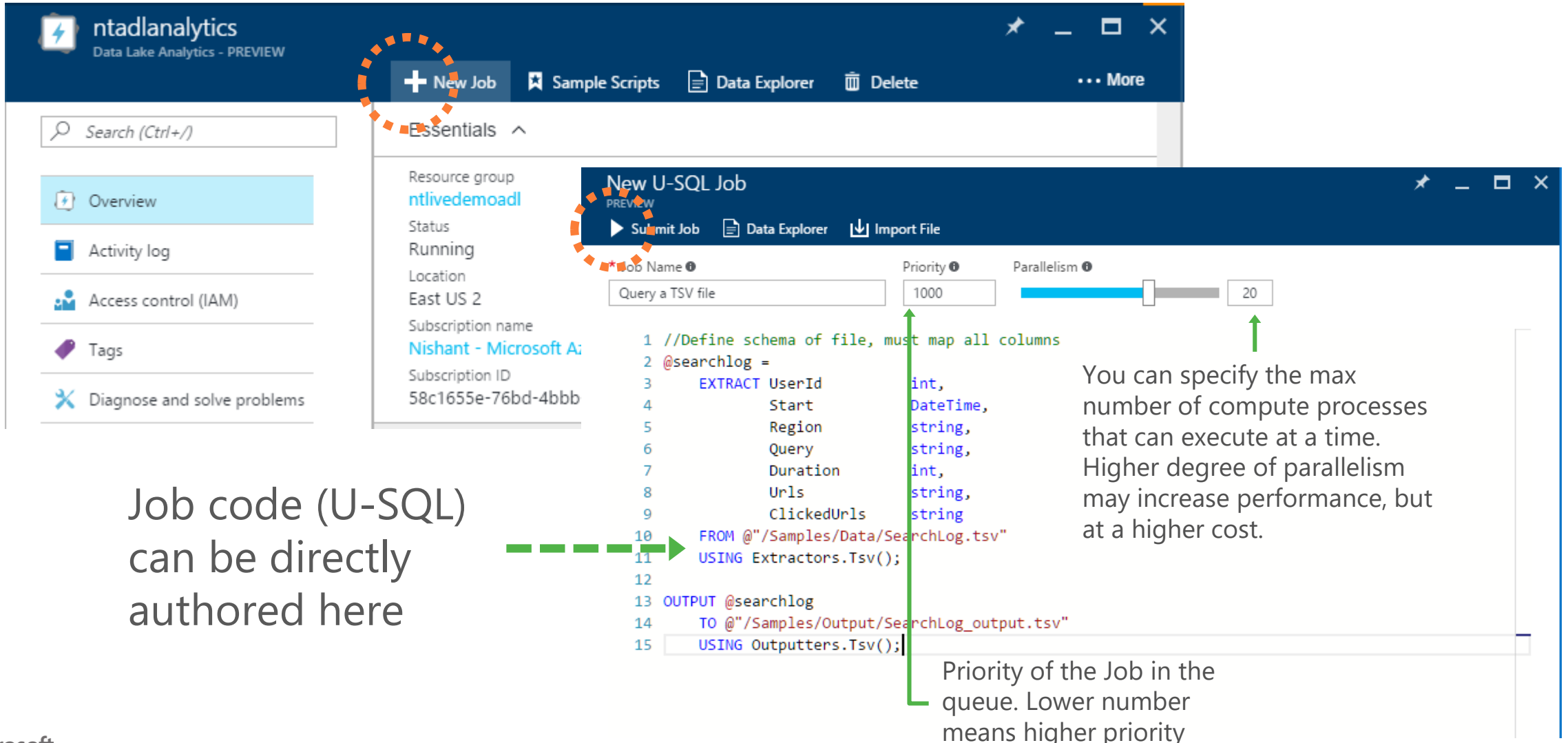
Provisioning users for Data Lake Store

Users must be explicitly granted *read, write or execute* permission on the Data Lake account.

Note, permissions are automatically set for new co-created Data Lake accounts.



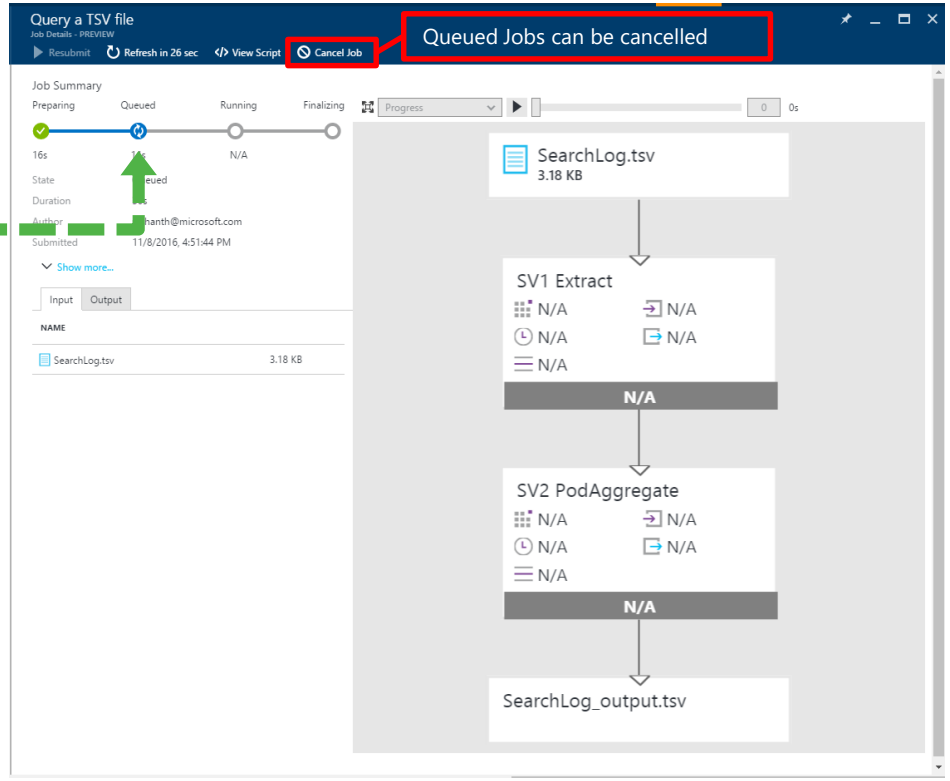
Creating, submitting a new job



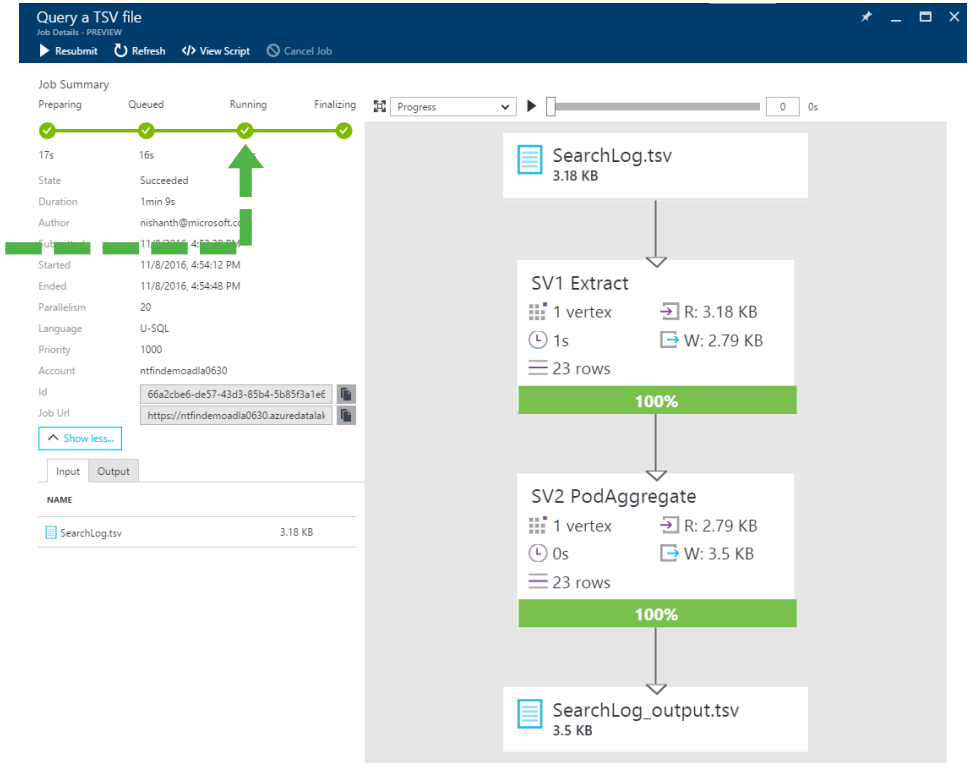
Job code (U-SQL) can be directly authored here

Job status: queued & running

Initially the job is in “Queued” state, waiting to run

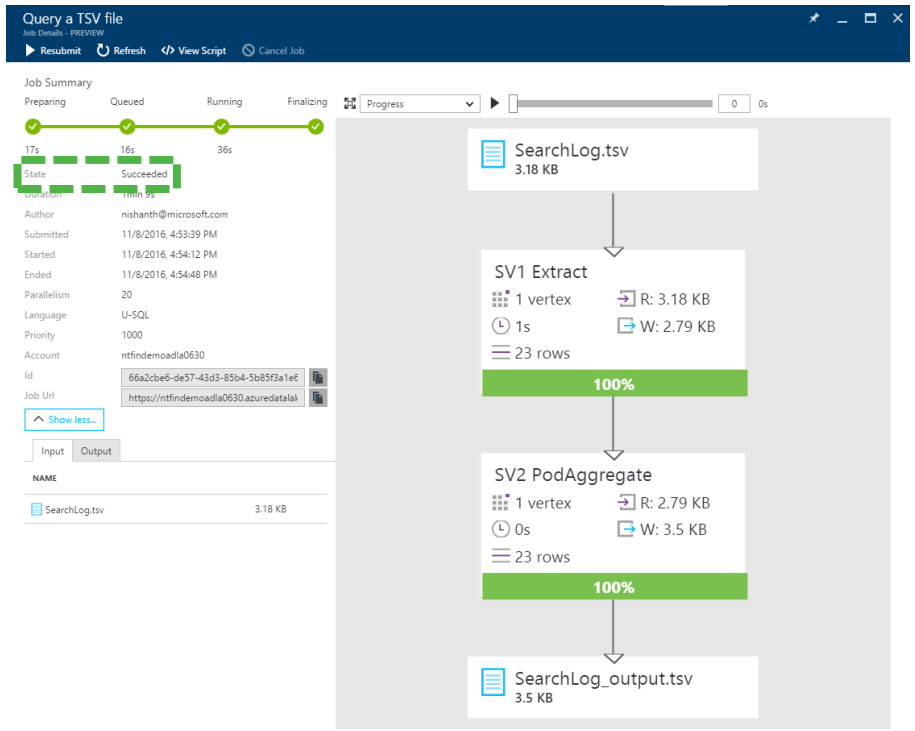


After the job is scheduled to run, the job status is “Running”

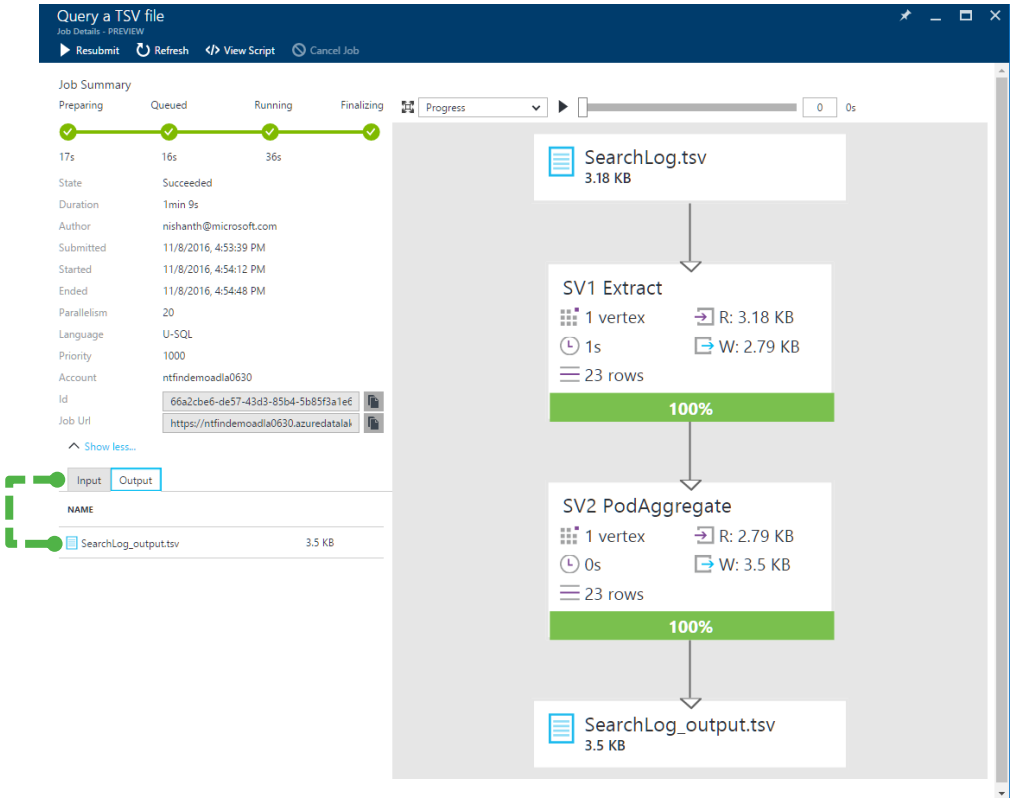


Job results

If the job completes the *final* status is succeeded



Input and output (results) are shown



Microsoft

File preview

- Input and output files can be previewed directly in the portal without having to download them
- The preview shows the first few rows
- Column numbers are automatically assigned
- Understands CSV and TSV formats

ADLA Billing

Microsoft

ADLA billing

- Accounts are **FREE!**
- Pay for the compute resources you want for your **queries**
- Pay for **storage separately**

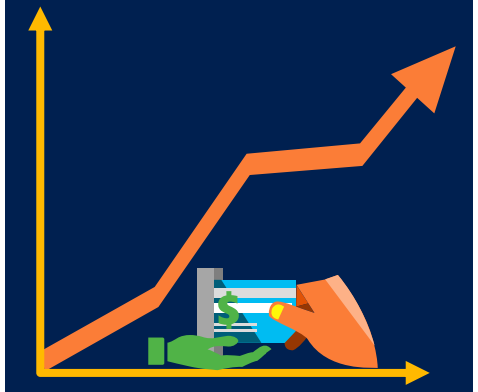
(query_hours * parallelism) * price/hour

USAGE	PREVIEW PRICE (UNTIL DECEMBER 31ST, 2016)	GA PRICE (STARTING JANUARY 1ST, 2017)*
ADLAU	$1 / hour	$2 / hour
Completed Job	$0.025 / Job	Free

*special monthly commitment discounted pricing available

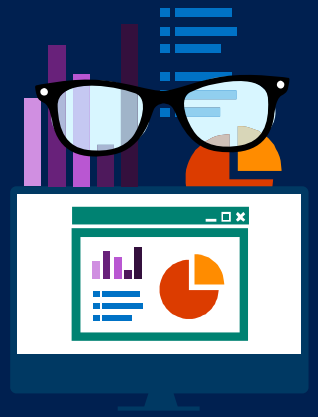
Azure Data Lake Analytics

Start in seconds
Scale instantly
Pay per job

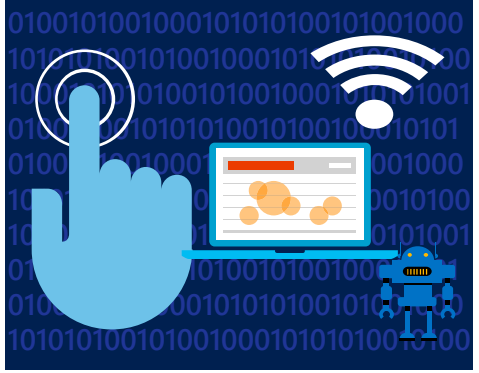


Develop massively parallel programs with simplicity

Debug and optimize your Big Data programs with ease



Virtualize your analytics

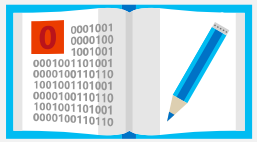


Enterprise-grade security, auditing and support

Microsoft

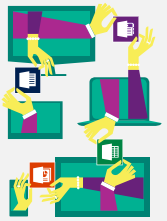
Visual Studio integration

What can you do with Visual Studio?

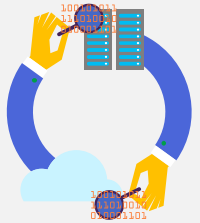


Author U-SQL scripts (with C# code)

Debug U-SQL and C# code



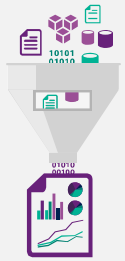
Submit and cancel U-SQL Jobs



Visualize physical plan of U-SQL query

Visualize and replay progress of job

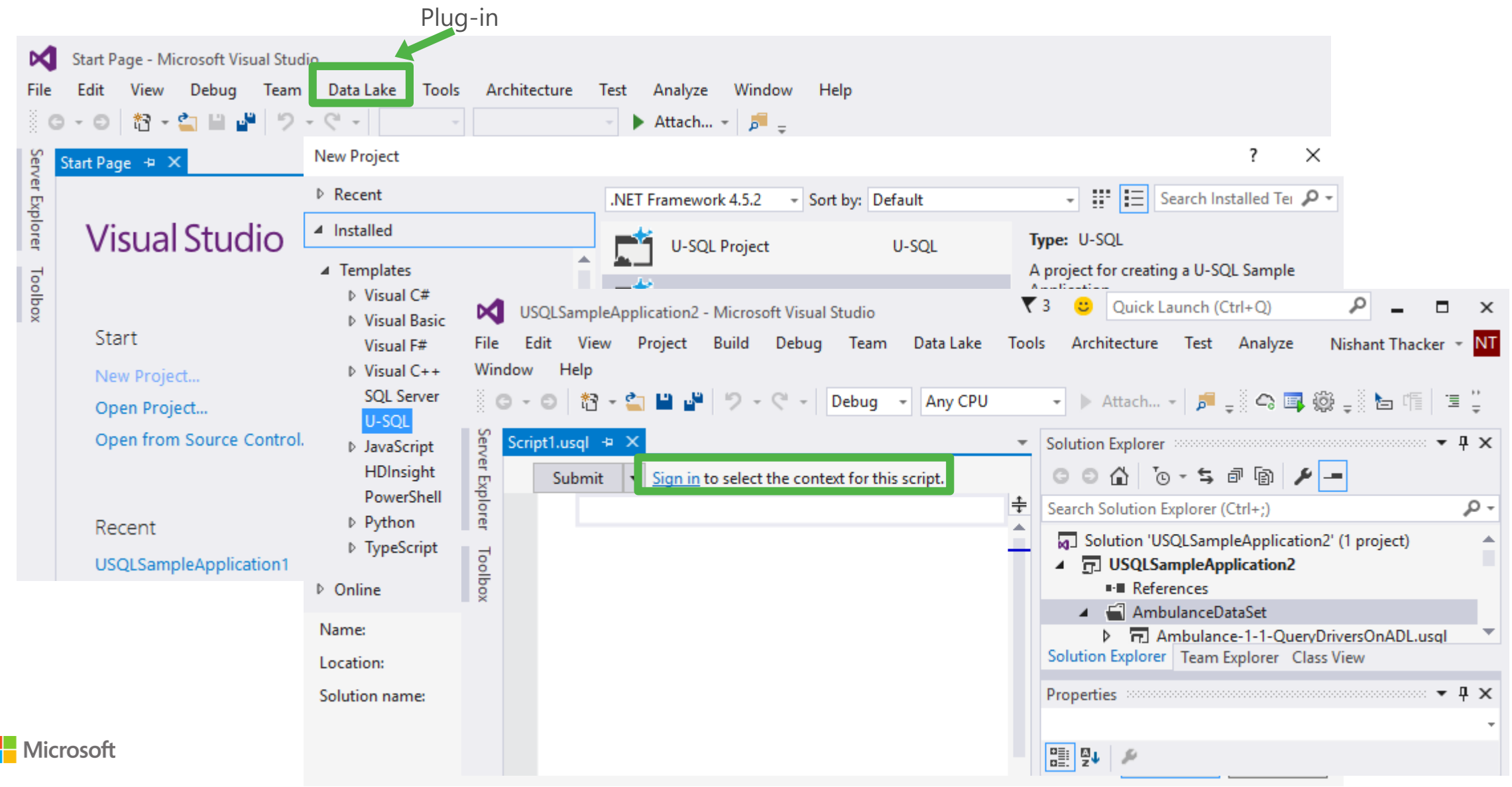
Fine-tune query performance



Create metadata objects

Browse metadata catalog

How to get going with ADL Tools for Visual Studio

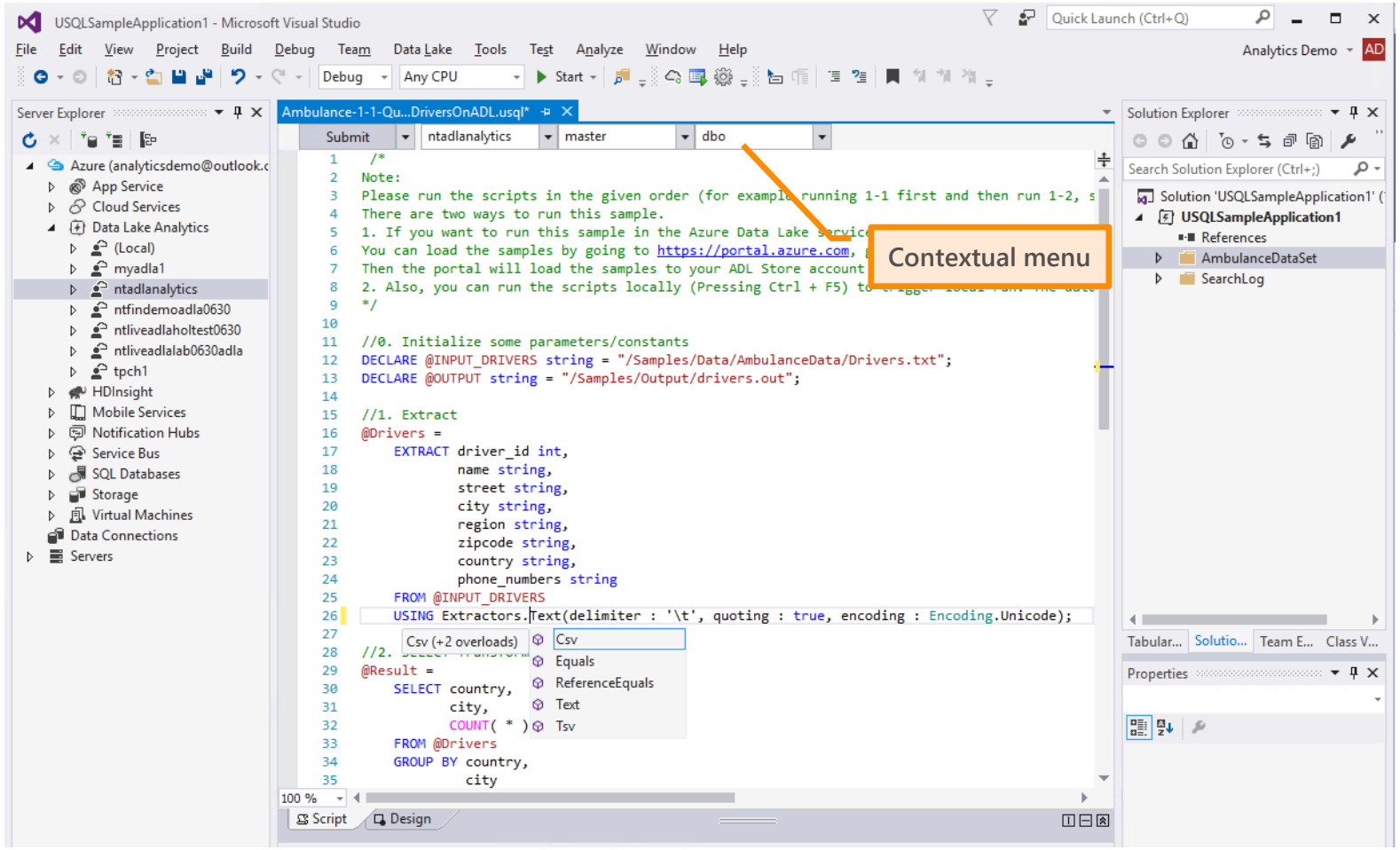


Authoring U-SQL queries

Visual Studio fully supports authoring U-SQL scripts

While editing, it provides:

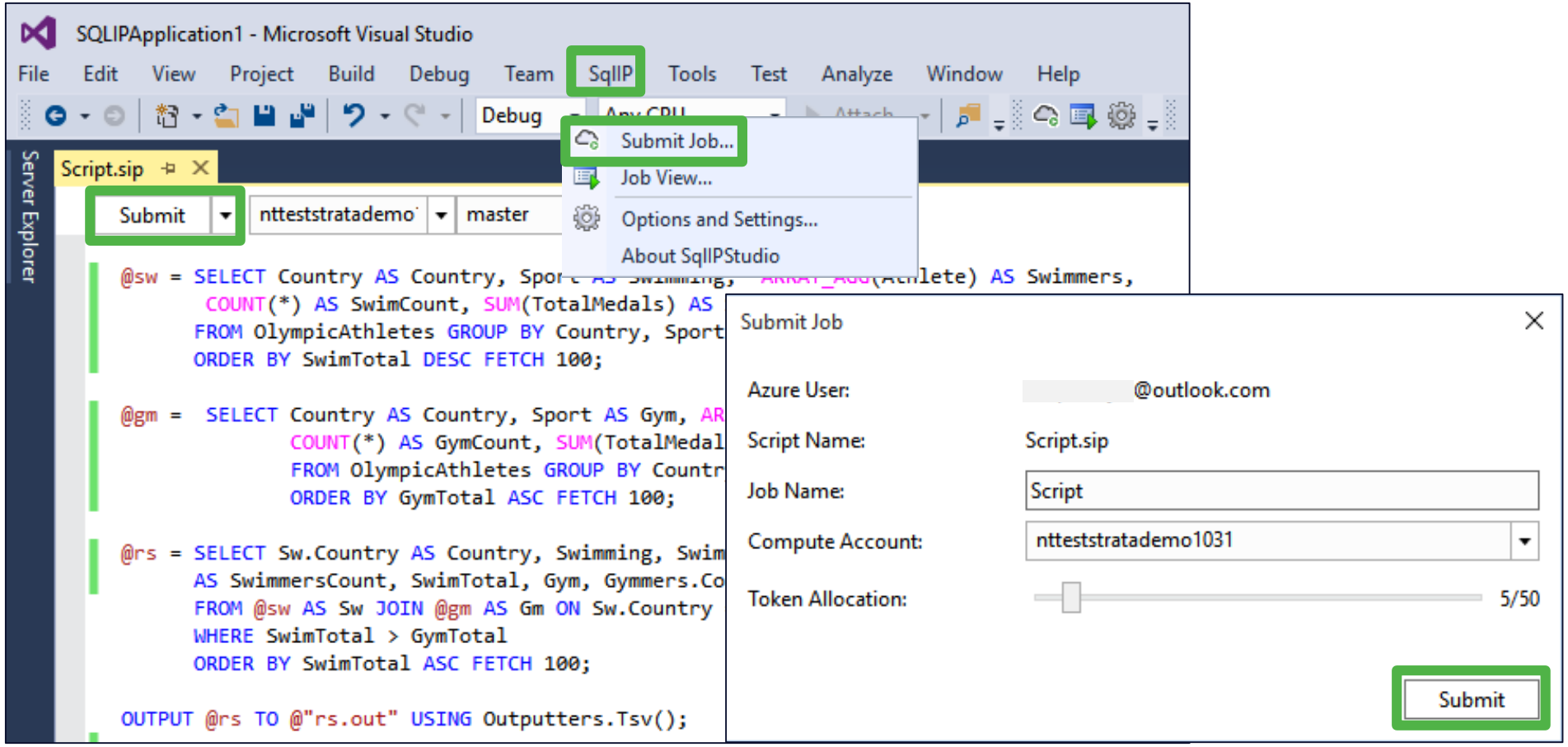
- IntelliSense
- Syntax color coding
- Syntax checking
- …



Submitting a U-SQL job

Jobs can be submitted directly from Visual Studio in two ways

You have to be logged into Azure and have to specify the target Azure Data Lake account.



Query design

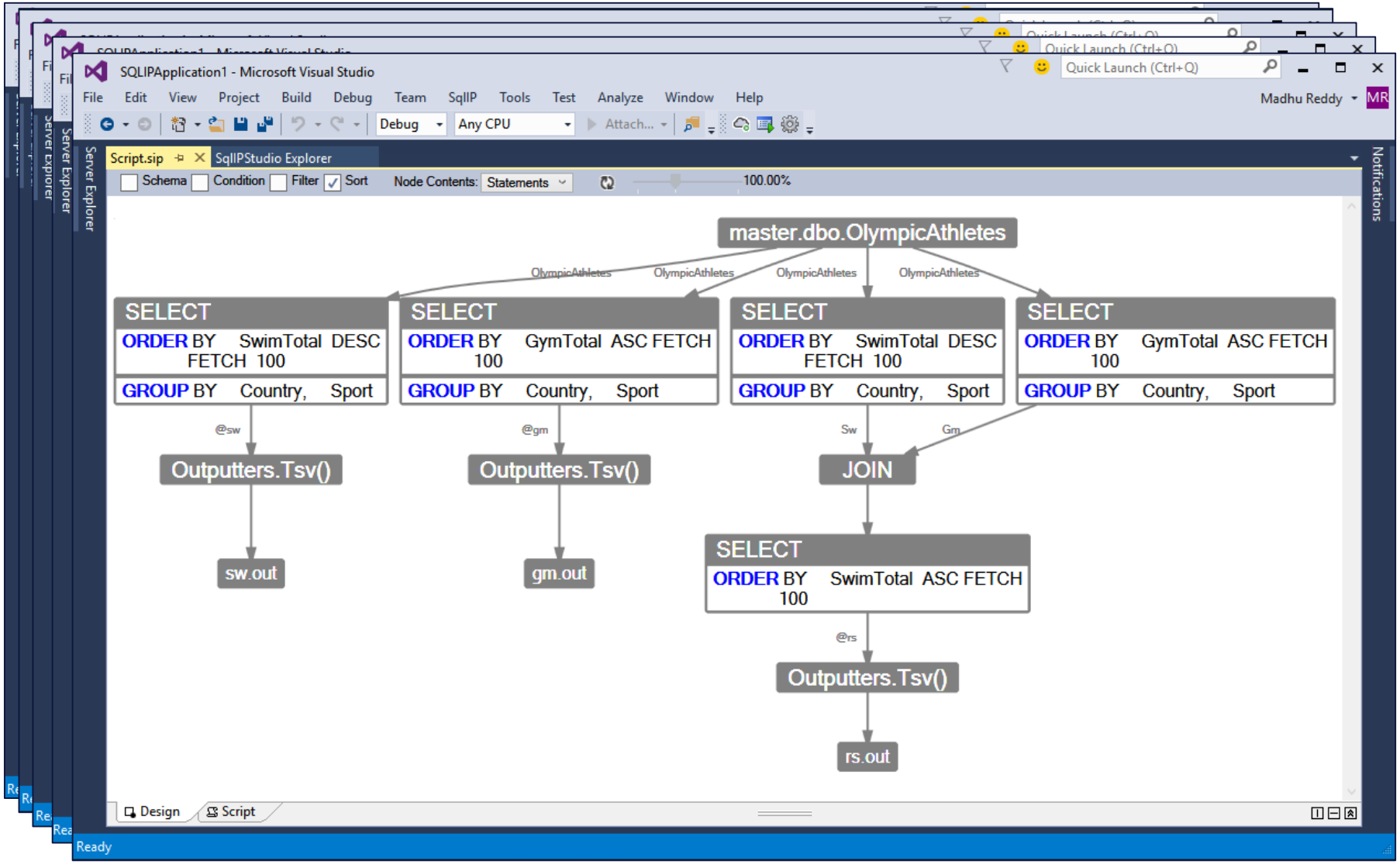
- U-SQL Studio lets you see the logical query design including:

Schema

Join conditions

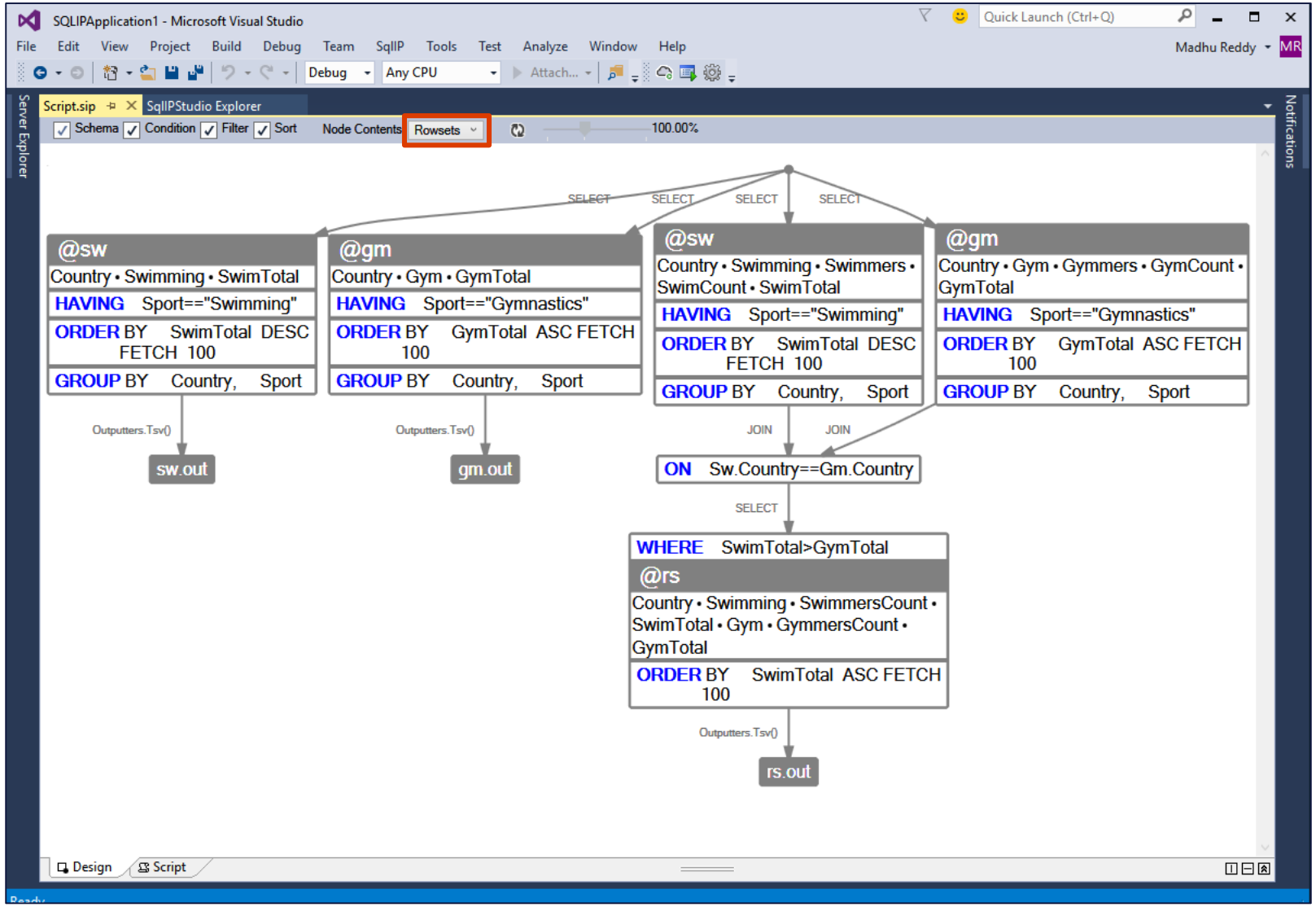
Filter plan

Sort plan



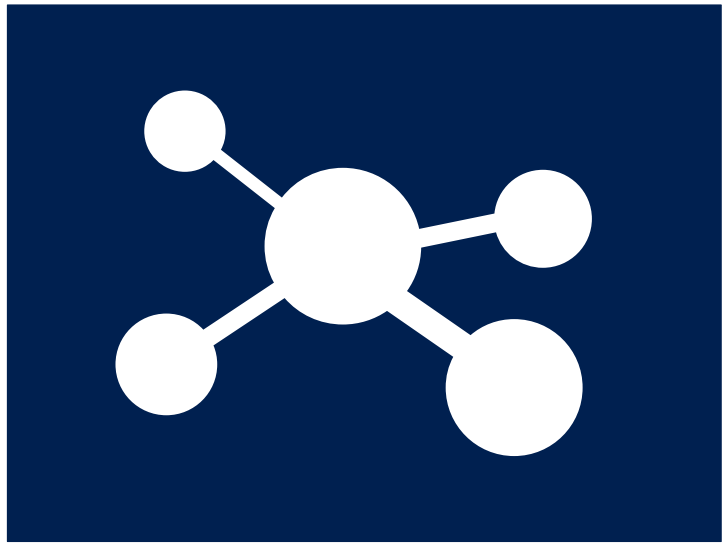
Query design -RowSet

- The query design can also be visualized in terms of the RowSets and the transformation applied to them.



Metadata objects

- ADL Analytics creates and stores a set of metadata objects in a catalog maintained by a metadata service
- Tables and TVFs are created by DDL statements (CREATE TABLE …)
- Metadata objects can be created directly through the Server Explorer



Azure Data Lake Analytics account

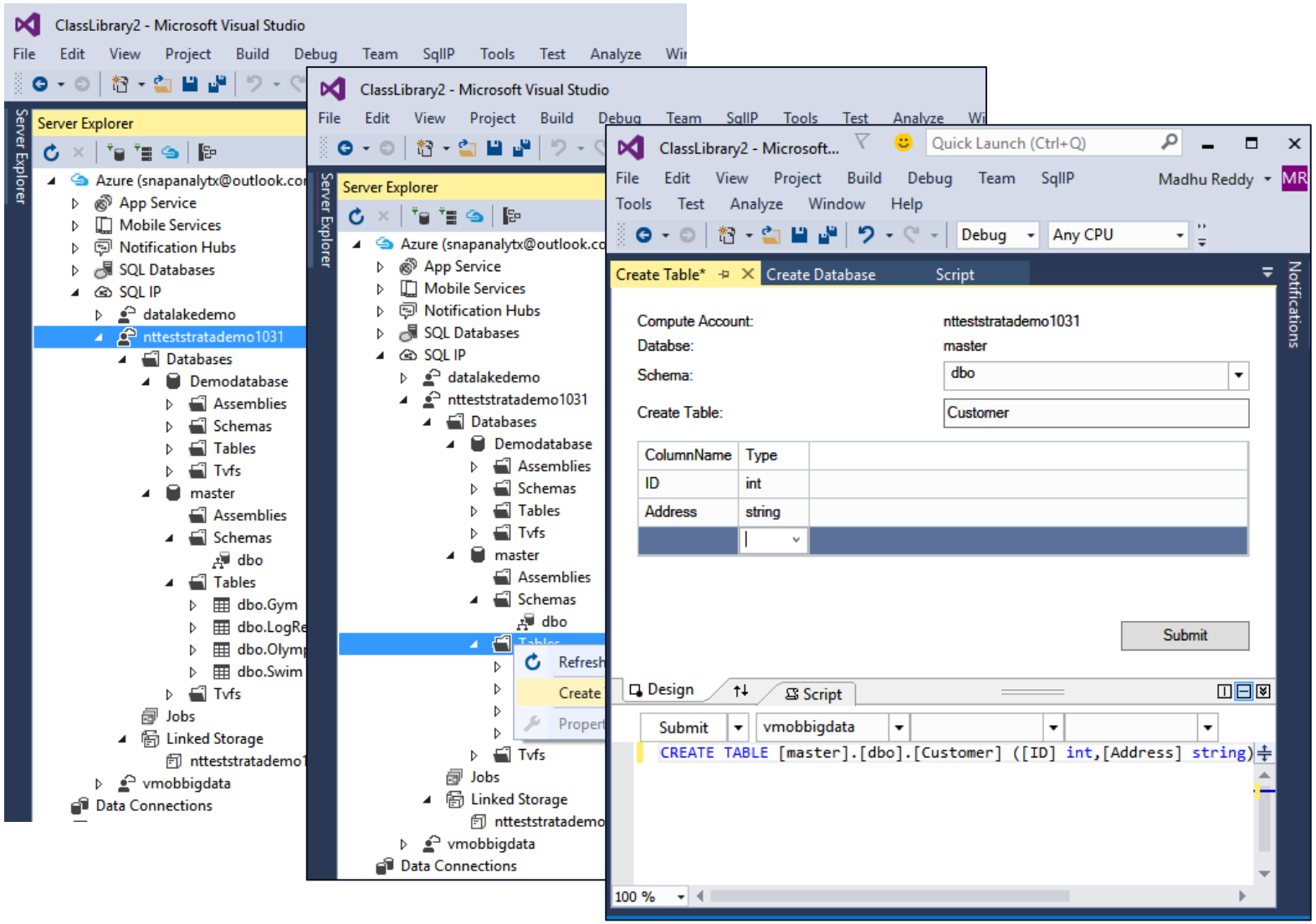
- Databases
  - Tables
  - Table valued functions
  - Jobs
  - Schemas
- Linked storage

Metadata catalog

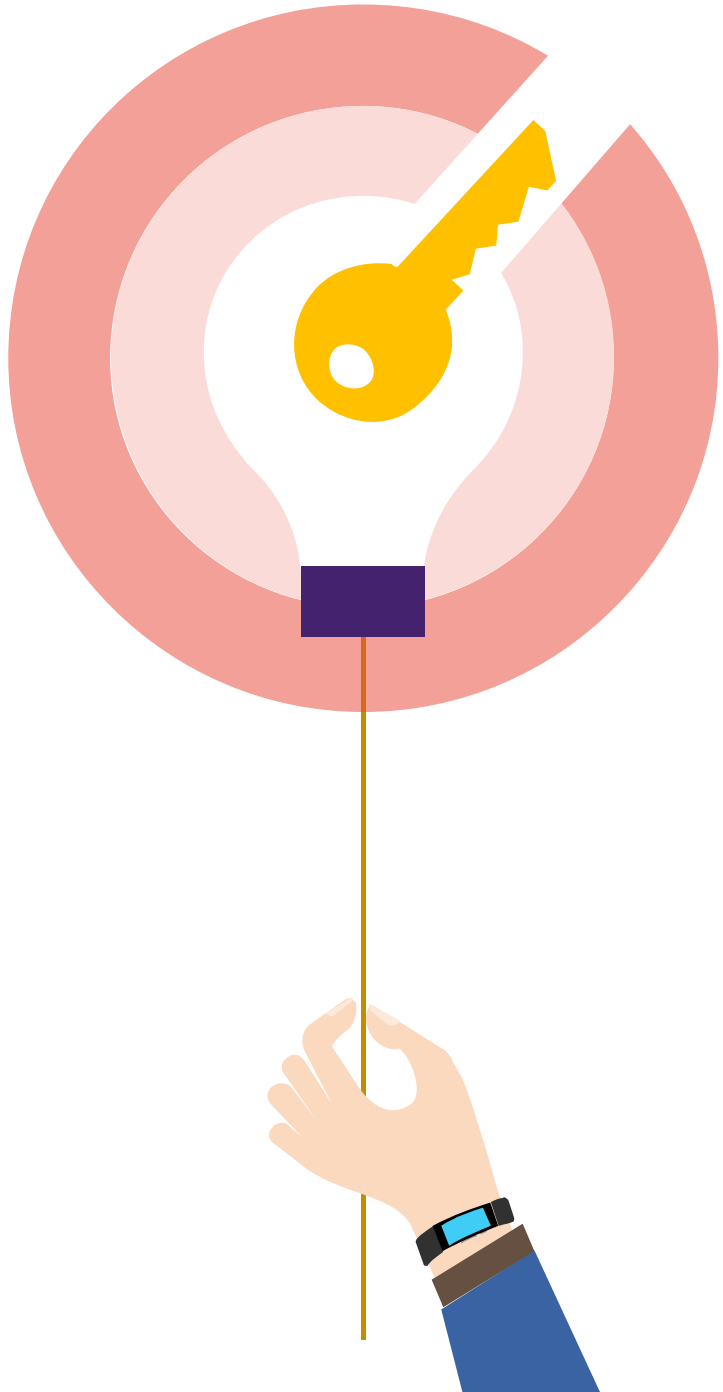
The metadata catalog can be browsed with the Visual Studio Server Explorer

Server Explorer lets you:

1. Create new tables, schemas and databases
2. Register assemblies



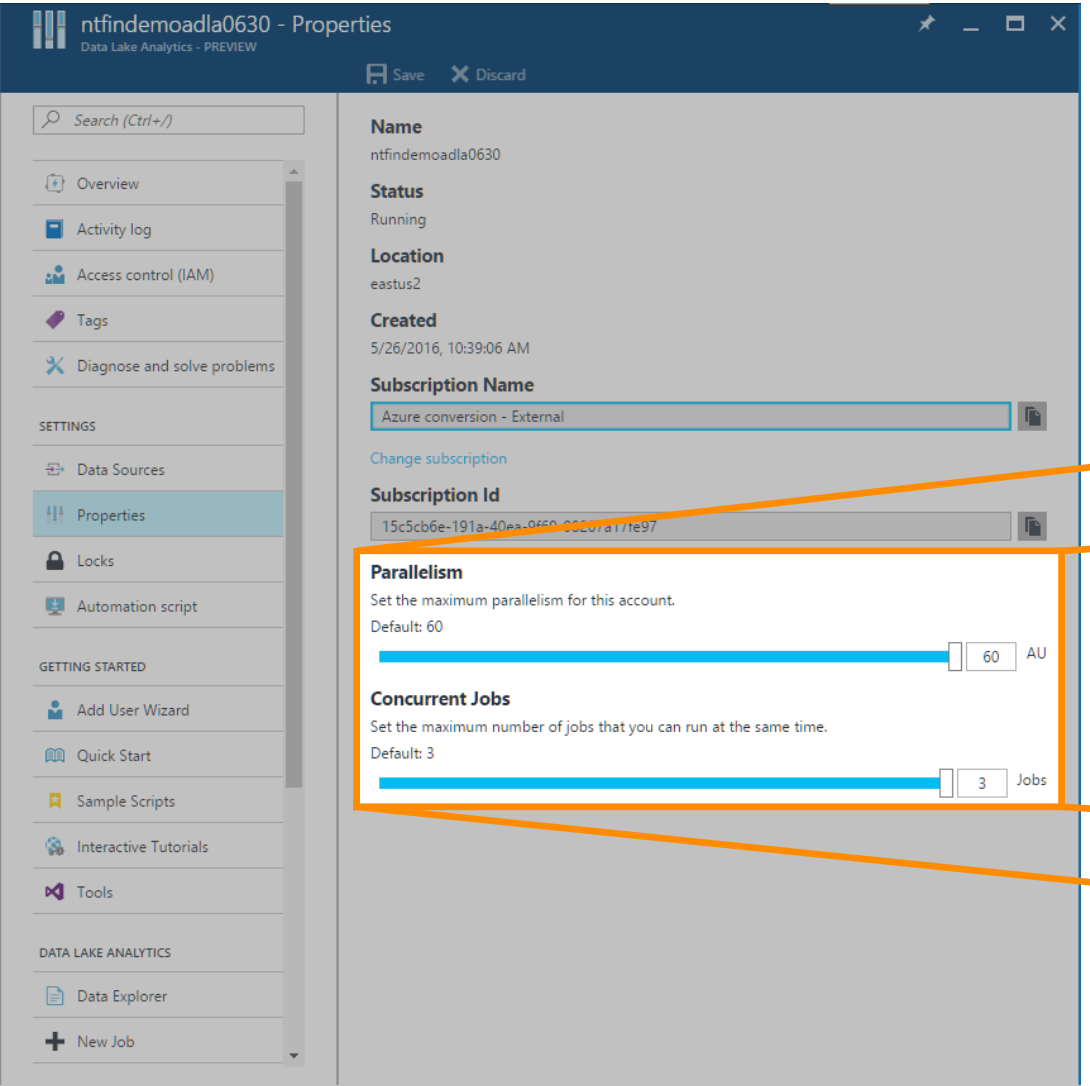
Key concepts



Microsoft

Analytics Account Properties

Azure Data Lake Analytics Unit



Parallelism of N = N ADLAUs

1 ADLAU ≈ a Container, 2 cores and 6 GB of memory

This is the number of ADLAUs for the account.

Parallelism

Set the maximum parallelism for this account.

Default: 60

60 AU

Concurrent Jobs

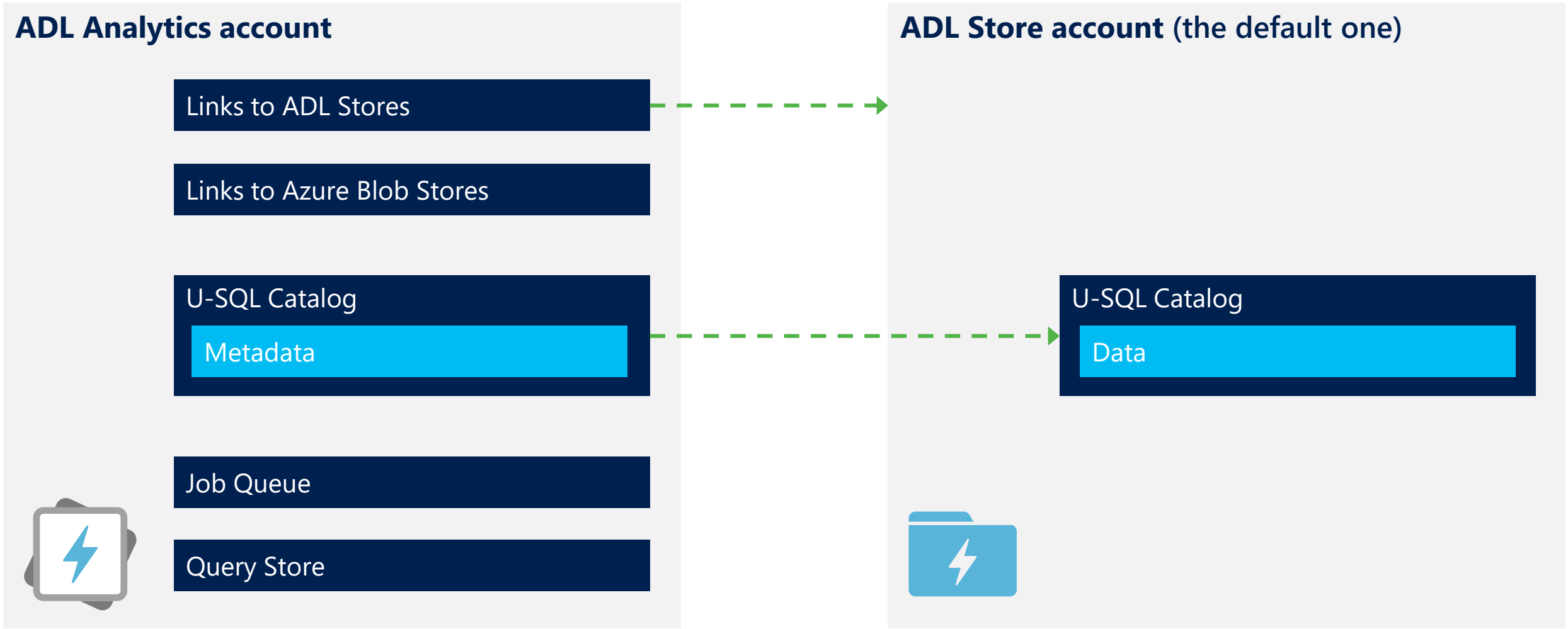
Set the maximum number of jobs that you can run at the same time.

Default: 3

3 Jobs

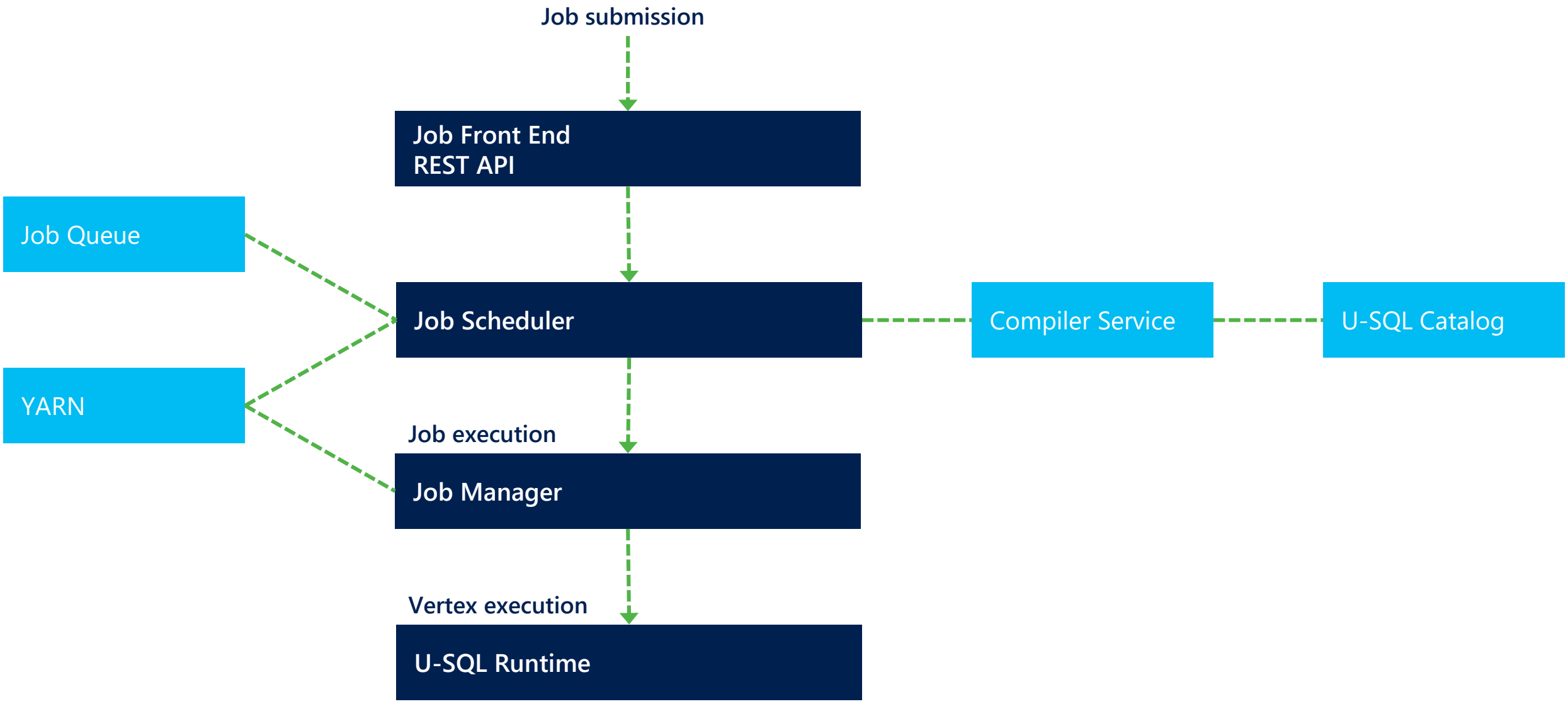
IGNORE THIS

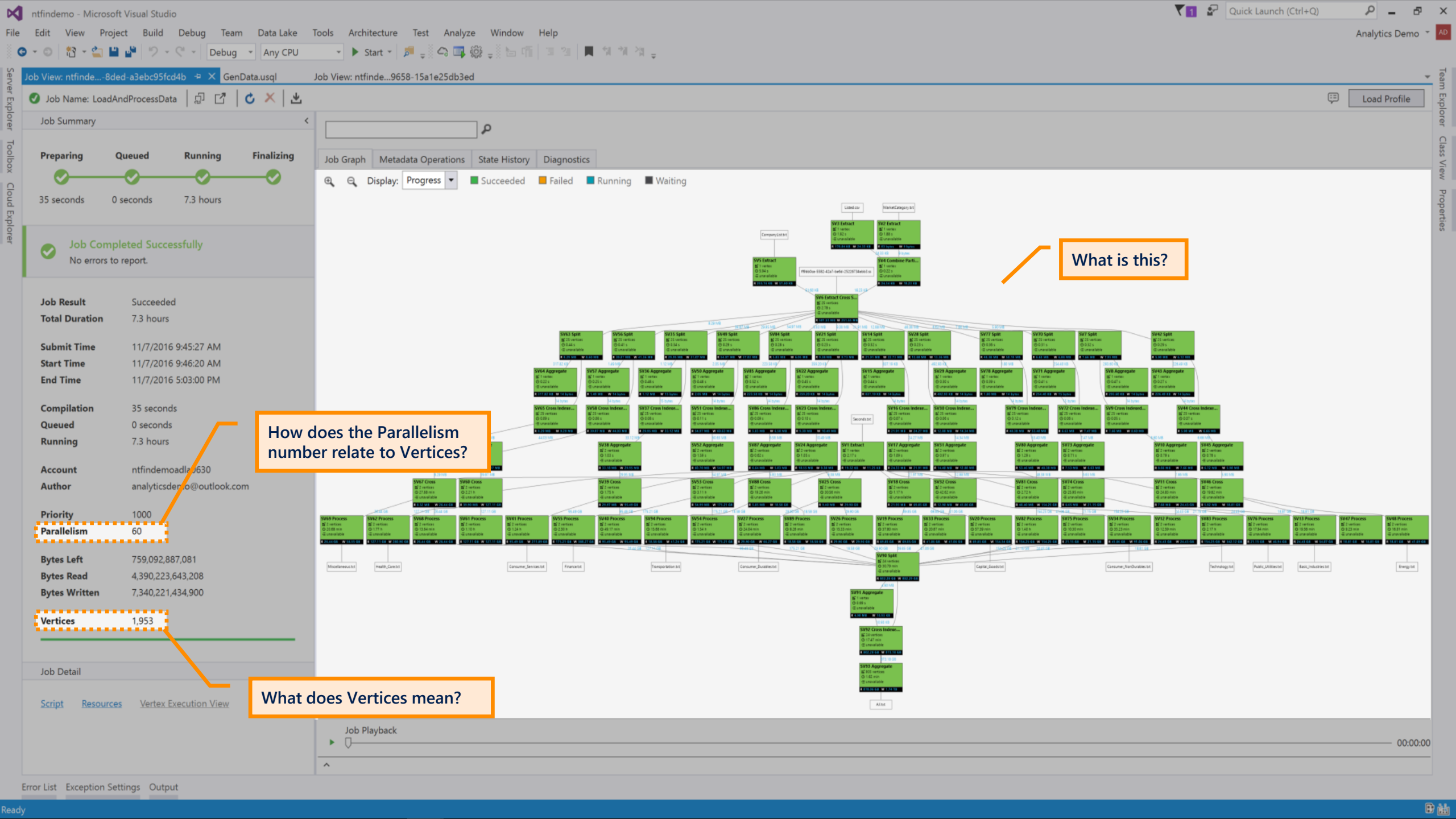
Data Lake account configuration



An ADL Store IS REQUIRED for ADL Analytics to function.

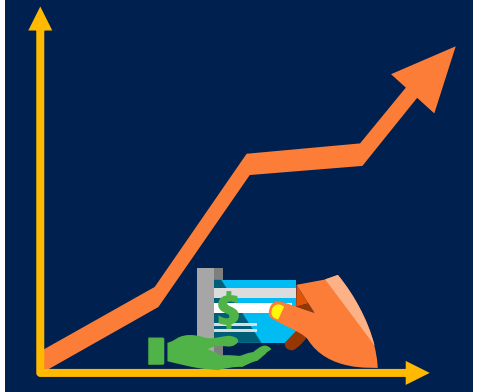
Simplified job workflow




ntfindemo - Microsoft Visual Studio
Job Summary
Preparing
Queued
Running
Finalizing
35 seconds
0 seconds
7.3 hours
Job Completed Successfully
No errors to report.
Job Result Succeeded
Total Duration 7.3 hours
Submit Time 11/7/2016 9:45:27 AM
Start Time 11/7/2016 9:46:20 AM
End Time 11/7/2016 5:03:00 PM
Compilation 35 seconds
Queued 0 seconds
Running 7.3 hours
Priority 1000
Parallelism 60
Bytes Left 759,092,887,081
Bytes Read 4,390,223,643,208
Bytes Written 7,340,221,434,900
Vertices 1,953
Job Detail
Script
Resources
Vertex Execution View
Job Graph
Metadata Operations
State History
Diagnostics
Display: Progress
Succeeded
Failed
Running
Waiting
Load Profile
What is this?
How does the Parallelism number relate to Vertices?
What does Vertices mean?
Job Playback
00:00:00

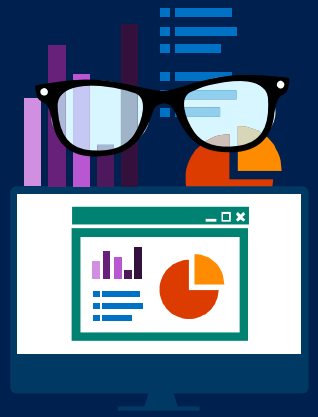
Azure Data Lake Analytics

Start in seconds
Scale instantly
Pay per job

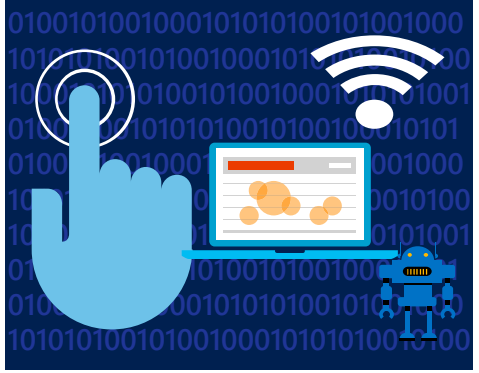


Develop massively parallel programs with simplicity

Debug and optimize your Big Data programs with ease



Virtualize your analytics

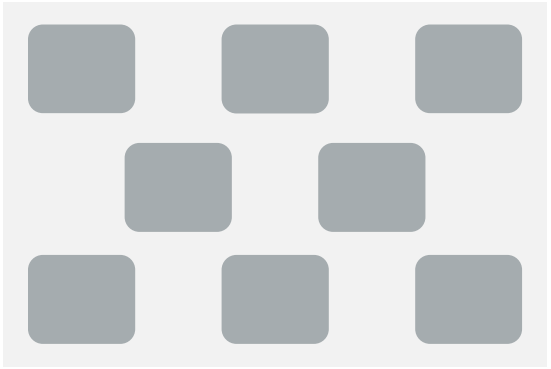


Enterprise-grade security, auditing and support

Concepts: jobs, stages and vertices

- Each job is broken into ‘n’ number of **vertexes**
- Each vertex is some work that needs to be done

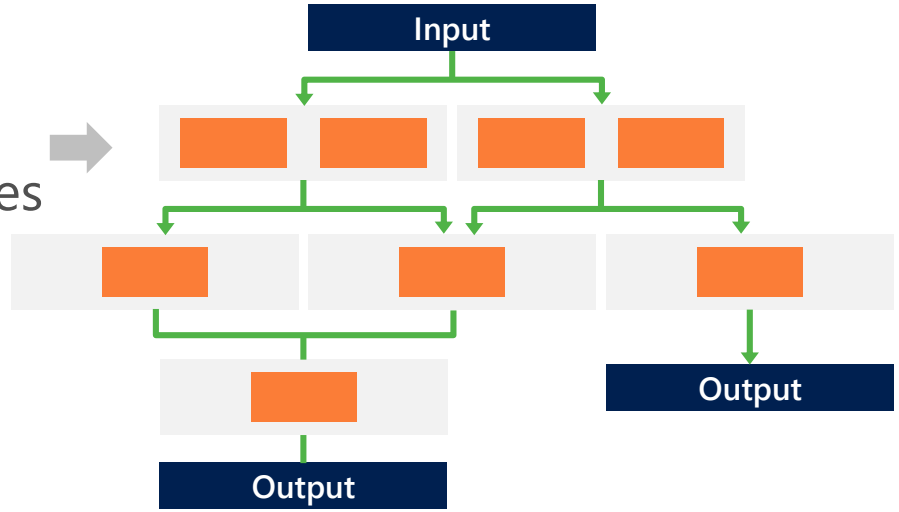
```
@sw = SELECT Country AS Country, Sport AS Swimming, SUM(TotalMedals) AS SwimTotal
      FROM OlympicAthletes GROUP BY Country, Sport HAVING Sport == "Swimming"
      ORDER BY SwimTotal DESC FETCH 100;

@gm =  SELECT Country AS Country, Sport AS Gym, SUM(TotalMedals) AS GymTotal
       FROM OlympicAthletes GROUP BY Country, Sport HAVING Sport == "Gymnastics"
       ORDER BY GymTotal ASC FETCH 100;

OUTPUT @sw TO @"sw.out" USING Outputters.Tsv();
OUTPUT @gm TO @"gm.out" USING Outputters.Tsv();

@sw = SELECT Country AS Country, Sport AS Swimming,  ARRAY_AGG(Athlete) AS Swimmers,
       COUNT(*) AS SwimCount, SUM(TotalMedals) AS SwimTotal
      FROM OlympicAthletes GROUP BY Country, Sport HAVING Sport == "Swimming"
      ORDER BY SwimTotal DESC FETCH 100;

@gm =  SELECT Country AS Country, Sport AS Gym, ARRAY_AGG(Athlete) AS Gymmers,
              COUNT(*) AS GymCount, SUM(TotalMedals) AS GymTotal
              FROM OlympicAthletes GROUP BY Country, Sport HAVING Sport == "Gymnastics"
              ORDER BY GymTotal ASC FETCH 100;

@rs = SELECT Sw.Country AS Country, Swimming, Swimmers.Count
      AS SwimmersCount, SwimTotal, Gym, Gymmers.Count AS GymmersCount, GymTotal
      FROM @sw AS Sw JOIN @gm AS Gm ON Sw.Country == Gm.Country
      WHERE SwimTotal > GymTotal
      ORDER BY SwimTotal ASC FETCH 100;

OUTPUT @rs TO @"rs.out" USING Outputters.Tsv();
```



- Vertexes are organized into **stages**
  - Vertexes in each stage do the same work on the same data
  - Vertex in one stage may depend on a vertex in a earlier stage
- Stages themselves are organized into an acyclic graph

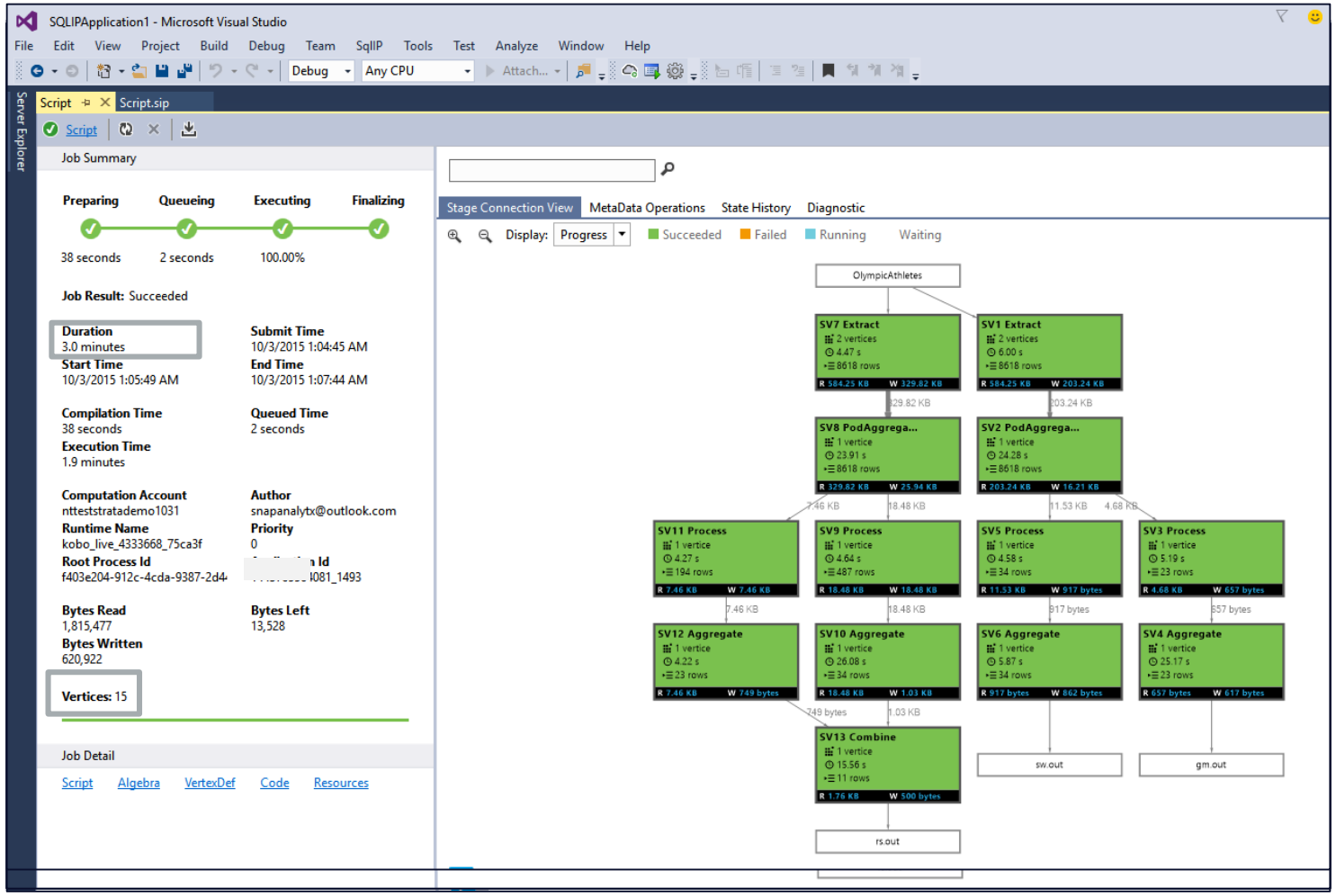
6 Stages

8 Vertexes

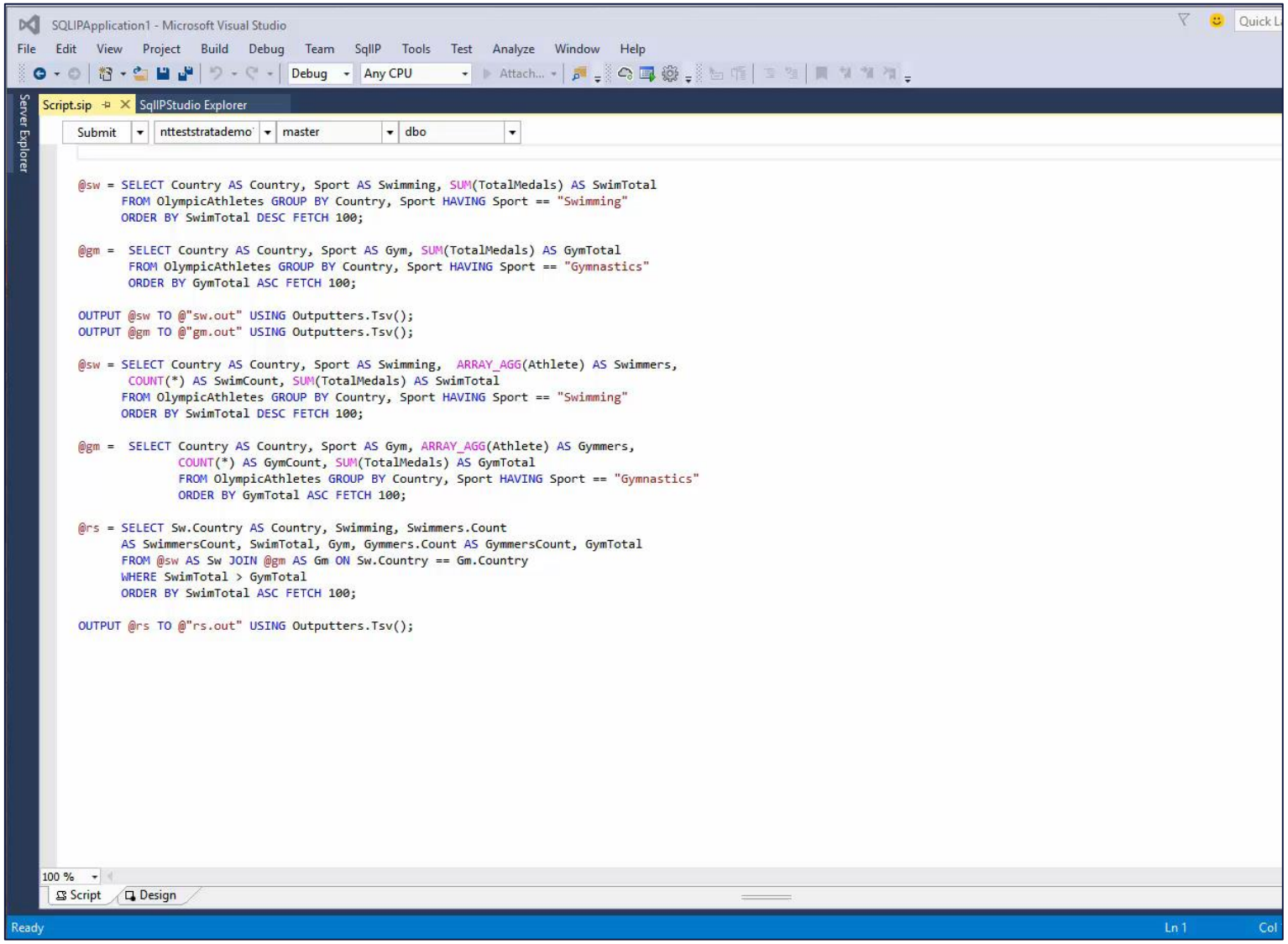


Job execution graph

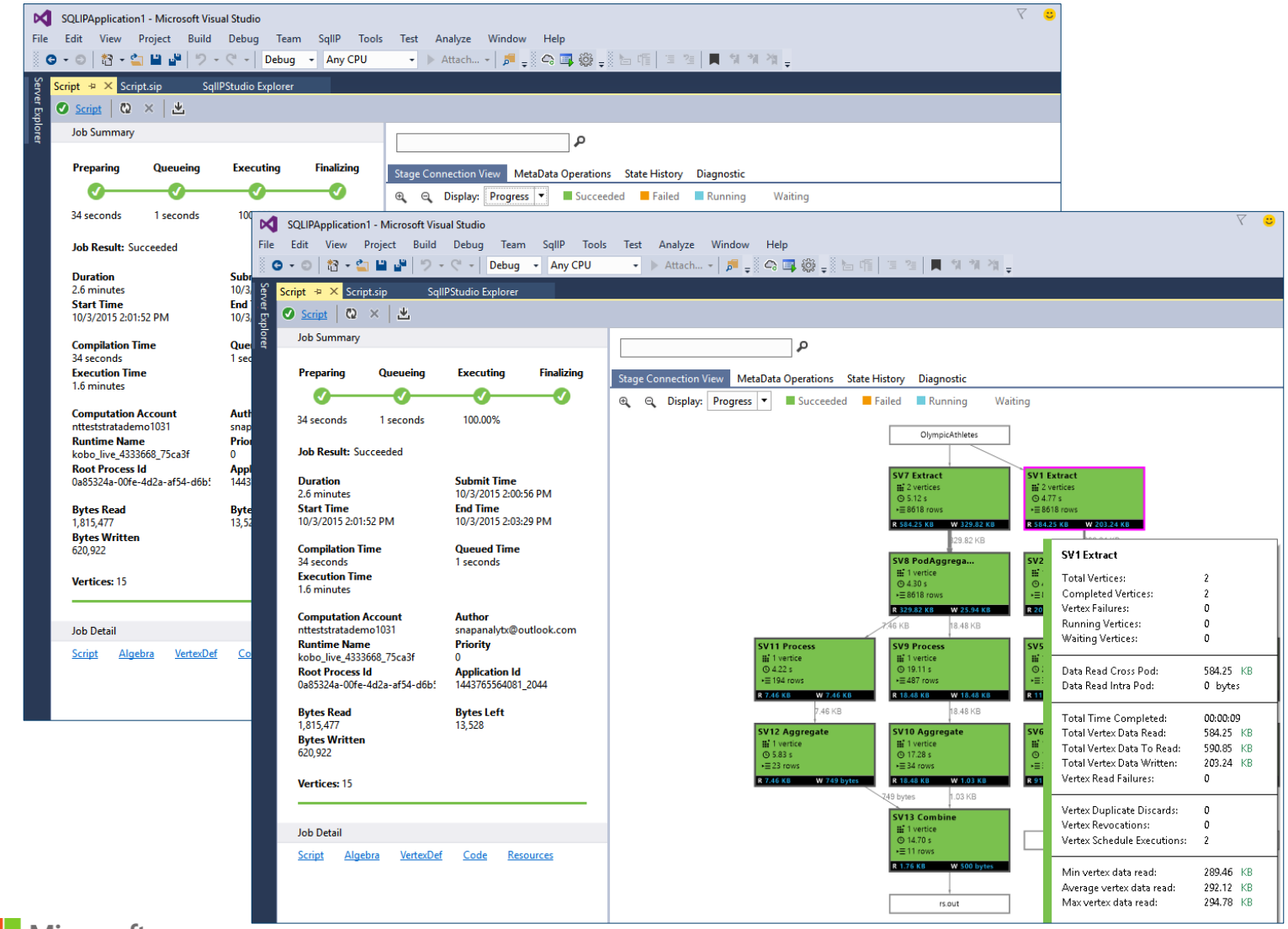
- After a job is submitted the progress of the execution of the job as it goes through the different stages is shown and updated continuously
- Important stats about the job are also displayed and updated continuously



Job execution graph (video)



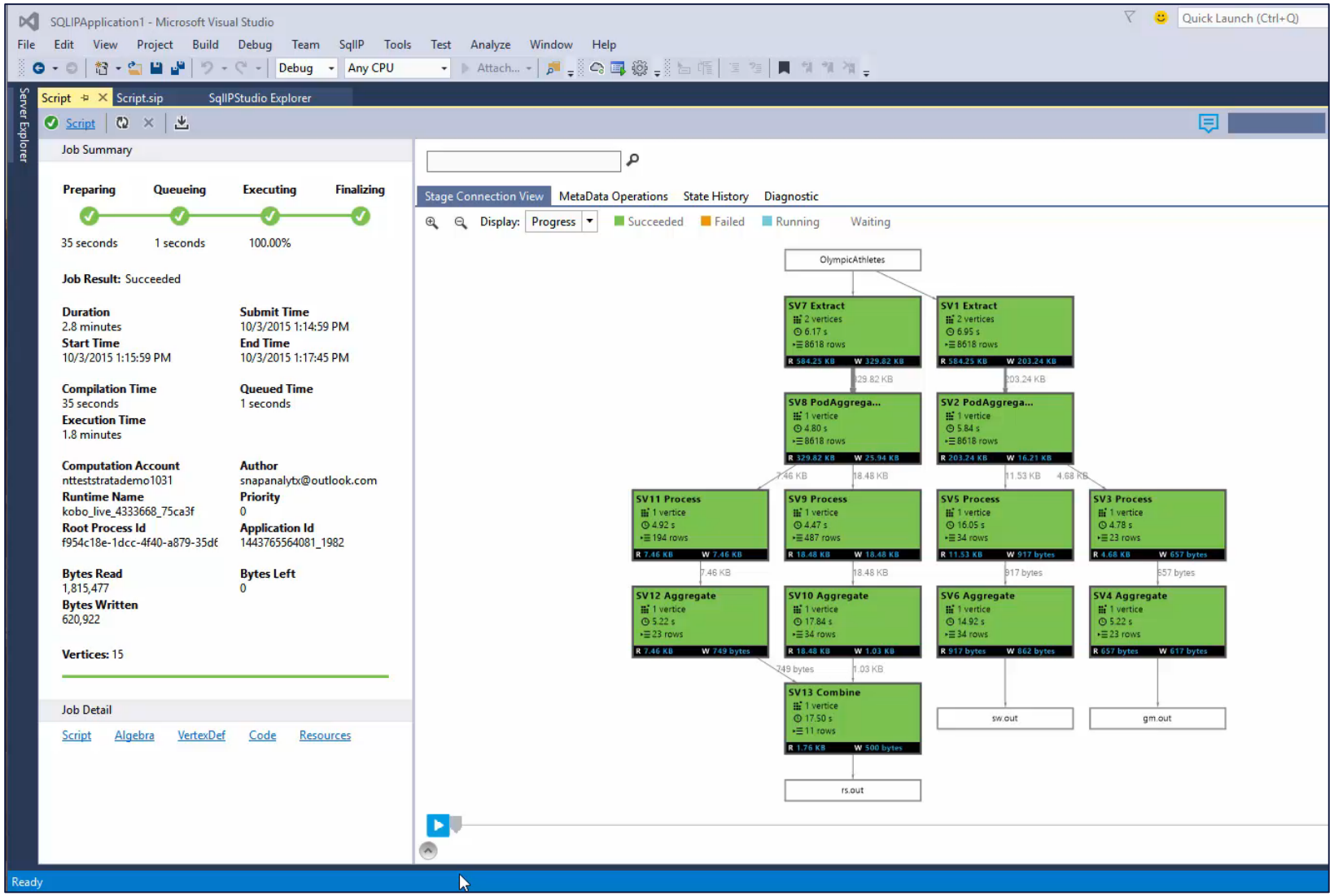
Job execution graph – node details



Hovering over them, you can get details about the nodes.

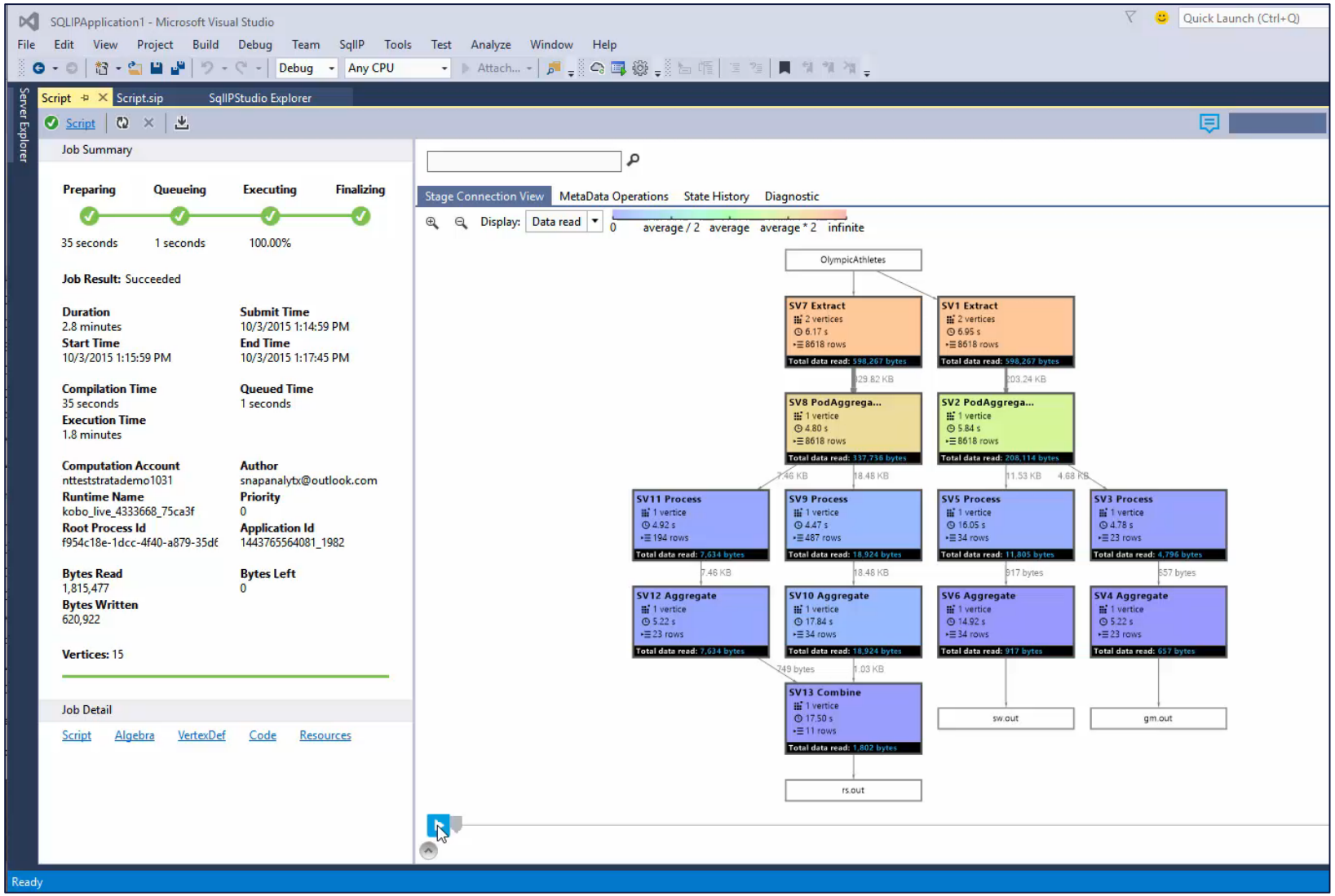
Job execution “progress” playback (video)

- For performance tuning, identify bottlenecks and debugging, you can playback the job execution graph



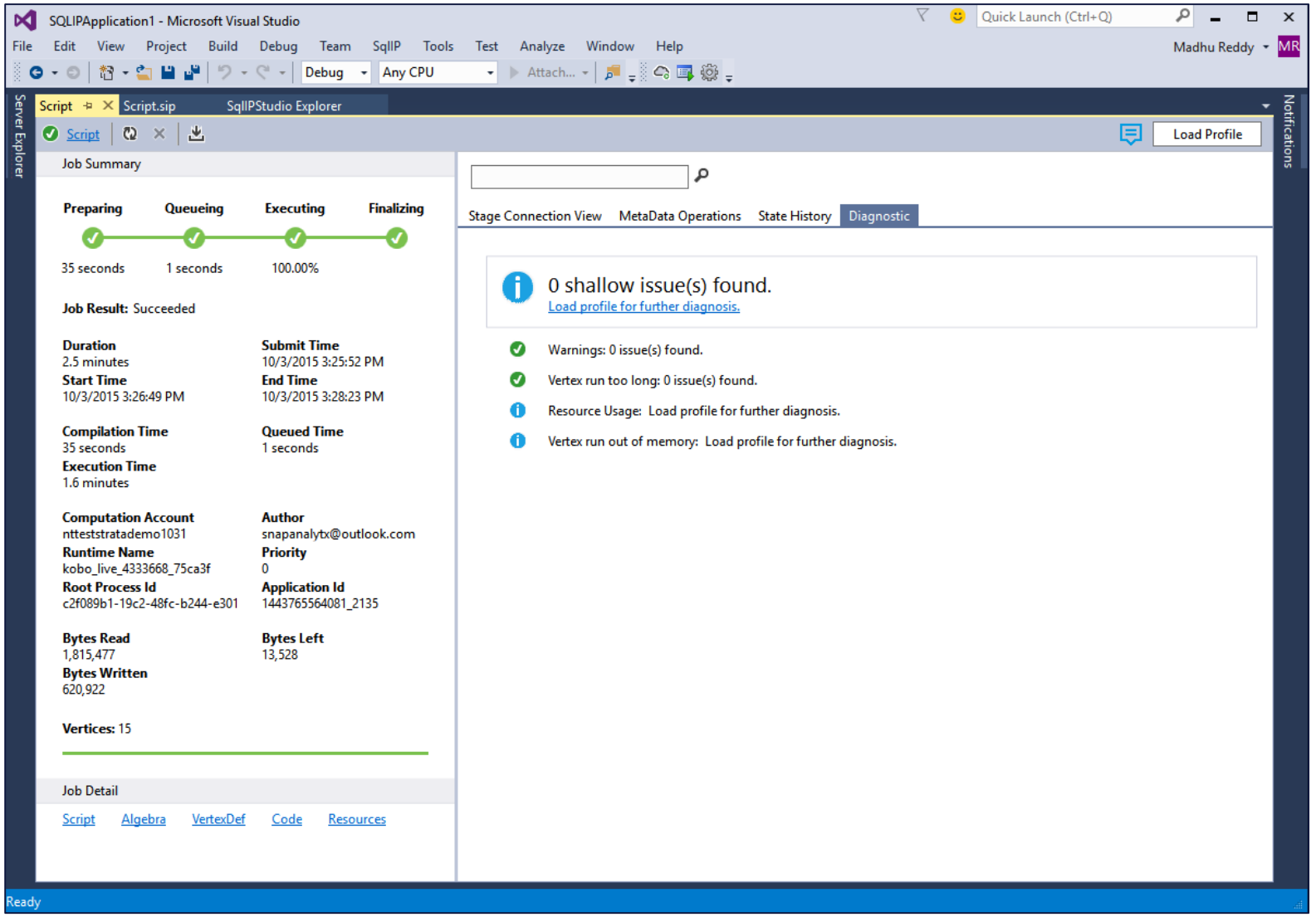
“Data read” playback (video)

- For performance tuning, identify bottlenecks and debugging, you can playback the job execution graph



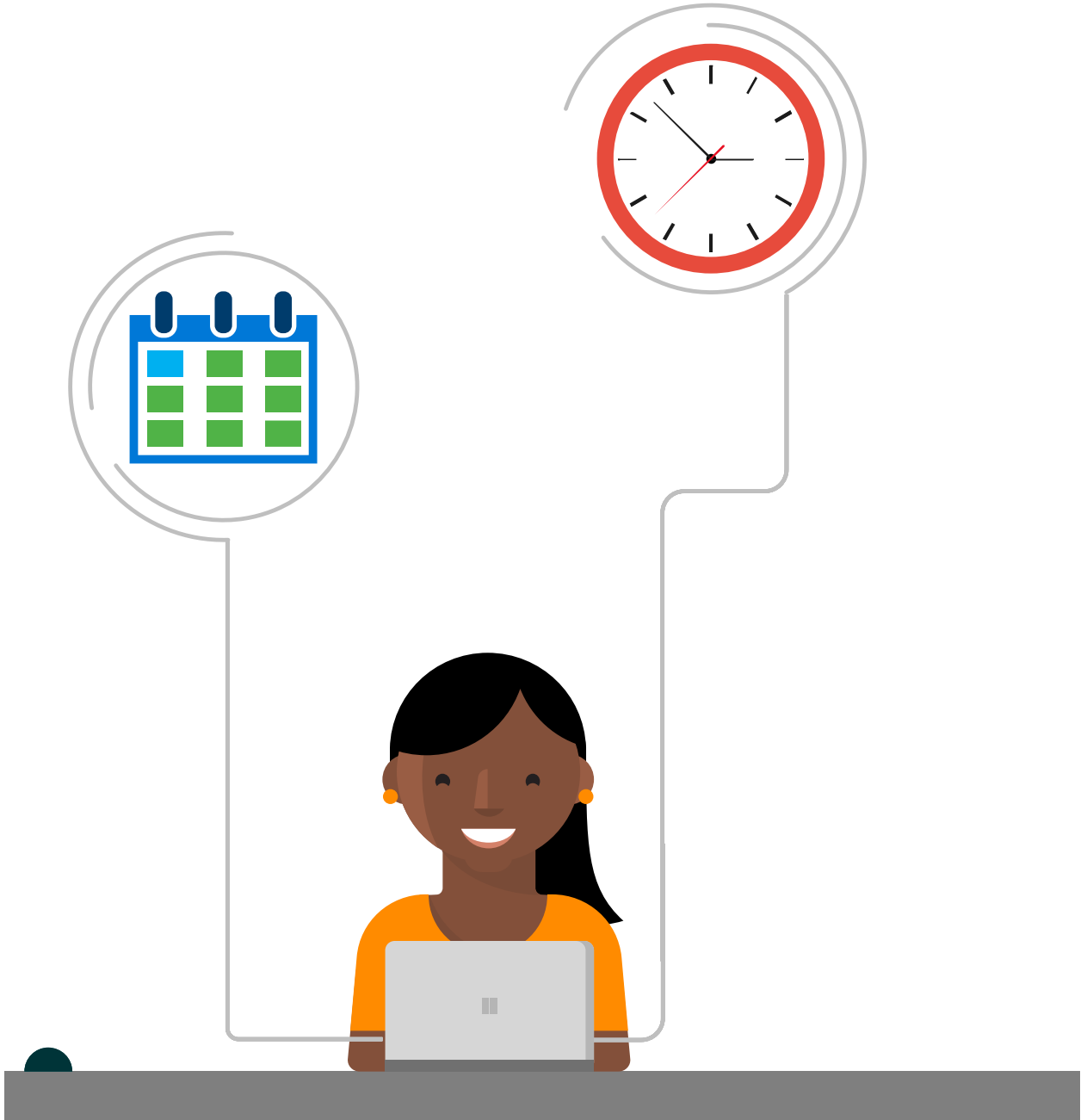
Job diagnostics

- Diagnostics information is shown to help with debugging and performance issues



Job scheduling

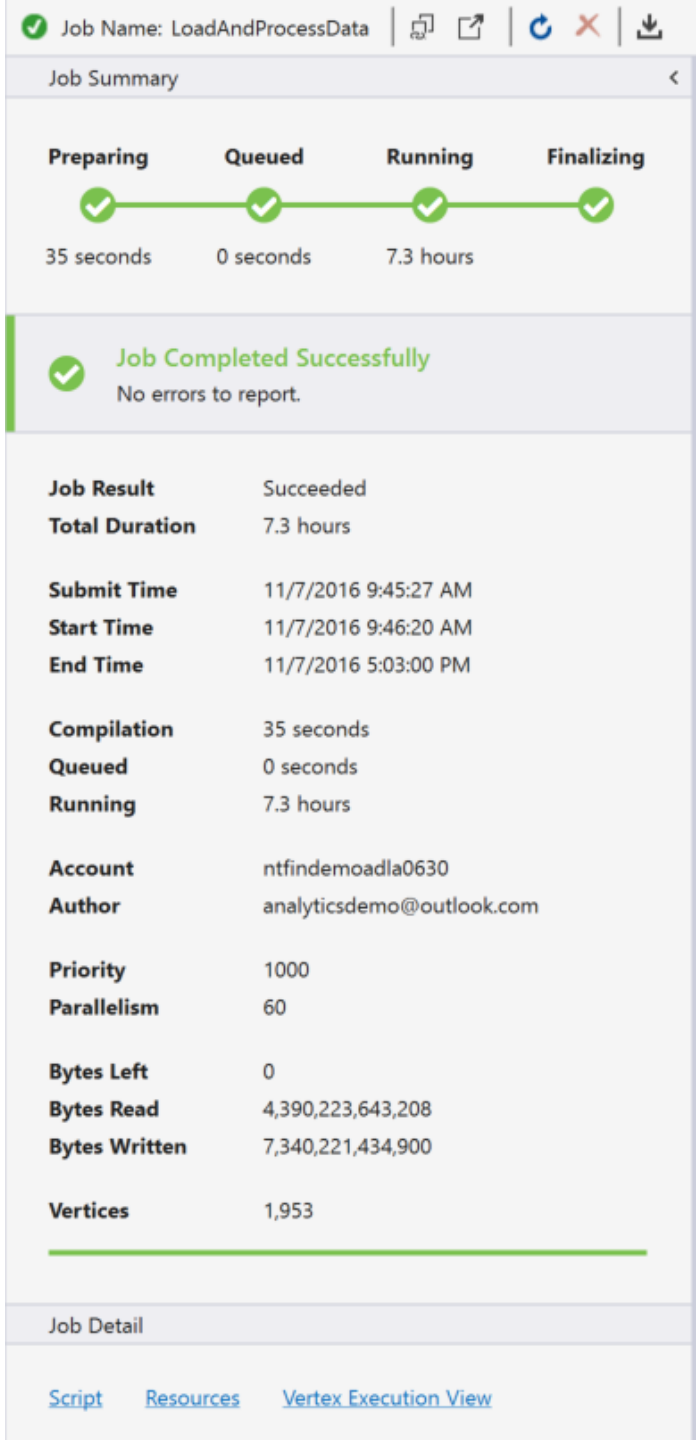
States, queue, priority



Microsoft

Visual Studio: Job states

UX	Job State
Preparing	New
	Compiling
	The script is being compiled by the Compiler Service
Queued	Queued
	All jobs enter the queue.
	Scheduling
	Are there enough ADLAUs to start the job?
	Starting
	If yes, then allocate those ADLAUs for the job
Running	Running
Finalizing	The U-SQL runtime is now executing the code on 1 or more ADLAUs or finalizing the outputs
Ended (Succeeded, Failed, Cancelled)	Ended
	The job has concluded.



Why does a job get queued?

Local cause

Possible condition:

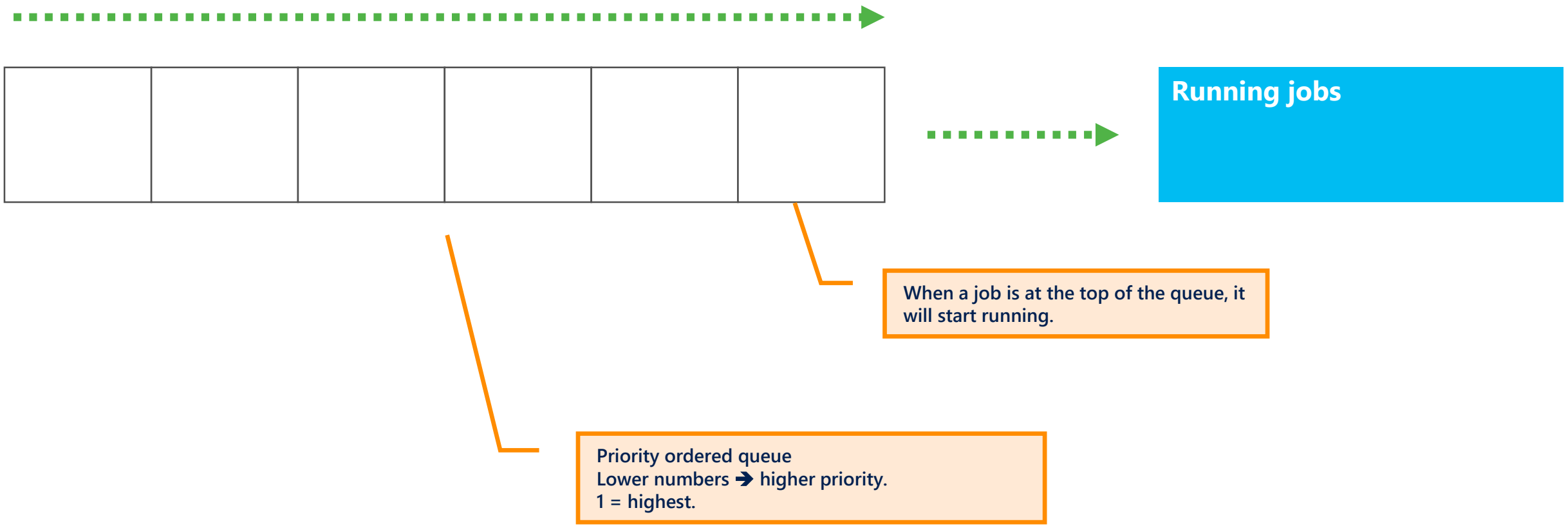
- Not enough containers available to your account

Global cause (very rare)

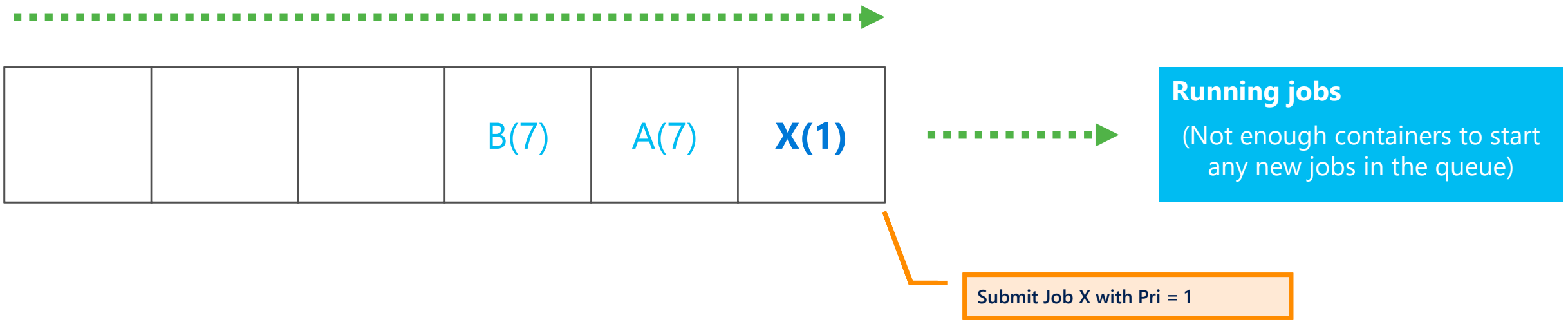
Possible conditions:

- System-wide shortage of containers
- System-wide shortage of bandwidth

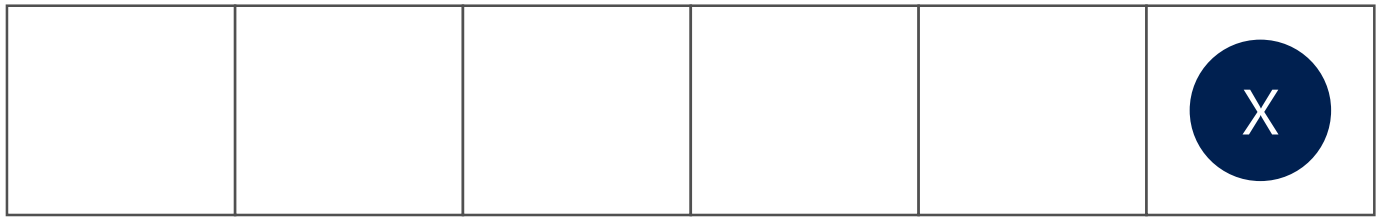
The job queue



Priority controls queue order



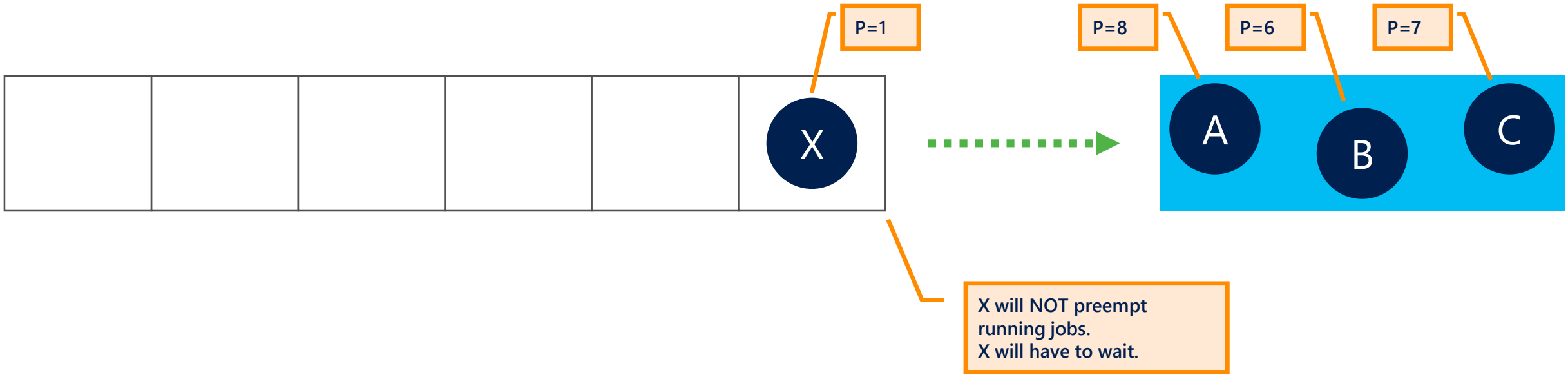
Jobs only start if all requested containers can be reserved



Running jobs
- Using N containers
- Only 3 containers available to the account

X needs 4 containers
X will have to wait

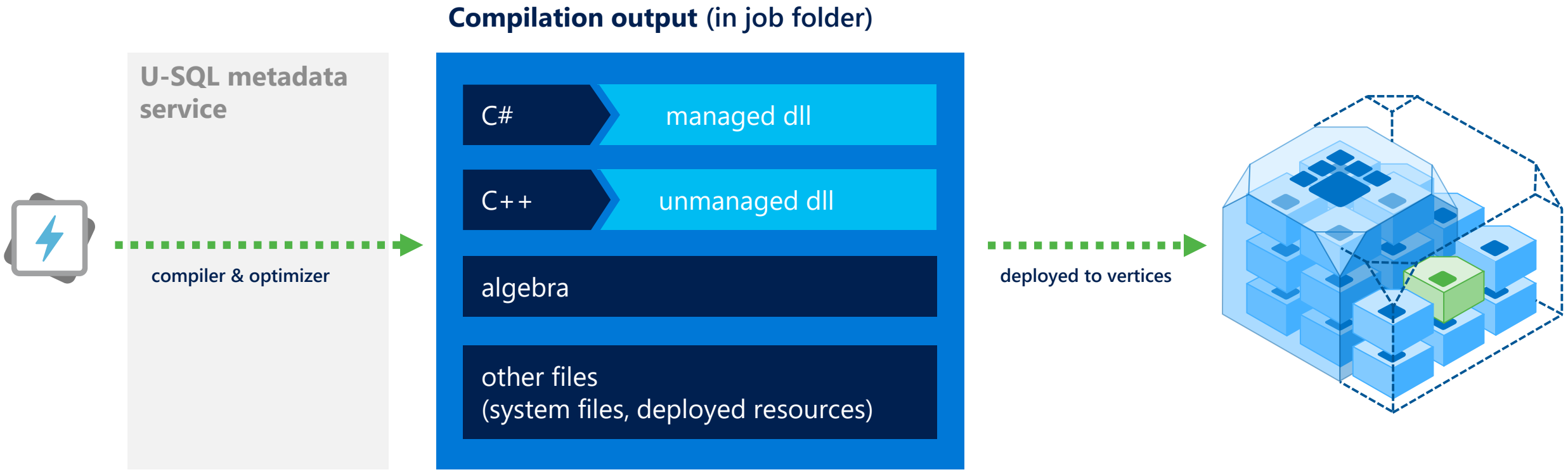
Priority doesn’t preempt running jobs



U-SQL job compilation

Microsoft

U-SQL compilation process



The job folder

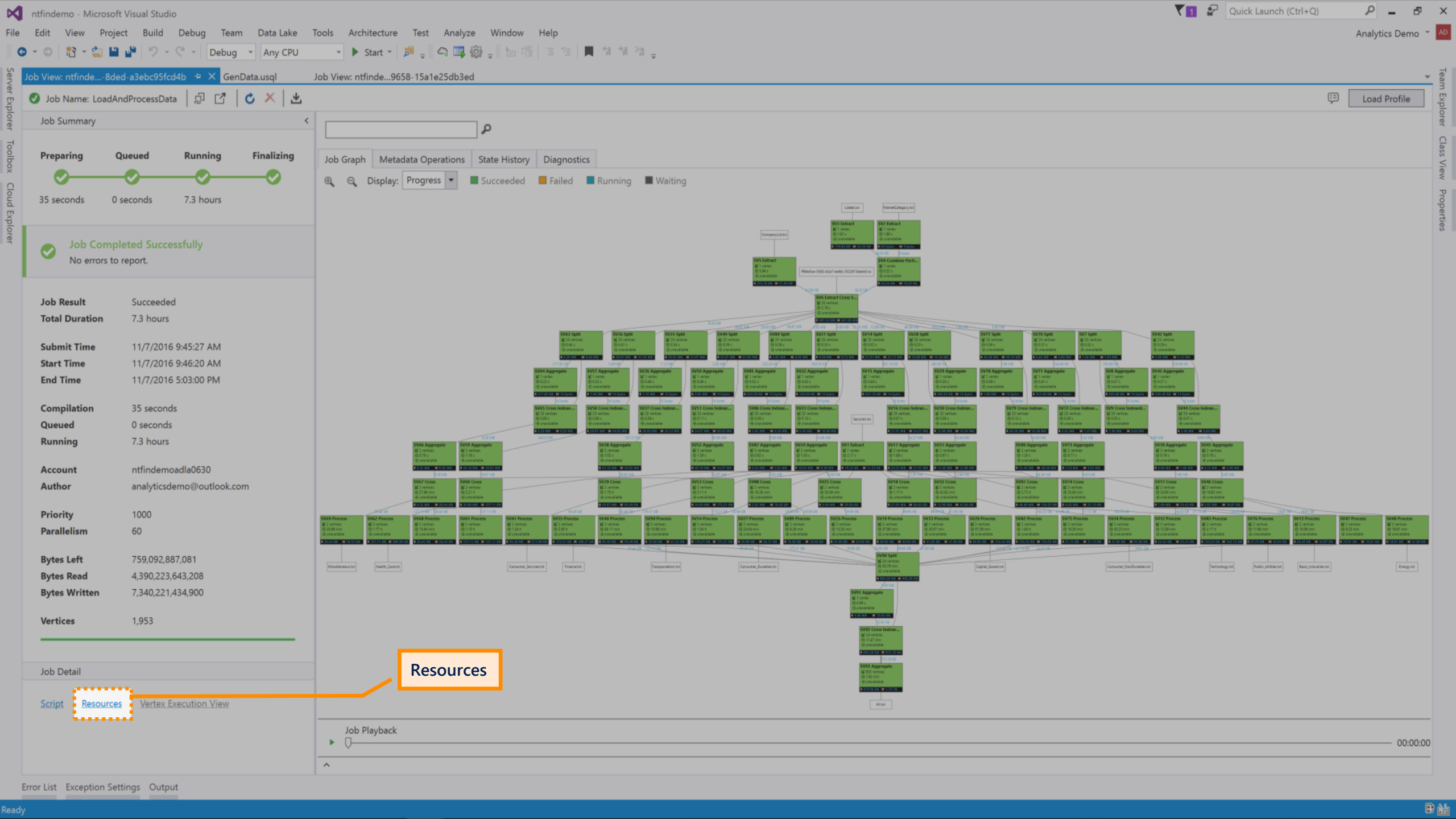
Inside the Default ADL Store:

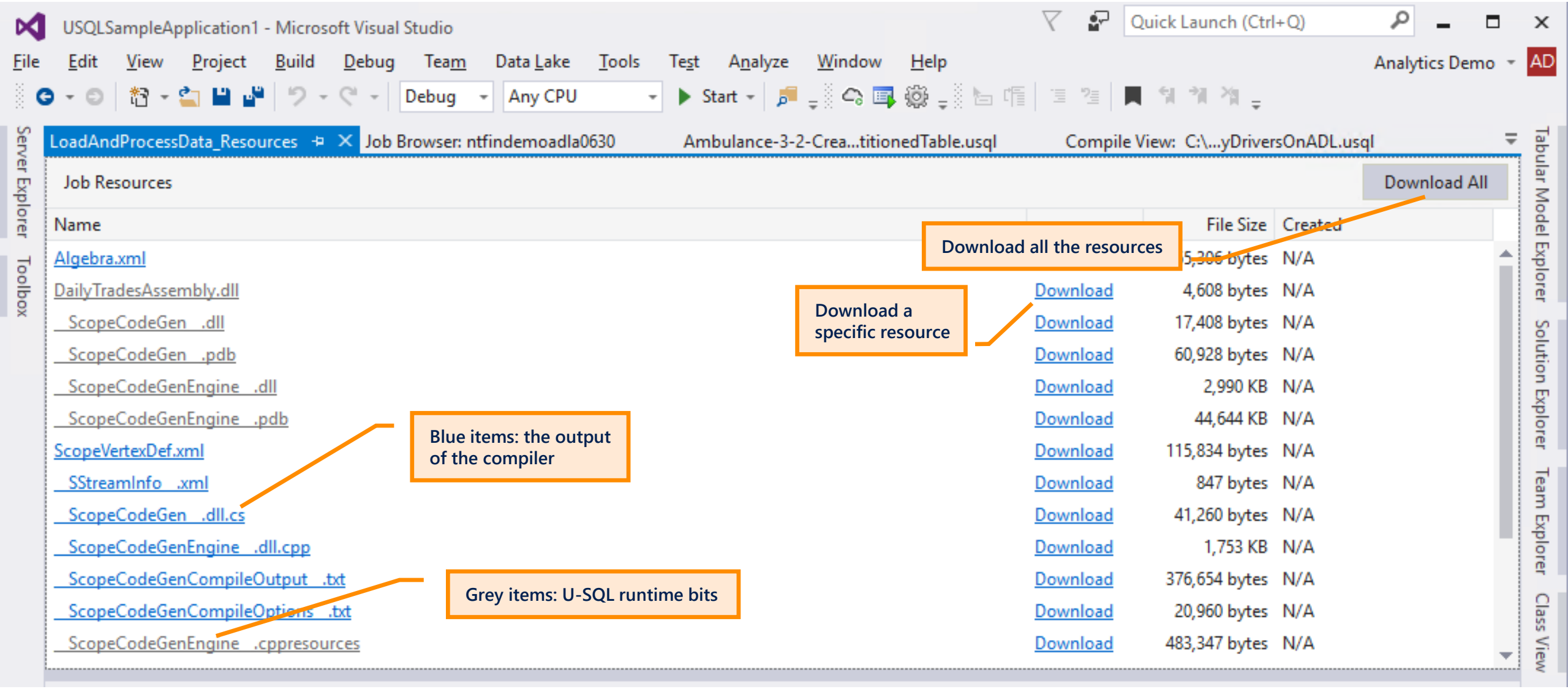
General Form

```
/system/jobservice/jobs/Usql/
        YYYY/MM/DD/hh/mm/
                JOBID
```

An Example

```
/system/jobservice/jobs/Usql/
        2016/01/20/00/00/
                17972fc2-4737-48f7-81fb-49af9a784f64
```


ntfindemo - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Data Lake Tools Architecture Test Analyze Window Help
Analytics Demo
Debug
Any CPU
Start
Job View: ntfinde...-8ded-a3ebc95fcd4b
GenData.usql
Job View: ntfinde...9658-15a1e25db3ed
Job Name: LoadAndProcessData
Load Profile
Job Summary
Preparing
Queued
Running
Finalizing
35 seconds
0 seconds
7.3 hours
Job Completed Successfully
No errors to report.
Job Result Succeeded
Total Duration 7.3 hours
Submit Time 11/7/2016 9:45:27 AM
Start Time 11/7/2016 9:46:20 AM
End Time 11/7/2016 5:03:00 PM
Compilation 35 seconds
Queued 0 seconds
Running 7.3 hours
Account ntfindemoadla0630
Author analyticsdemo@outlook.com
Priority 1000
Parallelism 60
Bytes Left 759,092,887,081
Bytes Read 4,390,223,643,208
Bytes Written 7,340,221,434,900
Vertices 1,953
Job Detail
Script
Resources
Vertex Execution View
Job Graph
Metadata Operations
State History
Diagnostics
Display: Progress
Succeeded
Failed
Running
Waiting
Job Playback
00:00:00
Error List
Exception Settings
Output
Ready
Server Explorer
Toolbox
Cloud Explorer
Team Explorer
Class View
Properties
Resources

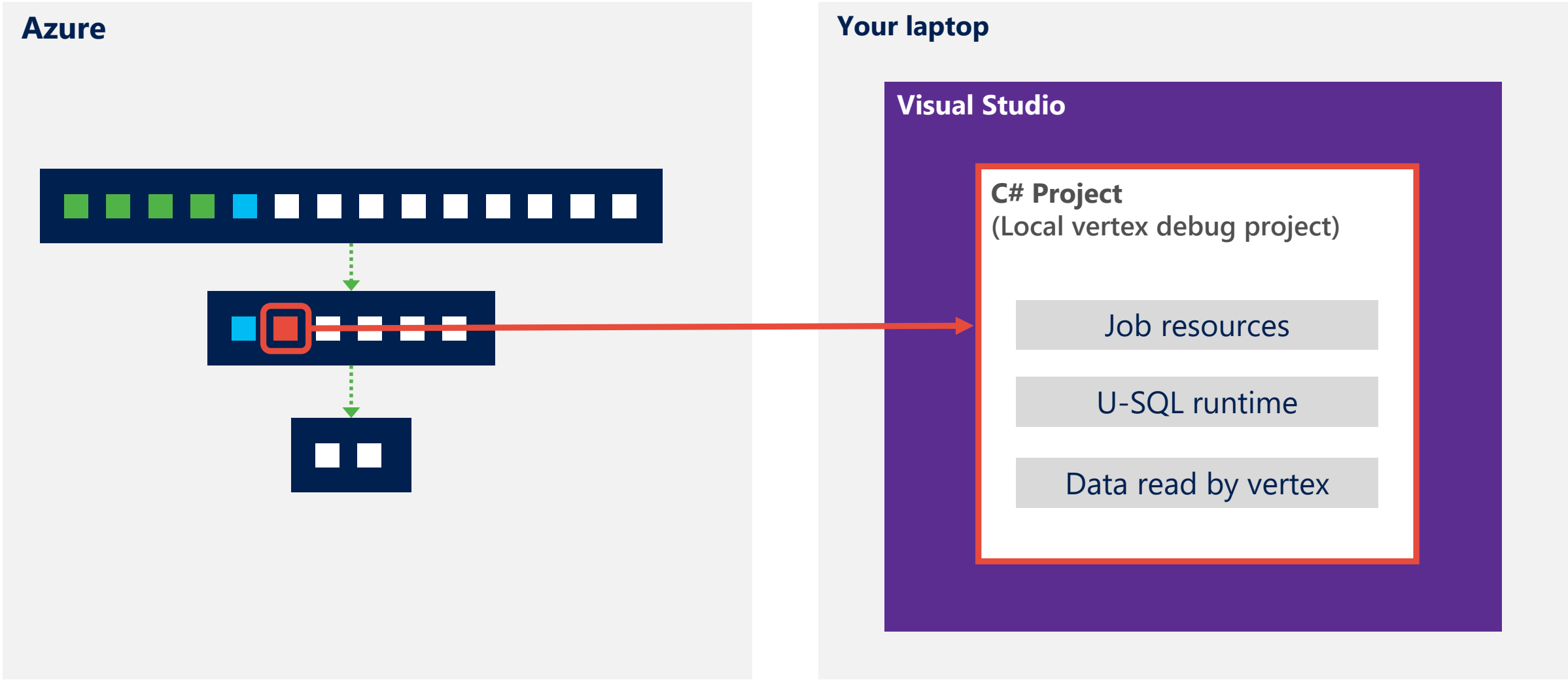

USQLSampleApplication1 - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Data Lake Tools Test Analyze Window Help
Analytics Demo
AD
Debug
Any CPU
Start
Server Explorer
Toolbox
LoadAndProcessData_Resources
Job Browser: ntfindemoadla0630
Ambulance-3-2-Crea...titionedTable.usql
Compile View: C:\...yDriversOnADL.usql
Job Resources
Download All
Name
File Size
Created
Download all the resources
Download a specific resource
Blue items: the output of the compiler
Grey items: U-SQL runtime bits
Algebra.xml N/A
DailyTradesAssembly.dll Download 4,608 bytes N/A
__ScopeCodeGen__.dll Download 17,408 bytes N/A
__ScopeCodeGen__.pdb Download 60,928 bytes N/A
__ScopeCodeGenEngine__.dll Download 2,990 KB N/A
__ScopeCodeGenEngine__.pdb Download 44,644 KB N/A
ScopeVertexDef.xml Download 115,834 bytes N/A
__SStreamInfo__.xml Download 847 bytes N/A
__ScopeCodeGen__.dll.cs Download 41,260 bytes N/A
__ScopeCodeGenEngine__.dll.cpp Download 1,753 KB N/A
__ScopeCodeGenCompileOutput__.txt Download 376,654 bytes N/A
__ScopeCodeGenCompileOptions__.txt Download 20,960 bytes N/A
__ScopeCodeGenEngine__.cppresources Download 483,347 bytes N/A
Tabular Model Explorer
Solution Explorer
Team Explorer
Class View

Local vertex debug

(coming soon)

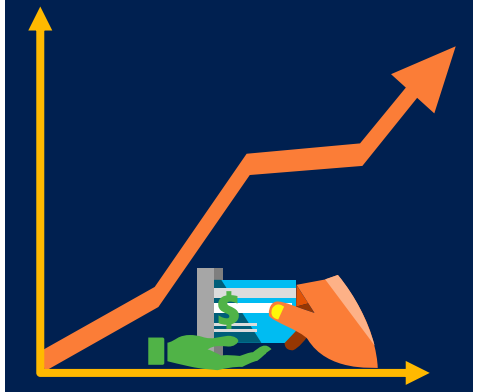
Microsoft

Debug a vertex = run the vertex on your machine



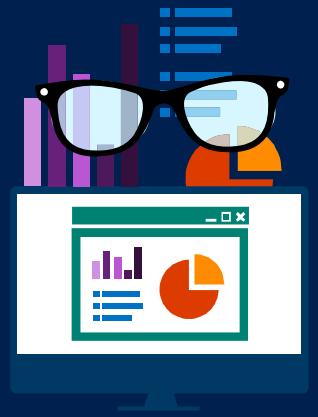
Azure Data Lake Analytics

Start in seconds
Scale instantly
Pay per job

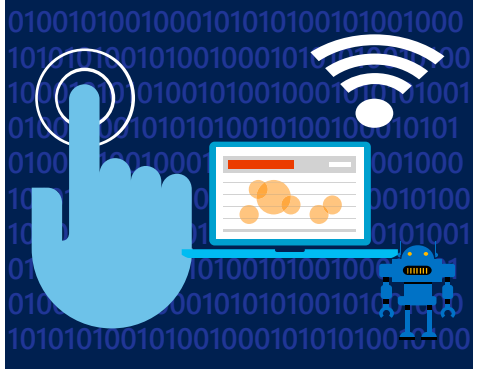


Develop massively parallel programs with simplicity

Debug and optimize your Big Data programs with ease



Virtualize your analytics



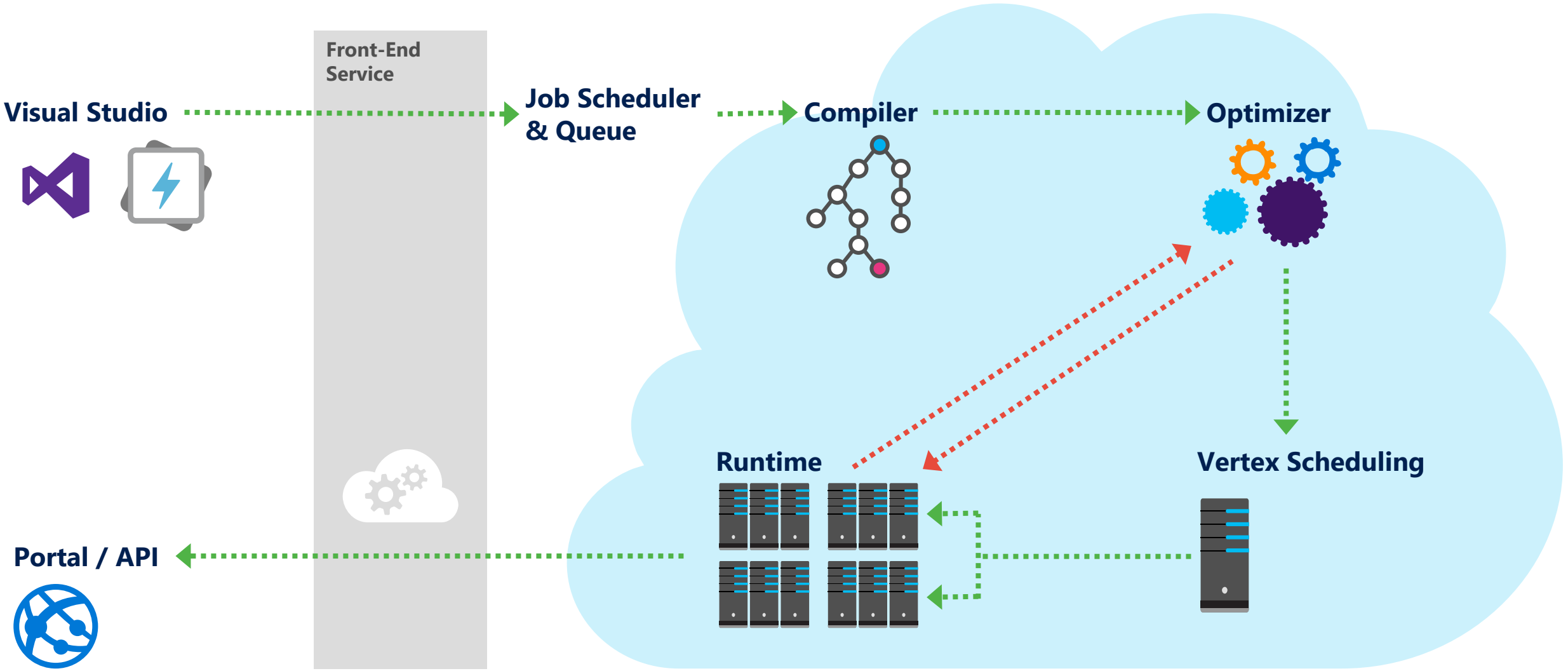
Enterprise-grade security, auditing and support

Query execution

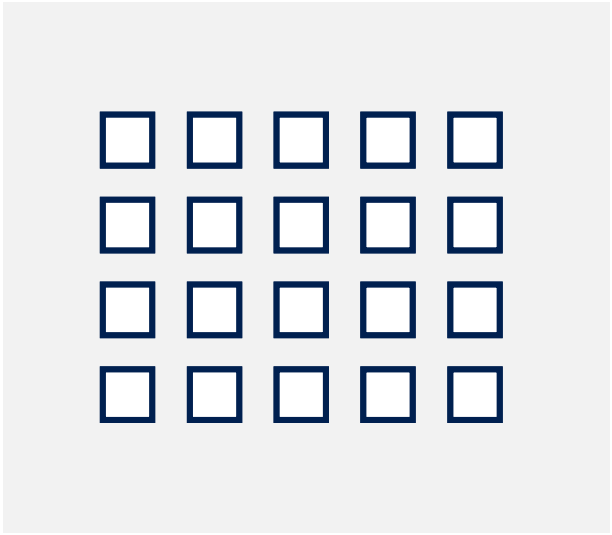
Plans, vertices, stages, parallelism, ADLAUs

Microsoft

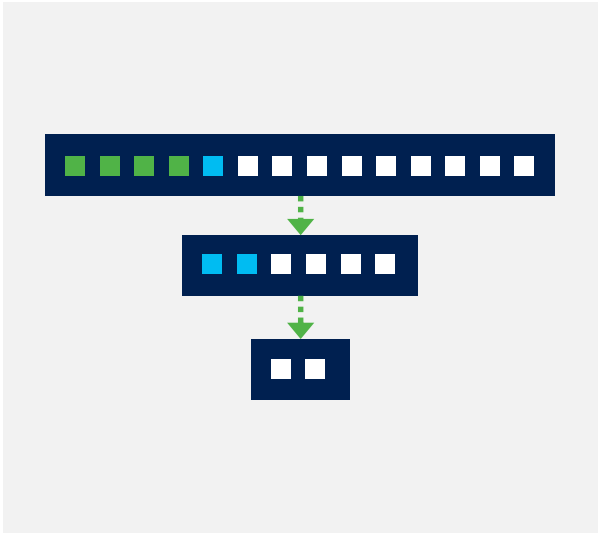
Query Life


Front-End Service
Visual Studio
Job Scheduler & Queue
Compiler
Optimizer
Runtime
Vertex Scheduling
Portal / API

Logical to physical plan

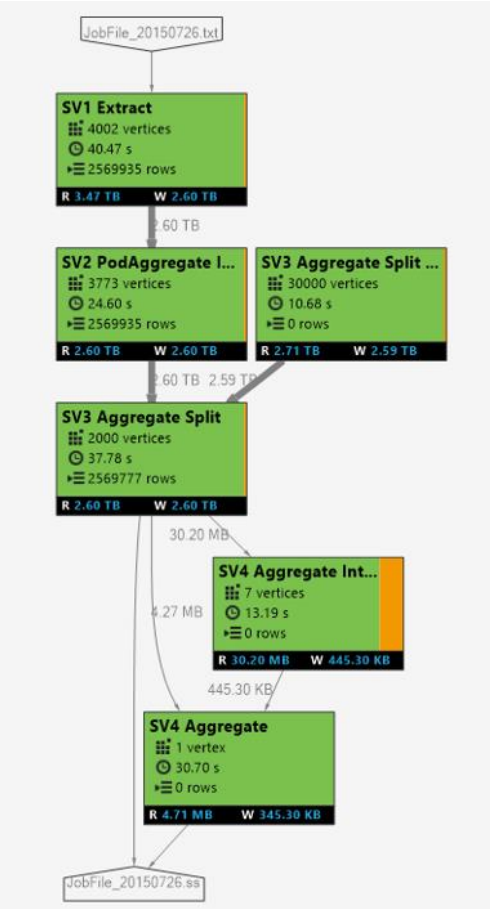


Each square = “a vertex” represents a fraction of the total



Vertices in each SuperVertex (aka “Stage) are doing the same operation on the same data.

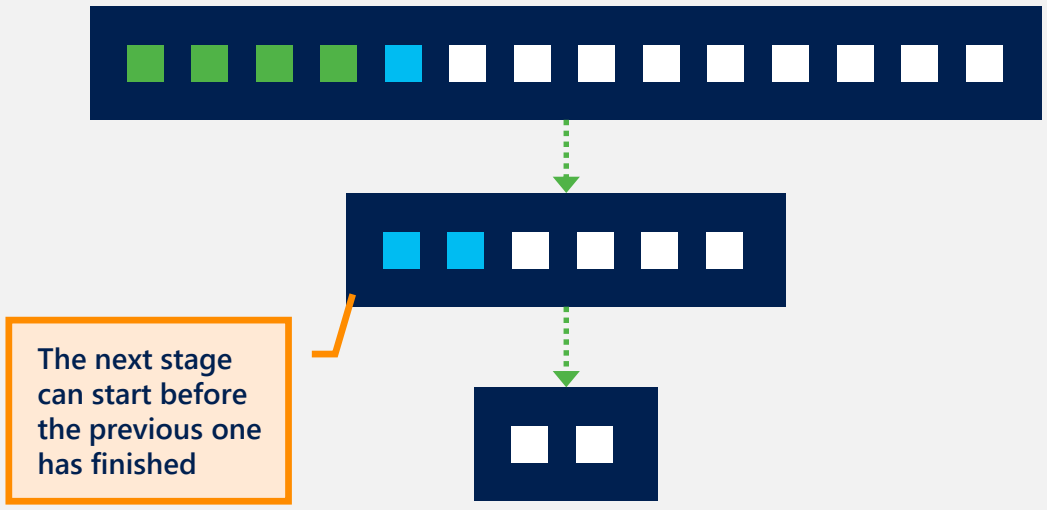
Vertices in a later stages may depend on a vertex in an earlier stage



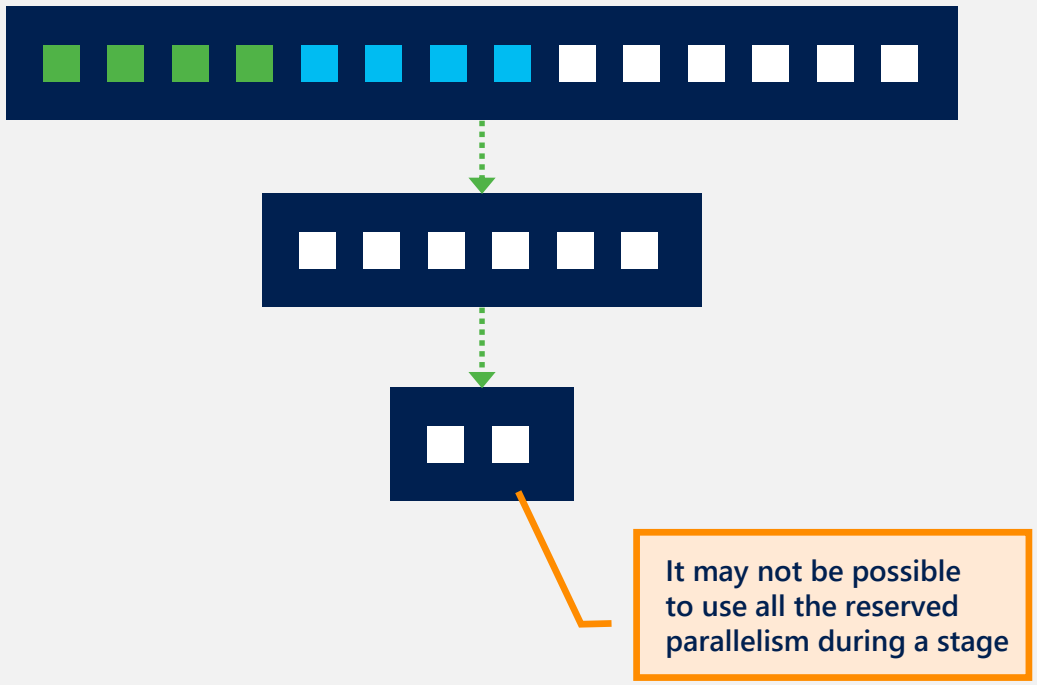
Visualized like this

Execution with requested parallelism

Requested parallelism = 1
(reserve enough to do 1 vertex at a time)

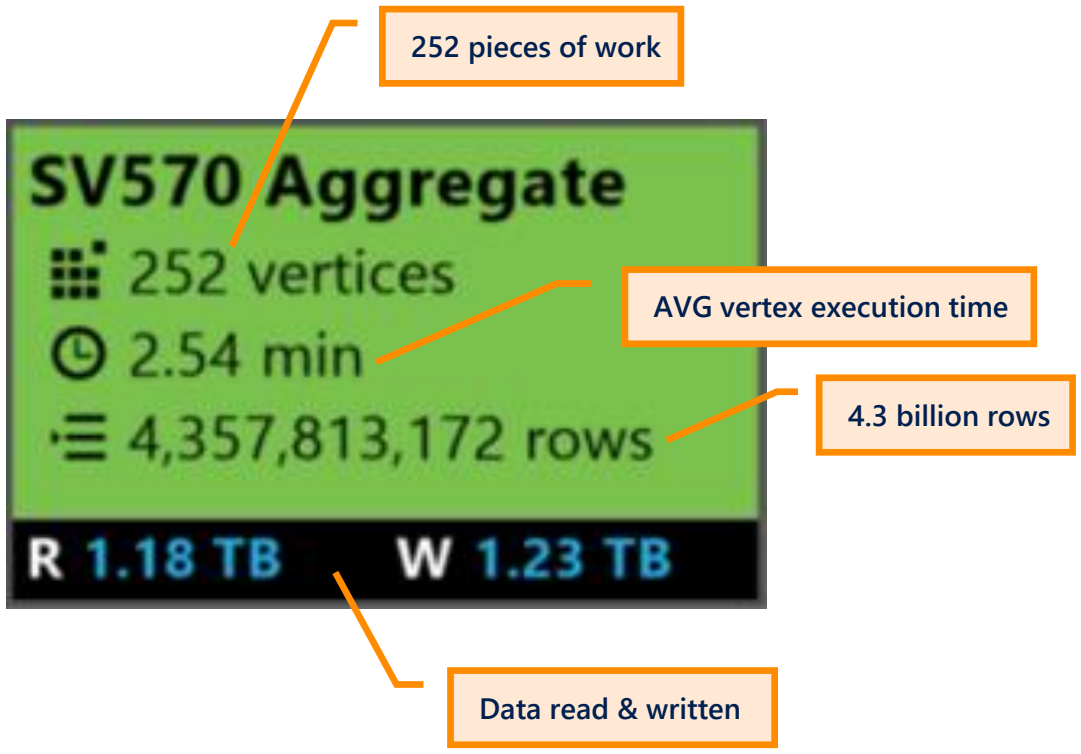


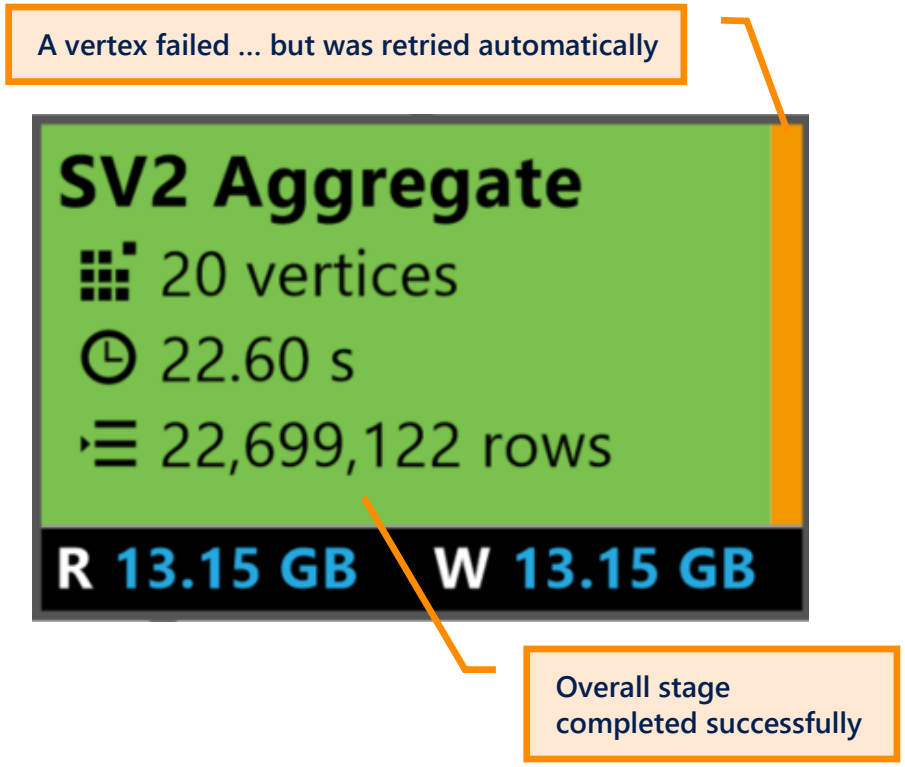
Requested parallelism = 4
(reserve enough to do 4 vertices at a time)



Job resources are copied to each vertex

Stage details



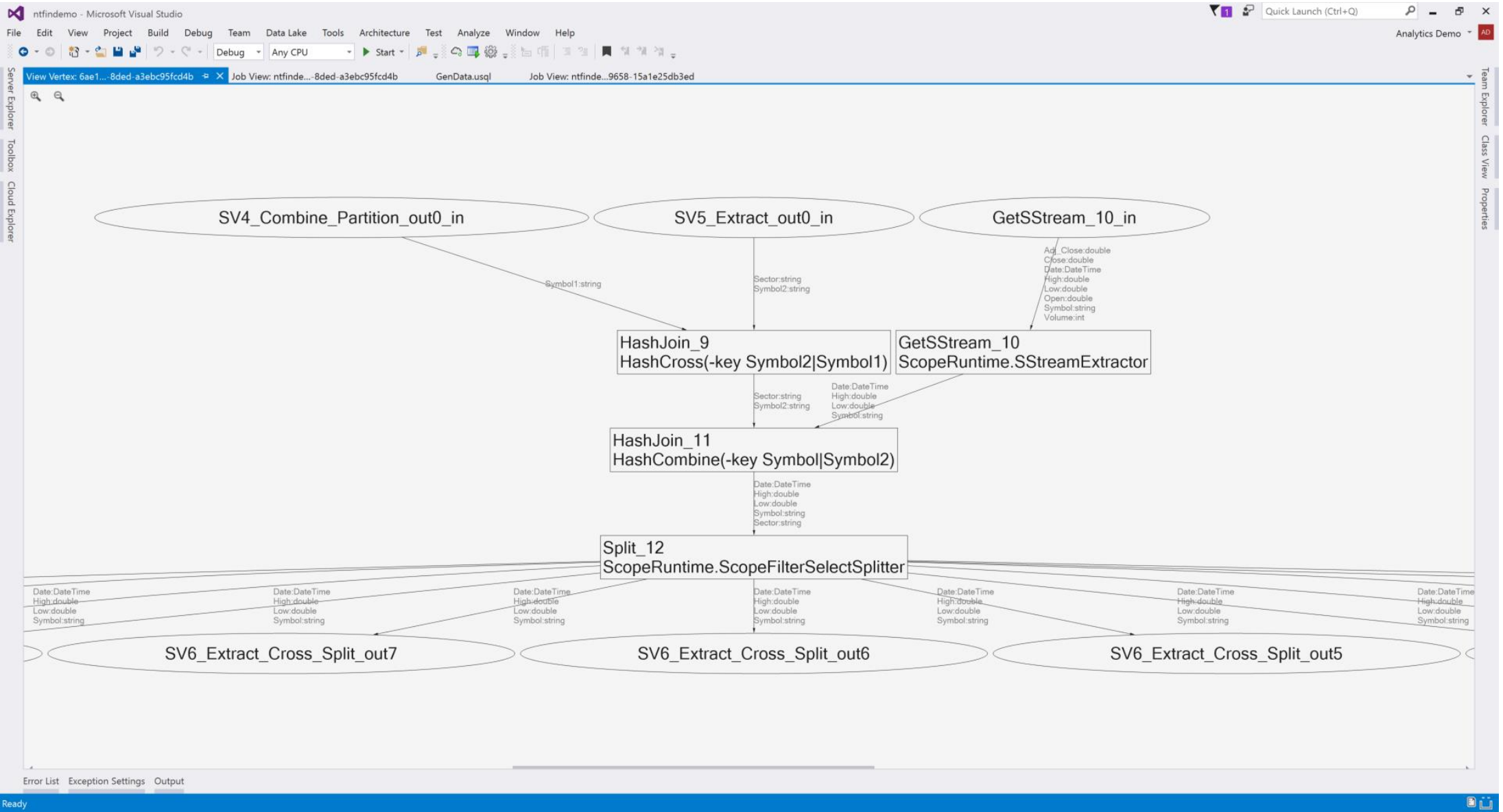


A vertex might fail because:

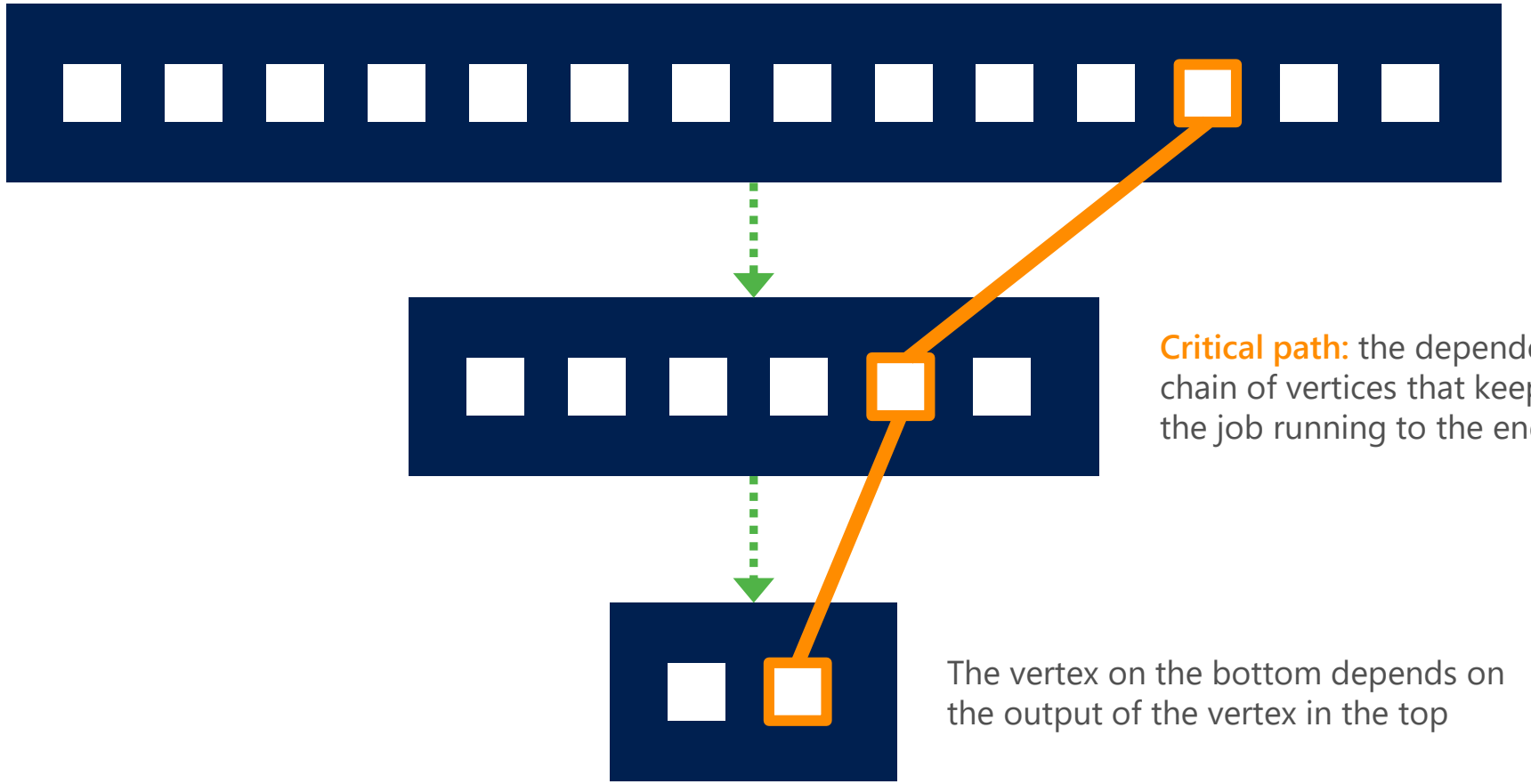
- Router congested
- Hardware failure (ex: hard drive failed)
- VM had to be rebooted

U-SQL job will automatically schedule a vertex on another VM.

Inside a Stage: The Operator Graph

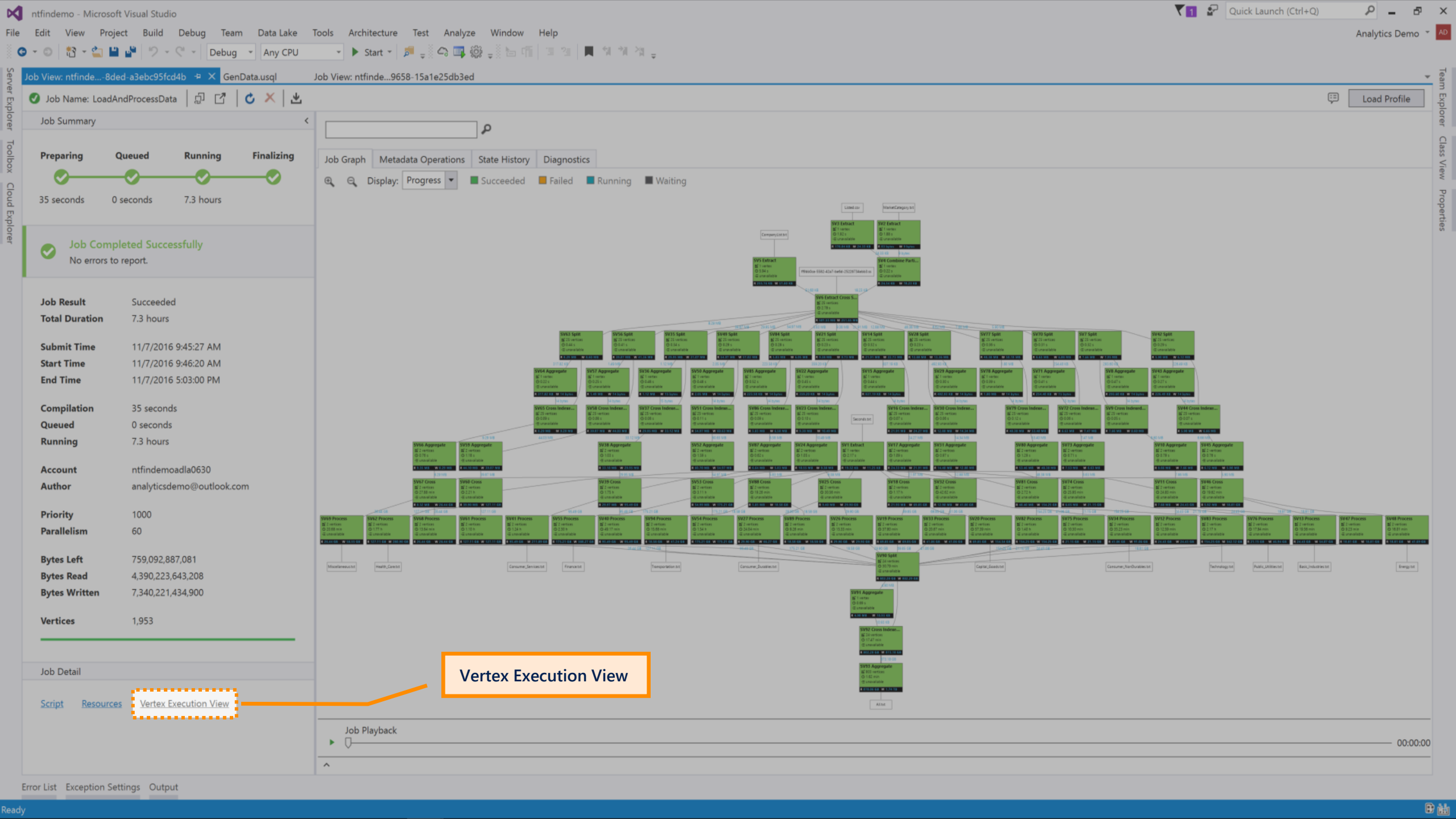


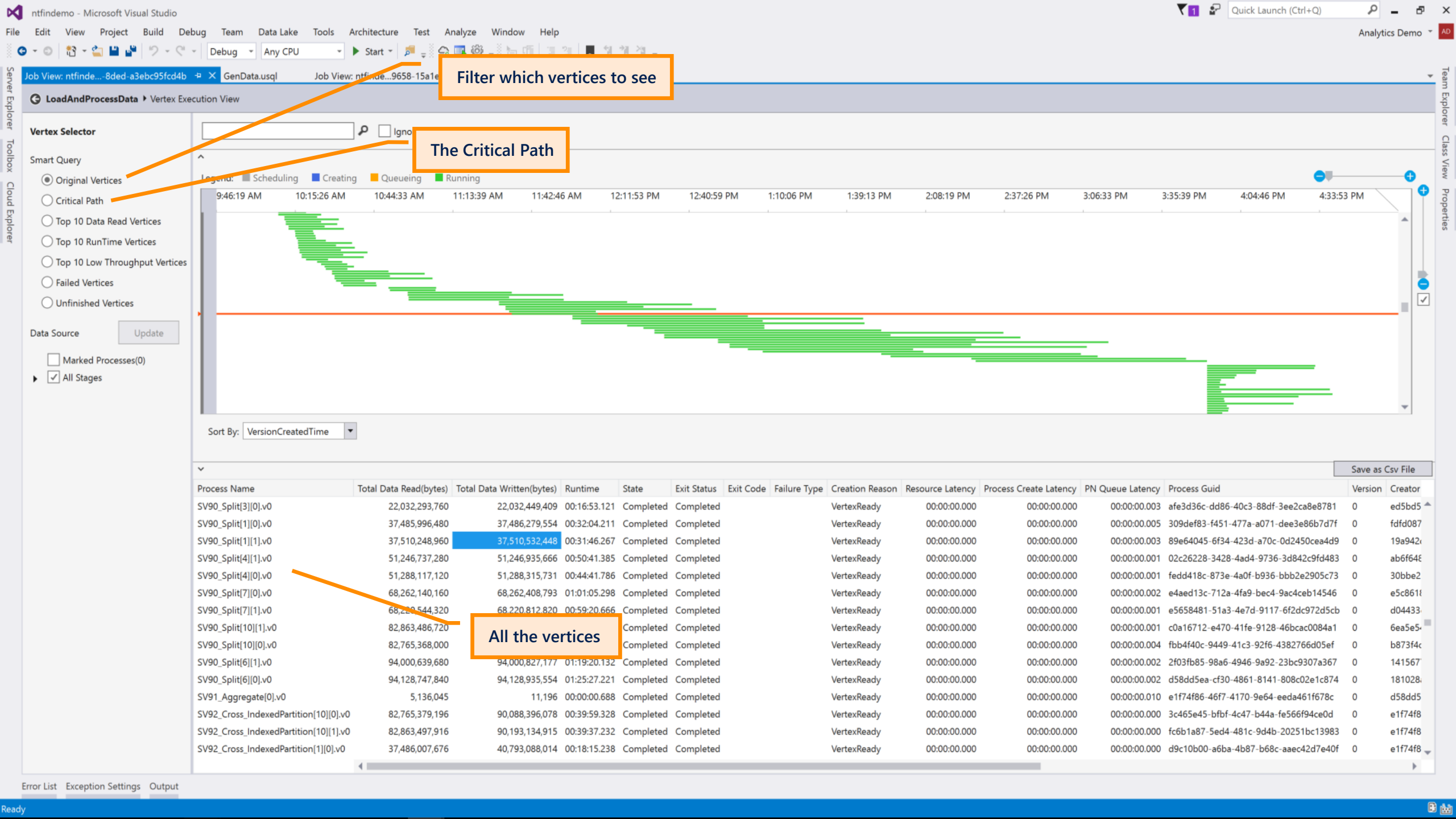
Vertex relationships



Critical path: the dependency chain of vertices that keeps the job running to the end

The vertex on the bottom depends on the output of the vertex in the top


ntfindemo - Microsoft Visual Studio
Job View: ntfinde...-8ded-a3ebc95fcd4b
GenData.usql
Job View: ntfinde...9658-15a1e25db3ed
Job Name: LoadAndProcessData
Load Profile
Job Summary
Preparing
Queued
Running
Finalizing
35 seconds
0 seconds
7.3 hours
Job Completed Successfully
No errors to report.
Job Result Succeeded
Total Duration 7.3 hours
Submit Time 11/7/2016 9:45:27 AM
Start Time 11/7/2016 9:46:20 AM
End Time 11/7/2016 5:03:00 PM
Compilation 35 seconds
Queued 0 seconds
Running 7.3 hours
Account ntfindemoadla0630
Author analyticsdemo@outlook.com
Priority 1000
Parallelism 60
Bytes Left 759,092,887,081
Bytes Read 4,390,223,643,208
Bytes Written 7,340,221,434,900
Vertices 1,953
Job Detail
Script
Resources
Vertex Execution View
Job Graph
Metadata Operations
State History
Diagnostics
Display: Progress
Succeeded
Failed
Running
Waiting
Job Playback
00:00:00
Vertex Execution View
Error List
Exception Settings
Output
Ready



Process Name	Total Data Read(bytes)	Total Data Written(bytes)	Runtime	State	Exit Status	Exit Code	Failure Type	Creation Reason	Resource Latency	Process Create Latency	PN Queue Latency	Process Guid	Version	Creator
SV90_Split[3][0].v0	22,032,293,760	22,032,449,409	00:16:53.121	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.003	afe3d36c-dd86-40c3-88df-3ee2ca8e8781	0	ed5bd5
SV90_Split[1][0].v0	37,485,996,480	37,486,279,554	00:32:04.211	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.005	309def83-f451-477a-a071-dee3e86b7d7f	0	fdfd087
SV90_Split[1][1].v0	37,510,248,960	37,510,532,448	00:31:46.267	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.003	89e64045-6f34-423d-a70c-0d2450cea4d9	0	19a942
SV90_Split[4][1].v0	51,246,737,280	51,246,935,666	00:50:41.385	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.001	02c26228-3428-4ad4-9736-3d842c9fd483	0	ab6f64
SV90_Split[4][0].v0	51,288,117,120	51,288,315,731	00:44:41.786	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.001	fedd418c-873e-4a0f-b936-bbb2e2905c73	0	30bbe2
SV90_Split[7][0].v0	68,262,140,160	68,262,408,793	01:01:05.298	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.002	e4aed13c-712a-4fa9-bec4-9ac4ceb14546	0	e5c861
SV90_Split[7][1].v0	68,220,544,320	68,220,812,820	00:59:20.666	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.001	e5658481-51a3-4e7d-9117-6f2dc972d5cb	0	d04433
SV90_Split[10][1].v0	82,863,486,720			Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.001	c0a16712-e470-41fe-9128-46bcac0084a1	0	6ea5e5
SV90_Split[10][0].v0	82,765,368,000			Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.004	fbb4f40c-9449-41c3-92f6-4382766d05ef	0	b873f4
SV90_Split[6][1].v0	94,000,639,680	94,000,827,177	01:19:20.132	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.002	2f03fb85-98a6-4946-9a92-23bc9307a367	0	141567
SV90_Split[6][0].v0	94,128,747,840	94,128,935,554	01:25:27.221	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.002	d58dd5ea-cf30-4861-8141-808c02e1c874	0	181028
SV91_Aggregate[0].v0	5,136,045	11,196	00:00:00.688	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.010	e1f74f86-46f7-4170-9e64-eeda461f678c	0	d58dd5
SV92_Cross_IndexedPartition[10][0].v0	82,765,379,196	90,088,396,078	00:39:59.328	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.000	3c465e45-bfbf-4c47-b44a-fe566f94ce0d	0	e1f74f8
SV92_Cross_IndexedPartition[10][1].v0	82,863,497,916	90,193,134,915	00:39:37.232	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.000	fc6b1a87-5ed4-481c-9d4b-20251bc13983	0	e1f74f8
SV92_Cross_IndexedPartition[1][0].v0	37,486,007,676	40,793,088,014	00:18:15.238	Completed	Completed			VertexReady	00:00:00.000	00:00:00.000	00:00:00.000	d9c10b00-a6ba-4b87-b68c-aaec42d7e40f	0	e1f74f8

Efficiency
Cost vs. latency

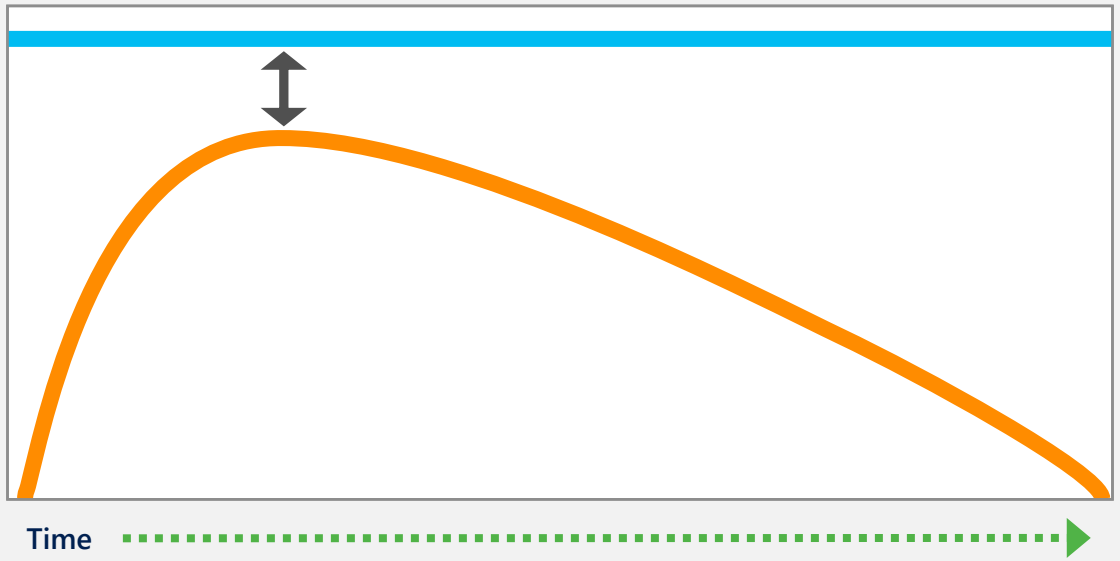
Microsoft

ADLAU allocation

Example: allocating 10 ADLAUs for a 10 minute job
Cost: 10 min * 10 ADLAUs = 100 ADLAU minutes

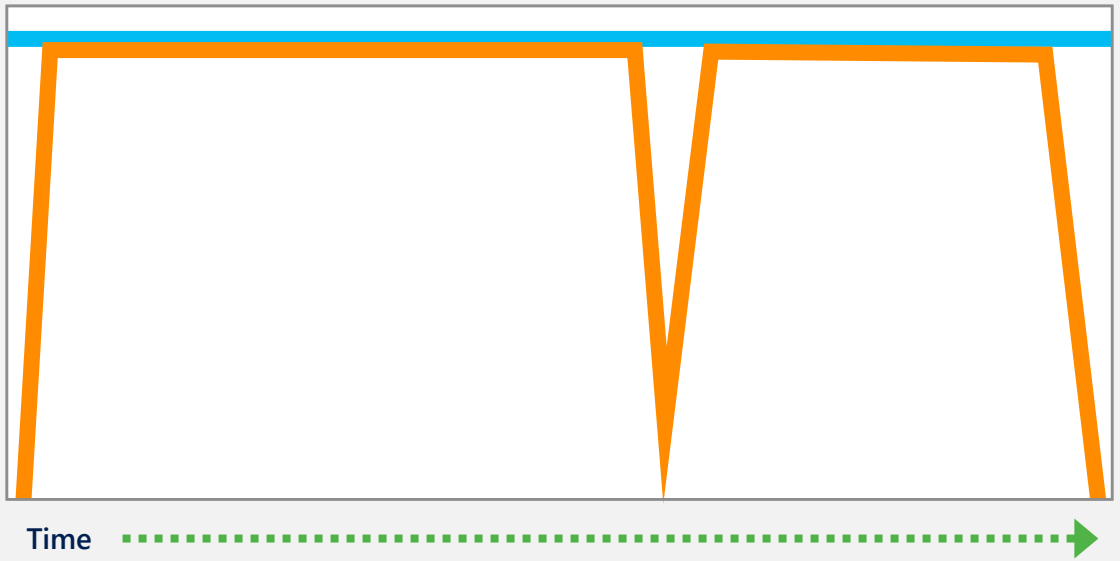
Blue line: allocated
Red line: running

Over-allocation



- Consider using fewer ADLAUs
- You are paying for the area under the blue line
- You are only using the area under the red line

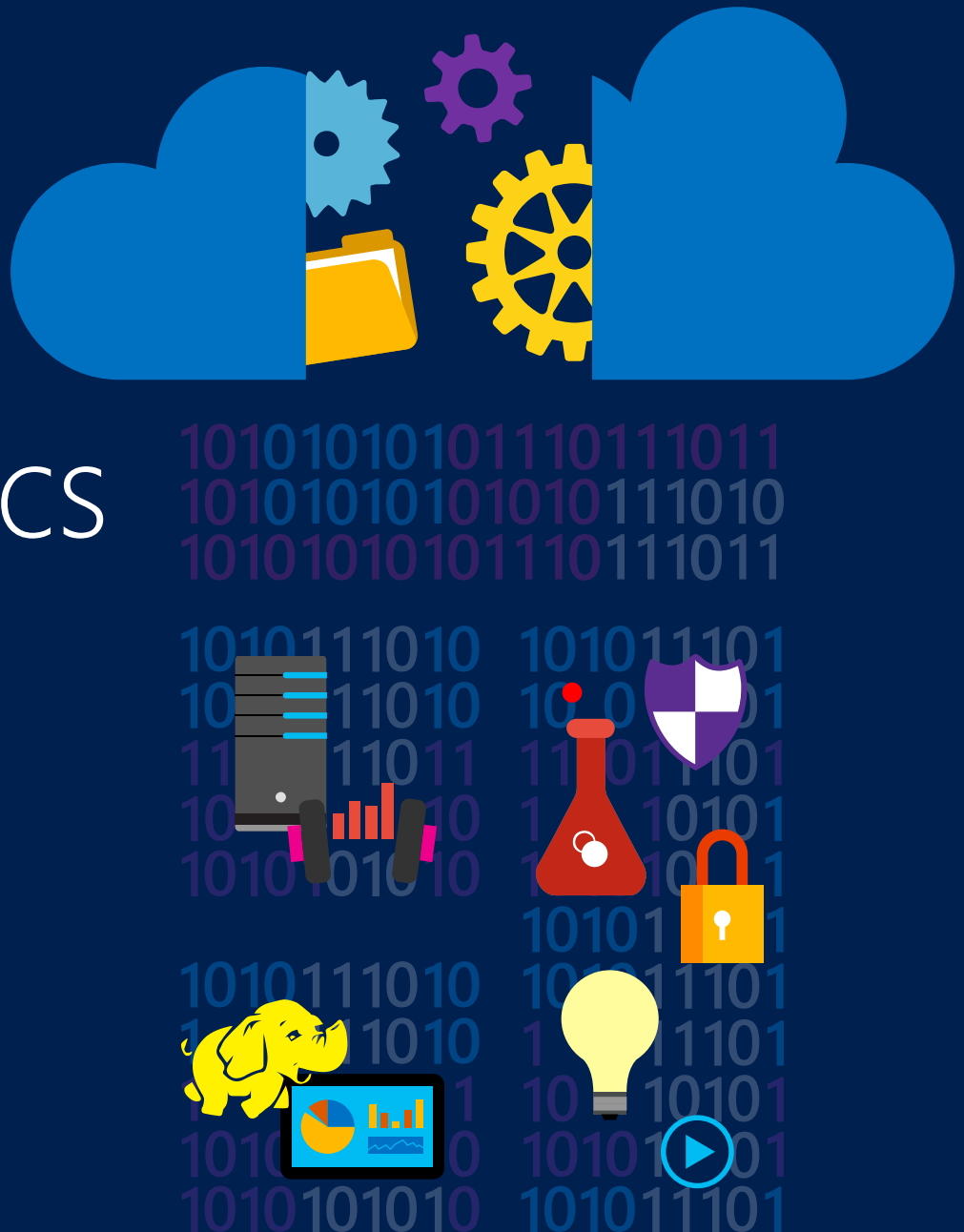
Under-allocation



- Consider using more ADLAUs

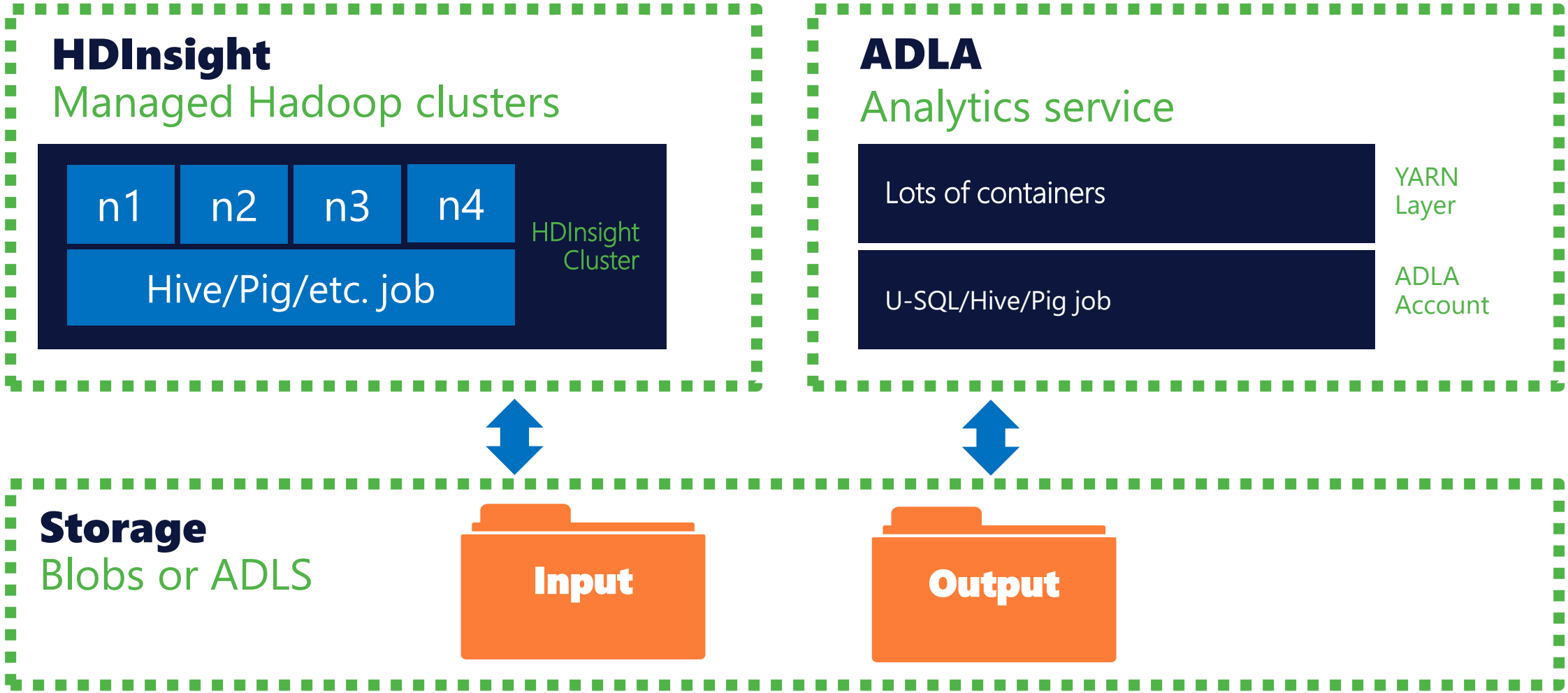
Azure Data Lake Analytics

Compare with HDInsight



Microsoft

Analytics: Two form factors



ADLA complements HDInsight

Target the same scenarios, tools, and customers

HDInsight

- For developers familiar with the Open Source: Java, Eclipse, Hive, etc.
- Clusters offer customization, control, and flexibility in a managed Hadoop cluster

ADLA

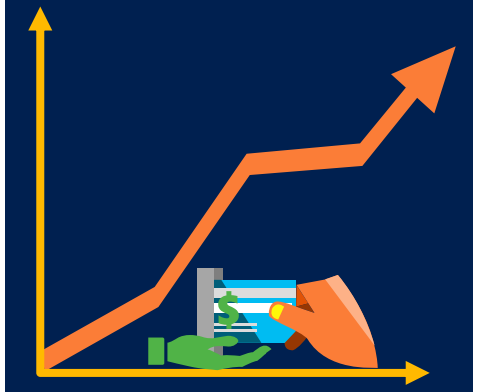
- Enables customers to leverage existing experience with C#, SQL & PowerShell
- Offers convenience, efficiency, automatic scale, and management in a “job service” form factor

Choose the right service

Decision factor	Example
Capability	Hadoop, Spark vs U-SQL
Control	G-Series VMs Customization scripts
Ecosystem	DataMeer, AtScale, Cask etc.
Form Factor	Job Service Vs Clusters

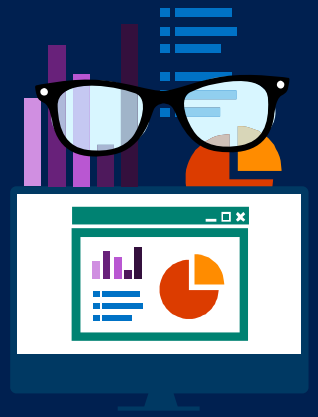
Azure Data Lake Analytics

Start in seconds
Scale instantly
Pay per job

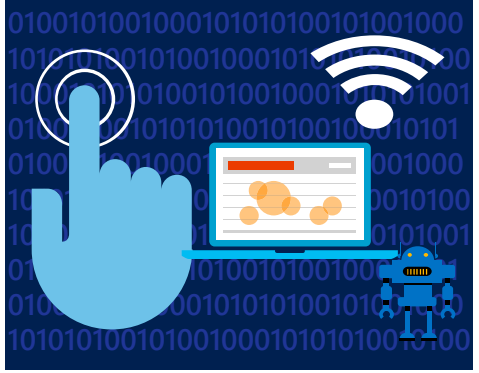


Develop massively parallel programs with simplicity

Debug and optimize your Big Data programs with ease



Virtualize your analytics

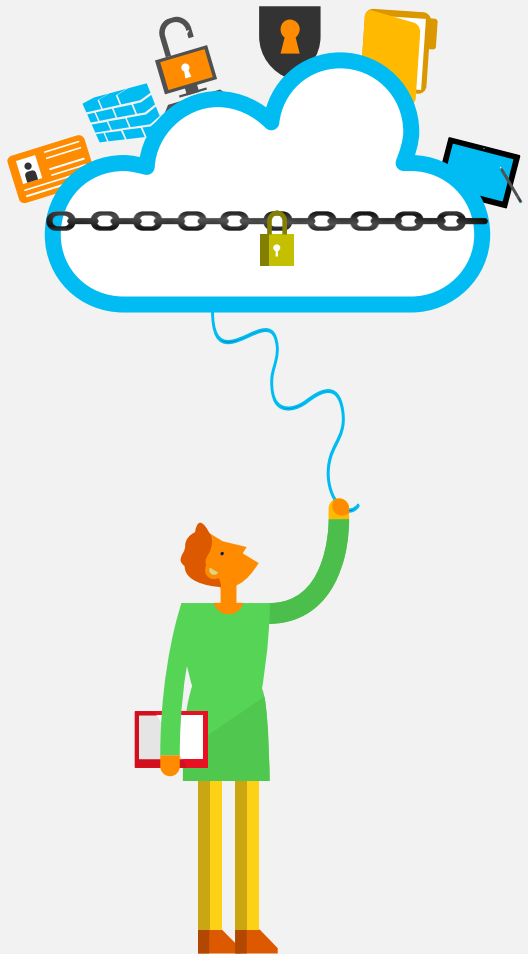


Enterprise-grade security, auditing and support

Microsoft

ADL Store: enterprise-grade security

- Enterprise-grade security permits even sensitive data to be stored securely
- Regulatory compliance can be enforced
- Integrates with Azure Active Directory for authentication
- Data is encrypted at rest
- POSIX-style permissions on files and folders
- Audit logs for all operations

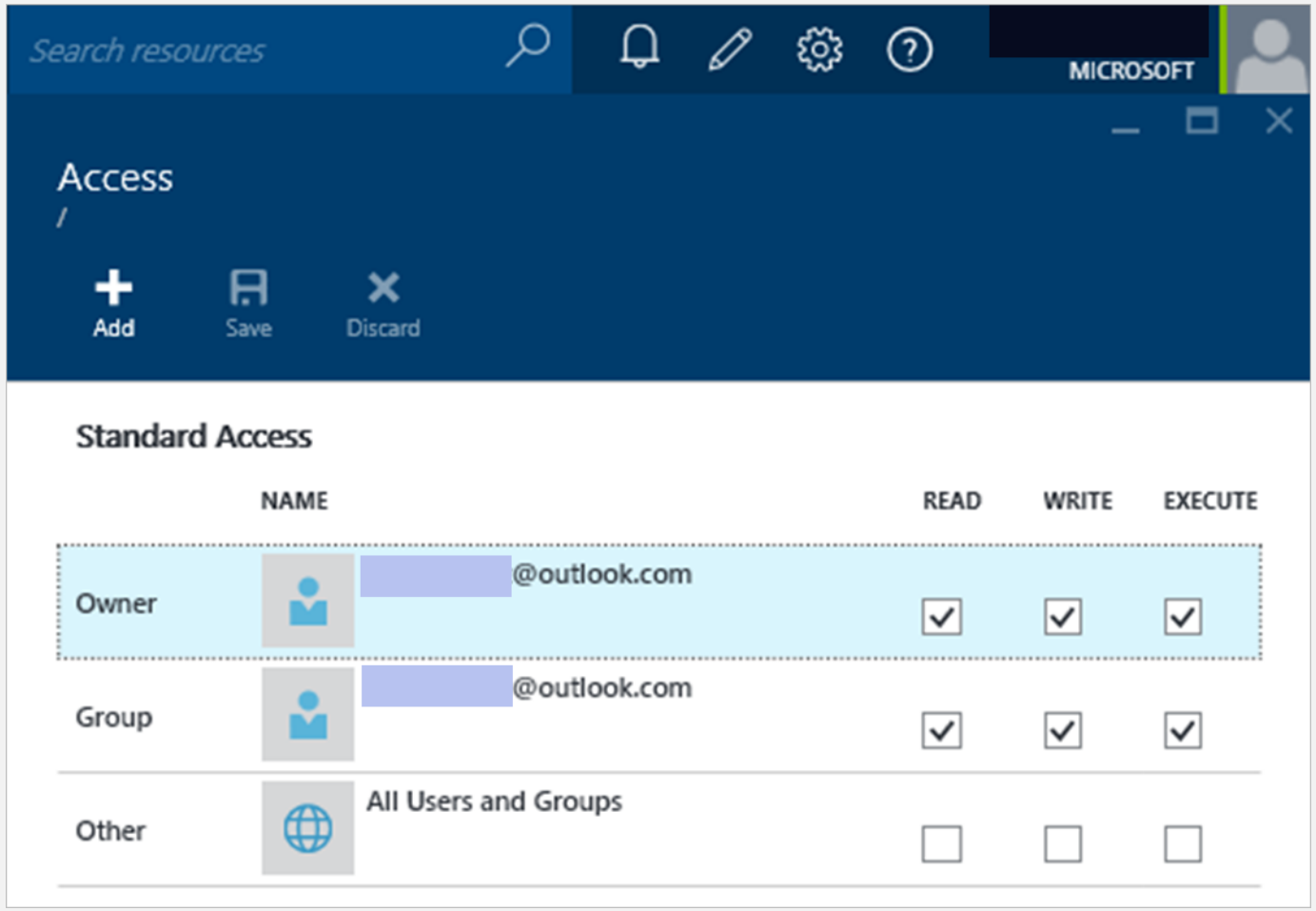


ADL Store Security: AAD integration

- Multi-factor authentication based on OAuth2.0
- Integration with on-premises AD for federated authentication
- Self-service password management
- Role-based access control
- Privileged account management
- Device registration
- Application usage monitoring and rich auditing
- Security monitoring and alerting
- Fine-grained ACLs for AD identities

ADL Store security: Role-based access

- Each file and directory is associated with an owner and a group
- Files or directories have separate permissions (read(r), write(w), execute(x)) for owners, members of the group, and for all other users
- Fine-grained access control lists (ACLs) rules can be specified for specific named users or named groups



ADL Store Security: encryption-at-rest

- Transparently encrypts data flowing to and from public networks as well as at rest
- Transparent server-side encryption
- User can manage their own encryption keys or let Azure Data Lake Store manage the key using Azure Key Vault

Get started today!

For more information visit:
http://azure.com/datalake